FEDERAL UNIVERSITY OF AMAZONAS - UFAM

INSTITUTE OF COMPUTING - ICOMP

POSTGRADUATE PROGRAM IN INFORMATICS - PPGI

A Catalog of Micro Frontends Anti-patterns

Nabson Paiva Souza da Silva

Manaus - AM

2025

Nabson Paiva Souza da Silva

A Catalog of Micro Frontends Anti-patterns

Master's Thesis submitted for evaluation as a partial requirement for obtaining the title of Master of Science in Computer Science from the Graduate Program in Computer Science, Institute of Computing.

Supervisor

Prof. Dr. Tayana Uchôa Conte

Federal University of Amazonas - UFAM

Institute of Computing - IComp

Manaus - AM

2025

Ficha Catalográfica

Elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

S586c Silva, Nabson Paiva Souza da
A Catalog of Micro Frontends Anti-patterns / Nabson Paiva Souza da Silva. - 2025.
301 f. : il., color. ; 31 cm.

Orientador(a): Tayana Uchôa Conte.
Dissertação (mestrado) - Universidade Federal do Amazonas, Programa de Pós-Graduação em Informática, Manaus, 2025.

1. Micro frontends. 2. Anti-patterns. 3. Software Architecture. 4. Microservices. I. Conte, Tayana Uchôa. II. Universidade Federal do Amazonas. Programa de Pós-Graduação em Informática. III. Título

---

Ministério da Educação
Universidade Federal do Amazonas
Coordenação do Programa de Pós-Graduação em Informática

FOLHA DE APROVAÇÃO

"A CATALOG OF MICRO FRONTENDS ANTI-PATTERNS"

NABSON PAIVA SOUZA DA SILVA

Dissertação de Mestrado defendida e aprovada pela banca examinadora constituída pelos Professores:

Profa. Dra. Tayana Uchôa Conte - PRESIDENTE

Prof. Dr. Márcio de Medeiros Ribeiro - MEMBRO EXTERNO

Prof. Dr. Igor Fábio Steinmacher - MEMBRO EXTERNO

MANAUS, 25 de junho de 2025.

Documento assinado eletronicamente por **Márcio de Medeiros Ribeiro**, **Usuário Externo**, em 30/06/2025, às 10:56, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Igor Fabio Steinmacher**, **Usuário Externo**, em 30/06/2025, às 15:57, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Tayana Uchoa Conte**, **Professor do Magistério Superior**, em 02/07/2025, às 12:03, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Avenida General Rodrigo Octávio, 6200 - Bairro Coroado I Campus Universitário Senador Arthur Virgílio Filho, Setor Norte - Telefone: (92) 3305-1181 / Ramal 1193
CEP 69080-900, Manaus/AM, coordenadorppgi@icomp.ufam.edu.br

I dedicate this work to my parents, whose strength, dedication, and constant support have taught me the true meaning of perseverance and commitment.

Acknowledgements

I thank my family for their unwavering support and encouragement. Christian Silva, thank you for always standing by my side and being the perfect husband. Francinilda Silva, thank you for always guiding me along the right path and being such a loving mother. Rubem Silva, thank you for your patience in raising me and for being a so protective and worrying father. Natália Ferreira, thank you for being the older sister who always believed in me and never let me give up. Raykennedy Ferreira, thank you for being the brother who was always available to help. Anna Letícia Ferreira and Mannuela Ferreira, thank you for being the best nieces ever and always brightening my day. Ana Cila Souza and Ana Carolina Souza, thank you for embracing me as part of your family. Nadir Paiva (in memoriam), thank you for all the love you gave me during my childhood.

I thank all the friends I have made throughout my entire academic journey. My undergraduate fellows: Marcos Lopes, Fernando Nogueira, Bianca Dias, and specially João Lucas Fernandes, who helped me during the web application implementation. To my companions in the Usability and Software Engineering (USES) Research Group, especially Eriky Rodrigues, Ayumi Santana, Reine Santos, João Bernardo, Lennon Chaves, Lucas Araújo, and Marcia Lima. To the friends I have made in life: Alenka Rocha, Nailson Filho, and Gustavo Gomes. To the researchers who collaborated in some of the studies presented in this Thesis: Eriky Rodrigues and Matheus de Oliveira.

I am deeply grateful to my supervisor, Prof. Dr. Tayana Conte, for her invaluable guidance and support throughout my academic journey. Your dedication and care in teaching me how to be a researcher, professor, and advisor have inspired me to always

strive for excellence. As I always say, thank you for your unwavering belief in me.

Finally, I thank Prof. Dr. Igor Steinmacher and Prof. Dr. Márcio Ribeiro for accepting the invitation to serve on this Master's Thesis Committee. I also sincerely thank Prof. Dr. Bruno Gadelha and Prof. Dr. Sheila Reinehr for agreeing to serve as alternate members. Your contributions to Software Engineering are genuinely inspiring, and it is an honor to have you as part of this important milestone in my academic journey.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. This work was partially supported by Amazonas State Research Support Foundation - FAPEAM - through the POSGRAD project.

The best preparation for tomorrow is doing your best today.

H. Jackson Brown, Jr.

A Catalog of Micro Frontends Anti-patterns

Author: Nabson Paiva Souza da Silva

Supervisor: Prof. Dr. Tayana Uchôa Conte

Abstract

Micro frontend (MFE) is an architectural style derived from Microservices (MS) that decomposes a monolithic frontend application into smaller, manageable, and independently deployable slices. Despite its increasing adoption, the field remains relatively underexplored, particularly in terms of identifying challenges and documenting best practices. Therefore, the goal of this Master's Thesis is to propose and evaluate an artifact that supports developers in implementing MFE architectures. We introduce a catalog of MFE anti-patterns that document common problems and practical solutions. The initial version of the catalog was developed based on established MS anti-patterns and real-world issues. To verify the prevalence of these anti-patterns in MFE architectures and assess whether the proposed solutions effectively address them, we conducted a Personal Opinion Survey with industry practitioners. Additionally, we developed a web application designed to showcase the anti-patterns and foster collaboration within the MFE community. Furthermore, we ran a controlled experiment to compare the catalog with practitioner-provided examples and guidelines in solving MFE maintenance challenges, assessing how students used the catalog and whether it enhanced their perceived learning. Finally, we performed a Multivocal Literature Review to expand the catalog by adding anti-patterns proposed in grey literature sources. After each study, we refined the catalog to produce a final version that helps developers identify, solve, and prevent problems when working with MFE architectures. The contributions of this Thesis include centralized documentation of common issues and solutions when

developing MFE architectures, empirical evidence on how the catalog can be used, a web application that showcases the anti-patterns and promotes collaboration within industry practitioners, and the development of MFE teaching material that instructors can integrate into software architecture curricula. We believe that the results of this work have the potential to drive significant advances in both the practice and theory of MFE, helping shape future research and improve industry adoption. As part of future work, we aim to evaluate its utility in supporting practitioners as they evolve real-world MFE architectures and explore automated anti-pattern detection tools.

Keywords: Micro frontends, Anti-patterns, Software Architecture, Microservices.

A Catalog of Micro Frontends Anti-patterns

Author: Nabson Paiva Souza da Silva

Supervisor: Prof. Dr. Tayana Uchôa Conte

Resumo

Micro frontends (MFE) são um estilo arquitetural derivado de microsserviços (MS) que decompõe uma aplicação frontend monolítica em partes menores, gerenciáveis e implantáveis de forma independente. Apesar da adoção crescente, o campo ainda é relativamente pouco explorado, especialmente no que diz respeito à identificação de desafios e à documentação de boas práticas. Portanto, o objetivo desta dissertação de mestrado é propor e avaliar um artefato que apoie desenvolvedores na implementação de arquiteturas MFE. Apresentamos um catálogo de anti-padrões de MFE que documenta problemas comuns e soluções práticas. A versão inicial do catálogo foi desenvolvida com base em anti-padrões de MS consolidados e problemas observados na prática. Para verificar a prevalência desses anti-padrões em arquiteturas MFE e avaliar se as soluções propostas os resolvem efetivamente, conduzimos um survey com profissionais da indústria. Além disso, desenvolvemos uma aplicação web projetada para apresentar os anti-padrões e fomentar a colaboração na comunidade de MFE. Também realizamos um experimento controlado para comparar o catálogo com exemplos e diretrizes fornecidos por profissionais, avaliando como os estudantes utilizaram o catálogo e se ele contribuiu para sua percepção de aprendizado. Por fim, realizamos uma Revisão Multivocal da Literatura para expandir o catálogo, incorporando anti-padrões propostos em fontes de literatura cinzenta. Após cada estudo, refinamos o catálogo para produzir uma versão final que auxilia desenvolvedores a identificar, resolver e prevenir problemas ao trabalhar com arquiteturas MFE. As contribuições desta

dissertação incluem a documentação centralizada de problemas e soluções comuns no desenvolvimento de MFE, evidências empíricas sobre o uso do catálogo, uma aplicação web que apresenta os anti-padrões e promove a colaboração entre profissionais da indústria, e o desenvolvimento de materiais didáticos sobre MFE que podem ser integrados a disciplinas de arquitetura de software. Acreditamos que os resultados deste trabalho têm potencial para gerar avanços significativos tanto na prática quanto na teoria de MFE, contribuindo para pesquisas futuras e para a adoção da arquitetura na indústria. Como trabalho futuro, pretendemos avaliar a utilidade do catálogo no apoio a profissionais na evolução de arquiteturas MFE reais e explorar ferramentas de detecção automática de anti-padrões.

Palavras-chave: Micro frontends, Anti-padrões, Arquitetura de Software, Microsserviços.

List of Figures

Figure 1 – Diagram illustrating the research methodology based on Mafra, Barcelos & Travassos (2006). Blue rounded items show what we made after each step. 33

Figure 2 – Diagram exemplifying the composition of a fragment in a screen in each composition approach. 39

Figure 3 – The three forms of communication in MFEs: (1) Parent to Fragment: communication initiated by the parent (screen) towards a specific MFE fragment; (2) Fragment to Parent: communication initiated by a fragment back to the screen; and (3) Fragment to Fragment: communication directly between different fragments. 40

Figure 4 – Number of papers returned on the ACM Digital Library. 48

Figure 5 – Number of papers returned on the IEEE Xplore Library. 49

Figure 6 – Number of papers returned on the Google Scholar. 49

Figure 7 – Advanced search in Google Scholar. 50

Figure 8 – Process followed to propose and refine the MFE anti-patterns. 63

Figure 9 – Pie chart illustrating responses to the question: "Are you currently working with Micro Frontends?". 68

Figure 10 – Pie chart illustrating responses to the question: "How much professional experience do you have with Micro Frontends?". 68

Figure 11 – Bar chart presenting the responses to the question: "What type of role do you play or have you played when working with Micro Frontends?". 69

Figure 12 – Boxplot illustrating the harmfulness ratings for each anti-pattern. . . 71

Figure 13 – Resulting themes and meta-themes from the Thematic Analysis on practitioners feedback. 72
Figure 14 – Cyclic communication between fragments on the same screen. 79
Figure 15 – Communication between fragments using an Event Store. 79
Figure 16 – MFE-A is a central point (Hub) of dependency between the other MFEs. 81
Figure 17 – Home screen is a screen with several fragments, and when Fragment B raises an error, the entire screen becomes unavailable. 82
Figure 18 – When Fragment B raises an error, an user-friendly fallback message is rendered so the entire screen remains available. 83
Figure 19 – Home screen of the catalog's web application. 89
Figure 20 – Search results screen of the catalog's web application. 90
Figure 21 – Example of the anti-patterns details screen of the catalog's web application. 90
Figure 22 – About screen of the catalog's website. 91
Figure 23 – Timeline of the experiment execution. 97
Figure 24 – Timeline of the experiment execution. 100
Figure 25 – Boxplots presenting the data distribution on the assessment samples. 105
Figure 26 – Boxplots presenting the data distribution on the perceived learning before and after engaging with the MFE anti-patterns catalog. 107
Figure 27 – Responses for each sentence in the Perceived Usefulness construct. . 108
Figure 28 – Responses for each sentence in the Perceived Ease Of Use construct. 109
Figure 29 – Responses for each sentence in the Behavioral Intention construct. . . 109
Figure 30 – Responses for each sentence related to the catalog's utility for learning. 110
Figure 31 – To allow physical products payment, optional physical product-specific attributes are added and digital product-specific attributes become optional. 116
Figure 32 – General product attributes are non-optional and create optional generic attributes that can be used by any type of product. 117
Figure 33 – Page from (ANTUNES et al., 2024) with components as MFE, configuring nano frontends. 118

Figure 34 – Mega frontend break into two MFEs. 119
Figure 35 – CI and CD cycle. Source: <https://www.abtasty.com/resources/ci-cd/> 120
Figure 36 – Micro Frontends and Microservices grouped in cross-functional teams defined by domain. 121
Figure 37 – Mental map summarizing all references of the Dependent Deploy Anti-pattern. 131
Figure 38 – Search and selection process. 132
Figure 39 – Scatter plot showing the distribution of publications by type and year. 134
Figure 40 – Final set of MFE anti-patterns from the MLR. New anti-patterns are marked with a plus symbol (+) and updated ones with an asterisk (*). 135
Figure 41 – Intersections between publications considering the final anti-patterns. 136
Figure 42 – Pie chart showing the distribution of author profiles. 139
Figure 43 – Anti-pattern details page on the web application presenting the Hub-like Dependency in the new template. 152
Figure 44 – Page with a presentation explaining what micro frontends are. 153
Figure 45 – MFEs with non centralized log stream by Rappl (2024). 154
Figure 46 – Access to different domains through a Federated API. Source: Wessels (2020). 156
Figure 47 – Unidirectional data flow following the Model-View-Intent (MVI) pattern. 157
Figure 48 – Micro Frontends and Microservices grouped in cross-functional teams defined by domain. 159
Figure 49 – Cyclic communication between fragments on the same screen. 160
Figure 50 – Communication between fragments using an Event Store. 161
Figure 51 – Wrapper component as a solution to extended libraries. 162
Figure 52 – MFE-A is a central point (Hub) of dependency between the other MFEs. 170
Figure 53 – Home screen is a screen with several fragments, and when Fragment B raises an error, the entire screen becomes unavailable. 171

Figure 54 – When Fragment B raises an error, an user-friendly fallback message is rendered so the entire screen remains available. 171
Figure 55 – To allow physical products payment, optional physical product-specific attributes are added and digital product-specific attributes become optional. 173
Figure 56 – Mega frontend break into two MFEs. 175
Figure 57 – Page from a diagram editing platform with components as MFE, Nano Frontends. 179
Figure 58 – CI and CD cycle. Source: <https://www.abtasty.com/resources/ci-cd/> 180
Figure 59 – Black box pattern to avoid one MFE for all. 182
Figure 60 – Anti-corruption layer to connect to legacy systems. 185
Figure 61 – My profile screen of mfe-users. 206
Figure 62 – Personal data screen of mfe-users. 206
Figure 63 – Login screen of mfe-security. 207
Figure 64 – Two-factor authentication screen of mfe-security. 207
Figure 65 – Products sale screen of mfe-store. 208
Figure 66 – History screen of mfe-store. 208
Figure 67 – Product details screen of mfe-store. 209
Figure 68 – Recommended products fragment of mfe-store. 209
Figure 69 – Purchases screen of mfe-purchase. 210
Figure 70 – Cart screen of mfe-purchase. 210
Figure 71 – Search screen of mfe-search. 211
Figure 72 – Home screen of mfe-search. 211
Figure 73 – Profile screen of mfe-users. 213
Figure 74 – Login screen of mfe-security. 214
Figure 75 – First screen of sign up of mfe-security. 215
Figure 76 – Card invoices screen of mfe-cards. 216
Figure 77 – Limits screen of mfe-cards. 217
Figure 78 – Current card invoice screen of mfe-cards. 218

Figure 79 – Account statement screen of mfe-digital-account. 219
Figure 80 – Account balance screen of mfe-digital-account. 220
Figure 81 – Pix transfer screen of mfe-digital-account. 221
Figure 82 – Loans list screen of mfe-loan. 222
Figure 83 – Loan details screen of mfe-loan. 223
Figure 84 – Loan section fragment of mfe-loan. 223
Figure 85 – Investments list screen of mfe-investment. 224
Figure 86 – Investment details screen of mfe-investment. 225
Figure 87 – Loan section fragment of mfe-investment. 225
Figure 88 – Home screen of mfe-home. 226
Figure 89 – Meli+ Offer Screen: The screen displays the benefits and available subscription plans. 228
Figure 90 – Subscription Terms and Conditions Screen: The screen shows the terms and conditions required for the subscription. 228
Figure 91 – Offer Component with Details: This component should be displayed on the homepage, showing detailed information about the offer. . . . 229
Figure 92 – Simple Offer Component: This component should be displayed in the user account menu, providing an overview of the offers. 229
Figure 93 – Offer Component for the Header: This component should be displayed in the header, highlighting the current offer. 230
Figure 94 – Plan Selection Modal: A modal that allows the user to choose the desired subscription plan. 230
Figure 95 – Screen showing the list of all favorite items and the user's product lists: This screen displays all the favorite items and the product lists created by the user. 231
Figure 96 – Screen showing products belonging to a specific list: This screen presents the products that are part of a specific user-created list. . . . 231
Figure 97 – Button to add a product to a list: This button opens a modal allowing the user to choose which list they want to add the product to or create a new list. 232

Figure 98 – Modal to create a new list: This modal allows the user to create a new product list. 232
Figure 99 – Main Screen of NuCoin. 235
Figure 100–Screen showing NuCoin balance statement. 236
Figure 101–Screen detailing raffles of NuCoin. 237
Figure 102–Screen with benefits of NuCoin. 238
Figure 103–Screen to choose what to charge to the credit. 239
Figure 104–Screen showing purchases that can be paid in installments. 240
Figure 105–Fragment with the option to parcel a Pix payment. 240

List of Tables

Table 1 – Architects Don't Code Anti-pattern (BRADA; PICHA, 2019) 42
Table 2 – Summary of papers retrieved from Google Scholar 51
Table 3 – MS anti-patterns used to propose the MFE anti-patterns. 52
Table 4 – Summary of the survey participants' characterization. 67
Table 5 – Overall results from quantitative analysis ranked by harmfulness score. 70
Table 6 – MFE sessions' content. 98
Table 7 – Adapted TAM constructs and the sentences related to the catalog's utility for learning. 102
Table 8 – Summary of the experiment's participants' characterization. 104
Table 9 – Quality Assessment checklist for Grey Literature publications. 128
Table 10 – Quality Assessment checklist for Grey Literature publications. 130
Table 11 – Selected publications. 133
Table 12 – Anti-patterns original names and references. 137

List of abbreviations and acronyms

CDN Content Delivery Network

CSC Client–side Composition

DDD Domain–driven Design

DoD Definition of Done

EC Exclusion Criteria

ESC Edge–side Composition

ESI Edge Side Include

GT Grounded Theory

IC Inclusion Criteria

ICR Intercoder Reliability

MFE Micro Frontend

MS Microservice

POC Proof–of–concept

RQ Research Question

RR Rapid Review

SEO Search Engine Optimization

SLR Systematic Literature Review

SSC Server–side Composition

UI User Interface

UX User Experience

CONTENTS

1 INTRODUCTION 30
1.1 Research Objectives 32
1.2 Research Methodology 32
1.3 Organization 34

2 BACKGROUND 36
2.1 Microservices 36
2.2 Micro frontends 37
2.3 Anti-patterns 40
2.4 Related work 42

3 PROPOSAL OF A MICRO FRONTENDS ANTI-PATTERNS CATALOG 46
3.1 Rapid Review 46
3.1.1 Search Protocol 47
3.1.2 Conducting the Rapid Review 48
3.1.2.1 ACM Digital Library 48
3.1.2.2 IEEE Xplore 48
3.1.2.3 Google Scholar 49
3.1.3 Results 51
3.2 Initial version of the catalog 52
3.2.1 Cyclic Dependency 53
3.2.2 Knot Micro Frontend 54
3.2.3 Hub-like Dependency 55
3.2.4 Nano Frontend 55

3.2.5 Mega Frontend . . . 56
3.2.6 Micro Frontend Greedy . . . 56
3.2.7 No CI/CD . . . 57
3.2.8 No Versioning . . . 57
3.2.9 Lack of skeleton . . . 58
3.2.10 Common Ownership . . . 59
3.2.11 Golden Hammer . . . 59
3.2.12 Micro Frontend as the goal . . . 60

4 CATALOG IMPROVEMENT BASED ON PRACTITIONER'S FEEDBACK . . . 62
4.1 Study Design . . . 62
4.2 Results . . . 66
4.2.1 Participants' Characterization . . . 66
4.2.2 Quantitative Analysis . . . 68
4.2.3 Thematic Analysis . . . 72
4.3 Discussion . . . 75
4.4 Threats to validity . . . 76
4.5 Improved catalog based on practitioners' feedback . . . 78
4.5.1 Cyclic Dependency . . . 78
4.5.2 Knot Micro Frontend . . . 80
4.5.3 Hub-like Dependency . . . 81
4.5.4 Nano Frontend . . . 82
4.5.5 Mega Frontend . . . 83
4.5.6 Micro Frontend Greedy . . . 84
4.5.7 No CI/CD . . . 85
4.5.8 No Versioning . . . 85
4.5.9 Lack of Skeleton . . . 86
4.5.10 Common Ownership . . . 87
4.5.11 Golden Hammer . . . 88
4.6 Catalog's web application . . . 89

5 STUDENT FEEDBACK ON THE CATALOG AND ITS CONTRIBUTION FOR LEARNING . . . 92
5.1 Study Design . . . 93
5.1.1 Goal and Research Questions . . . 93
5.1.2 Planning . . . 94
5.1.2.1 Context Selection . . . 94
5.1.2.2 Variable selection . . . 95
5.1.2.3 Hypothesis formulation . . . 95
5.1.2.4 Selection of subjects . . . 95
5.1.2.5 Experiment design . . . 96
5.1.2.6 Instrumentation . . . 97
5.1.3 Execution . . . 98
5.1.4 Analysis and Interpretation . . . 101
5.2 Results . . . 103
5.2.1 Participants' Characterization . . . 103
5.2.2 RQ1: Supporting Materials Comparison . . . 105
5.2.3 Perceived learning Difference . . . 106
5.2.4 How students used the catalog . . . 107
5.2.4.1 TAM and learning statements analysis . . . 107
5.2.4.2 Qualitative feedback analysis . . . 110
5.3 Discussion . . . 112
5.4 Threats to validity . . . 113
5.5 Improved catalog based on students' feedback . . . 115
5.5.1 Knot Micro Frontend . . . 116
5.5.2 Nano Frontend . . . 117
5.5.3 Mega Frontend . . . 119
5.5.4 No CI/CD . . . 120
5.5.5 Common Ownership . . . 121

6 CATALOG EXPANSION THROUGH A MULTIVOCAL LITERATURE REVIEW . . . 123

6.1 MLR Protocol . . . 124
6.1.1 Goal and Research Questions . . . 124
6.1.2 Search String . . . 124
6.1.3 Sources . . . 125
6.1.4 Selection Criteria . . . 126
6.1.5 Search and Selection Process . . . 126
6.1.5.1 First Filter . . . 127
6.1.5.2 Quality Assessment . . . 127
6.1.5.3 Second Filter . . . 129
6.1.6 Data Extraction . . . 129
6.1.7 Data Synthesis . . . 130
6.2 Results . . . 131
6.2.1 Selected publications . . . 131
6.2.2 Publications overview . . . 132
6.2.3 RQ1: Which Micro Frontends anti-patterns have been proposed in White and Grey Literature? . . . 133
6.2.4 RQ2: How are Micro Frontends anti-patterns classified? . . . 138
6.2.5 RQ3: Are the proposed Micro Frontend anti-patterns grounded in theory or based on professional experience? . . . 138
6.2.6 RQ4: What is the profile of the authors who propose Micro Frontend anti-patterns in Grey Literature? . . . 139
6.3 Discussion . . . 139
6.4 Threats to Validity . . . 141
6.5 Catalog's improvement . . . 142
6.5.1 Avoiding observability . . . 142
6.5.2 Access to different domains . . . 143
6.5.3 Bidirectional Data Flow . . . 143
6.5.4 Chatty Micro Frontends . . . 144
6.5.5 Dependent deployment . . . 144
6.5.6 Dismissing human factors . . . 145

6.5.7 Distributed Data Inconsistency . . . 145
6.5.8 Framework Frenzy . . . 145
6.5.9 Global state communication . . . 145
6.5.10 Nano Frontend . . . 146
6.5.11 One Micro Frontend for all . . . 146
6.5.12 Partial UI Migration . . . 146
6.5.13 Spaghetti Architecture . . . 147
6.5.14 Unmediated Legacy Integration . . . 147

7 FINAL CATALOG . . . 149
7.1 Template evolution . . . 149
7.2 Web application evolution . . . 151
7.3 The Final Micro Frontends Anti-patterns Catalog . . . 153
7.3.1 Avoiding observability . . . 153
7.3.2 Access to different domains . . . 154
7.3.3 Bidirectional Data Flow . . . 155
7.3.4 Chatty Micro Frontends . . . 157
7.3.5 Common Ownership . . . 158
7.3.6 Cyclic Dependency . . . 159
7.3.7 Dependency hell . . . 161
7.3.8 Dependent deployment . . . 163
7.3.9 Dismissing human factors . . . 164
7.3.10 Distributed Data Inconsistency . . . 165
7.3.11 Framework Frenzy . . . 166
7.3.12 Global state communication . . . 167
7.3.13 Golden Hammer . . . 168
7.3.14 Hammering APIs . . . 169
7.3.15 Hub-like Dependency . . . 170
7.3.16 Knot Micro Frontend . . . 172
7.3.17 Lack of Skeleton . . . 173
7.3.18 Mega Frontend . . . 174

7.3.19 Micro Frontends Greedy . . . 175
7.3.20 Micro Frontend as the goal . . . 176
7.3.21 Nano Frontend . . . 177
7.3.22 No CI/CD . . . 178
7.3.23 No Versioning . . . 180
7.3.24 One Micro Frontend for all . . . 181
7.3.25 Partial UI Migration . . . 182
7.3.26 Spaghetti Architecture . . . 183
7.3.27 Unmediated Legacy Integration . . . 184

8 FINAL CONSIDERATIONS . . . 186
8.1 Research Overview . . . 186
8.2 Contributions . . . 188
8.3 Future Work . . . 189

References . . . 191

APPENDIX A SURVEY THEMATIC ANALYSIS . . . 198

APPENDIX B CONTROLLED EXPERIMENT OBJECTS . . . 204
B.1 Object 1 - Mercado Livre . . . 204
B.1.1 mfe-users . . . 205
B.1.2 mfe-security . . . 206
B.1.3 mfe-store . . . 207
B.1.4 mfe-purchase . . . 209
B.1.5 mfe-search . . . 210
B.1.6 mfe-home . . . 211
B.2 Object 2 - Nubank . . . 212
B.2.1 mfe-users . . . 212
B.2.2 mfe-security . . . 213
B.2.3 mfe-cards . . . 215
B.2.4 mfe-digital-account . . . 218

B.2.5 mfe-loan 221
B.2.6 mfe-investment 224
B.2.7 mfe-home 226

APPENDIX C CONTROLLED EXPERIMENT ASSESSMENTS . . 227
C.1 Object 1 Questions 227
C.2 Object 2 Questions 234

APPENDIX D CONTROLLED EXPERIMENT CODING 243

APPENDIX E MLR PUBLICATIONS 247

APPENDIX F MLR DUPLICATE EXCLUSION 269

APPENDIX G MLR FIRST FILTER 271

APPENDIX H MLR QUALITY ASSESSMENT 279

APPENDIX I MLR SECOND FILTER 281

APPENDIX J MLR DATA EXTRACTION 283

1

Introduction

Software architecture focuses on software design at the highest level of abstraction (KAZMAN et al., 2023). At this level, architects are concerned not with specific classes or interfaces but how the system's components, modules, or layers are integrated (VALENTE, 2020). During the design phase, architects face critical decisions that shape the system, as these choices are often difficult to modify later. To support these decisions, architects can rely on architectural styles, which provide guidelines on organizing the system's modules (MEDVIDOVIC; TAYLOR, 2010). Examples of such architectural styles include Microservices and Micro Frontends.

As a monolithic application grows, it becomes challenging to scale due to limitations like technology constraints, the necessity for vertical scaling only, and the need to reboot the entire application with each deployment (DRAGONI et al., 2017). To address these issues, developers are adopting the Microservice (MS) architectural style to create autonomous, distributed, and loosely coupled services (DMITRY; MANFRED, 2014; LEWIS; FOWLER, 2014; ERL, 2016). This architecture enables teams to work independently, reducing the development time for new features. However, different teams often still need to share the same codebase for the presentation layer.

Micro Frontend (MFE) is an architectural style that extends the principles of MS to the frontend, breaking down a complex frontend application into smaller, manageable, and independently deployable slices (MEZZALIRA, 2021a; PELTONEN; MEZZALIRA; TAIBI, 2021). This approach facilitates independent testing, development, and deployment of front-end components, enabling teams to work autonomously and reducing the

development time for new features (GEERS, 2020). However, some issues faced when adopting MFEs are increased payload size, UX consistency, complex monitoring and debugging, state management, and duplicated code (PELTONEN; MEZZALIRA; TAIBI, 2021). Many companies, such as SAP, Springer, Zalando, NewRelic, Ikea, Starbucks, and DAZN, have successfully implemented the MFE architecture (TAIBI; MEZZALIRA, 2022), showcasing its potential in diverse domains.

Despite its widespread adoption in the industry, the academic literature about guidelines and best practices when implementing MFE is relatively limited, especially when compared to the numerous experience reports and case studies documenting its implementation (ANTUNES et al., 2024; CAPDEPON et al., 2023; KAUSHIK; KUMAR; RAJ, 2024; PERLIN et al., 2023; MÄNNISTÖ; TUOVINEN; RAATIKAINEN, 2023; PÖLÖSKEI; BUB, 2021; MORAES et al., 2024). This disparity suggests a challenge in transferring knowledge from industry practice to academic research and underscores a gap in understanding how enterprises implement MFE in real-world architectures.

Over time, software architecture can deteriorate due to developers' insufficient understanding of the specific architectural style (TAIBI; LENARDUZZI; PAHL, 2020). This issue is particularly critical in MFE architectures, as they are inherently complex, and there is no well-defined method for evaluating or documenting both good and bad practices. Therefore, the objective of this master's thesis is to develop artifacts that can support developers in implementing and maintaining MFE architectures. We propose a catalog of MFE anti-patterns to preserve architectural integrity and assist developers in making well-informed decisions. Anti-patterns address emerging issues, common mistakes, poorly implemented solutions, misapplied best practices, and deviations from established process models (BRADA; PICHA, 2019).

To evaluate the catalog, we first conducted a Personal Opinion Survey (KITCHENHAM; BUDGEN; BRERETON, 2015) with practitioners to validate whether the identified problems are prevalent in MFE architectures and if the proposed solutions address them effectively. Based on their feedback, we improved the catalog. We then carried out a controlled experiment with undergraduate students to compare the use of the catalog to MFE guidelines provided on blogs and evaluate whether the catalog enhances

students' perception of learning. Again, the catalog was refined based on feedback. Our next steps involve expanding the catalog and empirically evaluating its use by practitioners in real-world architectures. In the following sections, we outline our research objectives and present the methodology used in this study.

1.1 Research Objectives

The primary objective of this thesis is **to develop an artifact to support developers when implementing and maintaining MFEs architectures**. To satisfy the primary objective, we devised the following specific objectives:

- Build a comprehensive body of knowledge on the real-world challenges developers face when implementing Micro Frontend architectures.
- Develop and iteratively improve an artifact to support developers when implementing and maintaining Micro Frontend architectures.
- Examine how the artifact supports inexperienced developers in identifying and avoiding common pitfalls while maintaining Micro Frontend architectures.

1.2 Research Methodology

The research methodology employed in this thesis is an adapted version of the evidence-based methodology proposed by Mafra, Barcelos & Travassos (2006). Figure 1 illustrates the steps of our adapted approach. The methodology consists of the following steps:

1. **Secondary Study**: After conducting an ad hoc review, we identified a Multivocal Literature Review by Peltonen, Mezzalira & Taibi (2021) that maps existing knowledge on MFEs, highlighting their motivations, benefits, and challenges. The authors state that MFE anti-patterns have not yet been identified. To complement their review and verify whether any anti-patterns have been published since its

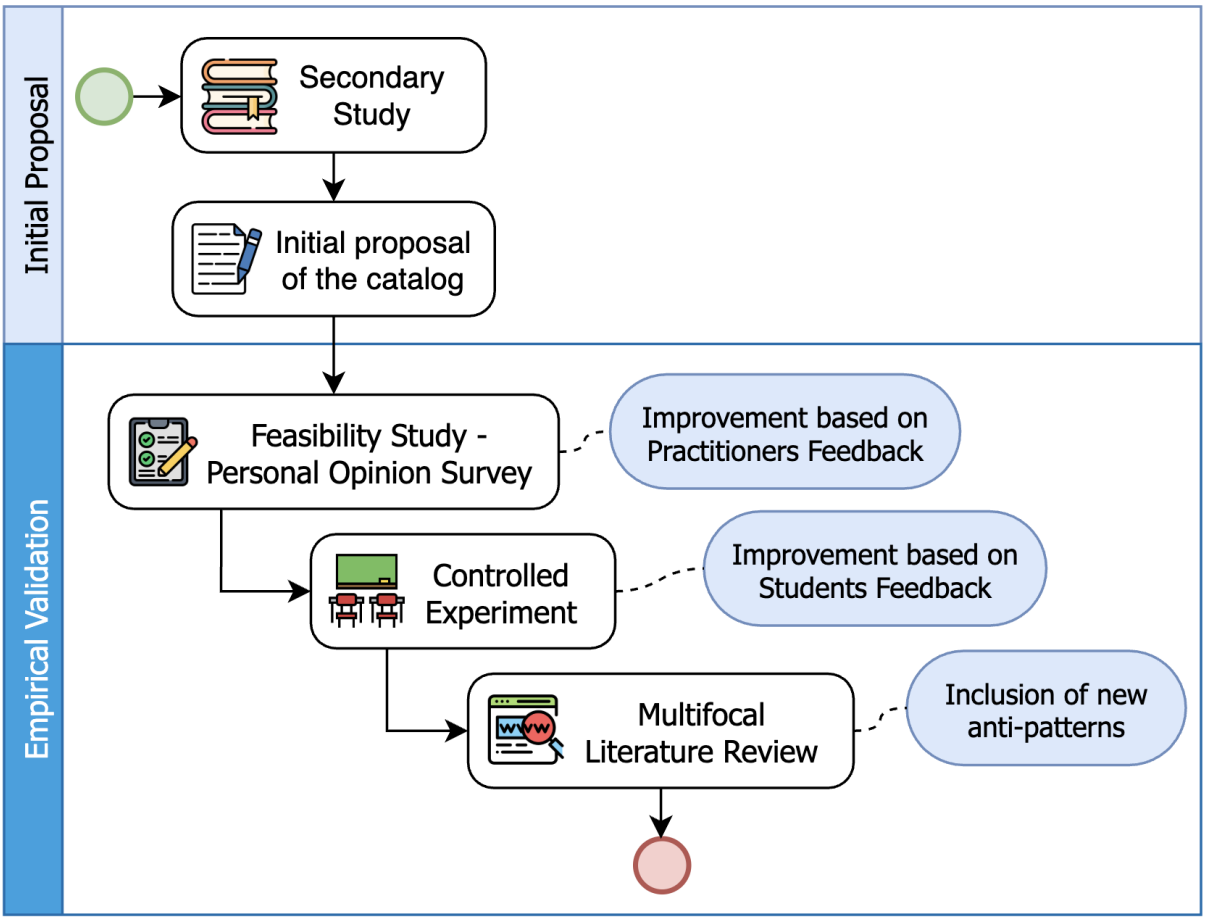

Figure 1 – Diagram illustrating the research methodology based on Mafra, Barcelos & Travassos (2006). Blue rounded items show what we made after each step.

completion, we conducted a Rapid Review (CARTAXO; PINTO; SOARES, 2018) and found no documented MFE anti-patterns.

2. **Initial proposal of the catalog**: To document the challenges faced when implementing and maintaining MFEs architectures, we propose a catalog of MFEs anti-patterns based on MSs anti-patterns. We propose 12 anti-patterns, each composed by name, category, problem, solution, and example.

3. **Feasibility study**: We conducted a Personal Opinion Survey (KITCHENHAM; BUDGEN; BRERETON, 2015) with practitioners to assess and improve the anti-patterns and evaluate if they represent real problems on MFE architectures. We improved the catalog anti-patterns based on practitioners' feedback.

4. **Controlled experiment**: We conducted a controlled experiment (WOHLIN et al., 2012) with Computer Science undergraduate students to compare the catalog to

the guidelines proposed by practitioners on websites and blogs, and evaluate whether the catalog improves the perceived learning of students. We improved the catalog anti-patterns based on students' feedback.

5. **Multivocal Literature Review**: To uncover micro frontend (MFE) anti-patterns proposed by practitioners in informal sources, we plan to conduct a Multivocal Literature Review (GAROUSI; FELDERER; MÄNTYLÄ, 2019). By incorporating anti-patterns shared by developers in online communities, blogs, and technical forums, we aim to enrich and refine our catalog, producing a more comprehensive and practice-informed final version.

1.3 Organization

This chapter presented the context, research objectives, and methodology of this master's thesis. The remaining chapters are organized as follows:

- **Chapter 2**: Presents the theoretical background of our research, including definition of MSs, MFEs, and anti-patterns. Additionally, we discuss the related work.
- **Chapter 3**: Presents the Rapid Review we conducted to identify published MFE anti-patterns, details the process of proposing the MFE anti-patterns catalog, and introduces its initial version.
- **Chapter 4**: Presents the Personal Opinion Survey we conducted to evaluate and enhance the catalog based on feedback from practitioners. We also present some of the improved anti-patterns refined based on the feedback provided by practitioners.
- **Chapter 5**: Presents the controlled experiment we conducted to compare the catalog and guidelines proposed by developers and assess whether the catalog improves the students' perceived learning. We also present some of the improved anti-patterns refined based on the feedback provided by students.

- **Chapter 6**: Presents the Multivocal Literature Review conducted to expand the catalog by adding anti-patterns proposed by developers in grey literature sources. We present the newly added anti-patterns as well as improvements made to some of the existing ones based on the review results.
- **Chapter 7**: Presents the final refinements made to the catalog and introduces its final version.
- **Chapter 8**: Presents a summary, the contributions, and concludes this research.

2

BACKGROUND

Micro Frontend (MFE) is an architectural style that draws heavily from the principles of Microservices (MS), making it essential first to understand the foundational concepts of both. This chapter provides the necessary background to contextualize and deepen the understanding of this research. Section 2.1 presents the core principles of MS. Section 2.2 delves into the concept and implementation of MFEs. Section 2.3 introduces the definition of anti-patterns, as our artifact is a catalog of MFE anti-patterns. Finally, Section 2.4 reviews related work to offer additional context.

2.1 Microservices

Microservice is an architectural style presented as an alternative to monolith architectures (ABGAZ et al., 2023). Lewis & Fowler (2014) first defined MS as an approach to developing a single application as a suite of small services, each running in its process and communicating with lightweight mechanisms. A service is a self-contained, loosely coupled software unit designed to perform a specific business function. The difference between this style and other service-based architectures, such as Service-Oriented Architecture (SOA), is that microservices are smaller in code volume, have access to only their databases, have lightweight communication patterns, and are defined by domain (RICHARDSON, 2018).

The modular design of MS ensures that each service has a well-defined and

focused set of responsibilities, which promotes maintainability and scalability (ERL, 2016). Adopting MS provides several benefits, such as enabling Continuous Delivery and Continuous Integration for large and complex architectures, offering small and easy-to-maintain services, allowing independent deployment and scaling, fostering team autonomy, isolating failures, and enabling the adoption of different technologies based on the needs of each service (GEERS, 2020). However, there are also some obstacles to using MS, such as the challenge of defining the correct set of microservices, increased complexity in the development, testing, and deployment of the entire system, the need for careful coordination when developing features that access multiple microservices, and the difficulty of deciding when to adopt this architecture (GEERS, 2020; RICHARDSON, 2018; NEWMAN, 2021).

2.2 Micro frontends

The term "Micro Frontend" was first coined by ThoughtWorks in 2016 (THOUGHTWORKS, 2016) as an architectural style inspired by MS architecture. The main idea is to decompose a monolithic frontend application into smaller parts that can be developed, deployed, and updated independently, promoting greater flexibility and maintainability (MEZZALIRA, 2021a; PELTONEN; MEZZALIRA; TAIBI, 2021). MFE can also be considered an organizational approach. The application is divided into vertical slices built from the database to the user interface and run by a dedicated team (GEERS, 2020). In an MFE architecture, the web application integrates different features or business sub-domains, and each software team should have only one domain to handle (PELTONEN; MEZZALIRA; TAIBI, 2021). Many companies adopted the MFE architecture, such as SAP, Springer, Zalando, NewRelic, Ikea, Starbucks, and DAZN (TAIBI; MEZZALIRA, 2022).

MFEs share the main principles, benefits, and issues of MS (THOUGHTWORKS, 2016; TAIBI; MEZZALIRA, 2022; MEZZALIRA, 2021a). The motivations are development scalability and codebase growth, and the benefits include support for different technologies, autonomous cross-functional teams, development, deployment, manage-

ment independence, and better testability (PELTONEN; MEZZALIRA; TAIBI, 2021). However, the primary drawbacks of Micro Frontends include increased payload size, as each MFE often bundles its dependencies, potentially duplicating libraries and inflating the assets delivered to the client (PELTONEN; MEZZALIRA; TAIBI, 2021; GEERS, 2020). Monitoring and debugging also become more complex, as issues may span multiple independently deployed frontends, making it harder to trace logs and isolate the root cause. Additionally, state management is challenging, as inconsistencies can arise when each MFE maintains its state without proper coordination. Finally, duplicated code is a common issue since UI components are frequently reimplemented across MFEs, resulting in inconsistencies and increased maintenance effort.

Each MFE implements a set of screens and fragments. Fragments are reusable User Interface (UI) components that can be combined to form screens across different MFEs (GEERS, 2020). Some fragments might need context information, but the team, including the fragment in their MFE, does not have to know the fragments' state or implementation details. The MFEs must integrate a coherent application to deliver a unified User Experience (UX) (GEERS, 2020). Achieving this requires developers to make informed decisions regarding the composition, communication, and routing between the MFEs.

The first decision is about composition, the process of requesting the fragments and putting them in the correct slots in a screen (GEERS, 2020). There are three approaches to compose MFEs: Server-side, Edge-side, and Client-side. Figure 2 shows a diagram exemplifying the composition of a screen with one fragment.

1. **Client-side Composition (CSC)**: An application shell loads MicroFrontends inside itself. An application shell is technically represented by an HTML file always present during the user session containing a small JavaScript code for loading and orchestrating different MFEs (TAIBI; MEZZALIRA, 2022; PELTONEN; MEZZALIRA; TAIBI, 2021; GEERS, 2020). CSC needs MFEs-specific frameworks (MORAES et al., 2024), such as single-spa (SINGLE-SPA, 2016), qiankun (QIANKUN, 2019) and Garfish (GARFISH, 2021), or with frameworkless technologies like Webpack, Module Federation, iFrames, and web components.

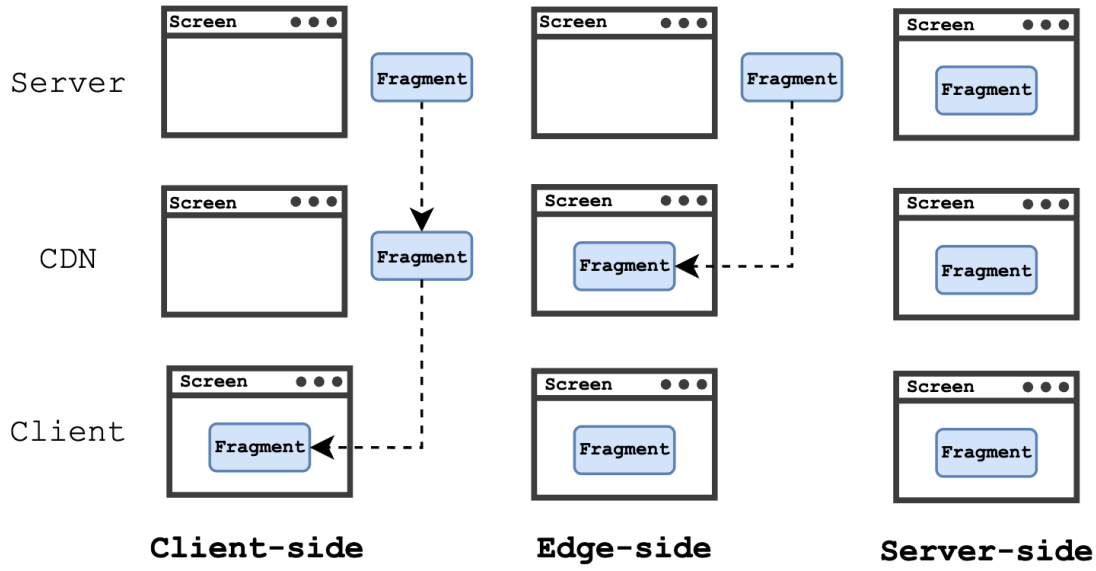

Figure 2 – Diagram exemplifying the composition of a fragment in a screen in each composition approach.

2. **Edge-side Composition (ESC)**: The web page is assembled at the Content Delivery Network (CDN) level, using an XML-based markup language called Edge Side Include (ESI) (PELTONEN; MEZZALIRA; TAIBI, 2021). One of the drawbacks of this implementation is that ESI is not implemented in the same way by each CDN provider, which can lead to many refactors and new logic implementation.

3. **Server-side Composition (SSC)**: The origin server is composing the view by retrieving all the different MFEs and assembling the final page. It can happen at runtime or compile time (PELTONEN; MEZZALIRA; TAIBI, 2021; GEERS, 2020). It is the simplest approach because it enables the development of MFEs as packages (MORAES et al., 2024). However, this approach does not meet some of the main principles of MFE, such as technology agnosticism and independent delivery.

Once the fragments are composed into a screen, the team needs to decide how the MFEs will communicate with each other. Effective communication outlines how the screen and fragments interact to deliver a seamless UX (TAIBI; MEZZALIRA, 2022). Geers (2020) defines three primary forms of communication: Parent to Fragment, Fragment to Parent, and Fragment to Fragment, as shown in Figure 3.

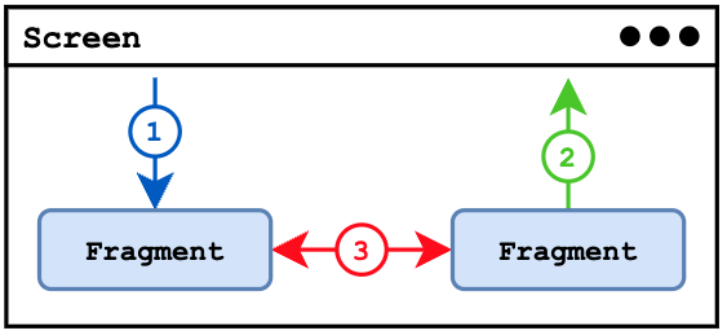

Figure 3 – The three forms of communication in MFEs: (1) Parent to Fragment: communication initiated by the parent (screen) towards a specific MFE fragment; (2) Fragment to Parent: communication initiated by a fragment back to the screen; and (3) Fragment to Fragment: communication directly between different fragments.

1. **Parent to Fragment Communication**: A change in the screen is propagated down to one or more fragments so they can update themselves. This form is also called Parent-child Communication.

2. **Fragment to Parent Communication**: A change on a fragment sends a message to the screen so it can update itself. This form is also called Child-parent Communication.

3. **Fragment to Fragment Communication**: A change on a fragment sends a message to one or more fragments composed on the same screen. This form is also called Child-child Communication.

The final decision involves routing, which delineates the navigation from one view to another (TAIBI; MEZZALIRA, 2022). Usually, hyperlinks are necessary to navigate between screens. When adopting CSC, the application shell manages routing, which knows all routes and MFEs, mapping URLs to the correct MFE.

2.3 Anti-patterns

Anti-patterns are recurring design practices, choices, or solutions to common problems. Despite appearing reasonable and effective, they lead to negative consequences and undermine the system's overall quality (CERNY et al., 2023). An anti-pattern is similar to

a pattern, but instead of providing a practical solution, it offers an approach that appears to be a solution on the surface. However, it ultimately leads to more problems (KOENIG, 1998). While a problem and its optimal solution characterize patterns, anti-patterns involve two solutions (BROWN et al., 1998). The first solution is a commonly occurring solution that generates overwhelmingly negative consequences. The second solution is a commonly occurring method in which the anti-pattern can be resolved and reengineered into a more beneficial form.

Koenig (1998) define three forms to define anti-patterns: (1) Degenerative Form: the most basic one, is a textual description without any structure, template, or separate content sections for various aspects of the pattern; (2) Mini-AntiPattern: a more structured approach, consists of name, problem, and solution; and (3) Formal Definitions: including the Full AntiPattern Template and the Laplante-Neil Structure, consists of multiple fields detailing various dimensions of the anti-pattern. Brada & Picha (2019) discusses the template used in the C2 Wiki repository, which is a Formal Definition. They also propose a new template to document software process anti-patterns that balance brevity and information clarity based on the formal definitions. Table 1 presents an example of a software process anti-pattern from Brada & Picha (2019).

Table 1 – Architects Don't Code Anti-pattern (BRADA; PICHA, 2019)

Name	Architects Don't Code
Also Known As	-
Summary	System architects do not participate in development efforts (e.g., because their time is expensive). Thus they ultimately create designs just "on paper" which might be flawed but which the developers are supposed to follow.
Context	More likely in waterfall projects. The development organization has many junior programmers and relatively few experts, such as senior programmers or solution architects.
Forces	(1) Having an expert architect should bring about consistency, cleanliness, modularity, and other characteristics of efficient software; and (2) Expert time is expensive, rare or both.
Symptoms and Consequences	(1) People with the architect role do not interact with coding tasks (tickets); (2) Architects do not generate or modify any source code artifacts; (3) Architects only interact with non-coding people, tasks and artifacts; and (4) Code reflects a design that the architect never thought of because the one he came up with was flawed.
Causes	The architect never codes and is uninterested in "implementation details".
Solution	Get architects involved at an implementation level to get their feet wet from time to time and become aware of how (changes in) their design affect the project. (See the pattern "Architect Also Implements".)
Related Anti-patterns	Viewgraph Engineering – similar in kind (technical role does not get hands dirty in technical tasks)
Sources	Cunningham, W. Management Anti Pattern Road Map [online]. <http://wiki.c2.com/?ManagementAntiPatternRoadMap>

2.4 Related work

Several papers present case studies or experience reports on implementing MFE architectures. Männistö, Tuovinen & Raatikainen (2023) presented their experience through the migration of a monolithic frontend to the MFE architecture. Capdepon et al. (2023) proposed an approach to migrate from monolithic mobile architecture to MFE. Moraes et al. (2024) provided an experience report demonstrating how the same application can be implemented using different MFE approaches. Perlin et al. (2023) presented a case study of how to implement a MFE application with Webpack. Kaushik, Kumar &

Raj (2024) proposed a framework for the design of web applications with MFEs and MS. They conducted a case study to empirically analyze and evaluate the effectiveness of the proposed framework. Pavlenko et al. (2020) implemented MFEs case study to report and discuss the issues risen during development. All these papers can assist in making architectural decisions when implementing a MFE architecture. However, they cannot be used to evaluate an architecture or serve as a guide for identifying hidden problems within it.

Although some practitioners have shared their experience on anti-patterns in MFE applications on blogs and keynotes (MEZZALIRA, 2023; SHINDE, 2022; RAPPL, 2024), no scientific studies have been conducted to propose a catalog of MFE anti-patterns. We conducted a Rapid Review and found no studies proposing anti-patterns for MFEs (see Chapter 3). Mezzalira (2021a) briefly mentions that sharing any state between micro-frontends is considered an anti-pattern but does not provide an in-depth definition or exploration of this or other anti-patterns. Peltonen, Mezzalira & Taibi (2021) conducted a Multivocal Literature Review and stated that researchers have not yet deeply investigated MFEs and patterns and anti-patterns have not been defined. Taibi & Mezzalira (2022) presented a set of development aproaches based on their experience and reported the lack of pattern and anti-patterns definition. Moraes et al. (2024) reported a case study findings, which revealed that involuntary anti-patterns may occur since they are not yet mature, generating severe negative impacts on software projects.

Given the shared characteristics of MS and MFE architectural styles, we also searched for papers proposing MS anti-pattern. Through a Systematic Literature Review (SLR), Cerny et al. (2023) crafted a catalog with 58 disjoint MS anti-patterns grouped into five categories: Intra-service, Inter-service, Service interaction, Security and Team Anti-patterns. Tighilt et al. (2020) proposed a catalog with 16 MS anti-patterns based on a SLR and the analysis of 67 systems, examining them for potential violations of MS principles and design practices that could be indicative of anti-patterns. Building on practitioner experience, Taibi, Lenarduzzi & Pahl (2020) introduced a catalog and a taxonomy of the most common MS anti-patterns. Their three-year interview study identified 20

anti-patterns, including organizational and technical anti-patterns. Bogner et al. (2019) conducted a SLR to propose a taxonomy of 36 MS anti-patterns. Additionally, they developed a collaborative web application that allows users to explore and interact with their catalog. While both architectural styles share common anti-patterns, a dedicated catalog for MFEs is necessary to identify and address these issues specifically within the MFEs context.

On the educational side, previous studies have identified several challenges in teaching software architecture. According to Galster & Angelov (2016), students accustomed to seeking optimal programming solutions often find the "wicked" nature of architectural decision-making frustrating. The same study notes that small classroom examples often fail to convey the importance of architectural decisions, while the scarcity of realistic examples limits students' practical learning opportunities. Kazman et al. (2023) argue that undergraduate students often lack development experience and tend to think like programmers, which makes it difficult for them to understand high-level architecture and abstraction, as they focus on implementation details rather than high-level design. Lago & Vliet (2005) highlight that the absence of stakeholders to provide clear business rules during the software architecture design process poses a significant challenge. To improve software architecture courses, Mannisto, Savolainen & Myllarniemi (2008) state that they should emphasize using existing systems for maintenance and evolution tasks rather than solely designing architectures from scratch. This approach is more aligned with industry practices, where software architects commonly work with established systems.

To the best of our knowledge, no papers specifically discuss MFE education, and only a few report on experiences teaching MSs. Christensen (2022) describe the curriculum of an undergraduate course centered on DevOps and MSs, with a stronger emphasis on DevOps. They provide teaching guidelines that include focusing solely on architectural migration without adding new features to MSs to avoid overcomplication, using a technology stack familiar to students, and defining a monolith with clear domains and boundaries. Similarly, Lange, Koschel & Hausotter (2019) report on a MSs course conducted in collaboration with industry, where students attended lectures

before migrating a monolith to a MSs architecture. Cordeiro et al. (2019) propose an approach for teaching MSs involving lectures delivered by researchers and industry professionals. In their approach, students are organized into teams and tasked with developing different domains of a system, simulating real-world collaboration and distributed development. Overall, these MSs courses emphasize implementing new architectures migrated from monoliths but lack focus on exercising architecture design itself.

3

Proposal of a Micro Frontends Anti-patterns Catalog

To support developers during the implementations and maintenance of MFE architectures, we propose a catalog of MFE anti-patterns to document common issues and effective solutions to them. This chapter presents a Rapid Review we conducted to search for MFEs anti-patterns in the literature and proposes the initial version of our catalog of MFEs anti-patterns. Section 3.1 outlines the protocol, conduction, and results of the Rapid Review. Section 3.2 explains how we defined the anti-patterns and presents their initial version.

3.1 Rapid Review

Rapid Reviews (RRs) are lightweight secondary studies focused on delivering evidence to practitioners in a timely manner (CARTAXO; PINTO; SOARES, 2018). To detail the process and findings of the RR, Section 3.1.1 outlines the research question, the search string, and the inclusion and exclusion criteria; Section 3.1.2 presents the conduction of the search on different libraries; and Section 3.1.3 discusses the results of the RR.

3.1.1 Search Protocol

This RR was undertaken to identify MFE anti-patterns documented in the scientific literature. To achieve this objective, we formulated the following Research Question (RQ): **Are there MFE anti-patterns reported within the literature?** We designed a research string to gather all works related to MFE and anti-patterns, considering all possible variations in the terminology. The resulting search string is:

> (“microfrontend” OR “micro frontend” OR “micro–frontend” OR “mfe”) AND (“antipattern” OR “anti-pattern” OR “anti pattern”)

After defining the search string, the selection procedure followed the steps: (1) search and extract the studies in the ACM Digital Library[1], IEEE Xplore[2], and Google Scholar[3] using the search string; (2) eliminate duplicate studies; and (3) filter studies by title and abstracts using inclusion and exclusion criteria. We included the following Inclusion Criteria (IC):

- **IC1**: The paper must be in the context of MFE development;
- **IC2**: The paper propose or review MFE anti-patterns;

We also added the following Exclusion Criteria (EC):

- **EC1**: The paper is not a peer–reviewed publications (including preface, book, editorial, PhD dissertations, Master’s thesis, Graduate thesis, poster, panel, lecture, roundtable, workshop or demonstration);
- **EC2**: The paper is not written in english;
- **EC3**: The paper is a duplicate;
- **EC4**: The paper does not attend any of the inclusion criteria;

[1] <https://dl.acm.org/>
[2] <https://ieeexplore.ieee.org/>
[3] <https://scholar.google.com/scholar>

3.1.2 Conducting the Rapid Review

This subsection presents the papers retrieved from each digital library. During this process, we were able to set filters for publication years from 2019 to 2024.

3.1.2.1 ACM Digital Library

We searched the ACM Digital Library and retrieved no papers, as depicted in Figure 4. We performed the search by restricting it to papers published in the last six years (from 2019 to 2024) and applying search strings in the title and abstract as filters.

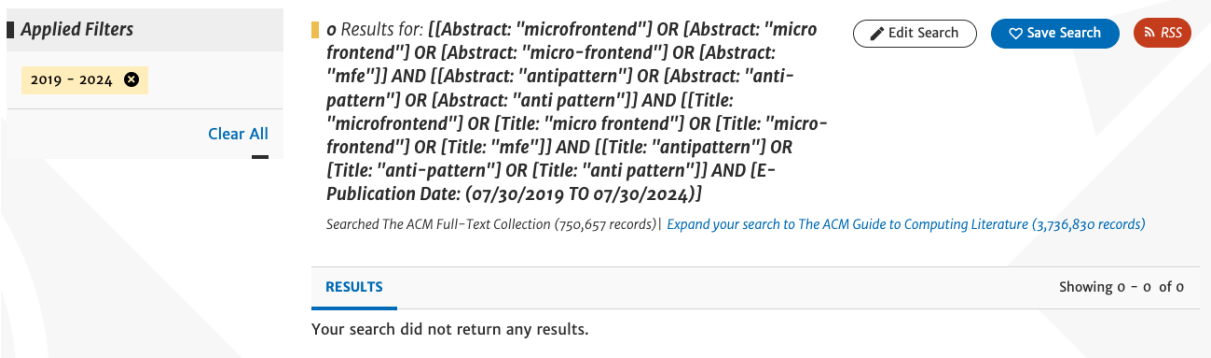

Figure 4 – Number of papers returned on the ACM Digital Library.

3.1.2.2 IEEE Xplore

We searched the IEEE Xplore Digital Library and retrieved no papers, as depicted in Figure 5. We performed the search by restricting it to papers published in the last six years (from 2019 to 2024) and applying search strings in the title and abstract as filters.

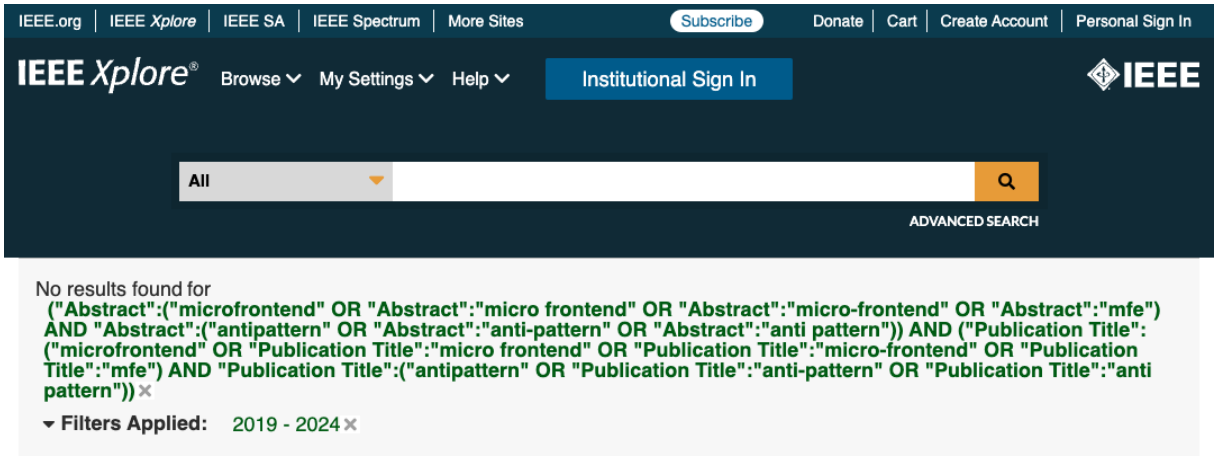

Figure 5 – Number of papers returned on the IEEE Xplore Library.

3.1.2.3 Google Scholar

We searched the Google Scholar and retrieved 14 papers, as depicted in Figure 6. We performed the search by restricting it to papers published in the last six years (from 2019 to 2024) and applying search strings in the title and abstract as filters, as shown in Figure 7. After reviewing all 14 papers, none of them were selected according to the inclusion and exclusion criteria, as shown in Table 2

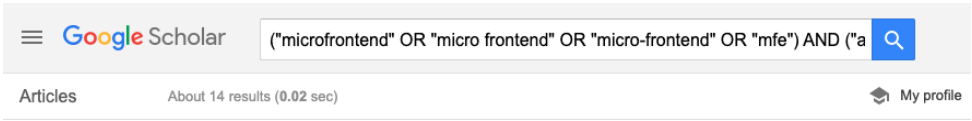

Figure 6 – Number of papers returned on the Google Scholar.

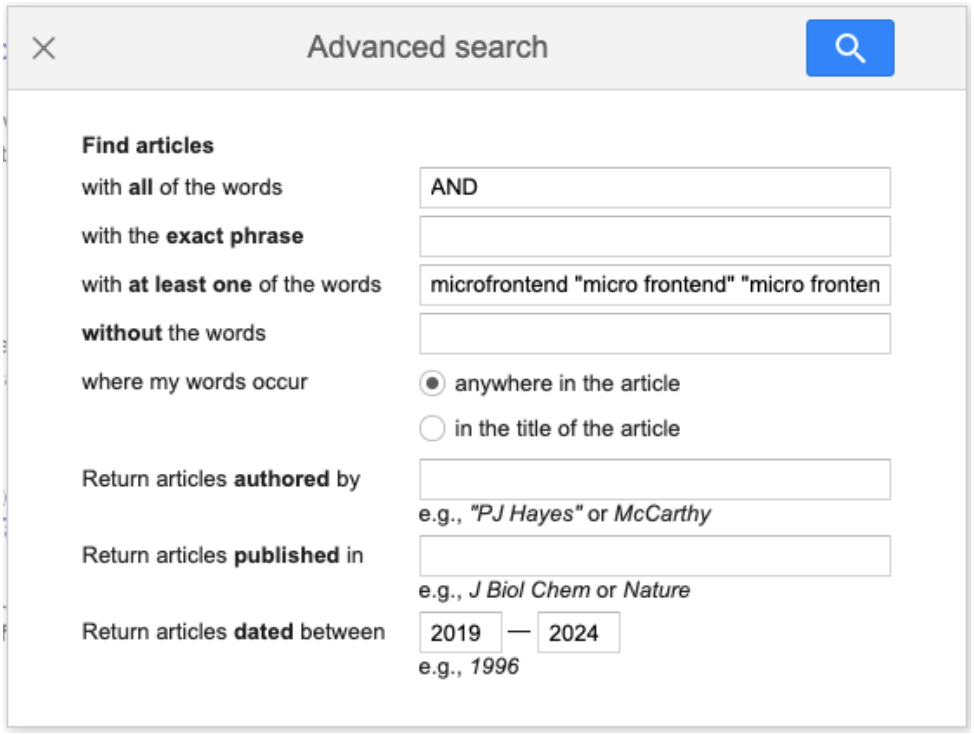

Figure 7 – Advanced search in Google Scholar.

Table 2 – Summary of papers retrieved from Google Scholar

REF.	**Title**	**Exclusion Reason**
(CAPDEPON et al., 2023)	Migration Process from Monolithic to Micro Frontend Architecture in Mobile Applications	Do not propose MFE anti-patterns (EC4)
(TOKUC, 2024)	Suitability of Micro-Frontends for an AI as a Service Platform	Master's thesis (EC1)
(MEZZALIRA, 2021a)	Building Micro-Frontends	Book (EC1)
(KLIMM, 2021b)	Design Systems for Micro Frontends	Graduate thesis (EC1)
(JUMPPONEN, 2021)	Modern Software Architecture	Book (EC1)
(VELEPUCHA; FLORES, 2023)	A Survey on Microservices Architecture: Principles, Patterns and Migration Challenges	The paper is focused on Microservices anti-patterns (EC4)
(FORD et al., 2021)	Software Architecture: The Hard Parts	Book (EC1)
(MARTINS, 2022)	Development of an e-portfolio social network using emerging web technologies	Master's thesis (EC1)
(KLIMM, 2021a)	Design Systems for Micro Frontends	The paper is a duplicate (EC3)
(FORD et al., 2022)	Building Evolutionary Architectures	Book (EC1)
(TUTISANI, 2023)	Design and Architecture	Book chapter (EC1)
(CHAPETON, 2022)	Collaborative Geovisual Analytics	Book (EC1)
(OBIORA et al., 2021)	Forecasting Hourly Solar Radiation Using Artificial Intelligence Techniques	Not related to MFE (EC4)
(SILVA, 2023)	Micro frontends numa aplicação de pré-contabilidade	Graduate thesis not written in English (EC1 and EC2)

3.1.3 Results

The rapid review conducted did not identify any existing studies focused on MFE anti-patterns, highlighting a significant research gap in the field. This dearth of dedicated research underscores the need for further investigation into this critical area of software development.

3.2 Initial version of the catalog

We proposed an initial catalog of 12 MFE anti-patterns derived from a conceptual adaptation of existing MS anti-patterns (as presented in Table 3) combined with a reflective analysis of real-world issues encountered in the development of MFE architectures. This process involved reviewing each anti-pattern from the microservices literature and evaluating its manifestation in MFE architectures based on the author's practical experience in MFE projects.

Table 3 – MS anti-patterns used to propose the MFE anti-patterns.

#	**Anti-pattern**	**References**
1	Cyclic Dependency	(TAIBI; LENARDUZZI; PAHL, 2020; TIGHILT et al., 2020; PARKER et al., 2023; CERNY et al., 2023; BOGNER et al., 2019)
2	Hub-like Dependency	(CERNY et al., 2023)
3	The Knot or Knot Service	(PARKER et al., 2023; CERNY et al., 2023)
4	Microservice Greedy	(TAIBI; LENARDUZZI; PAHL, 2020; PARKER et al., 2023; CERNY et al., 2023)
5	Nano Service	(TIGHILT et al., 2020; PARKER et al., 2023; CERNY et al., 2023; BOGNER et al., 2019)
6	Mega Service	(TAIBI; LENARDUZZI; PAHL, 2020; TIGHILT et al., 2020; CERNY et al., 2023; BOGNER et al., 2019)
7	Wrong Cuts	(TAIBI; LENARDUZZI; PAHL, 2020; PARKER et al., 2023; CERNY et al., 2023; BOGNER et al., 2019)
8	No API Versioning	(TAIBI; LENARDUZZI; PAHL, 2020; TIGHILT et al., 2020; CERNY et al., 2023; BOGNER et al., 2019)
9	No DevOps Tools or CI/CD	(TAIBI; LENARDUZZI; PAHL, 2020; TIGHILT et al., 2020; CERNY et al., 2023)
10	Microservices as the Goal	(TAIBI; LENARDUZZI; PAHL, 2020)
11	Lack of Microservice Skeleton	(TAIBI; LENARDUZZI; PAHL, 2020)
12	Golden Hammer	(BOGNER et al., 2019)

Each MFE anti-pattern follows the Mini-AntiPattern format (name, problem, and solution) and includes an example to illustrate the problem within an MFE context. We classified the anti-patterns into four categories:

1. **Intra-fronted category**: considers a single MFE component and its design.

2. **Inter-frontend category**: considers the structural division and communication involving two or more MFEs.

3. **Operations category**: related to the operational practices and continuous maintenance of the application.

4. **Development category**: related to the development team and their decisions around the architecture.

Each following subsection presents an anti-pattern, including it category, problem, example, and solution.

3.2.1 Cyclic Dependency

Category: Inter-frontends

Problem: Two or more MFEs directly or indirectly depend on each other, resulting in high coupling between screens and fragments, compromising MFEs' independence and modularity. Thus, changes in one MFE require coordination with the others. Circular dependencies lead to challenges in a system's maintenance and evolution, compromising agility and the ability to scale developments efficiently.

Example: Consider an e-commerce platform with a payment screen implemented in one MFE. This screen contains a fragment from another MFE used to calculate the shipping cost. When the screen displays changes to the items in the cart, the fragment redoes the delivery calculation, which updates the total purchase amount displayed on the screen. This exchange of information between the screen and the fragment results in high coupling. It is essential to assess whether the payment screen and shipping calculation belong to the same domain and if their implementation is part of the same MFE.

Solution: High coupling between two or more MFEs indicates they should compose a single MFE. Thus, it is necessary to review the definition of MFE boundaries and ensure they align with the application domains. A careful analysis of the domains

and reassessing the functional division of the MFEs can help reduce coupling, improve modularity, and increase the independence between components.

3.2.2 Knot Micro Frontend

Category: Inter-frontends

Problem: A Knot is a node composed of three or more MFEs whose communication with each other has a low level of abstraction, exposing the specific details of each MFE. The navigation or data exchange between screens and fragments strongly depends on each MFE context. The problem worsens as the addition of new MFEs to the node occurs without implementing a standardized communication interface. This results in a strongly coupled node, making it difficult to maintain and deploy new functionalities.

Example: Suppose an e-commerce system has MFEs for Digital Products (mfe-digital-products) and Payments (mfe-payments). The product details screen of mfe-digital-products navigates to the payment screen of mfe-payments, passing the product data as a parameter. At a later stage, a Physical Products MFE (mfe-physical-products) is implemented, including screens like delivery tracking, address listing, and address registration. For the unpaid physical products, a modification in the mfe-payments payment screen happens for receiving data of either digital or physical products. Later, adding new product types requires constantly adjusting the payment screen of mfe-payments to display the data for these products. Implementing a communication interface with the following fields would allow the addition of new products without requiring adaptations in mfe-payments: $imageUrl : string; title : string; description : string[], shippingData : ...|null$. This interface definition allows the transmission of product details in a uniform format for all types of products. Furthermore, it permits the optional transmission of delivery information and specifies whether delivery information can be provided.

Solution: A practical solution is implementing communication interfaces between MFEs, allowing an MFE to know what is necessary to integrate with another while abstracting the specific details of its data or implementations.

3.2.3 Hub-like Dependency

Category: Inter-frontends

Problem: A screen of an MFE integrates fragments from several other MFEs, becoming a central point of interdependence. Any issue occurring in the main screen or one of its fragments can affect all other fragments present on it.

Example: Consider a digital banking system where the main screen is an MFE that integrates several fragments from other MFEs, such as an investment list, a chart showing bitcoin value variations, account balance, and credit card statement amount. If any critical issue happens on the main screen, all its functionalities become inaccessible to the end-user

Solution: Avoiding screens that serve as a starting point for other functionalities is recommended. When it is not possible to avoid it, we recommend implementing a resilient screen, incorporating error-handling procedures for all MFE fragments, along with a fallback mechanism. The fallback mechanism should allow access to system functionalities in case of any critical issue with the main screen.

3.2.4 Nano Frontend

Category: Intra-frontends

Problem: The front end decomposes into numerous small MFEs with few screens or fragments. Small MFEs do not justify the cost of their maintenance. Furthermore, the presence of nano frontends can lead to issues of high coupling and the manifestation of other anti-patterns, such as cyclic dependency

Example: In an e-commerce setting, separate MFEs implement the product listing and product details screens. Since both are part of the product context, their implementation should happen within a single MFE encompassing all product screens.

Solution: The issue of nano frontends arises when the definition of boundaries is inadequately and excessively granular. Adhering to Domain-driven Design (DDD) principles is necessary to ensure an effective decomposition of MFEs. So, redesigning the architecture by grouping MFEs with the same domain is necessary. It promotes a

cohesive structure aligned with the business requirements.

3.2.5 Mega Frontend

Category: Intra-frontends

Problem: Decomposing the architecture into a few MFEs encompassing numerous screens and fragments manifest this anti pattern. The MFE inherits the challenges of a monolithic frontend, such as difficulties in testing, slow builds and deployments, high coupling between its components, lack of modularity, and limited scalability.

