Poder Executivo
Ministério da Educação
Universidade Federal do Amazonas
Faculdade de Medicina

Programa de Pós-Graduação em Cirurgia (PPGRACI)
Mestrado Profissional em Cirurgia

TÁBITA MAIKA DE PAIVA BASILIO

SOFTWARES E APLICATIVOS: DA IDEIA À IMPLEMENTAÇÃO – UM GUIA PRÁTICO PARA MELHORAR O APRENDIZADO DE PROFISSIONAIS DA ÁREA DA SAÚDE.

Orientadora: Prof(a). Dr(a) Silvania da Conceição Furtado
Co-Orientador: Waldir Sabino da Silva Junior

MANAUS
2025

Poder Executivo
Ministério da Educação
Universidade Federal do Amazonas
Faculdade de Medicina

Programa de Pós-Graduação em Cirurgia (PPGRACI)
Mestrado Profissional em Cirurgia

TÁBITA MAIKA DE PAIVA BASILIO

SOFTWARES E APLICATIVOS: DA IDEIA À IMPLEMENTAÇÃO – UM GUIA PRÁTICO PARA MELHORAR O APRENDIZADO DE PROFISSIONAIS DA ÁREA DA SAÚDE.

Dissertação apresentada à Universidade Federal do Amazonas do Programa de Pós-Graduação em Cirurgia- Mestrado Profissional em Cirurgia, na Área de concentração: Educação, pesquisa, assistência e inovação em cirurgia para a obtenção do Título de Mestre em Cirurgia.

Linha de atuação: Educação, pesquisa, assistência e inovação em cirurgia.
Mestranda: Tábita Maika de Paiva Basilio
Orientador(a): Prof(a) Dr(a) Silvania da Conceição Furtado
Co-Orientador: Waldir Sabino da Silva Junior

MANAUS – AM
2025

Ficha Catalográfica

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

B312s Basilio, Tábita Maika de Paiva
Softwares e aplicativos: da ideia à implementação - um guia prático para melhorar o aprendizado de profissionais da área da saúde. / Tábita Maika de Paiva Basilio. - 2025.
104 f. : il., color. ; 31 cm.

Orientador(a): Silvania da Conceição Furtado .
Coorientador(a): Waldir Sabino da Silva Junior.
Dissertação (mestrado) - Universidade Federal do Amazonas, Programa de Pós-Graduação em Cirurgia, Manaus, 2025.

1. Educação em saúde. 2. Tecnologia. 3. Aplicativos. 4. Saúde. I. Furtado, Silvania da Conceição. II. da Silva, Waldir Sabino. III. Universidade Federal do Amazonas. Programa de Pós-Graduação em Cirurgia. IV. Título

---

FOLHA DE APROVAÇÃO

TÁBITA MAIKA DE PAIVA BASILIO

SOFTWARES E APLICATIVOS: DA IDEIA À IMPLEMENTAÇÃO – UM GUIA PRÁTICO PARA MELHORAR O APRENDIZADO DE PROFISSIONAIS DA ÁREA DA SAÚDE.

Aprovada em: 26 de junho de 2025

BANCA EXAMINADORA

Profº Dr. Silvania da Conceição Furtado, Presidente
Universidade Federal do Amazonas

Profº Dr. Juscimar Carneiro Nunes, Membro
Universidade Federal do Amazonas

Profº Dr. Carlos Maurício Serodio Figueiredo, Membro
Universidade Estadual do Amazonas

Profº Dr. Ronilson Ferreira Freitas, Suplente
Universidade Federal do Amazonas

Profº Dr. Tiago Eugenio de Melo, Suplente
Universidade Estadual do Amazonas

DEDICATÓRIA

A Deus, ao meu marido, meu pai, minha mãe (in memoriam), e minha irmã pelo incentivo na realização deste trabalho.

AGRADECIMENTOS

Agradeço, primeiramente, a Deus, pela força, sabedoria e serenidade concedidas ao longo dessa jornada. Sem Sua presença constante, os desafios teriam sido ainda maiores.

Ao meu marido, meu companheiro de vida, agradeço o amor, paciência, incentivo e apoio incondicional em todos os momentos.

Aos meus pais, que sempre foram minha base. À memória da minha mãe e ao meu pai, minha eterna gratidão por tudo que me ensinaram e pelo amor que sempre me deram. Que este trabalho seja também uma homenagem a eles.

À minha irmã, por seu carinho, escuta e presença constante, mesmo nos momentos mais difíceis. Sua companhia foi essencial nesse processo.

À minha orientadora, professora Silvania da Conceição Furtado, agradeço imensamente pela orientação segura, pelas palavras de encorajamento e pela confiança no meu trabalho. Sua dedicação e compromisso foram fundamentais para a realização desta dissertação.

Aos colegas de mestrado, meu sincero agradecimento pelas trocas de experiências, pelo apoio mútuo e pela convivência enriquecedora ao longo desses anos.

A todos que, de alguma forma, contribuíram para a construção deste trabalho, deixo aqui meu muito obrigado.

Quem tem coragem para enfrentar os perigos, vence-os antes que eles o ameacem.

Públio Siro

RESUMO

JUSTIFICATIVA: A importância dos softwares para a sociedade e, em particular, para a área da saúde, é inegável. Eles aumentam a eficiência e a qualidade dos cuidados, e ampliam o acesso aos serviços de saúde, contribuindo para o bem-estar geral da população quanto para a inovação no setor. Diante disso, este trabalho teve como objetivo elaborar um guia prático para orientar o desenvolvimento de softwares na área da saúde, voltado a pesquisadores sem formação em tecnologia. **OBJETIVOS: Geral:** Desenvolver um guia prático sobre criação de softwares para a área da saúde. **Específicos:** Apresentar as etapas básicas necessárias para criação e desenvolvimento de um software voltado para área da saúde; disponibilizar o guia como ferramenta de apoio para pesquisadores sem conhecimento prévio em tecnologia. **MÉTODO:** Este estudo seguiu o *check list* SQUIRE-EDU, que é uma extensão das diretrizes de publicação SQUIRE 2.0, destinada a orientar a preparação de manuscritos que descrevem ciclos interativos de melhoria na educação de profissões de saúde. **RESULTADO:** Durante o Mestrado Profissional, houve participação nos programas *Ocean Lab* e *Ocean Launch*, da Samsung Brasil, que oferecem suporte e capacitação para projetos de base tecnológica e inovação. Os conhecimentos adquiridos na pesquisa e nos treinamentos resultaram na criação da startup *Symmety AI*, atualmente em processo de formalização e captação de recursos. Além disso, está em andamento a participação no *Programa Catalisa SEBRAE – 1º Ciclo*, voltado à implementação da startup. **CONCLUSÃO:** Ao traduzir conceitos complexos da área de tecnologia para uma linguagem acessível, o guia contribui para reduzir as barreiras que historicamente dificultam a interlocução entre as áreas da saúde e da tecnologia. O material elaborado busca não apenas orientar, mas também inspirar profissionais da saúde a se tornarem protagonistas na criação de soluções digitais que atendam às demandas específicas de suas práticas clínicas, educacionais e assistenciais. Além disso, espera-se que este produto educacional fomente uma cultura de inovação no âmbito do PPGRACI e de outros programas de pós-graduação, colaborando diretamente para a geração de produtos tecnológicos aplicáveis, eficientes e alinhados às necessidades reais dos serviços de saúde. Assim, este trabalho reforça o papel da educação interprofissional como ferramenta estratégica para promover inovação, qualidade e segurança nos serviços, contribuindo para o avanço da saúde digital no Brasil. Espera-se que, com a publicação do guia, alunos do PPGRACI e de outros programas de pós-graduação, bem como profissionais da saúde, sejam incentivados a desenvolver softwares que solucionem desafios do setor, contribuindo para a inserção desses produtos no mercado tecnológico.