Example: An e-commerce system is decomposed into just two MFEs, with mfe-users related to users and mfe-shopping related to products and purchases. The latter MFE includes screens that display product listings, product details, purchase confirmations, and purchase history. Decomposing the mfe-shopping into at least two MFEs is necessary: one containing the product listing and product details screens, belonging to the product domain; and another containing the confirmation and purchase history screens, belonging to the purchase domain.

Solution: Reevaluate the architecture and divide the MFEs into granular units, separating functionalities into smaller and specialized MFEs. This approach aids in reducing complexity, enhancing maintainability, and fostering a modular and scalable architecture.

3.2.6 Micro Frontend Greedy

Category: Intra-frontends

Problem: When a developer is uncertain about creating a new MFE, the common practice is to opt for its creation. Whenever a need arises to develop a new set of screens or fragments, a new MFE is instantiated. It can lead to the creation of nano frontends or mega frontends.

Example: Within a banking application, an MFE encompasses screens for security validation, utilizing confirmation code submission via email. Subsequently, the

need arose to implement a new validation method, now employing facial recognition. The screens in this new flow differ from those in the previous flow, resulting in its implementation through a new MFE. Creating a new MFE might not be advisable, as two MFEs have the same context and functionalities.

Solution: To assess whether a new screen or fragment can integrate an existing MFE, a comprehensive review of all existing MFEs is essential. It may avoid unnecessary MFE proliferation, promoting a cohesive and sustainable architecture.

3.2.7 No CI/CD

Category: Operation

Problem: The company lacks an automated Continuous Integration (CI) and Continuous Delivery (CD) pipeline, so developers must manually execute tests and perform deployments. This manual process becomes burdensome, especially with the potential existence of multiple MFEs. It increases development time, reduces productivity, and raises the risk of errors in the production environment.

Example: Upon releasing a new system version, a developer must conduct manual tests and ensure all unit tests pass. However, developers may skip the tests and manually deploy the changes without realizing some tests are failing, introducing bugs, which is avoidable with an automated CI pipeline. Even if the tests pass, there is still a risk of making mistakes during deployment, which could render the system unavailable. Automating the deployment process with CD ensures correct and consistent execution.

Solution: Implement an automated and replicable CI/CD process that extends for new MFEs, ensuring they will have automated test execution and deployment consistently and efficiently.

3.2.8 No Versioning

Category: Operation

Problem: The MFEs are not semantically versioned. Small and large changes

can impact the integration between different MFEs and cause errors. Consequently, the MFEs become less independent, requiring coordinated deployments.

Example: Consider a payment confirmation page with a fragment for calculating shipping costs. Whenever the user inputs shipping information into the fragment, the system generates a delivery charge and adds it to the total purchase amount displayed on the screen. Suppose the delivery charge's return value format changes and the fragment is not versioned. The delivery charge will not be added to the total purchase amount, potentially resulting in a display error or even mistakenly free deliveries. However, if the fragment is versioned, the screen will not be affected by the format change, as it will continue to use the previous version of the fragment and can be updated later when necessary.

Solution: It is essential to adopt the Semantic Versioning standard for versioning MFEs, where the developer must assign a Major version when changes introduce incompatibilities, create a Minor version for new functionalities that do not cause incompatibilities, and apply a Patch version for bug fixes that do not introduce incompatibilities. The versioning practice ensures that changes do not impact functioning versions. Consequently, coordinated deployments are unnecessary, as other MFEs can update their versions as needed.

3.2.9 Lack of skeleton

Category: Operation

Problem: No skeleton or predefined boilerplate is available as a base for creating new MFEs, which leads to the creation of MFEs from scratch or duplicating an existing MFE. The consequences include wasted time, increased risk of errors, duplicated code across MFEs, and a need for more standardization in development.

Example: At the beginning of a specific system development, the developers create an MFE from scratch without adhering to a specific pattern. The second MFE is developed by copying files and code blocks from the first, changing specific parts. The exact process happens when creating new MFEs. Then, a diverse set of MFEs emerges,

which hampers the establishment of automated pipelines, fosters code duplication, and complicates developer interchange between teams.

Solution: Create a repository containing the necessary base code to build an MFE called boilerplate, which includes all the required libraries for the MFE's operation, adhere to the code standards established by the team, and have a file with commonly used commands for developers. It is also crucial to include comprehensive documentation detailing the entire process of creating a new MFE, providing instructions on how to add automated CI/CD, integrate the MFE into the existing system, and other relevant aspects.

3.2.10 Common Ownership

Category: Development

Problem: A single team is responsible for managing all MFEs. It happens when there is no team division or when they are divided based on technical aspects such as data, front-end, and back-end teams. The team does not leverage the benefits of having independent teams provided by the MFE Architecture.

Example: A small company chooses to adopt the MFE architecture. Due to the insufficient number of developers, it is not feasible to create teams by domain. So, developers compose a single team responsible for all MFEs. In this scenario, the cost of maintaining different micro frontends is not justified and is only an additional challenge for the development team.

Solution: Context should be the defining factor when structuring development teams. Therefore, defining the boundaries of teams and MFEs is essential according to Domain-driven Design (DDD), so a team will be responsible only for MFEs within its domain.

3.2.11 Golden Hammer

Category: Development

Problem: All MFEs utilize the same technology, even if it does not meet the specific needs of each MFE. It happens due to developers' familiarity with only one specific technology. This approach limits the architecture, failing to take advantage of the benefits of the possibility of a heterogeneous architecture, which is one of the main attractions of adopting MFEs

Example: A web application contains MFEs implemented using ReactJS framework with Client-side Rendering, even those encompassing essential pages such as the landing page. This technological uniformity overlooks the necessity for Search Engine Optimization (SEO) strategies to ensure better rankings on search engines like Google. It would be advisable to utilize ReactJS with Server-side Rendering or employ a static rendering framework such as NextJS, enabling better optimization for search engines.

Solution: To choose the most suitable technology that addresses the specific challenges of each MFE, which includes adopting the correct programming languages, frameworks, and libraries during its development.

3.2.12 Micro Frontend as the goal

Category: Development

Problem: Adopting the MFE architecture in inappropriate contexts can lead to more issues than benefits, especially in systems with few screens and low complexity or in companies lacking a sufficient number of developers to create dedicated teams for different application domains. In such situations, the maintenance costs of the architecture may outweigh the expected benefits, making its implementation unfeasible.

Example: A personal notes application is divided into the notes and user domains, each comprising its own MFE. The notes domain contains functionalities for note management, containing operations such as listing, creating, editing, and deleting notes. The user's domain encompasses login, registration, and profile management functionalities. In this context, using MFEs results in unnecessary maintenance and development challenges due to the low volume of screens and the low probability of increasing complexity in the application. Adopting a monolithic frontend is a suitable

option.

Solution: Software teams must consider carefully different aspects of adopting MFE architecture. Considering the system's complexity, the feasibility of maintaining automated CI/CD pipelines and the team's restructuring according to different domains is necessary.

4

Catalog Improvement Based on Practitioner's Feedback

After proposing the catalog, we need to validate whether the problems occur in MFE architectures and the solutions address them effectively. Therefore, this chapter presents the results of the Personal Opinion Survey we conducted to gather feedback from 20 industry practitioners with experience in MFE development on the anti-patterns proposed in Chapter 3. Section 4.1 outlines the study design. Section 4.2 presents the quantitative and qualitative results. Section 4.3 discusses the interpretation of the results. Section 4.4 addresses the threats to validity. Section 4.5 presents some of the improved anti-patterns refined based on practitioners' feedback. Finally, Section 4.6 introduces the web application we developed to showcase the anti-patterns.

4.1 Study Design

Figure 8 illustrates the process we followed to refine the MFE anti-patterns based on practitioners' feedback. To evaluate the proposed anti-patterns, we conducted a Personal Opinion Survey (KITCHENHAM; BUDGEN; BRERETON, 2015) that included open and closed questions. The intended audience for the survey included software industry practitioners with experience in MFE development. This approach allowed the validation of different aspects of the anti-patterns by incorporating the insights of

practitioners specialized in the MFE field. We also included a consent form as part of our survey instruments, allowing participants to provide their consent to participate. We assured participants that their participation was voluntary and that they could withdraw at any time. All collected data was anonymized, ensuring privacy and enabling participants to share valuable insights without concern. Our survey comprised three sections.

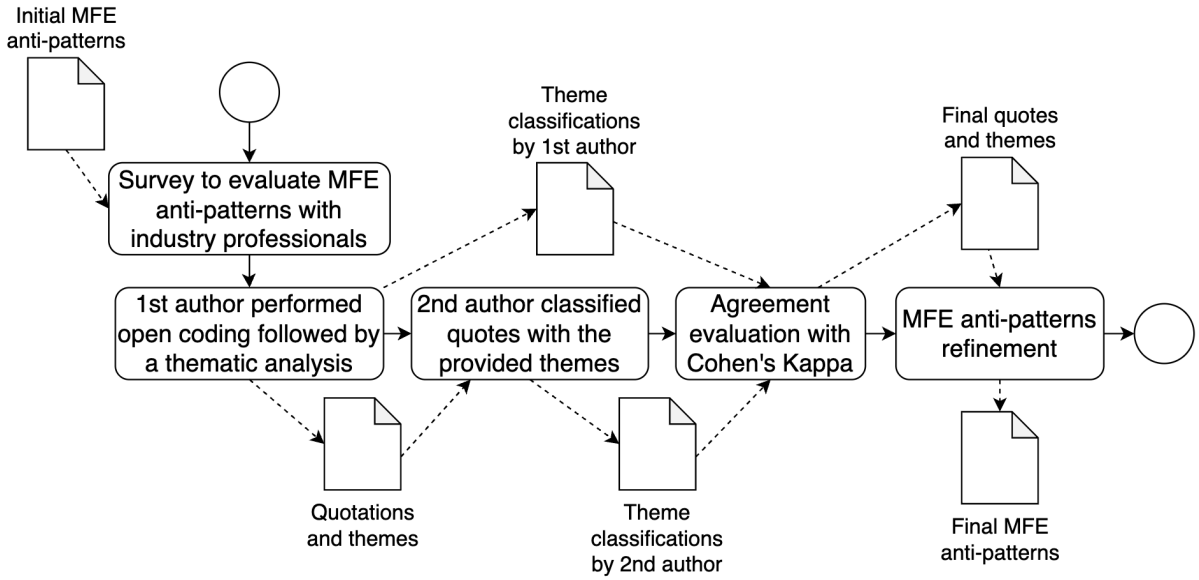

Figure 8 – Process followed to propose and refine the MFE anti-patterns.

The survey's first section aims to understand the level of practitioner experience and the role of participants in the context of MFE development, ensuring that the feedback provided comes from practitioners with practical knowledge in this area. Additionally, the questions allowed the definition of a characterization of each participant. The questions presented below focus on determining whether the practitioners currently work with MFE, their experience in this area quantified in years, and their role in MFE-related projects.

- Are you currently working with Micro Frontends?
- How much practitioner experience do you have with Micro Frontends?
- What type of role do you play or have you played when working with Micro Frontends?

The survey's second section aimed to validate each of the 12 anti-patterns proposed in the catalog. This involved analyzing each anti-pattern through a series of questions designed to: (1) verify whether the descriptions of the problems and proposed solutions were clear and understandable to participants; (2) determine if the proposed solution effectively addresses the problem described in the anti-pattern, and if not, gather alternative solutions; (3) identify the practical occurrence of the anti-pattern in participants' projects; and (4) assess participants' perception of the anti-pattern's impact on MFE projects by assigning a harmfulness value. To define the harmfulness value of each anti-pattern, we used a 10-point Likert scale based on Taibi, Lenarduzzi & Pahl (2020), where 1 means "Not harmful" and 10 indicates "Extremely harmful". Each page of this section included the following questions to evaluate each anti-pattern individually:

- Is the anti-pattern problem clearly stated?
- If you disagree, please provide a description of what is not clear
- Is the anti-pattern solution clearly stated?
- If you disagree, please provide a description of what is not clear
- Does the proposed solution address the problem presented in the anti-pattern?
- In case you have a different suggestion on the problem solution, please provide a description
- Have you ever encountered this problem in any project you have previously worked on or are currently working on?
- How harmful do you think this anti-pattern problem is?

The third section of the survey enabled participants to offer insights based on their practical experience, providing additional feedback and suggestions for improving the catalog. The questions were designed to identify any MFE-related issues not covered in the catalog, assess the potential impact of the catalog on the quality of MFE solutions, and gather recommendations for enhancing the catalog:

- Based on your experience, is there any issue related to Micro Frontends not covered in the presented anti-patterns?
- If you do agree, please provide a description of the problem.
- How do you think this catalog would help improve the quality of micro frontend architecture in your work?
- Do you have any suggestions for improving the anti-patterns catalog?

After collecting participants' responses to the survey, we conducted a quantitative analysis to identify the most common anti-patterns and calculated the median harmfulness score for each. To further investigate the harmfulness scores, we employed several statistical tests. First, we used the Shapiro-Wilk test (SHAPIRO; WILK, 1965) to determine whether the score samples for each anti-pattern were normally distributed. Since only half of the samples followed a normal distribution, we opted for non-parametric tests. As the same participants rated different anti-patterns, we treated the samples as dependent and selected the Friedman test (FRIEDMAN, 1937), which is suitable for dependent samples. We applied the av post-hoc Dunn test (DUNN, 1964) to compare pairs of medians, as it is commonly used following Friedman test when significant differences are found.

For the qualitative feedback, we performed a thematic analysis grounded in coding reliability (BRAUN; CLARKE, 2021). Thematic analysis allows for organizing and categorizing feedback in a structured manner, which helps in identifying recurring patterns (themes). This approach aids in understanding participants' perspectives better and directing improvements based on the specific objectives of each piece of feedback. We assessed intercoder reliability (ICR) between two researchers to ensure the robustness of the thematic analysis. The second researcher was included to evaluate the consistency and objectivity of the coding process. The first researcher conducted an initial open coding of all responses, resulting in the definition of eight themes. Subsequently, the second researcher independently applied these themes to the same quotations. To measure agreement between the researchers, we calculated Cohen's Kappa (COHEN, 1960) and resolved any discrepancies through discussion to reach

a consensus. We used practitioners' feedback to refine the catalog and finalize the anti-patterns.

4.2 Results

In this section, we present the analysis of the results obtained from the survey responses. Subsection 4.2.1 outlines the participants' characterization; Section 4.2.2 summarizes the quantitative results; and Subsection 4.2.3 presents the thematic analysis findings identified from the collected qualitative data.

4.2.1 Participants' Characterization

We directly contacted a total of 37 practitioners to participate in the survey. However, due to the open-access nature of the survey link, precise control over participant recruitment was not feasible. As a consequence, we could not determine an exact response rate (RALPH et al., 2020). A total of 20 practitioners volunteered to participate in the survey. More than half of the participants work at large companies with over 90 software engineers and projects involving 40+ MFEs and 70+ MS, with engineering teams structured as independent units. The remaining participants work on other software companies, but in smaller projects. This diversity ensures the identified anti-patterns are applicable across different scales and organizational structures. Before responding, each participant signed a consent form, ensuring the confidentiality of the information.

Table 4 presents a summary of participants' characterization. Each column presents the answer of one of the characterization question: column "Working with MFE" refers to "Are you currently working with Micro Frontends?", column "Experience with MFE" refers to "How much professional experience do you have with Micro Frontends?" and column "Role" refers to "What type of role do you play or have you played when working with Micro Frontends?". We assigned each participant an identifier ranging from P1 to P20.

Figure 9 presents a pie chart illustrating responses to the question "Are you

Table 4 – Summary of the survey participants' characterization.

ID	Working with MFE	Experience with MFE	Role
P1	No	More than 2 years	Fullstack Developer
P2	No	Between 1 and 2 years	Software Architect
P3	Yes	More than 2 years	Software Architect
P4	Yes	More than 2 years	Frontend Developer
P5	Yes	More than 2 years	Fullstack Developer
P6	Yes	More than 2 years	Frontend Developer
P7	No	More than 2 years	Fullstack Developer
P8	Yes	More than 2 years	Software Engineering Team Leader
P9	No	Between 1 and 2 years	Fullstack Developer
P10	Yes	More than 2 years	Fullstack Developer
P11	Yes	More than 2 years	Frontend Developer
P12	Yes	More than 2 years	Fullstack Developer
P13	Yes	Less than 1 year	Fullstack Developer
P14	Yes	More than 2 years	Frontend Developer
P15	Yes	More than 2 years	Mobile
P16	Yes	More than 2 years	Frontend Developer
P17	Yes	More than 2 years	Frontend Developer
P18	Yes	Between 1 and 2 years	Fullstack Developer
P19	No	Between 1 and 2 years	Fullstack Developer
P20	Yes	More than 2 years	Frontend Developer

currently working with Micro Frontends?". The results show that 78.9% of the participants are currently involved in MFE software projects. Figure 10 presents a pie chart illustrating responses to the question "How much professional experience do you have with Micro Frontends?". The results show that 73.7% of the participants have more than 2 years of experiente and 21.1% have between 1 and 2 years of experience. These results show that participants have a solid knowledge and practical experience with MFEs, making them capable of evaluating the proposed anti-patterns.

Figure 11 presents a bar chart presenting the responses to the question "What type of role do you play or have you played when working with Micro Frontends?". The results indicate that 8 participants are frontend or mobile developers and 9 are full stack developers. Additionally, 3 hold roles such as project leaders or software architects. This distribution suggests that the anti-patterns were primarily evaluated from a developer's perspective, but also reflect insights from non-developer roles.

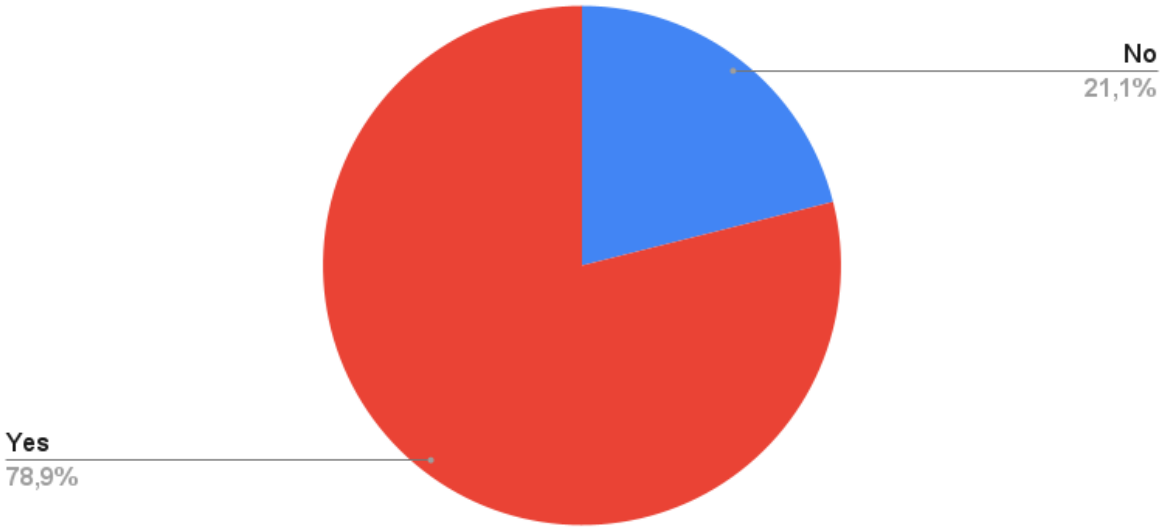

Figure 9 – Pie chart illustrating responses to the question: "Are you currently working with Micro Frontends?".

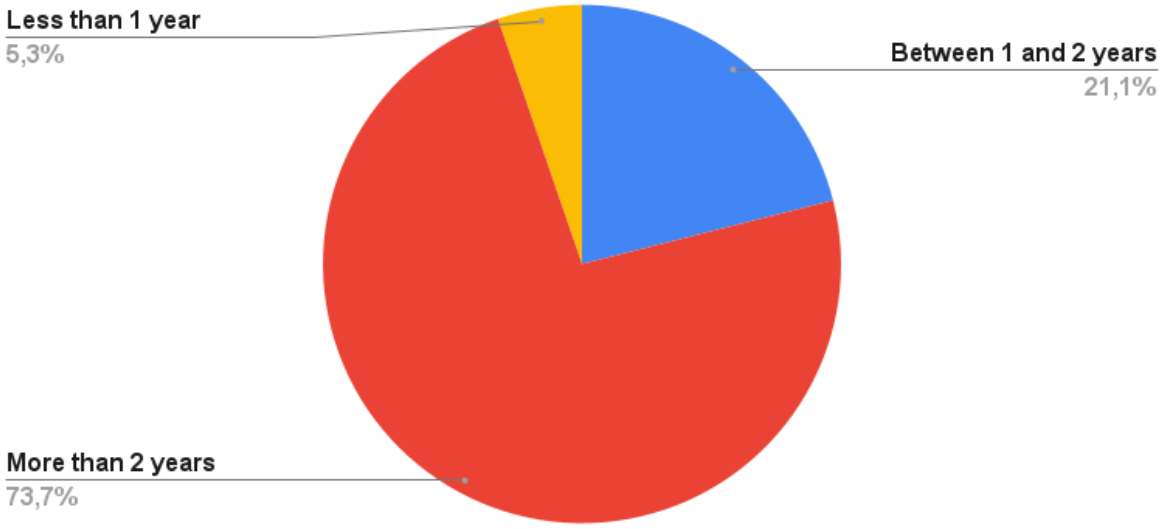

Figure 10 – Pie chart illustrating responses to the question: "How much professional experience do you have with Micro Frontends?".

4.2.2 Quantitative Analysis

Table 5 summarizes our quantitative results, which we ranked according to the harmful score values. Column 1 presents the titles of the anti-patterns evaluated in the survey. Columns 2 and 3 report the participants' agreement rates regarding the clarity of the problem presentation and the proposed solutions of the anti-patterns. Column 4 presents the participants' agreement rates regarding the effectiveness of the proposed solutions

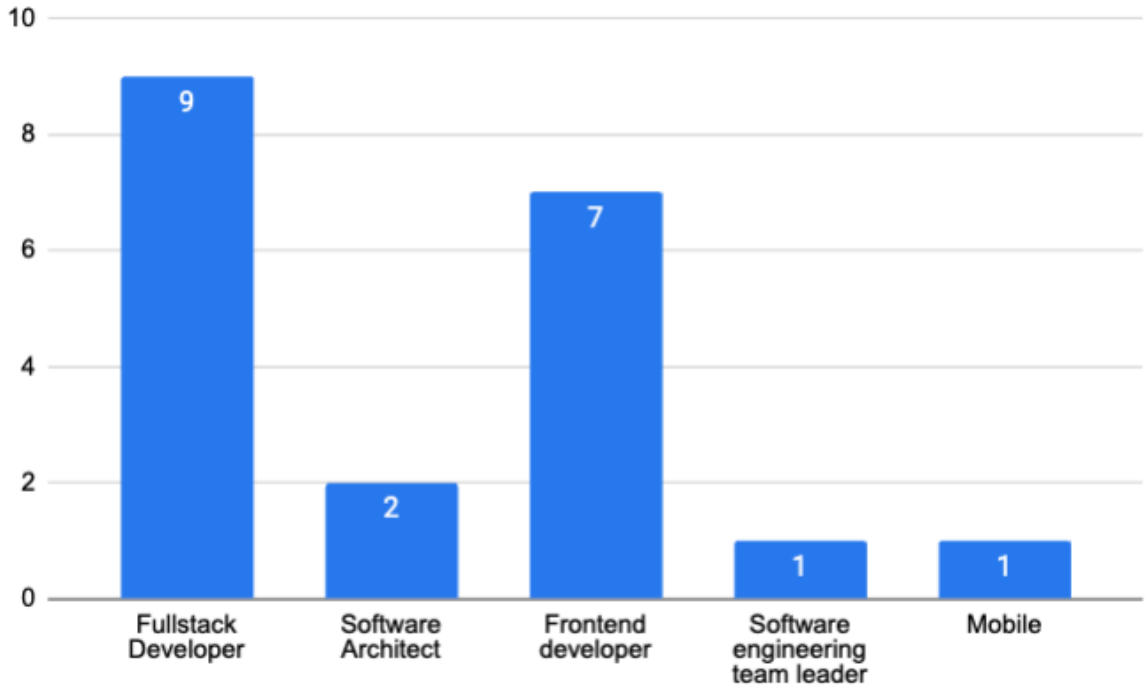

Figure 11 – Bar chart presenting the responses to the question: "What type of role do you play or have you played when working with Micro Frontends?".

for the problems identified in the anti-patterns. Column 5 shows the percentage of participants who have encountered the problems described in the anti-patterns in their practitioner experience within the software industry. Column 6 presents the median values of harmfulness attributed to the anti-patterns by the participants.

Regarding the clarity of the problems and solutions presented in the anti-patterns, the results show that the participants had a thorough understanding, as demonstrated by the high values in Columns 2 and 3, which range from 95% up to 100%. Moreover, the high values presented in Column 3 indicate a positive efficacy of the solutions, on which a large majority of participants recognized the proposed solutions as effective means of addressing the issues posed by the anti-patterns.

The values in column 5 indicate that practitioners frequently encounter the anti-patterns Cyclic Dependency, Knot Frontend, Hub-like Dependency, Mega Frontend and No CI/CD in software projects. Notably, seven of the remaining eight anti-patterns were reported by more than 50% of participants, highlighting their common occurrence as well. The only anti-pattern observed by less than 50% of participants is the Nano Frontend, in contrast to the Mega Frontend. This suggests that practitioners are more

Table 5 – Overall results from quantitative analysis ranked by harmfulness score.

Anti-pattern	**Problem clearly stated rate**	**Solution clearly stated rate**	**Solution addresses the problem rate**	**Seen in practice rate**	**Harmfulness**
No CI/CD	100.00%	100.00%	100.00%	90.00%	10
No Versioning	100.00%	100.00%	95.00%	70.00%	9
Common Ownership	95.00%	95.00%	100.00%	55.00%	8
Cyclic Dependency	95.00%	95.00%	100.00%	85.00%	8
Hub-like Dependency	95.00%	90.00%	95.00%	95.00%	8
Knot Micro Frontend	95.00%	95.00%	100.00%	80.00%	8
Lack of Skeleton	100.00%	100.00%	100.00%	65.00%	8
Micro Frontend as the Goal	100.00%	100.00%	100.00%	60.00%	8
Mega Frontend	100.00%	100.00%	100.00%	90.00%	7
Micro Frontends Greedy	95.00%	100.00%	100.00%	55.00%	7
Nano Frontend	100.00%	100.00%	100.00%	35.00%	7
Golden Hammer	95.00%	100.00%	100.00%	70.00%	6

likely to create larger MFEs than smaller ones.

Given the limited sample size, we assessed the reliability of the harmfulness scores using several statistical tests. First, we applied the Shapiro-Wilk Test with a 95% confidence level to determine if the samples were normally distributed. The test indicated that only half of the anti-patterns (namely Cyclic Dependency, Knot Micro Frontend, Nano Frontend, Mega Frontend and Golden Hammer) follow a normal distribution.

We then used the Friedman test to evaluate whether there were statistically significant differences in the harmfulness scores. The Friedman test yielded a $p - value$ of 0.0000001581 (< 0.05), which strongly suggests that there are statistically significant differences in harmfulness scores among the anti-patterns. Subsequently, the post-hoc Dunn test revealed that only two anti-patterns exhibited statistically significant differences compared to others: Golden Hammer differed from Hub-like Dependency,

No CI/CD, Lack of Skeleton and Common Ownership; and Micro Frontend as the Goal differed from Hub-like Dependency.

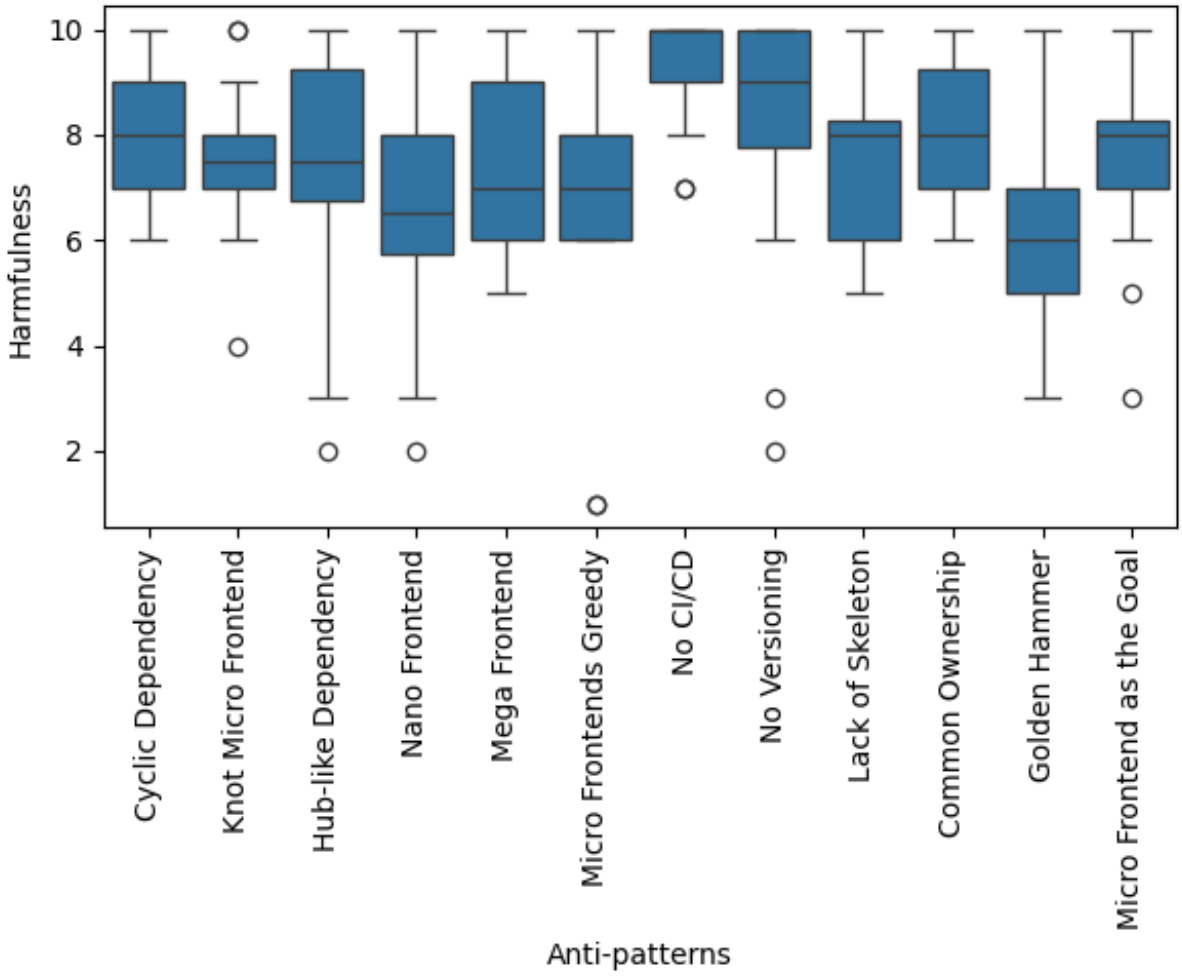

Figure 12 – Boxplot illustrating the harmfulness ratings for each anti-pattern.

While Dunn's test did not indicate significant differences between most pairs, the boxplot visualization (Figure 12) reveals notable trends. No CI/CD is perceived as significantly more harmful than other anti-patterns, as evidenced by its higher median score. Conversely, Golden Hammer is considered the least harmful, with the lowest median harmfulness score (6). These findings suggest that while statistical significance was not universally achieved, there is a strong perception that the absence of CI/CD is particularly harmful, whereas the reuse of familiar technologies is seen as acceptable.

4.2.3 Thematic Analysis

Based on the open coding quotations, the author of this Thesis defined eight themes (Figure 13). Subsequently, a research collaborator independently categorized the same quotations using the established themes. On measuring the ICR, we obtained a Cohen's Kappa with a value of 0.84. According to Landis & Koch (1977), this is considered an almost perfect score, highlighting the reliability of our coding process and the robustness of the identified themes. In the following paragraphs, we present a brief explanation of each category and its main findings. The complete thematic analysis results can be found in Appendix A.

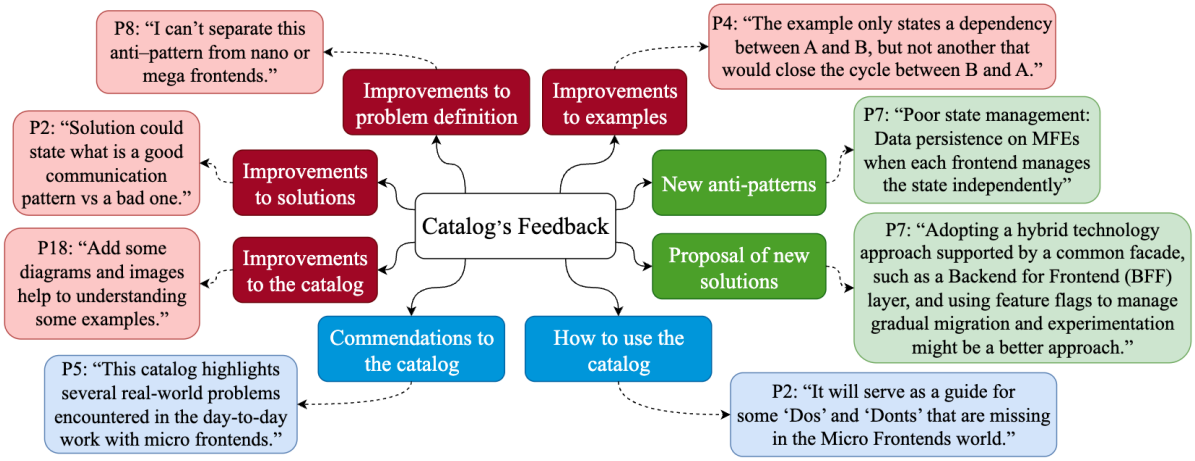

Figure 13 – Resulting themes and meta-themes from the Thematic Analysis on practitioners feedback.

Commendations to the catalog – *compliments to the catalog*: P2 acknowledged the novelty of the anti-patterns due to the lack of ones that focus on the MFE development, as stated in "The term and technology is fairly new, and such patterns are not well established in the software community yet." P5 praised the catalog for addressing practical issues that practitioners face when working with MFE, as commented in "This catalog highlights several real-world problems encountered in the day-to-day work with micro frontends."

How to use the catalog – *how the catalog can be used to improve the development and maintenance of MFE architectures*: P2 highlighted that the catalog may provide essential guidance for best practices and avoidables in MFE development, as stated in "It will serve as a guide for some 'Dos' and 'Donts' that are missing in the Micro Frontends

world." P7 commented on the practical utility of the catalog in educating and integrating new team members in MFE software projects, as stated in "Training new team members and onboarding them to micro frontend projects." P9 observed that the catalog may help in whether to use or not MFE architecture in "think about architecture decisions and the decision to use micro frontends architecture or not."

Improvements to examples – *proposals for new examples or enhancements to the presented ones*: For Cyclic Dependency, P4 suggests that the example should include all necessary dependencies to fully close the cycle, as noted, "The example only states a dependency between A and B, but not another that would close the cycle between B and A." For Hub-like Dependency, P4 provided a practical scenario that emphasizes the importance of robust error handling in "A main banking screen that has charts, lists, and balances. If this screen implements its own data fetch function that fails and renders the screen useless, then it is a problem."

Improvements to problem definition – *enhancements to anti-patterns problems definitions*: P8 suggested the need for a clearer definition in Micro Frontends Greedy, once they had difficulty in distinguishing it from the Nano frontend or Mega frontends anti-patterns, as stated in "I can't separate this anti-pattern from nano or mega frontends." For Common Ownership, P15 pointed out that small teams can also benefit from characteristics inherent of software modularization, as mentioned in "Naturally, larger software involves more people, but I believe small teams can also benefit from software modularization, such as separation of layers and responsibilities, observability and maintainability."

Improvements to solutions – *enhancements to anti-patterns solutions*: P2 suggested that the Knot Micro frontend anti-pattern should clearly define the difference between a good communication pattern and a bad one, as stated in "Solution could state what is a good communication pattern vs a bad one." For Hub-like Dependency, many participants suggested that the solution of avoiding aggregator screens does not address the problem correctly because, in the context of MFE architecture, screens will include many MFE fragments, as P2 emphasized in "It is inevitable to have aggregators," and P20 in "But not use it goes against the main idea of the MFE to be contextually segregated." P5

emphasized that the solution for Nano Frontend will rely on the organizational context it is inserted in, as stated, "The solution here will depend entirely on the organizational context."

Improvements to the catalog – *enhancements to the catalog as a whole*: Most of the participants focused their suggestions for improving the catalog adding visual aids for enhancing the readability of the catalog, as P4 suggests in "I would use some flow charts to exemplify most of the anti-patterns and make it more readable," and P18 complements in "Add some diagrams and images help to understand some examples."

New anti-patterns – *proposals of new anti-patterns*: P2 discussed the importance of choosing between build-time and runtime integration based on team and user needs, as stated in "There are mainly two ways to integrate micro frontends: build-time and runtime. The decision on which to adhere reflects deeply in the teams' and users' needs more than the technical pros and cons each of them offer." P7 highlighted that data persistence on MFEs may become an anti—pattern due to the issue when different MFE manage the state, as commented, "Poor state management: Data persistence on MFEs when each frontend manages the state independently." Lastly, P8 suggests that Inconsistent User Experience is an existing issue in MFE.

Proposal of new solutions – *proposals of different solutions for specific anti-patterns*: For Mega Frontend, P4 suggested that better discussions between the product team and the development team could help define when features should be treated as different products, as stated in "Lack of communication between the product team and the development team. It should be well discussed between the two teams to define when two or more features are different products." For Golden Hammer, P7 proposed adopting hybrid technology approaches for solving the problem, as commented in "Adopting a hybrid technology approach supported by a common facade, such as a Backend for Frontend (BFF) layer, and using feature flags to manage gradual migration and experimentation might be a better approach."

4.3 Discussion

The proposed MFE anti-patterns are closely aligned with their MS counterparts, reflecting the inherent similarities between these architectural styles. Given the frequent evolution of software systems from monolithic architectures to MS and subsequently to MFE, anti-patterns related to development and operation like No CI/CD and Common Ownership are likely to persist in MFE if they were previously encountered in MS.

We observed that the proposed anti-patterns have varying impacts on developers and end users. The No Versioning and Hub-like Dependency anti-patterns significantly affect end users, potentially causing application crashes. The Golden Hammer anti-pattern has a moderate impact on both end users and developers, stemming from poor experiences due to inappropriate technology choices. The remaining anti-patterns primarily impact developers, complicating architecture maintenance and evolution, though they have a low direct impact on end users.

By accessing our online catalog, developers can learn how to avoid bad practices when working with MFE from an organizational and architectural point of view. The catalog can act as a checklist or as a management resource for experienced or new members of software projects. Moreover, the catalog may also help practitioners recognize bad practices that have become standardized within their organizations due to their routine use and familiarity. Anecdotal evidence suggests that the catalog is already valuable to developers. For instance, some survey participants reported using it in their daily work, as P12 mentioned: "Every time I have to implement a new feature on a micro frontend, I consult the catalog to remember the anti-patterns." This feedback highlights the catalog's role in enhancing development practices and fostering awareness of best practices in MFE design.

While this study has revealed a strong correlation between MS and MFE anti-patterns, there remain specific anti-patterns unique to MFE architectures that warrant further exploration. Issues related to UI inconsistency, the management of the state through global versus local stores, and the selection of inappropriate composition approaches have not yet been addressed by the proposed anti-patterns. Future research should focus on identifying and mitigating these and other MFE-specific anti-patterns

to enhance the overall quality and effectiveness of MFE architectures.

To ensure high-quality system design and prevent software degradation, it is crucial to identify anti-patterns early and perform the necessary refactoring. Automating the detection of these anti-patterns may allow for early intervention, thus significantly mitigating their impact on software projects. Therefore, future research should also prioritize the development of automated detection methods tailored to MFE architectures. Such advancements will not only improve system quality but also help prevent long-term negative consequences.

Lastly, it is also important to address new anti-patterns that occur during software development. As presented in the thematic analysis result, participants proposed new anti-patterns related to Inconsistent User Experience, Fragmented State Management, Complex Inter-MFE Communication, Overhead of Independent Deployments, Security and Authentication Challenges, Performance Bottlenecks, Security Risks and Observability. These anti-patterns' problems and solutions must be clearly defined and validated by practitioners. It highlights the need for ongoing research to address emerging challenges in the MFE field, ensuring the development of efficient MFE architectures.

4.4 Threats to validity

We assessed the Internal, External, Conclusion, and Construct threats to validity according to Wohlin et al. (2012).

Internal Validity: (1) The length of the form used to gather practitioner's feedback. A long form may fatigue the participant, affecting their responses. To address it, we provided an estimated completion time to participants when inviting them and structured the form to present each anti-pattern on a separate page, allowing participants to focus on one anti-pattern at a time. We also included a progress bar to give participants a clear indication of how many anti-patterns remained to be evaluated. (2) Participants representativeness. To address it, we focused on inviting participants with previous background in working with MFE in industry. To ensure that participants

were not biased by the results of previous works, we did not propose a predefined set of bad practices to the participants.

External Validity: Subjects sample size. To address this threat, we distributed the survey to a broader pool of software engineers, encompassing frontend developers, fullstack developers and team leaders. It allowed us to capture a wider range of perspectives from practitioners with varying levels of MFE knowledge and involvement. While the sample size may not be ideal for full population generalization, the diversity of participant roles strengthens the applicability of our findings to real-world MFE development practices.

Conclusion Validity: (1) One might reach incorrect conclusions given the data. On addressing it, multiple researchers were involved in the data interpretation. For both the quantitative and thematic analysis, two researchers handled the data interpretation and categorization of themes, and a third, who is an expert in Software Engineering with more than 20 years of experience, reviewed all the results. (2) The harmfulness scores calculated using the median may not fully capture the nuances of participants' perceptions. Although we do not provide enough evidence on harmfulness, on ranking the anti-patterns we illustrate which of them the practitioners consider the most harmful during software development. We also emphasize that our survey focuses on validating the initial anti-patterns, instead of generating the ranking itself.

Construct Validity: Influence of researcher bias on the qualitative results. To mitigate it, we employed a two-coder approach to the thematic analysis, within the second researcher independently reviewing the qualitative data and conducting a separate thematic analysis. Following this, we employed Cohen's Kappa (COHEN, 1960) to measure the level of agreement between the two coders. The resulting score of 0.84, classified as "excellent agreement," strengthens our confidence in the objectivity and trustworthiness of the identified themes.

4.5 Improved catalog based on practitioners' feedback

After analyzing practitioners' feedback, we refined the anti-patterns and generated an improved version of the catalog. Hub-like Dependency received the most suggestions for improvement, particularly regarding the problem definition, solution, and example, but most of the feedback focused on enhancing the solution. Another anti-pattern that attracted multiple comments was Cyclic Dependency, especially regarding the proposal of new solutions, which led us to improve its solution. Considering the feedback regarding the addition of images, we created one image for the problem and another for the solution of the Cyclic Dependency and Hub-like Dependency anti-patterns.

Other anti-patterns that were refined include Knot Micro Frontend, Nano Frontend, Mega Frontend, No CI/CD, No Versioning, Lack of Skeleton, Common Ownership, Micro Frontend Greedy, and Golden Hammer. The only anti-pattern we did not updated was the Micro Frontend as the goal. In the following subsections we present the new version of all improved anti-patterns.

4.5.1 Cyclic Dependency

Category: Inter-frontends

Problem: Two or more MFEs directly or indirectly depend on each other, resulting in high coupling between screens and fragments, compromising MFEs' independence and modularity (Figure 14). Thus, changes in one MFE require coordination with the others. Circular dependencies lead to challenges in a system's maintenance and evolution, compromising agility and the ability to scale developments efficiently.

Example: Consider an e-commerce application featuring a product details screen implemented on mfe-products. The screen integrates three fragments: one displaying the shopping cart from "mfe-checkout," another showing product recommendations from "mfe-recommender," and a third calculating shipping costs based on the selected delivery address from "mfe-delivery." When a recommended product is added to the cart, "mfe-recommender" notifies "mfe-checkout," which subsequently informs "mfe-delivery" to recalculate the shipping costs. If the shipping address is updated,

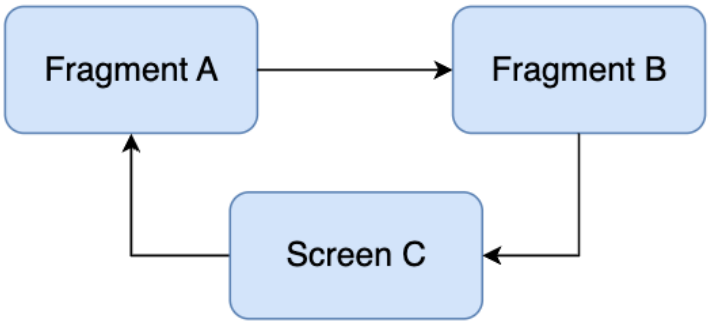

Figure 14 – Cyclic communication between fragments on the same screen.

"mfe-delivery" notifies all other MFEs to verify whether the products they display can be shipped to the new address; if not, those products are disabled. This interaction between the screen and its fragments results in high coupling, where changes or updates in one fragment often necessitate adjustments in others to maintain the overall functionality of the product details screen.

Solution: High coupling between MFEs can be effectively mitigated through event-based communication, which removes the need for direct dependencies between MFEs. Instead, interactions are handled indirectly via a centralized event store. On implementing the Publish-Subscribe (Pub-Sub) pattern, an MFE can publish an event to the browser, allowing other MFEs to subscribe and respond when the event occurs (Figure 15). To ensure consistency and reduce errors, it is recommended to centralize event definitions in a shared library.

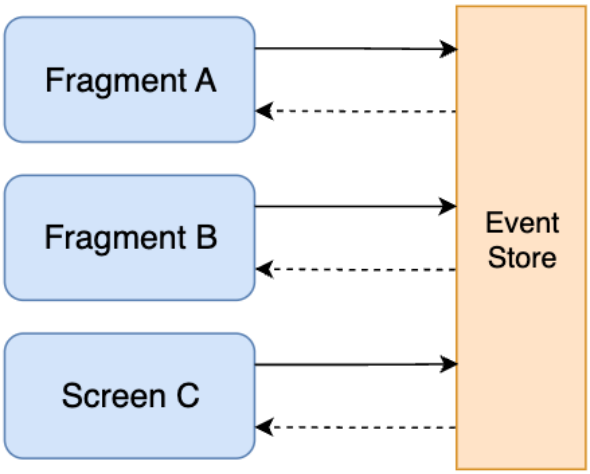

Figure 15 – Communication between fragments using an Event Store.

4.5.2 Knot Micro Frontend

Category: Inter-frontends

Problem: A Knot is composed of three or more MFEs whose communication with each other uses a context-specific interface. This means that navigation and data exchange between screens and fragments heavily depend on the unique context of each MFE involved. Adding new MFEs exacerbates the problem: as the number of MFEs grows, the interface complexity increases due to the introduction of new contexts, creating a highly coupled Knot that becomes difficult to maintain and integrate new functionalities.

Example: Suppose an e-commerce system has MFEs for Digital Products (mfe-digital-products) and Payments (mfe-payments). The product details screen of mfe-digital-products navigates to the payment screen of mfe-payments, passing the product data as a parameter. At a later stage, a Physical Products MFE (mfe-physical-products) is implemented, including screens like delivery tracking, address listing, and address registration. For the unpaid physical products, a modification in the mfe-payments payment screen happens for receiving data of either digital or physical products. Later, adding new product types requires constantly adjusting the payment screen of mfe-payments to display the data for these products. Implementing a communication interface with the following fields would allow the addition of new products without requiring adaptations in mfe-payments. This interface definition must allow the transmission of product details in a uniform format for all types of products. Furthermore, it permits the optional transmission of delivery information and specifies whether delivery information can be provided.

Solution: A practical solution to address the problem of Knots is to implement domain-driven communication interfaces that are both generic and flexible. These interfaces should define a contract based on the domain model, specifying the essential fields required for each MFE to function correctly and interact with others. On designing new fields or attributes, it is essential to ensure their consistency and reusability and minimize tight coupling so other MFEs can utilize them. We recommend including a generic field in the interface containing a list of objects with standard properties such as

label, value, and type, allowing each MFE to display data on the screen without needing to understand the specific meaning or context of the values. This approach reduces coupling between MFEs, while maintaining benefits such as modularity, scalability, and adaptability to new requirements.

4.5.3 Hub-like Dependency

Category: Inter-frontends

Problem: A screen of a MFE integrates fragments from several other MFEs, becoming a central point of interdependence (Figure 16). Any issue occurring in the main screen or one of its fragments can affect all other fragments present on it.

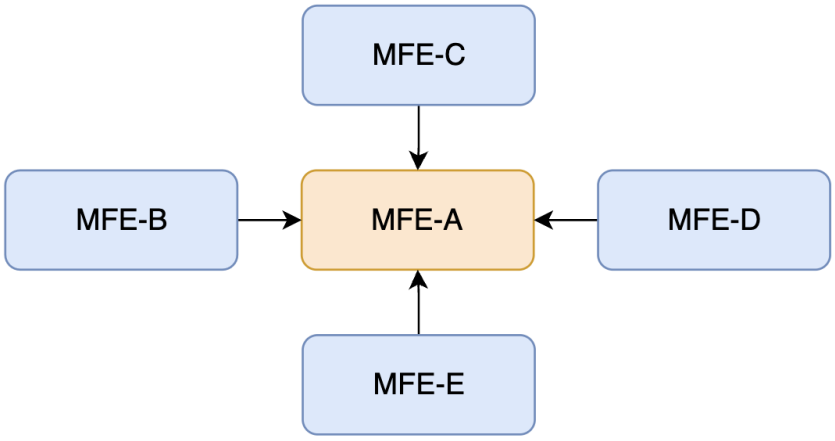

Figure 16 – MFE-A is a central point (Hub) of dependency between the other MFEs.

Example: Consider a digital banking system where the main screen is an MFE that integrates several fragments from other MFEs, such as an investment list, a chart showing bitcoin value variations, account balance, and credit card statement amount. This structure introduces a significant vulnerability: a single faulty fragment can potentially disrupt the entire main screen (Figure 17). Consider a scenario where an issue within the investment list fragment causes it to malfunction. This malfunction could manifest as data display errors, unresponsive controls, or even complete crashes. The consequences of such an incident extend beyond the affected fragment, rendering the entire main screen and other fragments unusable and inaccessible to the user.

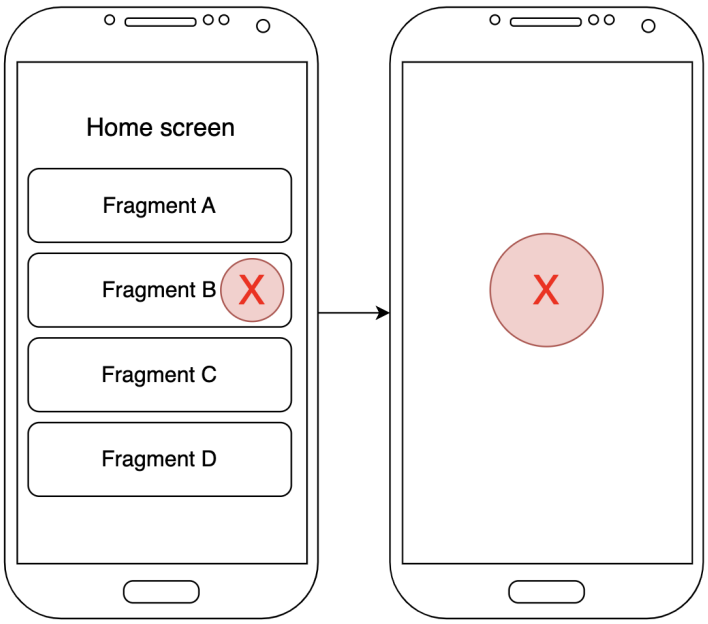

Figure 17 – Home screen is a screen with several fragments, and when Fragment B raises an error, the entire screen becomes unavailable.

Solution: To prevent a single fragment failure from crashing the entire main screen, the screen should be kept as simple as possible, and each fragment should implement robust error handling mechanisms. This can be achieved by implementing a strategy where uncaught errors within a fragment gracefully degrade its functionality, displaying an user-friendly fallback message (Figure 18). This approach ensures that users are informed of the issue without hindering their interaction with the remaining functionalities on the main screen.

4.5.4 Nano Frontend

Category: Intra-frontends

Problem: The frontend decomposes into numerous small MFEs with few screens or fragments. Small MFEs do not justify the cost of their maintenance. Furthermore, the presence of nano frontends can lead to issues of high coupling and the manifestation of other anti-patterns, such as cyclic dependency

Example: In an e-commerce setting, separate MFEs implement the product listing and product details screens. Since both are part of the product context, their

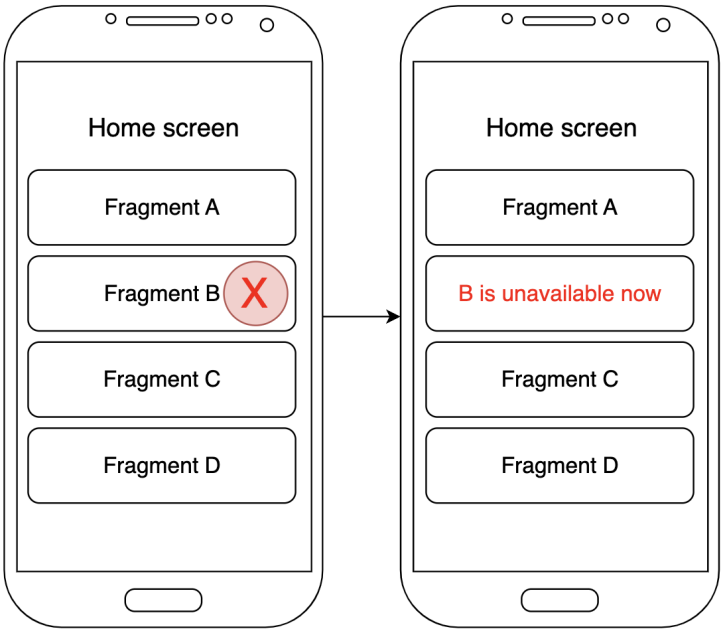

Figure 18 – When Fragment B raises an error, an user-friendly fallback message is rendered so the entire screen remains available.

implementation should happen within a single MFE encompassing all product screens.

Solution: The issue of nano frontends arises when the definition of boundaries is inadequately and excessively granular. Adhering to Domain-driven Design (EVANS, 2004) principles is necessary to ensure an effective decomposition of MFEs. Therefore, the development team must work closely with the product team to gain a deep understanding of the domains and reflect them accurately in the architecture. To solve this issue, the architecture must be redesigned by grouping MFEs with the same domain is necessary. For minor variations within a domain, consider using templates or component libraries. This approach avoids creating a separate MFE for each slight variation, promoting efficiency and code reuse.

4.5.5 Mega Frontend

Category: Intra-frontends

Problem: Decomposing the architecture into a few MFEs encompassing numerous screens and fragments manifest this anti-pattern. The MFE inherits the challenges of a monolithic frontend, such as difficulties in testing, slow builds and deployments,

high coupling between its components, lack of modularity, and limited scalability.

Example: An e-commerce system is decomposed into just two MFEs, with mfe-users related to users and mfe-shopping related to products and purchases. The latter MFE includes screens that display product listings, product details, purchase confirmations, and purchase history. Decomposing the mfe-shopping into at least two MFEs is necessary: one containing the product listing and product details screens, belonging to the product domain; and another containing the confirmation and purchase history screens, belonging to the purchase domain.

Solution: To avoid this problem, the development team must work closely with the product team to gain a deep understanding of the domains and reflect them accurately in the architecture. To fix this issue, the team should reevaluate the architecture and divide the MFEs into more granular units, separating functionalities into smaller and specialized MFEs based on domains. This approach helps reduce complexity, enhance maintainability, and foster a modular and scalable architecture.

4.5.6 Micro Frontend Greedy

Category: Intra-frontends

Problem: When a developer is uncertain about creating a new MFE, the common practice is to opt for its creation. Whenever a need arises to develop a new set of screens or fragments, a new MFE is instantiated. This can lead to an explosion in the number of MFEs, making the system difficult to understand and increasing the likelihood of both nano and mega frontends emerging.

Example: Within a banking application, an MFE encompasses screens for security validation, utilizing confirmation code submission via email. Subsequently, the need arose to implement a new validation method, now employing facial recognition. The screens in this new flow differ from those in the previous flow, resulting in its implementation through a new MFE. Creating a new MFE might not be advisable, as two MFEs have the same context and functionalities.

Solution: To determine where to implement a new feature composed of a set

of screens and/or fragments, the domain of the new feature must first be defined. If it falls within the domain of an existing MFE, it should be implemented there. In this case, a summary of all MFEs, their contexts, and domains can help identify the best fit for the new feature. If it belongs to a brand new domain, one or more MFEs should be defined based on the domain definition. Establishing well-defined domains relies on the collaboration between the development and product teams to accurately define boundaries.

4.5.7 No CI/CD

Category: Operation

Problem: The company lacks an automated Continuous Integration (CI) and Continuous Delivery (CD) pipeline, so developers must manually execute tests and perform deployments. This manual process becomes burdensome, especially with the potential existence of multiple MFEs. It increases development time, reduces productivity, and raises the risk of errors in the production environment.

Example: Upon releasing a new system version, a developer must conduct manual tests and ensure all unit tests pass. However, developers may skip the tests and manually deploy the changes without realizing some tests are failing, introducing bugs, which is avoidable with an automated CI pipeline. Even if the tests pass, there is still a risk of making mistakes during deployment, which could render the system unavailable. Automating the deployment process with CD ensures correct and consistent execution.

Solution: Implement an automated and replicable CI/CD process that extends for new MFEs, ensuring they will have automated test execution and deployment consistently and efficiently. This should be part of the Definition of Done (DoD) of the architecture.

4.5.8 No Versioning

Category: Operation

Problem: The MFEs are not versioned. Small and large changes can impact the integration between different MFEs and cause errors. Consequently, the MFEs become less independent, requiring coordinated deployments.

Example: Consider a payment confirmation page with a fragment for calculating shipping costs. Whenever the user inputs shipping information into the fragment, the system generates a delivery charge and adds it to the total purchase amount displayed on the screen. Suppose the delivery charge's return value format changes and the fragment is not versioned. The delivery charge will not be added to the total purchase amount, potentially resulting in a display error or even mistakenly free deliveries. However, if the fragment is versioned, the screen will not be affected by the format change, as it will continue to use the previous version of the fragment and can be updated later when necessary.

Solution: Adopting a versioning approach like Semantic Versioning is essential to ensure that changes do not impact functioning versions. For example, consider a fragment that is used in screens across different MFEs in a client-side rendering scenario. Without versioning, any change to the fragment's parameters or return values could break the interaction on all the screens it integrates with. However, with versioning, such updates would not impact the current versions used by other MFEs, as they can continue to request the previous version of the fragment and update at their convenience. This approach helps maintain a stable environment and minimizes disruptions caused by updates.

4.5.9 Lack of Skeleton

Category: Operation

Problem: No skeleton or predefined boilerplate is available as a base for creating new MFEs. This leads to the creation of MFEs from scratch or based on an existing MFE and inheriting its issues. The consequences include wasted time, increased risk of errors, duplicated code across MFEs, and a need for more standardization in development.

Example: At the beginning of a specific system development, the developers

create an MFE from scratch without adhering to a specific pattern. The second MFE is developed by copying files and code blocks from the first, changing specific parts. The exact process happens when creating new MFEs. Then, a diverse set of MFEs emerges, which hampers the establishment of automated pipelines, fosters code duplication, and complicates developer interchange between teams.

Solution: Whenever a new technology is used to implement a MFE, the development team must create a repository containing the necessary base code, known as a boilerplate. The boilerplate should enable the creation of new MFEs with the same technology by simply cloning it. Keeping the boilerplate updated with new design patterns and library versions is crucial. Additionally, the development team should create comprehensive documentation detailing the entire process of creating a new MFE, regardless of the technology. This documentation should provide instructions on adding automated CI/CD, integrating the MFE into the existing system, and addressing other relevant aspects.

4.5.10 Common Ownership

Category: Development

Problem: A single team is tasked with managing all MFEs, which can occur either due to a lack of team division or when teams are segmented based on technical aspects such as data, frontend, and backend. However, one of the key benefits of MFE architecture is independence, so adopting MFE Architecture without distinct teams to operate independently negates this advantage.

Example: A small company chooses to adopt the MFE architecture. Due to the insufficient number of developers, it is not feasible to create teams by domain. So, developers compose a single team responsible for all MFEs. In this scenario, the cost of maintaining different micro frontends is not justified and is only an additional challenge for the development team.

Solution: Context should be the defining factor when structuring development teams. Therefore, defining the boundaries of teams and MFEs is essential according

to Domain-driven Design (EVANS, 2004), so a team will be responsible only for MFEs within its domain. Creating shared libraries can facilitate boundary definition and promote greater team independence.

4.5.11 Golden Hammer

Category: Development

Problem: All MFEs utilize the same technology, even if it does not meet the specific needs of each MFE. It happens due to developers' familiarity with only one specific technology. This approach limits the architecture, failing to take advantage of the benefits of the possibility of a heterogeneous architecture, which is one of the main attractions of adopting MFEs.

Example: A web application contains MFEs implemented using ReactJS framework with Client-side Rendering, even those encompassing essential pages such as the landing page. This technological uniformity overlooks the necessity for Search Engine Optimization (SEO) strategies to ensure better rankings on search engines like Google. It would be advisable to utilize ReactJS with Server-side Rendering or employ a static rendering framework such as NextJS, enabling better optimization for search engines.

Solution: To choose the most suitable technology that addresses the specific challenges of each MFE, which includes adopting the correct programming languages, frameworks, and libraries during its development. When uncertain about a particular technology, conducting a proof-of-concept (POC) can validate its suitability. Testing new technologies through POCs helps validate their suitability without compromising the establishment of standardized patterns within the company. However, it's important to note that increasing the variety of technologies can increase the complexity of the architecture.

4.6 Catalog's web application

Drawing on the collaborative repository model proposed by Bogner et al. (2019), we developed an web application to showcase all anti-patterns.[1] The web application allows users to search for any word in the anti-patterns name, problem, solution, and example. Figure 19 presents the home screen of the web application; Figure 20 presents the search results screen after searching for "fragment"; Figure 21 presents an example of the screen that details a anti-pattern; and Figure 22 presents the about screen.

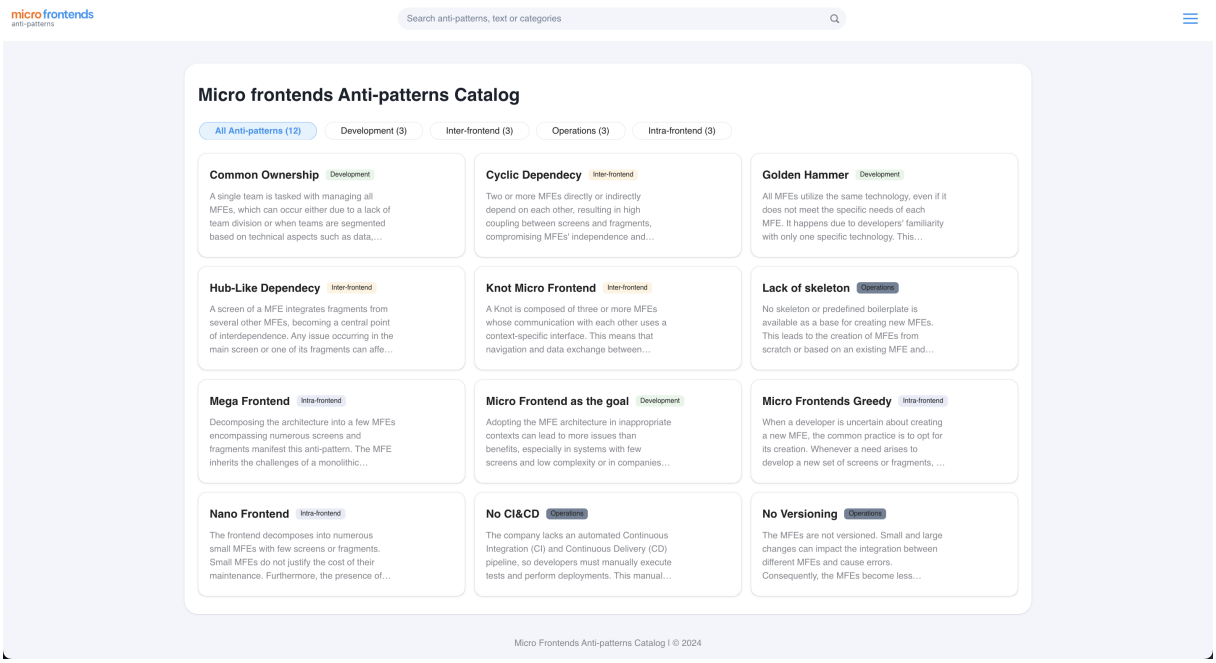

Figure 19 – Home screen of the catalog's web application.

The application was built with ReactJS (REACT, 2025) and is hosted on GitHub Pages, making it publicly accessible. Community members can contribute to the catalog by submitting Pull Requests (PRs) to the GitHub repository[2] following the contribution guidelines outlined in the CONTRIBUTING.md file[3]. All anti-patterns are stored in the `src/anti-patterns` folder[4], with each JSON file corresponding to a specific anti-pattern. Thanks to GitHub Actions (GITHUB, 2025), changes to the codebase are automatically reflected in the application when a PR is merged. This allows MFE

1 <https://mfe-anti-patterns.online/micro-frontends-anti-patterns/#/catalog>
2 <https://github.com/nabsonp/micro-frontends-anti-patterns>
3 <https://github.com/nabsonp/micro-frontends-anti-patterns/blob/main/CONTRIBUTING.md>
4 <https://github.com/nabsonp/micro-frontends-anti-patterns/blob/main/src/anti-patterns/index.ts>

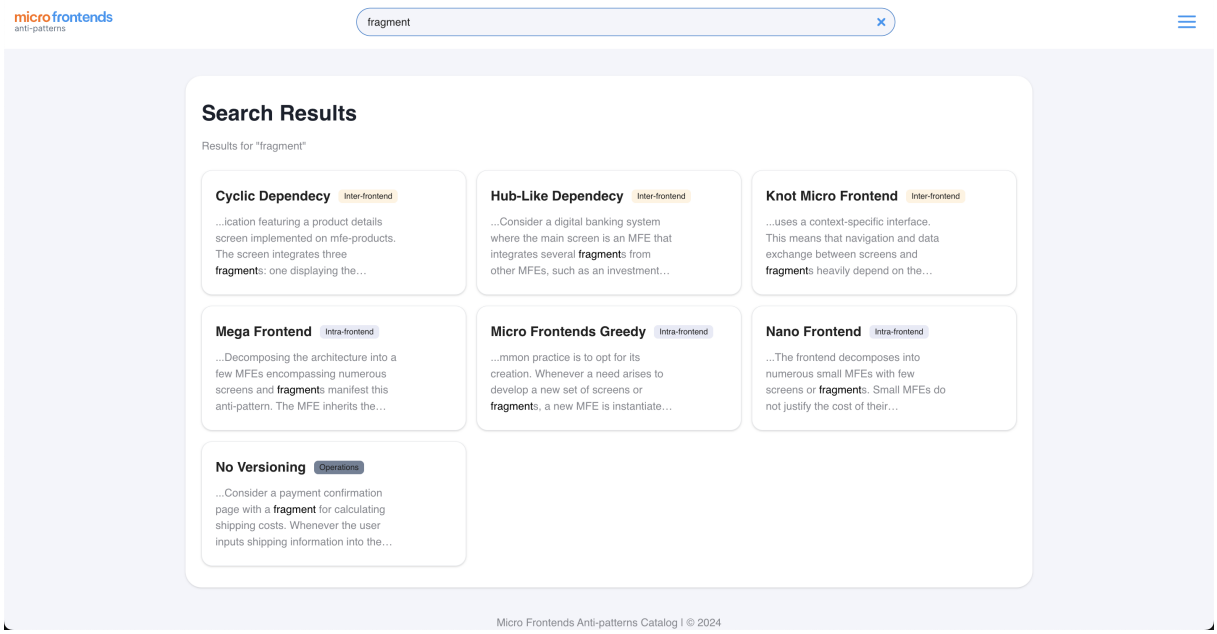

Figure 20 – Search results screen of the catalog's web application.

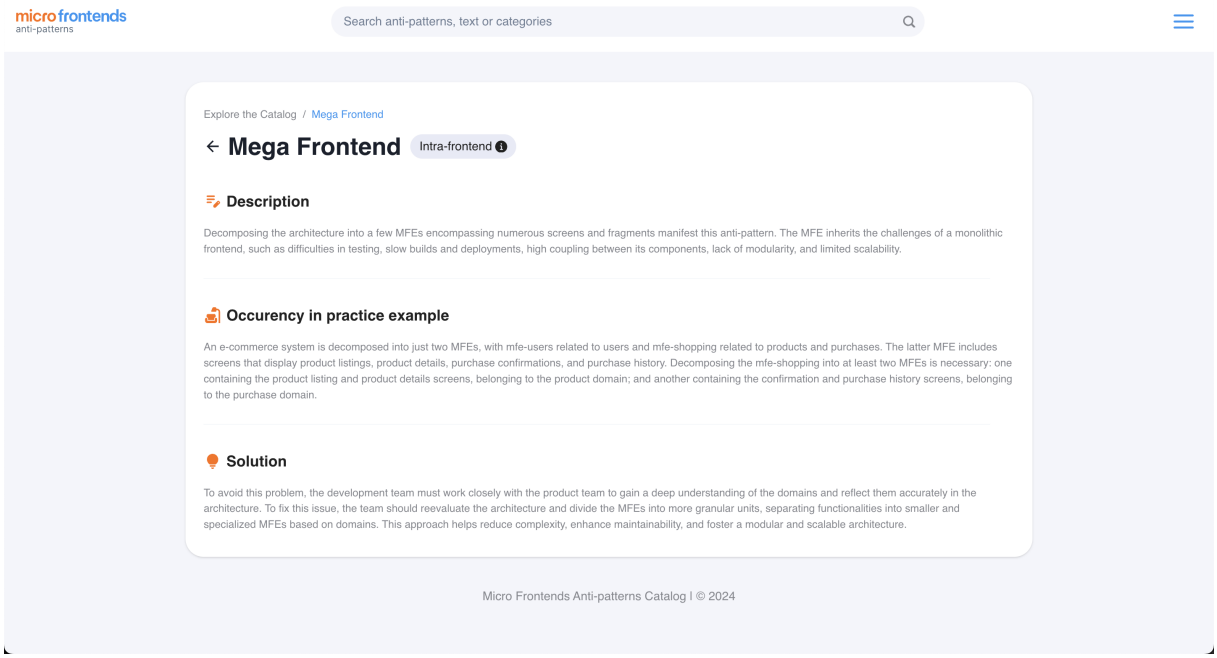

Figure 21 – Example of the anti-patterns details screen of the catalog's web application.

developers to propose new anti-patterns or update the descriptions of existing ones, fostering community collaboration.

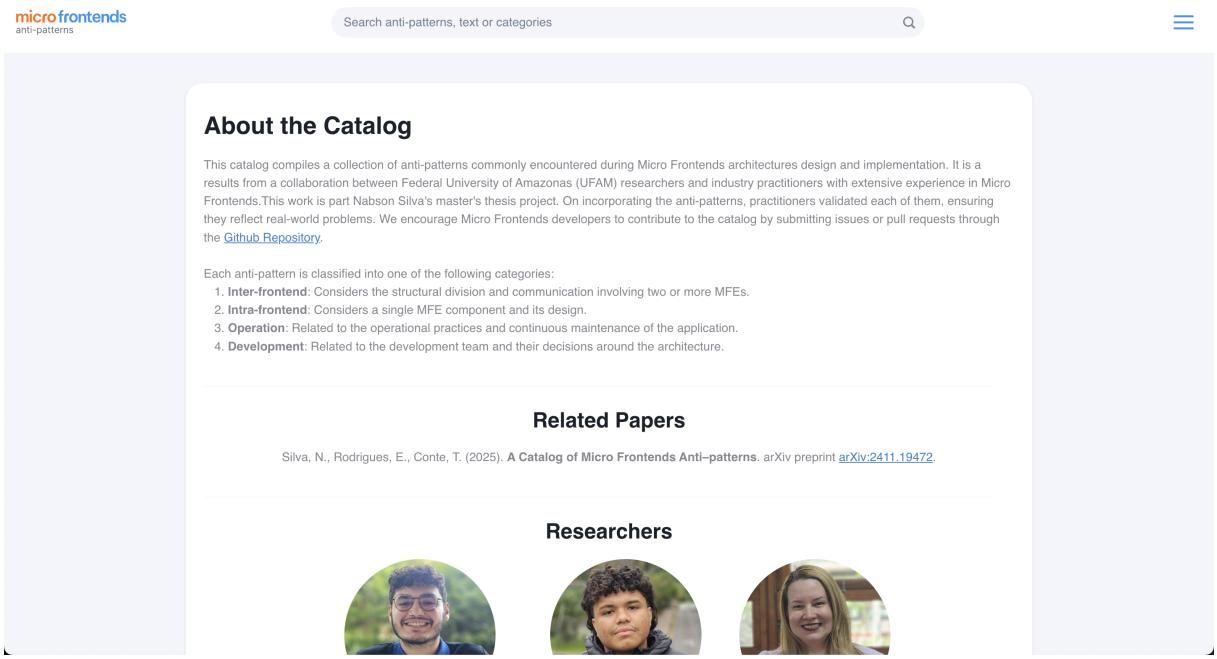

Figure 22 – About screen of the catalog's website.