Palavras-chave: Educação em saúde, tecnologia, aplicativos, saúde

ABSTRACT

BACKGROUND: The importance of software for society, and in particular for the health sector, is undeniable. It increases the efficiency and quality of care, and expands access to health services, contributing to the general well-being of the population and to innovation in the sector. In view of this, this study aimed to develop a practical guide to guide the development of software in the health sector, aimed at researchers without training in technology. **OBJECTIVES: General:** To develop a practical guide on creating software for the health sector. **Specific:** To present the basic steps necessary for the creation and development of software aimed at the health sector; to make the guide available as a support tool for researchers without prior knowledge of technology. **METHOD:** This study followed the SQUIRE-EDU checklist, which is an extension of the SQUIRE 2.0 publication guidelines, designed to guide the preparation of manuscripts that describe interactive cycles of improvement in health professions education. **RESULT:** During the Professional Master's degree, the student participated in the Ocean Lab and Ocean Launch programs of Samsung Brazil, which offer support and training for technology-based and innovation projects. The knowledge acquired in the research and training resulted in the creation of the startup Symmety AI, which is currently in the process of formalization and fundraising. In addition, the student is currently participating in the Catalisa SEBRAE Program – 1st Cycle, aimed at implementing the startup. **CONCLUSION:** By translating complex concepts from the technology area into accessible language, the guide helps to reduce the barriers that have historically hindered dialogue between the areas of health and technology. The material developed seeks not only to guide, but also to inspire health professionals to become protagonists in the creation of digital solutions that meet the specific demands of their clinical, educational and care practices. In addition, it is expected that this educational product will foster a culture of innovation within the PPGRACI and other postgraduate programs, directly contributing to the generation of applicable, efficient technological products aligned with the real needs of health services. Thus, this work reinforces the role of interprofessional education as a strategic tool to promote innovation, quality and safety in services, contributing to the advancement of digital health in Brazil. It is expected that, with the publication of the guide, students from PPGRACI and other postgraduate programs, as well as health professionals, will be encouraged to develop software that solves challenges in the sector, contributing to the insertion of these products in the technological market.

Keywords: Health education, technology, applications, health

LISTA DE SIGLAS

1. **ANPD:** Autoridade nacional de proteção de dados
2. **Anvisa:** Agência nacional de vigilância sanitária
3. **AWS:** Amazon web services
4. **EMRs:** Registros médicos eletrônicos
5. **GDPR:** Regulamento geral sobre proteção de dados
6. **IA:** Inteligência artificial
7. **LGPD:** Lei geral de proteção de dados
8. **LLMs:** Modelos de linguagem de grande escala
9. **Lot:** Internet das coisas
10. **ML:** Machine learning
11. **ROI:** Retorno sobre o investimento
12. **SDLC:** Ciclo de vida de desenvolvimento de software

SUMÁRIO

INTRODUÇÃO

Os softwares, componentes essenciais da tecnologia moderna, desempenham um papel fundamental na operacionalização de dispositivos eletrônicos e na facilitação de uma vasta gama de tarefas, desde as mais simples às mais complexas (Sommerville, 2016). Eles consistem em um conjunto de instruções programadas que permitem que computadores, smartphones e outros dispositivos eletrônicos executem funções específicas (Pressman & Maxim, 2020). Divididos amplamente em software de sistema e software aplicativo (Tanenbaum & Bos, 2015), eles abrangem sistemas operacionais, que gerenciam os recursos de hardware e fornecem uma plataforma para aplicativos especializados que atendem às necessidades particulares dos usuários (Silberschatz *et al*., 2018).

Na sociedade contemporânea, o impacto dos softwares é onipresente e revolucionário. Eles impulsionaram a automação industrial, aprimoraram a comunicação global, além de facilitar o gerenciamento de grandes volumes de dados promovendo avanços em áreas como inteligência artificial, aprendizado de máquina e ciência de dados (Russell & Norvig, 2010; Mitchell, 1997). Os aplicativos móveis transformaram a maneira como nos conectamos, trabalhamos e nos divertimos, enquanto os softwares empresariais e de produtividade através de processos otimizados aprimoram a eficiência organizacional (Sommerville, 2016).

Nesta era digital, a regulamentação de softwares e aplicativos voltados para a área da saúde está redefinindo a forma como pacientes, profissionais da área da saúde e instituições de saúde interagem e gerenciam informações (Bashshur *et al*., 2015). Essas ferramentas tecnológicas oferecem uma variedade de funcionalidades, desde facilitar o agendamento de consultas até fornecer acesso rápido a informações médicas e promover o monitoramento remoto da saúde (Kvedar *et al*., 2014). Ao possibilitar uma comunicação mais eficaz entre os pacientes e falar sobre cuidados de saúde, os softwares e aplicativos estão promovendo uma maior participação dos pacientes em sua própria saúde, melhorando a eficácia dos cuidados e aumentando a eficiência dos serviços de saúde (Halamka *et al*., 2013). Segundo Silva *et al*. (2020), “a integração de soluções tecnológicas no ambiente de saúde não apenas facilita o trabalho dos profissionais, mas também melhora significativamente a experiência do paciente”.

Entretanto, o desenvolvimento de um software envolve uma série de etapas complexas, desde a concepção e *design* até a modificação, passando por testes de validação e usabilidade, implementação e manutenção (Sommerville, 2016). Metodologias como *Agile* e *DevOps* transformaram o ciclo de vida de desenvolvimento de software, promovendo uma abordagem mais interativa, colaborativa e eficiente (Fowler & Highsmith, 2001; Kim *et al*., 2016). Além disso, linguagens de programação variadas, como Python, Java, JSON e C++, fornecem uma base sobre como os softwares são construídos, cada um com suas próprias características e áreas de aplicação específicas (Lutz, 2013; Deitel & Deitel, 2015).

Diante de tanta especificidade para concretizar a implementação desse tipo de tecnologia na área da saúde e, considerando que profissionais desta área têm pouca ou nenhuma intimidade com o vocabulário próprio utilizado no cenário tecnológico, a comunicação interdisciplinar fica prejudicada. Cada fase do desenvolvimento é fundamental para garantir que o produto final atenda às necessidades dos usuários e funcione de maneira eficiente e segura. De acordo com Oliveira (2019), “um planejamento detalhado e uma execução cuidadosa são fundamentais para o sucesso de qualquer projeto de software, especialmente na área da saúde, onde a precisão e a confiabilidade são imperativas”.

Assim, este projeto se propõe a produzir um guia prático que visa fornecer uma abordagem estruturada para profissionais da saúde que desejam desenvolver suas próprias soluções tecnológicas. Ao entenderem as etapas e os requisitos do processo de desenvolvimento, esses profissionais poderão colaborar mais efetivamente com equipes de Tecnologia da Informação (TI) e garantir que os softwares criados atendam às suas necessidades específicas. Como destaca Souza (2021), “a participação ativa dos profissionais da saúde no desenvolvimento de software é essencial para a criação de ferramentas que realmente façam a diferença no dia a dia clínico”.

JUSTIFICATIVA

O Programa de Pós-Graduação em Cirurgia (PPGRACI-UFAM) é um mestrado profissional que tem como meta principal o desenvolvimento de soluções em saúde com ênfase em ambientes hospitalares. Entretanto, passada mais de uma década de sua atuação no âmbito da Pós-Graduação no estado do Amazonas, os produtos gerados oriundos das dissertações, no que diz respeito a produtos tecnológicos que tenham efetivamente chegado à fase de implementação, ainda são incipientes.

Este trabalho se propôs a desenvolver um guia prático para desenvolvimento de softwares voltados para área da saúde, a fim de auxiliar os pesquisadores que não possuem experiência na área de tecnologia na elaboração e execução de projetos de mestrado cujo produto proposto seja uma solução tecnológica como softwares e aplicativos.

A intenção do desenvolvimento deste trabalho é conseguir suprir a necessidade de um guia elaborado com uma linguagem “amigável” para orientar os pesquisadores no caminho a seguir quando estiverem desenvolvendo este tipo de pesquisa junto ao PPGRACI. Além disso, outros programas de pós-graduação também poderão se beneficiar com o produto gerado.

REVISÃO DA LITERATURA

Tecnologias em Saúde

História

A evolução da tecnologia na saúde testemunhou avanços inovadores que revolucionaram a medicina e os cuidados com a saúde ao longo dos séculos. Nos séculos XX e XXI, a integração da informática e biotecnologia acelerou ainda mais os avanços na saúde. A introdução de computadores nos anos 1960 possibilitou a chegada de sistemas de registros eletrônicos de saúde, melhorando a organização e o acesso às informações dos pacientes (Oliveira, 2023).

A adoção de registros médicos eletrônicos (EMRs) na gerontologia trouxe benefícios significativos, permitindo análise de dados, desenvolvimento de ferramentas prognósticas e previsão de riscos de doenças (BEDNORZ; JYLHÄVÄ; RAITANEN, 2023). Além disso, a pandemia de Covid-19 destacou a importância de tecnologias como a telemedicina, aplicativos móveis de saúde e sistemas de saúde baseados em blockchain para enfrentar desafios de saúde pública (KUO; KIM; OHNO-MACHADO, 2021).

Softwares

Conjuntos de instruções que permitem a operação de dispositivos eletrônicos, são elementos base na era digital. Os softwares além de serem objeto auxiliar do conhecimento, oferecem uma melhor abrangência para as diversas áreas da saúde, desde o diagnóstico de doenças até o auxílio no tratamento de pacientes. Dessa forma é relevante analisar como a relação entre a ciência e a tecnologia tem trazido benefícios para a população em geral, sobretudo na área de saúde. Além disso, é importante observar que a incorporação dos seus resultados e avanços sejam implementados em bases mais sólidas sempre levando em consideração o bem-estar da população (Sommerville, 2016).

A integração de softwares na sociedade moderna representa um dos pilares fundamentais que sustentam a evolução tecnológica e a melhoria da qualidade de vida. Em praticamente todas as esferas, desde a comunicação até a educação, os softwares desempenham um papel central, automatizando processos, facilitando o acesso à informação e promovendo a eficiência operacional (Sommerville, 2016). No contexto da saúde, a importância dos softwares se torna ainda mais evidente e impactante (Bashshur *et al*., 2015).

Softwares médicos e aplicativos de saúde foram transformados de maneira que os serviços de saúde são prestados, fornecendo soluções inovadoras para desafios antigos e emergentes (Kvedar *et al*., 2014). Eles facilitam o diagnóstico preciso e rápido por meio de ferramentas de imagem e análise de dados, otimizam a gestão hospitalar e de prontuários eletrônicos, e promovem a telemedicina, que expande o acesso a cuidados médicos para populações remotas e sub atendidas. Além disso, softwares de monitoramento remoto permitem o acompanhamento contínuo de pacientes com condições crônicas, melhorando o controle da saúde e reduzindo a necessidade de internações (Bashshur *et al*., 2015).

A análise de grandes volumes de dados médicos, viabilizada por softwares avançados, está impulsionando a pesquisa médica e o desenvolvimento de novos tratamentos, ao mesmo tempo em que a inteligência artificial e o aprendizado de máquina estão revolucionando a personalização dos cuidados de saúde, oferecendo tratamentos adaptados às necessidades específicas de cada paciente (Kvedar *et al*., 2014).

A digitalização dos registros médicos facilita a colaboração entre profissionais de saúde, melhora a precisão dos dados e aumenta a segurança dos pacientes (Bashshur *et al*., 2015). Além dos benefícios diretos aos pacientes e profissionais de saúde, os softwares são significativamente importantes para a sustentabilidade dos sistemas de saúde, otimizando recursos e custos operacionais (Katz & Shapiro, 1994). Eles também desempenham um papel importante na educação e treinamento de novos profissionais, por meio de simulações e ferramentas de aprendizagem interativa (Sommerville, 2016).

A importância dos softwares para a sociedade e, em particular, para a área da saúde, é inegável. Eles não apenas melhoram a eficiência e a qualidade dos cuidados, mas também promovem um acesso mais equitativo e abrangente aos serviços de saúde, contribuindo para o bem-estar geral da população e a inovação contínua no setor (Halamka *et al*., 2013).

Qualidade e Segurança de Softwares

No processo de avaliação da qualidade de software, não basta apenas identificar que atributos determinam essa qualidade, mas também que procedimentos adotar, para controlar seu processo de desenvolvimento, de forma a atingir o nível de qualidade desejado. Isto é realizado através da aplicação de

métricas de forma organizada e bem projetada, tornando os desenvolvedores mais conscientes da relevância do gerenciamento e dos compromissos para com a qualidade (Belchior, 1997). A qualidade de um software é um conjunto de propriedades a serem satisfeitas em determinado grau, de modo que o software satisfaça as necessidades de seus usuários (Jensen *et al*. 2012).

Por outro lado, a segurança de software, um aspecto crítico, envolve a proteção contra vulnerabilidades e ameaças cibernéticas, exigindo práticas rigorosas de segurança e atualizações constantes. Com o advento de novas tecnologias e a crescente digitalização, a evolução contínua dos softwares permanece vital para a inovação e a adaptação às novas necessidades e desafios. Nos últimos anos, os desenvolvedores de software gastaram aproximadamente 30-40% de seu orçamento em verificação e validação de software (Red Hat, 2020).

Evolução Histórica dos Softwares

O conceito de software remonta aos primeiros computadores digitais na década de 1940, com Alan Turing e John von Neumann contribuindo significativamente para sua fundamentação teórica (Turing, 1936; von Neumann, 1945). O primeiro software reconhecido foi criado por Tom Kilburn em 1948, permitindo a execução de um programa armazenado em um computador (Kilburn, 1948). Desde então, a evolução dos softwares passou por várias fases:

Décadas de 1950 e 1960: Desenvolvimento de linguagens de programação como Fortran, COBOL e LISP, focando em aplicações científicas e comerciais (Backus *et al*., 1957; Hopper, 1958; McCarthy, 1960).

Década de 1970: Introdução de sistemas operacionais como Unix e o surgimento da engenharia de software como disciplina formal (Ritchie & Thompson, 1974).

Década de 1980: Popularização dos computadores pessoais e surgimento de interfaces gráficas, como o sistema operacional Windows (Microsoft, 1985).

Década de 1990 em diante: Expansão da internet, desenvolvimento de softwares baseados na web e o surgimento de metodologias ágeis e *DevOps* (Fowler & Highsmith, 2001; Kim *et al*., 2016).

Evolução Tecnológica dos Softwares no Século XXI

A Revolução da Internet e a Era da Conectividade

No início do século XXI, a internet se consolidou como uma plataforma essencial para o desenvolvimento de software. A popularização da banda larga e o surgimento da Web 2.0 permitiram a criação de aplicativos mais interativos e colaborativos. Segundo Oliveira (2019), "a transição para a Web 2.0 marcou uma mudança paradigmática, onde os usuários passaram a ser não apenas consumidores, mas também produtores de conteúdo". Essa era de conectividade facilitou o desenvolvimento de softwares baseados em nuvem, que oferecem maior flexibilidade e acessibilidade.

Dispositivos Móveis e Aplicativos

A proliferação de dispositivos móveis, como smartphones e tablets, revolucionou o desenvolvimento de software. Aplicativos móveis se tornaram uma parte integral da vida cotidiana, oferecendo soluções para comunicação, entretenimento, saúde e muito mais. De acordo com Silva *et al*. (2020), "o desenvolvimento de aplicativos móveis exigiu novas abordagens de *design* e programação, focadas na usabilidade e na experiência do usuário". A criação de lojas de aplicativos, como a App Store e o Google Play, facilitou a distribuição e o acesso a uma vasta gama de softwares.

Inteligência Artificial:

Machine Learning

A inteligência artificial (IA) e suas subáreas como o *machine learning* (ML) emergiram como tecnologias disruptivas no desenvolvimento de software. Essas tecnologias permitem a criação de sistemas que podem aprender e se adaptar com o tempo, oferecendo soluções mais inteligentes e personalizadas. Refere-se à capacidade de máquinas realizarem tarefas que normalmente exigiriam inteligência humana, como reconhecimento de fala, tomada de decisões e tradução de idiomas.

No contexto da saúde, a IA pode ser usada para melhorar diagnósticos, personalizar tratamentos e otimizar a gestão hospitalar (OMS, 2021)

Machine Learning (ML), ou aprendizado de máquina, é um subcampo da IA que permite que os sistemas aprendam e melhorem automaticamente a partir da experiência, sem serem explicitamente programados para tal. Isso é feito através da análise de grandes volumes de dados e da identificação de padrões que podem ser usados para fazer previsões ou tomar decisões. Segundo Machado (2018), "a IA e o ML têm o potencial de transformar diversos setores, desde a saúde até a indústria financeira, ao automatizar processos complexos e melhorar a tomada de decisões".

Eficiência das LLMs na Saúde

Os Modelos de Linguagem de Grande Escala (LLMs), como o GPT-4, são uma aplicação avançada de ML que pode processar e gerar texto de maneira muito semelhante à humana. De acordo com Russell e Norvig (2020), as LLMs podem ser usadas em ambiente hospitalar para:

- Apoio ao Diagnóstico: Analisando registros médicos e sugerindo possíveis diagnósticos com base em sintomas e históricos de pacientes.
- Assistência na Pesquisa: Ajudando a revisar literatura médica e identificar novas tendências ou correlações em dados clínicos.
- Comunicação com Pacientes: Facilitando a comunicação entre médicos e pacientes, especialmente em situações em que há barreiras linguísticas.