5

Student Feedback on the Catalog and Its Contribution For Learning

Despite widespread adoption in the industry, MFE has yet to be incorporated into the software architecture course curriculum. This gap reflects the scarcity of academic research on MFE education, contrasting with the abundance of experience reports and case studies detailing its implementation (ANTUNES et al., 2024; CAPDEPON et al., 2023; KAUSHIK; KUMAR; RAJ, 2024; PERLIN et al., 2023; MÄNNISTÖ; TUOVINEN; RAATIKAINEN, 2023; PÖLÖSKEI; BUB, 2021; MORAES et al., 2024). Without formal educational resources, practitioners may encounter challenges in effectively implementing the architecture, potentially leading to suboptimal outcomes that hinder the realization of its full benefits.

This chapter addresses the gap in MFE education by reporting our experience teaching Micro Frontends in an undergraduate Computer Science course. We evaluated our MFE anti-patterns catalog in comparison to a presentation we designed containing practitioner-provided guidelines. Both are supporting materials to help students understand Micro Frontends by illustrating real-world scenarios and highlighting common challenges faced by developers. To assess their effectiveness, we conducted a controlled experiment comparing the two materials in terms of their impact on students' learning. Additionally, we examined whether the anti-patterns catalog improves students' per-

ceived learning and how it was used during assessments. Our goal was to investigate the catalog's potential as a pedagogical tool to support both learning and architectural decision-making in the context of MFE development.

This chapter is structured as follows: Section 5.1 describes the study design. Section 5.2 presents the results of the experiment. Section 5.3 discusses the results. Section 5.4 examines the threats to validity. Finally, Section 5.5 presents some of the improved anti-patterns refined based on students' feedback.

5.1 Study Design

To answer RQ1 and RQ2, we designed a controlled experiment following the guidelines proposed by Wohlin et al. (2012). The following subsections provide a detailed description of each aspect of the study.

5.1.1 Goal and Research Questions

The experiment's goal is to explore effective teaching strategies for MFE by comparing the MFE anti-patterns catalog with practitioner-provided guidelines as supporting materials. Additionally, we aim to analyze whether the MFE anti-patterns catalog enhance students perceived learning and how it can be used during MFE maintenance. Therefore, we aim to answer the following Research Questions (RQ):

RQ1

Which supporting material–an MFE anti-patterns catalog or practitioner-provided guidelines–leads to higher student assessment scores?

To address RQ1, we compared the mean scores from the two MFE assessments. In the first assessment, students consulted practitioner-provided guidelines; in the second, they used the MFE anti-patterns catalog.

RQ2

Does the catalog of MFE anti-patterns enhance students' perceived learning about MFE?

For RQ2, we asked students to rate their perceived learning about MFE before and after engaging with the catalog and then compared the two sets of responses.

RQ3

How do students use the MFE anti-patterns catalog, and do they intend to adopt it when solving MFE challenges and learning about MFE?

For RQ3, we evaluated the catalog's utility, ease of use, and students' intention to use it in the future by applying the original Technology Acceptance Model (TAM) (VENKATESH; BALA, 2008). In addition to the TAM constructs, we included four statements to assess how the catalog supported learning about MFE. We also collected qualitative feedback on how students used the catalog and analyzed it through Grounded Theory procedures (CORBIN; STRAUSS, 2014).

5.1.2 Planning

We planned the experiment according to the guidelines of Wohlin et al. (2012).

5.1.2.1 Context Selection

We conducted the experiment with undergraduate students learning about MFE for the first time without prior experience in this architectural style. This setup simulates junior developers entering a company and needing to work with MFE architectures. Since no published MFE architecture specifications described complete applications—including the MFEs implemented, their screens and fragments, and their communication and composition strategies—we developed two MFE applications for students to analyze during the experiment.

5.1.2.2 Variable selection

The independent variable is the supporting material consulted during the assessments, with two treatments: (1) the catalog of MFE anti-patterns and (2) the practitioner-provided guidelines. The dependent variables are the students' assessment scores and perceived learning scores. The assessment scores are real values ranging from 0 to 10, reflecting the actual grades students received on the exercises. In contrast, the perceived learning scores range from 0 to 5. We adopted a different scale to reduce potential bias in self-assessment, as students might feel uncomfortable assigning themselves the maximum grade (i.e., 10), fearing it could be misinterpreted as their exercise grade or lead to penalties. This distinction aimed to encourage more honest and unconstrained reflections on their perceived learning. We measured perceived learning as real values to enable statistical comparison between samples (WOHLIN et al., 2012) and evaluate whether the catalog positively influenced students' learning perception, following the approach of meireles2024experience.

5.1.2.3 Hypothesis formulation

We formulated two hypotheses, each corresponding to RQ1 and RQ2. The null hypothesis H_0, related to RQ1, states that there is no significant difference in students' mean assessment scores when supported by practitioner-provided guidelines compared to when using the anti-patterns catalog. The second null hypothesis, H_0', related to RQ2, states that there is no significant difference in students' perceived learning before and after interacting with the MFE anti-patterns catalog. As RQ3 is addressed through qualitative methods, we did not define a hypothesis for it.

5.1.2.4 Selection of subjects

Participant selection was based on convenience sampling (WOHLIN et al., 2012), targeting undergraduate Computer Science students enrolled in the Systems Analysis and Design course at UFAM. Participation in the study was voluntary, and only students

who signed the consent form and attended at least all but one session were included in the experiment.

5.1.2.5 Experiment design

Students completed two assessments related to MFE, which questions can be found in Appendix C. We designed two MFE architecture specifications (Objects) to support the assessments: Object 1 represented an e-commerce web application, while Object 2 represented a mobile application. A full description of the objects can be found in Appendix B. To compare the proposed supporting materials (treatments), we employed a crossover design (VEGAS; APA; JURISTO, 2015) applied to the specification Objects, rather than to the treatments. This decision was necessary because the treatments required prior instruction and could not be delivered independently for each group without risking contamination threat validity (WOHLIN et al., 2012). Since students continued attending shared sessions over time, separating them into different groups would not prevent cross-influence. Therefore, we alternated only the architecture objects analyzed in each assessment to maintain comparability, minimizing bias.

The students were divided into two groups, Group A and Group B. Since none of the students had prior experience with MFE, we balanced the groups based on software development experience type (backend, frontend, or full-stack) and duration, if any. We employed a crossover design (VEGAS; APA; JURISTO, 2015) to analyze the two objects under consideration. During the first assessment, both groups consulted the presentation from the fourth session, which included practitioner-provided guidelines and MFE examples. Group A completed the first assessment based on Object 1, while Group B completed it based on Object 2. For the second assessment, both groups accessed a web application containing the MFE anti-patterns catalog, with the Objects reversed: Group A analyzed Object 2, and Group B analyzed Object 1. Figure 23 presents an overview of the experiment design.

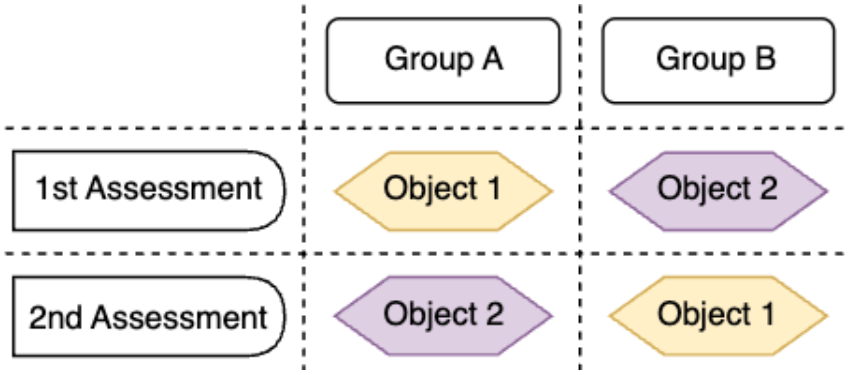

Figure 23 – Timeline of the experiment execution.

5.1.2.6 Instrumentation

The instruments for this experiment include a set of forms, training session documents, architecture descriptions, and an activity script that must be followed during the execution phase. We describe each instrument in detail as follows:

1. **Theoretical training**: Presentations delivered during lecture sessions and the code and script presented during the laboratory session.

2. **Consent form**: form that outlines the purpose of the experiment and details how it will be conducted. We emphasize that participation is voluntary, allowing participants to withdraw from the experiment at any time without affecting their scores on the assessments. Furthermore, we explain that we will use the collected data for quantitative and qualitative analysis and may include it in scientific publications. Participants signed the consent form before the experiment.

3. **Characterization form**: a set of questions designed to gather information about the participants' professional experience as software developers and their familiarity with MFE architectures.

4. **Objects**: description of two MFE architectures that students will evaluate during the experiment. The objects include a brief description of the software and its functionalities, images showcasing its screens, and a comprehensive list of MFEs detailing their context, screens, and fragments. A full description of the objects can be found in Appendix B.

5. **Assessment Forms**: The two MFE assessment forms, each focused on a different object. All assessment's questions can be found in Appendix C.

6. **Feedback form**: This form includes the TAM constructs, the learning-related statements, and the open-ended questions that allow participants to provide further insights.

5.1.3 Execution

To teach students about MFE and MS, we delivered 4 theoretical lecture sessions and 1 laboratory session. Table 6 presents the sessions and its content. Before the first lecture, students already had knowledge about architectural styles like Layered Architectures, Data-centered architectures, and Pipers & Filters (VALENTE, 2020), and about architecture visualization with C4 Model (BROWN, 2023).

Table 6 – MFE sessions' content.

#	Session	Description
1	**Microservices**	Monoliths, microservices, and patterns for microservices.
2	**Micro Frontends**	Definition, benefits, challenges, and frontend integration.
3	**Micro Frontends Hands-on**	Lab session where students implemented routing, composition, and communication in a web e-commerce application.
4	**Examples of Micro Frontends Architectures**	Public MFE examples and guidelines from experience reports, case studies and blogs.
5	**Micro Frontends Anti-patterns**	Definition of anti-patterns and an explanation of the 12 MFE anti-patterns.

Since MFE is based on MS, the first lecture session aims to introduce the definition, benefits, and challenges of MS. The lecture begins by detailing the monolithic architectural style and the issues that motivated the creation of MS (NEWMAN, 2021). Then, we presented the concept of MS and their key principles. To illustrate some architectural examples, we discussed the following patterns: Remote Procedure Invocation, Asynchronous Messaging, API Gateway, Backend for Frontend (BFF), and API Composition (RICHARDSON, 2018).

In the second lecture session, we introduced the concept, benefits, and challenges of MFE primarily based on Geers (2020) and Peltonen, Mezzalira & Taibi (2021). We also delved into implementing frontend integration through composition, communication, and routing. Finally, we presented an MFE architecture we developed, which would be necessary in the hands-on session of the following lecture session. The primary goal of this lecture was to provide the theoretical foundation of MFE.

Then, during the third session, we conducted a lab session where students could engage with the implementation of an e-commerce application built using 3 MFEs.[1] We explained how the application uses the Single-SPA framework (SINGLE-SPA, 2016) to compose the MFEs and then assigned three exercises focused on routing, composition, and communication. Our goal in this session was to make students see how MFE work in practice, since it's concepts may be confusing.

In the fourth lecture session, we showcased the architectures published by Antunes et al. (2024), Moraes et al. (2024), and Silva (2024) to analyze their design choices and discuss potential alternatives. Additionally, we presented the MFE guidelines published by Taibi & Mezzalira (2022), Aplyca (2024), Kofler (2020), Anks (2023), and Shukla (2023). We aimed to present practitioners-provided examples and guidelines on how MFE has been implemented, enhancing students training to assess MFE architectures effectively.

Following the fourth lecture, we invited students to voluntarily participate in the experiment by signing a consent form and completing a characterization form. We then used the characterization data to balance the groups, ensuring that the number of participants with expertise in each development area was approximately equal and that the overall experience level was pretty distributed between the groups. Finally, in the fifth lecture session, we presented each of the 12 MFE anti-patterns (see Section 4.5). This lecture was crucial for helping students understand the anti-patterns, enabling them to apply this knowledge in the assessments effectively.

Figure 24 presents the timeline of the experiment execution. The experiment consisted of two assessments conducted on different days. In the first assessment, both groups had access to the guidelines and examples presented in the fourth lecture.

[1] <https://github.com/nabsonp/mfe-hands-on>

Group A analyzed Object 1, while Group B analyzed Object 2. After completing the first assessment, we delivered the fifth lecture introducing the MFE anti-patterns catalog, which students would use during the second assessment. In the second assessment, Group A analyzed Object 2, and Group B analyzed Object 1, with both groups granted access to the web application containing the anti-patterns catalog. After completing the second assessment, students also completed the feedback form.

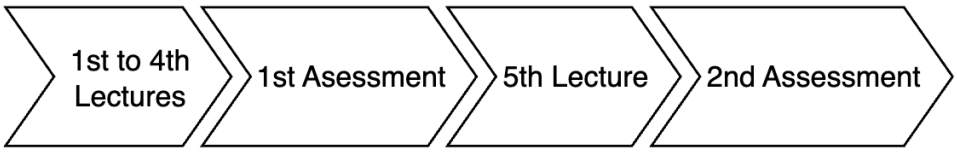

Figure 24 – Timeline of the experiment execution.

We conducted two comparable assessments related to MFE, each containing similar questions focused on analyzing two distinct architectures. The questions simulated tasks a new developer might face when joining a team working with MFE architectures. The two systems—an e-commerce platform and a digital bank-were inspired by real-world MFE implementations. We selected these domains because students were already familiar with them, reducing the risk of confusion due to unfamiliar business contexts. Each assessment consists of two parts:

1. **Architecture Analysis**: we asked students to evaluate the proposed MFE architecture and indicate whether they would design it differently, providing justifications for their decisions.

2. **Maintenance and Evolution**: eight questions that required students to develop a new feature, understand and resolve a bug in the application, or refactor a part of the architecture.

To ensure the realism of the assessments and the proposed architectures, we asked two experienced MFE professionals to complete the assessments and provide feedback on whether the problems reflected real-world scenarios and whether the architectures resembled those commonly seen in the industry. We chose not to ask students to design a MFE architecture from scratch, as, in practice, they are more likely to

maintain and evolve an existing architecture than creating one from scratch (GALSTER; ANGELOV, 2016; KAZMAN et al., 2023; MANNISTO; SAVOLAINEN; MYLLARNIEMI, 2008). We ensured that every question could be answered using either the anti-patterns catalog or the practitioner-provided guidelines, allowing for a fair comparison between the two supporting materials.

5.1.4 Analysis and Interpretation

Before conducting the data analysis, we excluded responses from students who missed more than one lecture to include only those who participated in the majority of the training sessions. We conducted the quantitative analysis using two groups of paired samples: (1) student assessment scores when consulting practitioner-provided guidelines versus the anti-patterns catalog, and (2) perceived learning scores before and after engaging with the catalog. We used the assessment score samples to test the null hypothesis H_0 and the perceived learning samples to test the null hypothesis H'_0.

We began by examining boxplots to visually compare the samples within each group. Next, we applied the Shapiro–Wilk test (WOHLIN et al., 2012) to determine whether the samples followed a normal distribution. Based on the results, we selected appropriate paired statistical tests: the Paired t-Student Test (WOHLIN et al., 2012) for normally distributed samples, and the Wilcoxon Signed Rank Test (WOHLIN et al., 2012) for non-normal samples.

To investigate how students (representing novice MFE developers) perceived the catalog's usefulness during architectural decision-making in the assessments, we applied the original TAM (VENKATESH; BALA, 2008). TAM assesses users' perceptions of a technology's usefulness and ease of use, which are the two primary factors influencing technology acceptance behavior (LAITENBERGER; DREYER, 1998). We measured the constructs of perceived usefulness, perceived ease of use, and behavioral intention using a 5-point Likert scale (LIKERT, 1932), ranging from "Strongly disagree" to "Strongly agree." In addition to the TAM constructs, we defined four sentences for measuring the catalog's utility for learning MFE. Table 7 presents our adapted version

of the TAM questionnaire and the four items related to the catalog's perceived utility for learning.

Table 7 – Adapted TAM constructs and the sentences related to the catalog's utility for learning.

Perceived Usefulness (PU)	
PU01	Using the anti-patterns catalog for MFE improves my performance when developing MFE-oriented architectures.
PU02	Using the anti-patterns catalog for MFE improves my productivity when developing MFE-oriented architectures.
PU03	Using the anti-patterns catalog for MFE improves my effectiveness in communicating with other developers when developing MFE-oriented architectures.
PU04	I consider the anti-patterns catalog for MFE useful for developing MFE-oriented architectures.
Perceived Ease Of Use (PEOU)	
PEOU01	My interaction with the anti-patterns catalog for MFE was clear and understandable.
PEOU02	Interacting with the anti-patterns catalog for MFE did not require much mental effort from me.
PEOU03	I consider the anti-patterns catalog for MFE easy to use.
PEOU04	I find it easy to use the anti-patterns catalog for MFE during the development of MFE architectures.
Behavioral Intention (BI)	
BI01	Assuming I have enough time to design and develop an MFE-oriented architecture, I would use the anti-patterns catalog for MFE.
BI02	Considering that I can choose other tools to assist in the development of MFE-oriented architectures, I intend to use the anti-patterns catalog for MFE.
BI03	I intend to use the anti-patterns catalog for MFE the next time I work on an MFE-oriented architecture.
Useful For Learning (UFL)	
UFL01	I consider the anti-patterns catalog for MFE useful for learning about MFE.
UFL02	I find it easy to use the anti-patterns catalog for MFE to learn about MFE-oriented architectures.
UFL03	The anti-patterns catalog for MFE facilitated learning about best practices in software architecture.
UFL04	The anti-patterns catalog for MFE contributed to my understanding of common challenges in MFE architectures.

We collected qualitative feedback on how students used the catalog, its perceived benefits and challenges, how it influenced their learning, and their overall impressions of the catalog. We analyzed the data using Straussian Grounded Theory (GT) pro-

cedures (CORBIN; STRAUSS, 2014). We first applied open coding, which involved creating codes representing relevant concepts to understand how students used the catalog. Then, when applying axial coding, we identified connections among the codes and grouped them into broader categories to identify the primary uses of the catalog. We did not apply the GT's selective coding, as our objective was not to develop a theory but to explore how the catalog can be used during MFE development.

5.2 Results

This section presents the participants' characterization and the results for each RQ. For RQ1 and RQ2, we analyze the quantitative data collected during the experiment using boxplots and statistical tests to compare the samples. For RQ3, we present the results of the TAM evaluation and the learning-related statements, and summarize insights from the qualitative analysis of students' feedback.

5.2.1 Participants' Characterization

The class had 34 students, of which 23 were eligible to participate in the study after excluding those who missed more than one lecture. All students signed a consent form and completed the characterization form prior to the experiment. We assigned each participant an identifier ranging from P1 to P23.

Table 8 presents a summary of experiment participants' characterization. Each column presents the answer of one of the characterization question: column "Has development experience" refers to "Do you have any experience with web development outside the context of a course (e.g., R&D projects, industry, etc.)?"; column "Experience Time" refers to "If you have experience in industry or in RD projects, how much experience do you have?"; column "Experience with MFE" refers to "Before the classes, had you had any prior exposure to Micro Frontends architectures?"; and column "Group" informs the group the participant was assigned to. We also asked participants to describe their experience with MFEs, if any. However, since no participant had prior experience

with MFEs, all answers were left empty and were therefore omitted from the table.

Table 8 – Summary of the experiment's participants' characterization.

ID	Has development experience	Experience time	Experience with MFE	Group
P1	No	-	No	A
P2	No	-	No	B
P3	No	-	No	B
P4	Yes, as Backend Developer	Between 1 and 2 years	No	B
P5	No	-	No	B
P6	Yes, as Full Stack Developer	Between 1 and 2 years	No	B
P7	No	-	No	A
P8	No	-	No	A
P9	No	-	No	A
P10	No	-	No	B
P11	No	-	No	A
P12	Yes, as Full Stack Developer	Less than 1 year	No	A
P13	No	-	No	B
P14	No	-	No	B
P15	No	-	No	A
P16	No	-	No	B
P17	No	-	No	B
P18	No	-	No	B
P19	No	-	No	A
P20	No	-	No	A
P21	No	-	No	A
P22	No	-	No	A
P23	Yes, as Frontend Developer	Between 2 and 3 years	No	A

To ensure a fair and balanced distribution of skills among the groups, we conducted the group balancing based on the type and duration of experience in software development:

- **Backend Development**: one participant with 1 to 2 years of experience (P4).
- **Full Stack Development**: 2 participants, one with 1 to 2 years of experience and one with less than 1 year (P6 and P12).
- **Frontend Development**: one participant with 2 to 3 years of experience (P23).

We balanced the groups to ensure that the number of participants in each area of expertise was approximately balanced. Additionally, the experience levels were distributed relatively evenly across the groups. Group A included one participant with less than 1 year of experience in Full Stack Development (P12) and one participant with 2 to 3 years of experience in Frontend Development (P23). Group B included one participant with 1 to 2 years of experience in Backend Development (P4) and one participants with 1 to 2 years of experience in Full Stack Development (P6).

5.2.2 RQ1: Supporting Materials Comparison

To address RQ1, we evaluated the students' scores from the first and second assessments. Figure 25 presents the boxplots for the two samples, which show similar distributions and are close to each other. Performing the Shapiro-Wilk Test on the first sample yielded a p-value of 0.764. The second sample reported its mean, median, and mode as 3.917, 4.000, and 4.000, respectively. Performing the Shapiro-Wilk Test on the second sample yielded a p-value of 0.848. With a significance level of $\alpha = 0.05$, both samples are considered to follow a normal distribution.

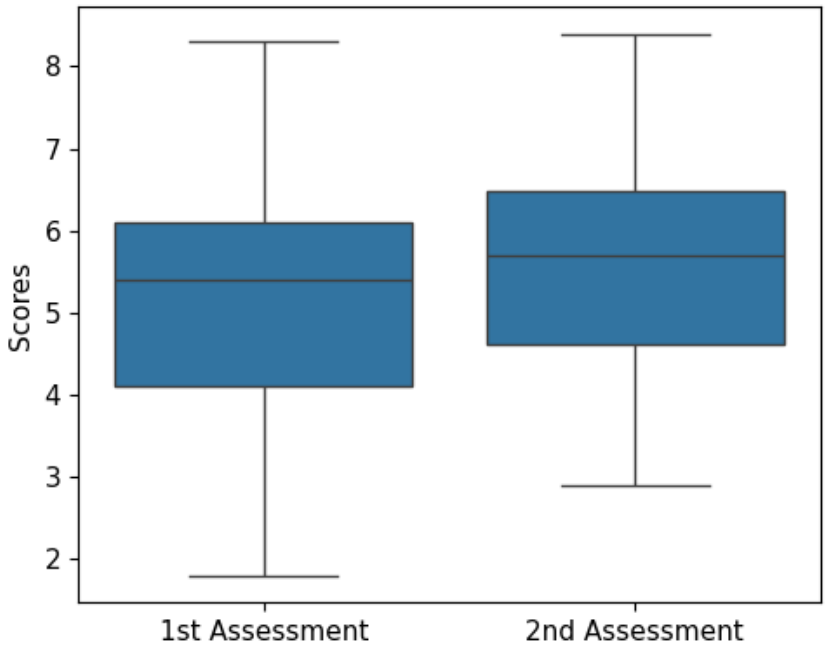

Figure 25 – Boxplots presenting the data distribution on the assessment samples.

We selected the Paired t-Student Test for the sample comparison, a parametric test for dependent and normally distributed samples (WOHLIN et al., 2012). With the significance level set at $\alpha = 0.05$, the test yielded a p-value of 0.298, indicating that we cannot reject the null hypothesis H_0. Therefore, we conclude that both supporting materials are equally effective in helping students learn about MFE.

RQ1 Summary

There was no significant difference in students' assessment scores when using the MFE anti-patterns catalog versus the practitioner-provided guidelines. This result indicates that both supporting materials are equally effective in helping students learn about micro frontends.

5.2.3 Perceived learning Difference

To address RQ2, we evaluated the students' self-assessed perceived learning before and after engaging with the catalog. Figure 26 presents the boxplots for the two samples, indicating that the second sample appears to have higher values than the first one. Performing the Shapiro–Wilk Test on both samples yielded a p-value of 0.216 for the first sample and < 0.001 for the second sample. Considering a significance level of $\alpha = 0.05$, the second sample is not normally distributed.

As one of the samples is not normally distributed, we selected the Wilcoxon Signed Rank Test to compare the samples, as it is an appropriate non-parametric test for comparing dependent samples (WOHLIN et al., 2012). Using a significance level of $\alpha = 0.05$, the test yielded a p-value of 0.001, allowing us to reject the null hypothesis H_0'. Thus, we can assert a statistically significant difference in students' self-perceived learning before and after using the catalog for solving MFE architectural problems. Since the distribution of perceived learning after engaging with the catalog shows higher values than before, we conclude that using the catalog enhances students' self-perception of learning about MFE.

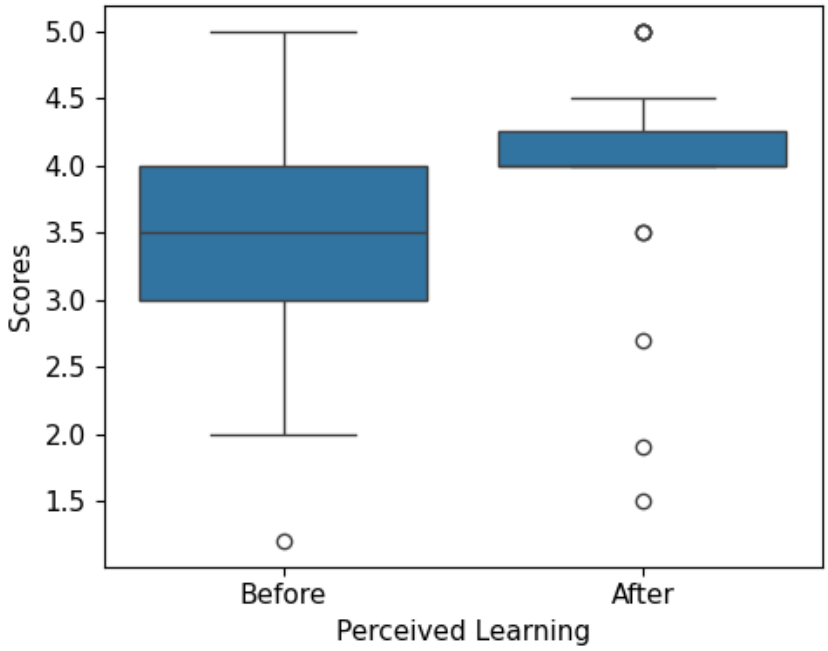

Figure 26 – Boxplots presenting the data distribution on the perceived learning before and after engaging with the MFE anti-patterns catalog.

RQ2 Summary

After using the MFE anti-patterns catalog, students reported a statistically significant increase in their perceived learning, suggesting that exposure to real-world problems and examples helps students feel that they have learned more effectively.

5.2.4 How students used the catalog

To address RQ3, we asked students to answer a online form to respond if they agree with the TAM and the learning-related statements, and we collected qualitative feedback on how students used the catalog, its benefits and challenges, how it influenced their learning, and their overall impressions of the catalog.

5.2.4.1 TAM and learning statements analysis

Figure 27 summarizes the results on the Perceived Usefulness construct. Upon analyzing PU01, PU02, and PU04, the students generally agreed that the catalog is a useful tool for

developing MFE architectures, highlighting its potential to improve their performance and productivity during development. Although they did not create entirely new architectures during the assignment, they were tasked with proposing new solutions for existing ones. Regarding PU03, some feedback reflected a neutral stance, which may be attributed to the fact that the students did not interact with other developers while proposing their architectural solutions.

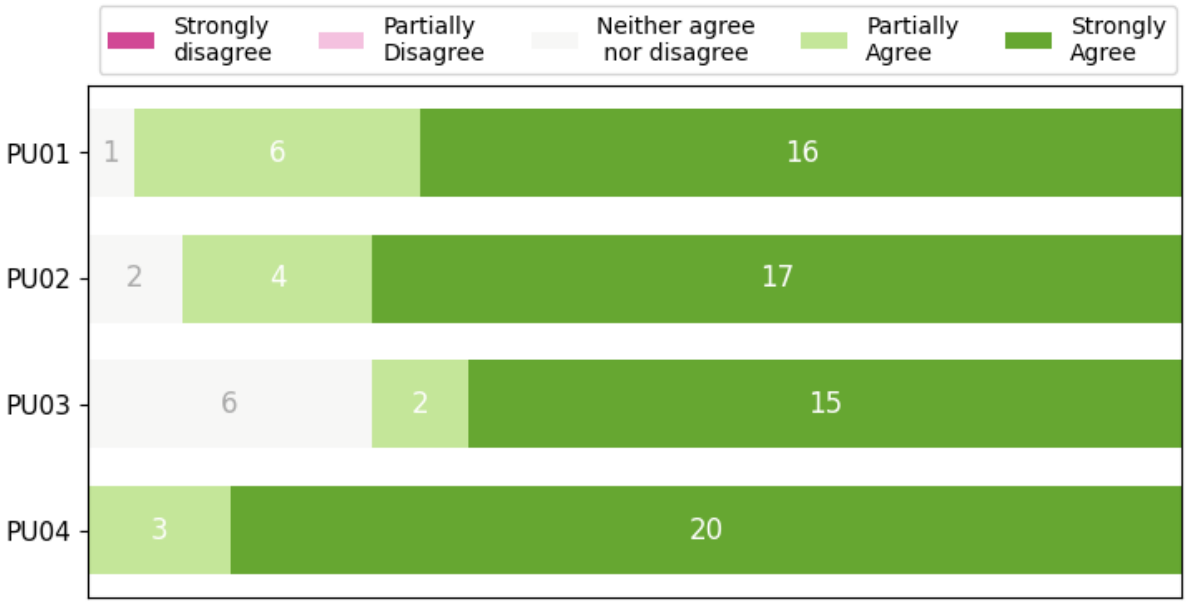

Figure 27 – Responses for each sentence in the Perceived Usefulness construct.

Figure 28 presents the Perceived Ease Of Use construct results. Upon analyzing PEOU02 and PEOU03, all the students agreed that the tool was easy to use and did not require much mental effort. Regarding PEOU01, only one response indicated a neutral stance, suggesting that interacting with the catalog was clear and comprehensible. On PEOU04, its feedback indicates that using the catalog to develop MFE architectures may be easy, as it presents neutral responses or partial agreements. These results support the notion that the catalog offers a low-friction experience, which may facilitate its adoption in educational and professional contexts.

Figure 29 summarizes the results for the Behavioral Intention construct. Upon analyzing BI01, students generally expressed a positive intention to use the catalog, especially when given sufficient time for design and development tasks. Most participants also agreed they would use the catalog to support MFE development (BI02) and

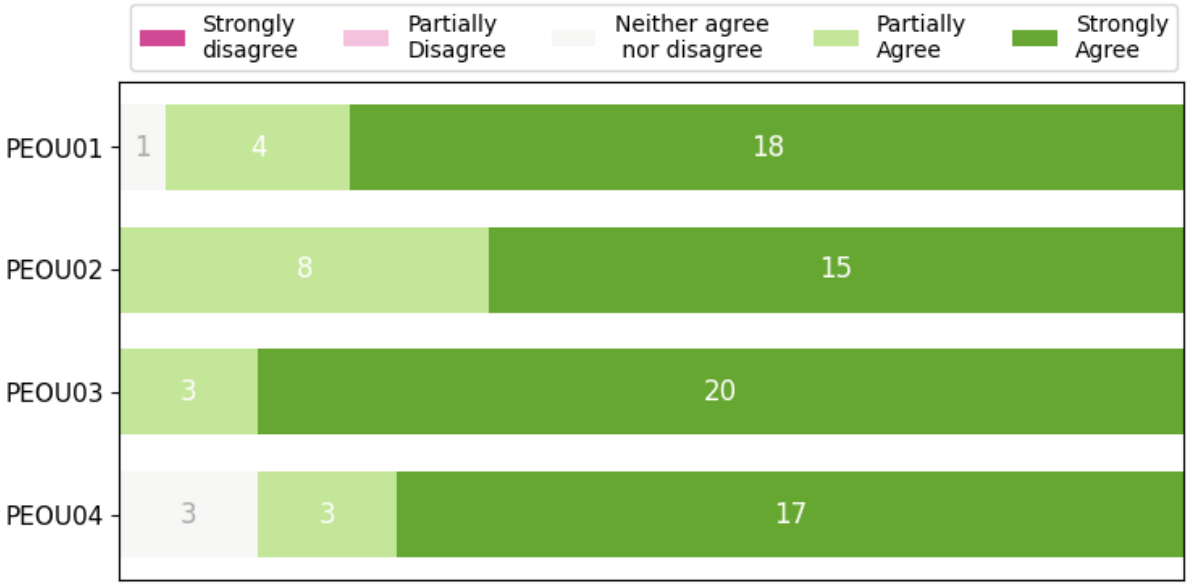

Figure 28 – Responses for each sentence in the Perceived Ease Of Use construct.

future MFE projects (BI03). These findings suggest that the catalog holds potential as a valuable tool for both in-training software engineers and practitioners.

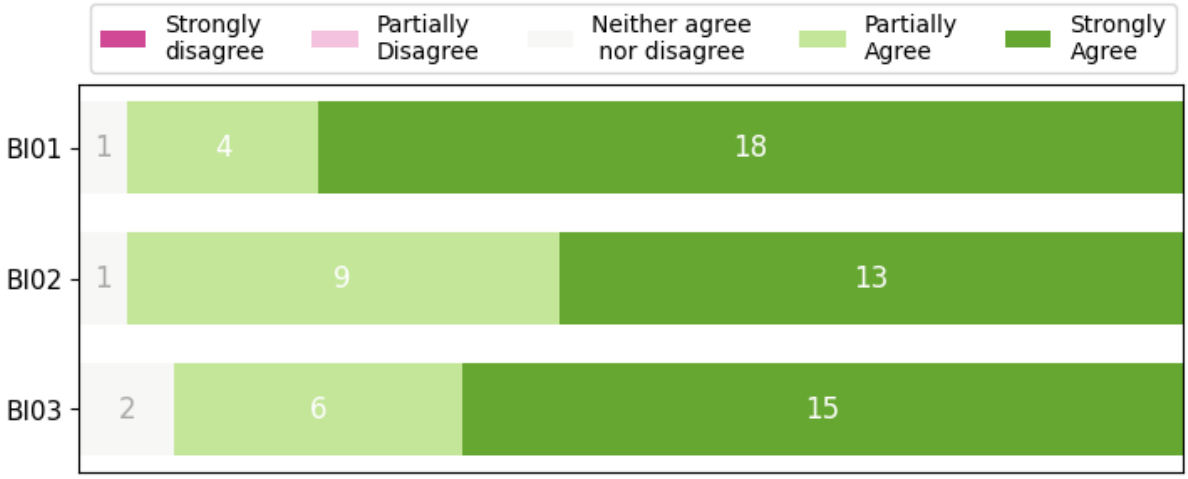

Figure 29 – Responses for each sentence in the Behavioral Intention construct.

Figure 30 summarizes the results related to the catalog's utility for learning provided by the students through the survey. Regarding the statements in UFL01, UFL02, UFL03, and UFL04, there were no disagreements or neutral responses, as most were "Strongly agree" and "Partially agree." It suggests that students recognized the catalog's contribution to the learning process concerning software system architecture and MFE architectures. We further discuss the enhancement in learning during the

qualitative analysis results.

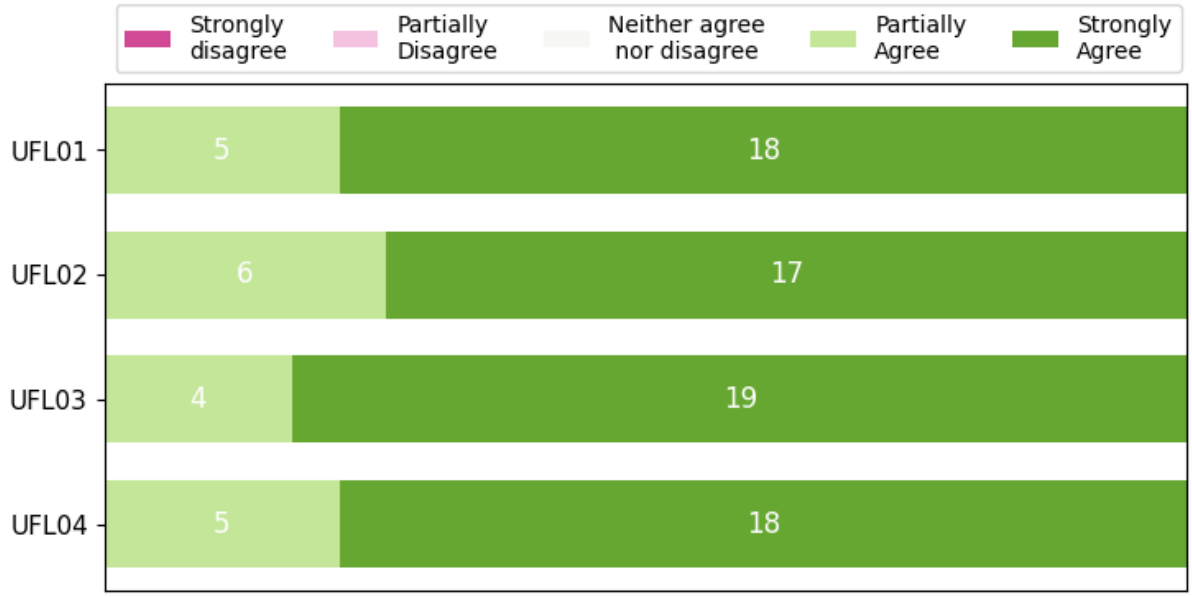

Figure 30 – Responses for each sentence related to the catalog's utility for learning.

5.2.4.2 Qualitative feedback analysis

We analyzed students' qualitative feedback using ATLAS.ti[2], applying open and axial coding (CORBIN; STRAUSS, 2014) to understand how students used the catalog during the second assessment. The quotes and the related codes are available in Appendix D. Based on the relationships among the open codes (highlighted with underlines), we identified three categories that summarize the main ways in which students interacted with the catalog:

1. **Identify problems and solutions** – As anticipated, students used the catalog to Identify problems and solutions, as expressed by P1: "*identify the problems and the solutions to the respective questions.*" However, their approaches to identifying problems varied according to the features they relied on within the catalog. Some students emphasized the role of examples in Identify problems by examples, as noted by P3: "*the examples helped me identify the problem.*" Additionally, students

[2] <https://atlasti.com/>

used examples to Understand problems by examples, as stated by P1: "*seeing examples and descriptions helped me understand the problem in some questions.*" Others relied on the visual elements, using the catalog to Understand problems by images, as illustrated by P4: "*the use of images in some explanations helped more than just text.*" Additionally, some students also used the solution description, as seen in P5's comment about to Identify problems by solutions: "*the description of the problems and solutions in the catalog helped me a lot to draw a parallel with the problems proposed.*" Moreover, the catalog also serves to Guide architectural decisions, as stated by P7: "*allowed me to address these problems and made it easier to make decisions.*" These findings suggest that offering multiple pathways–textual descriptions, examples, visuals, and solution-based reasoning–can support diverse cognitive strategies among students. Therefore, the catalog enhances accessibility and understanding by accommodating different ways of thinking and learning.

2. **Support efficient and structured search for MFE problems** – The catalog's web application enabled a structured and efficient browsing experience, helping students access and explore the anti-patterns more effectively. For example, P7 highlighted the ease of Consulting by categories: *"it is easy to visualize, especially with the use of categories."* Even when facing difficulties in understanding the anti-patterns, P4 used the catalog to Consult several anti-patterns at once and made a suggestion to improve problem identification: *"I had difficulties understanding the description of each anti-pattern; I had to open all of them to see the description. There could be a summary in each card of the catalogs on the main screens."* Similarly, P20 suggested Linking anti-patterns to improve navigability: *"when opening an anti-pattern, the page could show others from the same topic or similar ones."*

3. **Reinforce and deepen MFE knowledge** – Students revisited and deepened their understanding of MFE concepts through real-world examples, using the catalog to consolidate and expand their prior knowledge. The catalog can be used to Learn based on practical problems, as P17 remarked: "*Seeing the anti-patterns in practice increased my knowledge about the subject.*" P13 highlighted that the catalog was used to Understand common challenges: "*I could understand real situations and*

challenges faced during architectural decisions." In addition, the catalog was useful to Review MFE concepts, as noted by P21: "*[the catalog] helped me remember some concepts I did not recall.*" P4 reinforced this point and suggested improvements to enhance its role as a learning tool by adding the MFE concepts presented in class: "*[the catalog] helped by centralizing content about MFEs. However, the catalog could include summaries and slides presented in the classes.*" These insights demonstrate the catalog's potential as a reference material and an educational tool that reinforces learning through contextualized, practice-based content.

RQ3 Summary

Students agreed that the MFE anti-patterns catalog is useful, easy to use, and helpful for learning about MFE. They also expressed an intention to use it in future development tasks. Qualitative analysis revealed three primary ways students used the catalog: (1) to identify problems and solutions, (2) to support efficient and structured searches for MFE issues, and (3) to reinforce and deepen their understanding of MFE concepts. These findings highlight the catalog's effectiveness as a problem-solving aid and an educational resource that accommodates diverse learning strategies.

5.3 Discussion

Our supporting materials allowed us to teach how MFE is implemented in real-world scenarios without requiring students to engage with low-level programming details. Students could effectively learn about MFE without developing a complete MFE-based system. The course focused primarily on architectural design and included a single lab session to provide hands-on exposure to an MFE implementation, helping students better understand MFE in practice. As a result, the supplementary resources enabled students to grasp key architectural concepts more clearly, without being overwhelmed by implementation complexity. A potential approach would be to balance high-level architectural instruction with practical coding exercises, which can aid in consolidating learning. However, this balance must be carefully managed, as coding exercises

may detract from architectural understanding, especially for students with limited development experience.

Students reported greater learning gains when exposed to real-world examples presented by the two supporting materials. Feedback indicated that the catalog could even have more examples to support architectural decision-making better. This underscores the limitations of relying solely on textbook-based or overly didactic examples, which often fail to reflect the complexity and nuance of real-world scenarios. Therefore, integrating practical examples into architecture courses is essential to deepen students' understanding and prepare them to apply concepts effectively. Given the detail level of each supporting material, the guidelines are more appropriate for students with no prior exposure to MFE, whereas the catalog is better suited for those with foundational knowledge who aim to reinforce their learning through practical application and problem identification.

Students feedback indicated that the catalog served as a quick, easy, and centralized source of information, making it an effective support tool. It proved particularly valuable during architectural decision-making, with students emphasizing the importance of having easy access to resources that facilitate such processes. Additionally, students suggested that the catalog could include more information on MFE, further highlighting its potential as a reference and a learning resource. Teaching students how to use practical tools that can be seamlessly integrated into their future development workflows significantly enhances their learning experience.

5.4 Threats to validity

We evaluated the validity of the experiment results based on the four types of threats to validity defined by Wohlin et al. (2012):

Internal Validity: A potential threat is interactions with selection, where the sample's characteristics might influence their response to the treatment. To mitigate this, we balanced the groups based on participants' software development experience, ensuring neither group had an advantage. Diffusion, where one group might replicate

the other's treatment, was addressed by placing groups in separate labs, restricting access to only their assigned object, and renaming the object during the second assessment. Instrumentation, related to experimental artifacts' quality, was mitigated by validating all artifacts through a pilot test. To address the learning effect, where participants acquire knowledge during the experiment, we implemented a crossover of objects to balance learning across treatments. The catalog was written in english, but we instructed students to translate the text using Google Translate so it wouldn't affect their use. Finally, training could be confounding if the ad-hoc and anti-pattern lectures were not equivalent. To counter this, we ensured all questions could be answered using the either materials.

External Validity: One threat to external validity is the interaction of setting and treatment, as the objects used in the experiment were not real-world architectures and might not fully represent realistic scenarios. To address this threat and ensure that the objects are realistic, the author of this Thesis proposed a realistic architecture based on his professional experience and validated it with industry professionals with previous MFE expertise. Another threat is the interaction of selection and treatment, as the participants might not represent real-world developers. To mitigate this, we conducted training sessions, used a class of students in the major's sixth semester, and discarded data from students who missed more than one lecture. These measures ensured that participants were comparable to novice developers with limited MFE experience.

Construct Validity: A potential construct validity threat is that the measure may not accurately represent its effect. To address this, we conducted a pilot study with two practitioners whose feedback aligned with the correction criteria, ensuring its reliability. Regarding experimenter expectancy, where questions might elicit desired responses, we based questions on real-world problems and validated them during the pilot study, avoiding the direct use of catalog examples to reduce bias. To mitigate the effects of treatment orders, we selected a crossover design and conducted treatments in separate weeks to reduce carryover. Finally, we did not inform participants that the catalog originated from our prior work to prevent hypothesis guessing, minimizing biased feedback.

Conclusion Validity: Low statistical power, which could hinder the detection of significant effects, is a significant threat to conclusion validity. We used paired statistical tests to mitigate this and implemented a crossover design to collect more data, increasing the analyses' power. Additionally, fishing and the error rate could be a concern if one object favored the tool over the other. We designed both objects with equivalent architectures and similar problems to address this, ensuring fairness between treatments.

5.5 Improved catalog based on students' feedback

Students provided general feedback on the catalog instead of commenting on each anti-pattern individually. The main challenge they highlighted was the large amount of text. Some students reported difficulties understanding the anti-pattern descriptions, often needing to open them multiple times to grasp the full context. To address this, we shortened some descriptions and rewrote unclear phrases to improve readability. Additionally, students suggested displaying related anti-patterns from the same category or similar ones at the bottom of the page when viewing an anti-pattern, making it easier for users to explore connected content. We plan to add this feature in future versions of the web application.

Several students mentioned that the catalog could benefit from more visual elements. They suggested adding more images to make the examples more engaging and easier to remember. Therefore, we added images illustrating problem, example and/or solution to Knot Micro Frontend, Mega Frontend, Nano Frontend, No CI/CD, and Common Ownership anti-patterns. Additionally, students recommended incorporating the slides and summaries from the classes into the catalog to provide more relevant and comprehensive content. In the subsequent subsections we present the Knot Micro Frontend, Nano Frontend, Mega Frontend, No CI/CD, and Common Ownership anti-patterns.

5.5.1 Knot Micro Frontend

Category: Inter-frontends

Problem: A Knot is composed of three or more MFEs whose communication with each other uses a context-specific interface. This means that navigation and data exchange between screens and fragments heavily depend on the unique context of each MFE involved. Adding new MFEs exacerbates the problem: as the number of MFEs grows, the interface complexity increases due to the introduction of new contexts, creating a highly coupled Knot that becomes difficult to maintain and integrate new functionalities.

Example: Suppose an e-commerce system has MFEs for Digital Products (mfe-digital-products) and Payments (mfe-payments). The payment screen of mfe-payments receives the digital product data as a parameter. At a later stage, a Physical Products MFE (mfe-physical-products) is implemented, so developers add physical product-specific attributes to the payment screen to allow digital or physical product payment (Figure 31). Later, adding new product types requires constantly adding attributes specific to each product type to the payment screen, so it becomes a highly coupled knot.

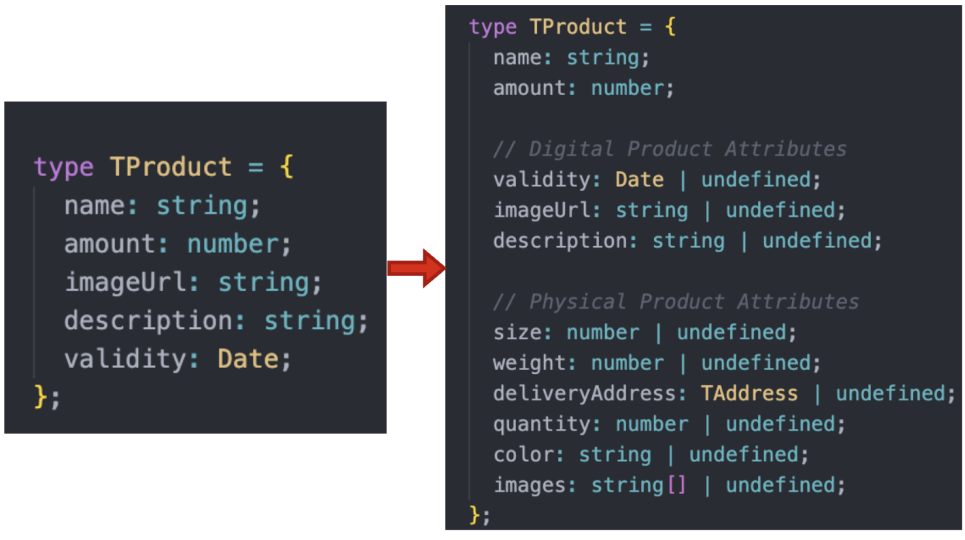

Figure 31 – To allow physical products payment, optional physical product-specific attributes are added and digital product-specific attributes become optional.

Solution: Implement generic and flexible interfaces based on domain. These interfaces should have non-optional domain attributes required for each MFE to func-

tion correctly and interact with others. On adding new attributes, developers must ensure consistency and reusability so other MFEs can utilize them. For example, the interface may include generic attributes–such as a list of objects with standard properties like label, value, and type–allowing each MFE to exchange data or render components without understanding the specific meaning or context of the values (Figure 32). This approach reduces coupling between MFEs while maintaining benefits such as modularity, scalability, and adaptability to new requirements.

```
type TProduct = {
  name: string;
  amount: number;
  quantity: number;
  images: string[];

  // Optional attributes
  additionalInformation: {
    label: string;
    value: string;
  }[] | undefined;
  imageUrl: string | undefined;
  deliveryAddress: TAddress | undefined;
  description: string | undefined;
};
```

Figure 32 – General product attributes are non-optional and create optional generic attributes that can be used by any type of product.

5.5.2 Nano Frontend

Category: Intra-frontends

Problem: The frontend decomposes into numerous small MFEs with few screens or fragments that do not represent a domain or subdomain of the application. Small MFEs do not justify the cost of their maintenance. Furthermore, the presence of Nano Frontends can lead to issues of high coupling and the manifestation of other anti-patterns, such as cyclic dependency

Example: In a scientific workflow platform, the UI is decomposed into excessively granular micro frontends: one for the pipeline diagram (pipeline-microfrontend), another for configuring each phase (phase-parametrization-microfrontend), and a third for listing executions (execution-table-microfrontend), as shown in Figure 33. Although these fragments interact closely and all belong to the same workflow domain, they are split into tiny independent MFEs, leading to high coupling, increased communication complexity, and duplicated effort across teams. By applying the solution, the architecture is redesigned to consolidate these small MFEs into a single, cohesive workflow MFE aligned with the domain.

Figure 33 – Page from (ANTUNES et al., 2024) with components as MFE, configuring nano frontends.

Solution: The issue of Nano Frontends arises when the definition of boundaries is inadequately and excessively granular. Adhering to Domain-driven Design (EVANS, 2004) principles is necessary to ensure an effective decomposition of MFEs. Therefore, the development team must work closely with the product team to gain a deep understanding of the domains and reflect them accurately in the architecture. To solve this issue, the architecture must be redesigned by grouping MFEs with the same domain is necessary. For minor variations within a domain, consider using templates or compo-

nent libraries. This approach avoids creating a separate MFE for each slight variation, promoting efficiency and code reuse.

5.5.3 Mega Frontend

Category: Intra-frontends

Problem: Decomposing the architecture into a few MFEs encompassing numerous screens and fragments, typically embracing more than one sub-domain, manifests this anti-pattern. The MFE inherits the challenges of a monolithic frontend, such as difficulties in testing, slow builds and deployments, high coupling between its components, lack of modularity, and limited scalability.

Example: An e-commerce system is decomposed into just two MFEs, with mfe-users related to users and mfe-shopping related to products and purchases. The latter MFE includes screens that display product listings, product details, purchase confirmations, and purchase history. Decomposing the mfe-shopping into at least two MFEs is necessary: one containing the product listing and product details screens, belonging to the product domain; and another containing the confirmation and purchase history screens, belonging to the purchase domain (Figure 34).

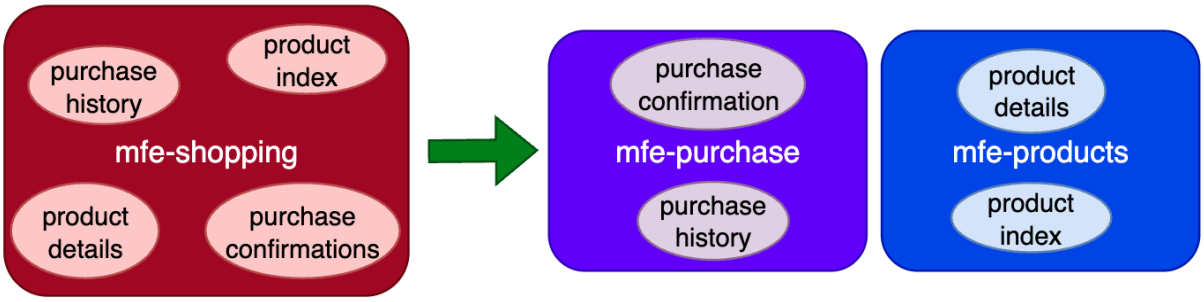

Figure 34 – Mega frontend break into two MFEs.

Solution: To avoid this problem, the development team must work closely with the product team to gain a deep understanding of the domains and reflect them accurately in the architecture. To fix this issue, the team should reevaluate the architecture and divide the MFEs into more granular units, separating functionalities into smaller and specialized MFEs based on domains. This approach helps reduce complexity, en-

hance maintainability, and foster a modular and scalable architecture.

5.5.4 No CI/CD

Category: Operation

Problem: The company lacks an automated Continuous Integration (CI) and Continuous Delivery (CD) pipeline, so developers must manually execute tests and perform deployments. This manual process becomes burdensome, especially with the potential existence of multiple MFEs. It increases development time, reduces productivity, and raises the risk of errors in the production environment.

Example: Upon releasing a new system version, a developer must conduct manual tests and ensure all unit tests pass. However, developers may skip the tests and manually deploy the changes without realizing some tests are failing, introducing bugs, which is avoidable with an automated CI pipeline (Figure 35). Even if the tests pass, there is still a risk of making mistakes during deployment, which could render the system unavailable. Automating the deployment process with CD ensures correct and consistent execution.

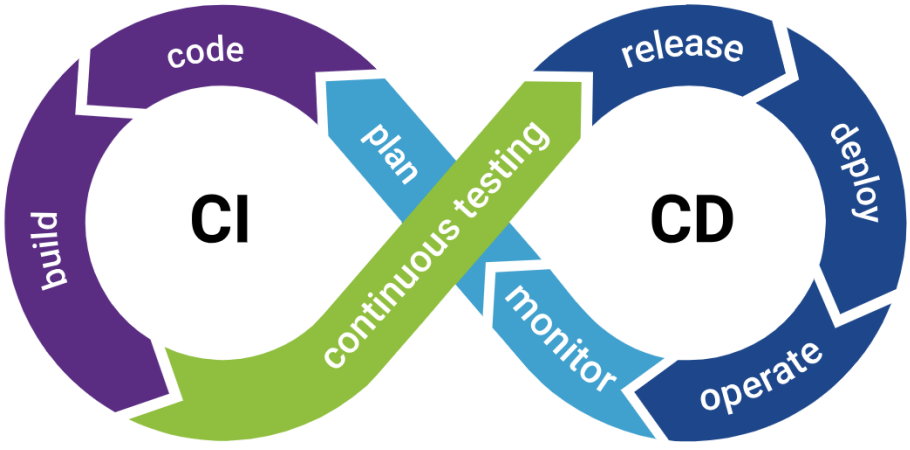

Figure 35 – CI and CD cycle. Source: <https://www.abtasty.com/resources/ci-cd/>

Solution: Implement an automated and replicable CI/CD process that extends for new MFEs, ensuring they will have automated test execution and deployment consistently and efficiently. This should be part of the Definition of Done (DoD) of the

architecture.

5.5.5 Common Ownership

Category: Development

Problem: A single team is tasked with managing all MFEs, which can occur either due to a lack of team division or when teams are segmented based on technical aspects such as data, frontend, and backend. However, one of the key benefits of MFE architecture is independence, so adopting MFE Architecture without distinct teams to operate independently negates this advantage.

Example: Consider an organization where a single centralized team is responsible for maintaining all micro frontends, including users, checkout, and products, regardless of their domain context. This setup leads to bottlenecks, reduces team autonomy, and creates friction between domain experts and technical teams. By restructuring teams according to Domain-driven Design principles, as illustrated in Figure 36, the organization creates cross-functional teams, each responsible for its micro frontends, services, and databases.

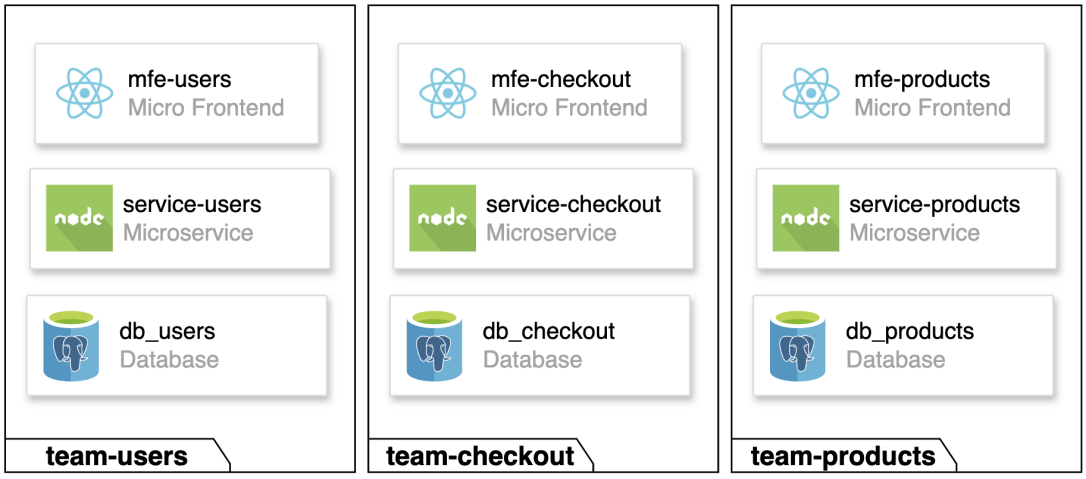

Figure 36 – Micro Frontends and Microservices grouped in cross-functional teams defined by domain.

Solution: Context should be the defining factor when structuring development teams. Therefore, defining the boundaries of teams and MFEs is essential according to Domain-driven Design (EVANS, 2004), so a team will be responsible only for MFEs

within its domain (Figure 36). Creating shared libraries can facilitate boundary definition and promote greater team independence.

6

Catalog Expansion Through a Multivocal Literature Review

Although the Rapid Review presented in Chapter 3 showed that the scientific literature does not document anti-patterns for micro frontends, developers frequently share such knowledge online through blog posts, technical documents, and videos based on their professional experience. Sharing knowledge in non-formally published sources (commonly referred to as Grey literature) is widespread among software engineering (SE) practitioners, as SE is a practitioner- and application-oriented field (GAROUSI; FELDERER; MÄNTYLÄ, 2019). However, these anti-patterns remain scattered, lack a standardized format, and require careful evaluation of credibility and relevance, especially when the authors are anonymous or their expertise cannot be verified.

To identify MFE anti-patterns proposed by practitioners in Grey Literature sources, and to summarize and standardize them, we conducted a Multivocal Literature Review (MLR) based on the guidelines proposed by Garousi, Felderer & Mäntylä (2019). Section 6.1 describes the MLR protocol, including its objective, research questions, search strategy, and data extraction process. Section 6.2 presents the findings of the MLR, while Section 6.3 discusses their implications. Section 6.4 examines the threats to the study's validity. Finally, Section 6.5 presents the newly identified anti-patterns and some improved anti-patterns based on the results.

6.1 MLR Protocol

This section presents the MLR protocol and describes our process of retrieving relevant documents and identifying new anti-patterns.

6.1.1 Goal and Research Questions

This MLR aims to expand our catalog of MFE anti-patterns by incorporating those proposed by developers in Grey Literature. In doing so, we seek to document more problems reported by MFE practitioners and enhance the quality and coverage of our catalog. To guide the search, we defined 4 RQs:

RQ1 Which Micro Frontends anti-patterns have been proposed in White and Grey Literature?

RQ2 How are Micro Frontends anti-patterns classified?

RQ3 Are the proposed Micro Frontend anti-patterns grounded in theory or based on professional experience?

RQ4 What is the profile of the authors who propose Micro Frontend anti-patterns?

RQ1 is the main research question, focused on uncovering anti-patterns reported in Grey Literature. RQ2 aims to identify existing classifications or categories that could improve the structure of our current catalog. RQ3 investigates the basis of the proposed anti-patterns, assessing whether they are grounded in theoretical knowledge or derived from professional experience. Finally, RQ4 explores the background of the authors who propose these anti-patterns, helping us understand whether they are researchers or practitioners.

6.1.2 Search String

In the search string, we included all known variations of the terms *"micro frontend"* and *"anti-pattern"*. We selected a set of control publications that address the topic to

ensure that the search string retrieved relevant publications. The posts from Shinde (2022), Mezzalira (2020), and Rappl (2024) were written by authors with experience in MFE and present anti-patterns in a structured format, including both the problem and the proposed solution. Additionally, we included our paper proposing a catalog of anti-patterns (SILVA; RODRIGUES; CONTE, 2025) to ensure the string could retrieve relevant academic publications. We refined the search string iteratively until it retrieved all control studies. The final version appears below:

> (“anti-pattern” OR “anti-patterns” OR “antipattern” OR “antipatterns” OR “anti pattern” OR “anti patterns”) AND (“microfrontend” OR “microfrontends” OR “micro frontend” OR “micro frontends” OR “micro-frontend” OR “micro-frontends”)

6.1.3 Sources

We conducted the automated search across both academic and non-academic sources to retrieve publications from white and Grey Literature. We selected the IEEE Explore[1] and the ACM Digital Library [2] for academic sources, as these digital libraries cover the most relevant journals and conference proceedings in software engineering (KITCHENHAM; BUDGEN; BRERETON, 2015). We used Google [3], Medium [4], and X (formerly known as Twitter) [5] to capture Grey Literature, encompassing the leading platforms where developers typically share technical knowledge. Additionally, we selected Google Scholar [6] to retrieve preprints of academic papers that had not yet been formally published.

1 <https://ieeexplore.ieee.org/Xplore/home.jsp>
2 <https://dl.acm.org/>
3 <https://www.google.com.br/>
4 <https://medium.com/>
5 <https://x.com>
6 <https://scholar.google.com.br/>

6.1.4 Selection Criteria

We defined one inclusion criterion (IC) encompassing publications presenting one or more MFE anti-patterns, each described with its associated problem and proposed solution(s):

- **IC1**: Document one or more MFE anti-patterns by describing their problem and proposing one or more solutions.

We designed an exclusion criteria (EC) to remove publications that do not document MFE anti-patterns or fail to describe both the problem and at least one solution; that are not written in English or Portuguese; that are not related to MFE; or for which full access was not available. Finally, we included a criterion to exclude duplicate publications. Therefore, the EC are:

- **EC1**: Does not document MFE anti-patterns by describing their problem and proposing one or more solutions;
- **EC2**: Publication is not in English or Portuguese;
- **EC3**: Publication is not about MFEs;
- **EC4**: Full access to the publication was not possible;
- **EC5**: Duplicate publication.

6.1.5 Search and Selection Process

We conducted the automated search from November 28th, 2024, to December 1st, 2024. After conducting the automated search on all selected sources, we conducted the selection process in three phases: the first filter, the quality assessment, and the second filter. The following sections describe each of these phases.

6.1.5.1 First Filter

First, we removed duplicates based on similar titles and URLs. Then, we conducted the first filter by applying the selection criteria to the publication's title, abstract, and body, in this order. If a publication was not included based on its title, we read the abstract (in the case of academic papers) or the first paragraph (in the case of Grey Literature publications) to decide whether to include it. For textual Grey Literature publications not included based on the abstract or initial paragraph, we searched for the terms "antipattern", "anti-pattern", and "anti pattern" in the body. We read the whole paragraph in which the term appeared to determine whether it satisfied IC1. If the publication was not included after these three steps, we excluded it based on EC1. For videos, we evaluated inclusion based only on their titles. In cases of doubt, we included the publication for complete evaluation during the second filter.

To avoid single-researcher bias, we first evaluated the selection criteria in collaboration with another researcher before applying the first filter to all publications resulting from the duplicate removal. The author of this thesis and the research collaborator independently applied the first filter to a sample of 19 publications. We then measured the level of agreement between the two researchers using Cohen's Kappa (COHEN, 1960) to ensure the criteria were clear and well-defined. The result indicated an almost perfect agreement ($k = 0.881$), according to the interpretation proposed by Landis & Koch (1977).

6.1.5.2 Quality Assessment

We performed the Quality Assessment (QA) to evaluate the extent to which the Grey Literature publications are valid and unrestricted from bias (GAROUSI; FELDERER; MÄNTYLÄ, 2019). The QA phase is essential for assessing publications that have not undergone peer review to ensure the quality of the results. We defined a QA checklist based on the guidelines proposed by Garousi, Felderer & Mäntylä (2019), which includes a set of questions grouped under specific criteria. However, we excluded the Impact and Novelty criteria, as our goal was not to search for novel research results. We

also discarded some questions from the remaining criteria as they were not relevant to our research goal. Each question has a set of possible answers associated with a corresponding score. The resulting in the checklist presented in Table 9.

Criteria	Questions	Possible Answer
Date	Q1: Does the post have a clearly stated date?	0: No
		1: Yes
Authority of the Producer	Q2: Has the author published other posts about Micro Frontends?	0: No
		1: Yes
	Q3: Does the author have experience in real-world Micro Frontend architectures?	0: No
		1: Yes
Methodology	Q4: Does the anti-patterns have a clearly stated origin or reference?	0: Not
		1: Yes
Impact	Q5: Does the post have positive comments, likes or have more than 5 back links?	0: Not
		1: Yes
Position w.r.t. related sources	Q6: Have key related Grey Literature or formal sources linked to or discussed?	0: Not
		1: Yes
Outlet type	Q7: What is the level of the publication's outlet control?	0: Low
		0.5: Moderate
		1: High

Table 9 – Quality Assessment checklist for Grey Literature publications.

We answered Q1 with "Yes" only if we could identify the publication's year. To answer Q2 and Q3, we searched the authors' profiles on LinkedIn [7] and reviewed their other publications related to MFE to assess their background and expertise in the field. If we could not identify the author, both questions were answered with "Not." We answered Q4 with "Yes" only if the author explicitly stated whether the anti-patterns were based on their professional experience or referenced other publications. To address Q5, we examined the publication's comments and likes and measured backlinks using a backlink checker [8]. For Q6, we verified whether the publication referenced relevant MFE literature. Finally, we classified the publications according to the types of grey literature sources in Software Engineering proposed by Garousi, Felderer & Mäntylä (2019), which define three ordered tiers based on the level of outlet control:

1. **High**: books, magazines, government reports, white papers;

[7] <https://www.linkedin.com>
[8] <https://www.seoreviewtools.com/valuable-backlinks-checker/>

2. **Moderate**: annual reports, news articles, presentations, videos, Q&A sites (such as Stack Overflow), wiki articles; and

3. **Low**: blogs, emails, tweets.

To determine which publications would proceed to the next phase, we calculated each publication's average score ranging from 0 to 1 and established a threshold of 0.5. A researcher collaborator evaluated each publication score to ensure the quality of the assessment. In cases of disagreement, we discussed the differences until reaching a consensus.

6.1.5.3 Second Filter

During the second filter, we thoroughly reviewed the 36 publications (reading them entirely or watching them in the case of videos) to decide whether to include them for data extraction, based on the selection criteria. Two researchers conducted the filtering independently and discussed differences until reaching a consensus.

6.1.6 Data Extraction

To conduct the data extraction from the selected publications, we defined the extraction form presented in Table 10. We designed the last four fields to answer the RQs, while the remaining fields aim to summarize the results. Since almost all selected publications are from grey literature (except for the paper we published proposing the second version of our catalog) the authors did not follow a structured approach for documenting the anti-patterns. Therefore, we extracted the problems and solutions by identifying relevant quotes that describe them. Additionally, we identified the names and categories of each anti-pattern. Two researchers independently performed this extraction and discussed the results to reach a consensus.

Table 10 – Quality Assessment checklist for Grey Literature publications.

Field	RQ	Expected Outcome
Title	-	Publication's title
Source		Source retrieved from
Author		Author's name and filiation
Publication type		Video, blog, post, or paper
Publication Date		Date
Author's Profile	RQ4	Researcher or practitioner
Anti-patterns origin	RQ3	Theory or professional experience
Anti-patterns categories	RQ2	List of categories
Proposed Anti-patterns	RQ1	List of anti-patterns with their names, category (if any), and quotes describing their problem and proposed solutions

6.1.7 Data Synthesis

To answer RQ1, we listed all extracted anti-patterns and unified the definition of similar anti-patterns proposed in different publications. We identified similar anti-patterns and linked them to support the creation of a consolidated description of their problems and solutions. Based on the anti-pattern description, we retained its original name or renamed them to better reflect the underlying problem. To assist in analyzing similar anti-patterns and reaching a unified description, we designed mind maps that included the consolidated anti-pattern name, the names given in each source as children nodes, and the corresponding problem and solution quotes as subnodes (see Figure 37). We then reviewed each anti-pattern and defined a final description of its problem and solutions. A senior researcher reviewed this process to ensure fidelity to the original publications.

Regarding RQ2, we listed the extracted categories. Since only our publication classified the anti-patterns, we applied this same classification to all identified anti-patterns. The categorization was also reviewed by a senior researcher. To answer RQ3, we examined the references cited in the publications to determine whether the proposed anti-patterns were grounded in the authors' professional experience or based on theoretical sources. As for RQ4, we relied on the answers to the questions under the *Authority of the Producer* criterion in the Quality Assessment checklist.

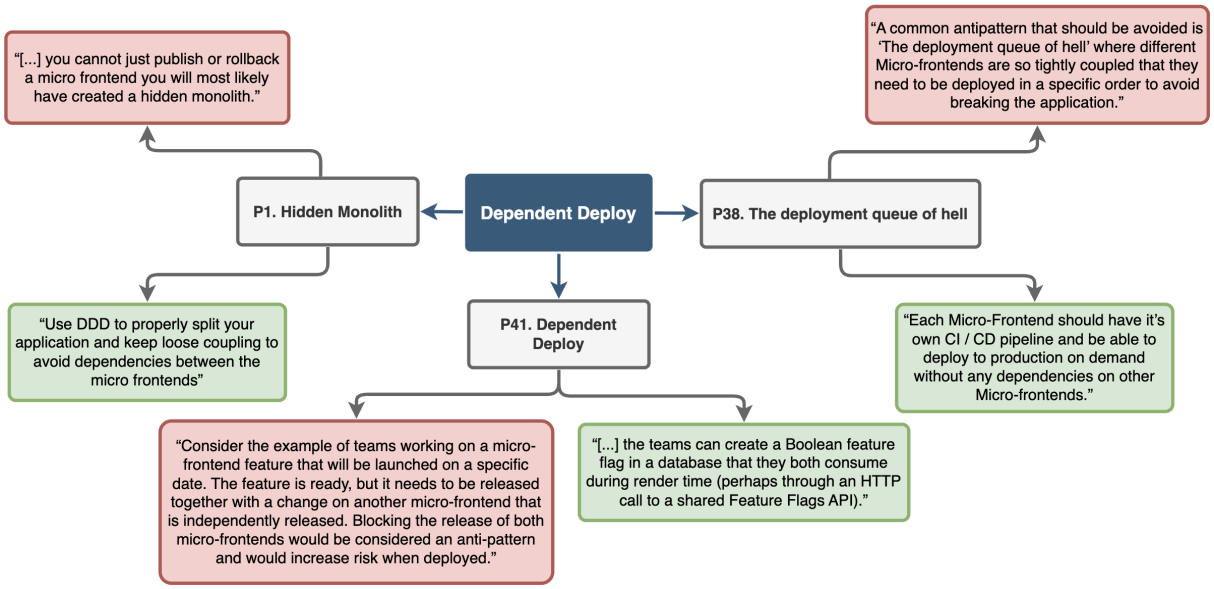

Figure 37 – Mental map summarizing all references of the Dependent Deploy Anti-pattern.

6.2 Results

In this section, we present the results of the search and selection processes, including the answers to the RQs.