Foundation Vision Models na Saúde

Os Foundation Vision Models são modelos de IA treinados em grandes conjuntos de dados visuais, como imagens médicas, para realizar tarefas como reconhecimento de padrões e detecção de anomalias (Pearson & Topol, 2019) na área da saúde, esses modelos podem ser utilizados para:

- Diagnóstico por Imagem: Analisando radiografias, tomografias e ressonâncias magnéticas para identificar doenças como câncer e fraturas ósseas.
- Monitoramento de Pacientes: Acompanhando sinais vitais e detectando mudanças sutis que podem indicar complicações.

- Cirurgia Assistida por IA: Ajudando cirurgiões a planejar e executar procedimentos com maior precisão.

Internet das Coisas (IoT)

A Internet das Coisas (IoT) conecta dispositivos físicos à internet, permitindo a coleta e troca de dados em tempo real. Essa tecnologia tem aplicações vastas, desde casas inteligentes até monitoramento de saúde. Oliveira (2019) destaca que "a IoT está transformando a maneira como interagimos com o mundo ao nosso redor, proporcionando maior eficiência e conveniência". No setor de saúde, dispositivos IoT podem monitorar pacientes remotamente, melhorando o cuidado e a gestão de doenças crônicas.

Uma revisão sistemática analisou a eficácia da IoT na área da saúde e encontrou resultados promissores. A pesquisa destacou que a IoT pode melhorar significativamente o monitoramento de pacientes, a gestão de doenças crônicas e a eficiência dos serviços de saúde. No entanto, também foram identificados desafios, como a segurança dos dados e a necessidade de infraestrutura adequada (Rosa, Sousa; Silva, 2020).

De acordo com o ministério da saúde (2021), podemos citar como exemplos de uso de Lot na saúde:

Monitoramento Remoto de Pacientes: Dispositivos vestíveis, como relógios inteligentes, podem monitorar sinais vitais (batimentos cardíacos, níveis de oxigênio no sangue, etc.) e enviar esses dados em tempo real para os médicos. Isso é especialmente útil para pacientes com doenças crônicas, permitindo um acompanhamento contínuo sem a necessidade de visitas frequentes ao hospital.

Gestão de Medicamentos: Dispositivos conectados podem lembrar os pacientes de tomar seus medicamentos na hora certa e até mesmo notificar os médicos ou cuidadores se uma dose for esquecida.

Sensores em Ambientes Hospitalares: Sensores podem monitorar a temperatura, umidade e outros fatores ambientais em hospitais para garantir condições ideais para pacientes e equipamentos médicos.

Dispositivos de Reabilitação: Equipamentos de fisioterapia conectados podem ajustar automaticamente os exercícios com base no progresso do paciente e enviar relatórios detalhados para os fisioterapeutas.

Computação em Nuvem

A computação em nuvem revolucionou o desenvolvimento e a distribuição de software, permitindo que aplicativos e dados sejam acessados de qualquer lugar com uma conexão à internet. Em vez de armazenar dados e executar programas em um computador local ou servidor físico, esses recursos são disponibilizados por meio da internet, permitindo acesso remoto a partir de qualquer lugar. Silva *et al*. (2020) afirmaram que "a computação em nuvem oferece escalabilidade, flexibilidade e redução de custos, tornando-se uma escolha popular para empresas de todos os tamanhos". Serviços como *Amazon Web Services* (AWS), *Microsoft Azure* e *Google Cloud Platform* são exemplos de plataformas que suportam o desenvolvimento de software em nuvem.

A computação em nuvem representa uma revolução na forma como armazenamos e acessamos dados, oferecendo uma combinação de eficiência, economia e flexibilidade. Com a nuvem, os dados e aplicativos podem ser acessados de qualquer lugar com uma conexão à internet, facilitando o trabalho remoto e a colaboração. No entanto, é importante considerar tanto os benefícios quanto às limitações ao adotar essa tecnologia, pois o acesso aos serviços em nuvem depende de uma conexão estável à internet e embora os provedores de nuvem ofereçam segurança robusta, sempre há riscos associados ao armazenamento de dados sensíveis em servidores externos e ainda que a nuvem possa reduzir custos iniciais, os custos de assinatura podem se acumular ao longo do tempo, especialmente para empresas que necessitam de grandes quantidades de recursos (Rodrigues; Galdino; Neto, 2019).

Segurança e Privacidade

Com o aumento da digitalização, a segurança e a privacidade dos dados se tornaram preocupações centrais no desenvolvimento de software. A conformidade com regulamentações como o GDPR (Regulamento Geral sobre a Proteção de Dados) na Europa e a LGPD (Lei Geral de Proteção de Dados) no Brasil é essencial para proteger as informações dos usuários. Segundo Souza (2021), "a implementação de medidas robustas de segurança é o ponto-chave para garantir a confiança dos usuários e a integridade dos sistemas".

A Lei Geral de Proteção de Dados Pessoais (LGPD), Lei n° 13.709/2018, foi promulgada para proteger os direitos fundamentais de liberdade e de privacidade, e a livre formação da personalidade de cada indivíduo. A Lei fala sobre o tratamento de dados pessoais, dispostos em meio físico ou digital, feito por pessoa física ou jurídica de direito público ou privado, englobando um amplo conjunto de operações que podem ocorrer em meios manuais ou digitais.

No âmbito da LGPD, o tratamento dos dados pessoais pode ser realizado por dois agentes de tratamento – o Controlador e o Operador. Além deles, há a figura do Encarregado, que é a pessoa indicada pelo Controlador para atuar como canal de comunicação entre o Controlador, o Operador, os titulares dos dados e a Autoridade Nacional de Proteção de Dados (ANPD).

A lei estabelece uma estrutura legal de direitos dos titulares de dados pessoais. Esses direitos devem ser garantidos durante toda a existência do tratamento dos dados pessoais realizado pelo órgão ou entidade. Para o exercício dos direitos dos(as) titulares, a LGPD prevê um conjunto de ferramentas que aprofundam obrigações de transparência ativa e passiva, e criam meios processuais para mobilizar a Administração Pública (GOV,2024).

A Agência Nacional de Vigilância Sanitária (Anvisa) publicou a Portaria 1.184/2023, que institui a Política de Proteção de Dados Pessoais da agência. A portaria é um passo importante para que a Anvisa se adeque à Lei Geral de Proteção de Dados Pessoais (LGPD), Lei 13.709/2018. (Ministério da saúde, 2023).

OBJETIVO E ABRANGÊNCIA

Art. 1º A Política de Proteção de Dados Pessoais da Agência Nacional de Vigilância Sanitária tem como objetivo estabelecer, no âmbito da Agência, diretrizes para a proteção dos dados pessoais, para o cumprimento da legislação, normas, orientações e demais atos quanto à privacidade, à proteção dos dados pessoais, à transparência, ao acesso às informações públicas e à proteção das liberdades e dos direitos fundamentais dos indivíduos.

Parágrafo único. Esta Política se aplica aos servidores, colaboradores, terceirizados, estagiários, fornecedores,

> prestadores de serviço e todos que realizem atividades que envolvam, de forma direta ou indireta, tratamento de dados pessoais custodiados pela Agência.

Classificação dos Softwares

Os softwares podem ser classificados nas seguintes categorias principais (Sommerville, 2016):

a) Software de Sistema: Inclui sistemas operacionais, drivers e importadores que gerenciam os recursos do hardware e fornecem uma plataforma para a execução de aplicativos (Tanenbaum & Bos, 2015).

   Sistemas Operacionais como Windows, macOS, Linux (Silberschatz *et al*., 2018).

   Drivers: Facilitam a comunicação entre hardware e software (Tanenbaum & Bos, 2015).

   Utilitários: Ferramentas de manutenção e segurança, como antivírus e desfragmentadores de disco (Sommerville, 2016).

b) Aplicativo de Software: Projetado para realizar tarefas específicas para o usuário final (Deitel & Deitel, 2015).

   Aplicativos de Escritório: Microsoft Office, Google Workspace (Deitel & Deitel, 2015).

   Softwares de *Design* e Multimídia: Adobe Photoshop, AutoCAD (Lutz, 2013).

   Aplicativos Móveis: WhatsApp, Instagram (Katz & Shapiro, 1994).

c) Softwares Específicos para Setores: Sistemas de gerenciamento hospitalar, softwares de contabilidade (Bashshur et al., 2015).

Metodologias de Desenvolvimento de Software

As metodologias ágeis, como *Scrum* e *Kanban*, têm sido amplamente adotadas no desenvolvimento de software devido à sua flexibilidade e capacidade de adaptação às mudanças de requisitos. Segundo Machado (2018), "os métodos ágeis permitem uma maior interação entre os desenvolvedores e os usuários finais, o que é importante para garantir que o software atenda às necessidades reais dos profissionais da saúde". Além disso, a identificação de problemas para definir requisitos é uma etapa fundamental, pois envolve a coleta e análise das necessidades dos usuários. Alflen e Prado (2018) destacaram que "a qualidade dos requisitos é diretamente influenciada pelas técnicas de elicitação utilizadas".

Metodologias como Agile, Scrum e DevOps revolucionaram o desenvolvimento de software, promovendo ciclos de desenvolvimento mais curtos, maior colaboração entre equipes e entregas contínuas (Fowler & Highsmith, 2001; Kim *et al*., 2016).

Metodologia Scrum

Scrum é uma metodologia ágil de desenvolvimento de software que se destaca pela sua abordagem interativa e incremental. Criada por Jeff Sutherland e Ken Schwaber nos anos 1990, Scrum tem como objetivo principal aumentar a produtividade das equipes e a qualidade dos produtos entregues. É baseada no empirismo, ou seja, decisões são tomadas com base na experiência e na observação.

Estrutura e Papeis no Scrum

Scrum Master

O Scrum Master é responsável por garantir que a equipe siga os princípios e práticas do Scrum. Ele atua como um facilitador, removendo impedimentos e

ajudando a equipe a melhorar continuamente. Segundo Schwaber e Sutherland (2017), "o *Scrum Master* não é um gerente de projeto tradicional, mas sim um líder servo que apoia a equipe".

Product Owner

O *Product Owner* é o responsável por maximizar o valor do produto e gerenciar o *backlog* do produto. Ele define as prioridades e garante que a equipe esteja trabalhando nas tarefas mais importantes. De acordo com Schwaber e Sutherland (2017), "o *Product Owner* deve ter uma visão clara do produto e ser capaz de comunicar essa visão para a equipe".

Benefícios do Scrum

- Flexibilidade e Adaptabilidade: Permite ajustes rápidos às mudanças de requisitos e prioridades.
- Transparência: Através de reuniões diárias (*Daily Stand-ups*) e revisões de *sprint*, todos os membros da equipe têm uma visão clara do progresso e dos obstáculos.
- Melhoria Contínua: A retrospectiva do *sprint* permite que a equipe identifique e implemente melhorias contínuas no processo.

Desenvolvimento de Softwares

O desenvolvimento de softwares é um processo complexo que envolve várias etapas, conhecido como Ciclo de Vida de Desenvolvimento de Software (SDLC). As principais fases incluem:

Etapas Principais e Pontos Necessários:

1.Identificação do Problema e Definição dos Objetivos

Identificação do Problema: Analise as necessidades e problemas existentes na área da saúde específica e que podem ser resolvidos com soluções tecnológicas. (PRESSMAN; MAXIM, 2016).

Definição dos Objetivos: Estabeleça claramente o que você deseja alcançar com o software ou aplicativo. Defina os objetivos de maneira específica, mensurável, alcançável, relevante e temporal (SMART) (DORSEY, 2015).

SMART é um acrônimo que ajuda a definir objetivos de forma clara e eficiente, cada letra representa um critério que deve ser considerado ao estabelecer um objetivo.

1. Específico (Specific): O objetivo deve ser claro e específico. Isso ajuda a focar os esforços e a definir claramente o que se deseja alcançar.
2. M - Mensurável (Measurable): O objetivo deve ser mensurável, ou seja, deve ser possível acompanhar o progresso e saber quando ele foi alcançado.
3. A - Alcançável (Achievable): O objetivo deve ser realista e alcançável, considerando os recursos e as limitações existentes.
4. R - Relevante (Relevant): O objetivo deve ser relevante e alinhado com outras metas e prioridades.
5. T - Temporal (Time-bound): O objetivo deve ter um prazo definido para ser alcançado. Isso ajuda a manter o foco e a motivação.

2. Pesquisa e Análise

Pesquisa de Mercado: Verifique se já existem soluções semelhantes no mercado. Identifique lacunas e oportunidades de melhoria.

Análise de Requisitos: Colete e documente os requisitos funcionais (o que o software deve fazer) e não funcionais (desempenho, segurança, etc.). Envolva *stakeholders*, como outros profissionais de saúde e pacientes, nesta etapa. (KENDALL; KENDALL, 2010).

3. Planejamento do Projeto

Escolha da Metodologia de Desenvolvimento: Decida se você seguirá uma metodologia ágil (como Scrum ou Kanban) ou tradicional (como Waterfall).

Cronograma e Recursos: Elabore um cronograma detalhado, defina marcos importantes e aloque os recursos necessários (equipe, orçamento, ferramentas). (SILVA, 2018).

4. *Design* e Prototipagem

Design de Interface do Usuário (UI): Crie *wireframes* e protótipos da interface do usuário. Foque na usabilidade e acessibilidade.

Wireframes são representações visuais simplificadas de um projeto digital, como um site ou aplicativo. Eles funcionam como um “esqueleto” ou “planta baixa” que mostra a estrutura básica e os componentes de uma página, sem se aprofundar em detalhes de design, cores ou imagens. Em essência, um wireframe é um esboço que ajuda a planejar a disposição dos elementos e a funcionalidade de uma interface.

Experiência do Usuário (UX): Garanta que a navegação seja intuitiva e que o aplicativo atenda às necessidades dos usuários finais.

Entenda quem são os usuários e quais são suas necessidades, isso pode ser feito através de pesquisas, entrevistas e testes de usabilidade. Também é importante manter a simplicidade, pois é fundamental para uma navegação intuitiva.
Evite menus complexos e sobrecarregados, a consistência na navegação ajuda os usuários a se familiarizar rapidamente com o aplicativo. Mantenha a estrutura de menu, terminologia e design em todas as páginas (GARRETT, 2011).

5. Desenvolvimento

Escolha das Tecnologias: Selecione as linguagens de programação, *frameworks* e plataformas adequadas (por exemplo, Java, Swift, React Native).

Implementação: Codifique o software seguindo as melhores práticas de desenvolvimento, como controle de versão (Git) e revisão de código (PRESSMAN; MAXIM, 2016).

6. Teste e Validação

Testes Unitários e de Integração: Verifique se os componentes individuais e suas interações funcionam corretamente.

Os testes unitários são projetados para verificar se componentes individuais de um software funcionam corretamente. Eles focam na menor unidade testável do código, como funções, métodos ou classes.

Os testes de integração verificam se diferentes módulos ou componentes de um sistema funcionam bem juntos. Eles são mais abrangentes que os testes unitários e podem envolver interações com bancos de dados, APIs e outros serviços externos.

Testes de Usabilidade: Realize testes com usuários reais para identificar problemas na interface e na experiência do usuário.

Imagine que você está testando a usabilidade de um aplicativo de agendamento de consultas médicas. Aqui está um exemplo de como você pode conduzir o teste:

- Objetivo: Verificar se os usuários conseguem agendar uma consulta médica facilmente.
- Participantes: 10 usuários com diferentes níveis de familiaridade com tecnologia.
- Cenário de Teste: “Agende uma consulta com um cardiologista para a próxima semana.”
- Observações: Durante o teste, você observa que muitos usuários têm dificuldade em encontrar a opção de especialidade médica. Isso indica que a navegação precisa ser mais intuitiva.

Testes de Aceitação: Certifique-se de que o software atende aos requisitos definidos inicialmente.

Suponha que você está realizando testes de aceitação para um sistema de gestão hospitalar. Aqui está um exemplo de como você pode conduzir o teste:

- Critérios de Aceitação: O sistema deve permitir que os usuários registrem novos pacientes, agendem consultas e gerem relatórios médicos.
- Casos de Teste:
    - “Registrar um novo paciente e verificar se todas as informações são salvas corretamente.”
    - “Agendar uma consulta e verificar se ela aparece no calendário do médico.”
    - “Gerar um relatório médico e verificar se todos os dados estão corretos.”
- Testadores: 5 funcionários do hospital, incluindo recepcionistas e médicos.
- Observações: Durante os testes, os testadores encontram um problema ao gerar relatórios médicos. O problema é registrado e corrigido antes do lançamento do sistema (JORGESON, 2013).

7. Implantação

Preparação do Ambiente: Configure o ambiente de produção, incluindo servidores, bancos de dados e redes.

Implantação Gradual: Considere uma abordagem de lançamento gradual para mitigar riscos e corrigir problemas antes de uma implementação completa. (PRESSMAN; MAXIM, 2016).