6.2.1 Selected publications

Figure 38 presents the search and selection processes. The search string returned 352 publications, primarily from sources of Grey Literature: Google returned 148, Medium returned 106, X returned 13, and Google Scholar returned 52. In contrast, academic sources yielded only 3 publications, 2 from ACM and 1 from IEEE. We identified each publication using an ID in the format *PN*, where *N* represents the position in which the publication was retrieved during the search process. The full list of publications can be found at Appendix E. Among the retrieved results, we excluded 37 duplicates, resulting in 284 unique publications. The excluded duplicate publications can be found at Appendix F

During the first filter, we excluded 228 publications based on the inclusion and exclusion criteria, leaving 56 publications for the QA phase. The results of the first filter

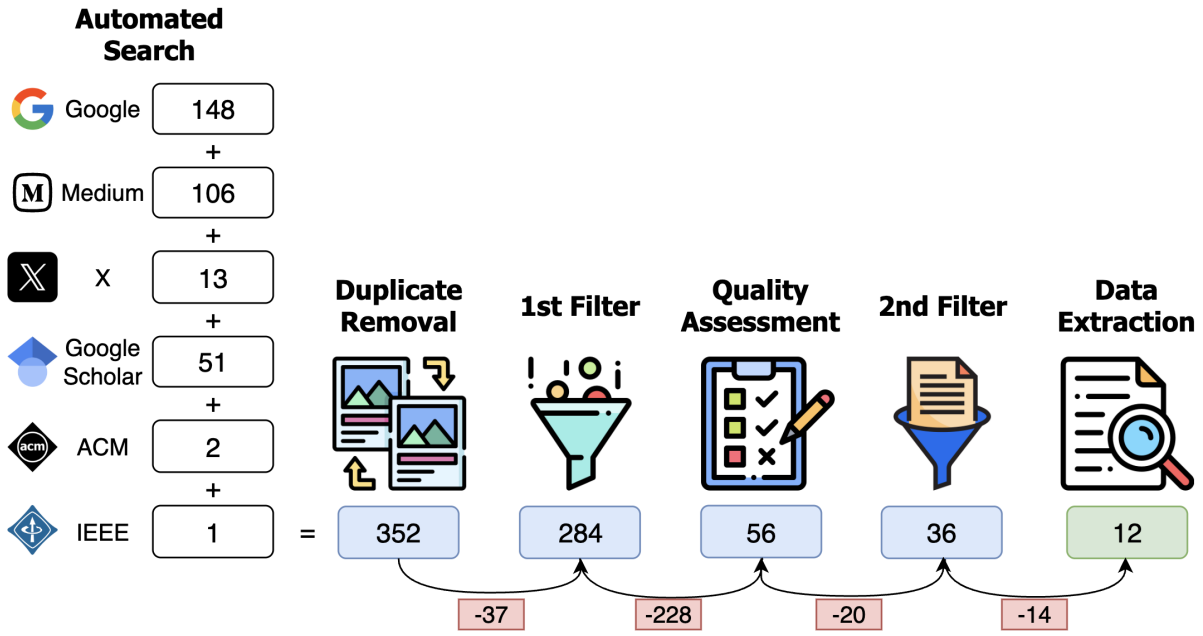

Figure 38 – Search and selection process.

can be found at Appendix G. After applying the QA criteria, only 36 publications were considered to meet the quality standards and advanced to the second filter. The results of the quality assessment can be found at Appendix H. In the second filter, we excluded 14 publications, resulting in 12 publications used for data extraction. The results of the second filter can be found at Appendix I and the data extraction forms of each selected publication can be found at Appendix J. Table 11 presents the 12 resulting publications from the selection process.

6.2.2 Publications overview

Figure 39 summarizes the distribution of publication types and years. Publications from Rappl (2024), Mezzalira (2020), Shinde (2022), and Silva, Rodrigues & Conte (2025) are the control publications we selected during the search string refinement process. The selected publications were published between 2020 and 2025. All of them are from Grey Literature, except the last one, our peer-reviewed paper proposing the MFE anti-patterns catalog. Blogs are the most common publication types (RAPPL, 2024; SHINDE, 2022; CASAS, 2020; GKAMPERLO, 2020; WESSELS, 2020). The two videos correspond to technical presentations, and the audio refers to a podcast about MFE. Among the Grey

Table 11 – Selected publications.

ID	Title	#Anti-patterns	Reference
P01	Top 10 Micro Frontend Anti-Patterns	10	(RAPPL, 2024)
P02	Microfrontends Anti-Patterns: Seven Years in the Trenches	7	(MEZZALIRA, 2020)
P07	Micro-Frontends anti-patterns by Luca Mezzalira	8	(MEZZALIRA, 2024)
P08	4 Micro-Frontend Anti-Patterns	4	(SHINDE, 2022)
P30	Chapter 4. Discovering Micro-Frontend Architectures	2	(MEZZALIRA, 2021b)
P38	Rules of Micro-Frontends	1	(CASAS, 2020)
P41	Understanding and implementing microfrontends on AWS - AWS Prescriptive Guidance	2	(FIGUS ALEXANDER GUENSCHE; MEZZALIRA, 2024)
P55	TechLead Journal: 47 - Micro-Frontends and the Socio-Technical Aspect	2	(MEZZALIRA, 2021c)
P58	Compositional Qualities of Microfrontends: The LdoD Archive	5	(RAIMUNDO, 2023)
P108	Micro-frontend “Blackbox Pattern”	1	(GKAMPERLO, 2020)
P155	Micro Front-End Architecture at Enterprise Scale	1	(WESSELS, 2020)
P309	A Catalog of Micro Frontends Anti–patterns	12	(SILVA; RODRIGUES; CONTE, 2025)

Literature sources, the most structured publications are from Figus Alexander Guensche & Mezzalira (2024), Raimundo (2023), and Mezzalira (2021b), a technical document, a master’s thesis, and a book chapter, respectively.

6.2.3 RQ1: Which Micro Frontends anti-patterns have been proposed in White and Grey Literature?

After data extraction, we identified a total of 47 MFE anti-patterns. To ensure accuracy and minimize redundancy, we thoroughly reviewed all identified anti-patterns and consolidated similar entries, resulting in 27 distinct final MFE anti-patterns. Figure 40 presents a mind map displaying the final anti-patterns grouped by category. We in-

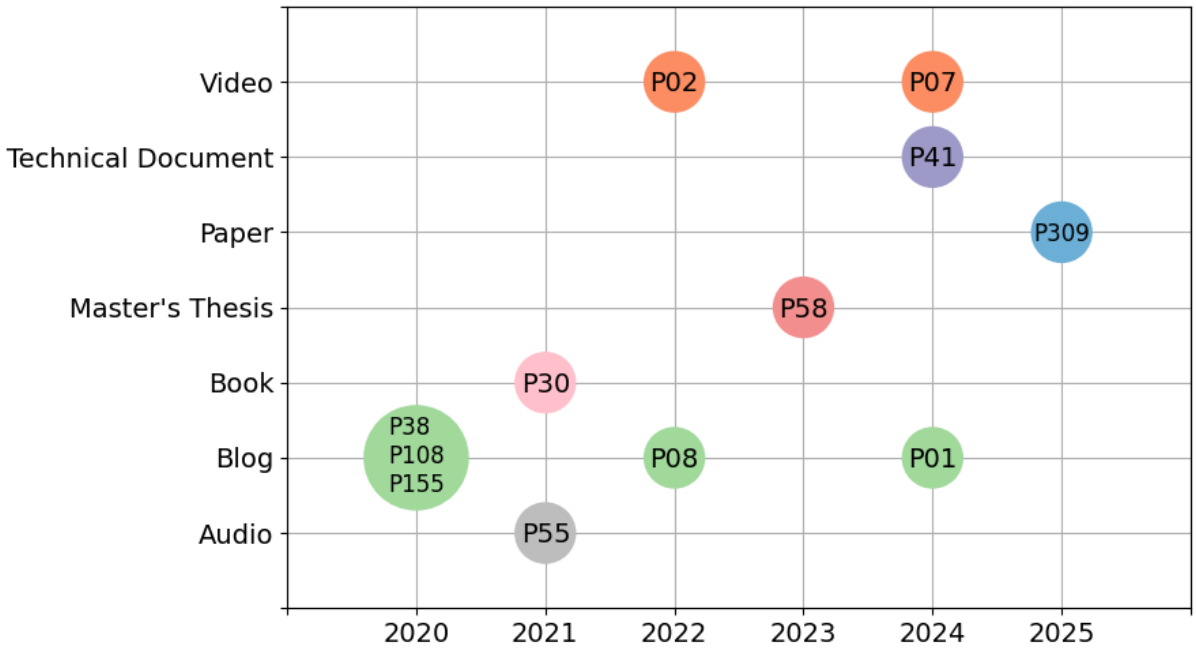

Figure 39 – Scatter plot showing the distribution of publications by type and year.

cluded two anti-patterns in the inter-frontends, operations, and development categories and eight anti-patterns in the intra-frontends category. Additionally, we refined the description of the Nano Frontend based on similar anti-patterns identified. We did not modify the Hub-like Dependency anti-pattern, as its original formulation already encompassed the problem and solution of the similar anti-pattern we identified. The complete description of the newly added anti-patterns and the improvements made based on the MLR results can be found in Section 6.5.

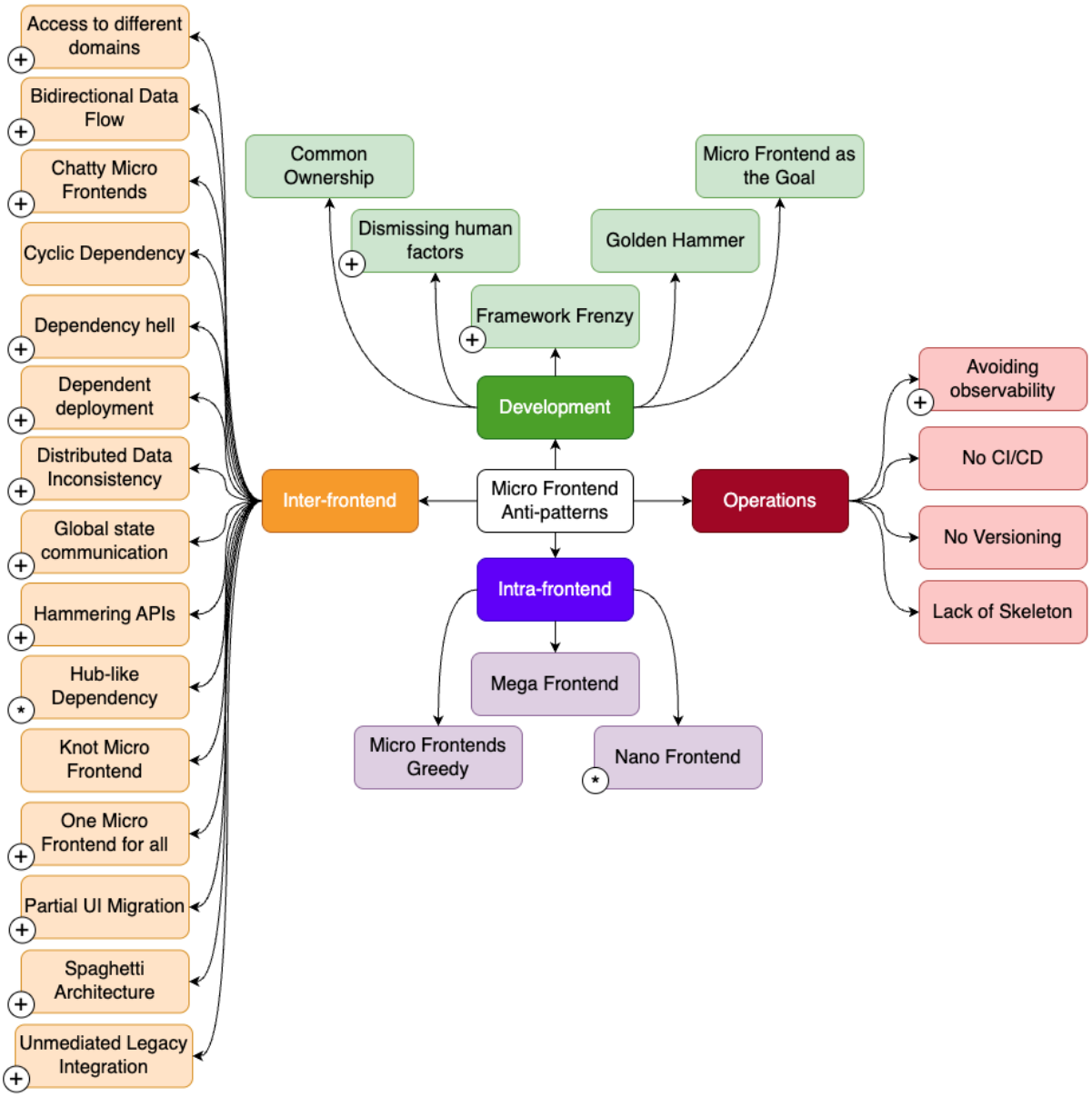

Figure 40 – Final set of MFE anti-patterns from the MLR. New anti-patterns are marked with a plus symbol (+) and updated ones with an asterisk (*).

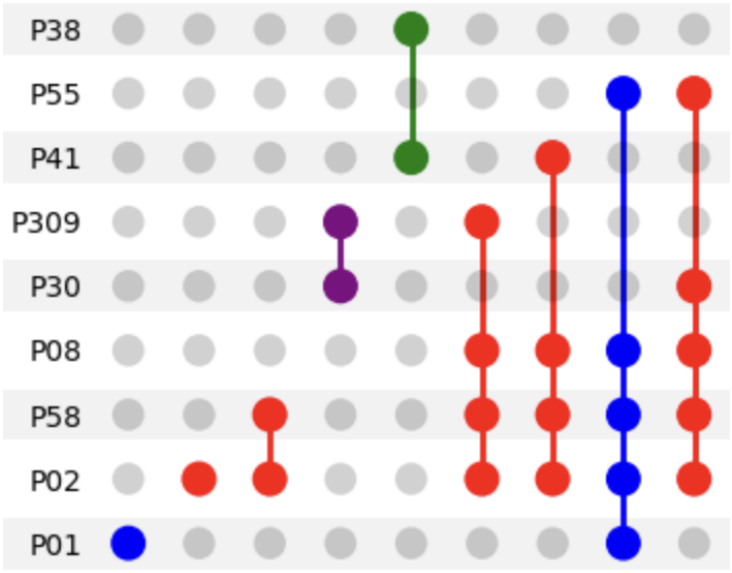

Figure 41 – Intersections between publications considering the final anti-patterns.

Table 12 presents the new and improved anti-patterns, their original names (if any), and the publications in which they were referenced. Eight anti-patterns recur frequently across several publications. These were initially proposed by Mezzalira (2020) in P2. We also selected additional publications by Mezzalira (P07, P30, P41, and P55), which propose the same anti-patterns. We did not exclude these publications as duplicates since each offers distinct explanations that help provide a deeper understanding of the anti-patterns. Figure 41 depicts the intersections of anti-patterns among the publications. The most cited anti-patterns are Framework frenzy (P01, P02, P08, P55, and P58) and Global state communication (P02, P08, P30, P55, and P58).

Table 12 – Anti-patterns original names and references.

Name	**Author's name**	**Publication**
Avoiding observability	Avoiding observability	P01
Access to different domains	-	P155
Bidirectional Data Flow	A Return Ticket, Please	P02
Chatty Micro Frontends	Chatty Frontends	P01
Dependency hell	The Dependencies Hell	P02
	Dependency Hell	P08
	Poor Dependency Management in Micro Frontends	P41
	Dependency Hell	P58
Dependent deployment	Hidden Monolith	P01
	The deployment queue of hell	P38
	Dependent Deploy	P41
Dismissing human factors	Dismissing human factors	P01
Distributed Data Inconsistency	Distributed Data Inconsistency	P01
Framework Frenzy	Framework Madness	P01
	Hydra of Lerna - (Multi-Frameworks Approach)	P02
	Multi-Frameworks Approach	P08
	Multiple technologies in the same application	P55
	Multi-Frameworks Approach	P58
Global state communication	Relax, It's just Code	P02
	Global state communication	P08
	Sharing state across Micro Frontends	P30
	Global State	P55
	Shared Global State	P58
Hammering APIs	Let's Hammer the APIs	P02
Hub-like Dependency	Tight Coupling	P30
	Hub-like Dependency	P309
Nano Frontend	Yin and Yang (Micro Frontends and Components)	P02
	Micro-Frontend Vs Component	P08
	Micro-Frontend versus Component	P58
	Nano Frontend	P309
One Micro Frontend for all	-	P108
Partial UI Migration	Bye Bye big-bang - Iterative deployment	P07
Spaghetti Architecture	Spaghetti Architecture	P01
	Tight Coupling	P01
Unmediated Legacy Integration	The Swiss Army Knife	P02
	Integration Bottleneck Anti-Pattern	P58

6.2.4 RQ2: How are Micro Frontends anti-patterns classified?

The Grey Literature publications did not follow a consistent structure for documenting anti-patterns, and none included categorization. The only publication that presents anti-pattern categories is our paper, P309. Based on it, the categories of MFE anti-patterns are:

1. **Intra-fronted category**: considers a single MFE component and its design.
2. **Inter-frontend category**: considers the structural division and communication involving two or more MFEs.
3. **Operations category**: related to the operational practices and continuous maintenance of the application.
4. **Development category**: related to the development team and their decisions around the architecture.

6.2.5 RQ3: Are the proposed Micro Frontend anti-patterns grounded in theory or based on professional experience?

All identified anti-patterns are grounded in professional experience. The same individual authored publications P02, P07, P41, and P55: Luca Mezzalira, an active practitioner in the MFE community who also wrote a book on MFE (MEZZALIRA, 2021a). In addition, the anti-patterns presented in P58 reference those from P07, meaning Mezzalira contributed to five resulting publications. The author of P01, Florian Rappl, is also an active practitioner and has likewise written a book on MFE (RAPPL, 2021). We consider the anti-patterns proposed in our previous work (SILVA; RODRIGUES; CONTE, 2025) grounded in professional experience. Although inspired by MS anti-patterns, they are mainly based on our MFE professional experience. The remaining authors all have hands-on experience working with MFE, and the proposed anti-patterns are based on that practical knowledge. This indicates that the catalog reflects issues developers encounter in real-world architectures and offers solutions derived from practice.

6.2.6 RQ4: What is the profile of the authors who propose Micro Frontend anti-patterns in Grey Literature?

Figure 42 presents a pie chart showing the distribution of author profiles. When a publication had more than one author, we considered the profile of the majority of the authors. The majority of authors are practitioners, accounting for 77.8% of the total. Only two authors are researchers: one is ourselves, and the other is a master's candidate who references anti-patterns drawn from a Grey Literature publication. These results demonstrate that practitioners are predominantly proposing MFE anti-patterns but still require consolidation and validation through formal studies, underscoring the contribution of this research.

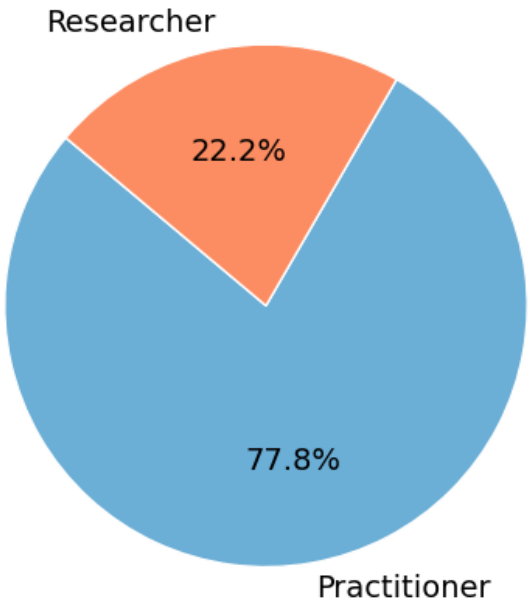

Figure 42 – Pie chart showing the distribution of author profiles.

6.3 Discussion

This study identified 352 publications through the automated search process. After applying the selection criteria, we selected 12 publications for data extraction. From these, we extracted 47 anti-patterns, which were reviewed and consolidated to eliminate overlaps. This resulted in 27 distinct final anti-patterns, comprising both new additions

and improved descriptions of previously cataloged ones. The methodological processes we followed to reach these results provide evidence of the relevance and rigor of our findings.

Among the resulting anti-patterns, the *Inter-Frontend* category increased from 3 to 15 anti-patterns, which may indicate that the most common issues in MFE are related to decisions about composition and communication. Additionally, two anti-patterns were added to the *Development* category and one to the *Operations* category, thereby introducing more problems related to the development team and the operational practices employed to maintain the architecture. No anti-patterns were added to the *Intra-Frontend* category; however, the Nano Frontend anti-pattern is one of the most cited, indicating that, when considering a single MFE, the most recurring problem is defining MFEs that are too small. The identification of anti-patterns similar to those we had previously proposed, such as the Nano Frontend and Hub-like Dependency, further strengthened the validity of the catalog.

Since no other author proposing MFE anti-patterns has structured them into categories, our work stands out by introducing a systematic organization. We used the categories of our original catalog to group the newly identified anti-patterns, demonstrating their applicability. As a result, these categories can now serve as a framework for organizing and classifying future anti-patterns, providing a solid foundation for researchers and practitioners.

During the data extraction of the proposed solutions, some of them may inadvertently lead to the emergence of other anti-patterns. For example, when implementing the solution of the Access to different domains Anti-patterns, developers may create a centralized state to store the Federation API responses, leading to the Global state communication Anti-pattern. These findings underscore the need for further research to investigate the inter-dependencies among anti-patterns and to assess whether the application of the anti-pattern solutions might unintentionally introduce new architectural problems.

The main implications of these results are the consolidation of knowledge scattered across grey literature into a structured and comprehensive catalog, addressing

the fragmentation of MFE anti-pattern discussions on the internet. Incorporating anti-patterns drawn from grey literature into the catalog also fosters greater collaboration, as we can now engage directly with the original authors to encourage their contributions and validations. This collaborative potential strengthens the catalog as a living resource that evolves alongside the community's needs and innovations. Adding new and improving anti-patterns expands the catalog's coverage and positions it as a valuable reference point for advancing research and practice in the MFE domain.

6.4 Threats to Validity

We considered the threats to validity outlined by Ampatzoglou et al. (2019), who categorize them into threats to study selection, data, and research validity.

Study Selection Validity: This category encompasses threats that can influence the search and selection processes. One such threat is the *Selection of Digital Libraries*, which refers to the risk of choosing sources that are either too specific, too broad or lack credibility. We mitigated this by selecting the central scientific databases commonly used in Software Engineering (IEEE Xplore and ACM DL) and general-purpose databases, such as Google and Google Scholar, to cover Grey Literature. Another threat is the *Construction of the Search String*, which involves potential issues when researchers formulate the search string. To address this, we defined a set of control publications representing the expected retrieval scope and iteratively refined the search string based on them. *Study Inclusion/Exclusion Bias* refers to errors that may occur during selection. We mitigated this by carefully analyzing the inclusion and exclusion criteria, measuring agreement between two researchers to ensure clarity, and reviewing the results of each selection phase collaboratively to ensure quality. Finally, given the unstructured nature of Grey Literature, we avoided excluding relevant documents based solely on their titles by examining the first paragraph as a proxy for the abstract and scanning for keywords to assess relevance.

Data Validity: This category includes threats arising during data extraction and analysis. One such threat is *Data Extraction Bias*, which refers to issues that may occur in

the extraction phase. To mitigate this, we defined an extraction form aligned with all RQs and specified the expected data for each field, ensuring that the extraction process would capture relevant and structured information, thereby facilitating later analysis. Additionally, one researcher performed the initial data extraction, while another reviewed the quotes related to the anti-patterns' problems and solutions. Another threat is *Unverified Data Extraction*, where external or internal evaluators do not review extracted data. To address this, a senior researcher reviewed the extracted problems and solutions, as well as the synthesis into unified definitions, to ensure that the final anti-patterns accurately reflected their original descriptions.

Research Validity: This category encompasses threats that can arise across all phases of the study and concern the overall research design. One such threat is *Repeatability*, which refers to the ability to replicate the secondary study. To mitigate this, we involved three researchers throughout the MLR, thoroughly documented the protocol, and made all results publicly available. Another threat is the *Selection of Research Method*, which relates to choosing a method that does not align with the study's objectives. Given that, aside from our work, no scientific publications had proposed anti-patterns, we selected the Multivocal Literature Review (MLR) as the research method to enable the inclusion of Grey Literature results. We also followed the guidelines provided by (GAROUSI; FELDERER; MÄNTYLÄ, 2019) to support the planning and execution of the study.

6.5 Catalog's improvement

The following subsections present the newly added anti-patterns incorporated into the catalog and a new version of the Nano Frontend anti-pattern, which was refined based on similar instances found in the literature.

6.5.1 Avoiding observability

Category: Operations

Problem: No observability was implemented anywhere. You can not debug an error or identify the micro frontend that originated it because no performance metrics or error logs were implemented.

Solution: Collect metrics and traces and implement centralized logs to debug a problem and identify its cause efficiently.

6.5.2 Access to different domains

Category: Inter-frontend

Problem: A micro frontend that directly accesses APIs from multiple business domains leads to tightly coupled and hard-to-maintain architectures. Over time, this approach results in unmanageable dependencies, hinders API deprecation, and introduces issues such as over-fetching and cache synchronization problems.

Solution: Apply the principle of API Federation to expose data from multiple domain APIs through a single, centralized, and strongly typed interface. Implement a local state that is updated by a unified gateway using GraphQL, which defines the available `Query`, `Mutation`, and `Subscription` interfaces to interact with the backend and combines all APIs into a single federated API that all micro front-ends can consume. MFEs can consume these interfaces by sending queries, mutations, and subscriptions to their local state, which communicates with the backend through the unified GraphQL API, acting as the single point of access to all backend APIs.

6.5.3 Bidirectional Data Flow

Category: Intra-frontend

Problem: Bidirectional communication between the host application (container) and remote micro frontends.

Solution: Adopt a unidirectional data flow inspired by patterns like Flux or Model-View-Intent (MVI). In this model, data flows in a single direction: Action $\rightarrow$ State Update $\rightarrow$ View Update. This reduces coupling, simplifies debugging, and makes the

system more predictable.

6.5.4 Chatty Micro Frontends

Category: Inter-frontend

Problem: Excessive communication between fragments caused by broadcasting events for every action, leading to performance overhead.

Solution: Emit only meaningful events and introduce them gradually, based on the presence of interested consumers. Avoid broadcasting events for every action. This approach reduces noise, minimizes coupling, and helps maintain a clear and intentional event-driven architecture.

6.5.5 Dependent deployment

Category: Inter-frontend

Problem: Micro frontends become so tightly coupled that their deployment requires coordination across multiple teams. In some cases, one MFE cannot be deployed or rolled back without affecting another. In others, features must be released simultaneously across MFEs, forcing teams into synchronized release cycles. This leads to deployment queues, where MFEs must be deployed in a specific order to avoid breaking the application, increasing operational complexity and risk.

Solution: Promote independent deployment for each micro frontend. Ensure that every MFE has its own CI/CD pipeline and can be deployed or rolled back without relying on other MFEs. When a coordinated release is necessary, use feature flags that all relevant MFEs can check at runtime, allowing features to be toggled on or off independently of the deployment timeline. Additionally, apply Domain-Driven Design principles to define proper boundaries between MFEs, reducing tight coupling and enabling true autonomy at the deployment level.

6.5.6 Dismissing human factors

Category: Development

Problem: Teams pursue technical goals and deadlines without considering the well-being, morale, or work-life balance of their members. Micro frontends are misused as a technical solution, while their true potential to improve team autonomy and structure is neglected.

Solution: Strengthen teams and avoid central management as much as possible.

6.5.7 Distributed Data Inconsistency

Category: Inter-frontend

Problem: Inconsistency in the data shared across micro frontends due to lack of change propagation. If the original data changes, those changes must be propagated. However, if a micro frontend holds a replicated version of the data, it's unclear whether it is allowed to update it. This causes divergence between the original and the duplicate.

Solution: Keep data where it belongs. Do not allow other micro frontends to access the data directly; instead, expose it indirectly through attributes or properties.

6.5.8 Framework Frenzy

Category: Development

Problem: Introducing multiple frameworks without a real need, disregarding the complexity of communication between components built with different technologies.

Solution: Whenever possible, use a single framework across all micro frontends. If there is an opportunity to standardize, prefer consistency over unnecessary flexibility.

6.5.9 Global state communication

Category: Inter-frontend

Problem: Using shared states violates the principle of segregation, compromising the independence of each micro frontend and increasing coupling.

Solution: Each micro frontend should have its own event store. To enable communication, use an event emitter-based approach instead of sharing state directly.

6.5.10 Nano Frontend

Category: Intra-frontend

Problem: A micro frontend is created with only a few screens or fragments. This typically occurs when teams confuse micro frontends with UI components, resulting in fragmented and ineffective architectural decomposition.

Solution: Define micro frontend boundaries based on business subdomains, following Domain-Driven Design principles. Micro frontends should encapsulate cohesive sets of features aligned with domain-level concerns. Avoid creating MFEs for isolated UI fragments or for purely technical reasons. Each MFE should represent a meaningful subdomain within the organization.

6.5.11 One Micro Frontend for all

Category: Inter-frontend

Problem: A single micro frontend is created and imported by all other MFEs.

Solution: Use the blackbox pattern to encapsulate shared functionality. In this model, the component exposes a clear input, renders itself into the DOM with its own internal workflow, and provides an output that other micro frontends can consume. This preserves the independence of MFEs while still enabling interoperation through well-defined boundaries.

6.5.12 Partial UI Migration

Category: Operations

Problem: Extracting a portion of the UI to create a micro frontend and embedding it back into the legacy monolith often slows down the system and provides no real benefit from adopting micro frontends.

Solution: Follow a path-based migration strategy, which allows gradual migration of frontend segments by switching versions through URL base paths. This enables progressive adoption of micro frontends without entangling them with the monolith.

6.5.13 Spaghetti Architecture

Category: Inter-frontend

Problem: Micro frontends are structured in a disorganized way, leading to a tangled web of dependencies and interactions. This high degree of coupling between MFEs makes the system difficult to scale, test, and maintain. Maintain loose coupling between micro frontends.

Solution: Avoid direct references that rely on internal implementation details of other MFEs, such as URLs, module paths, or internal names. Instead, design MFEs to interact through well-defined contracts, events, or shared interfaces.

6.5.14 Unmediated Legacy Integration

Category: Intra-frontend

Problem: Integrating a legacy system into a micro frontend architecture without proper isolation often leads to architectural misalignment and increased complexity. Legacy systems may use incompatible technologies or communication mechanisms that do not fit the micro frontend model. A common mistake is to modify or extend the main integration layer of the application to accommodate these differences, introducing tight coupling, pollution of domain boundaries, and long-term maintainability issues.

Solution: Introduce an Anti-Corruption Layer (ACL) between the micro frontend application and the legacy system. This layer acts as a translator, isolating the legacy system and ensuring that its communication model and domain concepts do

not leak into the modern architecture. By encapsulating the integration logic in a dedicated boundary, teams can preserve the integrity of the micro frontend system without modifying its core integration layer.

7

FINAL CATALOG

After collecting feedback from participants and incorporating new anti-patterns from grey literature, we evolved our catalog by introducing a new template designed to add more details and simplify descriptions for easier understanding. Consequently, we refined all anti-pattern entries to align with and fully populate the new template and updated the web application to support it. Section 7.1 presents the new template and explains how we addressed the additional fields. Section 7.2 describes the updated page format of the web application showcasing the new template. Finally, Section 7.3 presents the final version of the catalog, which documents all anti-patterns using the new structure.

7.1 Template evolution

After improving the anti-patterns based on the results of the Personal Opinion Survey (see Chapter 4), some anti-patterns now have lengthy descriptions, which were identified as an issue by the students who used the catalog in the controlled experiment (see Chapter 5). Additionally, after the MLR (see Chapter 6), we discovered new anti-patterns and relationships between similar ones. Considering the catalog's growth in both quantity and complexity, we decided to adopt an extended template to document them. Based on the Full AntiPattern Template (BROWN et al., 1998), the general template adopted in the C2 Wiki repository [1], and the template defined by Brada & Picha

[1] <https://wiki.c2.com/?AddAntiPatternToTheAntiPatternsCatalog>

(2019) to document Software Process anti-patterns, we defined an anti-pattern template that includes the following fields to structure the description of our anti-patterns:

- **Name**: the original or most commonly name adopted to refer the anti-pattern;
- **Also Known As**: other names (aliases) under which the anti-pattern is known, if any (optional);
- **Category**: category used to group anti-patterns;
- **Problem**: description of a problem and a bad solution commonly adopted;
- **Symptoms and Consequences**: identifiable phenomena signifying the presence of the anti-pattern, describing the negative effects of the problem;
- **Solution**: the improved solution that should be implemented to more effectively address the identified problem;
- **Resulting Context**: what happens when you implement the solutions, good consequences, and possible bad ones that must be analyzed; and
- **Example**: how the anti-pattern has occurred in real world, may include diagrams and illustrations;
- **Solution Pitfalls**: potential problems or other anti-patterns that can occur after implementing the solution (optional);
- **Related Anti-Patterns**: the more generic or specific variants of the same anti-pattern, opposite extremes of the same bad practice, patterns sharing several similar symptoms, etc. (optional);
- **References**: references to literature in which a description of the particular anti-pattern was found.

To address the new fields, we restructured the problem and solution descriptions, adding additional information based on the MLR results. The **Also Known As** field was populated based on similar anti-patterns identified during the MLR. The original **Problem** field was split into two, **Problem** and **Symptoms and Consequences**, to

simplify the problem definition and highlight its consequences. Similarly, we split the original **Solution** field into **Solution** and **Resulting Context** to describe the solution and its benefits or risks separately. We addressed the **Example** field of the new anti-patterns added after the MLR by including examples from their original publications or by creating examples based on the identified problem and solution. We also added the **Solution Pitfalls** field to complement the anti-pattern description and indicate which anti-patterns may arise after the solution is implemented. The **Related Anti-Patterns** field includes anti-patterns that address similar problems, or that may emerge after solution implementation. Finally, the **References** field provides links to all publications citing the anti-pattern based on the MLR results. The simplified descriptions will make the anti-patterns easier to understand, and the connections between them will enhance the catalog's overall usability.

7.2 Web application evolution

To present the anti-patterns using the new template, we made several updates to the web application. We also incorporated some feedback provided by students during the controlled experiment. First, we added the new fields to the anti-pattern details page, as shown in Figure 43. We also updated the JSON file format used to persist the anti-patterns and guidelines, enabling the addition of new anti-patterns according to the updated template. Finally, to help novice developers understand key MFE concepts, we added a page containing the presentation used to train students during the controlled experiment. Figure 44 shows the page with the MFE training presentations.

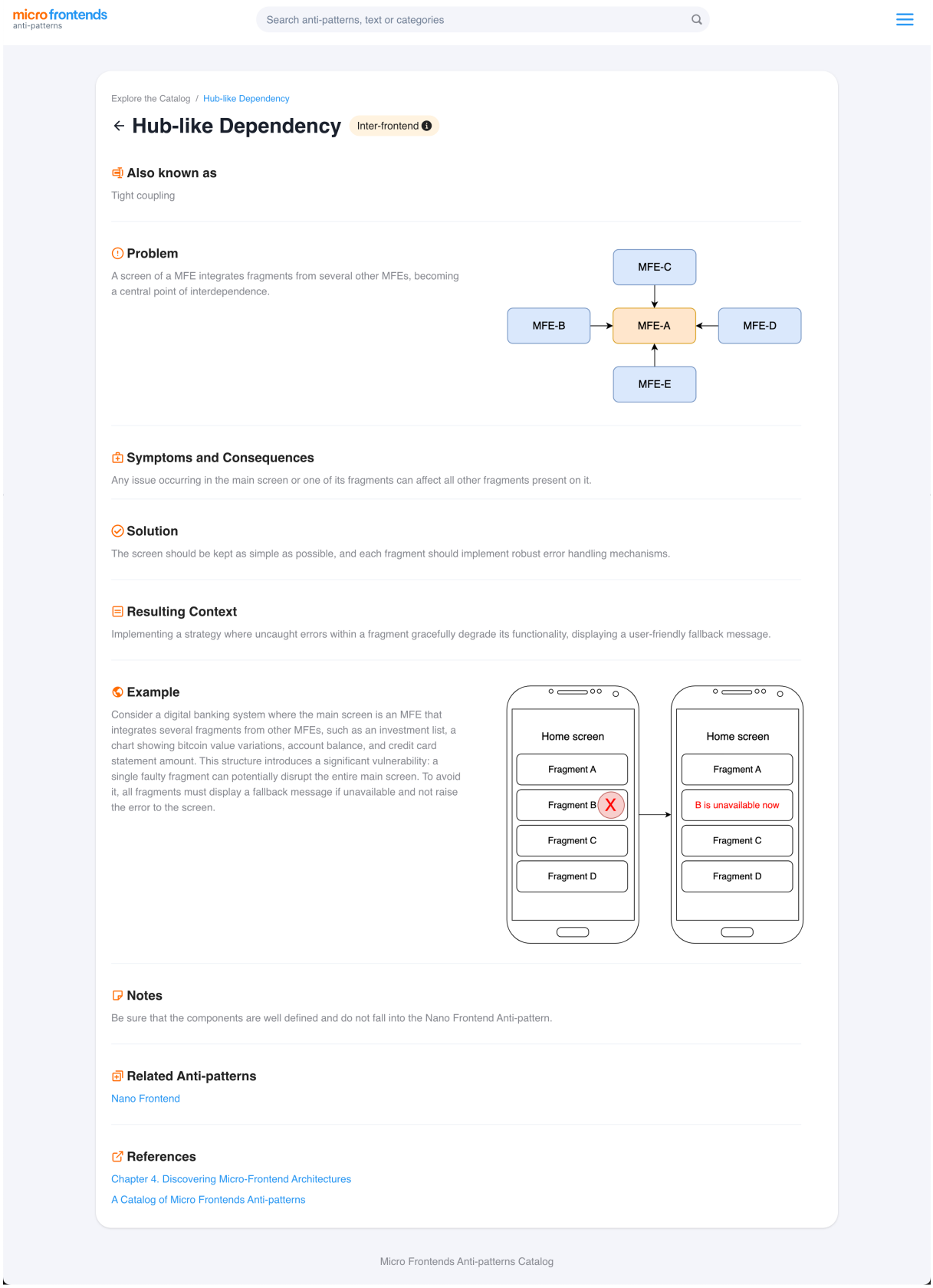

Figure 43 – Anti-pattern details page on the web application presenting the Hub-like Dependency in the new template.

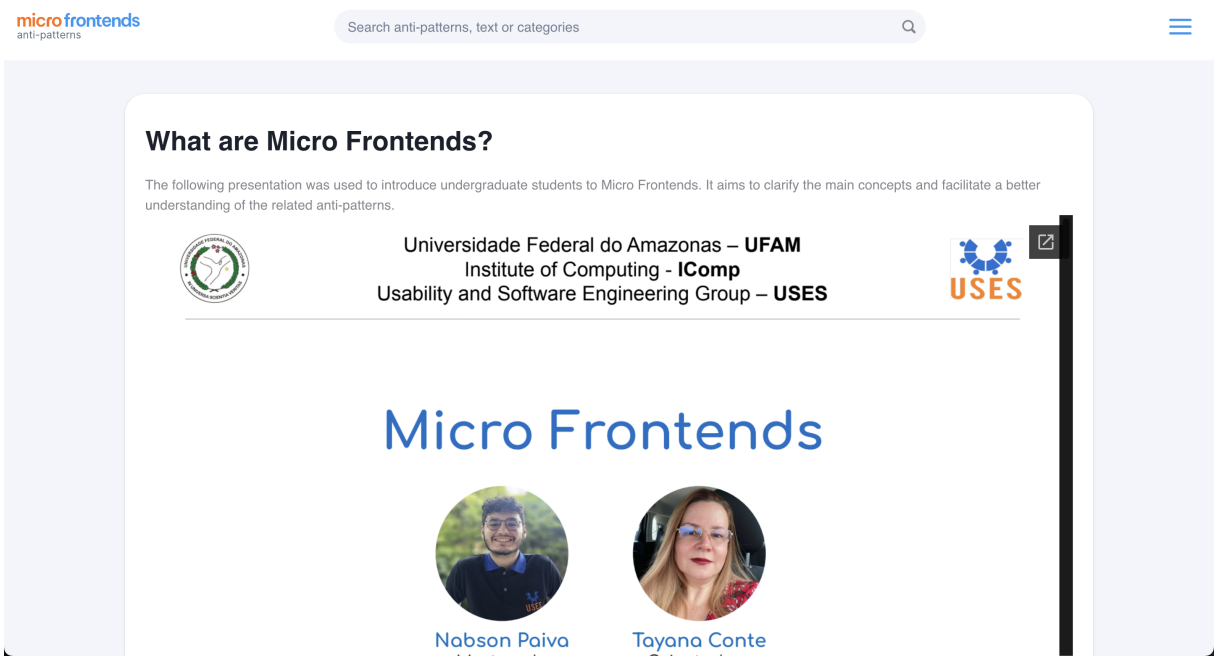

Figure 44 – Page with a presentation explaining what micro frontends are.

7.3 The Final Micro Frontends Anti-patterns Catalog

In this section, we present the final catalog of MFE anti-patterns, with each anti-pattern described in one table structured based on the new template.

7.3.1 Avoiding observability

Also Known As: -

Category: Operations

Problem: No observability is implemented anywhere.

Symptoms and Consequences: You can not debug an error or identify the micro frontend that originated it because no performance metrics or error logs were implemented.

Solution: Collect metrics, traces, and implement centralized logs.

Resulting Context: Developers can debug a problem and identify its cause efficiently by consulting traces and logs from all MFEs in a centralized way.

Example: Consider a system with three micro frontends: MF A, MF B, and MF C. Each performs user interactions, backend requests, and updates its internal state. However, none of them send logs, errors, or performance metrics to a centralized logging system, as illustrated in Figure 45. When a production issue occurs, such as a failed API call or a broken UI interaction, developers cannot determine which micro frontend caused the problem because no data is available in the logging layer. This lack of observability leads to delayed debugging, frustrated users, and increased operational costs. After applying the solution, all MFEs emit structured logs, metrics, and traces to a centralized observability layer. As a result, developers can efficiently identify the source of issues, monitor performance trends, and proactively resolve problems.

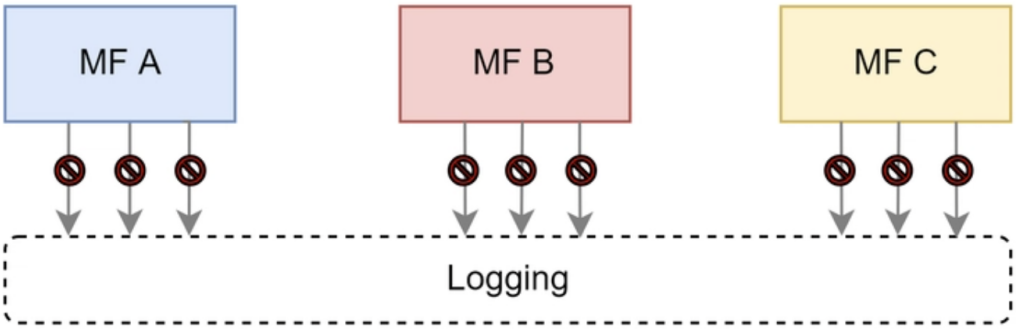

Figure 45 – MFEs with non centralized log stream by Rappl (2024).

Solution Pitfalls: -

Related Anti-Patterns: -

References: Rappl (2024)

7.3.2 Access to different domains

Also Known As: -

Category: Inter-frontend

Problem: A micro frontend directly accesses APIs from multiple business domains.

Symptoms and Consequences: The problem leads to tightly coupled and hard-to-maintain architectures. Over time, this approach results in unmanageable dependen-

cies, hinders API deprecation, and introduces issues such as over-fetching and cache synchronization problems.

Solution: Apply the principle of API Federation to expose data from multiple domain APIs through a single, centralized, and strongly typed interface. Implement a local state that is updated by a unified gateway using GraphQL, which defines interfaces to interact with the backend and combines all APIs into a single federated API that all micro front-ends can consume.

Resulting Context: MFEs can consume these interfaces by sending queries, mutations, and subscriptions to their local state, which communicates with the backend through the unified GraphQL API, acting as the single point of access to all backend APIs.

Example: Consider a system where the product MFE directly queries multiple backend APIs, such as the employees domain, assets domain, billing domain, and user domain, tightly coupling the frontend to each domain's implementation. This results in a brittle architecture, making it challenging to update APIs and leading to issues like over-fetching and inconsistent caching across domains. After applying the solution, the system introduces a GraphQL API as a federated API layer, as illustrated in Figure 46. The MFEs interact only with the GraphQL API, sending `queries`, `mutations`, and `subscriptions` to a single point of access. The federated API aggregates and orchestrates calls to the underlying domain APIs, simplifying the frontend architecture, reducing coupling, and ensuring that MFEs only retrieve the data they need.

Solution Pitfalls: Developers must consider the Global state communication Anti-pattern consequences after implementing the solution.

Related Anti-Patterns: Global state communication

References: Wessels (2020)

7.3.3 Bidirectional Data Flow

Also Known As: A Return Ticket, Please

Category: Inter-frontend

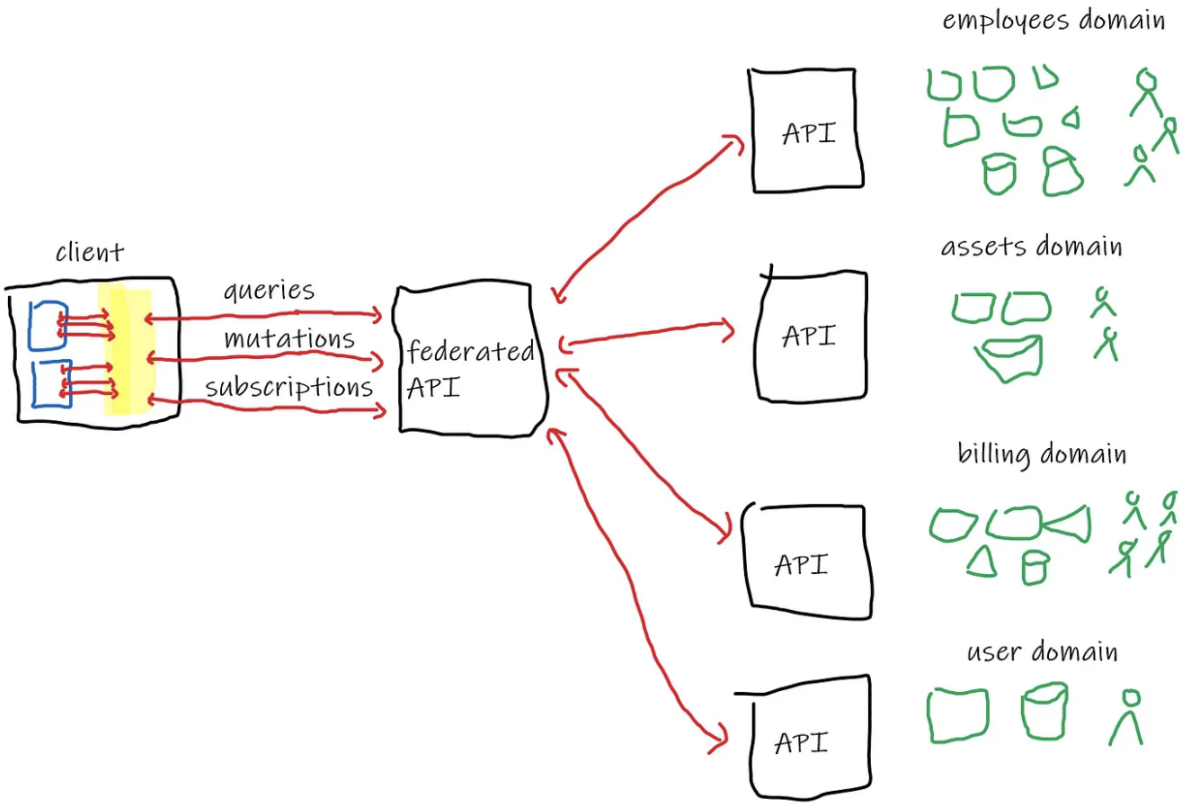

Figure 46 – Access to different domains through a Federated API. Source: Wessels (2020).

Problem: Bidirectional communication between the host application (container) and remote micro frontends.

Symptoms and Consequences: Difficult to maintain when it happens across the entire application.

Solution: Adopt a unidirectional data flow inspired by patterns like Flux or Model-View-Intent (MVI). In this model, data flows in a single direction: Action → State Update → View Update.

Resulting Context: Unidirectional data flow reduces coupling, simplifies debugging, and makes the system more predictable.

Example: Consider a dashboard application where the host container and several remote micro frontends (such as analytics, notifications, and user profile widgets) continuously exchange data back and forth. For example, when the analytics MFE updates a filter, it sends data to the host, which then propagates it back to the notifications MFE and others. Over time, this bidirectional communication creates complex dependencies,

making it difficult to trace the origin of state changes or debug unexpected behaviors. After applying the solution, the team restructures the system to follow a unidirectional data flow pattern inspired by the Model-View-Intent (MVI) design pattern, as illustrated in Figure 47. In this pattern, the user interacts with the system, generating actions that flow through the intent and model layers, thereby updating the model's state. The view then observes these state changes and updates the UI accordingly.

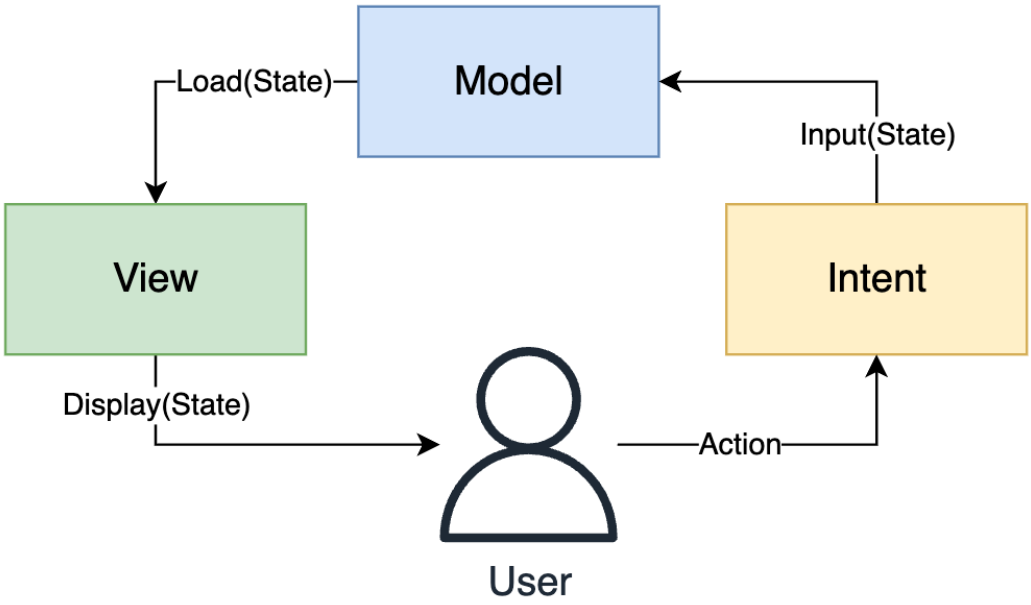

Figure 47 – Unidirectional data flow following the Model-View-Intent (MVI) pattern.

Solution Pitfalls: -

Related Anti-Patterns: -

References: Mezzalira (2020)

7.3.4 Chatty Micro Frontends

Also Known As: Chatty Frontends

Category: Inter-frontend

Problem: Excessive communication between fragments caused by broadcasting events for every action.

Symptoms and Consequences: Performance overhead by broadcasting several events.

Solution: Emit only meaningful events and introduce them gradually, based on the presence of interested consumers. Avoid broadcasting events for every action.

Resulting Context: Publishing only meaningful events reduces noise, minimizes coupling, and helps maintain a clear and intentional event-driven architecture.

Example: Consider that a search MFE broadcasts every keystroke event, causing unnecessary load on the analytics and recommendation MFEs, which listen but do not need to react to such fine-grained updates. This leads to increased performance overhead and unnecessary processing. After applying the solution, the development team revises the event system to emit only meaningful events, such as "search submitted" or "filter applied," and only when there are known consumers registered to handle those events. This change reduces event noise, improves performance, and simplifies the event-driven architecture, making the system easier to understand and maintain.

Solution Pitfalls: -

Related Anti-Patterns: -

References: Rappl (2024)

7.3.5 Common Ownership

Also Known As: -

Category: Development

Problem: A single team is tasked with managing all MFEs.

Symptoms and Consequences: Can occur either due to a lack of team division or when teams are segmented based on technical aspects such as data, frontend, and back-end. Adopting MFE without distinct teams compromises the intended independence of the architecture.

Solution: Define the boundaries of teams and MFEs according to Domain-driven Design (DDD). Creating shared libraries can facilitate boundary definition and promote greater team independence.

Resulting Context: Each team will be responsible only for MFEs within its domain and the MFEs will share components and code through shared libraries.

Example: Consider an organization where a single centralized team is responsible for maintaining all micro frontends, including users, checkout, and products, regardless of their domain context. This setup leads to bottlenecks, reduces team autonomy, and creates friction between domain experts and technical teams. By restructuring teams according to Domain-driven Design principles, as illustrated in Figure 48, the organization creates cross-functional teams, each responsible for its micro frontends, services, and databases.

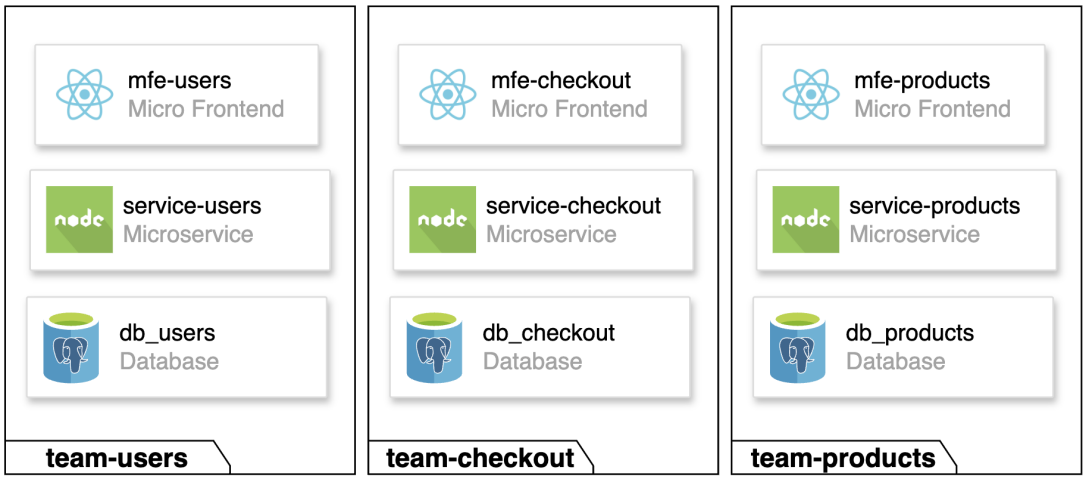

Figure 48 – Micro Frontends and Microservices grouped in cross-functional teams defined by domain.

Solution Pitfalls: The shared libraries must be carefully handled to avoid Dependency hell.

Related Anti-Patterns: Dependency hell

References: Silva, Rodrigues & Conte (2025)

7.3.6 Cyclic Dependency

Also Known As:

Category: Inter-frontend

Problem: Two or more MFEs directly or indirectly depend on each other (Figure 49).

Symptoms and Consequences: High coupling between screens and fragments, compromising MFEs' independence and modularity. Thus, changes in one MFE require

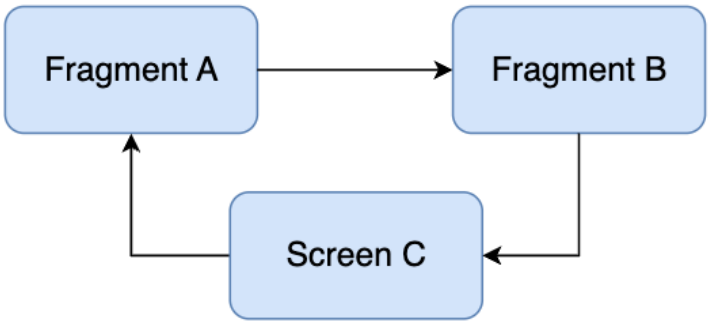

Figure 49 – Cyclic communication between fragments on the same screen.

coordination with the others. Circular dependencies lead to challenges in a system's maintenance and evolution, compromising agility and the ability to scale developments efficiently.

Solution: Implement event-based communication, which removes the need for direct dependencies between MFEs. Instead, interactions are handled indirectly via a centralized event store.

Resulting Context: On implementing the Publish-Subscribe (Pub-Sub) pattern, an MFE can publish an event to the browser, allowing other MFEs to subscribe and respond when the event occurs. To ensure consistency and reduce errors, it is recommended to centralize the the the event definitions in a shared library.

Example: Consider an e-commerce application that features a product details screen implemented in mfe-products. The screen integrates three fragments: one displaying the shopping cart from "mfe-checkout," another showing product recommendations from "mfe-recommender," and a third calculating shipping costs based on the selected delivery address from "mfe-delivery." When a recommended product is added to the cart, "mfe-recommender" notifies "mfe-checkout," which subsequently informs "mfe-delivery" to recalculate the shipping costs. If the shipping address is updated, "mfe-delivery" notifies all other MFEs to verify whether the products they display can be shipped to the new address; if not, those products are disabled. Adopting the Pub-Sub pattern, all components (screen or fragments) would dispatch events to the Event Store, with would notify only the components subscribed in the event (Figure 50).

Solution Pitfalls: The shared libraries must be carefully handled to avoid Dependency hell.

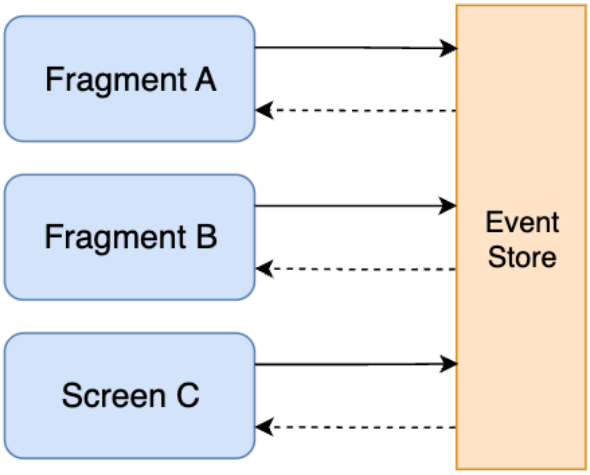

Figure 50 – Communication between fragments using an Event Store.

Related Anti-Patterns: Dependency hell

References: Silva, Rodrigues & Conte (2025)

7.3.7 Dependency hell

Also Known As: The Dependencies Hell; Poor Dependency Management in Micro Frontends.

Category: Inter-frontend

Problem: Micro frontends share external libraries without consistent dependency management.

Symptoms and Consequences: Version mismatches, compatibility issues, and broken functionality. These problems are amplified when teams extend shared libraries to meet specific needs of individual MFEs, resulting in forks or diverging implementations that no longer align with the original library.

Solution: Adopt clear and consistent strategies for managing shared dependencies in distributed frontend architectures. Use techniques such as import maps, module federation, or even share-nothing approaches when appropriate.

Resulting Context: All MFEs use the same versions of shared libraries to avoid extending core libraries for specific use cases—instead, favoring composition over extension. External dependencies are wrapped in internal abstractions to reduce tight

coupling and minimize the risk of breakage. Modular and well-separated libraries are created, and version control, integration, and testing are automated to detect divergence early and maintain alignment across teams.

Example: Consider all micro frontends initially relying on a core library at version 1.1.0. Later, the library released version 1.2.0, which included new features. One micro frontend (MFE-A) wants to adopt these features and upgrades to the new version. At the same time, another micro frontend (MFE-B) extends the core library by adding custom code on top of it. When the core library eventually releases a major update (version 2.1.0), the extension built by MFE-B breaks, creating conflicts and forcing the team to maintain a separate, extended version of the library. To avoid this type of dependency hell, teams should decouple feature extensions from core libraries by using internal wrappers that isolate custom functionality from the base library (Figure 51). Additionally, it is crucial to ensure that all micro frontends utilize consistent versions of shared libraries, employing centralized dependency management strategies to maintain alignment across the system.

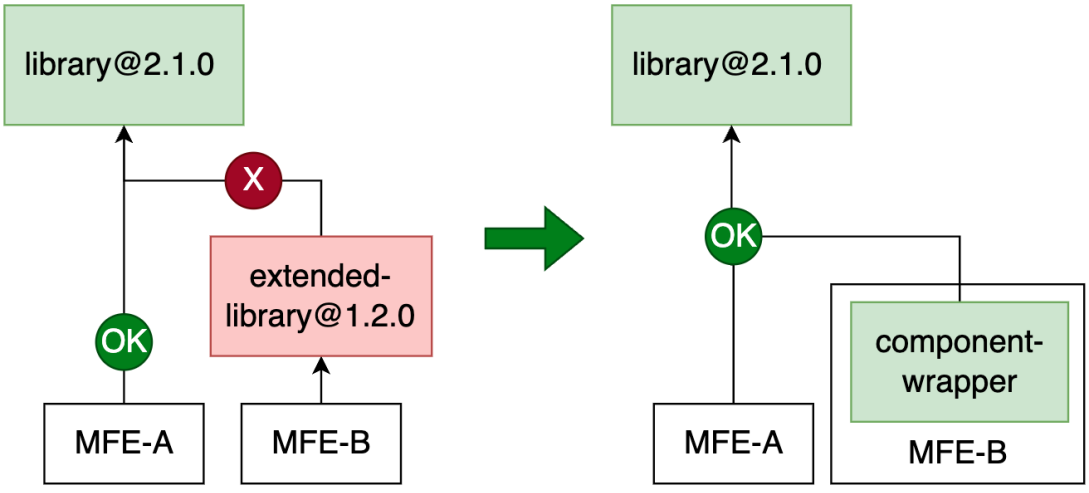

Figure 51 – Wrapper component as a solution to extended libraries.

Solution Pitfalls: Be careful using wrappers to change components locally because it can lead to UI inconsistency. Always try to keep the same dependencies between all MFEs.

Related Anti-Patterns: No CI/CD

References: Mezzalira (2020), Shinde (2022), Figus Alexander Guensche & Mez-

zalira (2024), Raimundo (2023)

7.3.8 Dependent deployment

Also Known As: Hidden Monolith, The deployment queue of hell, Dependent Deploy

Category: Inter-frontend

Problem: Micro frontends become so tightly coupled that their deployment requires coordination across multiple teams.

Symptoms and Consequences: One MFE cannot be deployed or rolled back without affecting another. Features must be released simultaneously across MFEs, forcing teams into synchronized release cycles. This leads to deployment queues, where MFEs must be deployed in a specific order to avoid breaking the application, increasing operational complexity and risk.

Solution: Promote independent deployment for each micro frontend. Implement feature flags using a tool like Flagsmith [2] that all relevant MFEs can check at runtime when a coordinated release is necessary, allowing features to be toggled on or off independently of the deployment timeline.

Resulting Context: Every MFE has its own CI/CD pipeline and can be deployed or rolled back without relying on other MFEs. MFE boundaries are defined according to Domain-Driven Design principles, reducing tight coupling and enabling true autonomy at the deployment level.

Example: In a SaaS application, a new feature—team-based project tagging—is developed across three MFEs: MFE-projects, MFE-teams, and MFE-dashboard. The MFE-projects team finishes its part early and needs to deploy it soon because other unrelated fixes and improvements are queued for release. However, the MFE-teams and MFE-dashboard teams are still finalizing their portions of the feature and cannot yet release. Without a proper mechanism, the MFE-projects team is blocked, forced to wait for the others, creating a deployment queue and increasing risk. By applying the solution, the MFE-projects team deploys its changes immediately but guards the

[2] https://www.flagsmith.com/

new feature behind a feature flag, keeping it off in production until all dependent MFEs are ready. Once all teams have deployed their parts, the feature flag is toggled on, activating the new functionality across the system without requiring perfectly synchronized deployments.

Solution Pitfalls: Be careful when defining the boundaries to avoid Nano and Mega Frontends.

Related Anti-Patterns: Nano Frontend, Mega Frontends, Spaghetti Architecture, No Versioning

References: Rappl (2024), Casas (2020), Figus Alexander Guensche & Mezzalira (2024)

7.3.9 Dismissing human factors

Also Known As: -

Category: Development

Problem: Teams pursue technical goals and deadlines without considering the well-being, morale, or work-life balance of their members.

Symptoms and Consequences: Micro frontends are misused as a technical solution, while their true potential to improve team autonomy and structure is neglected. Developers suffer from overworking, micro management and unrealistic expectations.

Solution: Strengthen teams and avoid central management as much as possible.

Resulting Context: Superiors empowering teams to work independently and without controlling every action or demanding unrealistic deadlines.

Example: In a company migrating to a micro frontend architecture, leadership pressures all teams to quickly break apart a monolithic frontend, assigning technical tasks like splitting components and wiring up module federation without giving teams the time or autonomy to reorganize around domain ownership. Developers are micro-managed, expected to coordinate constant cross-team changes, and overloaded with deadlines, turning the migration into a stressful technical exercise rather than an opportunity to improve team autonomy. The company should be refocusing on empowering

teams, allowing them to control their micro frontends, align work with their domain expertise, and set realistic delivery goals.

Solution Pitfalls: Make sure that micro frontend is the right architectural style for your scenario and divide teams according to domains to avoid Micro Frontend as the goal and Common Ownership anti-patterns, respectively.

Related Anti-Patterns: Common Ownership, Micro Frontend as the goal

References: Rappl (2024)

7.3.10 Distributed Data Inconsistency

Also Known As: -

Category: Inter-frontend

Problem: Inconsistency in the data shared across micro frontends due to lack of change propagation.

Symptoms and Consequences: The original data changes and those changes are not propagated. If a micro frontend holds a replicated version of the data, it's unclear whether it is allowed to update it. This causes divergence between the original and the duplicate.

Solution: Keep data where it belongs. Do not allow other micro frontends to access the data directly; instead, expose it indirectly through attributes or properties.

Resulting Context: MFEs do not duplicate data and always access the source MFE to retrieve it, which exposes the data through attributes or properties.

Example: In a financial platform, the MFE-account is responsible for managing the account balance, while the MFE-transactions handles user-initiated fund transfers. Instead of querying the up-to-date balance from MFE-account, the transactions MFE relies on a locally stored copy of the balance retrieved during a previous session. When a user attempts to transfer funds, MFE-transactions validates the operation based on its outdated balance, believing sufficient funds are available. However, since the actual balance in MFE-account has already decreased due to other transactions, the transfer request ultimately fails when it reaches the backend. By applying the solution, the

transactions MFE no longer duplicates the balance but always queries the authoritative source (MFE-account or a backend API) at the moment of the operation.

Solution Pitfalls: Avoid implementing centralized states to avoid the Global state communication Anti-pattern.

Related Anti-Patterns: Global state communication

References: Rappl (2024)

7.3.11 Framework Frenzy

Also Known As: Framework Madness, Hydra of Lerna, Multiple frameworks approach, Multiple technologies in the same application, Multiple-frameworks approach.

Category: Development

Problem: Introducing multiple frameworks without a real need, disregarding the complexity of communication between components built with different technologies.

Symptoms and Consequences: Several MFE are implemented with different frameworks and developers must be expert in several frameworks, loading may take long time to load all dependencies from each framework.

Solution: Whenever possible, use a single framework across all micro frontends. If there is an opportunity to standardize, prefer consistency over unnecessary flexibility.

Resulting Context: All MFEs are implemented using the same framework and technologies, facilitating maintenance and onboarding of new developers.

Example: In a content management platform, the MFE-editor is built using React, the MFE-dashboard uses Angular, and the MFE-analytics is implemented in Vue.js. Although the teams initially chose the best technology for their specific needs, over time, the project became increasingly difficult to maintain: developers need expertise in three frameworks to work across MFEs, shared libraries must be rewritten for each framework, and the application's startup time grows due to loading the dependencies of all three frameworks. By applying the solution, the organization gradually standardizes all MFEs onto React, simplifying development, improving team flexibility, reducing duplicated effort, and streamlining application performance by minimizing unnecessary

framework overhead.

Solution Pitfalls: Be careful to avoid the Golden Hammer Anti-pattern when using the same framework every time.

Related Anti-Patterns: Golden Hammer

References: Rappl (2024), Mezzalira (2020), Shinde (2022), Mezzalira (2021c), Raimundo (2023)

7.3.12 Global state communication

Also Known As: Relax, It's just code; Sharing state across Micro Frontends; Global State; Shared Global State.

Category: Inter-frontend

Problem: Using shared states violates the principle of segregation, compromising the independence of each micro frontend and increasing coupling.

Symptoms and Consequences: Changes in the shared state need coordination with every MFEs, otherwise you may introduce bugs in the ones reading the state, which decreases independence.

Solution: Each micro frontend should have its own event store. To enable communication, use an event emitter-based approach instead of sharing state directly.

Resulting Context: Every MFE may have a local state (if needed), and they communicate by subscribing in the events dispatched by other MFEs, not directly accessing their state.

Example: In a financial platform, the MFE-account maintains the current account balance in a shared global state, allowing other MFEs to directly access it when needed. When the team decides to internationalize the app and extend currency support, MFE-account updates the global state structure to include both the numeric balance and its associated currency. Since other MFEs directly depend on the shared state without proper contracts or decoupling, these changes break their logic. By applying the solution, each MFE stops directly depending on shared state and instead subscribes to events or queries data through well-defined, versioned interfaces, ensuring that updates in one

MFE do not unexpectedly break others.

Solution Pitfalls: When creating local states, be careful to avoid the Distributed Data Inconsistency Anti-pattern.

Related Anti-Patterns: Distributed Data Inconsistency

References: Mezzalira (2020), Shinde (2022), Mezzalira (2021b), Mezzalira (2021c), Raimundo (2023)

7.3.13 Golden Hammer

Also Known As: -

Category: Development

Problem: All MFEs utilize the same technology, even if it does not meet the specific needs of each MFE.

Symptoms and Consequences: It happens due to developers' familiarity with only one specific technology. This approach limits the architecture, failing to take advantage of the benefits of the possibility of a heterogeneous architecture, which is one of the main attractions of adopting MFEs.

Solution: To choose the most suitable technology that addresses the specific challenges of each MFE, which includes adopting the correct programming languages, frameworks, and libraries during its development. When uncertain about a particular technology, conducting a proof-of-concept (POC) can validate its suitability.

Resulting Context: Testing new technologies through POCs to validate their suitability without compromising the establishment of standardized patterns within the company.

Example: A web application contains MFEs implemented using ReactJS framework with Client-side Rendering, even those encompassing essential pages such as the landing page. This technological uniformity overlooks the necessity for Search Engine Optimization (SEO) strategies to ensure better rankings on search engines like Google. It would be advisable to utilize ReactJS with Server-side Rendering or employ a static rendering framework such as NextJS, enabling better optimization for search engines.

Solution Pitfalls: Increasing the variety of technologies can increase the complexity of the architecture and lead to the Framework Frenzy Anti-pattern.

Related Anti-Patterns: Framework Frenzy

References: Silva, Rodrigues & Conte (2025)

7.3.14 Hammering APIs

Also Known As: Let's hammer the APIs

Category: Inter-frontend

Problem: Multiple micro frontends independently call the same API endpoint.

Symptoms and Consequences: Duplicated requests, unnecessary load on backend services, and scalability issues.

Solution: Evaluate whether the concerned micro frontends truly belong to separate domains. If not, consider merging them into a single MFE that handles the request once. If separation is justified, redesign the architecture to centralize the API call in a shared container or parent component, which makes the request once and distributes the data to the dependent micro frontends.

Resulting Context: Every MFE make calls to routes related to their domains.

Example: In an e-commerce platform, both the MFE-list and the MFE-recommendations independently call the same product details API to fetch pricing and availability information for the same set of items. As a result, every time a user visits the product page, the backend receives duplicated requests from both MFEs, unnecessarily increasing load and degrading performance. By applying the solution, the architecture is redesigned so that a shared parent container makes a single API call to retrieve the product data and passes the relevant pieces down as props or events to the child MFEs, reducing redundant traffic and improving scalability without sacrificing the separation of frontend concerns.

Solution Pitfalls: When merging MFEs, be careful to not fall into the Mega Frontends Anti-pattern. When redesigning the architecture, consider using GraphQL, as proposed in the solution of the Access to different domains Anti-pattern.

Related Anti-Patterns: Access to different domains, Mega Frontend, Nano Frontend

References: Mezzalira (2020)

7.3.15 Hub-like Dependency

Also Known As: Tight coupling

Category: Inter-frontend

Problem: A screen of a MFE integrates fragments from several other MFEs, becoming a central point of interdependence (Figure 52).

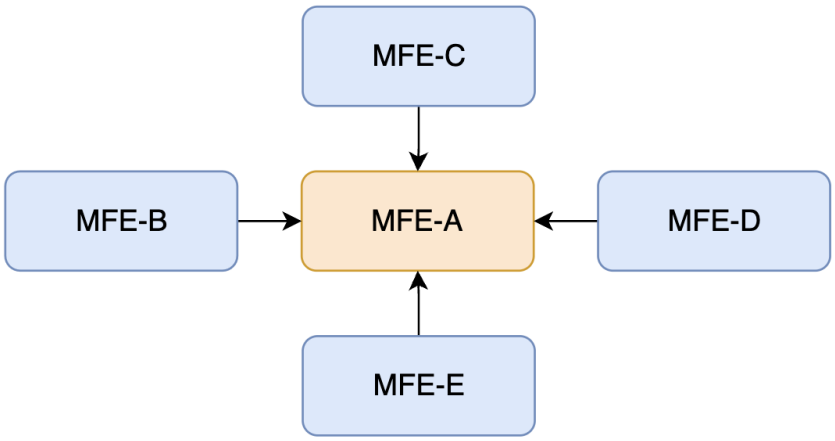

Figure 52 – MFE-A is a central point (Hub) of dependency between the other MFEs.

Symptoms and Consequences: Any issue occurring in the main screen or one of its fragments can affect all other fragments present on it.

Solution: The screen should be kept as simple as possible, and each fragment should implement robust error handling mechanisms.

Resulting Context: Implementing a strategy where uncaught errors within a fragment gracefully degrade its functionality, displaying a user-friendly fallback message.

Example: Consider a digital banking system where the main screen is an MFE that integrates several fragments from other MFEs, such as an investment list, a chart showing bitcoin value variations, account balance, and credit card statement amount. This structure introduces a significant vulnerability: a single faulty fragment can poten-

tially disrupt the entire main screen (Figure 53). To avoid it, all fragments must display a fallback message if unavailable and not raise the error to the screen (Figure 54).

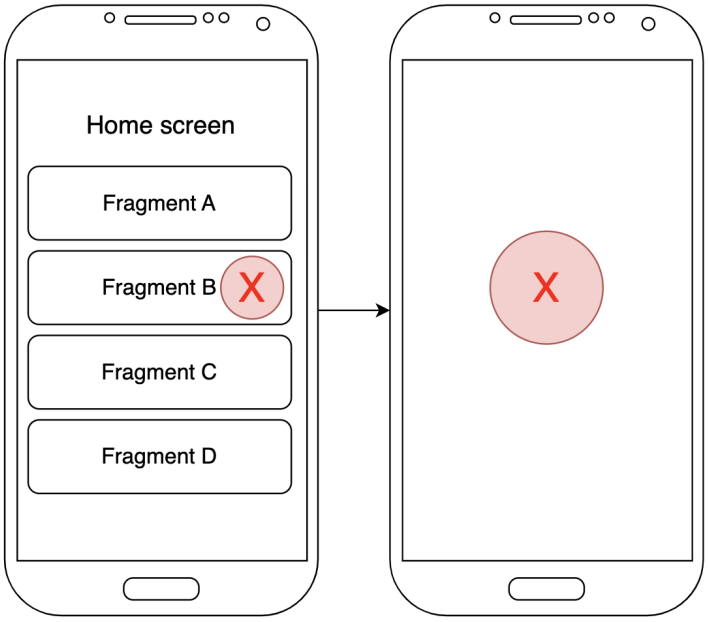

Figure 53 – Home screen is a screen with several fragments, and when Fragment B raises an error, the entire screen becomes unavailable.

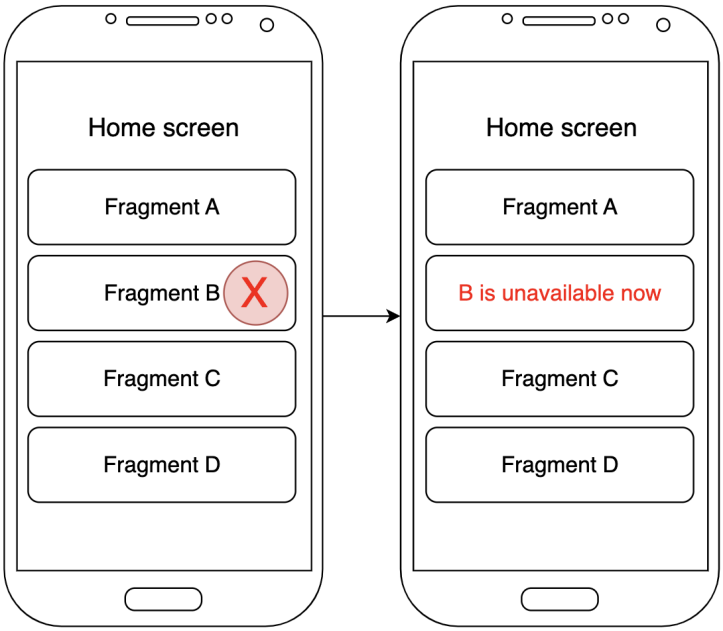

Figure 54 – When Fragment B raises an error, an user-friendly fallback message is rendered so the entire screen remains available.

Solution Pitfalls: Be sure that the components are well defined and do not fall into the Nano Frontend Anti-pattern.

Related Anti-Patterns: Nano Frontend

References: Mezzalira (2021b), Silva, Rodrigues & Conte (2025)

7.3.16 Knot Micro Frontend

Also Known As: -

Category: Inter-frontend

Problem: Three or more micro frontends communicate directly using context-specific interfaces.

Symptoms and Consequences: The tight coupling between MFEs leads to brittle integrations, duplicated contract logic, and increased complexity when extending or replacing individual MFEs. The lack of a shared communication protocol prevents scalability and maintainability.

Solution: Implement a domain-driven communication interface that is both generic and flexible. Define essential shared fields required for interoperability while allowing each MFE to include context-specific data through a structured, extensible payload (e.g., a generic field containing a list of typed objects).

Resulting Context: Implementing generic interfaces reduces coupling and enables each MFE to evolve independently while still supporting collaboration across the architecture.

Example: Suppose an e-commerce system has MFEs for Digital Products (mfe-digital-products) and Payments (mfe-payments). The payment screen of mfe-payments receives the digital product data as a parameter. At a later stage, a Physical Products MFE (mfe-physical-products) is implemented, so developers add physical product-specific attributes to the payment screen to allow digital or physical product payment (Figure 55). Later, adding new product types requires constantly adding attributes specific to each product type to the payment screen, so it becomes a highly coupled knot.

Solution Pitfalls: This anti-pattern is a case of the Spaghetti Architecture.

Related Anti-Patterns: Spaghetti Architecture, Dependent deployment

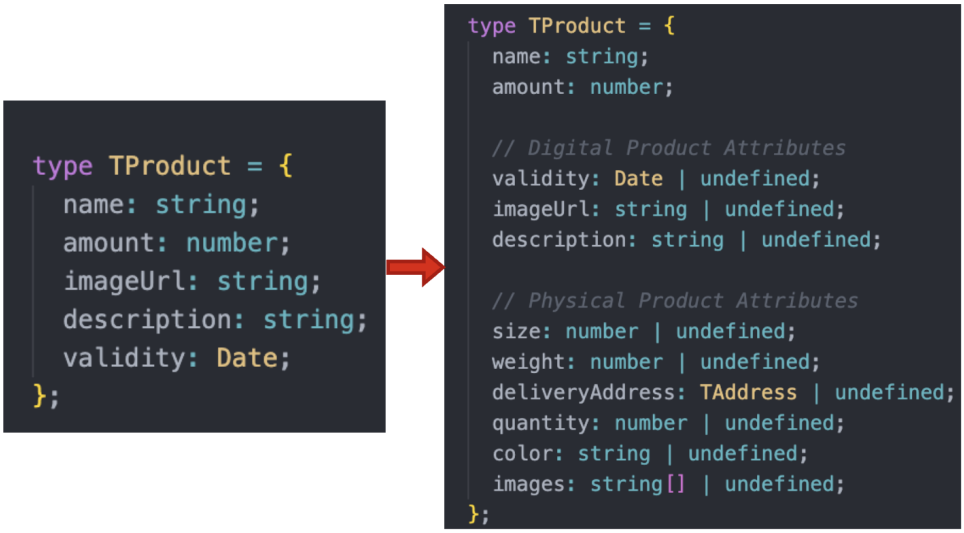

Figure 55 – To allow physical products payment, optional physical product-specific attributes are added and digital product-specific attributes become optional.

References: Silva, Rodrigues & Conte (2025)

7.3.17 Lack of Skeleton

Also Known As: -

Category: Operations

Problem: No skeleton or predefined boilerplate is available as a base for creating new micro frontends.

Symptoms and Consequences: This leads to inconsistencies across projects, repeated setup work, and increased onboarding time for new team members.

Solution: Whenever a new technology is adopted for implementing a micro frontend, the development team should create a boilerplate repository containing the necessary base code. In addition, maintain comprehensive documentation that outlines every step required to create a new MFE, regardless of the technology stack.

Resulting Context: The skeleton promotes consistency, reduces setup time, and accelerates adoption across teams. Developers can create new MFE based on a documentation with instructions for cloning the skeleton repository, configuring CI/CD pipelines, setting up monitoring tools, applying required design patterns during development, and other relevant guidelines.

Example: At the beginning of a specific system development, the developers create an MFE from scratch without adhering to a specific pattern. The second MFE is developed by copying files and code blocks from the first, changing specific parts. The exact process happens when creating new MFEs. Then, a diverse set of MFEs emerges, which hampers the establishment of automated pipelines, fosters code duplication, and complicates developer interchange between teams.

Solution Pitfalls: -

Related Anti-Patterns: No CI/CD

References: Silva, Rodrigues & Conte (2025)

7.3.18 Mega Frontend

Also Known As: -

Category: Intra-frontend

Problem: A single micro frontend takes on multiple responsibilities or concerns that should be separated.

Symptoms and Consequences: This often happens when an MFE includes several screens or unrelated fragments, resulting in a bloated unit.

Solution: Each micro frontend should focus on a specific subdomain of the application. Reevaluate and decompose large MFEs into smaller, more specialized units based on domain concerns.

Resulting Context: Development teams collaborate closely with product teams to gain a clear understanding of domain boundaries and reflect them in the system's architecture.

Example: An e-commerce system is decomposed into just two MFEs, with mfe-users related to users and mfe-shopping related to products and purchases. The latter MFE includes screens that display product listings, product details, purchase confirmations, and purchase history. Decomposing the mfe-shopping into at least two MFEs is necessary: one containing the product listing and product details screens, belonging to the product domain; and another containing the confirmation and purchase history

screens, belonging to the purchase domain (Figure 56).

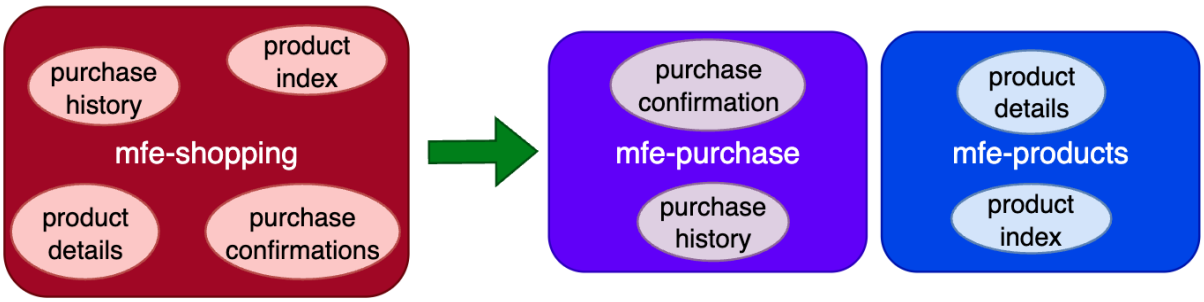

Figure 56 – Mega frontend break into two MFEs.

Solution Pitfalls: When redesigning the architecture, be careful to not create Nano Frontends. Additionally, analyze whether you should be implementing MFEs at all and not fall into the Micro Frontends as the Goal Anti-pattern.

Related Anti-Patterns: Nano Frontend, Micro Frontends as the Goal

References: Silva, Rodrigues & Conte (2025)

7.3.19 Micro Frontends Greedy

Also Known As: -

Category: Intra-frontend

Problem: When a developer is uncertain about creating a new MFE, the common practice is to opt for its creation.

Symptoms and Consequences: Whenever a need arises to develop a new set of screens or fragments, a new MFE is instantiated. This can lead to an explosion in the number of MFEs, making the system difficult to understand and increasing the likelihood of both nano and mega frontends emerging.

Solution: To determine where to implement a new feature composed of a set of screens and/or fragments, the domain of the new feature must first be defined. If it falls within the domain of an existing MFE, it should be implemented there. In this case, a documentation summarizing of all MFEs, with descriptions their contexts and links or screenshots of their screens and fragments can help identify the best fit for the

new feature. You can use a tool like Backstage [3] to document all MFE automatically. If it belongs to a brand new domain, one or more MFEs should be defined based on the domain definition. Establishing well-defined domains relies on the collaboration between the development and product teams to accurately define boundaries.

Resulting Context: Well-defined MFEs increase independence between teams and ease maintenance.

Example: Within a banking application, an MFE encompasses screens for security validation, utilizing confirmation code submission via email. Subsequently, the need arose to implement a new validation method, now employing facial recognition. The screens in this new flow differ from those in the previous flow, resulting in its implementation through a new MFE. Creating a new MFE might not be advisable, as two MFEs have the same context and functionalities.

Solution Pitfalls: Be careful to not create Mega Frontends when adding new features to existing MFEs or Nano Frontends when creating a new MFE with a domain too small.

Related Anti-Patterns: Nano Frontend, Mega Frontend

References: Silva, Rodrigues & Conte (2025)

7.3.20 Micro Frontend as the goal

Also Known As: -

Category: Development

Problem: Adopting the MFE architecture in inappropriate contexts.

Symptoms and Consequences: Can lead to more issues than benefits, especially in systems with few screens and low complexity or in companies lacking a sufficient number of developers to create dedicated teams for different application domains. In such situations, the maintenance costs of the architecture may outweigh the expected benefits, making its implementation unfeasible.

Solution: Software teams must consider carefully different aspects of adopting

[3] https://backstage.io/

MFE architecture. Considering the system's complexity, the feasibility of maintaining automated CI/CD pipelines and the team's restructuring according to different domains is necessary.

Resulting Context: Adoption of MFE only when feasible.

Example: A personal notes application is divided into the notes and user domains, each comprising its own MFE. The notes domain contains functionalities for note management, containing operations such as listing, creating, editing, and deleting notes. The user's domain encompasses login, registration, and profile management functionalities. In this context, using MFEs results in unnecessary maintenance and development challenges due to the low volume of screens and the low probability of increasing complexity in the application. Adopting a monolithic frontend is a suitable option.

Solution Pitfalls: -

Related Anti-Patterns: No CI/CD

References: Silva, Rodrigues & Conte (2025)

7.3.21 Nano Frontend

Also Known As: Yin and Yang (Micro Frontends and Components), Micro Frontends Vs Components, Micro Frontends versus Components

Category: Intra-frontend

Problem: A micro frontend is created with only a few screens or fragments.

Symptoms and Consequences: This typically occurs when teams confuse micro frontends with UI components, resulting in fragmented and ineffective architectural decomposition.

Solution: Define micro frontend boundaries based on business subdomains, following Domain-Driven Design principles. Avoid creating MFEs for isolated UI fragments or for purely technical reasons.

Resulting Context: The defined MFEs encapsulate cohesive sets of features aligned with domain-level concerns. Each MFE represent a meaningful subdomain

within the organization. The development team must work closely with the product team to gain a deep understanding of the domains and reflect them accurately in the architecture. For minor variations within a domain, MFEs use templates or component libraries. This approach avoids creating a separate MFE for each slight variation, promoting efficiency and code reuse.

Example: In a diagram editing platform, the UI is split into excessively granular micro frontends: one for the header menu and document actions (mfe-header), one for managing the shape objects (mfe-objects), another for the canvas where diagrams are drawn (mfe-drawer), one for styling and configuration options (mfe-style), and yet another for the footer navigation tabs (mfe-footer), as shown in Figure 57. Although these elements are all part of the same diagramming domain and operate in tight coordination, they are unnecessarily separated into fine-grained MFEs. This decomposition increases communication overhead, requires tight synchronization between teams, and makes UI maintenance more complex than necessary. By applying the solution, the architecture is redesigned to consolidate these micro frontends into a single domain-aligned MFE responsible for the diagramming experience while reusing internal components where variation is needed.

Solution Pitfalls: When redesigning nano frontends, be careful to not create Mega Frontends by merging unrelated MFEs.

Related Anti-Patterns: Micro Frontends Greedy, Mega Frontend

References: Mezzalira (2020), Shinde (2022), Raimundo (2023), Silva, Rodrigues & Conte (2025)

7.3.22 No CI/CD

Also Known As: -

Category: Operations

Problem: The company lacks an automated Continuous Integration (CI) and Continuous Delivery (CD) pipeline, so developers must manually execute tests and perform deployments.

Figure 57 – Page from a diagram editing platform with components as MFE, Nano Frontends.

Symptoms and Consequences: This manual process becomes burdensome, especially with the potential existence of multiple MFEs. It increases development time, reduces productivity, and raises the risk of errors in the production environment.

Solution: Implement an automated and replicable CI/CD process that extends for new MFEs, ensuring they will have automated test execution and deployment consistently and efficiently. This should be part of the Definition of Done (DoD) of the architecture.

Resulting Context: Updated in MFEs always trigger CI pipelines to run tests and the deployment of them is made automatically by the CD pipeline, which eases the maintenance and evolution of the MFEs.

Example: Upon releasing a new system version, a developer must conduct manual tests and ensure all unit tests pass. However, developers may skip the tests and manually deploy the changes without realizing some tests are failing, introducing bugs, which is avoidable with an automated CI pipeline (Figure 58). Even if the tests pass, there is still a risk of making mistakes during deployment, which could render the

system unavailable. Automating the deployment process with CD ensures correct and consistent execution.

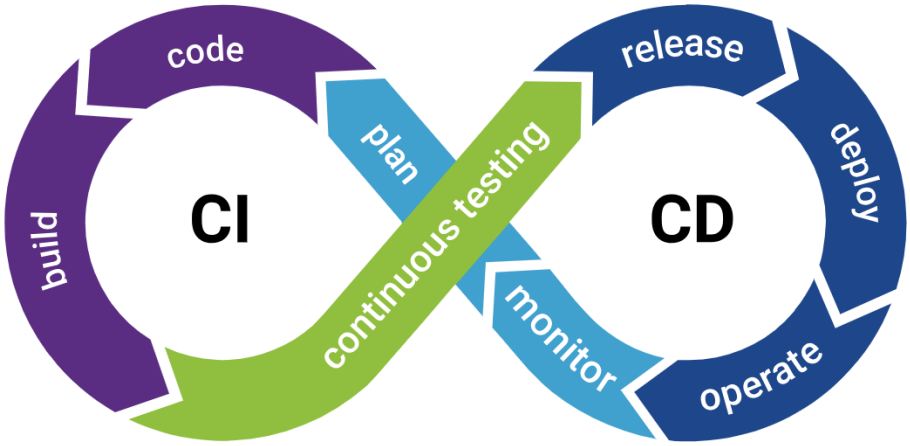

Figure 58 – CI and CD cycle. Source: <https://www.abtasty.com/resources/ci-cd/>

Solution Pitfalls: Make sure to document how to implement the CI/CD pipelines to new MFE according to the solution of the Lack of Skeleton Anti-pattern.

Related Anti-Patterns: Lack of Skeleton

References: Silva, Rodrigues & Conte (2025)

7.3.23 No Versioning

Also Known As: -

Category: Operations

Problem: Micro frontends are not versioned.

Symptoms and Consequences: Without proper version control, changes in one MFE can inadvertently affect others, leading to instability, incompatibility between components, and difficulties in rollback or maintenance.

Solution: Adopt a clear versioning strategy, such as Semantic Versioning, to track and communicate changes effectively.

Resulting Context: Versioning ensures that updates are predictable and that existing versions continue to function without disruption, enabling safe evolution and better coordination across teams.

Example: Consider a payment confirmation page with a fragment for calculating shipping costs. Whenever the user inputs shipping information into the fragment, the system generates a delivery charge and adds it to the total purchase amount displayed on the screen. Suppose the delivery charge's return value format changes and the fragment is not versioned. The delivery charge will not be added to the total purchase amount, potentially resulting in a display error or even mistakenly free deliveries. However, if the fragment is versioned, the screen will not be affected by the format change, as it will continue to use the previous version of the fragment and can be updated later when necessary.

Solution Pitfalls: -

Related Anti-Patterns: Spaghetti Architecture

References: Silva, Rodrigues & Conte (2025)

7.3.24 One Micro Frontend for all

Also Known As: -

Category: Inter-frontend

Problem: A single micro frontend is created and imported by all other MFEs.

Symptoms and Consequences: Misuse of local storage, pub/sub mechanisms, and web socket mechanisms to implement communication between the MFE for all and others. Having a single point of failure that can breaks the entire application.

Solution: Use the blackbox pattern to encapsulate shared functionality. In this model, the component exposes a clear input, renders itself into the DOM with its own internal workflow, and provides an output that other micro frontends can consume.

Resulting Context: This preserves the independence of MFEs while still enabling inter operation through well-defined boundaries.

Example: In a banking application, several MFEs—such as MFE-accounts, MFE-cards, and MFE-loans—need to display a UI for executing money transactions. Instead of having a tightly coupled shared micro frontend imported directly by all others, the system applies the blackbox pattern using a `TransactionsProvider` component

(Figure 59). When a transaction is needed, an MFE triggers an `openTransaction` event with the needed arguments. The `TransactionsProvider` renders itself into the DOM, manages the transaction flow independently, and, upon completion, dispatches a `transactionFinished` event with the result (success or failed).

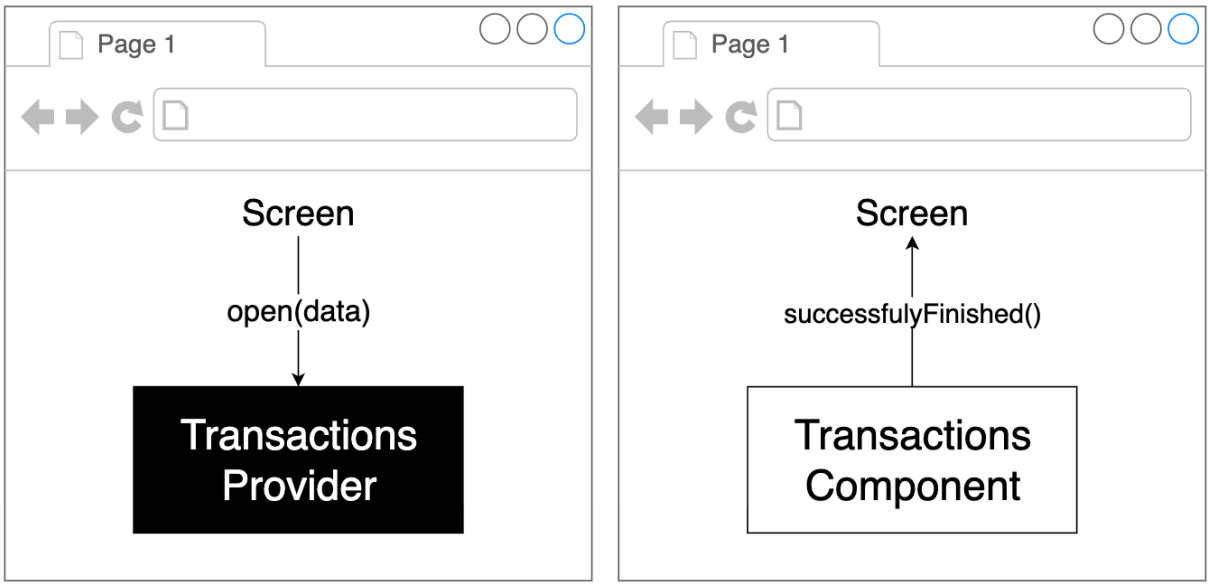

Figure 59 – Black box pattern to avoid one MFE for all.

Solution Pitfalls: -

Related Anti-Patterns: Global state communication

References: Gkamperlo (2020)

7.3.25 Partial UI Migration

Also Known As: Bye Bye Big Bang - Iterative Deployment

Category: Inter-frontend

Problem: Extracting a portion of the UI to create a micro frontend and embedding it back into the legacy monolith.

Symptoms and Consequences: It often slows down the system and provides no real benefit from adopting micro frontends.

Solution: Follow a path-based migration strategy, which allows gradual migration of frontend segments by switching versions through URL base paths.

Resulting Context: This enables progressive adoption of micro frontends without entangling them with the monolith.

Example: In an e-commerce platform, the development team begins migrating the checkout process to a new micro frontend but embeds it inside the legacy monolithic frontend using an iframe. Although technically the checkout is now isolated, every time the monolith renders the checkout page, it loads both the heavy legacy app and the micro frontend, causing performance degradation and offering little of the true independence or scalability benefits promised by micro frontends. By applying the solution, the team switches to a path-based migration strategy, routing `/checkout` directly to the new micro frontend via a separate deployment, while the monolith continues serving other routes like `/home` and `/products`, enabling gradual migration without tightly coupling the systems.

Solution Pitfalls: -

Related Anti-Patterns: Unmediated Legacy Integration

References: Mezzalira (2024)

7.3.26 Spaghetti Architecture

Also Known As: Tight coupling

Category: Inter-frontend

Problem: Micro frontends are structured in a disorganized way.

Symptoms and Consequences: leading to a tangled web of dependencies and interactions. This high degree of coupling between MFEs makes the system difficult to scale, test, and maintain.

Solution: Maintain loose coupling between micro frontends. Avoid direct references that rely on internal implementation details of other MFEs, such as URLs, module paths, or internal names.

Resulting Context: MFEs designed to interact through well-defined contracts, events, or shared interfaces.

Example: In a travel booking platform, the MFE-flights, MFE-hotels, and MFE-

packages reference each other directly using hardcoded module imports, internal URLs, and shared utility functions not exposed through official interfaces. For instance, MFE-hotels directly imports a pricing formatter from MFE-flights, and MFE-packages calls internal APIs exposed only for local use by MFE-hotels. Over time, this creates a tangled web of dependencies where any change in one MFE risks breaking others, making the system fragile, hard to test, and difficult to scale. By applying the solution, the teams refactor the system to remove internal references, replacing them with well-defined contracts, shared interfaces, or event-based communication, restoring clear boundaries and reducing coupling between MFEs.

Solution Pitfalls: -

Related Anti-Patterns: Knot Micro Frontend

References: Rappl (2024)

7.3.27 Unmediated Legacy Integration

Also Known As: The Swiss Army Knife, Integration Bottleneck Anti-pattern

Category: Inter-frontend

Problem: Integrating a legacy system into a micro frontend architecture without proper isolation often leads to architectural misalignment and increased complexity.

Symptoms and Consequences: Legacy systems may use incompatible technologies or communication mechanisms that do not fit the micro frontend model. A common mistake is to modify or extend the main integration layer of the application to accommodate these differences, introducing tight coupling, pollution of domain boundaries, and long-term maintainability issues.

Solution: Introduce an Anti-Corruption Layer (ACL) between the micro frontend application and the legacy system. This layer acts as a translator, isolating the legacy system and ensuring that its communication model and domain concepts do not leak into the modern architecture.

Resulting Context: By encapsulating the integration logic in a dedicated boundary, teams can preserve the integrity of the micro frontend system without modifying

its core integration layer.

Example: In a retail platform migrating to micro frontends, the team needs to integrate a legacy UI module responsible for managing product promotions, built years ago using jQuery and global JavaScript variables. Instead of isolating this legacy module, the developers directly import its scripts into the new React-based micro frontend container and wire up shared states and callbacks, causing tight coupling, polluting the modern architecture with outdated patterns, and increasing the risk of breaking the entire system with any legacy update. By applying the solution, the team wraps the jQuery-based promotion module inside an Anti-Corruption Layer (ACL) component (Figure 60), which handles mounting, unmounting, and communication through well-defined events or props, ensuring the legacy logic stays isolated and the modern React micro frontends remain clean, maintainable, and decoupled.

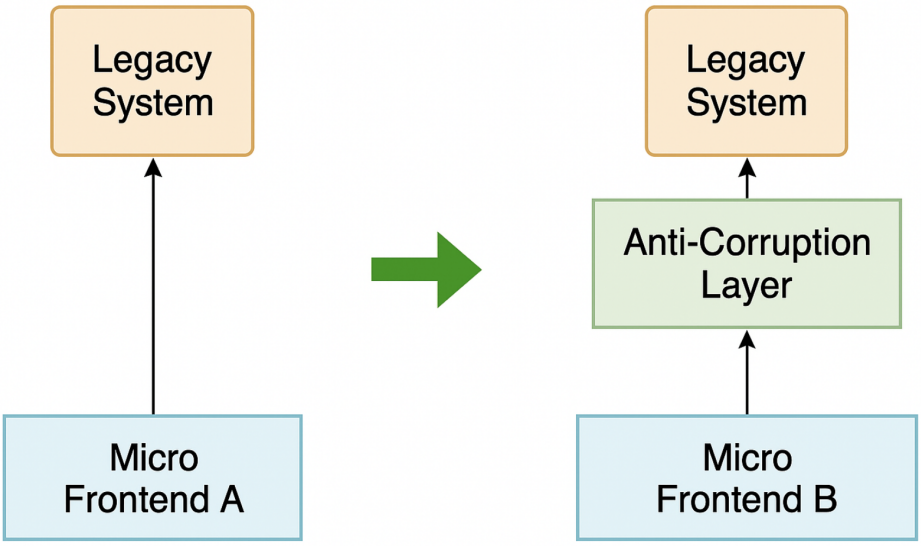

Figure 60 – Anti-corruption layer to connect to legacy systems.

Solution Pitfalls: -

Related Anti-Patterns: Partial UI Migration

References: Mezzalira (2020), Raimundo (2023)

8

Final Considerations

Concluding this Master's Thesis, this chapter provides a summary of our work (Section 8.1), highlights the contributions of this thesis (Section 8.2), and presents future work (Section 8.3).

8.1 Research Overview

This research presents a catalog of MFE anti-patterns derived from MS anti-patterns. To validate whether the identified problems are prevalent in MFE architectures and whether the proposed solutions effectively address them, we conducted a Personal Opinion Survey with 20 industry practitioners. Based on their feedback, we improved the anti-patterns and showcased them in a web application to foster community collaboration. Then, we conducted a controlled experiment to compare the effectiveness of an MFE anti-patterns catalog with publicly available examples and guidelines as teaching materials. In this study, we also evaluated how students used the catalog, their perceptions of its utility, and whether it enhanced their perceived learning about MFE architectures. Finally, we conducted a Multivocal Literature Review to expand the catalog by adding anti-patterns proposed by practitioners in Grey Literature sources.

The survey results show that all identified anti-patterns have been encountered by participants in real-world MFE projects, each receiving varying harmfulness scores. Participants emphasized the catalog's utility as a valuable resource for improving MFE architecture, highlighting its potential to guide both novice and experienced developers

in avoiding common pitfalls. Additionally, participants provided valuable insights by suggesting new anti-patterns. This highlights the importance of creating a collaborative platform where researchers and practitioners can jointly propose, discuss, and validate new anti-patterns. Our web application facilitates this process by enabling practitioners to share their knowledge and experiences, ensuring the catalog remains comprehensive and up-to-date.

Before the controlled experiment, we conducted sessions to teach students about MFE, which we presented as a teaching report. When analyzing the experiment results, statistical tests indicated that both supporting materials were equally effective in helping students learn about MFE and supporting decision-making. The practitioner-provided guidelines supported students in understanding how MFE is applied in real-world scenarios, offering a practical and comprehensive perspective beyond traditional textbook content. Access to the MFE anti-patterns catalog significantly increased students' perceived learning. Students also reported positive experiences using the catalog, emphasizing its usefulness in identifying problems and solutions, conducting efficient searches for architectural issues, and reviewing core MFE concepts. Overall, the feedback suggests that engaging with tools grounded in real-world architectural challenges can effectively prepare in-training software engineers to address practical issues in MFE development.

As a result of the MLR, we expanded our catalog from 12 to 27 anti-patterns, making it a comprehensive source of knowledge on common MFE issues and effective solutions based on practitioners' experience. After consolidating similar entries, the Inter-frontend category faced the most significant growth, indicating that the main challenges in MFE are related to how micro frontends are composed and how communication between them is implemented. Additionally, we identified anti-patterns related to those we had previously proposed, which allowed us to improve and refine their descriptions. This study concludes our research by contributing to both researchers and practitioners, offering a central source of knowledge that helps identify, understand, and address recurring problems in the design and implementation of MFE architectures.

8.2 Contributions

This research has made several contributions to the MFEs field, spanning theoretical and practical aspects. We have already submitted two papers, one of which has been accepted, and we plan to submit one more. Below, we categorize and describe the contributions of this thesis thus far:

- **Theoretical contributions**:
    1. A Catalog of Micro Frontends Anti-patterns, highlighting common problems and effective solutions, bridging knowledge from industry to academia.
    2. Empirical evidence on the use of the Micro Frontends anti-patterns catalog by students while learning Micro Frontends.
    3. Empirical evidence comparing the Micro Frontends anti-patterns catalog with Micro Frontends guidelines, focusing on students' ability to make informed decisions regarding the maintenance of a Micro Frontends architecture.
    4. A Multivocal Literature Review that consolidates previously scattered practitioner knowledge on Micro Frontends, strengthening and enriching the theoretical and practical understanding of the field.
- **Practical contributions**:
    1. Development of a web application to present the Micro Frontends anti-patterns catalog, allowing developers to access it during the implementation and maintenance of Micro Frontends architectures while also enabling collaboration on the catalog.
    2. Development of Micro Frontends teaching materials that integrate theoretical and practical content, allowing instructors to incorporate this topic into Software Architecture courses seamlessly.
- **Accepted papers**:
    1. *A Catalog of Micro Frontends Anti-patterns* (SILVA; RODRIGUES; CONTE, 2025): Catalog's proposal and evaluation by practitioners through a Personal

Opinion Survey. This paper was accepted for publication at the IEEE/ACM International Conference on Software Engineering (ICSE) 2025 – Research Track.

- **Submitted papers**:

  1. *Evaluating Strategies for Teaching Micro Frontends: Do Anti-patterns Help?*: Experience report on teaching Micro Frontends to undergraduate Computer Science students, along with a controlled experiment comparing the catalog to guidelines provided by developers on the internet. This paper was submitted for publication at the XXXIX Brazilian Symposium on Software Engineering (SBES) 2025 – Education Track.

- **Planned papers**:

  1. *A Comprehensive Catalog of Micro Frontends Anti-patterns: A Multivocal Literature Review*: Report of the Multivocal Literature Review we conducted to expand the catalog by adding anti-patterns proposed by practitioners in Grey Literature sources. We intend to submit this paper to the Journal of Software and Systems (JSS).

8.3 Future Work

For future work, we plan to conduct a participatory case study to analyze how practitioners use the catalog and to provide empirical evidence of its impact on the overall quality of MFE architectures. We also aim to examine the effects of the proposed solutions, both positive and negative, investigating whether their application may introduce new anti-patterns. Based on this analysis, we plan to develop evolution maps that organize solutions according to the presence of specific anti-patterns, offering structured paths for architectural improvement.

Regarding anti-pattern detection, we intend to formalize the definition of anti-patterns using a formal notation or specification language, enabling precise representation and systematic analysis. We plan to develop automated detection tools to help

identify anti-patterns efficiently. We will analyze public MFE repositories on GitHub using both manual inspection and automated techniques. Finally, we aim to explore how the catalog can serve as a foundational knowledge base for building AI-powered agents that assist practitioners in making informed architectural decisions in micro frontends.

References

ABGAZ, Y. et al. Decomposition of monolith applications into microservices architectures: A systematic review. *IEEE Transactions on Software Engineering*, IEEE, v. 49, n. 8, p. 4213–4242, 2023. 36

AMPATZOGLOU, A. et al. Identifying, categorizing and mitigating threats to validity in software engineering secondary studies. *Information and software technology*, Elsevier, v. 106, p. 201–230, 2019. 141

ANKS, D. Mastering micro frontends: Best practices, pitfalls to avoid, tools and scaling strategies. *DEV Community*, nov 2023. Accessed: 2024-11-04. 99

ANTUNES, F. et al. Investigating benefits and limitations of migrating to a micro-frontends architecture. *arXiv preprint arXiv:2407.15829*, 2024. 14, 31, 92, 99, 118

APLYCA. *Best practices for micro frontends*. 2024. <https://www.aplyca.com/en/blog/best-practices-for-micro-frontends>. Accessed: 2024-11-03. 99

BOGNER, J. et al. Towards a collaborative repository for the documentation of service-based antipatterns and bad smells. In: IEEE. *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*. [S.l.], 2019. p. 95–101. 44, 52, 89

BRADA, P.; PICHA, P. Software process anti-patterns catalogue. In: *Proceedings of the 24th European Conference on Pattern Languages of Programs*. [S.l.: s.n.], 2019. p. 1–10. 19, 31, 41, 42, 150

BRAUN, V.; CLARKE, V. One size fits all? what counts as quality practice in (reflexive) thematic analysis? *Qualitative research in psychology*, Taylor & Francis, v. 18, n. 3, p. 328–352, 2021. 65

BROWN, S. *The C4 Model for Visualising Software Architecture*. [S.l.]: Leanpub, 2023. 98

BROWN, W. H. et al. *AntiPatterns: refactoring software, architectures, and projects in crisis*. [S.l.]: John Wiley & Sons, Inc., 1998. 41, 149

CAPDEPON, Q. et al. Migration process from monolithic to micro frontend architecture in mobile applications. In: *Proceeding of the International Workshop on Smalltalk Technologies*. [S.l.: s.n.], 2023. 31, 42, 51, 92

CARTAXO, B.; PINTO, G.; SOARES, S. The role of rapid reviews in supporting decision-making in software engineering practice. In: *Proceedings of the 22nd International*

Conference on Evaluation and Assessment in Software Engineering 2018. [S.l.: s.n.], 2018. p. 24–34. 33, 46

CASAS, R. *Rules of Micro-Frontends*. 2020. <https://www.infoxicator.com/rules-of-micro-frontends>. Blog. 132, 133, 164

CERNY, T. et al. Catalog and detection techniques of microservice anti-patterns and bad smells: A tertiary study. *Journal of Systems and Software*, Elsevier, v. 206, p. 111829, 2023. 40, 43, 52

CHAPETON, G. G. Collaborative geovisual analytics. 2022. 51

CHRISTENSEN, H. B. Teaching microservice architecture using devops—an experience report. In: SPRINGER. *European Conference on Software Architecture*. [S.l.], 2022. p. 117–130. 44

COHEN, J. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, Sage Publications Sage CA: Thousand Oaks, CA, v. 20, n. 1, p. 37–46, 1960. 65, 77, 127

CORBIN, J.; STRAUSS, A. *Basics of qualitative research: Techniques and procedures for developing grounded theory*. [S.l.]: Sage publications, 2014. 94, 103, 110

CORDEIRO, R. et al. Teaching complex systems based on microservices. *GROUP*, v. 1, p. 1, 2019. 45

DMITRY, N.; MANFRED, S.-S. On micro-services architecture. *International Journal of Open Information Technologies*, . . . , v. 2, n. 9, p. 24–27, 2014. 30

DRAGONI, N. et al. Microservices: yesterday, today, and tomorrow. *Present and ulterior software engineering*, Springer, p. 195–216, 2017. 30

DUNN, O. J. Multiple comparisons using rank sums. *Technometrics*, Taylor & Francis, v. 6, n. 3, p. 241–252, 1964. 65

ERL, T. *Service-oriented architecture: analysis and design for services and microservices*. [S.l.]: Prentice Hall Press, 2016. 30, 37

EVANS, E. *Domain-driven design: tackling complexity in the heart of software*. [S.l.]: Addison-Wesley Professional, 2004. 83, 88, 118, 121

FIGUS ALEXANDER GUENSCHE, H. H. M.; MEZZALIRA, L. *Understanding and implementing microfrontends on AWS - AWS Prescriptive Guidance*. 2024. <https://docs.aws.amazon.com/pdfs/prescriptive-guidance/latest/micro-frontends-aws/micro-frontends-aws.pdf>. Technical Document. 133, 162, 163, 164

FORD, N. et al. *Building Evolutionary Architectures*. [S.l.]: " O'Reilly Media, Inc.", 2022. 51

FORD, N. et al. *Software Architecture: The Hard Parts*. [S.l.]: " O'Reilly Media, Inc.", 2021. 51

FRIEDMAN, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, Taylor & Francis, v. 32, n. 200, p. 675–701, 1937. 65

GALSTER, M.; ANGELOV, S. What makes teaching software architecture difficult? In: *Proceedings of the 38th International Conference on Software Engineering Companion*. [S.l.: s.n.], 2016. p. 356–359. 44, 101

GARFISH. *Garfish*. 2021. Disponível em: <https://www.garfishjs.org>. 38

GAROUSI, V.; FELDERER, M.; MÄNTYLÄ, M. V. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Information and software technology*, Elsevier, v. 106, p. 101–121, 2019. 34, 123, 127, 128, 142

GEERS, M. *Micro Frontends in Action*. [S.l.]: Simon and Schuster, 2020. 31, 37, 38, 39, 99

GITHUB. *GitHub Actions: Automate your workflow from idea to production*. 2025. Accessed: 2025-02-15. Disponível em: <https://github.com/features/actions>. 89

GKAMPERLO, N. *Micro-frontend "Blackbox Pattern"*. 2020. <https://medium.com/@ngkamperlo/micro-frontend-blackbox-pattern-295c40b681e4>. Blog. 132, 133, 182

JUMPPONEN, R. Modern software architecture. 2021. 51

KAUSHIK, N.; KUMAR, H.; RAJ, V. Micro frontend based performance improvement and prediction for microservices using machine learning. *Journal of Grid Computing*, Springer, v. 22, n. 2, p. 1–26, 2024. 31, 43, 92

KAZMAN, R. et al. A better way to teach software architecture. In: *Software Architecture: Research Roadmaps from the Community*. [S.l.]: Springer, 2023. p. 101–110. 30, 44, 101

KITCHENHAM, B. A.; BUDGEN, D.; BRERETON, P. *Evidence-based software engineering and systematic reviews*. [S.l.]: CRC press, 2015. v. 4. 31, 33, 62, 125

KLIMM, M. C. *Design Systems for Micro Frontends*. Tese (Doutorado) — University of Applied Sciences, 2021. 51

KLIMM, M. C. *Design Systems For Micro Frontends-An Investigation Into The Development Of Framework-Agnostic Design Systems Using Svelte And Tailwind Css*. Tese (Doutorado) — Hochschulbibliothek der Technischen Hochschule Köln, 2021. 51

KOENIG, A. Patterns and antipatterns. In: *The patterns handbooks: techniques, strategies, and applications*. [S.l.: s.n.], 1998. p. 383–389. 41

KOFLER, J. Como os microfrontends podem ajudar a focar nas necessidades de negócios. *InfoQ Brasil*, 2020. Accessed: 2024-11-03. 99

LAGO, P.; VLIET, H. V. Teaching a course on software architecture. In: IEEE. *18th Conference on Software Engineering Education & Training (CSEET'05)*. [S.l.], 2005. p. 35–42. 44

LAITENBERGER, O.; DREYER, H. M. Evaluating the usefulness and the ease of use of a web-based inspection data collection tool. In: IEEE. *Proceedings Fifth International Software Metrics Symposium. Metrics (Cat. No. 98TB100262)*. [S.l.], 1998. p. 122–132. 101

LANDIS, J. R.; KOCH, G. G. The measurement of observer agreement for categorical data. *biometrics*, JSTOR, p. 159–174, 1977. 72, 127

LANGE, M.; KOSCHEL, A.; HAUSOTTER, A. Microservices in higher education. In: *International Conference on Microservices*. [S.l.: s.n.], 2019. 44

LEWIS, J.; FOWLER, M. *Microservices a definition of this new architectural term*. 2014. Disponível em: <https://www.martinfowler.com/articles/microservices.html>. 30, 36

LIKERT, R. A technique for the measurement of attitudes. *Archives of psychology*, 1932. 101

MAFRA, S. N.; BARCELOS, R. F.; TRAVASSOS, G. H. Aplicando uma metodologia baseada em evidência na definição de novas tecnologias de software. In: SBC. *Anais do XX Simpósio Brasileiro de Engenharia de Software*. [S.l.], 2006. p. 239–254. 13, 32, 33

MÄNNISTÖ, J.; TUOVINEN, A.-P.; RAATIKAINEN, M. Experiences on a frameworkless micro-frontend architecture in a small organization. In: IEEE. *2023 IEEE 20th International Conference on Software Architecture Companion (ICSA-C)*. [S.l.], 2023. p. 61–67. 31, 42, 92

MANNISTO, T.; SAVOLAINEN, J.; MYLLARNIEMI, V. Teaching software architecture design. In: IEEE. *Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA 2008)*. [S.l.], 2008. p. 117–124. 44, 101

MARTINS, P. J. P. *Development of an e-portfolio social network using emerging web technologies*. Tese (Doutorado), 2022. 51

MEDVIDOVIC, N.; TAYLOR, R. N. Software architecture: foundations, theory, and practice. In: *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2*. [S.l.: s.n.], 2010. p. 471–472. 30

MEZZALIRA, L. *Micro Frontends Anti-Patterns*. 2020. InfoQ Conference Talk. Accessed: 2025-03-17. Disponível em: <https://www.infoq.com/presentations/microfrontend-antipattern/>. 125, 132, 133, 136, 157, 162, 163, 167, 168, 170, 178, 185

MEZZALIRA, L. *Building Micro-Frontends*. [S.l.]: O'Reilly Media, Inc., 2021. 30, 37, 43, 51, 138

MEZZALIRA, L. *Chapter 4. Discovering Micro-Frontend Architectures*. 2021. <https://www.oreilly.com/library/view/building-micro-frontends/9781492082989/ch04.html>. Book. 133, 168, 172

MEZZALIRA, L. *TechLead Journal: #47 - Micro-Frontends and the Socio-Technical Aspect*. 2021. <https://techleadjournal.dev/page/16/>. Audio. 133, 167, 168

MEZZALIRA, L. *Microfrontends Anti-Patterns: Seven Years in the Trenches*. 2023. Disponível em: <https://www.infoq.com/presentations/microfrontend-antipattern/>. 43

MEZZALIRA, L. *Micro-Frontends anti-patterns by Luca Mezzalira*. 2024. <https://www.youtube.com/watch?v=3jygY3LGTKc&ab_channel=Apiumhub>. Video. 133, 183

MORAES, F. et al. Micro frontend-based development: Concepts, motivations, implementation principles, and an experience report. In: *Proceedings of the 26th International Conference on Enterprise Information Systems*. [S.l.: s.n.], 2024. v. 2, p. 175–184. 31, 38, 39, 42, 43, 92, 99

NEWMAN, S. *Building microservices*. [S.l.]: O'Reilly Media, Inc., 2021. 37, 98

OBIORA, C. N. et al. Forecasting hourly solar radiation using artificial intelligence techniques. *IEEE Canadian Journal of Electrical and Computer Engineering*, IEEE, v. 44, n. 4, p. 497–508, 2021. 51

PARKER, G. et al. Visualizing anti-patterns in microservices at runtime: A systematic mapping study. *IEEE Access*, IEEE, v. 11, p. 4434–4442, 2023. 52

PAVLENKO, A. et al. Micro-frontends: application of microservices to web front-ends. *J. Internet Serv. Inf. Secur.*, v. 10, n. 2, p. 49–66, 2020. 43

PELTONEN, S.; MEZZALIRA, L.; TAIBI, D. Motivations, benefits, and issues for adopting micro-frontends: a multivocal literature review. *Information and Software Technology*, Elsevier, v. 136, p. 106571, 2021. 30, 31, 32, 37, 38, 39, 43, 99

PERLIN, R. et al. An approach to follow microservices principles in frontend. In: IEEE. *2023 IEEE 17th International Conference on Application of Information and Communication Technologies (AICT)*. [S.l.], 2023. p. 1–6. 31, 42, 92

PÖLÖSKEI, I.; BUB, U. Enterprise-level migration to micro frontends in a multi-vendor environment. *Acta Polytechnica Hungarica*, v. 18, n. 8, p. 7–25, 2021. 31, 92

QIANKUN. *qiankun: Probably the most complete micro-frontends solution you ever met*. 2019. Disponível em: <https://qiankun.umijs.org>. 38

RAIMUNDO, J. L. P. *Compositional Qualities of Microfrontends: The LdoD Archive*. 2023. <https://fenix.tecnico.ulisboa.pt/downloadFile/281870113706102/49372-joao-raimundo.pdf>. Master's Thesis. 133, 162, 163, 167, 168, 178, 185

RALPH, P. et al. Empirical standards for software engineering research. *arXiv preprint arXiv:2010.03525*, 2020. 66

RAPPL, F. *The Art of Micro Frontends: Build websites using compositional UIs that grow naturally as your application scales*. [S.l.]: Packt Publishing Ltd, 2021. 138

RAPPL, F. *Top 10 Micro Frontend Anti-Patterns*. 2024. DEV Community. Accessed: 2025-03-17. Disponível em: <https://dev.to/florianrappl/top-10-micro-frontend-anti-patterns-3809>. 15, 43, 125, 132, 133, 154, 158, 164, 165, 166, 167, 184

REACT. *React: A JavaScript library for building user interfaces*. 2025. Accessed: 2025-02-15. Disponível em: <https://react.dev>. 89

RICHARDSON, C. *Microservices patterns: with examples in Java*. [S.l.]: Simon and Schuster, 2018. 36, 37, 98

SHAPIRO, S. S.; WILK, M. B. An analysis of variance test for normality (complete samples). *Biometrika*, Oxford University Press, v. 52, n. 3-4, p. 591–611, 1965. 65

SHINDE, S. *4 Micro-Frontend Anti-Patterns*. 2022. <https://levelup.gitconnected.com/four-micro-frontend-anti-patterns-58aaa9fe19d5>. Blog. 43, 125, 132, 133, 162, 163, 167, 168, 178

SHUKLA, V. A comprehensive guide to micro frontend architecture. *Medium*, jul 2023. Accessed: 2024-11-04. 99

SILVA, M. R. d. Arquitetura reativa cognitiva baseada em microsserviços e micro-frontends para melhorar a experiência do usuário em aplicações bancárias por meio de interfaces adaptativas. 2024. 99

SILVA, N.; RODRIGUES, E.; CONTE, T. A Catalog of Micro Frontends Anti-patterns. In: *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. [S.l.: s.n.], 2025. p. 616–616. ISSN 1558-1225. 125, 132, 133, 138, 159, 161, 169, 172, 173, 174, 175, 176, 177, 178, 180, 181, 188

SILVA, N. R. *Micro frontends numa aplicação de pré-contabilidade*. Tese (Doutorado), 2023. 51

SINGLE-SPA. *single-spa: A javascript router for front-end microservices*. 2016. Disponível em: <https://single-spa.js.org>. 38, 99

TAIBI, D.; LENARDUZZI, V.; PAHL, C. Microservices anti-patterns: A taxonomy. *Microservices: Science and Engineering*, Springer, p. 111–128, 2020. 31, 43, 52, 64

TAIBI, D.; MEZZALIRA, L. Micro-frontends: Principles, implementations, and pitfalls. *ACM SIGSOFT Software Engineering Notes*, ACM New York, NY, USA, v. 47, n. 4, p. 25–29, 2022. 31, 37, 38, 39, 40, 43, 99

THOUGHTWORKS. Micro frontends. *ThoughtWorks Technology Radar*, 2016. Disponível em: <https://www.thoughtworks.com/pt-br/radar/techniques/micro-frontends>. 37

TIGHILT, R. et al. On the study of microservices antipatterns: A catalog proposal. In: *Proceedings of the European Conference on Pattern Languages of Programs 2020*. [S.l.: s.n.], 2020. p. 1–13. 43, 52

TOKUC, K. *Suitability of Micro-Frontends for an AI as a Service Platform*. Tese (Doutorado) — Hochschule für Angewandte Wissenschaften Hamburg, 2024. 51

TUTISANI, T. Design and architecture. In: *Effective Software Development for the Enterprise: Beyond Domain Driven Design, Software Architecture, and Extreme Programming*. [S.l.]: Springer, 2023. p. 105–175. 51

VALENTE, M. T. Engenharia de software moderna. *Princípios e Práticas para Desenvolvimento de Software com Produtividade*, v. 1, n. 24, 2020. 30, 98

VEGAS, S.; APA, C.; JURISTO, N. Crossover designs in software engineering experiments: Benefits and perils. *IEEE Transactions on Software Engineering*, IEEE, v. 42, n. 2, p. 120–135, 2015. 96

VELEPUCHA, V.; FLORES, P. A survey on microservices architecture: Principles, patterns and migration challenges. *IEEE Access*, IEEE, 2023. 51

VENKATESH, V.; BALA, H. Technology acceptance model 3 and a research agenda on interventions. *Decision sciences*, Wiley Online Library, v. 39, n. 2, p. 273–315, 2008. 94, 101

WESSELS, B. *Micro Front-End Architecture at Enterprise Scale*. 2020. <https://medium.com/swlh/micro-front-end-architecture-at-enterprise-scale-updated-july-2020-9159a4e0cc49>. Blog. 15, 132, 133, 155, 156

WOHLIN, C. et al. *Experimentation in software engineering*. [S.l.]: Springer Science & Business Media, 2012. 33, 76, 93, 94, 95, 96, 101, 106, 113

A

Survey Thematic Analysis

This Appendix provides the complete thematic analysis performed during the qualitative analysis on practitioners feedback presented in Chapter 4.

Thematic Analysis					
Anti Pattern	Question	Participant	Answers	Quotations	Final theme
Cyclic Dependency	If you disagree, please provide a description of what is not clear	P4	The Problem is correctly explained but not as much exemplified. The example only states a dependency between A and B, but not another that would close the cycle between B and A.	The example only states a dependency between A and B, but not another that would close the cycle between B and A.	Improvements to examples
	If you disagree, please provide a description of what is not clear	P11	Alternative solutions can be considered for this problem, such as creating a single interface to access the functionalities shared between the MFEs.	creating a single interface to access the functionalities shared between the MFEs	Proposal of new solutions
	In case you have a different suggestion on the problem solution, please provide a description	P5	I agree with the solution; however, microfrontends inherently have the characteristic of being more oriented toward an organizational context (not always, but often), unlike microservices, which are more focused on solving software issues such as cost and scalability. The solution of composing microfrontends based on domain analysis can be valid in an organizational context where the responsible parties are close (or preferably the same team). In very large companies, due to the distances between teams, duplication may be a worse solution in terms of software but better in organizational terms. In highly coupled systems, creating intermediary layers can be a solution to avoid merging the MFEs, which could eventually turn these MFEs into a monolith due to the system's high coupling characteristic. So, in general, the presented solution makes technical sense, but the business context might require a solution that is not "optimal."	composing microfrontends based on domain analysis can be valid in an organizational context where the responsible parties are close	Improvements to solutions
				creating intermediary layers can be a solution to avoid merging the MFEs	Proposal of new solutions
				eventually turn these MFEs into a monolith due to the system's high coupling characteristic	Proposal of new solutions
		P7	Employing an event-driven architecture can address the issue of high coupling between MFEs without necessarily merging them into a single MFE	event-driven architecture can address the issue of high coupling between MFEs	Proposal of new solutions
		P18	maybe, instead of changing informations directly between the calculation fragment and component that shows a calculate value, its possible to use a global state or global event that "consume" this information without dependency each other	use a global state or global event that "consume" this information without dependency each other	Proposal of new solutions
		P20	It does address the problem correctly although it can have others solutions, for example a shared lib between the MFEs that can specifically handle this situation, a "bridge module" to manage dependent states/values, etc.	shared lib between the MFEs that can specifically handle this situation	Proposal of new solutions
				a "bridge module" to manage dependent states/values	Proposal of new solutions
Knot Micro Frontend	If you disagree, please provide a description of what is not clear	P2	I'm a bit confused if the problem is communicating among multiple MFEs or just stablishing a communication contract. If the problem is with stablishing clear communication contracts, than the number of MFEs involved is not important to the problem definition. It's similar to implementing APIs in Microservices where you should keep a clear contract between them.	confused if the problem is communicating among multiple MFEs or just stablishing a communication contract	Improvements to problem definitions
				If the problem is with stablishing clear communication contracts, than the number of MFEs involved is not important to the problem definition	Improvements to problem definitions
	If you disagree, please provide a description of what is not clear	P2	It's related to the first question. I think the solution could state what is a good communication pattern vs a bad one. In the mentioned example, the MFE payments should be aligned with the domain level of Product, agnostic to wether it is digital or physical. Stablishing communication patterns/contracts aligned to the domain level seems a good proposal in my opinion.	solution could state what is a good communication pattern vs a bad one	Improvements to solutions
				Stablishing communication patterns/contracts aligned to the domain level	Proposal of new solutions
	In case you have a different suggestion on the problem solution, please provide a description	P13	I see that there are room to explain how this interface can be prepared to accommodate future changes	explain how this interface can be prepared to accommodate future changes	Improvements to solutions
Hub-like Dependency	If you disagree, please provide a description of what is not clear	P5	Here I have a few points: any screen, whether it is a microfrontend or not, should be resilient and have good error handling. The fallback solution will depend on many factors, such as if the error is being caused by one of the MFEs that makes the entire screen stop functioning and if it is a critical MFE that must be present on the screen. So in general, although the path presented for solving the problems makes sense, this type of issue is not specific to MFEs but rather to "hub screens" in any type of context.	any screen, whether it is a microfrontend or not, should be resilient and have good error handling	Improvements to solutions
				The fallback solution will depend on many factors	Improvements to solutions
				this type of issue is not specific to MFEs but rather to "hub screens"	Improvements to problem definitions
	If you disagree, please provide a description of what is not clear	P2	"Avoiding screens that serve as a starting point for other functionalities is recommended" how can this be avoided? If the MFE is seen from the same perspective of a component composition, it is inevitable to have aggregators. What can be done to avoid aggregators? (and can they be avoided at all?)	how can this be avoided?	Improvements to solutions
				it is inevitable to have aggregators	Improvements to solutions
		P20	Is understandable to avoid this type of screens due to his heavy use of other MFEs, having multiple dependencies and allowing a screen to be heavier by consuming external resources to mount the screen. But not use it goes against the main idea of the MFE to be contextually segregated, that when a module is updated, there has to be no worries about the liveness of other modules that are mounted together with the module updated on that screen. Is expected due to the idea of the MFE that nothing is going to be affected and that the main change must affect and be tested in it's own context, event that is consumed in a starting-point screen.	But not use it goes against the main idea of the MFE to be contextually segregated	Improvements to solutions
			The solution solves only a part of the problem. While proper error-handling is a must in any given environment, if we are talking about front-end, it's possible for each MFE to have it's own error	error-handling is a must in any given environment,	Improvements to solutions

	In case you have a different suggestion on the problem solution, please provide a description	P4	possible for each MFE to have its own error handling functions, that should: A - Disable the feature whenever needed; B - Warn the user of any inconveniences. A main screen would simply import everything that is already done and aggregate it on the same screen. It is also very important to make this "hub screen" a very simple one, without tasks that can compromise all the other MFEs. For instance, let's use the same example: A main banking screen that has charts, lists and balances. If this screen implements its own data fetch function that fails and renders the screen useless, then it is a problem. But if the crypto chart fails by itself, and displays its own error message while all the other ones are still functional, then its not much of an issue.	A main screen would simply import everything that is already done and aggregate it on the same screen.	Proposal of new solutions
				very important to make this "hub screen" a very simple one	Improvements to solutions
				A main banking screen that has charts, lists and balances. If this screen implements its own data fetch function that fails and renders the screen useless, then it is a problem.	Improvements to examples
				But if the crypto chart fails by itself, and displays its own error message while all the other ones are still functional, then its not much of an issue.	Improvements to examples
		P8	The fallback should haven't call the issued fragments	The fallback should haven't call the issued fragments	Improvements to solutions
		P20	I agree that the solution is addressed to the problem but i don't think it's the best solution. Maybe segregate even more the contexts of each module so that one don't effect the other so drastically even that they share some data and states. The interdependencies must be reviewed so that they don't affect modules that didn't even was changed.	segregate even more the contexts of each module so that one don't effect the other so drastically even that they share some data and states	Proposal of new solutions
Nano Frontend	In case you have a different suggestion on the problem solution, please provide a description	P5	Just like in the first question, the solution here will depend entirely on the organizational context, but it makes sense.	the solution here will depend entirely on the organizational context	Improvements to solutions
		P10	I agree, but the solution can be a interface or abstraction on main domain like a super domain and the little fragments can be used like a template	I agree, but the solution can be a interface or abstraction on main domain like a super domain and the little fragments can be used like a template	Proposal of new solutions
Mega Frontend	In case you have a different suggestion on the problem solution, please provide a description	P4	The solution indeed solves the problem. But we could also look from another perspective and try to prevent the problem. In my opinion, the main issue that makes monoliths come into existence, is a lack of communication between the product team and the development team. It should be well discussed between the two teams to define when two or more features are different products.	lack of communication between the product team and the development team. It should be well discussed between the two teams to define when two or more features are different products	Proposal of new solutions
Micro Frontends Greedy	If you disagree, please provide a description of what is not clear	P8	I can't separate this anti-pattern from nano or mega frontends	I can't separate this anti-pattern from nano or mega frontends	Improvements to problem definitions
	If you disagree, please provide a description of what is not clear	P2	I don't disagree but I'd be bold and state micro frontends always are born from refactor and never from feature.	I'd be bold and state micro frontends always are born from refactor and never from feature.	Improvements to solutions
	In case you have a different suggestion on the problem solution, please provide a description	P7	1) In this case, i'd focus on the business context. Instead of conducting a comprehensive review of all existing MFEs to determine the best fit for the new functionality, it's more effective to evaluate the business context of the feature in collaboration with the relevant teams. By doing so, we can ensure that the feature is integrated into an MFE managed by the appropriate team, avoiding boundary issues and ensuring cohesive development. 2) In addition to evaluating the business context and team boundaries, it is important to consider how MFEs are named and categorized. Proper naming and handling of MFEs based on broader business domains rather than specific features can significantly simplify the decision-making process when integrating new functionalities That means, naming MFEs based on broader business domains (like "payment mfe," "user mfw," or "security mfe") as opposed to specific features (like "login mfe," "OTP mfe," etc.) can help maintain a more coherent and manageable architecture, in a way that avoids the greediness	Instead of conducting a comprehensive review of all existing MFEs to determine the best fit for the new functionality, it's more e ective to evaluate the business context of the feature in collaboration with the relevant teams.	Proposal of new solutions
				Proper naming and handling of MFEs based on broader business domains rather than specific features can significantly simplify the decision-making process when integrating new functionalities	Proposal of new solutions
				naming MFEs based on broader business domains (like "payment mfe," "user mfw," or "security mfe") as opposed to specific features (like "login mfe," "OTP mfe," etc.) can help maintain a more coherent and manageable architecture, in a way that avoids the greediness	Proposal of new solutions
		P13	It would be beneficial to have the domain and all responsibilities of each MFE documented and summarized. This way, when making decisions like this, the information can support whether or not to create a new MFE.	have the domain and all responsibilities of each MFE documented and summarized. This way, when making decisions like this, the information can support whether or not to create a new MFE.	Proposal of new solutions
No CI/CD	In case you have a different suggestion on the problem solution, please provide a description	P2	CI/CD should be included as definition of done of a micro frontend.	CI/CD should be included as definition of done of a micro frontend.	Improvements to solutions
No Versioning	In case you have a different suggestion on the problem solution, please provide a description	P5	Here, I see the following problem: versioning for micro frontends does not work as well as it does for libraries because the final build is usually unique. If we consider the Knot Micro Frontend anti-pattern, where there are versions where two screens, A and B, are present, and screen B changes its communication interface, screen A could not continue to depend on the previous version of screen B because there would be "two" screen Bs in the final build. For "smaller" dependencies, like an isolated component or a shared function, versioning might work well, but in this case, versioning might only partially solve the problem.	versioning for micro frontends does not work as well as it does for libraries because the final build is usually unique	Improvements to solutions
				For "smaller" dependencies, like an isolated component or a shared function, versioning might work well	Improvements to solutions
		P12	"It is essential to adopt the Semantic Versioning", não é essencial considerando que o versionamento semantico não é a única opção viável. Em projetos que possuem release diariamente, talvez o ideal seja usar de fato o calendar versioning.	considerando que o versionamento semantico não é a única opção viável. Em projetos que possuem release diariamente, talvez o ideal seja usar de fato o calendar versioning	Improvements to solutions

Lack of Skeleton	If you disagree, please provide a description of what is not clear	P2	The solution seems to be restrict in terms of technology decisions. One of the big advantages of Micro Frontends is being able to be tech agnostic, and have different MFEs in dffierent technologies to better suit its needs. Is this still a problem if such freedom is intended? How to provide boiler plates agnostic of frameworks?	Is this still a problem if such freedom is intended?	Improvements to problem definitions
				How to provide boiler plates agnostic of frameworks?	Improvements to solutions
	In case you have a different suggestion on the problem solution, please provide a description	P15	I add that keeping the skeleton updated is also very important	keeping the skeleton updated is also very important	Improvements to solutions
Common Ownership	If you disagree, please provide a description of what is not clear	P15	I don't believe software should be modularized due to team size. Naturally, larger software involves more people, but I believe small teams can also benefit from software modularization, such as separation of layers and responsibilities, observability and maintainability	Naturally, larger software involves more people, but I believe small teams can also benefit from software modularization, such as separation of layers and responsibilities, observability and maintainability	Improvements to problem definitions
	If you disagree, please provide a description of what is not clear	P15	Yes and No, teams with knowledge in more than one context tend to do better when solving unusual problems. In my opinion, I don't believe that the responsibility of a team that works with micro frontends should be restricted to its own context	I don't believe that the responsibility of a team that works with micro frontends should be restricted to its own context	Improvements to solutions
	In case you have a different suggestion on the problem solution, please provide a description	P7	If the problem of a single team managing all (or a lot of) MFEs persists even with context definition, defining shared components and libraries can make boundary definition easier. Shared components reduce duplication and standardize functionalities, simplifying boundary definition. And by standradizing functionalities, it could be possible to get rid of some MFEs after some refactoring.	defining shared components and libraries can make boundary definition easier.	Proposal of new solutions
				Shared components reduce duplication and standardize functionalities, simplifying boundary definition.	Proposal of new solutions
				by standradizing functionalities, it could be possible to get rid of some MFEs after some refactoring.	Proposal of new solutions
Golden Hammer	If you disagree, please provide a description of what is not clear	P11	Manter padrões de desenvolvimento comuns na organização, no meu ponto de vista é benéfica: 1- Tecnologias podem ser reutilizáveis em vários projetos. 2 - Melhoram as experiências de engenharia de software. 3 - Fornecem transparência ao design das aplicações. 4 - Acelera o processo de desenvolvimento.	Manter padrões de desenvolvimento comuns na organização, no meu ponto de vista é benéfica: 1- Tecnologias podem ser reutilizáveis em vários projetos. 2 - Melhoram as experiências de engenharia de software. 3 - Fornecem transparência ao design das aplicações. 4 - Acelera o processo de desenvolvimento.	Improvements to solutions
	In case you have a different suggestion on the problem solution, please provide a description	P7	It is hard to define the right technology on stone at one point in time and having it stay the most adequate during the product evolution. To address the specific needs of each MFE, adopting a hybrid technology approach supported by a common facade, such as a Backend for Frontend (BFF) layer, and using feature flags to manage gradual migration and experimentation might be a better approach (and would also save a lot of time on decision-making processes for big companies). This allows each MFE to utilize the most suitable technology, while maintaining overall architectural coherence. Feature flags enable testing and gradual rollout and routing of new technologies without disrupting the entire system	adopting a hybrid technology approach supported by a common facade, such as a Backend for Frontend (BFF) layer, and using feature flags to manage gradual migration and experimentation might be a better approach	Proposal of new solutions
	If you do agree, please provide a description of the problem.	P2	Selecting the wrong type of micro frontends based on the user needs. There are mainly two ways to integrate micro frontends: buildtime and runtime. The decision on which to adhere reflects deeply in the teams' and users' needs more than the technical pros and cons each of them offer. For example. in my experience each team dealt with its own MFE in a scilo, with the integrations happening only at the API level. Depending on other teams for frontend development in any way would only slow them down, so the runtime integration was the chosen integration strategy. If build time was chosen, they would still depend on each other for deployment even though the integration of the screens were minimum. Same thing could happen otherwise, when there is heavy integration that is safe to do it in buildtime. Choosing runtime integration strategy could generate multiple bugs and instability in different environments.	There are mainly two ways to integrate micro frontends: buildtime and runtime. The decision on which to adhere reflects deeply in the teams' and users' needs more than the technical pros and cons each of them o er	New anti-patterns
				in my experience each team dealt with its own MFE in a scilo, with the integrations happening only at the API level. Depending on other teams for frontend development in any way would only slow them down, so the runtime integration was the chosen integration strategy. If build time was chosen, they would still depend on each other for deployment even though the integration of the screens were minimum	New anti-patterns
		P3	Inconsistent User Experience Fragmented State Management Complex Inter-MFE Communication Overhead of Independent Deployments Security and Authentication Challenges	Inconsistent User Experience Fragmented State Management Complex Inter-MFE Communication Overhead of Independent Deployments Security and Authentication Challenges	New anti-patterns
		P5	- Performance Bottlenecks: addressing solutions for dynamic loading can be a good topic. - Security Risks: especially in the context where micro frontends use different technologies, which means each one can introduce different types of risk.	Performance Bottlenecks: addressing solutions for dynamic loading can be a good topic	New anti-patterns
				Security Risks: especially in the context where micro frontends use different technologies, which means each one can introduce different types of risk	New anti-patterns
		P6	Vários times pequenos cuidando de muitos mfes por vez. Imaginando um time que tenha 3 devs e mantém 5 mfes	Vários times pequenos cuidando de muitos mfes por vez. Imaginando um time que tenha 3 devs e mantém 5 mfes	New anti-patterns
			1) Poor state management: Data persistence on MFEs when each frontend manages the state independently	Poor state management: Data persistence on MFEs when each frontend manages the state independently.	New anti-patterns

Final Considerations		P7	Independently. For example, when a user navigates back to a page with different data or state, inconsistent or unexpected behavior can occur, leading to a poor user experience. This issue arises because each micro frontend typically manages its state independently,	For example, when a user navigates back to a page with different data or state, inconsistent or unexpected behavior can occur, leading to a poor user experience. This issue arises because each micro frontend typically manages its state independently	New anti-patterns
		P10	One of the patters is related to include use more than one technology to extract more from SEO or something else, that could cause a more complex architecture.	that could cause a more complex architecture	Improvements to solutions
			I think if you include observability overwhelmed, or patterns in observability that would be nice.	I think if you include observability overwhelmed, or patterns in observability that would be nice	New anti-patterns
	How do you think this catalog would help improve the quality of micro frontend architecture in your work?	P1	This catalog can act as a checklist to ensure good practices and avoid anti-patterns in the micro frontend context	This catalog can act as a checklist to ensure good practices and avoid anti-patterns in the micro frontend context	How to use the catalog
		P2	It will serve as a guide for some DOs and DONTs that are missing in the Micro Frontends world. The term and technology is fairly new and such patterns are not well stablished in the software community yet. This is great work!	It will serve as a guide for some DOs and DONTs that are missing in the Micro Frontends world	How to use the catalog
				This is great work	Commendations for the catalog
				The term and technology is fairly new and such patterns are not well stablished in the software community yet	Commendations for the catalog
		P4	It's a well detailed "bible", containing the most essentials "don'ts" of working with micro frontends. It is a nice guide for any new or experienced developer and would definitely read it and pass it along if it is ever published.	containing the most essentials "don'ts" of working with micro frontends	How to use the catalog
				It is a nice guide for any new or experienced developer	How to use the catalog
		P5	This catalog highlights several real-world problems encountered in the day-to-day work with micro frontends. Many proposed solutions are sensible in various contexts, and even those that seem less practical from my perspective serve as valuable discussion points. These discussions can help us develop effective solutions to the identified issues.	This catalog highlights several real-world problems encountered in the day-to-day work with micro frontends	Commendations for the catalog
				Many proposed solutions are sensible in various contexts, and even those that seem less practical from my perspective serve as valuable discussion points	Commendations for the catalog
				These discussions can help us develop e ective solutions to the identified issues	How to use the catalog
		P6	Ajudando a perceber alguns anti padrões que por conta do dia-a-dia de trabalho se tornam "padrões" na empresa, dando uma visão geral do problema e como podemos solucionar e evitar que eles se propaguem ainda mais.	Ajudando a perceber alguns anti padrões que por conta do dia-a-dia de trabalho se tornam "padrões" na empresa	How to use the catalog
				dando uma visão geral do problema e como podemos solucionar e evitar que eles se propaguem ainda mais	How to use the catalog
		P7	As a valuable resource for training new team members and onboarding them to micro frontend projects, also to keep up to date with latest patterns and to encounter possible problems as the architecture evolves	training new team members and onboarding them to micro frontend projects	How to use the catalog
				keep up to date with latest patterns and to encounter possible problems as the architecture evolves	How to use the catalog
		P8	Creating moments with teams to discuss this catalog and sharing this knowledge to improve the frontend architecture.	discuss this catalog and sharing this knowledge to improve the frontend architecture	How to use the catalog
		P9	Ajuda a pensar nas decisões de arquitetura e na decisão de usar a arquitetura de micro frontends ou não. Vi muitos problemas que já presenciei no dia a dia, porém não era um problema identificado e por isso era apenas ignorado. Acredito que possa ajudar a identificar problemas em andamento e trabalhar na solução.	pensar nas decisões de arquitetura e na decisão de usar a arquitetura de micro frontends ou não	How to use the catalog
				Vi muitos problemas que já presenciei no dia a dia, porém não era um problema identificado e por isso era apenas ignorado	How to use the catalog
				Acredito que possa ajudar a identificar problemas em andamento e trabalhar na solução	How to use the catalog
		P10	That will be a guide to think about our MFE and communication to back-end to.	guide to think about our MFE and communication to back-end to.	How to use the catalog
		P13	It makes a perfect checklist to use when designing a new MFE project or even to review an already existing one	checklist to use when designing a new MFE project	How to use the catalog
				review an already existing one	How to use the catalog
		P14	By listing directly many issues we may find while developing micro frontend architecture, it helps to avoid such mistakes	By listing directly many issues we may find while developing micro frontend architecture, it helps to avoid such mistakes	How to use the catalog
		P15	Help mainly with mega frontends and single technology in all micro frontends	Help mainly with mega frontends and single technology in all micro frontends	How to use the catalog
		P17	I think this anti-pattern catalog is very useful for sharing information about micro frontend, both for developers who do not have experience with micro frontend and for developers with experience. Helps make decisions about when to adopt this pattern or not, and when to break into a new MFE.	sharing information about micro frontend	How to use the catalog
				both for developers who do not have experience with micro frontend and for developers with experience	How to use the catalog
		P18	I think that a catalog for micro frontend architecture can significantly enhance the quality of development, a lot of points that were addressed in this form creates a lack of efficiency and scalability in all development and maintaining process, furthermore, have a catalog like a guide before thinking in MFE and during the development is a very useful tool for a dev team	significantly enhance the quality of development	Commendations for the catalog
				have a catalog like a guide before thinking in MFE	How to use the catalog
		P20	It will help by making me more self-aware of the possible breaches that the use of MFE can make. Also, the last few questions were about the use (or not) of a MFE and where it is necessary and where	making me more self-aware of the possible breaches that the use of MFE can make	How to use the catalog

		P20	it isn't, a very important point to be raised due to the "hype" that the use of a technology can have, and is not always the use case of the architecture/solution that is being made.	the last few questions were about the use (or not) of a MFE and where it is necessary and where it isn't, a very important point to be raised due to the "hype" that the use of a technology can have	How to use the catalog
	Do you have any suggestions for improving the anti-patterns catalog?	P4	Don't know if this is an issue with Forms, but a "wall of text" is never too friendly, specially on a bright computer screen. I would use some flow charts to exemplify most of the anti-patterns and make it more readable. Or event some prototyped examples, since most of the examples uses a lot of terms such as "screens". Would be nice to see a picture of it instead of plain description.	I would use some flow charts to exemplify most of the anti-patterns and make it more readable	Improvements to the catalog
				Would be nice to see a picture of it instead of plain description	Improvements to the catalog
		P7	Addressing common challenges faced by development teams according to their experience in real world scenarios and by technology	Addressing common challenges faced by development teams according to their experience in real world scenarios and by technology	Improvements to the catalog
		P8	This catalog need to be shared in a website after this study have finished like refactoring.guru	shared in a website after this study have finished like refactoring.guru	Improvements to the catalog
		P9	Não, achei ótima a separação por categorias.	ótima a separação por categorias	Commendations for the catalog
		P10	Like I said, observability patters is a good start	observability patterns is a good start	New anti-patterns
		P15	I believe that the problem of managing dependencies between modules would be a good anti-pattern to solve.	managing dependencies between modules would be a good anti-pattern to solve	New anti-patterns
		P17	Maybe if there were images for some anti-pattern it would be interesting. Showing the MFEs, the communication between them, etc.	images for some anti-pattern it would be interesting	Improvements to the catalog
		P18	maybe add some diagrams and images help to understanding some examples	add some diagrams and images help to understanding some examples	Improvements to the catalog

B

Controlled Experiment Objects

This Appendix presents the description of two objects used in the controlled experiment described in Chapter 5. These objects are specifications of two compatible Micro Frontend architectures, which include a description of the application and the proposed Micro Frontends.