8. Treinamento e Suporte

Treinamento dos Usuários: Ofereça treinamento aos profissionais de saúde que utilizarão o software.

Antes de iniciar o treinamento, é importante avaliar as necessidades específicas de cada grupo de usuários. Médicos, enfermeiros, secretárias e administradores terão diferentes necessidades e funcionalidades do software que precisam dominar.
Crie módulos de treinamento personalizados para cada grupo de usuários. Por exemplo:

- Médicos e Enfermeiros: Foco em funcionalidades clínicas, como acesso a históricos médicos, prescrição eletrônica e registro de consultas.
- Secretárias e Auxiliares: Treinamento em agendamentos, gestão de filas de espera e comunicação com pacientes.
- Administradores: Módulos sobre gestão financeira, relatórios e análise de dados.

Também é possível utilizar como ferramentas de treinamentos: videoaulas e tutoriais, manuais e guias de referências rápidas online, além de sessões de treinamentos presenciais.

Suporte Contínuo: Estabeleça um canal de suporte para resolver dúvidas e problemas que possam surgir.

O suporte contínuo é essencial para garantir que os usuários de um software, especialmente em ambientes críticos como o de saúde, possam obter ajuda e resolver problemas rapidamente. como por exemplo: Help Desk, Chat Online e Chatbots, Base de Conhecimento e FAQs, Suporte por E-mail e Telefone (FITZGERALD *et al*., 2013).

9. Monitoramento e Manutenção

Monitoramento Contínuo: Utilize ferramentas de monitoramento para garantir que o software funcione corretamente e que seja possível identificar possíveis problemas.

Para realizar o monitoramento contínuo de um software, existem várias ferramentas eficazes que podem ajudar a garantir o desempenho, a disponibilidade e a segurança do sistema.

Zabbix e Prometheus são exemplos de ferramentas de monitoramento de código aberto que oferecem monitoramento em tempo real de servidores, redes e aplicações.

Atualizações e Melhorias: Planeje atualizações regulares para adicionar novas funcionalidades e corrigir bugs.

Antes de iniciar qualquer atualização ou melhoria, é essencial planejar e avaliar as necessidades do software. Isso inclui:

- Identificar Requisitos: Determine quais funcionalidades precisam ser adicionadas ou melhoradas com base no *feedback* dos usuários e nas necessidades do negócio.
- Análise de Impacto: Avalie como as mudanças afetarão o sistema atual e os usuários. Isso ajuda a identificar possíveis riscos e a planejar mitigação (BARROS,2020).

10. Avaliação e *Feedback*

Coleta de *Feedback*: Solicite *feedback* dos usuários para avaliar a eficácia do software e identificar áreas de melhoria.

A coleta de feedback dos usuários é essencial para avaliar a eficácia do software e identificar áreas de melhoria. Aqui estão alguns métodos e exemplos práticos de como isso pode ser feito através de: Pesquisas de Satisfação, Formulários de Feedback In-App, Entrevistas com usuários, Feedback em Redes Sociais e Fóruns

Avaliação de Impacto: Meça o impacto do software na prática da saúde, incluindo melhorias em eficiência, precisão e satisfação dos usuários.

De acordo com Rubin e Chisnell (2008) medir a avaliação de impacto de um software envolve analisar como ele afeta os usuários, os processos de negócios e a organização como um todo. Alguns métodos podem ser utilizados, como:

1. Métricas de Uso
    - Descrição: Avaliar como os usuários interagem com o software, incluindo frequência de uso, funcionalidades mais utilizadas e tempo gasto em diferentes tarefas.
    - Exemplo: Um hospital pode monitorar quantas vezes os médicos acessam os prontuários eletrônicos e quais funcionalidades são mais utilizadas, como prescrição de medicamentos ou visualização de exames.

2. Satisfação do Usuário
    - Descrição: Coletar feedback dos usuários sobre sua satisfação com o software através de pesquisas, entrevistas e grupos de foco.
    - Exemplo: Após a implementação de um novo sistema de agendamento, uma clínica pode enviar pesquisas de satisfação aos pacientes para avaliar a facilidade de uso e a eficiência do sistema.

3. Desempenho do Sistema
    - Descrição: Medir o desempenho técnico do software, incluindo tempo de resposta, tempo de inatividade e taxa de erros.
    - Exemplo: Um sistema de gestão hospitalar pode ser avaliado com base no tempo de resposta ao acessar registros de pacientes e na frequência de falhas ou erros durante o uso.

4. Eficiência Operacional
    - Descrição: Avaliar como o software impacta a eficiência dos processos de negócios, como redução de tempo em tarefas administrativas e aumento da produtividade.
    - Exemplo: Um hospital pode medir a redução no tempo necessário para processar admissões de pacientes após a implementação de um novo sistema de gestão de admissões.

5. Retorno sobre o Investimento (ROI)
    - Descrição: Calcular o retorno financeiro obtido com a implementação do software em comparação com o custo de desenvolvimento e manutenção.

Objetivos

Geral:

- Elaborar um guia prático sobre desenvolvimento de softwares

Específicos

- Demonstrar quais etapas básicas são necessárias para criação de um software voltado para a área da saúde.

- Disponibilizar o produto gerado deste projeto para auxiliar outros pesquisadores que não possuem conhecimento sobre a área de tecnologia.

METODOLOGIA

Trata-se da geração de um produto educacional que agregará informações sobre produção/desenvolvimento de softwares e seguirá o *check list* SQUIRE-EDU, que é uma extensão das diretrizes de publicação SQUIRE 2.0. Este *chek list* se destina a orientar a preparação de manuscritos que descrevem ciclos interativos de melhoria na educação de profissões de saúde. SQUIRE-EDU incentiva a expansão da ciência educacional, oferecendo uma alternativa ao uso de um projeto de pesquisa. O SQUIRE-EDU também pode ser útil para auxiliar no desenho de propostas de intervenções, condução dos trabalhos e análise e divulgação das inovações educacionais (OGRINC, et al., 2019).

Para o conteúdo do Guia haverá um capítulo dedicado às questões éticas e legais necessárias para a concepção, elaboração e implementação de softwares na área da saúde.

LGPD - A execução de projetos de implementação de softwares na área da saúde deve estar condicionada ao disposto no art. 49 da Lei Geral de Proteção de Dados (LGPD). Esta lei visa proteger os direitos fundamentais de liberdade e de privacidade, e a livre formação da personalidade de cada indivíduo. A Lei fala sobre o tratamento de dados pessoais, dispostos em meio físico ou digital, feito por pessoa

física ou jurídica de direito público ou privado, englobando um amplo conjunto de operações que podem ocorrer em meios manuais ou digitais.

ANVISA - Na Resolução da Diretoria Colegiada - RDC Nº 185 de 2001 foi aprovado o regulamento técnico que trata de registro, alteração, revalidação e cancelamento do registro de produtos médicos na ANVISA (BRASIL, 2001). A nota técnica n° 04/2012 esclarece a RDC 185/01 e define produto médico como:

> Produto médico ativo para diagnóstico: Qualquer produto médico ativo, utilizado isoladamente ou em combinação com outros produtos médicos, destinado a proporcionar informações para a detecção, diagnóstico, monitoração ou tratamento das condições fisiológicas ou de saúde, enfermidades ou deformidades congênitas.
>
> Os suportes lógicos (software) que comandam um produto médico ou que tenham influência em seu uso, se enquadrarão automaticamente na mesma classe.
>
> Todos os produtos ativos destinados a controlar ou monitorar o funcionamento de produtos médicos ativos para terapia enquadrados na Classe III ou destinados a influenciar diretamente no funcionamento destes produtos, enquadram-se na Classe III.

RESULTADOS

O principal resultado deste trabalho foi a elaboração de um guia prático para desenvolvimento de softwares voltados para a área da saúde, destinado a profissionais e pesquisadores que não possuem formação prévia na área de tecnologia. O guia foi estruturado de forma didática, clara e sequencial, contemplando as seguintes etapas:

- Identificação do problema e definição dos objetivos do software;
- Pesquisa de mercado e análise de requisitos funcionais e não funcionais;
- Planejamento do projeto, incluindo definição de metodologia, cronograma e recursos;
- Design de interface e experiência do usuário (UI/UX), com foco na usabilidade;
- Desenvolvimento e escolha das tecnologias mais adequadas;
- Testes de usabilidade, testes unitários, de integração e de aceitação;
- Implantação e validação no ambiente real, incluindo estratégias de lançamento gradual;
- Capacitação dos usuários e suporte técnico contínuo;
- Monitoramento, manutenção corretiva e evolutiva, com ciclos de feedback;
- Avaliação de impacto e coleta de dados para melhoria contínua.