B.1 Object 1 - Mercado Livre

Mercado Livre is one of the largest e-commerce platforms in Latin America, where users can buy and sell a wide variety of products, ranging from electronics, clothing, and accessories to home and gardening items. The site offers a secure payment system, Mercado Pago, and shipping services through Mercado Envios, facilitating logistics and product delivery. Additionally, Mercado Livre features a user-friendly interface, seller ratings, special offers and promotions, and is known for the vast number of products available from both individual sellers and large companies.

Here is a list of the main features of Mercado Livre that will be considered during this exercise:

- **Buying and Selling Products**: Users can buy and sell a wide variety of products, from new items to used ones.
- **Search System and Filters**: Allows users to search for specific products using filters such as price, location, condition (new/used), and more.

- **Mercado Pago**: Integrated payment platform offering various payment options, such as credit card, boleto (bank slip), bank transfer, and account balance.
- **Mercado Envios**: Logistics and delivery service that allows order tracking, automatic shipping cost calculation, and nationwide delivery, with different shipping options.
- **Purchase and Sales History**: Users can track the history of their purchases and sales, managing past and current transactions.
- **Shopping Cart**: A feature that allows users to add multiple items from different sellers and purchase them in a single order.
- **Offers and Promotions**: A page dedicated to promotions and flash sales, with discounts and free shipping on certain products.

Consider that the website was developed using the Micro Frontends listed in the subsections below. Note that not all the screens of each are listed, only a few for illustration.

B.1.1 mfe-users

Includes screens that show the registered user's data, such as personal data, account information, address list, card list, etc. Figure 61 and Figure 62 present screens of the mfe-users.

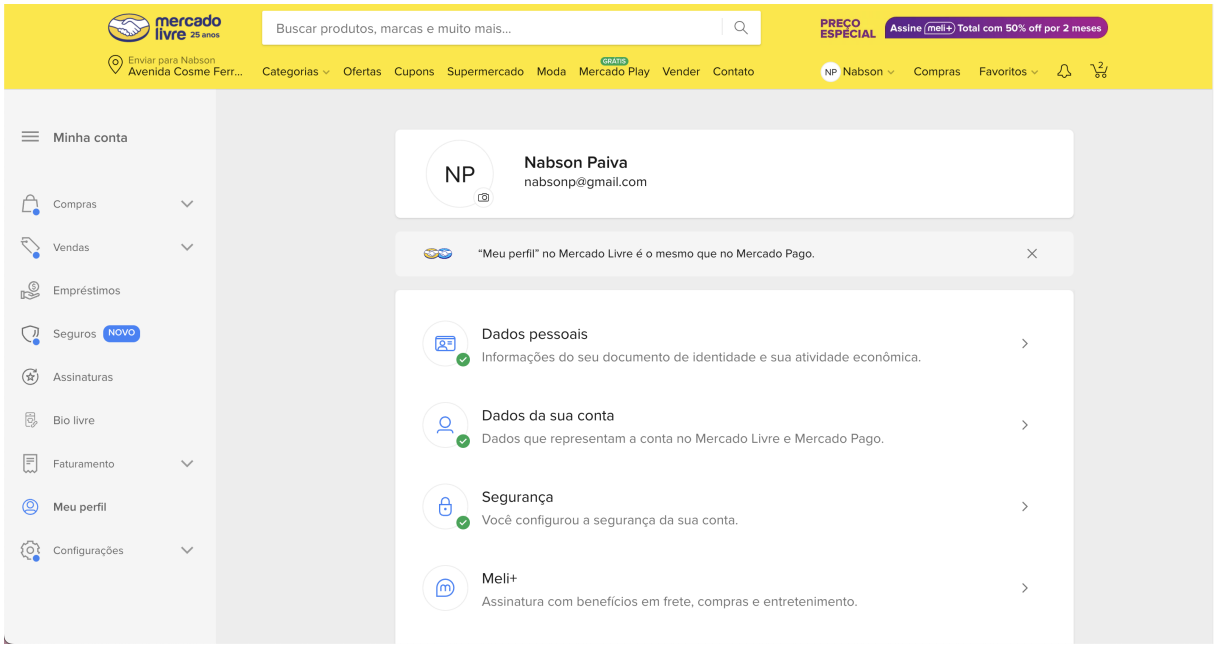

Figure 61 – My profile screen of mfe-users.

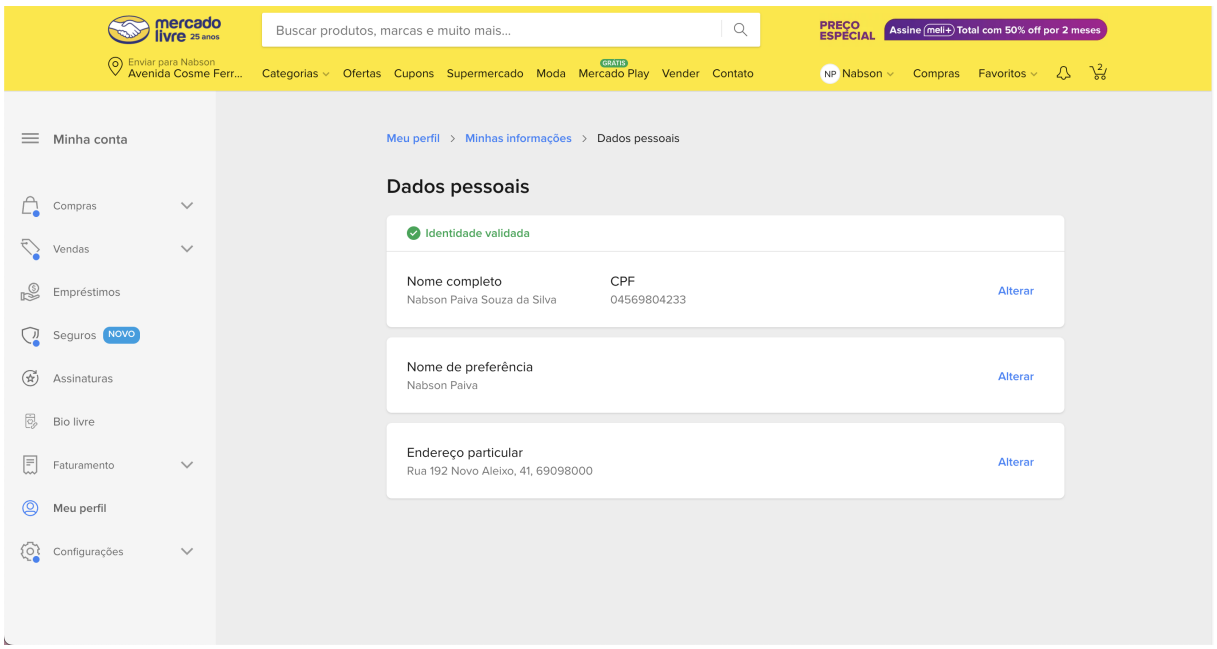

Figure 62 – Personal data screen of mfe-users.

B.1.2 mfe-security

Includes screens related to the security of the user's data and account, such as login, privacy policies, two-factor authentication, registration, etc. Figure 63 and Figure 64 present screens of the mfe-security.

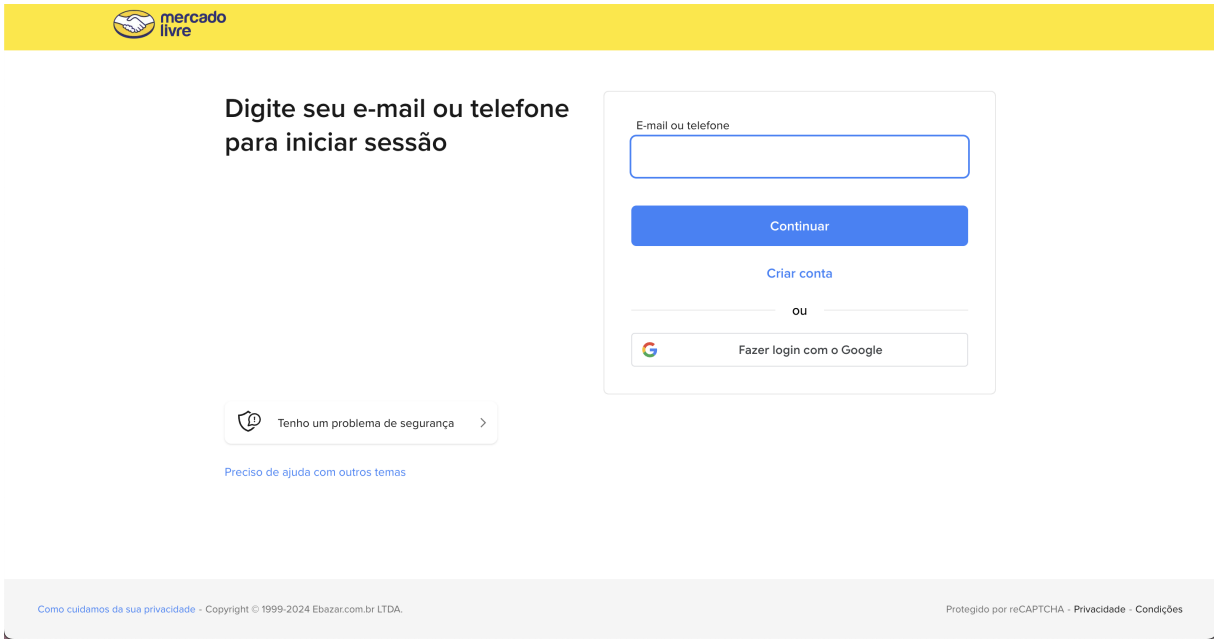

Figure 63 – Login screen of mfe-security.

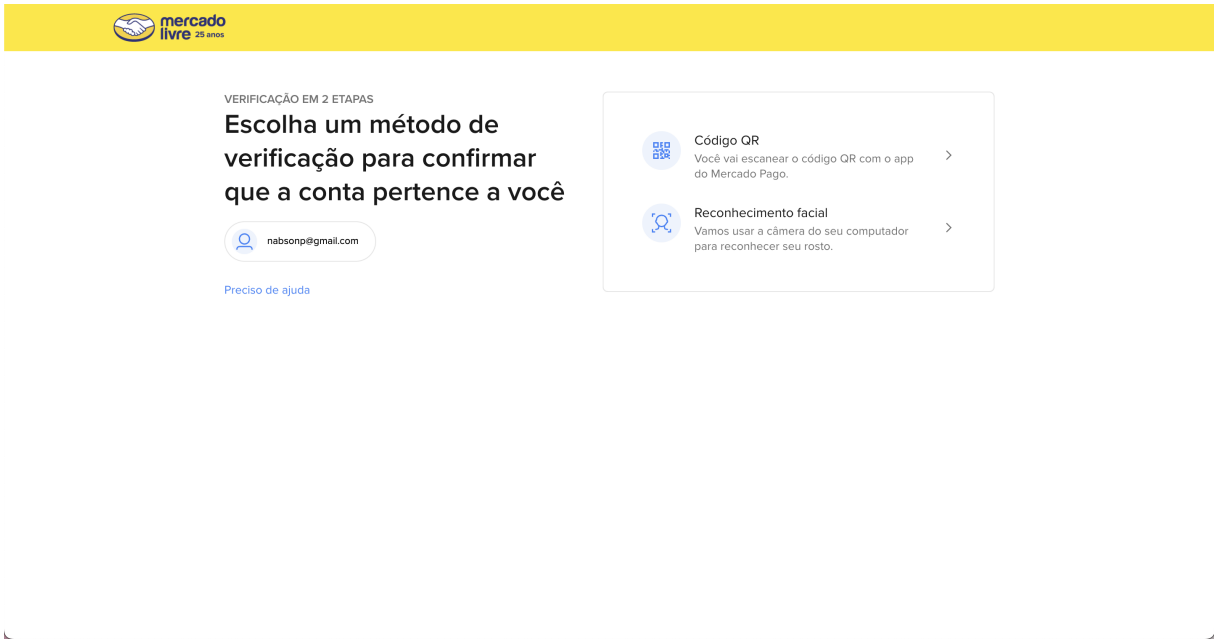

Figure 64 – Two-factor authentication screen of mfe-security.

B.1.3 mfe-store

Includes screens related to the stores and the products sold by them, such as: product details, search history, product sales, store registration. Figure 65, Figure 66, and Figure 67 present screens of the mfe-store. Figure 68 presents a fragment exported by

mfe-store.

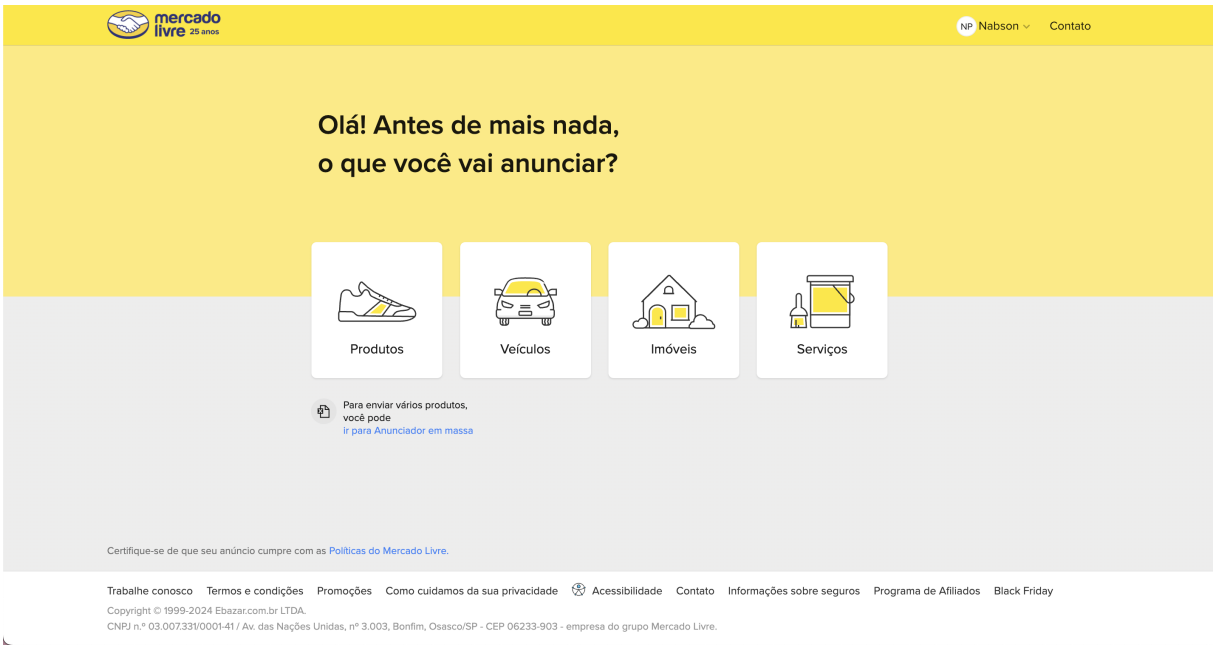

Figure 65 – Products sale screen of mfe-store.

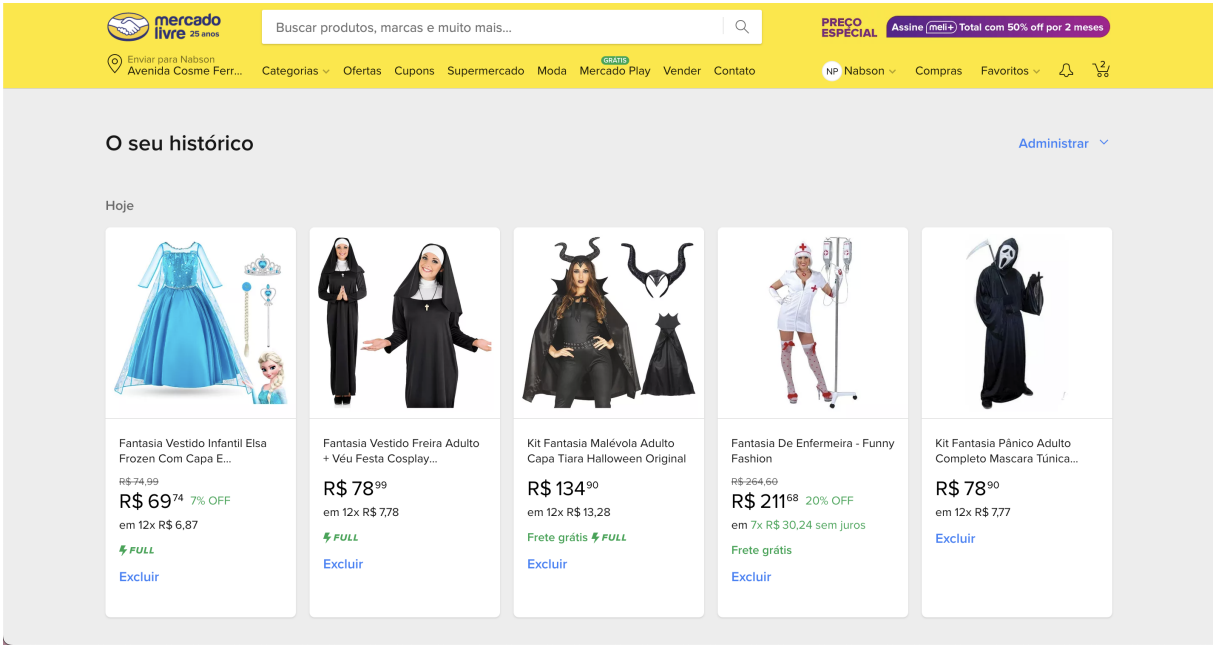

Figure 66 – History screen of mfe-store.

Figure 67 – Product details screen of mfe-store.

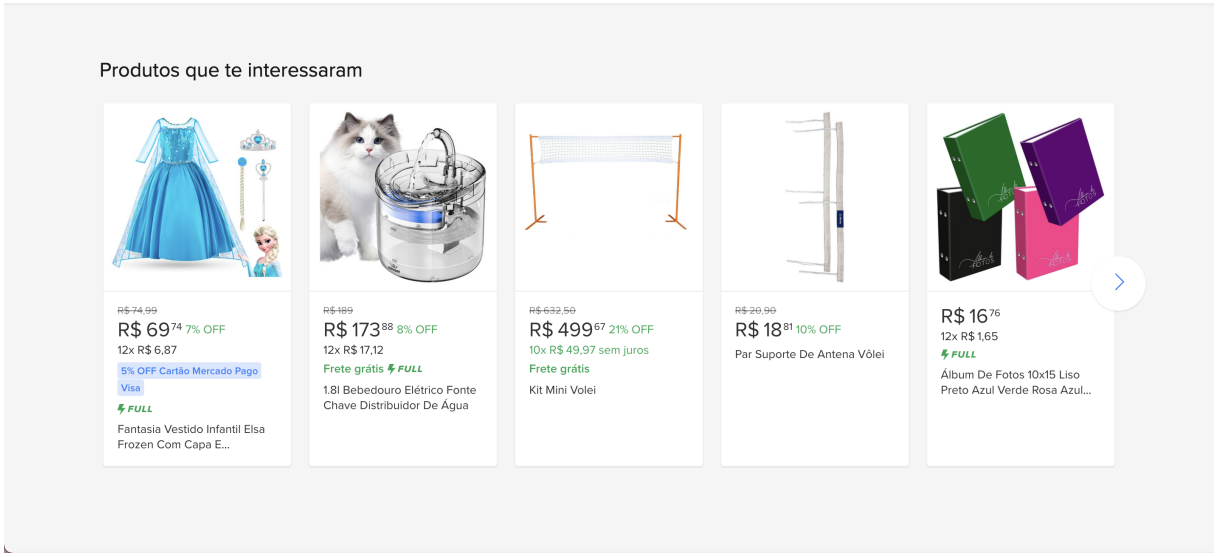

Figure 68 – Recommended products fragment of mfe-store.

B.1.4 mfe-purchase

Includes the implementation of all screens related to purchases, deliveries, and payments in Mercado Livre, including Cart, Payment Confirmation, Order Confirmation, Purchase History, Delivery Tracking, shipping cost calculation with different partners, among others. Figure 69 and Figure 70 present screens of the mfe-purchase.

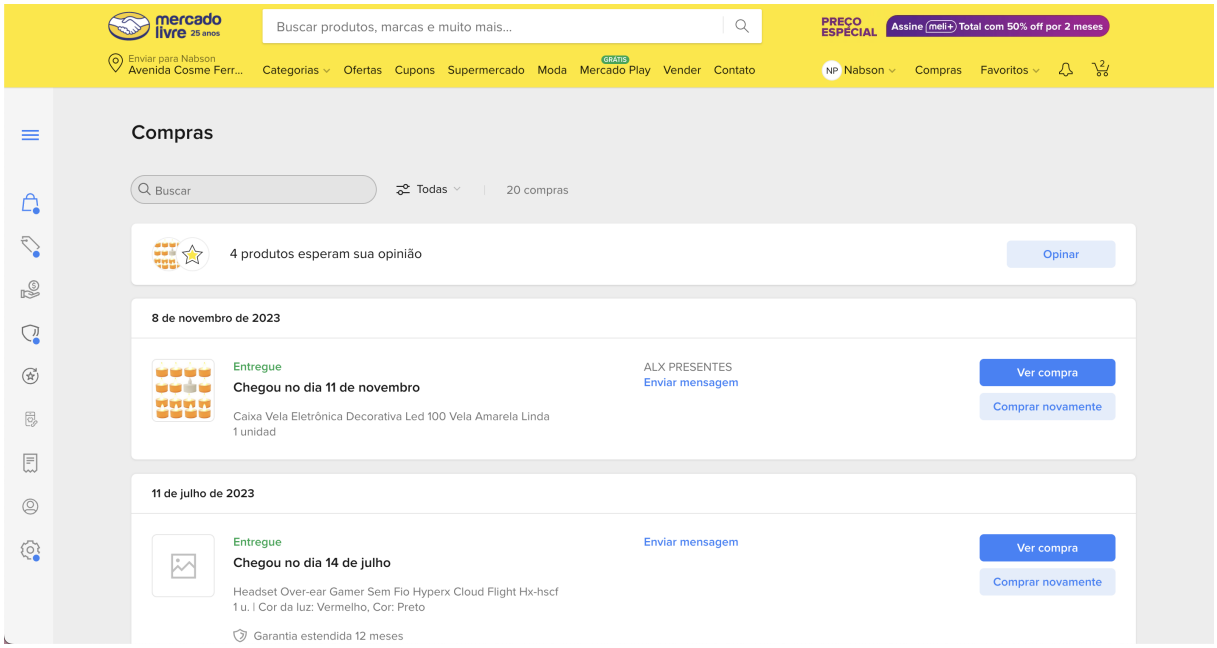

Figure 69 – Purchases screen of mfe-purchase.

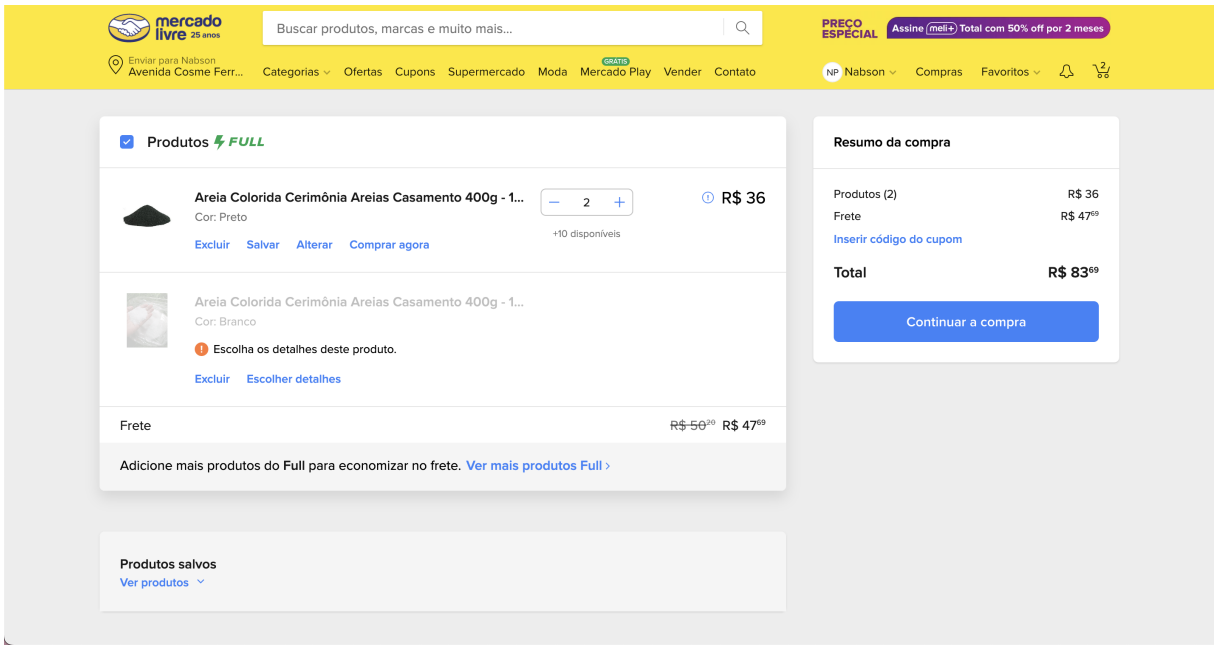

Figure 70 – Cart screen of mfe-purchase.

B.1.5 mfe-search

Includes the implementation of the search screen and is used by all MFEs that want to display a list of products, whether to show offers (mfe-store), or list products from a store. Figure 71 presents a screen of the mfe-search.

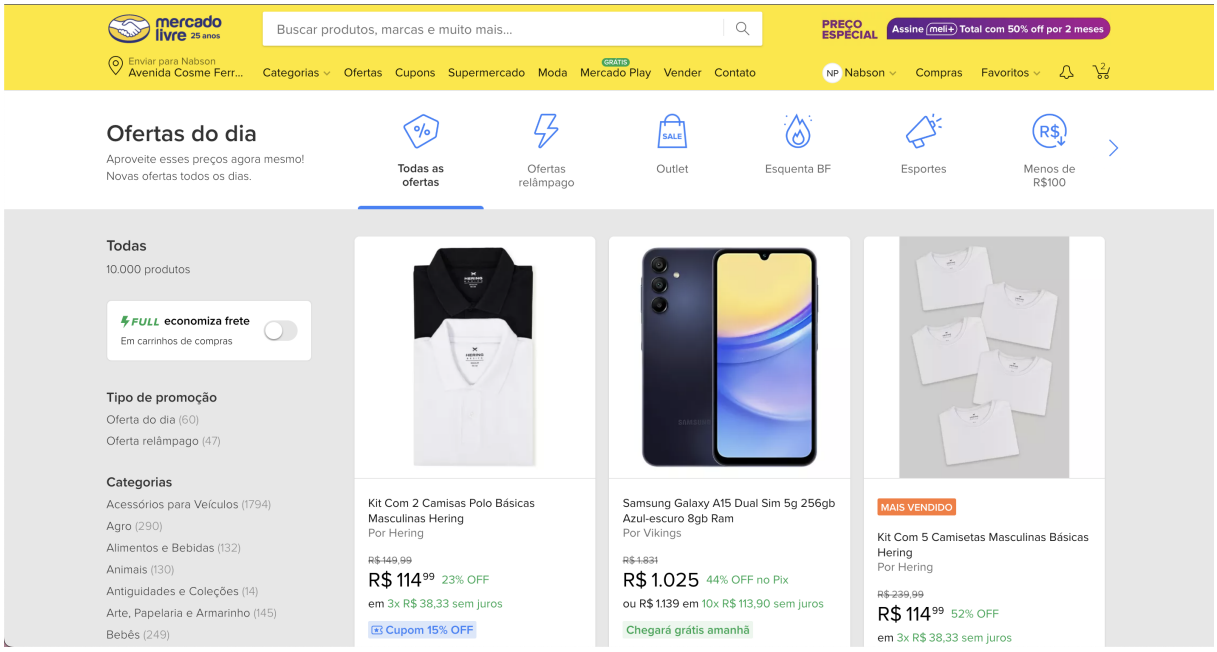

Figure 71 – Search screen of mfe-search.

B.1.6 mfe-home

Includes the implementation of the main Mercado Livre screen, which contains several fragments that lead to different MFEs. Figure 72 presents a screen of the mfe-search.

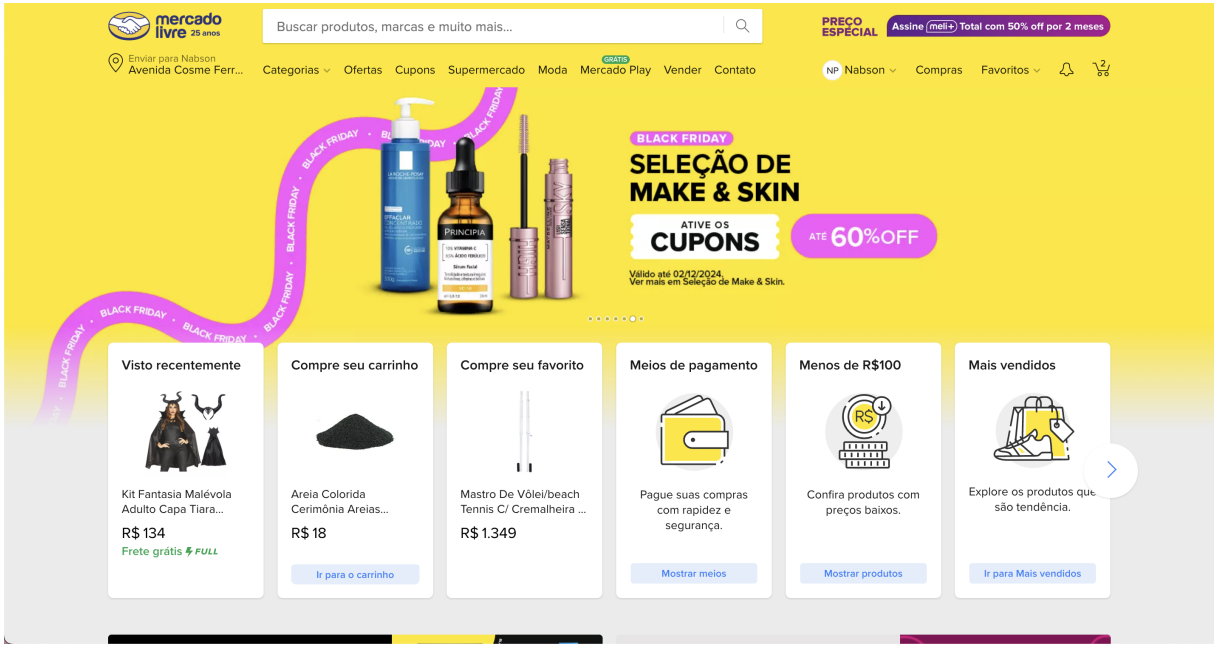

Figure 72 – Home screen of mfe-search.

B.2 Object 2 - Nubank

Nubank is a Brazilian digital bank that offers financial services through an intuitive mobile app with no bureaucracy. The Nubank app is a central piece of this experience and provides a range of features that allow users to manage their finances independently and efficiently, without the need to interact with physical branches.

- **Credit card**: management of physical card, credit limit management, virtual credit card management, access to card statements.
- **Digital account**: transfers via PIX and TED to the Nubank digital account (NuConta), as well as a detailed statement of account transactions.
- **CDB investments**: creation of personalized investment boxes, where the balance earns 100% of the CDI.
- **Payments**: payment of bills or PIX QR Codes directly in the app using the account balance or credit card limit.
- **Loans**: offering of personal loans, payroll loans, or FGTS anticipation through Nubank.

Consider that the app was developed using the following Micro Frontends listed in the subsections below. Note that not all screens of each one are listed, just some for illustration.

B.2.1 mfe-users

mfe-users: Possui a implementação da tela de dados do usuário e alteração de foto de perfil. Figure 73 presents a screen of the mfe-users.

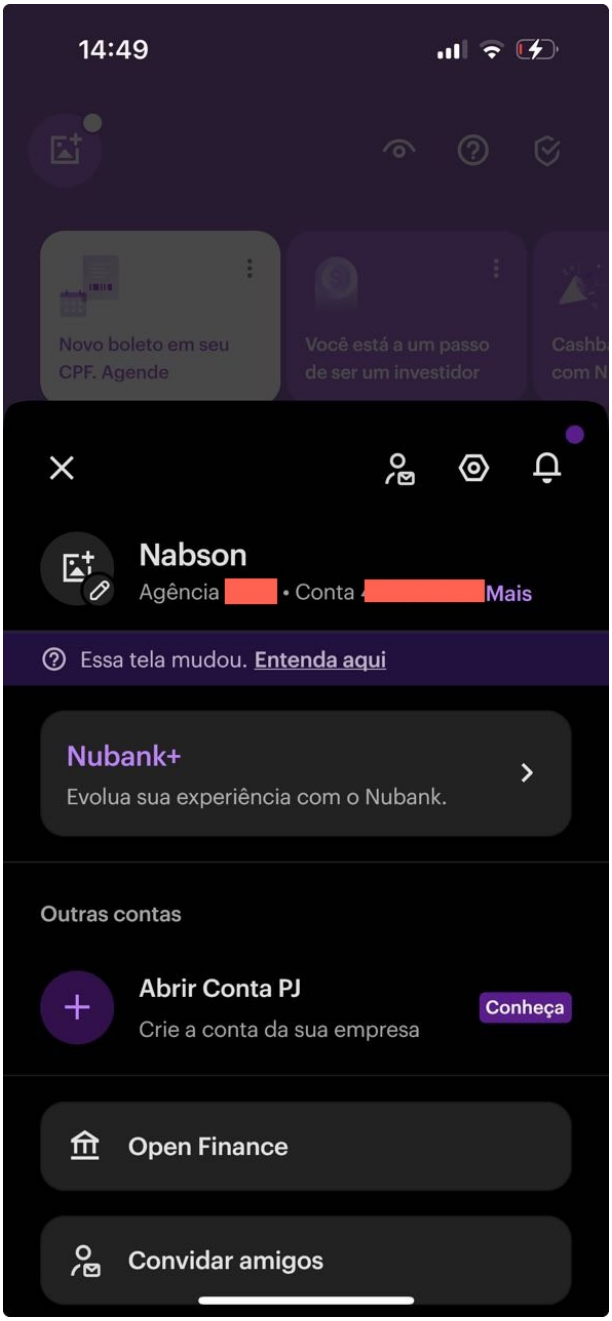

Figure 73 – Profile screen of mfe-users.

B.2.2 mfe-security

Includes the implementation of screens related to the security of the user's data and account, such as login, forgot my password, registration, etc. Figure 74 and Figure 75 present screens of mfe-security.

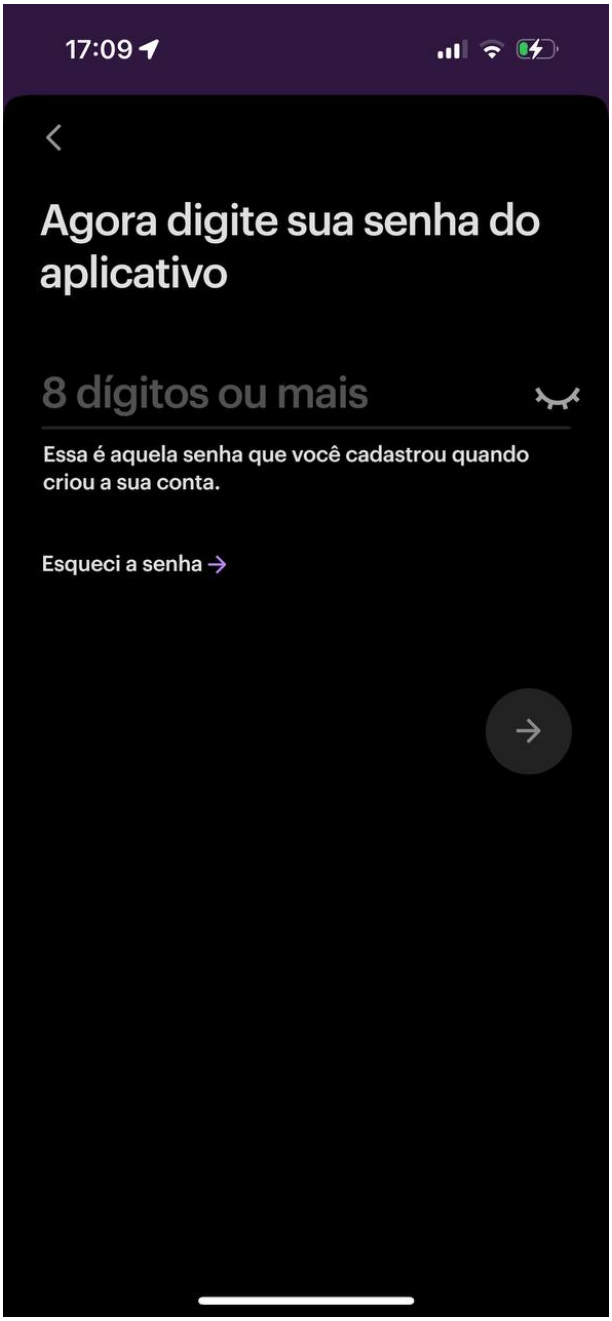

Figure 74 – Login screen of mfe-security.

Figure 75 – First screen of sign up of mfe-security.

B.2.3 mfe-cards

Includes the implementation of all screens related to credit cards (physical or virtual), including screens such as card statement, bill payment, limit adjustment, bill summary, bill details, automatic debit setup, among others. Figure 76, Figure 77, and Figure 78 present screens of mfe-cards.

Figure 76 – Card invoices screen of mfe-cards.

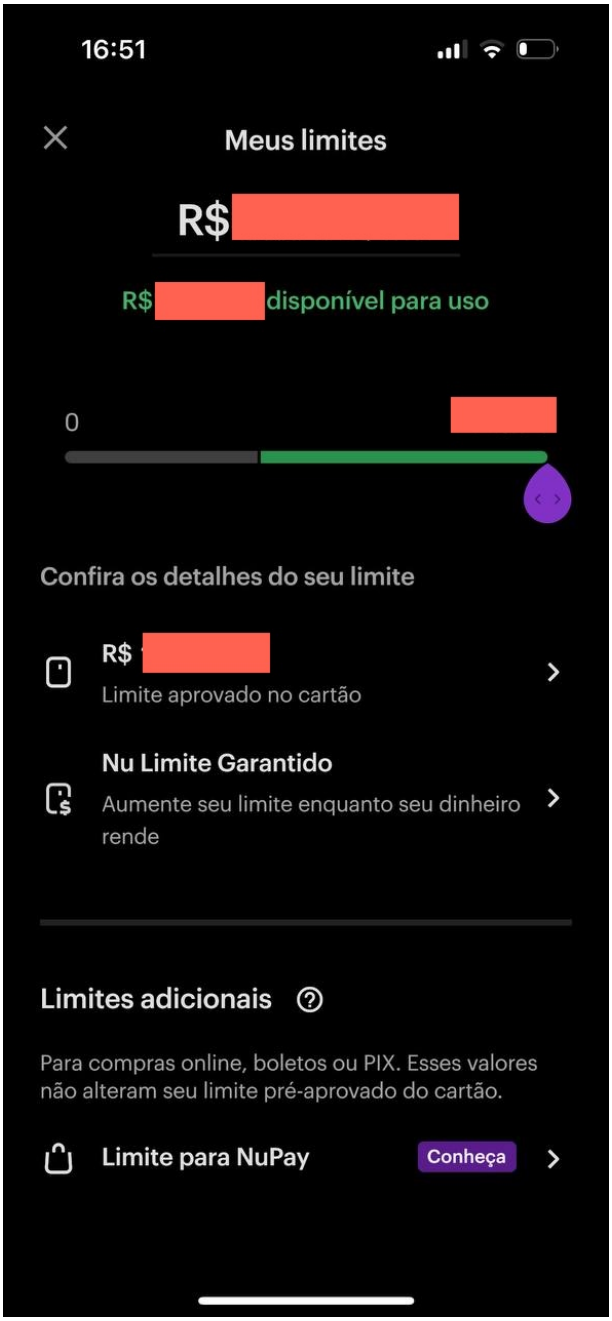

Figure 77 – Limits screen of mfe-cards.

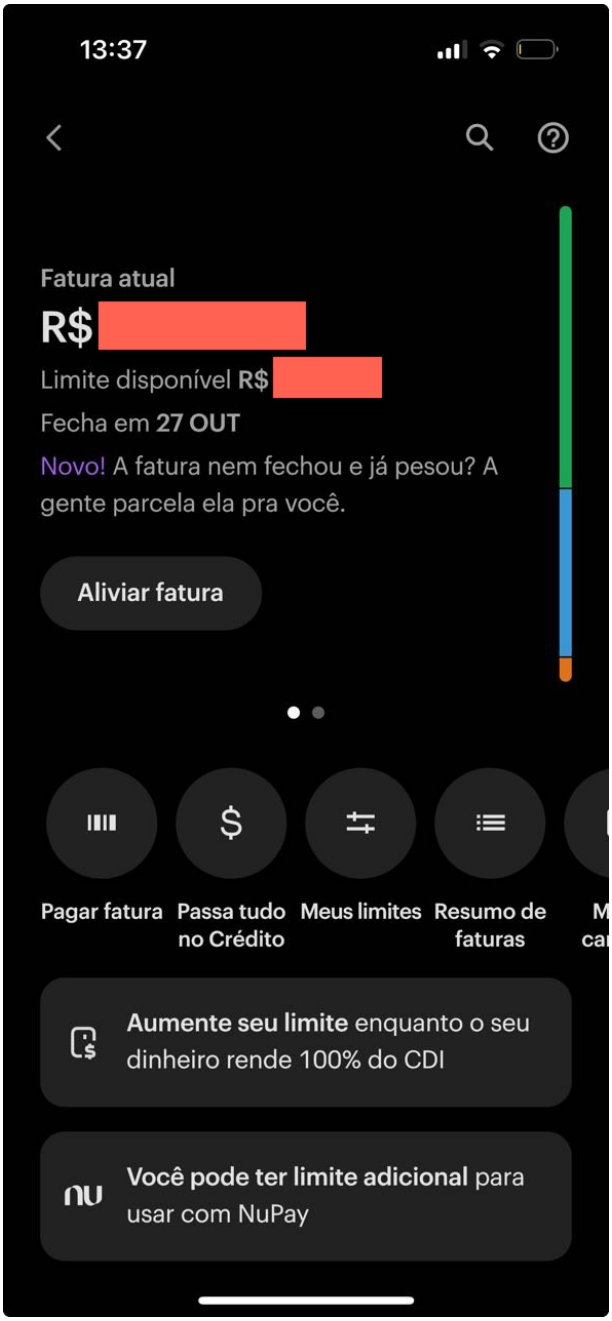

Figure 78 – Current card invoice screen of mfe-cards.

B.2.4 mfe-digital-account

Includes all screens related to the user's digital account, including user data, transfers via PIX, PIX key management, digital account statement, and bill payments. Figure 79, Figure 80, and Figure 81 present screens of mfe-digital-account.

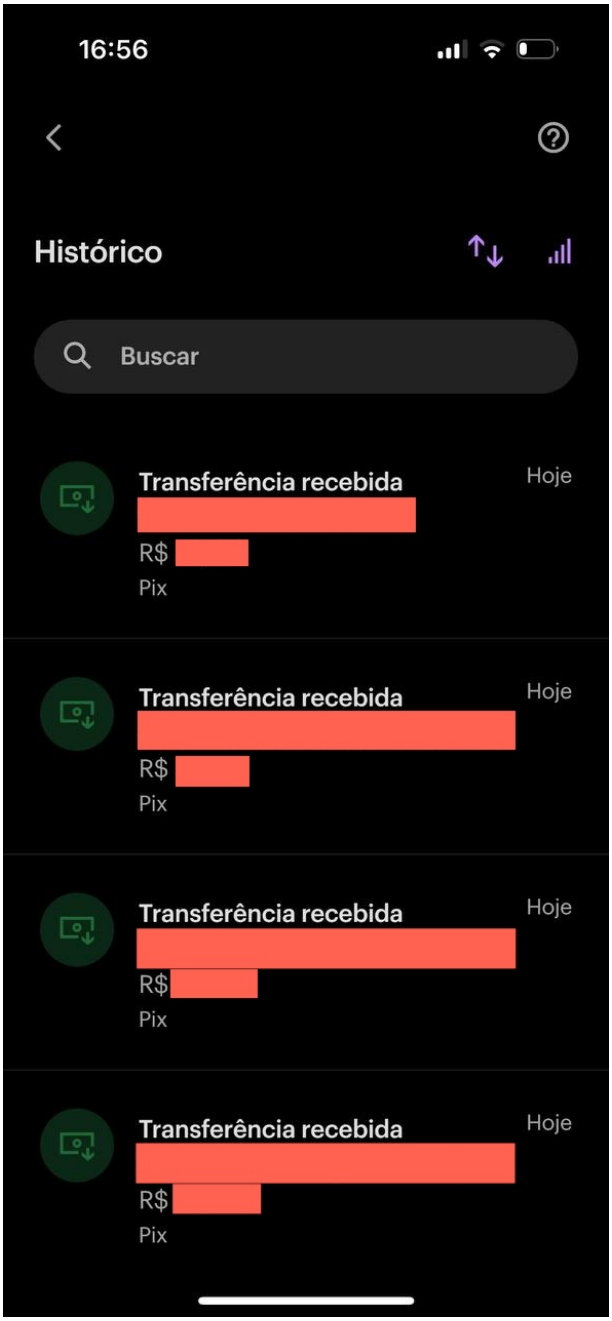

Figure 79 – Account statement screen of mfe-digital-account.

Figure 80 – Account balance screen of mfe-digital-account.

Figure 81 – Pix transfer screen of mfe-digital-account.

B.2.5 mfe-loan

Includes the screens related to Nubank's loan service, such as the loan list, loan simulation screen, loan option details screen, among others. Figure 82 and Figure 83 present screens of mfe-loan. Figure 84 presents a fragment of mfe-loan.

Figure 82 – Loans list screen of mfe-loan.

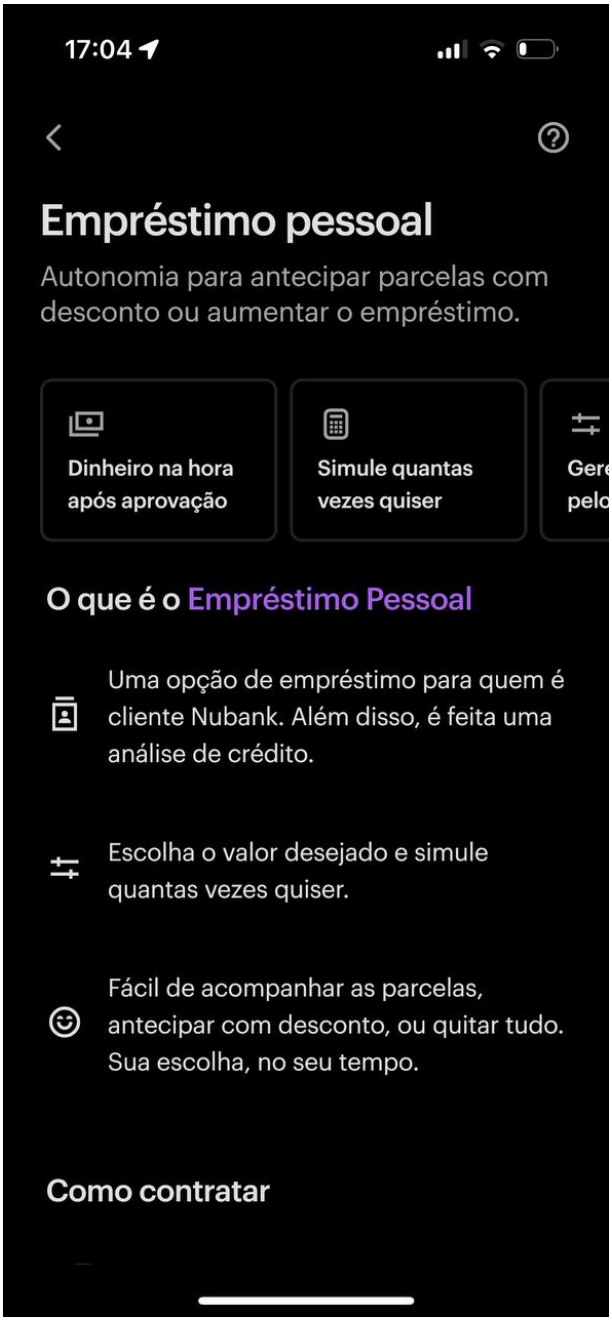

Figure 83 – Loan details screen of mfe-loan.

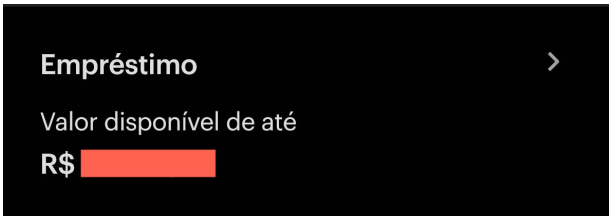

Figure 84 – Loan section fragment of mfe-loan.

B.2.6 mfe-investment

Includes the screens related to CDB investment boxes, such as the main investment screen, box details, box editing, saving and redeeming money from a box, among others. Figure 85 and Figure 86 present screens of mfe-investment. Figure 87 presents a fragment of mfe-investment.

Figure 85 – Investments list screen of mfe-investment.

Figure 86 – Investment details screen of mfe-investment.

Figure 87 – Loan section fragment of mfe-investment.

B.2.7 mfe-home

Includes the implementation of the main screen of Nubank, which contains several fragments that lead to different MFEs. Figure 88 presents a screen of mfe-investment.

Figure 88 – Home screen of mfe-home.

C

Controlled Experiment Assessments

This Appendix presents the questions of the two assessments of the controlled experiment described in Chapter 5. We defined nine questions and adjusted them to each object.

C.1 Object 1 Questions

Consider that you are a Junior Developer who will start working at the company that maintains the system being evaluated. You have just learned more about the system's architecture, and you have been asked to analyze the system in terms of MFE.

1. Would you model the system using Micro Frontends differently? Would you keep the same MFEs, create new ones, or alter the existing ones? Consider also the screens and fragments.

Consider that you are a Junior Developer who will start working at Mercado Livre. Answer how you would solve each of the tasks below. Feel free to propose new MFEs, shared libraries, screens, and/or fragments, as well as modify existing ones.

1. Mercado Livre is launching Meli+ (meliplus), the loyalty program offering free shipping on all purchases, cashback, exclusive installment options, and free access to Disney+. The feature consists of the visual components presented in Figures

89, 90, 91, 92, 93, and 94. What architectural changes are necessary to develop the above components?

Figure 89 – Meli+ Offer Screen: The screen displays the benefits and available subscription plans.

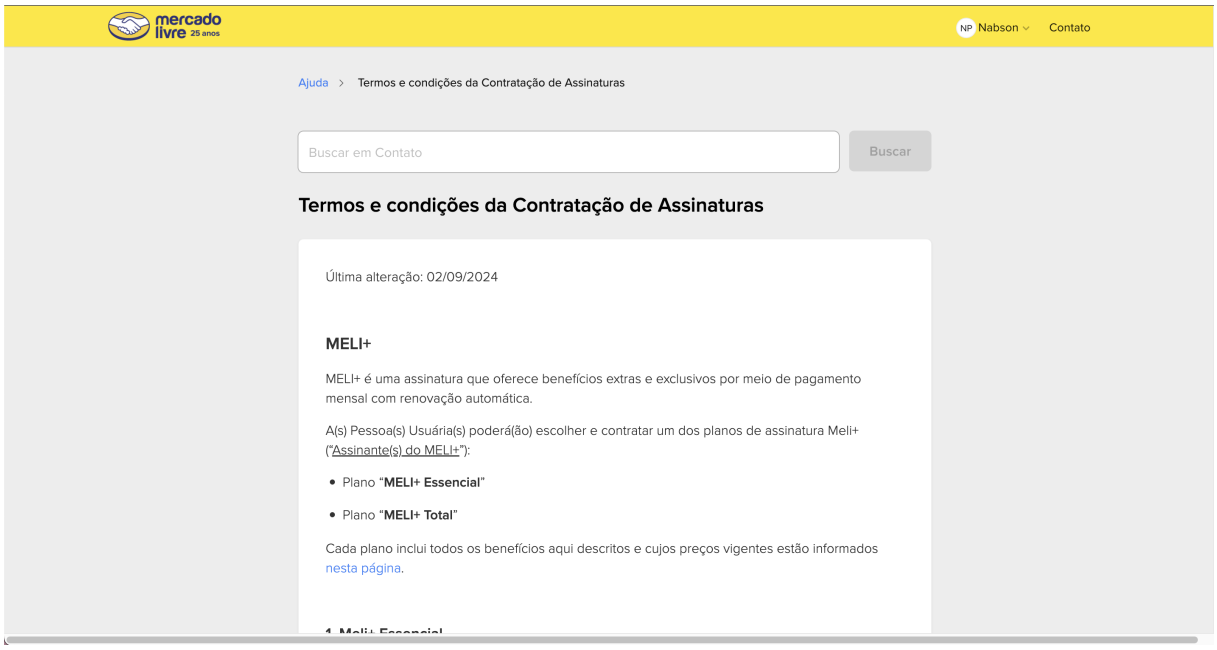

Figure 90 – Subscription Terms and Conditions Screen: The screen shows the terms and conditions required for the subscription.

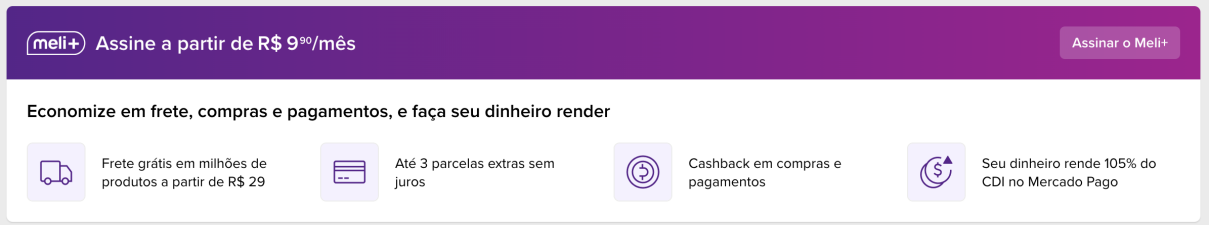

Figure 91 – Offer Component with Details: This component should be displayed on the homepage, showing detailed information about the offer.

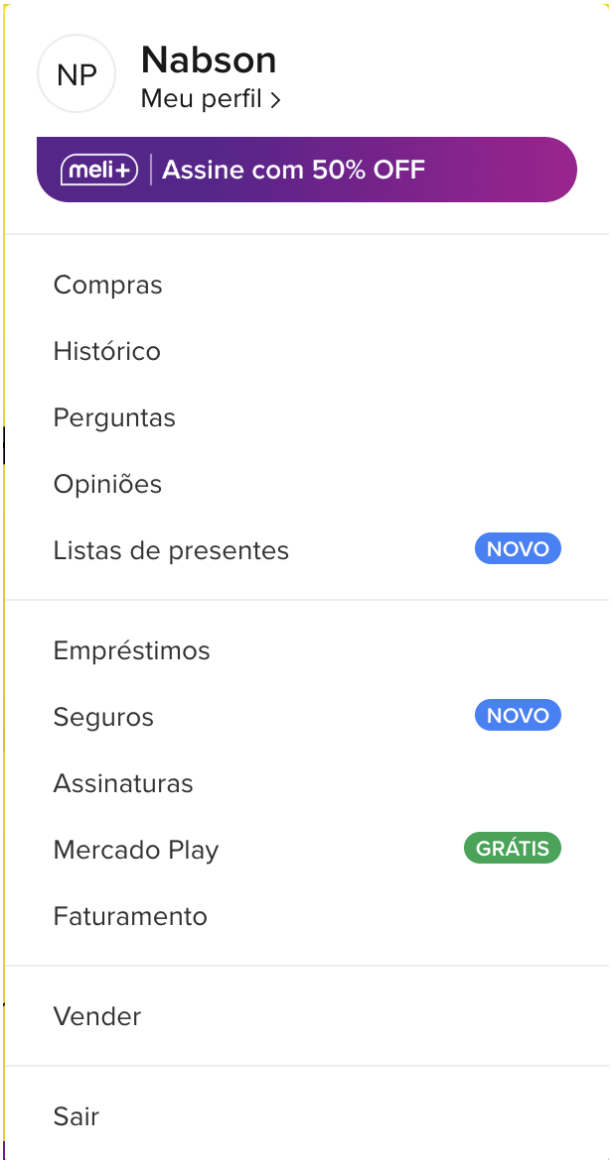

Figure 92 – Simple Offer Component: This component should be displayed in the user account menu, providing an overview of the offers.

Figure 93 – Offer Component for the Header: This component should be displayed in the header, highlighting the current offer.

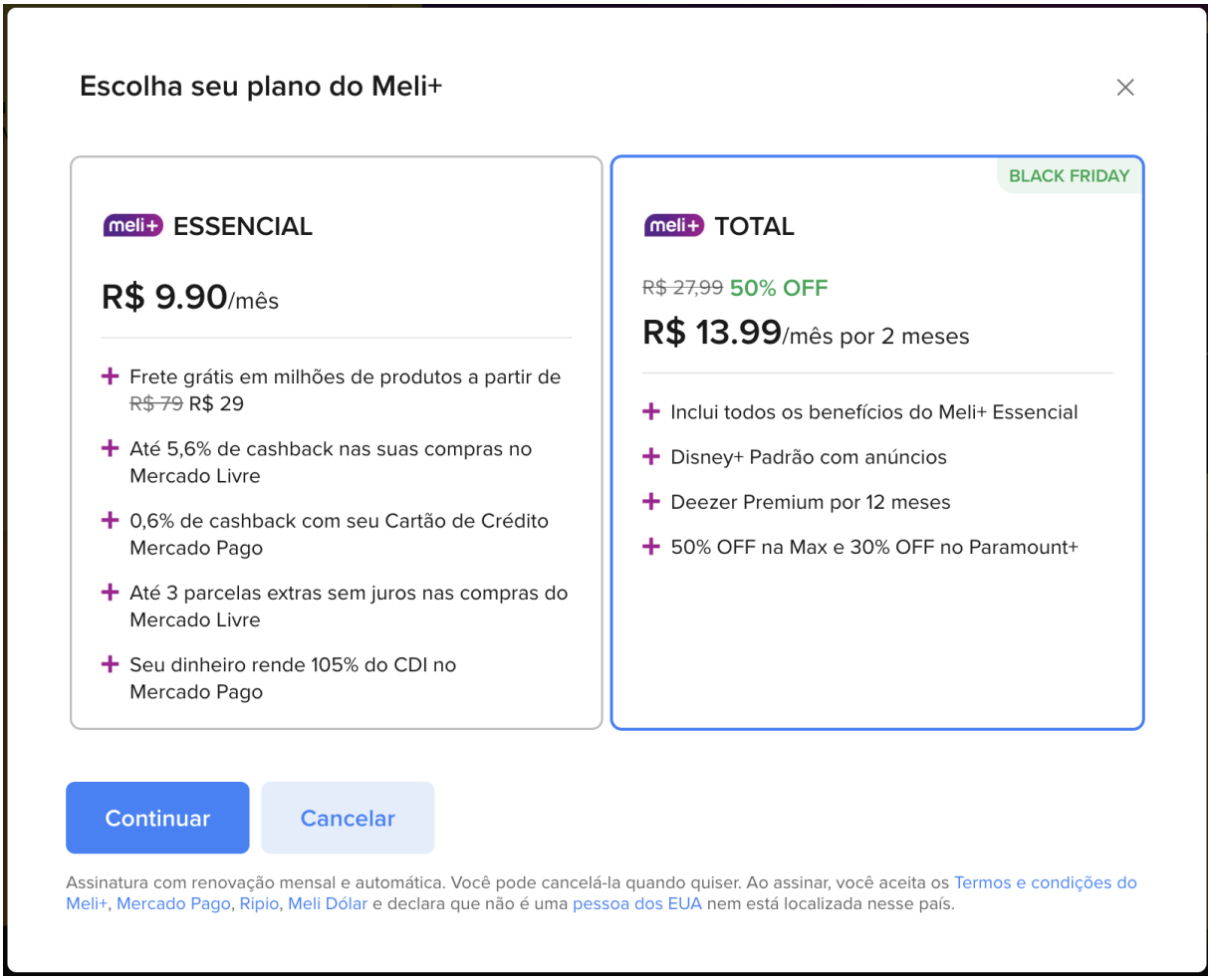

Figure 94 – Plan Selection Modal: A modal that allows the user to choose the desired subscription plan.

2. Mercado Pago has realized that it is important for the website to allow users to maintain one or more lists of favorite products. A user can have one or more lists and assign a name to each. The feature consists of the visual components presented in Figures 95, 96, 97, and 98. What architectural changes are necessary to develop the above components?

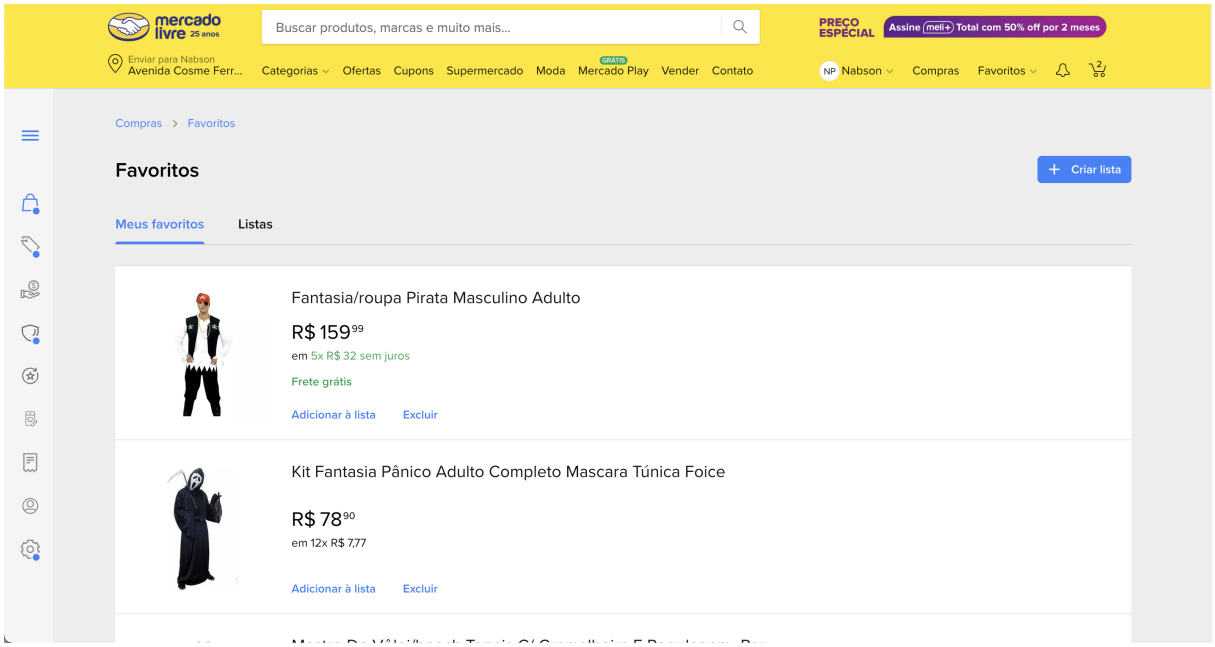

Figure 95 – Screen showing the list of all favorite items and the user's product lists: This screen displays all the favorite items and the product lists created by the user.

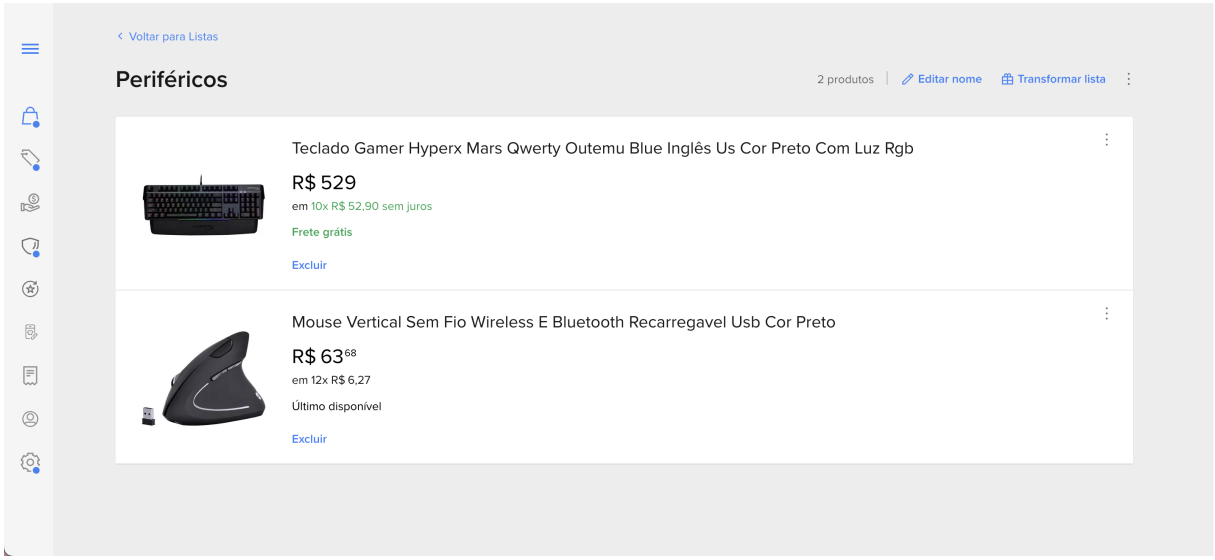

Figure 96 – Screen showing products belonging to a specific list: This screen presents the products that are part of a specific user-created list.

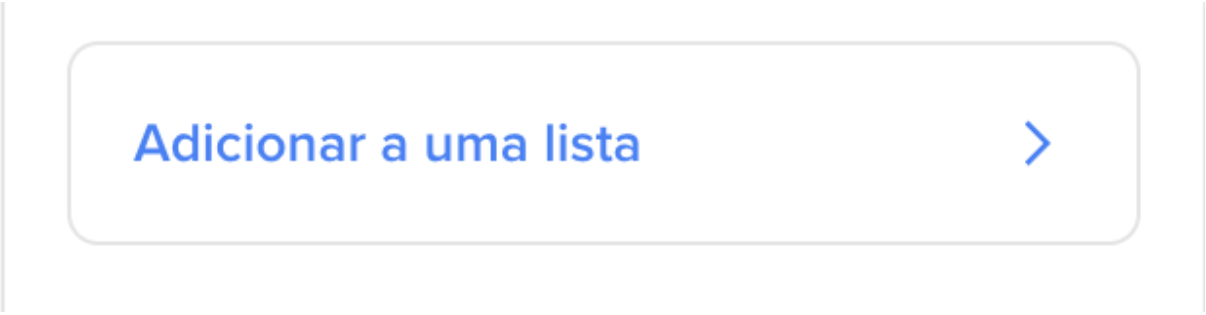

Figure 97 – Button to add a product to a list: This button opens a modal allowing the user to choose which list they want to add the product to or create a new list.

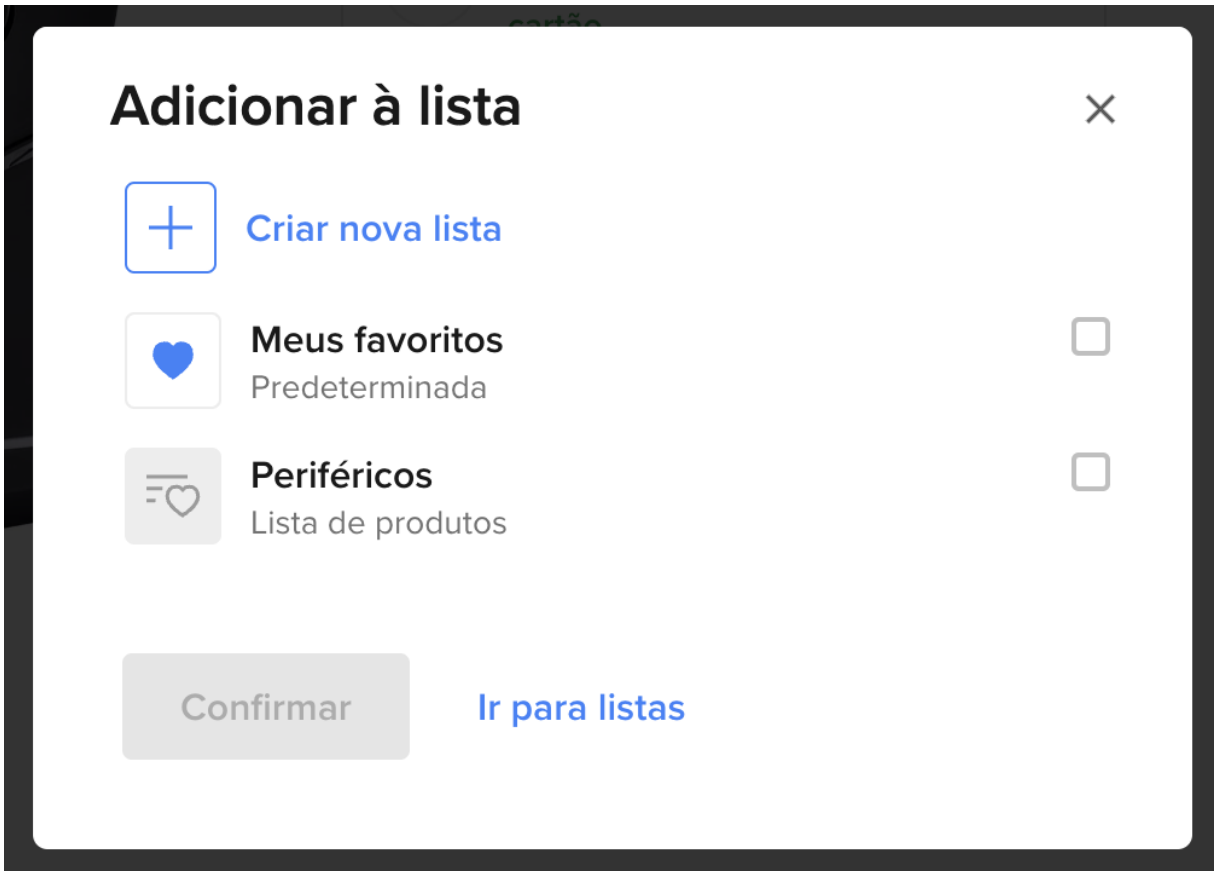

Figure 98 – Modal to create a new list: This modal allows the user to create a new product list.