Além da entrega do guia, este trabalho proporcionou reflexões e amadurecimento sobre os desafios enfrentados por pesquisadores da área da saúde na concepção de soluções tecnológicas. Como resultado prático, a autora participou de programas de incentivo à inovação tecnológica, como Ocean Lab, Ocean Launch (Samsung) e catalisa SEBRAE, o que culminou na criação da startup Symmety AI, atualmente em processo de formalização e captação de recursos.

O guia gerado será disponibilizado, de forma gratuita, aos discentes do PPGRACI/UFAM e de outros programas interessados, buscando fomentar a produção de softwares como produtos técnicos dentro dos mestrados profissionais.

CONCLUSÃO

O desenvolvimento deste guia prático cumpre seu propósito de servir como ferramenta orientadora para profissionais da saúde interessados em transformar problemas do seu cotidiano em soluções tecnológicas viáveis. Através de uma linguagem acessível e uma abordagem passo a passo, o material facilita o entendimento sobre os processos de desenvolvimento de softwares, desde a concepção da ideia até a sua implementação e manutenção.

Este trabalho contribui significativamente para reduzir as barreiras entre os campos da saúde e da tecnologia, oferecendo suporte prático para que pesquisadores, mesmo sem formação em áreas exatas, possam liderar ou colaborar efetivamente em projetos de desenvolvimento de softwares.

Além disso, o impacto deste produto transcende o ambiente acadêmico, estimulando o empreendedorismo, a inovação e a transformação digital na área da saúde. A expectativa é de que o guia fomente a geração de produtos aplicáveis, escaláveis e que atendam às reais demandas dos serviços de saúde, fortalecendo a missão do PPGRACI e de outros programas de pós-graduação na formação de profissionais mais preparados para os desafios contemporâneos.

Por fim, este trabalho se consolida como uma proposta inovadora, alinhada às tendências da saúde digital, e reafirma a importância de integrar conhecimentos interdisciplinares na busca por soluções que promovam melhorias na qualidade do cuidado, na gestão dos serviços e na formação profissional.

REFERÊNCIAS

ALFLEN, N. C.; PRADO, E. P. V. Técnicas de elicitação de requisitos no desenvolvimento de software: uma revisão sistemática da literatura. *Open Journal Systems*, [S. l.], 2018.

BARRA, S. et al. Métodos para desenvolvimento de aplicativos móveis em saúde: revisão integrativa da literatura. *Texto & Contexto - Enfermagem*, Florianópolis, v. 26, n. 4, 2017. DOI: https://doi.org/10.1590/0104-07072017002260017.

BARROS, M. A. *Monitoramento e desempenho de aplicações com Prometheus e Grafana*. Rio de Janeiro: Novatec, 2020.

BASHSHUR, R. L.; SHANNON, S. E.; SMITH, C. A evidência empírica para disciplinas de telemedicina em serviços de saúde mental e comportamental: uma revisão sistemática. *Journal of Telemedicine and Telecare*, [S. l.], v. 21, n. 4, p. 257-266, 2015.

BECK, K. et al. *Manifesto Ágil para Desenvolvimento de Software*. 2001. Disponível em: https://agilemanifesto.org/iso/ptbr/. Acesso em: 5 jun. 2025.

BEDNORZ, P.; JYLHÄVÄ, J.; RAITANEN, J. Use of Electronic Medical Records (EMR) in Gerontology: Benefits, Considerations and a Promising Future. *Clinical Interventions in Aging*, [S. l.], v. 18, p. 123–135, 2023. DOI: https://doi.org/10.2147/CIA.S38152074. Acesso em: 5 jun. 2025.

BRASIL. Ministério da Saúde. **Portaria nº 1.184, de 5 de setembro de 2023**. Institui a Política de Proteção de Dados Pessoais no âmbito da ANVISA. *Diário Oficial da União: seção 1*, Brasília, DF, 6 set. 2023.

CAPÉU VERMELHO. O que é a segurança da cadeia de suprimentos de software? *Chapéu Vermelho*, 2020. Disponível em: https://www.chapeuvermelho.com.br/seguranca-da-cadeia-de-suprimentos-de-software/. Acesso em: 10 mar. 2024.

CHACON, S.; STRAUB, B. *Pro Git*. 2. ed. Rio de Janeiro: Novatec, 2014.
DEITEL, P. J.; DEITEL, H. M. *Java: como programar*. 10. ed. São Paulo: Pearson, 2015.

DORSEY, M. *Goal Setting: How to Create an Action Plan and Achieve Your Goals*. 2. ed. Boston: Harvard Business, 2015.

FITZGERALD, B. et al. *Software Support and Maintenance: A Guide for IT Managers*. Londres: Springer, 2013.

FOWLER, M.; HIGHSMITH, J. O Manifesto Ágil. *Aliança Ágil*, 2001. Disponível em: https://agilemanifesto.org/. Acesso em: 5 jun. 2025.

GARRETT, J. J. *The Elements of User Experience: User-Centered Design for the Web and Beyond*. Berkeley: New Riders, 2011.

GIGER, M. L.; KARSSEMEIJER, N.; ARMATO, S. G. Computer-aided diagnosis in medical imaging. *IEEE Transactions on Medical Imaging*, v. 20, n. 12, p. 1205–1208, 2001.

HALAMKA, J. D.; SZOLOVITS, P. O futuro da saúde: como a tecnologia mudará a maneira como prestamos cuidados. *Journal of Health Engineering*, 2013, p. 1–10.

JIANG, F. et al. Artificial intelligence in healthcare: past, present and future. *Stroke and Vascular Neurology*, v. 2, n. 4, p. 230–243, 2017.

JORGESON, M. *Software Testing: A Craftsman's Approach*. 4. ed. Boca Raton: CRC Press, 2013.

KATZ, M. L.; SHAPIRO, C. Competição de sistemas e efeitos de rede. *Journal of Economic Perspectives*, v. 8, n. 2, p. 93–115, 1994.

KENDALL, K. E.; KENDALL, J. E. *Análise e Projeto de Sistemas*. 8. ed. São Paulo: Pearson, 2010.

KIM, G.; DEBOIS, P.; WILLIS, J. *O Manual DevOps: Como criar agilidade, confiabilidade e segurança de classe mundial em organizações de tecnologia*. [S. l.]: Imprensa da Revolução de TI, 2016.

KUO, T. T.; KIM, H. E.; OHNO-MACHADO, L. Blockchain applications in health care for COVID-19 and beyond: a systematic review. *The Lancet Digital Health*, [S. l.], v. 3, n. 12, p. e819–e829, 2021. DOI: https://doi.org/10.1016/S2589-7500(21)00210-7.

KVEDAR, J.; COYE, M. J.; EVERETT, W. Saúde conectada: uma revisão das evidências. *Journal of General Internal Medicine*, v. 29, n. 4, p. 587–595, 2014.

LUTZ, M. *Aprendendo Python*. 5. ed. Sebastopol: O'Reilly Media, 2013.

MACHADO, R. S. Desenvolvimento ágil de software: uma revisão sistemática da literatura. 2018. Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2018.

MAZZOCHI, D.; TRAIANO, V. Inteligência artificial: um conceito futurista no diagnóstico odontológico. *Voos: Revista Polidisciplinar*, v. 10, n. 3, p. 155-179, 2021.

MITCHELL, T. M. *Aprendizado de máquina*. São Paulo: McGraw-Hill, 1997.

NASCIMENTO NETO, F. do et al. Inteligência artificial e novas tecnologias em saúde: desafios e perspectivas. *Brazilian Journal of Development*, v. 6, n. 2, p. 9431–9445, 2020.

NIELSEN, J.; MOLICH, R. Heuristic evaluation of user interfaces. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1990. p. 249–256.

OLIVEIRA, M. *Desenvolvimento de software: práticas e metodologias*. São Paulo: Tech, 2019.

OLIVEIRA, R. A. et al. Segurança e privacidade no desenvolvimento de software para a saúde: uma revisão integrativa. *Journal of Health Informatics*, v. 11, n. 2, p. 45–56, 2019.

ORGANIZAÇÃO MUNDIAL DA SAÚDE. Relatório global sobre inteligência artificial na saúde. Genebra: OMS, 2021.

PEARSON, T.; TOPOL, E. *Deep Medicine: How Artificial Intelligence Can Make Healthcare Human Again*. Nova York: Basic Books, 2019.