3. Suppose that mfe-store is updated, and now the fragment displaying products the user is interested in needs a new parameter to be composed, the items in the cart, so it can offer related products. The mfe-store is deployed (sent to production), and the cart screen starts showing an error because mfe-purchase has not been updated to pass the parameter. What would you do to ensure the change in the

fragment does not generate an error on the cart screen, allowing the adjustment of the screen code to be done later, gradually and safely?

4. Suppose that every time you make a change to mfe-security, manual tests are required to ensure the application continues functioning. Note that these tests could be programmed as unit or integration tests. How would you ensure quality without the need for manual testing?

5. Suppose the fragment that shows products of interest to the user in mfe-store has been correctly deployed on the cart screen and has been working as expected for months without being altered. However, the fragment presents a bug whenever the cart has more than 10 items, and this list is passed as a parameter. When this happens, the cart screen shows an error, and the user cannot complete the purchase. What would you do to prevent the error in the fragment from affecting the entire screen?

6. Consider you are working on creating a new MFE. You need to create a new application-type component with single-spa and import all shared libraries, follow common coding patterns and structure, and register the new MFE in the root-config. You spent a lot of time setting up the new MFE before starting actual feature development. How could you streamline the creation of a new MFE next time?

7. Consider the following communication order between mfe-purchase, mfe-store, and mfe-users: (1) After successfully completing a purchase, mfe-purchase calls a function that receives a parameter from mfe-store to update the user's list to which the purchased product belongs (if in any list); (2) mfe-purchase also calls a function that receives a parameter from mfe-users to update the user's points based on the purchase value; and (3) the update of the user's total points causes mfe-users to call a function from mfe-store that informs the new total points so that new offers can be calculated for the user. Currently, there is difficulty in updating this flow, as a change can impact communication between all fragments. How would you increase the independence between MFEs while maintaining communication between them?

8. Initially, mfe-store was designed to support only physical products, such as electronics, clothing, and everyday items. Now, Mercado Livre wants to add a new type of product: services (such as consulting, private lessons, and technical services). The code for mfe-store was developed specifically for physical products, with rules involving freight calculation, stock, and handling, none of which apply to services, making development difficult. What would you do during the development of physical product sales to make mfe-store more generic and facilitate the implementation of new types of products?

C.2 Object 2 Questions

Consider that you are a Junior Developer who will start working at Nubank. You have just learned more about the architecture of the application, and you have been asked to analyze the architecture in terms of Micro Frontends (MFE):

1. Would you model the system using Micro Frontends differently? Would you keep the same MFEs, create new ones, or alter the existing ones? Also, consider the screens and fragments.

Consider that you are a Junior Developer who will start working at Nubank. Answer how you would solve each of the following tasks. Feel free to propose new MFEs, shared libraries, screens, and/or fragments, as well as alter the existing ones.

1. Nubank is launching a new investment method called NuCoin, a proprietary app currency that users accumulate by making purchases on their credit card. The feature consists of several screens and fragments, some of which are presented in Figures 99, 100, 101, and 102. What architectural changes are necessary for developing the components above?

Figure 99 – Main Screen of NuCoin.

Figure 100 – Screen showing NuCoin balance statement.

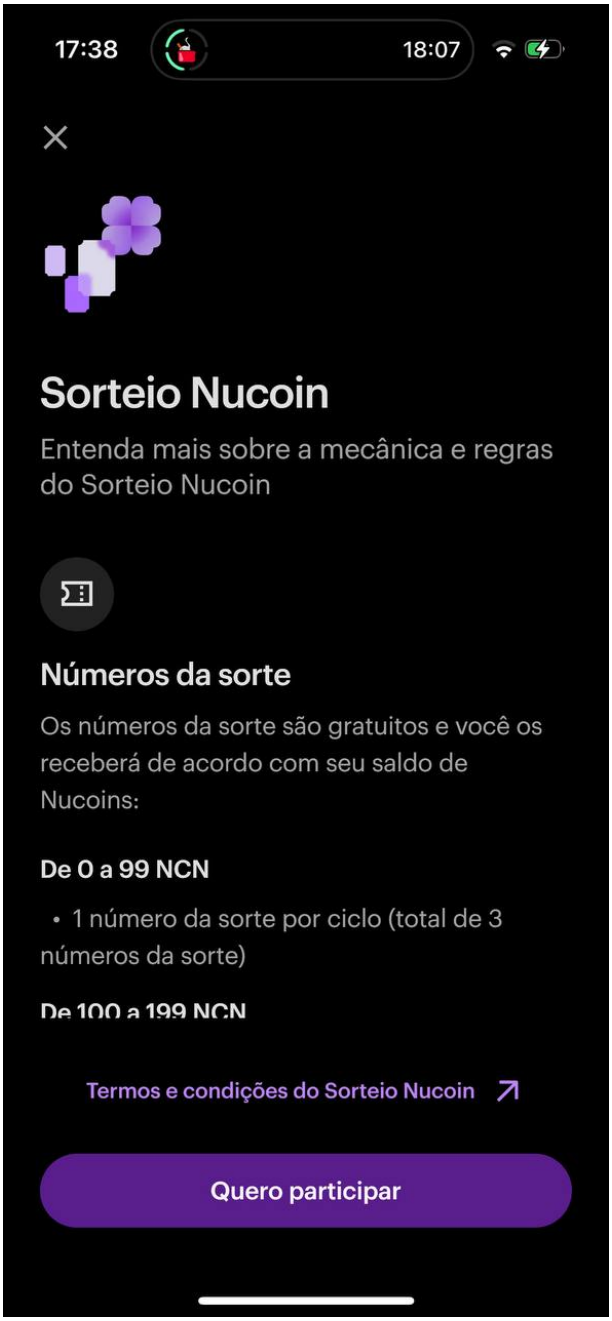

Figure 101 – Screen detailing raffles of NuCoin.

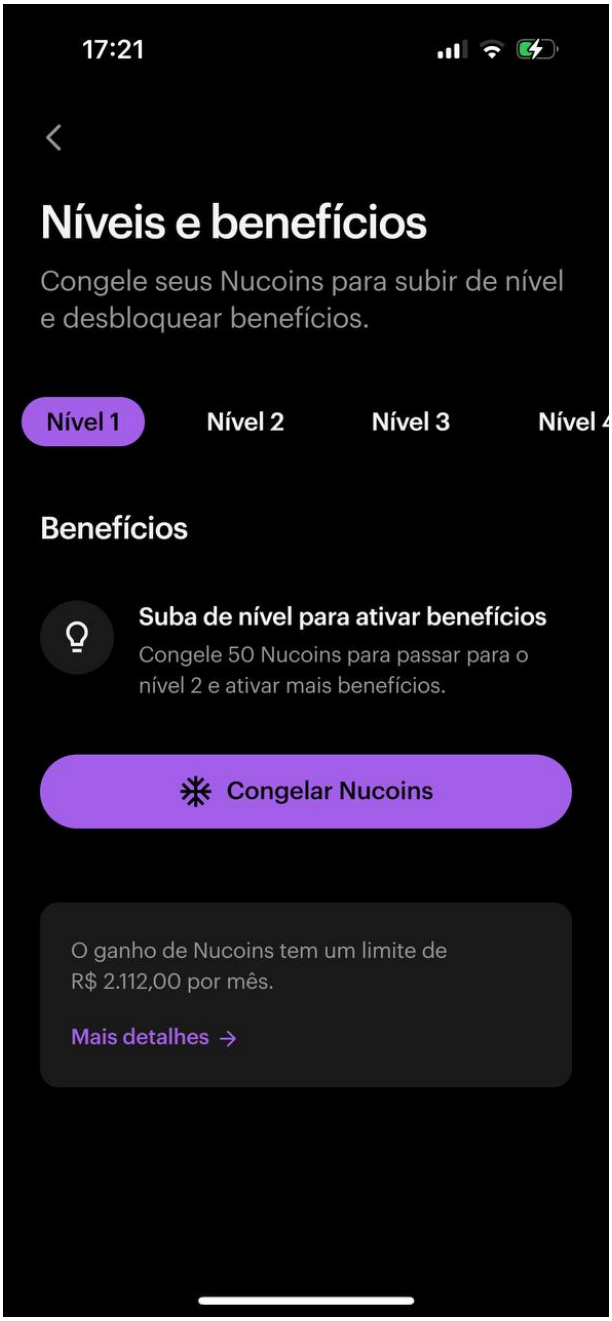

Figure 102 – Screen with benefits of NuCoin.

2. Nubank is launching a new feature called "Pass everything on credit", which allows sending Pix transfers and paying bills using the credit card. Additionally, cash purchases on the card can also be split into installments. This feature includes the visual components presented in Figures 103, 104, and 105. What architectural changes are necessary for developing the components above?

Figure 103 – Screen to choose what to charge to the credit.

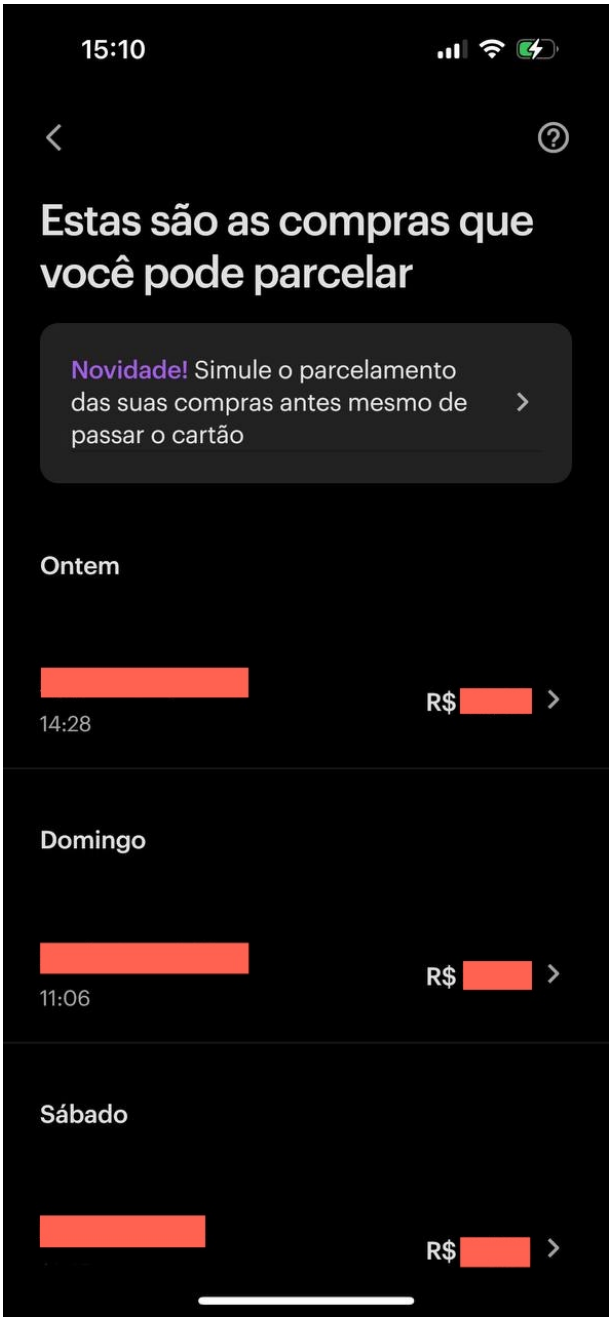

Figure 104 – Screen showing purchases that can be paid in installments.

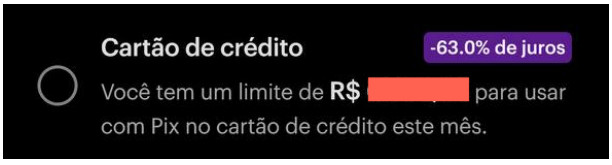

Figure 105 – Fragment with the option to parcel a Pix payment.

3. Suppose that mfe-loan is updated, and now the loan section fragment requires a new mandatory parameter, the user's current account balance. The mfe-loan

is deployed (sent to production), and the app's home screen, which imports this fragment in mfe-home, shows an error for not passing the new mandatory parameter to the fragment. What would you do to ensure that the deployment of mfe-loan does not cause an error on the home screen, allowing the screen code to be adjusted gradually and safely later?

4. Suppose that every time you make a change in mfe-cards, you need to access the server where the application is deployed and manually upload the code changes. Note that the deployment could be done using an automated sequence of commands. How would you ensure that the deployment of new code is not manual?

5. Suppose that the home screen implemented in mfe-home shows an error that is not related to any component implemented directly on the page, indicating that it might be caused by one of the screen fragments. What would you do to ensure that an error in a fragment does not make the entire screen unavailable?

6. Consider that you have completed the implementation of a new MFE, and now you need to deploy it and connect it with other MFEs in production. You encounter many difficulties in this process and realize that these are recurring problems. How could you speed up the deployment of a new MFE next time?

7. Consider the following communication order between 'mfe-digital-account', 'mfe-loan', and 'mfe-home': (1) After a successful Pix transfer, 'mfe-digital-account' calls a function that receives a parameter from 'mfe-home' to update the account balance shown on the home screen; (2) 'mfe-home' updates the account balance and changes the parameter passed to the loan section fragment of 'mfe-loan'; and (3) 'mfe-loan' then recalculates the loan offer based on the user's current balance. Currently, there is difficulty in updating this flow, as one change can impact communication between all the fragments. How would you increase the independence between the MFEs while maintaining communication between them?

8. Consider that mfe-investment is treated as the module that should offer various investment products. It started as a module exclusively for CDB investments, and now it has been adjusted to allow investments in Treasury Direct. Nubank

wants to implement a new investment model that operates very differently from CDB and Treasury Direct (it has specific screens and a different purchasing flow). This implementation will be problematic because the mfe-investment code is too specific to how CDB and Treasury Direct purchases work, and adding a new product won't be easy. What would you do during the development of CDB and Treasury Direct sales to make mfe-investment more generic and facilitate the implementation of new investment products?

D

Controlled Experiment Coding

This Appendix provides the complete open coding performed during the qualitative analysis on the students feedback presented in Chapter 5.

Open Coding on Students Feedback				
Question	ID	Answer	Quote	Code
Did you use or not use the anti-pattern catalog for MFE during the evaluation exercise? If so, how? If not, why did you not use it?	P1	I used it. I opened the catalog and read the described problems because, as they are anti-patterns, it means it is very likely that the problem I have is similar to one that has already occurred before, and then I just needed to read the solution to see if it was what I wanted to solve the initial problem.	I opened the catalog and read the described problems because, as they are anti-patterns, it means it is very likely that the problem I have is similar to one that has already occurred before	Identify common problems
			read the solution to see if it was what I wanted to solve the initial problem	Check if the solution is suitable
	P2	I used it to review certain concepts.	review certain concepts	Review MFE concepts
	P3	Yes, the catalog helped a lot because, besides explaining the anti-patterns, it showed a practical example and then the solution to the problem, which greatly helped in identifying how I would solve the proposed problems.	it showed a practical example and then the solution to the problem, which greatly helped in identifying how I would solve the proposed problems.	Identify problems by examples
	P4	I used it; with it, it was possible to have a description of common problems in MFE implementation and their solutions.	it was possible to have a description of common problems in MFE implementation and their solutions	Identify common problems
	P5	Yes, I was looking for problems similar to those presented in the exercise, and after that, I saw how they were solved. Of course, even with similar problems, some solutions proved to be very incompatible.	I was looking for problems similar to those presented in the exercise	Identify problems similar to the presented
			even with similar problems, some solutions proved to be very incompatible	Identify problems by solutions
	P6	Yes, I used it because it helped me understand the problems and know how to solve them in the best possible way.	it helped me understand the problems and know how to solve them in the best possible way	Understand common challenges
	P7	Yes, confirming what I learned in class and knowing how to better segment problems found in MFE. It allowed me to address these problems and made it easier to make decisions.	confirming what I learned in class	Review MFE concepts
			how to better segment problems found in MFE. It allowed me to address these problems and made it easier to make decisions.	Guide architectural decisions
	P8	Yes, as an aid to understanding some issues and developing their solutions.	understanding some issues	Understand common challenges
			developing their solutions	Solving common problems
	P9	Yes, I used it to solve scenarios that can occur in the development of an MFE-oriented architecture.	I used it to solve scenarios that can occur in the development of an MFE	Solving common problems
	P10	Yes, I used it to identify the problems and the solutions to the respective questions.	identify the problems and the solutions to the respective questions	Identify problems and solutions
	P11	I used it a lot, seeking to understand the different types of anti-patterns and their possible solutions. It makes problems that can be prevented much more visible before a product is 80%-100% complete, avoiding rework.	seeking to understand the different types of anti-patterns and their possible solutions	Understand common challenges
			It makes problems that can be prevented much more visible before a product is 80%-100% complete, avoiding rework.	Prevent problems
	P12	Yes, checking solutions for the problems I encountered during the exercise and seeing examples and descriptions helped me understand the problem in some questions.	checking solutions for the problems I encountered during the exercise	Identify problems and solutions
			seeing examples and descriptions helped me understand the problem in some questions	Understand problems by examples
	P13	Yes, I used it a lot to understand if the situation proposed by the question resembled any example described by the anti-patterns. Besides analyzing my possible architectural decisions, avoiding falling into a possible listed case.	understand if the situation proposed by the question resembled any example described by the anti-patterns.	Identify problems by examples
			possible architectural decisions, avoiding falling into a possible listed case	Guide architectural decisions
	P14	Yes, I used it; it served as a reminder.	it served as a reminder	Review MFE concepts
	P15	Yes, analyzing the addressed problems and the examples cited in the catalog to associate them with the scenario proposed in the exercise.	analyzing the addressed problems and the examples cited in the catalog to associate them with the scenario	Identify problems by examples
	P16	Yes, it helped me a lot when answering the questions because I could identify the problems that appeared as anti-patterns.	I could identify the problems that appeared as anti-patterns	Identify problems and solutions
	P17	I used it. It was very useful to see how the presented problems fit into the anti-patterns. Seeing the anti-patterns in practice increased my knowledge about the subject.	useful to see how the presented problems fit into the anti-patterns	Identify common problems
			Seeing the anti-patterns in practice increased my knowledge about the subject	Learn based on practical problems
	P18	Yes, I looked at the descriptions on the homepage. I checked if there was something similar to what I wanted, clicked to confirm through the example, and used the solution if that was the case.	I checked if there was something similar to what I wanted	Identify problems similar to the presented
			clicked to confirm through the example	Identify problems by examples
	P19	Yes, to check if any of the examples were in the catalog, considering that the catalog contains the most common anti-patterns, I believe it is useful to consult it to avoid errors that have already been documented.	to check if any of the examples were in the catalog	Identify problems and solutions
			avoid errors that have already been documented.	Prevent problems
	P20	I used it as a reference to see if someone had already reported the same problem and how they solved it.	reference to see if someone had already reported the same problem and how they solved it	Identify common problems
	P21	Yes, it helped me identify problems and demonstrate the best solutions.	identify problems and demonstrate the best solutions.	Identify problems and solutions
	P22	Yes, I used it to identify the proposed problems in the questions.	identify the proposed problems in the questions	Identify problems and solutions
	P23	Yes, I used it a lot. When faced with questions about changing the architecture, I often found myself consulting the catalog.	When faced with questions about changing the architecture, I often found myself consulting the catalog	Guide architectural decisions
Did you notice benefits or challenges when using the anti-pattern catalog for MFE? Ease or difficulties of use? If so, what were they?	P1	Benefits and conveniences. Since the catalog has several short descriptions of problems that have occurred before, it is very productive to check if the problem I face is in the catalog to see the solution other developers had, instead of thinking of one on my own.	catalog has several short descriptions of problems that have occurred before, it is very productive to check if the problem I face is in the catalog to see the solution other developers had	Identify common problems
	P3	Using the catalog helped standardize MFE creation, as I could see possible errors that could occur when creating an MFE and avoid them.	helped standardize MFE creation, as I could see possible errors that could occur when creating an MFE and avoid them	Guide architectural decisions
	P4	I had difficulties understanding the description of each anti-pattern; I had to open all of them to see the description. There could be a summary in each card of the catalogs on the main screens.	difficulties understanding the description of each anti-pattern; I had to open all of them to see the description. There could be a summary in each card of the catalogs on the main screens	Consult several anti-patterns at once
	P5	The benefits were the solutions shown in each concept, which are straightforward.	solutions shown in each concept, which are straightforward	Check straightforward solutions
	P6	Yes. As benefits, I noticed it is easy to understand and has great examples of occurrences that help us understand the correct way we can proceed.	easy to understand and has great examples of occurrences that help us understand the correct way we can proceed	Understand problems by examples

	P7	Yes, conveniences because it is easy to visualize, especially with the use of categories. However, one thing is that I believe some situations are not very clear. Maybe there is no anti-pattern that explicitly shows the problem, but their combination can clarify what needs to be done.	it is easy to visualize, especially with the use of categories	Consult by categories
			I believe some situations are not very clear. Maybe there is no anti-pattern that explicitly shows the problem, but their combination can clarify what needs to be done	Linking anti-patterns
	P8	The benefits I noticed were that it was an easy, quick, and accessible way to obtain information. A challenge was that some examples were not very clear, and I had to reread them several times.	an easy, quick, and accessible way to obtain information	Review MFE concepts
			some examples were not very clear, and I had to reread them several times	Understand problems by examples
	P9	It is very intuitive, easy to navigate, with simple and direct examples followed by their respective solutions. It is a great tool.		
	P10	Conveniences. The anti-pattern catalog is very intuitive. That way, I had no difficulties locating anything. Moreover, its division into concept, example, and solution greatly facilitates understanding.	its division into concept, example, and solution greatly facilitates understanding	Understand by structured anti-patterns
	P11	Benefits and conveniences. Interaction is very easy, and having the material with the set definition, example, and solution is much more intuitive and positive.		
	P12	Benefits because with the descriptions, I was able to identify the problems during the exercises, as well as possible solutions, and it was very easy to use, very intuitive.	with the descriptions, I was able to identify the problems during the exercises, as well as possible solutions	Identify problems and solutions
	P13	I believe the biggest challenge is the amount of text present. Perhaps using more images for the examples would be more effective.	the biggest challenge is the amount of text present. Perhaps using more images for the examples would be more effective	Understand by image
	P14	The only negative point I noticed was that the anti-pattern tags were not very useful, at least for the exercise we did.		
	P15	Benefits due to the clarity presented in the catalog and convenience because it was possible to associate the exercise context with the contexts, descriptions, and solutions shown in the catalog.	possible to associate the exercise context with the contexts, descriptions, and solutions shown in the catalog	Identify problems similar to the presented
	P16	It was very easy to use; the explanations were good, especially those with images.	the explanations were good, especially those with images	Understand by image
	P17	I found it easy to use.		
	P18	Yes, benefits, the site is very direct in what it proposes.		
	P19	I noticed benefits because, for me, it is more advantageous to avoid an error that is already documented than to end up reproducing an error that is already documented. The main conveniences I observed were: a practical example of the anti-patterns, their solution, and the categorization.	more advantageous to avoid an error that is already documented than to end up reproducing an error that is already documented	Prevent problems
			a practical example of the anti-patterns, their solution	Understand problems by examples
	P20	Several benefits due to the simplicity of the page and its objectivity. However, navigation bothered me a bit. If I perform a filtered search and return to the previous page, I go to where all are, not where the filters are. Another thing that could help is that, when opening an anti-pattern, the page could show others from the same topic or similar ones at the bottom.	when opening an anti-pattern, the page could show others from the same topic or similar ones at the bottom.	Linking anti-patterns
	P21	Ease of use.		
	P22	Benefits and conveniences. It is simple to consult, and the examples helped me identify the problem	and the examples helped me identify the problem	Identify problems by examples
	P23	Initially, it was in English, and that made it a bit difficult, but I found that the button filters helped a lot. I practically did not use the search input at the top of the screen.	I found that the button filters helped a lot	Consult by categories
Describe here if and how the anti-pattern catalog for MFE helped or hindered your learning about Micro Frontends.	P1	It helped because various problems and solutions are documented, thus contributing to sharing solutions for recurring problems instead of leaving the developer to think of a solution on their own.	various problems and solutions are documented, thus contributing to sharing solutions for recurring problems	Identify common problems
	P2	It provided a simpler and faster way to access information about the anti-patterns.	simpler and faster way to access information about the anti-patterns	Review MFE concepts
	P3	The catalog helped me learn about MFE because, in it, I saw possible errors that can occur in MFE creation and how to solve these problems.	learn about MFE because, in it, I saw possible errors that can occur in MFE creation and how to solve these problems	Learn based on practical problems
	P4	It helped by centralizing content about MFEs. However, the catalog could include summaries and slides presented in the classes, as I believe the purpose of the catalog would be to centralize MFE content. This way, it would be friendlier to new users.	helped by centralizing content about MFEs. However, the catalog could include summaries and slides presented in the classes	Review MFE concepts
	P5	A better description of the problems and solutions in the catalog helped me a lot to draw a parallel with the problems proposed in the activity.	better description of the problems and solutions in the catalog helped me a lot to draw a parallel with the problems proposed in the activity	Identify problems by solutions
	P6	Yes, it helped. I confess that it was even a bit easier to understand the anti-pattern catalog than the design patterns I learned in Assignment 2. But understanding the subject became easier, and I was able to learn and will try to use these patterns and anti-patterns in my future projects.	easier to understand the anti-pattern catalog than the design patterns I learned in Assignment 2	Understand common challenges
			will try to use these patterns and anti-patterns in my future projects	Guide architectural decisions
	P7	It helped describe better the problems encountered and how to solve them. I missed more examples and visual elements, which help to remember each anti-pattern.	describe better the problems encountered and how to solve them	Identify problems and solutions
			missed more examples and visual elements, which help to remember each anti-pattern	Understand by image
	P8	It helped me understand in an organized, clear, and effective way. By reading the problem, I could easily choose which anti-pattern to use.	understand in an organized, clear, and effective way	Understand by structured anti-patterns
			reading the problem, I could easily choose which anti-pattern to use	Identify problems and solutions
	P9	It helped me because it helped me avoid common mistakes that I would definitely have made if I did not know about them.	avoid common mistakes that I would definitely have made if I did not know about them	Prevent problems
	P10	With a good division in the catalog, I was able to understand, given that the way it is divided allows me to associate and identify the anti-patterns more easily.	given that the way it is divided allows me to associate and identify the anti-patterns more easily.	Understand by structured anti-patterns
	P11	It helped in problem abstraction. It became much easier to visualize and solve possible problems.	It became much easier to visualize and solve possible problems	Identify common problems
	P12	It helped, especially with practical examples and solutions for each type of anti-pattern.	especially with practical examples and solutions for each type of anti-pattern	Understand problems by examples
	P13	It helped a lot because, with it, I could understand real situations and challenges faced during architectural decisions of a system with MFEs.	I could understand real situations and challenges faced during architectural decisions of a system with MFEs	Understand common challenges
	P14	It helped, but I believe the classes were more important for my understanding; the catalog served as a review.	the catalog served as a review	Review MFE concepts

	P16	The catalog helped because it was very explanatory regarding common errors in MFE architecture.		
	P17	It helped a lot. It is good to see what we should not do when developing software.	good to see what we should not do when developing software	Prevent problems
	P18	After reading a problem description, checking if that problem is caused by a design issue helps a lot because then I can already see if it can be solved in a way that has already been experimented with by someone more experienced than me.	After reading a problem description, checking if that problem is caused by a design issue helps a lot because then I can already see if it can be solved in a way that has already been experimented	Identify problems and solutions
	P19	The catalog helped me identify common patterns that could be harmful to me in the future, potentially causing an error in a project's architecture, as it describes, exemplifies, and suggests a solution for these anti-patterns.	identify common patterns that could be harmful to me in the future, potentially causing an error in a project's architecture	Prevent problems
	P20	It helped especially when contextualizing myself with the problem. Many times, I had difficulty identifying problems because I did not know all the problems, whether it was indeed a problem, and if it was, how would I solve it? All these questions were greatly minimized by using the catalog. The catalog helped me diagnose problems I did not previously know.	I had difficulty identifying problems because I did not know all the problems, whether it was indeed a problem, and if it was, how would I solve it? All these questions were greatly minimized by using the catalog.	Identify problems and solutions
			The catalog helped me diagnose problems I did not previously know.	Identify common problems
	P21	MFE catalog helped me remember some concepts I did not recall, in addition to being very clear in its explanations and using examples.	helped me remember some concepts I did not recall	Review MFE concepts
			being very clear in its explanations and using examples	Understand problems by examples
	P22	It helped me due to the ease of accessing the anti-patterns that I probably would not have if I were searching for them on websites.	ease of accessing the anti-patterns that I probably would not have if I were searching for them on websites	Identify common problems
	P23	The catalog helped a lot when looking for categories of anti-patterns. It was a bit inconvenient to translate from English to Portuguese.	The catalog helped a lot when looking for categories of anti-patterns	Consult by categories
Do you have any suggestions for improving the anti-pattern catalog for MFEs? If you encountered any difficulty understanding the definition of an anti-pattern (considering the fields for problem, solution, and example), please indicate which anti-pattern it was and describe what was unclear.	P3	Availability of translation into other languages increases accessibility for people who do not master English very well.		
	P4	As I mentioned in the previous response, it would be possible to add summaries and slides presented in the classes to the catalog, as I believe the purpose of the catalog is to centralize MFE-related content, making it more user-friendly for new users.	add summaries and slides presented in the classes to the catalog	Review MFE concepts
	P5	If there is at least one more scenario as an application example, that would be ideal.	at least one more scenario as an application example, that would be ideal	Understand problems by examples
	P6	More figures and, if possible, code examples in any language.	More figures	Understand by image
			code examples in any language	Understand problems by examples
	P7	I suggest adding more visual elements and more examples for each anti-pattern, as well as a description of the "domain" of the category	adding more visual elements	Understand by image
			more examples for each anti-pattern	Understand problems by examples
	P8	I did not understand the Micro Frontend Knot very well.		
	P9	I might have missed it, but I could not find the anti-pattern that addresses error handling. If it is indeed not present in the catalog, I believe its inclusion would be a good addition. However, I might have just failed to locate it.		
	P10	Only adding of more anti-patterns for other types of systems.		
	P13	I believe using more visual resources could be very helpful to avoid the overwhelming amount of text in each listed anti-pattern.	using more visual resources could be very helpful to avoid the overwhelming amount of text	Understand by image
	P14	Within the definitions of the anti-patterns, I did not find any difficulty in understanding.		
	P16	It is a good catalog as it is, but the use of images in some explanations helped more than just text.	but the use of images in some explanations helped more than just text	Understand by image
	P19	I believe a dictionary of key terms related to architecture in the catalog could facilitate understanding, but it was easy to use.	dictionary of key terms related to architecture in the catalog could facilitate understanding	Identify problems by keywords
	P21	The catalog is perfect, no improvements are needed.		
	P22	Perhaps more examples and bibliographic references where it is possible to delve deeper into the anti-patterns.	more examples	Understand problems by examples
			bibliographic references where it is possible to delve deeper into the anti-patterns	Review MFE concepts

E

MLR Publications

This Appendix provides the complete list of publications retrieved during the search process from the MLR presented in Chapter 6.

ID	Base	Acesso em	Título	Link
P1	Google	28/11/2024 06:56:00	Top 10 Micro Frontend Anti-Patterns	https://dev.to/florianrappl/top-10-micro-frontend-anti-patterns-3809
P2	Google	28/11/2024 06:57:00	Microfrontends Anti-Patterns: Seven Years in the Trenches	https://www.infoq.com/presentations/microfrontend-antipattern/
P3	Google	28/11/2024 06:58:00	Entendendo os anti-patterns na arquitetura de Micro-Frontends	https://thedevconf.s3.sa-east-1.amazonaws.com/presentations/TDC2023BUS/webfrontend/JUR-3676_2023-09-15T105545_MFEs+anti-pattern.pdf
P4	Google	28/11/2024 06:58:00	Micro Frontends Anti-Patterns	https://www.geeksforgeeks.org/micro-frontends-anti-patterns/
P5	Google	28/11/2024 06:58:00	Frontend Nation 2024: Luca Mezzalira - Micro-Frontends Anti-Patterns	https://www.youtube.com/watch?v=lCiQb1DC6t4&ab_channel=VueSchool
P6	Google	28/11/2024 06:59:00	10 Deadly Sins of Micro Frontend Anti-Patterns That Could Derail Your Success	https://medium.com/@cannon_circuit/10-deadly-sins-of-micro-frontend-anti-patterns-that-could-derail-your-success-1186cc631aed
P7	Google	28/11/2024 06:59:00	Micro-Frontends anti-patterns by Luca Mezzalira (#GSAS24)	https://www.youtube.com/watch?v=3jygY3LGTKc&ab_channel=Apiumhub
P8	Google	28/11/2024 07:00:00	4 Micro-Frontend Anti-Patterns	https://javascript.plainenglish.io/four-micro-frontend-anti-patterns-58aaa9fe19d5
P9	Google	28/11/2024 07:00:00	How Micro-Frontends are reshaping Modern Web Architecture	https://lucamezzalira.medium.com/how-micro-frontends-are-reshaping-modern-web-architecture-0259ce7dfb7f
P10	Google	28/11/2024 07:00:00	Micro frontends anti-patterns – Luca Mezzalira	https://www.youtube.com/watch?v=T3NINYCP9gg&ab_channel=BECFinancialTechnologies
P11	Google	28/11/2024 07:01:00	Choosing the right architecture for your business	https://www.mulesoft.com/sem/lp/whitepaper/api/top-microservices-patterns?d=7013y0000026fABAAY&nc=7013y0000026fi3AAA&utm_content=7013y0000026fABAAY&utm_source=google&utm_medium=paid_search&utm_campaign=21250958287&utm_adgroup=161422060683&utm_term=microservices%20architecture%20pattern&utm_matchtype=p&gad_source=1&gclid=CjwKCAiAxqC6BhBcEiwAlXp454sleZjPlPcuJmP6_QAji9GvXGL64J4PHzZwdVrQ2l4EhVrPGv3fFRoCA5kQAvD_BwE&gclsrc=aw.ds
P12	Google	28/11/2024 07:02:00	Micro-Frontends anti-patterns by Luca Mezzalira (#GSAS24)	https://www.linkedin.com/posts/lucamezzalira_micro-frontends-anti-patterns-by-luca-mezzalira-activity-7265022027307094018-1zvt/
P13	Google	28/11/2024 07:03:00	Micro-frontends and composable frontend architectures	https://microfrontend.dev/
P14	Google	28/11/2024 07:03:00	Micro-Frontends anti-patterns by Luca Mezzalira (#GSAS24)	https://www.wearedevelopers.com/en/videos/420/micro-frontends-anti-patterns-739926065
P15	Google	28/11/2024 07:03:00	-	https://www.threads.net/
P16	Google	28/11/2024 07:04:00	Building Micro Frontends with React 18: Develop and deploy scalable applications using micro frontend strategies	https://www.amazon.com/Building-Micro-Frontends-React-microfrontend-ebook/dp/B0BQC16W6B
P17	Google	28/11/2024 07:04:00	Multi-Framework and -Version Micro Frontends with Module Federation: Your 4 Steps Guide	https://www.angulararchitects.io/blog/multi-framework-and-version-micro-frontends-with-module-federation-your-4-steps-guide/
P18	Google	28/11/2024 07:04:00	The Micro-Frontends future	https://lucamezzalira.com/category/micro-frontends/

P19	Google	28/11/2024 07:05:00	Micro Frontends Conference	https://hasgeek.com/jsfoo/microfrontends-conf/sub/micro-frontends-anti-patterns-Y7WWDT2M1zAbz9DFb79TxX
P20	Google	28/11/2024 07:05:00	Microfrontends Anti-Patterns: Seven Years in the Trenches	https://www.youtube.com/watch?v=n1XSeiLhBtE&ab_channel=InfoQ
P21	Google	28/11/2024 07:06:00	The Strengths and Benefits of Micro Frontends	https://www.toptal.com/front-end/micro-frontends-strengths-benefits
P22	Google	28/11/2024 07:06:00	Micro-frontends anti-patterns by Luca Mezzalira (#GSAS24)	https://www.wearedevelopers.com/en/videos/299/micro-frontends-anti-patterns
P23	Google	28/11/2024 07:07:00	Top 10 Microservice Anti-Patterns	https://blog.bitsrc.io/10-microservice-anti-patterns-278bcb7f385d
P24	Google	28/11/2024 07:07:00	Everything You Need to Know About Micro Frontends	https://newsletter.systemdesign.one/p/micro-frontends
P25	Google	28/11/2024 07:08:00	Microfrontends - Decoupling Frontends	https://aymanace2049.hashnode.dev/microfrontends-decoupling-frontends
P26	Google	28/11/2024 07:08	Luca Mezzalira (@amazonwebservices) - "Micro-frontends anti-patterns" at the Code Europe 2023	https://www.youtube.com/watch?v=R2Ydq5E_8ts&ab_channel=CodeEurope
P27	Google	28/11/2024 07:09	Luca Mezzalira - Micro-frontends anti-patterns |redev 2022	https://www.linkedin.com/posts/lucamezzalira_luca-mezzalira-micro-frontends-anti-patterns-activity-7006541261352402944-Z4xk/
P28	Google	28/11/2024 07:09	Micro-Frontend	https://awesome-architecture.com/micro-frontend/
P29	Google	28/11/2024 07:10	Micro-Frontend Mindmaps	https://github.com/santoshshinde2012/micro-frontends-mindmaps
P30	Google	28/11/2024 07:10	Chapter 4. Discovering Micro-Frontend Architectures	https://www.oreilly.com/library/view/building-micro-frontends/9781492082989/ch04.html
P31	Google	28/11/2024 07:11	Micro Frontends	https://www.infoq.com/micro-frontends/
P32	Google	28/11/2024 07:11	Micro-Frontends anti-patterns	https://hasgeek.com/jsfoo/microfrontends-conf/schedule/micro-frontends-anti-patterns-CzzTos6obdNKvoZGwtY3Vo
P33	Google	28/11/2024 07:12	Motivations, Benefits, and Issues for Adopting Micro-Frontends: A Multivocal Literature Review	https://arxiv.org/pdf/2007.00293
P34	Google	28/11/2024 07:12	Micro-frontends - reversing the anti-pattern!	https://2019.nidevconf.com/sessions/chriskitson/
P35	Google	28/11/2024 07:12	The Micro-Frontends future	https://lucamezzalira.com/2022/03/01/the-micro-frontends-future/
P36	Google	28/11/2024 07:13	Assessing the feasibility of Micro frontend architecture in native mobile app development	https://dl.acm.org/doi/10.1145/3691620.3695313
P37	Google	28/11/2024 07:13	Micro-Frontend - What & Why?	https://dev.to/dwarvesf/micro-frontend-what-why-ike
P38	Google	28/11/2024 07:13:00	Rules of Micro-Frontends	https://www.infoxicator.com/rules-of-micro-frontends
P39	Google	28/11/2024 07:13	Micro-Frontends anti-patterns - Luca Mezzalira, AWS | Craft Conference 2022	https://www.youtube.com/watch?v=EvD-gFX6kN0&ab_channel=CraftHubEvents
P40	Google	28/11/2024 07:14:00	Micro frontends numa aplicação de précontabilidade	https://recipp.ipp.pt/bitstream/10400.22/24267/1/Tese_5137_v2.pdf
P41	Google	28/11/2024 07:15:00	Understanding and implementing microfrontends on AWS AWS Prescriptive Guidance	https://docs.aws.amazon.com/pdfs/prescriptive-guidance/latest/micro-frontends-aws/micro-frontends-aws.pdf
P42	Google	28/11/2024 07:15	Micro-frontends: anti-patterns | Luca Mezzalira | EnterpriseNG 2021	https://www.youtube.com/watch?v=gfKTFqjsT1M&ab_channel=ng-conf

P43	Google	28/11/2024 07:16	Top 10 Microservices Anti-Patterns	https://app.daily.dev/posts/top-10-microservices-anti-patterns-lqjo4scgy
P44	Google	28/11/2024 07:16	Luca Mezzalira – Micro frontends anti patterns | Øredev 2022	https://www.youtube.com/watch?v=DaxJuB8AYqM&ab_channel=%C3%98redevConference
P45	Google	28/11/2024 07:17:00	Post	https://x.com/lucamezzalira/status/1859256584427225365
P46	Google	28/11/2024 07:17	Microfrontend Deep Dive	https://openmfe.org/development/microfrontend-deepdive/
P47	Google	28/11/2024 07:17:00	Post	https://x.com/lucamezzalira/status/1846075769140785268
P48	Google	28/11/2024 07:19:00	Micro Frontends: the Evolution of Frontend Architecture	https://www.youtube.com/watch?v=W4biNjfmvvI&ab_channel=InfoQ
P49	Google	28/11/2024 07:19:00	Blog	https://smapiot.com/blog/
P50	Google	28/11/2024 07:20:00	Motivations, Benefits, and Issues for Adopting Micro-Frontends: A Multivocal Literature Review	Motivations, Benefits, and Issues for Adopting Micro-Frontends: A Multivocal Literature Review
P51	Google	28/11/2024 07:20:00	Palestrante e/ou Coordenador Aquele que faz história no TDC	https://thedevconf.com/palestrante/pedro-henrique-oliveira
P52	Google	28/11/2024 07:21:00	MICRO FRONTENDS: A NEW PARADIGM FOR SCALABLE ANGULAR APPLICATIONS Authors	https://iaeme-library.com/index.php/IJCET/article/view/IJCET_15_05_041
P53	Google	28/11/2024 07:21:00	Micro Frontend-Based Development: Concepts, Motivations, Implementation Principles, and an Experience Report	https://www.scitepress.org/Papers/2024/126273/126273.pdf
P54	Google	28/11/2024 07:21:00	Micro Frontend	https://fizalihsan.github.io/technology/microfrontend.html
P55	Google	28/11/2024 07:22:00	TechLead Journal	https://techleadjournal.dev/page/16/
P56	Google	28/11/2024 07:22:00	Luca Mezzalira: Micro-frontends Anti-patterns	https://www.oreilly.com/library/view/software-architecture-superstream/0636920689546/video338478.html
P57	Google	28/11/2024 07:22:00	Post	https://x.com/lucamezzalira/status/1845413811659932005
P58	Google	28/11/2024 07:23:00	Compositional Qualities of Microfrontends: The LdoD Archive	https://fenix.tecnico.ulisboa.pt/downloadFile/281870113706102/49372-joao-raimundo.pdf
P59	Google	28/11/2024 07:24:00	Micro-Frontends Pattern: Revolutionizing Frontend Development	https://softwarepatternslexicon.com/patterns-js/5/6/1/
P60	Google	28/11/2024 07:25:00	Architecture: Micro frontends	https://andrepolischuk.com/architecture-micro-frontends/
P61	Google	28/11/2024 07:26:00	Motivations, benefits, and issues for adopting Micro-Frontends: A Multivocal Literature Review	https://www.sciencedirect.com/science/article/pii/S0950584921000549
P62	Google	28/11/2024 07:26:00	Micro Frontend Architecture	https://www.reddit.com/r/reactjs/comments/1gwdxua/micro_frontend_architecture/?rdt=64908
P63	Google	28/11/2024 07:26:00	Motivations, benefits, and issues for adopting Micro-Frontends: : A Multivocal Literature Review	https://dl.acm.org/doi/10.1016/j.infsof.2021.106571
P64	Google	28/11/2024 07:27:00	Vinta Software	https://www.instagram.com/vintasoftware/p/CF0EYzyH3bW/
P65	Google	28/11/2024 07:28:00	Anais do Simpósio Brasileiro de Engenharia de Software (SBES)	https://sol.sbc.org.br/index.php/sbes/issue/view/1362
P66	Google	28/11/2024 07:28:00	MICRO FRONTENDS: A NEW PARADIGM FOR SCALABLE ANGULAR APPLICATIONS	https://www.researchgate.net/publication/384940204_MICRO_FRONTENDS_A_NEW_PARADIGM_FOR_SCALABLE_ANGULAR_APPLICATIONS

P67	Google	28/11/2024 07:28:00	Frontend Insights	https://www.thoughtworks.com/insights/topic/frontend
P68	Google	28/11/2024 07:28:00	Micro-Frontends	https://conf.researchr.org/details/ecsa-2022/ecsa-2022-workshops-tutorials/2/Micro-Frontends
P69	Google	28/11/2024 07:30:00	Stackademic	https://blog.stackademic.com/tagged/microservice-architecture
P70	Google	28/11/2024 07:30:00	Microfrontend, patterns e antipatterns con Luca Mezzalira (AWS)	https://www.gitbar.it/episodes/ep131-microfrontend-patterns-e-antipatterns-con-luca-mezzalira-aws
P71	Google	28/11/2024 07:31:00	What are Performance Anti-Patterns in System Design	https://www.geeksforgeeks.org/what-are-performance-anti-patterns-in-system-design/
P72	Google	28/11/2024 07:32:00	Cloud Design Patterns: Building Reliable & Scalable Applications	https://anarsolutions.com/category/technology/design-patterns/
P73	Google	28/11/2024 07:32:00	Envisioning the future with Micro frontends web development	https://www.technoidentity.com/insights/envisioning-the-future-with-micro-frontends-web-development/
P74	Google	28/11/2024 07:32:00	Post	https://www.threads.net/?error=invalid_post
P75	Google	28/11/2024 07:33:00	MICRO FRONTENDS: A NEW PARADIGM FOR SCALABLE ANGULAR APPLICATIONS	https://iaeme.com/MasterAdmin/Journal_uploads/IJCET/VOLUME_15_ISSUE_5/IJCET_15_05_041.pdf
P76	Google	28/11/2024 07:33:00	Micro-frontends anti-patterns	https://www.ugidotnet.org/e/sessione/2890/Micro-frontends-anti-patterns
P77	Google	28/11/2024 07:34:00	Microfrontends should be your last resort	https://www.reddit.com/r/programming/comments/1fvwie3/microfrontends_should_be_your_last_resort/
P78	Google	28/11/2024 07:34:00	Micro Frontends: Architecting Front-End Applications as Independent Microservices	https://softwarepatternslexicon.com/patterns-ts/7/8/
P79	Google	28/11/2024 07:35:00	A model-driven approach for continuous performance engineering in microservice-based systems	https://ouci.dntb.gov.ua/en/works/4N2W0o24/
P80	Google	28/11/2024 07:35:00	Current Trends in Frontends	https://archive.qconlondon.com/londonmar/track/current-trends-frontends
P81	Google	28/11/2024 07:36:00	And “Back to Stage“ it was!	https://globaldigitalfactory.allianz.com/blog/and--back-to-stage--it-was-.html
P82	Google	28/11/2024 07:36	Microfrontends should be your last resort	https://www.breck-mckye.com/blog/2023/05/Microfrontends-should-be-your-last-resort/
P83	Google	28/11/2024 07:37:00	QCon London 2022 - Current Trends in Frontends	https://www.facebook.com/QCon/videos/qcon-london-2022-current-trends-in-frontends/1160012331485798/
P84	Google	28/11/2024 07:38:00	#47 - Micro-Frontends and the Socio-Technical Aspect - Luca Mezzalira	https://pt.everand.com/podcast/592268089/47-Micro-Frontends-and-the-Socio-Technical-Aspect-Luca-Mezzalira
P85	Google	28/11/2024 07:38:00	4 Micro-Frontend Anti-Patterns	https://devpress.csdn.net/cloudnative/62fb6fb0c677032930800063.html
P86	Google	28/11/2024 07:40:00	Micro-frontends anti-patterns	https://www.infoq.cn/video/5PdMk6GpiC0X5Sg8wgkU
P87	Google	28/11/2024 07:40:00	Chris Kitson: Micro-frontends - reversing the anti-pattern!	https://www.youtube.com/watch?v=fSn-kTYylqw&ab_channel=NIDevConf
P88	Google	28/11/2024 07:41:00	Best Practices Microservices	https://systemsarchitect.io/docs/requirements/systems/services/apis/best-practices-microservices
P89	Google	28/11/2024 07:41:00	React the Wrong Way: 4 Anti-Patterns to Avoid	https://blog.bitsrc.io/react-the-wrong-way-4-anti-patterns-to-avoid-2d68a28aac00

P90	Google	28/11/2024 07:41:00	Slicing your application into micro frontends	https://subscription.packtpub.com/book/programming/9781837631971/14/ch14lvl1sec86/slicing-your-application-into-micro-frontends
P91	Google	28/11/2024 07:42:00	Micro-frontend	https://pt.slideshare.net/slideshow/micro-frontend/181808173
P92	Google	28/11/2024 07:43:00	Enterprise Angular: Micro Frontends and Moduliths with Angular	https://leanpub.com/enterprise-angular
P93	Google	28/11/2024 07:43:00	Migration Process from Monolithic to Micro Frontend Architecture in Mobile Applications	https://ceur-ws.org/Vol-3627/paper05.pdf
P94	Google	28/11/2024 07:44:00	4 Lessons Learned from Building Microfrontends	https://thenewstack.io/4-lessons-learned-from-building-microfrontends/
P95	Google	28/11/2024 07:44:00	Anti Patterns	https://awesome-architecture.com/anti-patterns/anti-patterns/
P96	Google	28/11/2024 07:45:00	React context between microfrontends	https://stackoverflow.com/questions/71532470/react-context-between-microfrontends
P97	Google	28/11/2024 07:47:00	Dividing frontend from backend is an antipattern	https://www.thoughtworks.com/en-br/insights/blog/dividing-frontend-backend-antipattern
P98	Google	28/11/2024 07:47:00	Is Returning Composables from Composables an Anti-Pattern in Vue Applications?	https://markus.oberlehner.net/blog/is-returning-composables-from-composables-an-anti-pattern-in-vue-applications/
P99	Google	28/11/2024 07:48:00	Micro frontends numa aplicação de pré-contabilidade	https://recipp.ipp.pt/handle/10400.22/24267
P100	Google	28/11/2024 07:48:00	Using Bahmni Forms in OpenMRS Microfrontends	https://talk.openmrs.org/t/using-bahmni-forms-in-openmrs-microfrontends/28548
P101	Google	28/11/2024 07:49:00	5 Things to Read This Week - 25th Jun 2019	https://thatsabug.com/blog/2019-06-25/5_things/
P102	Google	28/11/2024 07:50:00	Micro Frontend	https://switch-case.io/post/micro-frontend/
P103	Google	28/11/2024 07:51:00	Micro-Frontends anti-patterns	https://www.fevr.it/eventi/2021/12/micro-frontends-anti-patterns/
P104	Google	28/11/2024 07:52:00	What are Micro-Frontends? Really...	https://www.infoxicator.com/what-are-micro-frontends-really
P105	Google	28/11/2024 07:53:00	Monorepos in JavaScript, Anti-Pattern	https://scribe.rip/@PepsRyuu/monorepos-in-javascript-anti-pattern-917603da59c8
P106	Google	28/11/2024 08:09:00	Microfrontends — when they aren't the answer (React, Angular, Vue etc)	https://javascript.plainenglish.io/microfrontends-when-they-arent-the-answer-react-angular-vue-etc-45dcf266f6f9
P107	Google	28/11/2024 08:10:00	Motivations, benefits, and issues for adopting Micro-Frontends: A Multivocal Literature Review	https://www.researchgate.net/publication/350347666_Motivations_benefits_and_issues_for_adopting_Micro-Frontends_A_Multivocal_Literature_Review
P108	Google	28/11/2024 08:10	Micro-frontend "Blackbox Pattern"	https://medium.com/@ngkamperlo/micro-frontend-blackbox-pattern-295c40b681e4
P109	Google	28/11/2024 08:10:00	Shift Remote FRONTEND: Micro Frontend Architecture: A Look Into the Future - Ante Tomic (Infobip)	https://pt.slideshare.net/slideshow/shift-remote-frontend-micro-frontend-architecture-a-look-into-the-future-ante-tomic-infobip/236680973
P110	Google	28/11/2024 08:11:00	MICRO-FRONTEND ARCHITECTURE WITH REACT: A COMPREHENSIVE GUIDE	https://iaeme.com/Home/article_id/IJCET_15_06_012
P111	Google	28/11/2024 08:11:00	Break up your monolith: give teams the freedom to scale with micro frontends	https://forto.com/en/blog/tech-blog-micro-frontends/

P112	Google	28/11/2024 08:11:00	#47 - Micro-Frontends and the Socio-Technical Aspect - Luca Mezzalira	https://techleadjournal.dev/episodes/47/
P113	Google	28/11/2024 08:13:00	Micro-Frontends Weekly - Issue #12	https://mfe-weekly.beehiiv.com/p/microfrontends-weekly-issue-12
P114	Google	28/11/2024 08:13:00	What are Micro-Frontends and How to Use Them	https://gotopia.tech/articles/161/what-are-microfrontends-and-how-to-use-them
P115	Google	28/11/2024 08:13:00	Micro Frontend Architecture - Luca Mezzalira, DAZN	https://www.youtube.com/watch?v=BuRB3djraeM&ab_channel=UXDX
P116	Google	28/11/2024 08:15:00	Luca Mezzalira (@amazonwebservices) - "Micro-frontends anti-patterns" at the Code Europe 2023	https://partner.biz.ua/ytbe/R2Ydq5E_8ts/luca-mezzalira-@amazonwebservices-micro-frontends-anti-patterns-at-the-code-europe-2023
P117	Google	28/11/2024 08:15:00	How Micro-frontend frameworks are replacing legacy monoliths	https://servian.dev/how-micro-frontend-frameworks-are-replacing-legacy-monoliths-f66f34d06a2
P118	Google	28/11/2024 08:16:00	Monorepo and Micro-Frontends with Jonathan Creamer	https://semaphoreci.com/blog/monorepo-micro-frontends-jonathan-creamer
P119	Google	28/11/2024 08:16:00	Design Systems for Micro Frontends	https://epb.bibl.th-koeln.de/frontdoor/deliver/index/docId/1666/file/SCIBachelor_Design_Systems_and_Micro_Frontends.pdf
P120	Google	28/11/2024 08:17:00	O que é Code Smell?	https://pt.stackoverflow.com/questions/100016/o-que-%C3%A9-code-smell
P121	Google	28/11/2024 08:17:00	Micro Frontends	https://dr-martin-kramer.com/micro-frontends/
P122	Google	28/11/2024 08:18:00	Angular v18 is now available!	https://blog.angular.dev/angular-v18-is-now-available-e79d5ac0affe
P123	Google	28/11/2024 08:18:00	What Are Micro-Frontends & How to Use Them • Luca Mezzalira & Lucas Dohmen • GOTO 2022	https://www.youtube.com/watch?v=-thWgobMW_I&ab_channel=GOTOConferences
P124	Google	28/11/2024 08:18:00	Luca Mezzalira - Micro Frontends With Module Federation	https://www.youtube.com/watch?v=tTwoSWJObZs&ab_channel=CITYJSCONFERENCE
P125	Google	28/11/2024 08:19:00	How Micro-frontend frameworks are replacing legacy monoliths	https://servian.dev/how-micro-frontend-frameworks-are-replacing-legacy-monoliths-f66f34d06a2
P126	Google	28/11/2024 08:19:00	Event listener performance antipatterns	https://subscription.packtpub.com/book/programming/9781804612279/11/ch11lvl1sec59/event-listener-performance-antipatterns
P127	Google	28/11/2024 08:20:00	Monorepo and Micro-Frontends with Jonathan Creamer	https://semaphoreci.com/blog/monorepo-micro-frontends-jonathan-creamer
P128	Google	28/11/2024 08:20:00	Micro Frontend Architecture - Luca Mezzalira, DAZN	https://www.youtube.com/watch?v=BuRB3djraeM&ab_channel=UXDX
P129	Google	28/11/2024 08:21:00	Angular v18 is now available!	https://blog.angular.dev/angular-v18-is-now-available-e79d5ac0affe
P130	Google	28/11/2024 08:21:00	Micro Frontends	https://dr-martin-kramer.com/micro-frontends/
P131	Google	28/11/2024 08:22:00	What Are Micro-Frontends & How to Use Them • Luca Mezzalira & Lucas Dohmen • GOTO 2022	https://www.youtube.com/watch?v=-thWgobMW_I&ab_channel=GOTOConferences
P132	Google	28/11/2024 08:23:00	Luca Mezzalira - Micro Frontends With Module Federation	https://www.youtube.com/watch?v=tTwoSWJObZs&ab_channel=CITYJSCONFERENCE
P133	Google	28/11/2024 08:24:00	Serverless-Side Rendering Micro-Frontends by Luca Mezzalira | SLA Conference	https://www.youtube.com/watch?v=QD2BvPfNc6c&ab_channel=ServerlessArchitectureConference
P134	Google	28/11/2024 08:24:00	Micro Frontends Conference 2023 - Luca Mezzalira: Micro Frontends Discovery	https://www.youtube.com/watch?v=U3KJiC9c_7I&ab_channel=smapiot

P135	Google	28/11/2024 08:25:00	[EN] Interview: Micro Frontends	https://www.youtube.com/watch?v=obwWdgxQUxQ&ab_channel=ManfredSteyer
P136	Google	28/11/2024 08:26:00	Monolith to Micro-Frontends using Webpack Module Federation // Ori Adijes, Architect at Windward	https://www.youtube.com/watch?v=Q9gDiA35LLo&ab_channel=IsraeliTechRadar
P137	Google	28/11/2024 08:26:00	Building Micro-Frontends • Luca Mezzalira & Lucas Dohmen • GOTO 2022	https://www.youtube.com/watch?v=DG9puFuUb7E&ab_channel=GOTOConferences
P138	Google	28/11/2024 08:27:00	Micro Frontends with Web Component: Part 1 – Introduction to Micro Frontends	https://www.youtube.com/watch?v=uJ_rqmZaNDQ&ab_channel=FiveMinuteTech
P139	Google	28/11/2024 08:27:00	Lightning Talk #2: Micro-frontend "Blackbox Pattern"	https://www.youtube.com/watch?v=CZeCgmoFEOw&ab_channel=AthensSDETMeetup
P140	Google	28/11/2024 08:27:00	Building micro frontends with Single-Spa – Kamil Dzieniszewski	https://www.youtube.com/watch?v=lQkR-Vlnbgs&ab_channel=BECFinancialTechnologies
P141	Google	28/11/2024 08:29:00	Beyond the basics of Module Federation micro frontends – Manfred Steyer	https://youtube.com/watch?v=Qcbmit5TRB8&ab_channel=BECFinancialTechnologies
P142	Google	28/11/2024 08:29:00	Micro Frontends - foundations | Tomasz Krajewski | Conf42 JavaScript 2021	https://www.youtube.com/watch?v=ydG6j83DTnU&ab_channel=Conf42
P143	Google	28/11/2024 08:29:00	Micro-Frontend	https://www.youtube.com/watch?v=tY9MXocUoWM&ab_channel=TechTalksWithSantosh
P144	Google	28/11/2024 08:30:00	Online: Demystifying Micro-Frontends	https://www.youtube.com/watch?v=Hz_XmnsJvFw&ab_channel=DataWorksMD
P145	Google	28/11/2024 08:31:00	Micro-Frontends Decisions Framework with Luca Mezzalira	https://www.youtube.com/watch?v=1co8055hi_I&ab_channel=JSWORLDConference
P146	Google	28/11/2024 08:31:00	Keynote Session: I don't understand micro-frontends | Luca Mezzalira	https://www.youtube.com/watch?v=UbblS2Twh8A&ab_channel=JSPolandConf
P147	Google	28/11/2024 08:32:00	Micro-frontends Anti-Patterns	https://cyberspaceandtime.com/MICROFRONTENDS-ANTIPATTERNS-SEVEN-YEARS-IN-THE-TRENCHES-kQnX11tCXhVS7Ze7li1LLFFh4LB9jt_gE.htm
P148	Google	28/11/2024 08:33:00	Clean Architecture | Testes Unitários no Frontend	https://www.youtube.com/watch?v=Xg38Hv6mLQU&ab_channel=JuniorMarques
P149	Medium	29/11/2024 07:46:00	UI microservices — reversing the anti-pattern (micro frontends)	https://medium.com/@kitson.mac/ui-microservices-reversing-the-anti-pattern-375bc22287b0
P150	Medium	29/11/2024 07:46:00	Top 10 Microservice Anti-Patterns	https://blog.bitsrc.io/10-microservice-anti-patterns-278bcb7f385d
P151	Medium	29/11/2024 07:47:00	Micro-frontends Weekly — 18/11/2022	https://medium.com/@luisvieira_gmr/micro-frontends-weekly-18-11-2022-39f11d1fb4d1
P152	Medium	29/11/2024 07:47:00	How Micro-frontend frameworks are replacing legacy monoliths	https://servian.dev/how-micro-frontend-frameworks-are-replacing-legacy-monoliths-f66f34d06a2
P153	Medium	29/11/2024 07:47:00	Micro-frontend "Blackbox Pattern"	https://medium.com/@ngkamperlo/micro-frontend-blackbox-pattern-295c40b681e4
P154	Medium	29/11/2024 07:48:00	Problems with Micro-frontends	https://medium.com/swlh/problems-with-micro-frontends-8a8fc32a7d58
P155	Medium	29/11/2024 07:48:00	Micro Front-End Architecture at Enterprise Scale (Updated July 2020)	https://medium.com/swlh/micro-front-end-architecture-at-enterprise-scale-updated-july-2020-9159a4e0cc49
P156	Medium	29/11/2024 07:49:00	React Anti Patterns	https://medium.com/@suraj.kc/react-anti-patterns-909ecf193701

P157	Medium	29/11/2024 07:49:00	Microfrontends — My talk from NI Dev Conf 2019	https://medium.com/@kitson.mac/microfrontends-my-talk-from-ni-dev-conf-2019-f648fe2cba32
P158	Medium	29/11/2024 07:50:00	Frontend Design Systems: Monolith Vs Multirepo Vs Monorepo	https://medium.com/@rajasaikiranvemula/frontend-design-systems-monolith-vs-multirepo-vs-monorepo-8395b4e4773d
P159	Medium	29/11/2024 07:51:00	-	https://medium.com/@kitson.mac/you-will-only-see-the-advantages-of-microfrontends-at-scale-d5cbaca374fa
P160	Medium	29/11/2024 07:51:00	Backends for Frontends Pattern (BFF)	https://medium.com/@salem.naser.elashry/backends-for-frontends-pattern-bff-da46bc22b9d0
P161	Medium	29/11/2024 07:52:00	Microfrontends — when they aren't the answer (React, Angular, Vue etc)	https://javascript.plainenglish.io/microfrontends-when-they-arent-the-answer-react-angular-vue-etc-45dcf266f6f9
P162	Medium	29/11/2024 07:52:00	Monorepos in JavaScript, Anti-Pattern	https://medium.com/@PepsRyuu/monorepos-in-javascript-anti-pattern-917603da59c8
P163	Medium	29/11/2024 07:52:00	Webpack module federation | Think twice before sharing a dependency	https://medium.com/@marvusm.mmi/webpack-module-federation-think-twice-before-sharing-a-dependency-18b3b0e352cb
P164	Medium	29/11/2024 07:53:00	Implementing Multi-Framework in Vanilla JS with Native Federation	https://singhdheerendra.medium.com/implementing-multi-framework-in-vanilla-js-with-native-federation-d7580f571a7d
P165	Medium	29/11/2024 07:53:00	A Better Frontend Component Structure: Component Trees	https://betterprogramming.pub/a-better-frontend-component-structure-component-trees-5a99ed6d1ece
P166	Medium	29/11/2024 07:54:00	-	https://medium.com/@kitson.mac/hi-paige-thanks-for-reaching-out-with-a-great-question-982e45ef90b9
P167	Medium	29/11/2024 07:54:00	A journey through frontend Aspect-Oriented Programming — Theory	https://medium.com/@exfabrica/a-journey-throuth-frontend-aspect-oriented-programming-35f81d8eb1d0
P168	Medium	29/11/2024 07:54:00	Common React Anti-patterns you should avoid	https://medium.com/@paulohfev/common-react-anti-patterns-you-should-avoid-eb9b605fded1
P169	Medium	29/11/2024 07:55:00	Creating Micro-frontends using Web Components (with support for Angular and React)	https://javascript.plainenglish.io/create-micro-frontends-using-web-components-with-support-for-angular-and-react-2d6db18f557a
P170	Medium	29/11/2024 07:56:00	Yosua Halim	https://yosua-halim.medium.com/lists
P171	Medium	29/11/2024 07:56:00	UX Fails: The "Junk Drawer" Problem	https://medium.com/design-bootcamp/ux-smack-down-the-junk-drawer-problem-a80b4e80a566
P172	Medium	29/11/2024 07:57:00	redux-thunk is bad architecture for organizations	https://jongleberry.medium.com/redux-thunk-is-bad-architecture-for-organizations-8205a792e5fe
P173	Medium	29/11/2024 07:57:00	Paul Sweeney	https://medium.com/@PepsRyuu/monorepos-in-javascript-anti-pattern-917603da59c8
P174	Medium	29/11/2024 07:58:00	Chris Kitson	https://medium.com/@kitson.mac
P175	Medium	29/11/2024 07:58:00	How to Choose Microservice's Boundaries?	https://blog.bitsrc.io/how-to-choose-microservices-boundaries-5c68b0b1af24
P176	Medium	29/11/2024 07:58:00	How to: Communication protocols	https://medium.com/xgeeks/how-to-communication-protocols-ab7037507345
P177	Medium	29/11/2024 07:59:00	Monorepo or Multirepo?	https://blog.kloia.com/monorepo-or-multirepo-ddcfe531b38b

P178	Medium	29/11/2024 07:59:00	Are TypeScript Barrel Files an Anti-pattern?	https://steven-lemon182.medium.com/are-typescript-barrel-files-an-anti-pattern-72a713004250
P179	Medium	29/11/2024 08:00:00	Writing Comments Is Lazy Coding	https://javascript.plainenglish.io/writing-comments-is-lazy-coding-0a0cbdc725ec
P180	Medium	29/11/2024 08:00:00	The Challenges and Traps of Architecting Sociotechnical Systems	https://medium.com/nick-tune-tech-strategy-blog/the-challenges-and-traps-of-architecting-sociotechnical-systems-94272a7c790f
P181	Medium	29/11/2024 08:01:00	Don't Build a Distributed Monoliths	https://medium.com/@osama94/dont-build-a-distributed-monoliths-46daa8b62390
P182	Medium	29/11/2024 08:01:00	React Anti-Pattern: Overloaded	https://codeburst.io/react-anti-pattern-overloaded-8bfacab49421
P183	Medium	29/11/2024 08:02:00	How to use GraphQL to build Backend-For-Frontends (BFFs)	https://blog.bitsrc.io/how-to-use-graphql-to-build-backend-for-frontends-bffs-4b7e5a0105d0
P184	Medium	29/11/2024 08:02:00	Why our team cancelled our move to microservices	https://steven-lemon182.medium.com/why-our-team-cancelled-our-move-to-microservices-8fd87898d952
P185	Medium	29/11/2024 08:02:00	Apollo: Swiggy's Code Analysis Platform	https://bytes.swiggy.com/apollo-swiggys-code-analysis-platform-b80e8292f29
P186	Medium	29/11/2024 08:03:00	Architecture Ownership Patterns for Team Topologies. Part 2: Single Team Patterns	https://medium.com/nick-tune-tech-strategy-blog/architecture-ownership-patterns-for-team-topologies-part-2-single-team-patterns-943d31854285
P187	Medium	29/11/2024 08:03:00	A holistic strategy for the selection of open-source packages	https://medium.com/syngenta-digitalblog/a-holistic-strategy-for-the-selection-of-open-source-packages-dc814d14163b
P188	Medium	29/11/2024 08:04:00	Architectural Patterns	https://medium.com/architectural-patterns
P189	Medium	29/11/2024 08:05:00	Diving into the BFF Tool Pool	https://blog.bitsrc.io/diving-into-the-bff-tool-pool-70169a91f2a9
P190	Medium	29/11/2024 08:06:00	Microservices	https://medium.com/@jplopez42/list/563965581a71
P191	Medium	29/11/2024 08:06:00	Guide to Advanced React Hooks	https://medium.com/jahia-techblog/guide-to-advanced-react-hooks-3402a1b397db
P192	Medium	29/11/2024 08:07:00	Why you shouldn't use access tokens in your front-end any more	https://abstarreveld.medium.com/why-you-shouldnt-use-access-tokens-in-your-front-end-any-more-490545665125
P193	Medium	29/11/2024 08:08:00	We're Writing Too Many Tests	https://steven-lemon182.medium.com/were-writing-too-many-tests-2155a681dbf2
P194	Medium	29/11/2024 08:09:00	Frontend Weekly Digest (4–10 Feb 2019)	https://frontender-ua.medium.com/frontend-weekly-digest-4-10-feb-2019-287b475dc794
P195	Medium	29/11/2024 08:09:00	REST is easy with GraphQL as a sidekick. Two are better than one.	https://sureshkandula.medium.com/rest-is-easy-with-graphql-as-a-sidekick-two-are-better-than-one-54e8e1915cfe
P196	Medium	29/11/2024 08:09:00	Vue.js App Performance Optimization: part 1 — Introduction to performance optimization and lazy loading.	https://itnext.io/vue-js-app-performance-optimization-part-1-introduction-to-performance-optimization-and-lazy-29e4ff101019
P197	Medium	29/11/2024 08:10:00	Clever Code is Really Bad	https://medium.com/@codecraftsphere/clever-code-is-really-bad-2400d51b3c42

P198	Medium	29/11/2024 08:10:00	Mastering CI/CD: A Comprehensive Guide to Implementing the Testing Pyramid in Your Pipeline Strategy	https://blog.bitsrc.io/mastering-ci-cd-a-comprehensive-guide-to-implementing-the-testing-pyramid-in-your-pipeline-68ed248dcc08
P199	Medium	29/11/2024 08:11:00	Storybook: A Must-Have Library for Creating Application Components	https://javascript.plainenglish.io/how-storybook-improved-our-teams-workflows-and-communication-2ee57fcb2a87
P200	Medium	29/11/2024 08:12:00	Serverless AWS CDK Pipeline Best Practices & Patterns — Part 1	https://blog.serverlessadvocate.com/serverless-aws-cdk-pipeline-best-practices-patterns-part-1-ab80962f109d
P201	Medium	29/11/2024 08:13:00	Implementing a timer using React	https://medium.com/front-end-weekly/implementing-a-timer-using-react-47f7bcab19bc
P202	Medium	29/11/2024 08:14:00	Recipes to write better Jest tests with the React Testing Library (Part 1)	https://builders.travelperk.com/recipes-to-write-better-jest-tests-with-the-react-testing-library-part-1-670aaf3451d1
P203	Medium	29/11/2024 08:15:00	Dependency injection: setting up InversifyJS IoC for Typescript Apps	https://medium.com/tkssharma/dependency-injection-setting-up-inversifyjs-ioc-for-typescript-apps-da65edfb1ea8
P204	Medium	29/11/2024 08:16:00	What I was doing wrong — Spring autowiring and feature flagging	https://medium.com/codex/what-i-was-doing-wrong-spring-autowiring-and-feature-flagging-12bb979a08f6
P205	Medium	29/11/2024 08:16:00	React: Dynamically Rendering Different Components without Switch: the Capitalized Reference Technique	https://j5cookie.medium.com/react-dynamically-rendering-different-components-without-switch-the-capitalized-reference-e668d89e460b
P206	Medium	29/11/2024 08:17:00	React	https://medium.com/@DotDev/list/react-f82304a94dd0
P207	Medium	29/11/2024 08:17:00	How to deploy an Angular app to Docker	https://javascript.plainenglish.io/build-angular-application-with-lint-unit-tests-chrome-headless-and-release-to-nginx-inside-bdc84ea9e5ab
P208	Medium	29/11/2024 08:18:00	The Singleton Pattern In TypeScript	https://blog.bitsrc.io/the-singleton-pattern-in-typescript-b906303fda93
P209	Medium	29/11/2024 08:19:00	Get on the Event Bus: Vue.js	https://medium.com/@johnbaldwin/get-on-the-event-bus-vue-js-abcf63d7342d
P210	Medium	29/11/2024 08:19:00	Friday Frontend: NodeConf Colombia Edition	https://medium.com/friday-frontend/friday-frontend-nodeconf-colombia-edition-c05752e4517f
P211	Medium	29/11/2024 08:21:00	[Vue]- (Vue Props and Emits-Is passing Vue props functions anti-pattern?)	https://medium.com/@yhosutun2491/vue-%E5%89%8D%E7%AB%AF%E6%96%B0%E6%89%8B%E6%B7%B1%E5%85%A5%E7%B3%BB%E5%88%97-vue-props-and-emits-is-passing-vue-props-functions-ani-pattern-6a3c9e90aab3
P212	Medium	29/11/2024 08:21:00	Understanding Search Algorithms in Flutter: Data Structures and Algorithms Explained	https://medium.com/@kasunpradeep.ds/understanding-search-algorithms-in-flutter-data-structures-and-algorithms-explained-cab6f53b41d8
P213	Medium	29/11/2024 08:22:00	Considerations to Using Content Delivery Network(CDN).	https://medium.com/@lan_carson/considerations-to-using-content-delivery-network-cdn-5efbcda95c12
P214	Medium	29/11/2024 08:22:00	NodeJS — Investing into clean architecture	https://medium.com/@epavliy/nodejs-investing-into-clean-architecture-69ebd3eb5e25
P215	Medium	29/11/2024 08:22:00	In the beginning, was the monolith (part 2) / database	https://medium.com/codex/in-the-beginning-was-the-monolith-part-2-database-ba4eed11fbbb
P216	Medium	29/11/2024 08:23:00	Merge Branches Sooner with Synchronous Code Review	https://steven-lemon182.medium.com/merge-branches-sooner-with-synchronous-code-review-78aa3cfb1df6

P217	Medium	29/11/2024 08:23:00	Is Your Agile Team Incremental, or Iterative?	https://steven-lemon182.medium.com/is-your-agile-team-incremental-or-iterative-33135a4c28b8
P218	Medium	29/11/2024 08:23:00	Managing Domain Events in a Microservices Environment	https://blog.bitsrc.io/managing-domain-events-in-a-microservices-environment-33eda865b187
P219	Medium	29/11/2024 08:25:00	How does APP_INITIALIZER work? So what do you need to know about dynamic configuration in Angular?	https://itnext.io/how-does-app-initializer-work-so-what-do-you-need-to-know-about-dynamic-configuration-in-angular-718e7c345971
P220	Medium	29/11/2024 08:25:00	( )Vue.js App : part2 — Lazy loading (anti-pattern)	https://medium.com/@upstairs0102/%E8%AD%AF-vue-js-app%E6%95%88%E8%83%BD%E5%84%AA%E5%8C%96-part2-lazy-loading%E8%B7%AF%E7%94%B1%E5%8F%8A%E7%AC%AC%E4%B8%89%E6%96%B9%E5%BA%AB%E6%89%93%E5%8C%85%E5%8F%8D%E5%90%91%E6%A8%A1%E5%BC%8F-anti-pattern-32f09c0a65c8
P221	Medium	29/11/2024 08:25:00	Docker and NPM— an introduction for UI developers	https://medium.com/@kitson.mac/docker-and-npm-an-introduction-for-ui-developers-22150cbfc5bb
P222	Medium	29/11/2024 08:26:00	Write Vue Like You Write React	https://betterprogramming.pub/write-vue-like-you-write-react-545eafb60e6a
P223	Medium	29/11/2024 08:27:00	Comparing React State Management Libraries: Redux, Zustand, Recoil, and MobX	https://javascript.plainenglish.io/comparing-react-state-management-libraries-redux-zustand-recoil-and-mobx-945402dc0cb
P224	Medium	29/11/2024 08:28:00	A (V)ay to write understandable software	https://medium.com/hepsiburadatech/a-way-to-write-understandable-software-b45c9ba4f228
P225	Medium	29/11/2024 08:28:00	How to Choose the Right Branching Strategy	https://steven-lemon182.medium.com/how-to-choose-the-right-branching-strategy-5ada39f49477
P226	Medium	29/11/2024 08:29:00	New Chaos Experiment Event	https://medium.com/weaveworks/chaos-meets-gitops-stress-test-and-improve-your-app-speed-using-chaosiq-with-weave-cloud-bfdf3b6f96c3
P227	Medium	29/11/2024 08:30:00	SonarQube and ReactJS	https://sylvainleroy.medium.com/sonarqube-and-reactjs-631d5a65d2f9
P228	Medium	29/11/2024 08:31:00	Branch-based WEB Acceptance Test With GitLab Review App	https://medium.com/trendyol-tech/branch-based-web-acceptance-test-with-gitlab-review-app-29fa608ce1e7
P229	Medium	29/11/2024 08:32:00	Upgrading to Angular 17 with Jorge Cano	https://medium.com/angularidades/upgrading-to-angular-17-with-jorge-cano-0c2f11393760
P230	Medium	29/11/2024 08:32:00	How To Write Modern React App Using gRPC And Envoy	https://medium.com/effective-development/how-to-write-modern-react-app-using-grpc-and-envoy-a9d9a4f2785e
P231	Medium	29/11/2024 08:32:00	State Management in React — Overview	https://medium.com/valtech-ch/state-management-in-react-overview-2647b06ec6ef
P232	Medium	29/11/2024 08:33:00	( )Vue.js App : part2 — Lazy loading (anti-pattern)	https://medium.com/@upstairs0102/%E8%AD%AF-vue-js-app%E6%95%88%E8%83%BD%E5%84%AA%E5%8C%96-part2-lazy-loading%E8%B7%AF%E7%94%B1%E5%8F%8A%E7%AC%AC%E4%B8%89%E6%96%B9%E5%BA%AB%E6%89%93%E5%8C%85%E5%8F%8D%E5%90%91%E6%A8%A1%E5%BC%8F-anti-pattern-32f09c0a65c8
P233	Medium	29/11/2024 08:33:00	Where Are You Putting Your Interfaces?	https://medium.com/@matteopampana/where-are-you-putting-your-interfaces-07cd5ae2bbd7

P234	Medium	29/11/2024 08:33:00	Hitchhiker's guide to Web Accessibility	https://blog.bitsrc.io/accessibility-for-astronauts-d9a8c729e6e6
P235	Medium	29/11/2024 08:34:00	I Hate JavaScript's for loops. Let Me Tell You Why.	https://blog.bitsrc.io/i-hate-javascripts-for-loops-let-me-tell-you-why-c18bc651d04a
P236	Medium	29/11/2024 08:34:00	Unidirectional Data Flow in Vue & how it helped us?	https://medium.com/trendyol-tech/unidirectional-data-flow-in-vue-how-it-helped-us-5c90efcce4fb
P237	Medium	29/11/2024 08:34:00	What I have learned Architecting Microservices	https://medium.com/hackernoon/what-i-have-learned-architecting-microservices-cbccc2182530
P238	Medium	29/11/2024 08:35:00	Don't Leak Your (Database) Internals	https://levelup.gitconnected.com/dont-leak-your-database-internals-1a6870aea4e6
P239	Medium	29/11/2024 08:36:00	Code Smells in Application Structure	https://levelup.gitconnected.com/code-smells-in-application-structure-4d30761bd055
P240	Medium	29/11/2024 08:36:00	Code Generation in React with RTK Query	https://steven-lemon182.medium.com/code-generation-in-react-with-rtk-query-e2410db6c868
P241	Medium	29/11/2024 08:37:00	Standardizing RESTful APIs	https://blog.jaykmr.com/standardizing-restful-apis-7f0b94d12d05
P242	Medium	29/11/2024 08:37:00	Robot Software Architect — An Introduction to AppMap	https://medium.com/sysco-labs/robot-software-architect-appmap-5e682afd3f1f
P243	Medium	30/11/2024 06:54:00	Vue vs React vs... Svelte?!	https://medium.com/@faulknerproject/vue-vs-react-vs-svelte-5f93d70d2618
P244	Medium	30/11/2024 06:54:00	Architectural Patterns For IoT — Why use Micro services ?	https://medium.com/@prashunjaveri/architectural-patterns-for-iot-why-use-micro-services-3154bfbafcce
P245	Medium	30/11/2024 06:54:00	HOW FUNCTIONAL PROGRAMMING CAN SOLVE YOUR BIGGEST CODING NIGHTMARES — UNLOCK THE POWER OF FP (PART 1 OF 4)	https://medium.com/@wackyworld_jenshenneberg/how-functional-programming-can-solve-your-biggest-coding-nightmares-unlock-the-power-of-fp-part-ed7013eed105
P246	Medium	30/11/2024 06:54:00	Evolving JavaScript Part 3	https://engineering.gusto.com/evolving-javascript-be0338eb0d38
P247	Medium	30/11/2024 06:56:00	DevPoint Labs: One Year Later	https://medium.com/@AndrewLocke/devpoint-labs-one-year-later-89a337676eae
P248	Medium	30/11/2024 06:56:00	Why Is My Jest Test Suite So Slow?	https://blog.bitsrc.io/why-is-my-jest-suite-so-slow-2a4859bb9ac0
P249	Medium	30/11/2024 06:57:00	Next.js 13: What Do The New Bleeding-Edge Features Actually Do?	https://blog.bitsrc.io/next-js-13-what-do-the-new-bleeding-edge-features-actually-do-d3e5fd418563
P250	Medium	30/11/2024 06:57:00	Оптимизация производительности приложения Vue.js: часть 2 — Ленивая загрузка маршрутов и анти-паттерны сторонних бандлов.	https://medium.com/@denistolkachev/%D0%BE%D0%BF%D1%82%D0%B8%D0%BC%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F-%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D0%B8-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F-vue-js-4a1fb16d6325
P251	Medium	30/11/2024 06:57:00	Angular + ngrx : Gestion des requêtes http avec ngrx/effects...	https://medium.com/@ylerjen/angular-gestion-des-requ%C3%AAtes-http-avec-ngrx-effects-1b51dc3890a4
P252	Medium	30/11/2024 06:57:00	React Context: Sync vs Async	https://levelup.gitconnected.com/react-context-sync-vs-async-8d67562d4b4a
P253	Medium	30/11/2024 06:58:00	Life after Redux	https://itnext.io/life-after-redux-21f33b7f189e

P254	Medium	30/11/2024 06:59:00	Developing isomorphic applications using webpack	https://medium.com/hackernoon/developing-isomorphic-applications-using-webpack-eca814a418ad
P255	Scholar	30/11/2024 07:09:00	Micro-Frontends: Principles, Implementations, and Pitfalls	https://dl.acm.org/doi/abs/10.1145/3561846.3561853
P256	Scholar	30/11/2024 07:10:00	Motivations, benefits, and issues for adopting Micro-Frontends: A Multivocal Literature Review	https://www.sciencedirect.com/science/article/pii/S0950584921000549
P257	Scholar	30/11/2024 07:12:00	Design patterns and anti-patterns in microservices architecture : a classification proposal and study on open source projects	https://www.politesi.polimi.it/handle/10589/186745
P258	Scholar	30/11/2024 07:13:00	Suitability of Micro-Frontends for an AI as a Service Platform	https://reposit.haw-hamburg.de/handle/20.500.12738/14575
P259	Scholar	30/11/2024 07:13:00	Development of an angular components library to be used in micro-frontend architecture	https://recipp.ipp.pt/handle/10400.22/26487
P260	Scholar	30/11/2024 07:14:00	Building Micro-Frontends • Luca Mezzalira	https://books.google.com.br/books?hl=pt-BR&lr=&id=NDpPEAAAQBAJ&oi=fnd&pg=PR2&dq=(%E2%80%9Canti-pattern%E2%80%9D+OR+%E2%80%9Canti-patterns%E2%80%9D+OR+%E2%80%9Cantipattern%E2%80%9D+OR+%E2%80%9Cantipatterns%E2%80%9D+OR+%E2%80%9Canti+pattern%E2%80%9D+OR+%E2%80%9Canti+patterns%E2%80%9D)+AND+(%E2%80%9Cmicrofrontend%E2%80%9D+OR+%E2%80%9Cmicrofrontends%E2%80%9D+OR+%E2%80%9Cmicro+frontend%E2%80%9D+OR+%E2%80%9Cmicro+frontends%E2%80%9D+OR+%22micro-frontend%22+OR+%22micro-frontends%22)&ots=aLc4gESofZ&sig=JNntF15wVduJTMduSanYuYvKjis&redir_esc=y#v=onepage&q=(%E2%80%9Canti-pattern%E2%80%9D%20OR%20%E2%80%9Canti-patterns%E2%80%9D%20OR%20%E2%80%9Cantipattern%E2%80%9D%20OR%20%E2%80%9Cantipatterns%E2%80%9D%20OR%20%E2%80%9Canti%20pattern%E2%80%9D%20OR%20%E2%80%9Canti%20patterns%E2%80%9D)%20AND%20(%E2%80%9Cmicrofrontend%E2%80%9D%20OR%20%E2%80%9Cmicrofrontends%E2%80%9D%20OR%20%E2%80%9Cmicro%20frontend%E2%80%9D%20OR%20%E2%80%9Cmicro%20frontends%E2%80%9D%20OR%20%22micro-frontend%22%20OR%20%22micro-frontends%22)&f=false
P261	Scholar	30/11/2024 07:15:00	Micro Frontend-Based Development: Concepts, Motivations, Implementation Principles, and an Experience Report	https://www.scitepress.org/Papers/2024/126273/126273.pdf

P262	Scholar	30/11/2024 07:15:00	Embracing Microservices Design	https://books.google.com.br/books?hl=pt-BR&lr=&id=0plEEAAAQBAJ&oi=fnd&pg=PP1&dq=(%E2%80%9Canti-pattern%E2%80%9D+OR+%E2%80%9Canti-patterns%E2%80%9D+OR+%E2%80%9Cantipattern%E2%80%9D+OR+%E2%80%9Cantipatterns%E2%80%9D+OR+%E2%80%9Canti+pattern%E2%80%9D+OR+%E2%80%9Canti+patterns%E2%80%9D)+AND+(%E2%80%9Cmicrofrontend%E2%80%9D+OR+%E2%80%9Cmicrofrontends%E2%80%9D+OR+%E2%80%9Cmicro+frontend%E2%80%9D+OR+%E2%80%9Cmicro+frontends%E2%80%9D+OR+%22micro-frontend%22+OR+%22micro-frontends%22)&ots=rBP8EfTh3Y&sig=SWldVOsEPk3cT8ugmraqwrjtkJs&redir_esc=y#v=onepage&q=(%E2%80%9Canti-pattern%E2%80%9D%20OR%20%E2%80%9Canti-patterns%E2%80%9D%20OR%20%E2%80%9Cantipattern%E2%80%9D%20OR%20%E2%80%9Cantipatterns%E2%80%9D%20OR%20%E2%80%9Canti%20pattern%E2%80%9D%20OR%20%E2%80%9Canti%20patterns%E2%80%9D)%20AND%20(%E2%80%9Cmicrofrontend%E2%80%9D%20OR%20%E2%80%9Cmicrofrontends%E2%80%9D%20OR%20%E2%80%9Cmicro%20frontend%E2%80%9D%20OR%20%E2%80%9Cmicro%20frontends%E2%80%9D%20OR%20%22micro-frontend%22%20OR%20%22micro-frontends%22)&f=false
P263	Scholar	30/11/2024 07:16:00	Software Architecture Patterns for Serverless Systems	https://books.google.com.br/books?hl=pt-BR&lr=&id=gUs2EAAAQBAJ&oi=fnd&pg=PP1&dq=(%E2%80%9Canti-pattern%E2%80%9D+OR+%E2%80%9Canti-patterns%E2%80%9D+OR+%E2%80%9Cantipattern%E2%80%9D+OR+%E2%80%9Cantipatterns%E2%80%9D+OR+%E2%80%9Canti+pattern%E2%80%9D+OR+%E2%80%9Canti+patterns%E2%80%9D)+AND+(%E2%80%9Cmicrofrontend%E2%80%9D+OR+%E2%80%9Cmicrofrontends%E2%80%9D+OR+%E2%80%9Cmicro+frontend%E2%80%9D+OR+%E2%80%9Cmicro+frontends%E2%80%9D+OR+%22micro-frontend%22+OR+%22micro-frontends%22)&ots=5mmd_zHTel&sig=krQMcjvjUBdiYxeE6mAbxz7xHec&redir_esc=y#v=onepage&q=(%E2%80%9Canti-pattern%E2%80%9D%20OR%20%E2%80%9Canti-patterns%E2%80%9D%20OR%20%E2%80%9Cantipattern%E2%80%9D%20OR%20%E2%80%9Cantipatterns%E2%80%9D%20OR%20%E2%80%9Canti%20pattern%E2%80%9D%20OR%20%E2%80%9Canti%20patterns%E2%80%9D)%20AND%20(%E2%80%9Cmicrofrontend%E2%80%9D%20OR%20%E2%80%9Cmicrofrontends%E2%80%9D%20OR%20%E2%80%9Cmicro%20frontend%E2%80%9D%20OR%20%E2%80%9Cmicro%20frontends%E2%80%9D%20OR%20%22micro-frontend%22%20OR%20%22micro-frontends%22)&f=false

P264	Scholar	30/11/2024 07:17:00	MICRO-FRONTEND ARCHITECTURE WITH REACT: A COMPREHENSIVE GUIDE	https://www.researchgate.net/profile/Veeranjaneyulu-Veeri/publication/385698951_MICRO-FRONTEND_ARCHITECTURE_WITH_REACT_A_COMPREHENSIVE_GUIDE/links/673134a1ecbbde716b66256e/MICRO-FRONTEND-ARCHITECTURE-WITH-REACT-A-COMPREHENSIVE-GUIDE.pdf?__cf_chl_tk=7FobqrHRC3uVy7YgNNrgtdT1dmnmXVHkI5YAGz6bXOo-1732965391-1.0.1.1-i5dN7SmsXCaTndk.aRZW4sjXtTx7kgMfv95yB_wk4FQ
P265	Scholar	30/11/2024 07:18:00	Migration Process from Monolithic to Micro Frontend Architecture in Mobile Applications	https://ceur-ws.org/Vol-3627/paper05.pdf
P266	Scholar	30/11/2024 07:18:00	Design Systems for Micro Frontends - An Investigation into the Development of Framework-Agnostic Design Systems using Svelte and Tailwind CSS	https://epb.bibl.th-koeln.de/frontdoor/index/index/docId/1666
P267	Scholar	30/11/2024 07:19:00	MDEPT: Microservices Design Evaluator and Performance Tester	https://link.springer.com/chapter/10.1007/978-3-031-70797-1_9
P268	Scholar	30/11/2024 07:19:00	Microservice API Pattern Detection : Using Business Processes and Call Graphs	https://trepo.tuni.fi/handle/10024/143568
P269	Scholar	30/11/2024 07:20:00	A Survey on Microservices Architecture: Principles, Patterns and Migration Challenges	https://ieeexplore.ieee.org/abstract/document/10220070
P270	Scholar	30/11/2024 07:21:00	Assessing the feasibility of Micro frontend architecture in native mobile app development	https://dl.acm.org/doi/abs/10.1145/3691620.3695313
P271	Scholar	30/11/2024 07:21:00	On Microservice Analysis and Architecture Evolution: A Systematic Mapping Study	https://www.mdpi.com/2076-3417/11/17/7856
P272	Scholar	30/11/2024 07:21:00	Survey on Tools and Techniques Detecting Microservice API Patterns	https://ieeexplore.ieee.org/abstract/document/9860229
P273	Scholar	30/11/2024 07:22:00	Development of an e-portfolio social network using emerging web technologies	https://core.ac.uk/download/pdf/588865803.pdf
P274	Scholar	30/11/2024 07:24:00	Design Systems for Micro Frontends An Investigation into the Development of Framework-Agnostic Design Systems using Svelte and Tailwind CSS	https://epb.bibl.th-koeln.de/frontdoor/deliver/index/docId/1666/file/SCIBachelor_Design_Systems_and_Micro_Frontends.pdf
P275	Scholar	30/11/2024 07:24:00	Modern Software Architecture	https://www.theseus.fi/handle/10024/497600
P276	Scholar	30/11/2024 07:24:00	Modern Web Performance Patterns	https://link.springer.com/chapter/10.1007/978-1-4842-6528-4_10
P277	Scholar	30/11/2024 07:25:00	Introduction to the Special Issue on: Grey Literature and Multivocal Literature Reviews (MLRs) in software engineering	https://www.sciencedirect.com/science/article/abs/pii/S095058492100152X
P278	Scholar	30/11/2024 07:25:00	Table of Contents	https://www.computer.org/csdl/proceedings-article/icsa-c/2023/645900z005/1MBDfiW0KI0

P279	Scholar	30/11/2024 07:26:00	Cloud Native Development Patterns and Best Practives	https://books.google.com.br/books?hl=pt-BR&lr=&id=DJdMDwAAQBAJ&oi=fnd&pg=PP1&dq=(%E2%80%9Canti-pattern%E2%80%9D+OR+%E2%80%9Canti-patterns%E2%80%9D+OR+%E2%80%9Cantipattern%E2%80%9D+OR+%E2%80%9Cantipatterns%E2%80%9D+OR+%E2%80%9Canti+pattern%E2%80%9D+OR+%E2%80%9Canti+patterns%E2%80%9D)+AND+(%E2%80%9Cmicrofrontend%E2%80%9D+OR+%E2%80%9Cmicrofrontends%E2%80%9D+OR+%E2%80%9Cmicro+frontend%E2%80%9D+OR+%E2%80%9Cmicro+frontends%E2%80%9D+OR+%22micro-frontend%22+OR+%22micro-frontends%22)&ots=GD0CrphBZi&sig=9RXuSghZq_rTkhu2gB_frJISAZA&redir_esc=y#v=onepage&q=(%E2%80%9Canti-pattern%E2%80%9D%20OR%20%E2%80%9Canti-patterns%E2%80%9D%20OR%20%E2%80%9Cantipattern%E2%80%9D%20OR%20%E2%80%9Cantipatterns%E2%80%9D%20OR%20%E2%80%9Canti%20pattern%E2%80%9D%20OR%20%E2%80%9Canti%20patterns%E2%80%9D)%20AND%20(%E2%80%9Cmicrofrontend%E2%80%9D%20OR%20%E2%80%9Cmicrofrontends%E2%80%9D%20OR%20%E2%80%9Cmicro%20frontend%E2%80%9D%20OR%20%E2%80%9Cmicro%20frontends%E2%80%9D%20OR%20%22micro-frontend%22%20OR%20%22micro-frontends%22)&f=false
P280	Scholar	30/11/2024 07:27:00	Restructuring an Enterprise Monolith into a Microservices Architecture	https://www.theseus.fi/handle/10024/853531
P281	Scholar	30/11/2024 07:27:00	The Effects of Architectural Design Decisions on Framework Adoption: A Comparative Evaluation of Meta-Frameworks in Modern Web Development	https://aaltodoc.aalto.fi/items/23050444-1912-48a9-a96a-95816cb3550c
P282	Scholar	30/11/2024 07:27:00	MASTER THESIS Term paper submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering at the University of Applied Sciences Technikum Wien - Degree Program Software Engineering	https://epub.technikum-wien.at/obvftwhsmmig/content/titleinfo/9746870/full.pdf
P283	Scholar	30/11/2024 07:28:00	Applying Model-Driven Engineering to Stimulate the Adoption of DevOps Processes in Small and Medium-Sized Development Organizations	https://link.springer.com/article/10.1007/s42979-021-00825-z
P284	Scholar	30/11/2024 07:29:00	Comparison of Static Analysis Architecture Recovery Tools for Microservice Applications	https://arxiv.org/abs/2403.06941
P285	Scholar	30/11/2024 07:30:00	The Effects of Architectural Design Decisions on Framework Adoption: A Comparative Evaluation of Meta-Frameworks in Modern Web Development	https://www.researchgate.net/profile/Joel-Hassan-3/publication/380342898_The_Effects_of_Architectural_Design_Decisions_on_Framework_Adoption_A_Comparative_Evaluation_of_Meta-Frameworks_in_Modern_Web_Development/links/6636430c7091b94e93ef23d8/The-Effects-of-Architectural-Design-Decisions-on-Framework-Adoption-A-Comparative-Evaluation-of-Meta-Frameworks-in-Modern-Web-Development.pdf

P286	Scholar	30/11/2024 07:30:00	From monolithic systems to Microservices: An assessment framework	https://www.sciencedirect.com/science/article/pii/S0950584921000793
P287	Scholar	30/11/2024 07:31:00	Software Architecture Patterns: The Hard Parts	https://books.google.com.br/books?hl=pt-BR&lr=&id=OX1EEAAAQBAJ&oi=fnd&pg=PP1&dq=(%E2%80%9Canti-pattern%E2%80%9D+OR+%E2%80%9Canti-patterns%E2%80%9D+OR+%E2%80%9Cantipattern%E2%80%9D+OR+%E2%80%9Cantipatterns%E2%80%9D+OR+%E2%80%9Canti+pattern%E2%80%9D+OR+%E2%80%9Canti+patterns%E2%80%9D)+AND+(%E2%80%9Cmicrofrontend%E2%80%9D+OR+%E2%80%9Cmicrofrontends%E2%80%9D+OR+%E2%80%9Cmicro+frontend%E2%80%9D+OR+%E2%80%9Cmicro+frontends%E2%80%9D+OR+%22micro-frontend%22+OR+%22micro-frontends%22)&ots=eS6mTodiVQ&sig=a0eHTA2ClwTrjcByqhkm354lgzo&redir_esc=y#v=onepage&q&f=false
P288	Scholar	30/11/2024 07:32:00	Cloud Native Microsservices	https://books.google.com.br/books?hl=pt-BR&lr=&id=NbE2EAAAQBAJ&oi=fnd&pg=PT26&dq=(%E2%80%9Canti-pattern%E2%80%9D+OR+%E2%80%9Canti-patterns%E2%80%9D+OR+%E2%80%9Cantipattern%E2%80%9D+OR+%E2%80%9Cantipatterns%E2%80%9D+OR+%E2%80%9Canti+pattern%E2%80%9D+OR+%E2%80%9Canti+patterns%E2%80%9D)+AND+(%E2%80%9Cmicrofrontend%E2%80%9D+OR+%E2%80%9Cmicrofrontends%E2%80%9D+OR+%E2%80%9Cmicro+frontend%E2%80%9D+OR+%E2%80%9Cmicro+frontends%E2%80%9D+OR+%22micro-frontend%22+OR+%22micro-frontends%22)&ots=J1o3pIE4Ni&sig=Z7fDXCgAPtBlAgCfMsalGaiCxvQ&redir_esc=y#v=onepage&q&f=false
P289	Scholar	30/11/2024 07:33:00	Causal inference of server- and client-side code smells in web apps evolution	https://link.springer.com/article/10.1007/s10664-024-10478-0
P290	Scholar	30/11/2024 07:34:00	A longitudinal exploratory study on code smells in server side web applications	https://link.springer.com/article/10.1007/s11219-021-09567-w
P291	Scholar	30/11/2024 07:34:00	Mastering Angular Components:	https://books.google.com.br/books?hl=pt-BR&lr=&id=iJplDwAAQBAJ&oi=fnd&pg=PP1&dq=(%E2%80%9Canti-pattern%E2%80%9D+OR+%E2%80%9Canti-patterns%E2%80%9D+OR+%E2%80%9Cantipattern%E2%80%9D+OR+%E2%80%9Cantipatterns%E2%80%9D+OR+%E2%80%9Canti+pattern%E2%80%9D+OR+%E2%80%9Canti+patterns%E2%80%9D)+AND+(%E2%80%9Cmicrofrontend%E2%80%9D+OR+%E2%80%9Cmicrofrontends%E2%80%9D+OR+%E2%80%9Cmicro+frontend%E2%80%9D+OR+%E2%80%9Cmicro+frontends%E2%80%9D+OR+%22micro-frontend%22+OR+%22micro-frontends%22)&ots=kWV-HbMFja&sig=u9ThX7puLI0bJb62AtDw5jovI2A&redir_esc=y#v=onepage&q&f=false
P292	Scholar	30/11/2024 07:35:00	Design and Architecture	https://link.springer.com/chapter/10.1007/978-1-4842-9385-0_4
P293	Scholar	30/11/2024 07:35:00	Collaborative geovisual analytics	https://research.utwente.nl/en/publications/collaborative-geovisual-analytics

P294	Scholar	30/11/2024 07:36:00	Web systems quality evolution: A web smells approach	https://repositorio.iscte-iul.pt/handle/10071/29044
P295	Scholar	30/11/2024 07:36:00	Developing Senior Skills: A Diary-Based Exploration by a Junior Software Engineer	https://www.theseus.fi/handle/10024/858294
P296	Scholar	30/11/2024 07:37:00	Enabling Microservice Success	https://books.google.com.br/books?hl=pt-BR&lr=&id=qkD9EAAAQBAJ&oi=fnd&pg=PT21&dq=(%E2%80%9Canti-pattern%E2%80%9D+OR+%E2%80%9Canti-patterns%E2%80%9D+OR+%E2%80%9Cantipattern%E2%80%9D+OR+%E2%80%9Cantipatterns%E2%80%9D+OR+%E2%80%9Canti+pattern%E2%80%9D+OR+%E2%80%9Canti+patterns%E2%80%9D)+AND+(%E2%80%9Cmicrofrontend%E2%80%9D+OR+%E2%80%9Cmicrofrontends%E2%80%9D+OR+%E2%80%9Cmicro+frontend%E2%80%9D+OR+%E2%80%9Cmicro+frontends%E2%80%9D+OR+%22micro-frontend%22+OR+%22micro-frontends%22)&ots=8ekjOAYBM3&sig=FT0EkOpl7vghPBt-5o8oVS6XGzc&redir_esc=y#v=onepage&q&f=falsee
P297	Scholar	30/11/2024 07:37:00	Maintain multiple microservices without compromising the autonomy of Agile teams	https://www.diva-portal.org/smash/get/diva2:1681218/FULLTEXT01.pdf
P298	Scholar	30/11/2024 07:38:00	Building Evolutionary Architectures	https://books.google.com.br/books?hl=pt-BR&lr=&id=CGudEAAAQBAJ&oi=fnd&pg=PT8&dq=(%E2%80%9Canti-pattern%E2%80%9D+OR+%E2%80%9Canti-patterns%E2%80%9D+OR+%E2%80%9Cantipattern%E2%80%9D+OR+%E2%80%9Cantipatterns%E2%80%9D+OR+%E2%80%9Canti+pattern%E2%80%9D+OR+%E2%80%9Canti+patterns%E2%80%9D)+AND+(%E2%80%9Cmicrofrontend%E2%80%9D+OR+%E2%80%9Cmicrofrontends%E2%80%9D+OR+%E2%80%9Cmicro+frontend%E2%80%9D+OR+%E2%80%9Cmicro+frontends%E2%80%9D+OR+%22micro-frontend%22+OR+%22micro-frontends%22)&ots=IJEiPS30Vb&sig=6HoWOah7kZad7y2ARaQHbt_MgME&redir_esc=y#v=onepage&q=(%E2%80%9Canti-pattern%E2%80%9D%20OR%20%E2%80%9Canti-patterns%E2%80%9D%20OR%20%E2%80%9Cantipattern%E2%80%9D%20OR%20%E2%80%9Cantipatterns%E2%80%9D%20OR%20%E2%80%9Canti%20pattern%E2%80%9D%20OR%20%E2%80%9Canti%20patterns%E2%80%9D)%20AND%20(%E2%80%9Cmicrofrontend%E2%80%9D%20OR%20%E2%80%9Cmicrofrontends%E2%80%9D%20OR%20%E2%80%9Cmicro%20frontend%E2%80%9D%20OR%20%E2%80%9Cmicro%20frontends%E2%80%9D%20OR%20%22micro-frontend%22%20OR%20%22micro-frontends%22)&f=false
P299	Scholar	30/11/2024 07:41:00	Devops for Growth	https://www.scoop.it/topic/devox
P300	Scholar	30/11/2024 07:41:00	Interactive learning - A scalable and adaptive learning approach for large courses	https://mediatum.ub.tum.de/1621369
P301	Scholar	30/11/2024 07:42:00	Micro frontends numa aplicação de pré-contabilidade	https://recipp.ipp.pt/handle/10400.22/24267

P302	Scholar	30/11/2024 07:43:00	Mikroslužby pro získávání zdravotnických dat	https://dspace.jcu.cz/bitstream/handle/20.500.14390/44804/bp_belimenko.pdf?sequence=1
P303	Scholar	30/11/2024 07:43:00	Primjena i implementacija skalabilne mikroservisne arhitekture pomoću programskog okvira Spring Cloud	https://repozitorij.unizg.hr/islandora/object/foi:7501
P304	Scholar	30/11/2024 07:45:00	Problemas e dificuldades na migração de sistemas monolíticos para microsserviços	https://bccdev.ime.usp.br/tccs/2021/thiagocf/monografia.pdf
P305	Scholar	30/11/2024 07:46:00	Modellbasiertes Management von Anwendungen	https://elib.uni-stuttgart.de/handle/11682/13927
P306	Twitter	30/11/2024 07:53:00	The first paper of Nabson Silva's master's degree has been accepted to @ICSEconf 2025!	https://x.com/adolfont/status/1853843982368837853
P307	Twitter	30/11/2024 07:54:00	Luca Mezzalira @lucamezzalira · 30 de out 🧯 SAVE THE DATE! I'm super excited to share that I'll be hosting a 3-hour webinar on micro-frontends on January 2nd! 🎉	https://x.com/lucamezzalira
P308	Twitter	30/11/2024 07:55:00	Tomasz Ducin @tomasz_ducin · 26 de set ✍️ Blogged: What is #Frontend #Architecture? Enjoy the read 😄	https://x.com/tomasz_ducin
P309	Twitter	30/11/2024 07:55:00	Post Ver novos posts Conversa Nabson Silva @nabsonp The first paper of my master's, titled "A Catalog of Micro Frontends Anti-patterns", has been accepted by the @ICSEconf (Research Track) 2025 🎊	https://x.com/nabsonp/status/1853665471008043251
P310	Twitter	30/11/2024 07:56:00	Ahead of the Micro Frontends conference, curated by @Areai51 , tell us: Why Micro Frontends?	https://x.com/jsfoo/status/1467859022820753410
P311	Twitter	30/11/2024 07:56:00	We had an insightful interview with @al94781 about his experience at #GSAS24! 🎤 He shared valuable reflections on the event and its impact on the software architecture community. Check it out! https://ow.ly/xvMc50U9OW2	https://x.com/gsas_io

P312	Twitter	30/11/2024 07:57:00	Tayana Conte repostou Nabson Silva @nabsonp · 5 de nov The first paper of my master's, titled "A Catalog of Micro Frontends Anti-patterns", has been accepted by the @ICSEconf (Research Track) 2025 🎊	https://x.com/TayanaConte
P313	Twitter	30/11/2024 07:58:00	ReactFoo repostou JSFoo @jsfoo · 26 de jan de 2022 This evening, @rajasegar_c and @upen_dev_singh will talk about tooling, processes, and use cases of monorepos and multi-repos for managing large codebases. 9 pm. Details at https://hasgeek.com/jsfoo/microfrontends-conf/updates/link-for-joining-twitter-chat-on-managing-large-co-3BqfkNumn2EDYK7ceB8fyf Thank you @magicbell_io for sponsoring the session.	https://x.com/reactfoo?lang=bn
P314	Twitter	30/11/2024 07:58:00	Soraya Santana de la Fe repostou Global Software Architecture Summit @gsas_io · 15 de out 🚀 We're kicking off day 2 of #GSAS24 with an insightful talk by @lucamezzalira on Micro-Frontends Anti-Patterns! #SoftwareArchitecture	https://mobile.x.com/Sory_Santana
P315	Twitter	30/11/2024 07:59:00	Manfred Steyer @ManfredSteyer [Hot off the press] Free eBook: Modern Angular, 2nd extended version ⏬ Download now! ✓ Standalone ✓ Improved APIs ✓ Signals ✓ Control Flow ✓ Performance ✓ Automatic Migration	https://x.com/ManfredSteyer/status/1780526114135236784

P316	Twitter	30/11/2024 08:00:00	Akshay Agrawal @akshaykagrawal My co-founder @themylesfiles and I have started Marimo Inc. to keep building the @marimo_io notebook and other Python data tools. We've raised a $5M seed round led by @antgoldbloom and @shyammani_ at @aixventureshq . Excited for the journey ahead!	https://x.com/antgoldbloom
P317	Twitter	30/11/2024 08:01:00	Fixado ETHan @gliechtenstein · 19 de abr de 2018 No one's building a dedicated native mobile app for their @ethereum DApp. No one even talks about this topic much at all. In this post, I discuss the challenges, build one myself, and propose a solution. Love to hear feedback! https://hackernoon.com/the-future-of-native-mobile-apps-on-blockchain-what-they-should-look-like-and-how-to-build-one-d25024db07d5 #ethereum #dapp #blockchain	https://x.com/gliechtenstein
P318	Twitter	30/11/2024 08:02:00	Lokalimde Çalışıyor ¯_(ツ)_/¯ @lokalimde · 21 de jun Yarın TR saatiyle 13:00'da Recursivity ve sosyal + kognitif boyutlarını, time-travel ve diğer boyutlarını konuşuyoruz diye düşündüm. Meraklısına, beklerim 🎉 Ek olarak Serverless paradigmanın maliyetlerine gireceğim; bu da ek olsun 😎	https://mobile.x.com/lokalimde
P319	ACM	01/12/2024 20:29:00	Micro-Frontends: Principles, Implementations, and Pitfalls	https://dl.acm.org/doi/10.1145/3561846.3561853
P320	ACM	01/12/2024 20:30:00	FSE 2024: Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering	https://dl.acm.org/doi/proceedings/10.1145/3663529
P321	IEEE	01/12/2024 20:48	Software Architecture Patterns for Serverless Systems: Architecting for innovation with events, autonomous services, and micro frontends	https://ieeexplore.ieee.org/document/10162936/

F

MLR Duplicate Exclusion

This Appendix presents the duplicate publications excluded during the selection process of the MLR presented in Chapter 6.

ID	Duplicated
P12	EC5
P14	EC5
P20	EC5
P22	EC5
P27	EC5
P50	EC5
P54	EC5
P61	EC5
P63	EC5
P66	EC5
P70	EC5
P75	EC5
P82	EC5
P91	EC5
P99	EC5
P103	EC5
P107	EC5
P116	EC5
P123	EC5
P125	EC5
P127	EC5
P128	EC5
P129	EC5
P130	EC5
P131	EC5
P132	EC5
P137	EC5
P153	EC5
P256	EC5
P259	EC5
P261	EC5
P274	EC5
P285	EC5
P301	EC5
P306	EC5
P312	EC5
P319	EC5

G

MLR First Filter

This Appendix presents the first filter results of the MLR presented in Chapter 6.

ID	Title	Abstract	Occurrence	Final Result
P1	IC1			IC1
P2	IC1			IC1
P3	IC1			IC1
P4	IC1			IC1
P5	IC1			IC1
P6	IC1			IC1
P7	IC1			IC1
P8	IC1			IC1
P9	->	->	EC1	EC1
P10	IC1			IC1
P11	->	->	EC3	EC3
P13	->	->	EC1	EC1
P15	->	->	EC3	EC3
P16	IC1			IC1
P17	->	IC1		IC1
P18	->	->	EC1	EC1
P19	->	EC1		EC1
P21	->	->	EC1	EC1
P23	->	->	EC3	EC3
P24	->	->	IC1	IC1
P25	->	->	IC1	IC1
P26	IC1			IC1
P28	->	EC1		EC1
P29	->	->	IC1	IC1
P30	->	->	IC1	IC1
P31	->	->	EC1	EC1
P32	IC1			IC1
P33	->	->	EC1	EC1
P34	->	->	EC1	EC1
P35	->	->	EC1	EC1
P36	->	->	EC1	EC1
P37	->	->	EC1	EC1
P38	->	IC1		IC1
P39	IC1			IC1
P40	->	->	IC1	IC1
P41	->	->	IC1	IC1
P42	IC1			IC1
P44	IC1			IC1
P43	->	->	EC3	EC3
P45	->	->	EC1	EC1
P46	->	->	IC1	IC1
P47	->	->	EC1	EC1
P48	->	->	EC1	EC1
P49	->	->	EC1	EC1
P51	->	->	EC4	EC4

P52	->	->	EC1	EC1
P53	->	IC1		IC1
P55	->	IC1		IC1
P56	IC1			IC1
P57	->	->	EC1	EC1
P58	->	IC1		IC1
P68	->	->	EC1	EC1
P59	->	->	EC1	EC1
P60	->	->	EC1	EC1
P62	->	->	EC1	EC1
P64	->	->	EC3	EC3
P65	->	->	EC3	EC3
P67	->	->	EC3	EC3
P69	->	->	EC3	EC3
P71	->	->	EC3	EC3
P72	->	->	EC3	EC3
P73	->	IC1		IC1
P74	->	->	EC1	EC1
P76	IC1			IC1
P77	->	->	EC1	EC1
P78	->	->	EC1	EC1
P79	->	->	EC3	EC3
P80	->	->	EC1	EC1
P81	->	->	IC1	IC1
P83	->	->	EC1	EC1
P84	->	IC1		IC1
P85	IC1			IC1
P86	IC1			IC1
P87	IC1			IC1
P88	->	->	EC3	EC3
P89	->	->	EC3	EC3
P90	->	->	EC1	EC1
P92	->	->	EC1	EC1
P93	->	->	EC1	EC1
P94	->	->	EC1	EC1
P95	->	->	EC3	EC3
P96	->	->	EC1	EC1
P97	->	->	EC3	EC3
P98	->	->	EC3	EC3
P100	->	->	IC1	IC1
P102	->	->	EC1	EC1
P101	->	->	EC3	EC3
P104	->	->	EC1	EC1
P105	->	->	EC3	EC3
P106	->	->	EC1	EC1
P108	IC1			IC1

P109	->	IC1		IC1
P110	->	IC1		IC1
P111	->	->	EC1	EC1
P112	->	IC1		IC1
P121	->	->	EC1	EC1
P113	->	->	EC1	EC1
P114	->	->	EC1	EC1
P115	->	->	EC1	EC1
P117	->	->	EC1	EC1
P118	->	->	EC1	EC1
P119	->	->	EC1	EC1
P120	->	->	EC3	EC3
P122	->	->	EC3	EC3
P124	->	->	EC1	EC1
P126	->	->	EC1	EC1
P133	->	->	EC1	EC1
P134	->	->	EC1	EC1
P135	->	->	EC1	EC1
P136	->	IC1		IC1
P138	->	->	EC1	EC1
P139	->	->	EC1	EC1
P140	->	->	EC1	EC1
P141	->	->	EC1	EC1
P142	->	->	EC1	EC1
P143	->	IC1		IC1
P144	->	IC1		IC1
P145	->	->	EC1	EC1
P146	->	->	EC1	EC1
P147	IC1			IC1
P148	IC1			IC1
P149	->	->	EC3	EC3
P150	->	->	EC3	EC3
P151	->	->	IC1	IC1
P152	->	->	IC1	IC1
P154	->	IC1		IC1
P155	->	IC1		IC1
P156	->	->	EC3	EC3
P157	->	IC1		IC1
P158	->	->	EC3	EC3
P159	->	->	EC1	EC1
P160	->	->	EC3	EC3
P161	->	IC1		IC1
P162	->	->	EC3	EC3
P163	->	->	IC1	IC1
P164	->	->	IC1	IC1
P165	->	->	EC3	EC3

P166	->	->	EC1	EC1
P167	->	->	IC1	IC1
P168	->	->	EC3	EC3
P169	->	->	EC1	EC1
P170	->	->	EC1	EC1
P171	->	->	EC3	EC3
P172	->	->	EC3	EC3
P173	->	->	EC3	EC3
P174	->	->	EC3	EC3
P175	->	->	EC3	EC3
P176	->	IC1		IC1
P177	->	->	EC3	EC3
P178	->	->	EC3	EC3
P179	->	->	EC3	EC3
P180	->	->	IC1	IC1
P181	->	->	EC3	EC3
P182	->	->	EC3	EC3
P183	->	->	EC3	EC3
P184	->	->	EC3	EC3
P185	->	->	EC3	EC3
P186	->	->	EC3	EC3
P187	->	->	EC3	EC3
P188	->	->	EC3	EC3
P189	->	->	EC3	EC3
P190	->	->	EC3	EC3
P191	->	IC1		IC1
P192	->	->	EC3	EC3
P193	->	->	EC3	EC3
P194	->	->	EC3	EC3
P195	->	->	EC3	EC3
P196	->	->	EC3	EC3
P197	->	->	EC3	EC3
P198	->	->	EC3	EC3
P199	->	->	EC3	EC3
P200	->	->	EC3	EC3
P201	->	->	EC3	EC3
P202	->	->	EC3	EC3
P203	->	->	EC3	EC3
P204	->	->	EC3	EC3
P205	->	->	EC3	EC3
P206	->	->	EC3	EC3
P207	->	->	EC3	EC3
P208	->	->	IC1	IC1
P209	->	->	EC3	EC3
P210	->	->	EC3	EC3
P211	->	->	EC2	EC2

P212	->	->	EC3	EC3
P213	->	->	EC3	EC3
P214	->	->	EC3	EC3
P215	->	->	EC3	EC3
P216	->	->	EC3	EC3
P217	->	->	EC3	EC3
P218	->	->	EC3	EC3
P219	->	->	EC3	EC3
P220	->	->	EC3	EC3
P221	->	->	EC3	EC3
P222	->	->	EC3	EC3
P223	->	->	EC3	EC3
P224	->	->	EC3	EC3
P225	->	->	EC3	EC3
P226	->	->	EC3	EC3
P227	->	->	EC3	EC3
P228	->	->	EC3	EC3
P229	->	->	EC3	EC3
P230	->	->	EC3	EC3
P231	->	->	EC3	EC3
P232	->	->	EC2	EC2
P233	->	->	EC3	EC3
P234	->	->	EC3	EC3
P235	->	->	EC3	EC3
P236	->	->	EC3	EC3
P237	->	->	EC3	EC3
P238	->	->	EC3	EC3
P239	->	->	EC3	EC3
P240	->	->	EC3	EC3
P241	->	->	EC3	EC3
P242	->	->	EC3	EC3
P243	->	->	EC3	EC3
P244	->	->	EC3	EC3
P245	->	->	EC3	EC3
P246	->	->	EC3	EC3
P247	->	->	EC3	EC3
P248	->	->	EC3	EC3
P249	->	->	EC3	EC3
P250	->	->	EC2	EC2
P251	->	->	EC2	EC2
P252	->	->	EC3	EC3
P253	->	->	EC3	EC3
P254	->	->	EC3	EC3
P255	IC1			IC1
P257	->	EC3		EC3
P258	->	EC2		EC2

P260	->	EC1		EC1
P262	->	EC3		EC3
P263	->	EC3		EC3
P264	->	EC1		EC1
P265	->	EC1		EC1
P266	->	EC1		EC1
P267	->	EC3		EC3
P268	->	EC3		EC3
P269	->	EC3		EC3
P270	->	EC1		EC1
P271	->	EC3		EC3
P272	->	EC3		EC3
P273	->	EC3		EC3
P275	->	EC3		EC3
P276	->	EC3		EC3
P277	->	EC3		EC3
P278	->	EC3		EC3
P279	->	EC3		EC3
P280	->	EC3		EC3
P281	->	EC3		EC3
P282	->	EC3		EC3
P283	->	EC3		EC3
P284	->	EC3		EC3
P286	->	EC3		EC3
P287	->	EC3		EC3
P288	->	EC3		EC3
P289	->	EC3		EC3
P290	->	EC3		EC3
P291	->	EC3		EC3
P292	->	EC3		EC3
P293	->	EC3		EC3
P294	->	EC3		EC3
P295	->	EC3		EC3
P296	->	EC3		EC3
P297	->	EC3		EC3
P298	->	EC3		EC3
P299	->	EC3		EC3
P300	->	EC3		EC3
P302	->	EC2		EC2
P303	->	EC3		EC3
P304	->	EC3		EC3
P305	->	EC3		EC3
P307	->	->	EC1	EC1
P308	->	->	EC3	EC3
P309	IC1			IC1
P310	->	->	EC1	EC1

P311	->	->	EC1	EC1
P313	->	->	EC1	EC1
P314	->	->	EC1	EC1
P315	->	->	EC1	EC1
P316	->	->	EC1	EC1
P317	->	->	EC1	EC1
P318	->	->	EC2	EC2
P320	->	EC1		EC1
P321	->	EC1		EC1

H

MLR Quality Assessment

This Appendix presents the quality assessment results of the MLR presented in Chapter 6.

	Date	Authority of the producer		Methodology	Impact	Position w.r.t. related sources	Outlet type		
ID	Does the post have a clearly stated date?	Has the author published other posts about Micro Frontends?	Does the author have experience in real-world Micro Frontend architectures?	Does the anti-patterns have a clearly A origin or reference?	Does the post have positive commnets, likes or have more than 3 backlinks?	Have key related GL or formal sources linked to or discussed?	1st tier GL = 1 2nd tier GL = 0.5 3rd tier GL = 0	Score	Result
P1	1	1	1	1	1	0	0	0,71	Accepted
P2	1	1	1	1	1	0	0.5	0,71	Accepted
P3	0	1	1	0	0	0	0.5	0,29	Rejected
P4	1	0	0	0	0	0	0	0,14	Rejected
P5	1	1	1	1	1	0	0.5	0,71	Accepted
P6	1	0	0	0	1	0	0	0,29	Rejected
P7	1	1	1	1	0	0	0.5	0,57	Accepted
P8	1	1	1	0	1	1	0	0,71	Accepted
P10	1	1	1	1	1	0	0.5	0,71	Accepted
P16	1	1	1	1	1	0	1	0,86	Accepted
P17	0	0	0	1	0	1	0	0,29	Rejected
P24	1	1	1	1	0	0	0.5	0,57	Accepted
P25	1	0	1	0	0	0	0	0,29	Rejected
P26	1	1	1	1	0	0	0.5	0,57	Accepted
P29	1	1	1	1	0	1	0.5	0,71	Accepted
P30	1	1	1	0	0	0	1	0,57	Accepted
P32	1	1	1	1	1	0	0.5	0,71	Accepted
P38	1	1	1	1	1	1	0	0,86	Accepted
P39	1	1	1	1	1	0	0.5	0,71	Accepted
P40	1	0	0	0	0	1	1	0,43	Rejected
P41	1	1	1	0	0	1	1	0,71	Accepted
P42	1	1	1	1	1	0	0.5	0,71	Accepted
P44	1	1	1	1	1	0	0.5	0,71	Accepted
P46	0	0	0	1	0	0	0	0,14	Rejected
P55	1	1	1	1	0	0	0	0,57	Accepted
P56	0	1	1	1	0	0	0.5	0,43	Rejected
P58	1	0	0	1	0	1	1	0,57	Accepted
P73	0	1	0	0	0	0	0	0,14	Rejected
P76	0	1	1	1	0	0	0	0,43	Rejected
P81	1	1	1	1	0	0	0.5	0,57	Accepted
P84	0	1	1	1	0	0	0	0,43	Rejected
P85	0	0	0	0	0	0	0	0,00	Rejected
P86	1	1	1	1	0	0	0.5	0,57	Accepted
P87	1	1	1	1	1	0	0.5	0,71	Accepted
P100	1	1	1	1	1	0	0	0,71	Accepted
P108	1	1	1	1	1	1	0	0,86	Accepted
P109	1	1	1	1	0	1	0.5	0,71	Accepted
P112	1	1	1	1	0	0	0.5	0,57	Accepted
P136	1	1	1	1	0	0	0.5	0,57	Accepted
P143	1	1	1	1	1	1	0.5	0,86	Accepted
P144	1	1	1	1	1	1	0.5	0,86	Accepted
P147	1	1	1	1	0	0	0.5	0,57	Accepted
P148	1	0	0	0	1	0	0.5	0,29	Rejected
P151	1	0	0	0	1	0	0	0,29	Rejected
P152	1	0	0	0	1	1	0	0,43	Rejected
P154	1	0	1	1	1	0	0	0,57	Accepted
P155	1	0	1	1	1	0	0	0,57	Accepted
P157	1	1	1	1	1	1	0	0,86	Accepted
P161	1	1	1	1	1	0	0	0,71	Accepted
P163	1	0	1	1	1	0	0	0,57	Accepted
P164	1	0	0	0	1	0	0	0,29	Rejected
P167	1	0	0	0	1	0	0	0,29	Rejected
P176	1	1	1	1	1	0	0	0,71	Accepted
P180	1	0	0	0	1	0	0	0,29	Rejected
P191	1	0	0	0	1	0	0	0,29	Rejected
P208	1	0	0	0	1	0	0	0,29	Rejected

I

MLR Second Filter

This Appendix presents the second filter results of the MLR presented in Chapter 6.

ID	Result
P1	IC1
P2	IC1
P5	EC5
P7	IC1
P8	IC1
P10	EC5
P16	EC1
P24	EC1
P26	EC5
P29	EC5
P30	IC1
P32	EC4
P38	IC1
P39	EC5
P41	IC1
P42	EC5
P44	EC5
P53	EC1
P55	IC1
P58	IC1
P81	EC1
P86	EC2
P87	EC1
P100	EC1
P108	IC1
P109	EC1
P110	EC1
P112	EC5
P136	EC1
P143	EC1
P144	EC1
P147	EC5
P154	EC1
P155	IC1
P157	EC5
P161	EC1
P163	EC1
P176	EC3
P255	EC1
P309	IC1

J

MLR Data Extraction

This Appendix provides the complete data extraction from the publications selected after de selection process from the MLR presented in Chapter 6.

P1

Title	Top 10 Micro Frontend Anti-Patterns
Source	Google
Author	Florian Rappl
Publication type	Blog
Publication date	2024/03/20
Author's profile	Practitioner
Anti-patterns origin	Professional experience.
Anti-patterns categories	-
Proposed Anti-patterns	**Hidden Monolith** Problem Source: ● "*[...] you cannot just publish or rollback a micro frontend you will most likely have created a hidden monolith.*" Proposed Solution: ● "*Use DDD to properly split your application and keep loose coupling to avoid dependencies between the micro frontends*"
	Chatty Frontends Problem Source: ● "*[...] over-communication in form of emitting events for pretty much every action results in an inefficient chatter between the different UI fragments.*" Proposed Solution: ● "*Only emit useful events, potentially introducing events only if there are interested parties*"
	Framework Madness Problem Source: ● "*[...] one of the most used reasons for advocating or introducing micro frontends historically has been the ability to render components from multiple frameworks. While this feature is certainly "cool", it is not exclusive to micro frontends nor is it a feature that should easily be abused.*" Proposed Solution: ● "*Try to settle on a single technology and only introduce support for other frameworks - only to be introduced in justified cases.*"

	Micro Everything Problem Source: ● "*[...] open a new sub-domain with a new micro frontend when you encounter a feature that does not easily fall into the existing sub-domains. [...] it will easily lead to tons of really little micro frontends being created.*" Proposed Solution: ● "*Don't split up too early, start in the closest possible domain / micro frontend and only extract once a clear sub-domain has emerged.*"
	Violating Single Responsibility Problem Source: ● "*[...] a single micro frontend takes on multiple responsibilities or concerns that should ideally be separated.*" Proposed Solution: ● "*[...] the term "micro" does not imply a certain lines of code limit or so, but rather a focus on a single sub-domain of your application.*"
	Spaghetti Architecture Problem Source: ● "*[...] software architecture that lacks clear structure and organization, resulting in a tangled mess of interconnected components and modules.*" Proposed Solution: ● "*Stay at loose coupling without relying on a web of calls to provide a feature.*"
	Distributed Data Inconsistency Problem Source: ● "*If the original data changes further changes need to be propagated, but this is not the only possible change. Another possibility is that the micro frontend that got a replica of the data needs the changes. Is that even allowed? Now the original and the duplicate will deviate again.*" Proposed Solution: ● "*Keep data where it belongs; let others not access the data directly, but only indirectly via attributes / props etc.*"
	Dismissing Human Factors Problem Source: ● "*[...] meeting technical objectives and deadlines is done without considering the impact on the well-being, morale, and work-life balance of the team members, [...] you should consider micro frontends as a pattern to improve the team organization and structure - not to make the team suffer.*" Proposed Solution: ● "*Strengthen teams and avoid central management as much as possible.*"

	Avoiding Observability Problem Source: ● "*[...] there is no observability implemented anywhere, e.g., if you don't know what micro frontend is the originator of an error or if you cannot debug the problematic part locally.*" Proposed Solution: ● "*In case something bad happens you want to see some logs to have at least kind of a trace where to start looking. [...] Using the right metrics we can at least establish Sourcelines and go from there. [...] Introduce central solutions to simplify logging, collecting traces, etc.*"
	Tight Coupling Problem Source: ● "*[...] tight coupling is usually the origin of many of the problems that arise when scaling micro frontends.*" Proposed Solution: ● "*Avoid direct references that require any technical knowledge (URLs, module paths, internal names, ...) of other micro frontends.*"

P2

Title	Microfrontends Anti-Patterns: Seven Years in the Trenches
Source	Google
Author	Luca Mezzalira
Publication type	Lecture
Publication date	2022/05/10-20
Author's profile	Practitioner
Anti-patterns origin	Professional experience.
Anti-patterns categories	-
Proposed Anti-patterns	**Yin and Yang (Micro-Frontends and Components)** Problem Source: ● "*When you start to have implementation of micro-frontends that contains a very granular approach where you have the header, the footer, you have the image that are all micro-frontends, probably they're not micro-frontends.*" Proposed Solution: ● "*[...] using multiple UI frameworks in the same single page application could cause more than one problem. That is also true with micro-frontends,*"
	Hydra of Lerna - (Multi-Frameworks Approach) Problem Source: ● ."*[...] using multiple UI frameworks in the same single page application could cause more than one problem. That is also true with micro-frontends,*" Proposed Solution: ● "*[...] there are certain situations where a multi-framework approach makes sense. For instance, when you're dealing with legacy systems. [...] Another solution is when we are migrating from a UI framework to another one.*"

	The Swiss Army Knife - (Write Programs That Do One Thing and Do it Well) Problem Source: ● "*[...] you have this legacy editor that is written with a different technology and maybe has a different way to communicate with the external world that doesn't fit inside your architecture.*" Proposed Solution: ● "*You need to maybe create what is called an anticorruption layer between your application, so your micro-frontends application and the legacy editor.*"
	The Dependencies Hell - (Do you Really Need that External Dependency?) Problem Source: ● "*We started to work with micro-frontends, and we see a lot of parts that could be wrapped in what is called the core library. [...] The team responsible for micro-frontend B are creating the core library extended. That is basically taking some part of the core library, rewriting others. Now we have a fork, or an extension of the core library that is breaking some stuff. [...] What can you do with the core library extended? You start to diverge. [...] in the long run, you will see that the core library extended, suddenly, it's a different library.* " Proposed Solution: ● "*[...] understand if you really need these kinds of core libraries. [...] Having a library that you share is great, but then try to work with* ***composition more than extension****, and also try to keep very separated certain libraries, and not do a library for everything.*"
	A Return Ticket, Please - (Unidirectional Data Flow at the Rescue) Problem Source: ● "*[...] where you have a bidirectional sharing or communication, it can be complicated, especially because we are sharing across multiple elements, and everyone can share everything to everyone. It is going to be a nightmare."* Proposed Solution: ● "*What we have learned in the past 15 years in the frontend communities [...] is the concept of unidirectional data flow [...] I know exactly when I dispatch an action, update the store, and update the view. [...] Same thing was introduced in the Model View Intent architecture, or MVI, [...] where you have a user that is triggering an action. Then we capture the action in the intent, we change the model or we pass the information that is needed for changing. Then we load the new state and the view is updating in the UI.*"

	Relax, It's Just Code - (Avoid Organizational Coupling) Problem Source: ● "*When someone is changing the global state [...] the structure, the signature, whatever, it means that everyone has to be aware. You need to coordinate across the teams. Basically, you're losing the benefit of micro-frontends, so it is independence that you are looking for.*" Proposed Solution: ● "*A way to solve it is using instead events, or a publish and subscribe pattern. [...] Everything that could work in a Pub/Sub pattern is definitely your friend in this case.*"
	Let's Hammer the APIs - (Multiple MFEs Calling the Same Endpoint) Problem Source: ● "*There are situations where you have multiple micro-frontends calling the same endpoints. [...] Now the authorization service has to scale for twice the traffic because there are two calls every time. [...] If there are huge numbers, it is there that there are some problems. [...] You need to scale the entire flow.*" Proposed Solution: ● "*Do you really need to have two micro-frontends, or you can have one micro-frontend only that is assigned to one team and is handling one request per time? [...] domain. Otherwise, probably you need to go back to the whiteboard and review how your APIs are working."* ● *"The other option is creating a container that is handling one request and having two components instead of micro-frontends that are [...] relying on the container that is handling the requests.*"

P7

Title	Micro-Frontends anti-patterns by Luca Mezzalira (#GSAS24)
Source	Google
Author	Luca Mezzalira
Publication type	Vídeo
Publication date	2024/11/19
Author's profile	Practitioner
Anti-patterns origin	Professional Experience
Anti-patterns categories	-
Proposed Anti-patterns	**Bye Bye big-bang - Iterative Deployment** Problem Source: ● "*[...] a portion of the UI and carve It Out, Create a micro frontend for that and put inside the old system that usually slow down everything because you don't have a real benefit of doing micro frontends [...]*" Proposed Solution: ● "*[...] there is a better method so you can start to use migration strategies by path [...] you start to build your different micro frontend and migrate portions of the page slowly but steadily [...]*"

P8

Title	4 Micro-Frontend Anti-Patterns
Source	Google
Author	Santosh Shinde
Publication type	Blog
Publication date	2022/07/29
Author's profile	Practitioner
Anti-patterns origin	Professional Experience
Anti-patterns categories	-
Proposed Anti-patterns	**Micro-Frontend Vs Component** Problem Source: ● "*Micro-frontends are the technical representation of a business subdomain, [...] However, we frequently mix components and micro-frontends because we are unsure of their differences, which will be our first anti-pattern.*" Proposed Solution: ● "*So set the boundaries of the micro-frontends based on the business subdomains [...] and avoid using components as micro-frontends because, by definition, micro-frontends represent business domains.*"
	Multi-Framework Approach Problem Source: ● "*As micro-frontend gives us the freedom to choose any framework for development, we must first determine whether it is truly required.*" Proposed Solution: ● "*[...] if you have the option to choose one framework for all the micro-frontends, then the best option is to use a single framework.*"
	Dependency Hell Problem Source: ● "*The problem, however, is that a dependency also depends on another dependency, and this causes us to have problems with backward and forward compatibility.*" Proposed Solution: ● "*[...] decouple your libraries from any feature extensions that they support by using wrappers that do not conflict with the functionality of the core libraries. Also, make sure that all micro-frontends are using the same version of your libraries.*"

	Global State Communication Problem Source: ● "*[...] do not use the shared state because it would violate the concept of segregation.*" Proposed Solution: ● "*The common approach is for each "micro frontend" to have its own store (i.e. Redux). To address this issue, you could use an event emitter-based approach to communication."*

P30

Title	Chapter 4. Discovering Micro-Frontend Architectures
Source	Google
Author	Luca Mezzalira
Publication type	Book
Publication date	2021/11/28
Author's profile	Practitioner
Anti-patterns origin	Professional Experience
Anti-patterns categories	-
Proposed Anti-patterns	**Sharing state across MFEs** Problem Source: ● "*[...] when you select a horizontal split, you have to avoid sharing any state across micro-frontends; this approach is an antipattern. [...] Many developers may be tempted to share states between micro-frontends, but this results in a socio-technical antipattern.*" Proposed Solution: ● "*[...] use the techniques mentioned in Chapter 3, such as an event emitter, custom events, or reactive streams using an implementation of the publish/subscribe (pub/sub) pattern for decoupling the micro-frontends and maintaining their independent nature. [...] uses an asynchronous communication, or event broker, to notify all the consumers interested in a specific event. [...] Other possibilities are implementing either an event emitter or a reactive stream [...]*"
	Several MFEs in the same view Problem Source: ● "*[...] there is a real risk of over-engineering the solution to have several tiny micro-frontends living together in the same view, which creates an antipattern.*" Proposed Solution: ● "*[...] reduce the number of micro-frontends in the same view, especially when multiple teams have to merge their work together.*"

P38

Title	Rules of Micro-Frontends
Source	Google
Author	Ruben Casas
Publication type	Blog
Publication date	2020/12/29
Author's profile	Practitioner
Anti-patterns origin	Professional Experience
Anti-patterns categories	-
Proposed Anti-patterns	**The deployment queue of hell** Problem Source: ● "*A common antipattern that should be avoided is 'The deployment queue of hell' where different Micro-frontends are so tightly coupled that they need to be deployed in a specific order to avoid breaking the application.*" Proposed Solution: ● "*Each Micro-Frontend should have it's own CI / CD pipeline and be able to deploy to production on demand without any dependencies on other Micro-frontends.*"

P41

Title	Understanding and implementing microfrontends on AWS - AWS Prescriptive Guidance
Source	Google
Author	Matteo Figus, Principal Solutions Architect, AWS; Alexander Guensche, Senior Solutions Architect, AWS; Harun Hasdal, Senior Solutions Architect, AWS; Luca Mezzalira, Principal Go to Market Specialist Solutions Architect Serverless UK, AWS
Publication type	Technical Document
Publication date	2024/06
Author's profile	Practitioner
Anti-patterns origin	Professional Experience
Anti-patterns categories	-
Proposed Anti-patterns	**Poor Dependency Management in Micro-Frontends** Problem Source: ● "*Ideally, your organization needs to mandate how dependencies in a distributed frontend architecture are maintained.*" Proposed Solution: ● "*Three viable strategies for mandating dependency maintenance are share nothing, using web standards such as import maps, and module federation. Other approaches are anti-patterns because they violate basic principles of distributed architectures.*"
	Dependent Deploy Problem Source: ● "*Consider the example of teams working on a micro-frontend feature that will be launched on a specific date. The feature is ready, but it needs to be released together with a change on another micro-frontend that is independently released. Blocking the release of both micro-frontends would be considered an anti-pattern and would increase risk when deployed.*" Proposed Solution: ● "*[...] the teams can create a Boolean feature flag in a database that they both consume during render time (perhaps through an HTTP call to a shared Feature Flags API).*"

P55

Title	TechLead Journal: #47 - Micro-Frontends and the Socio-Technical Aspect - Luca Mezzalira
Source	Google
Author	Luca Mezzalira
Publication type	Podcast
Publication date	2021/07/19
Author's profile	Practitioner
Anti-patterns origin	Professional Experience
Anti-patterns categories	-
Proposed Anti-patterns	**Global state** Problem Source: ● "*Many developers when they have multiple micro-frontends in the same view, and they need to communicate together, the first thing that they think about is having a global state. And that is definitely an anti-pattern.*" Proposed Solution: ● "*Instead, if you keep the state inside the micro-frontend, and you implement a Pub/Sub pattern where the communication is happening through an event emitter, or a custom event or a signal library or reactive stream.*"
	Multiple technologies in the same application Problem Source: ● "*[...] having multiple technologies in the same application.*" Proposed Solution: ● "*My suggestion is to define what the boundaries are. So the team can work with React 16, or React 17, React 18, Angular, whatever version. It doesn't matter. But try to stick with the same theme. If there is a situation where having for a certain period of time, multiple frameworks can be helpful.*"

P58

Title	Compositional Qualities of Microfrontends: The LdoD Archive
Source	Google
Author	João Luís Pacheco Raimundo
Publication type	Master Thesis
Publication date	2023/05
Author's profile	Researcher
Anti-patterns origin	Professional Experience
Anti-patterns categories	-
Proposed Anti-patterns	**MFEs versus Components** Problem Source: ● *"At first glance, the concepts of Component and MFEs may seem indistinguishable. However, while a component may be designed as an independent MFEs or as a simple extensible component whose behavior is highly affected by the environment, indicating that it is not independent [...]"* Proposed Solution: ● "*[...] an MFEs should always be a technical representation of a sub-domain designed to be self-contained, allowing for independent implementation with the same or different technology.*"
	Multi-frameworks approach Problem Source: ● "Considered an anti-pattern by some, and an advantage by others, is the approach of implementing MFEs with different web frameworks." Proposed Solution: ● "*Nevertheless, this approach could be advantageous in specific scenarios such as integrating legacy applications or applications developed with different technologies, where iterative integration can be done, allowing different technologies to coexist harmoniously.*"

	Integration Bottleneck Anti-Pattern Problem Source: ● "*To integrate a legacy or new application that uses different technology and communication protocols, there may be a tendency to extend the behavior of the main application integration layer to fit the new requirement.*" Proposed Solution: ● "*One effective solution is to extend the MSs pattern of the Anti-corruption layer, which is a logical layer that translates between two different DMs to prevent concepts from one model polluting another [23]. This can be applied to MFEs to enable appropriate translation between application interactions, ensuring no modifications in the main application integration layer.*"
	Dependency Hell: Problem Source: ● "*Using shared libraries can be problematic due to versioning issues that can introduce breaking changes. The problem becomes even more significant when there is a tendency to extend shared packages in order to achieve a more specific behavior for a particular MFE.*" Proposed Solution: ● "An effective solution is to work with smaller abstractions and prioritize composition over extension."
	Shared Global State: Problem Source: ● "*Using shared global state for MFEs communication purposes can lead to stronger coupling between MFEs.*" Proposed Solution: ● "*[...] using a publish-subscribe pattern such as event listeners or event emitters allows for greater independence of each MFE, enabling more efficient development and maintenance of the system.*"

P108

Title	Micro-frontend “Blackbox Pattern”
Source	Google
Author	Naim Gkamperlo
Publication type	Blog
Publication date	2020/05/04
Author’s profile	Practitioner
Anti-patterns origin	Professional Experience
Anti-patterns categories	-
Proposed Anti-patterns	**<No Name>** Problem Source: ● “*[...] there is usually a need to create a micro-frontend which can utilised by all the other micro-frontends. However, this is considered an anti-pattern [...]*” Proposed Solution: ● “*[...] a blackbox (hence the name of the pattern) which has an input, renders itself in the DOM and has its own workflow and produces an output for every other micro-frontend to get its results.*”

P155

Title	Micro Front-End Architecture at Enterprise Scale (Updated July 2020)
Source	Google
Author	Bernd Wessels
Publication type	Blog
Publication date	2020/10/07
Author's profile	Practitioner
Anti-patterns origin	Professional Experience
Anti-patterns categories	-
Proposed Anti-patterns	**<No Name>** Problem Source: ● "*Sometimes it might be tempting to align APIs with individual business solutions and micro front-ends. This is an anti-pattern [...] A naïve approach would be to let micro front-ends directly access multiple business domain APIs. This is an anti-pattern which eventually leads into a huge mess of unmanageable dependencies, prevents deprecation and causes massive over-fetching and cache synchronization issues.*" Proposed Solution: ● "*This is where the principle of API federation shines. The federated API is a controlled way of providing information from several domain APIs in a single strongly typed place. [...] The principles of being business domain driven, service oriented and technology agnostic all clearly point to GraphQL as being an actual standard on the web platform specifically designed for domain driven environments. [...] Micro front-ends can now query, mutate and subscribe the local state in exactly the same way they do for the backend. Improving simplicity and reducing cognitive load even further [...] With GraphQL we can now combine all these domains into a single federated API that can be consumed by all micro front-ends.*"

P309

Title	A Catalog of Micro Frontends Anti–patterns
Source	Google
Author	Nabson Silva, Eriky Rodrigues, Tayana Conte
Publication type	Conference Paper
Publication date	2024/11/29
Author's profile	Practitioner
Anti-patterns origin	Professional Experience
Anti-patterns categories	Intra-frontend, Inter-frontend, Operations and Development
Proposed Anti-patterns	**Cyclic Dependency** Problem Source: • "*Two or more MFEs directly or indirectly depend on each other, resulting in high coupling between screens and fragments, compromising MFEs' independence and modularity*" Proposed Solution: • "*High coupling between MFEs can be effectively mitigated through event-based communication [...] On implementing the Publish-Subscribe (Pub-Sub) pattern, an MFE can publish an event to the browser, [...]*"
	Knot Micro Frontend Problem Source: • "*A Knot is composed of three or more MFEs whose communication with each other uses a context-specific interface.*" Proposed Solution: • "*[...] implement domain-driven communication interfaces that are both generic and flexible. [...] specifying the essential fields required for each MFE to function correctly and interact with others. [...] We recommend including a generic field in the interface containing a list of objects*"
	Hub-like Dependency Problem Source: • "*A screen of an MFE integrates fragments from several other MFEs, becoming a central point of interdependence.*" Proposed Solution: • "*[...] each fragment should implement robust error handling mechanisms*"

	Nano Frontend Problem Source: ● "*The frontend decomposes into numerous small MFEs with few screens or fragments*" Proposed Solution: ● "*Adhering to Domain-driven Design [33] principles is necessary to ensure an effective decomposition of MFEs.*"
	Mega Frontend Problem Source: ● "*Decomposing the architecture into a few MFEs encompassing numerous screens and fragments manifests this anti-pattern.*" Proposed Solution: ● "*[...] development team must work closely with the product team to gain a deep understanding of the domains and reflect them accurately in the architecture.*"
	Micro Frontend Greedy Problem Source: ● "*Whenever a need arises to develop a new set of screens or fragments, a new MFE is instantiated.*" Proposed Solution: ● "*[...] the domain of the new feature must first be defined. If it falls within the domain of an existing MFE, it should be implemented there. [...] a summary of all MFEs, their contexts, and domains can help identify the best fit for the new feature. If it belongs to a brand new domain, one or more MFEs should be defined based on the domain definition.*"
	No CI/CD Problem Source: ● "*The company lacks an automated Continuous Integration (CI) and Continuous Delivery (CD) pipeline [...]*" Proposed Solution: ● "*Implement an automated and replicable CI/CD process that extends for new MFEs [...]*"
	No Versioning Problem Source: ● "*The MFEs are not versioned.*" Proposed Solution: ● "*Adopting a versioning approach like Semantic Versioning is essential to ensure that changes do not impact functioning versions.*"

	Lack of Skeleton Problem Source: ● "*No skeleton or predefined boilerplate is available as a base for creating new MFEs.*" Proposed Solution: ● "*Whenever a new technology is used to implement an MFE, the development team must create a repository containing the necessary base code, known as a boilerplate.*" ● "*[...] the development team should create comprehensive documentation detailing the entire process of creating a new MFE, regardless of the technology.*"
	Common Ownership Problem Source: ● "*A single team is tasked with managing all MFEs [...]*" Proposed Solution: ● "*[...] defining the boundaries of teams and MFEs is essential according to Domain-driven Design […]*" ● "*Creating shared libraries can facilitate boundary definition and promote greater team independence.*"
	Golden Hammer Problem Source: ● "*All MFEs utilize the same technology, even if it does not meet the specific needs of each MFE.*" Proposed Solution: ● "*To choose the most suitable technology that addresses the specific challenges of each MFE [...] When uncertain about a particular technology, conducting a proof-of-concept (POC) can validate its suitability*"
	Micro Frontend as the Goal Problem Source: ● "*Adopting the MFE architecture in inappropriate contexts [...]*" Proposed Solution: ● "*Considering the system's complexity, the feasibility of maintaining automated CI/CD pipelines and the team's restructuring according to different domains is necessary.*"