PRESSMAN, R. S.; MAXIM, B. R. *Engenharia de software: uma abordagem profissional*. 9. ed. Porto Alegre: McGraw-Hill, 2020.

PUCRS. Computação em nuvem: conceito e benefícios. Porto Alegre, 2024. Disponível em: https://www.pucrs.br/blog/computacao-em-nuvem/. Acesso em: 5 jun. 2025.

RODRIGUES, L. et al. Aplicação da computação em nuvem em pequenas e médias empresas: revisão sistemática. *Revista Prospectus*, v. 1, n. 1, p. 87–116, ago./fev. 2019.

ROSA, C.; SOUSA, P.; SILVA, J. Inovação em saúde e internet das coisas: um panorama do desenvolvimento científico e tecnológico. *Perspectivas em Ciência da Informação*, v. 25, n. 3, p. 164–181, set. 2020.

ROSMAN, G.; RUS, D.; MEIRELES, O. R. Artificial intelligence in surgery: promises and perils. *Annals of Surgery*, v. 268, n. 1, p. 70–76, 2018.

RUBIN, J.; CHISNELL, D. *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*. 2. ed. Indianapolis: Wiley, 2008.

RUSSELL, S. J.; NORVIG, P. *Inteligência Artificial: Uma Abordagem Moderna*. 3. ed. São Paulo: Prentice Hall, 2010.

SASSA, A.; ALMEIDA, I.; PEREIRA, T. Scrum: uma revisão sistemática da literatura. 2022.

SCHWABER, K.; SUTHERLAND, J. *The Scrum Guide*. 2017. Disponível em: https://scrumguides.org/. Acesso em: 5 jun. 2025.

SILBERSCHATZ, A.; GALVIN, P. B.; GAGNE, G. *Conceitos de sistema operacional*. 10. ed. Rio de Janeiro: Wiley, 2018.

SILVA, J. et al. Tecnologia e saúde: integração de soluções tecnológicas no ambiente clínico. São Paulo: Editora Saúde, 2020.

SILVA, M. A. *Gerência de projetos: metodologias e práticas*. São Paulo: Atlas, 2018.

SILVA, M. J. et al. Aplicativos móveis desenvolvidos para a área da saúde no Brasil: revisão integrativa da literatura. *Revista Brasileira de Enfermagem*, v. 71, n. 3, p. 1234–1242, 2018.

SOMMERVILLE, I. *Engenharia de software*. 9. ed. São Paulo: Pearson, 2016.

SOUZA, R. *Inovação em saúde: o papel da tecnologia na transformação dos serviços de saúde*. São Paulo: Editora Inovação, 2021.

TANENBAUM, A. S.; BOS, H. *Sistemas operacionais modernos*. 4. ed. São Paulo: Pearson, 2015.

ANEXOS

Poder Executivo
Ministério da Educação
Universidade Federal do Amazonas
Faculdade de Medicina

ANUÊNCIA DO(A) ORIENTADOR(A)

Eu Profª. Drª. Silvania da Conceição Furtado, na qualidade de professor (a) orientador(a) credenciado(a) pelo Programa de Pós-graduação em cirurgia (Mestrado profissional) da Universidade Federal do Amazonas – UFAM, me comprometo, a orientar o(a) aluno (a) Tábita Maika Gama de Paiva. Título do projeto de pesquisa: Softwares e aplicativos: da ideia à implementação – um guia prático para melhorar o aprendizado de profissionais da área da saúde.

Linha de Pesquisa: Educação, pesquisa, assistência e inovação em cirurgia.

Manaus, 05/06/2025

gov.br Documento assinado digitalmente
SILVANIA DA CONCEICAO FURTADO
Data: 05/06/2025 11:32:01-0300
Verifique em https://validar.iti.gov.br

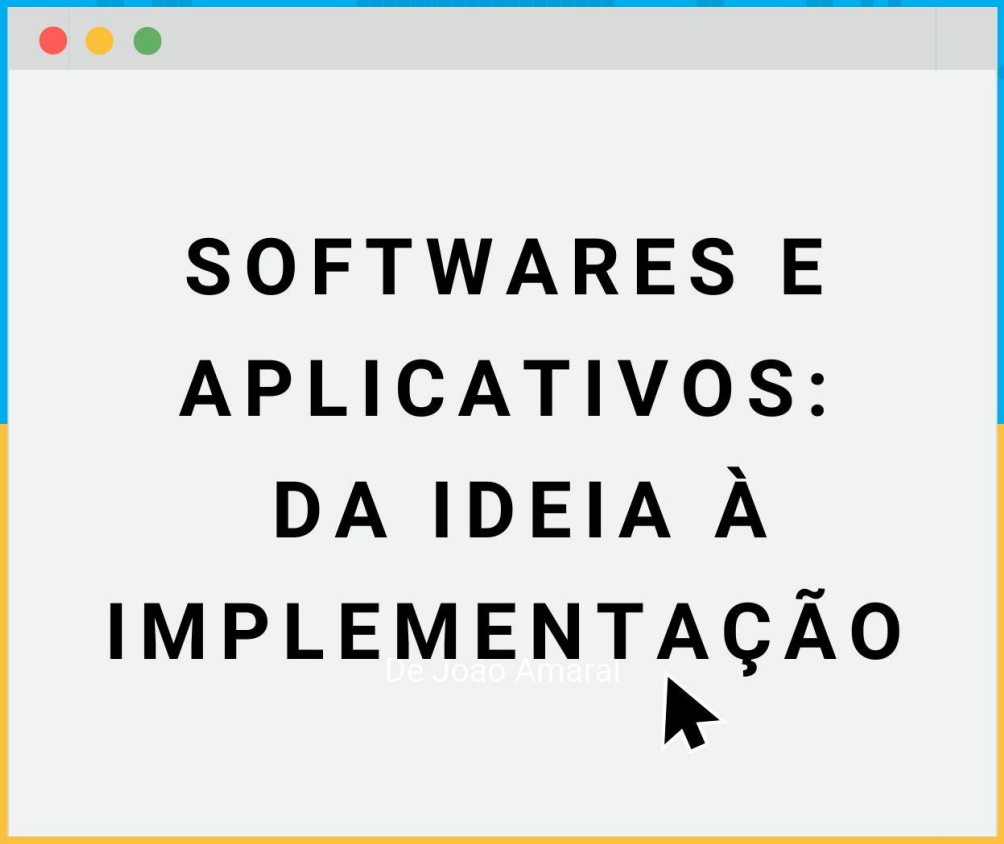
Assinatura e carimbo do(a) Orientador (a)

Revised Standards for Quality Improvement Reporting Excellence (SQUIRE 2.0)
September 15, 2015

Text Section and Item Name	Section or Item Description
Notes to authors	• The SQUIRE guidelines provide a framework for reporting new knowledge about how to improve healthcare • The SQUIRE guidelines are intended for reports that describe system level work to improve the quality, safety, and value of healthcare, and used methods to establish that observed outcomes were due to the intervention(s). • A range of approaches exists for improving healthcare. SQUIRE may be adapted for reporting any of these. • Authors should consider every SQUIRE item, but it may be inappropriate or unnecessary to include every SQUIRE element in a particular manuscript. • The SQUIRE Glossary contains definitions of many of the key words in SQUIRE. • The Explanation and Elaboration document provides specific examples of well-written SQUIRE items, and an in-depth explanation of each item. • Please cite SQUIRE when it is used to write a manuscript.
Title and Abstract	
1. Title	Indicate that the manuscript concerns an initiative to improve healthcare (broadly defined to include the quality, safety, effectiveness, patient-centeredness, timeliness, cost, efficiency, and equity of healthcare)
2. Abstract	a. Provide adequate information to aid in searching and indexing b. Summarize all key information from various sections of the text using the abstract format of the intended publication or a structured summary such as: background, local problem, methods, interventions, results, conclusions
Introduction	*Why did you start?*
3. Problem Description	Nature and significance of the local problem
4. Available knowledge	Summary of what is currently known about the problem, including relevant previous studies

5. Rationale	Informal or formal frameworks, models, concepts, and/or theories used to explain the problem, any reasons or assumptions that were used to develop the intervention(s), and reasons why the intervention(s) was expected to work
6. Specific aims	Purpose of the project and of this report
Methods	*What did you do?*
7. Context	Contextual elements considered important at the outset of introducing the intervention(s)
8. Intervention(s)	a. Description of the intervention(s) in sufficient detail that others could reproduce it b. Specifics of the team involved in the work
9. Study of the Intervention(s)	a. Approach chosen for assessing the impact of the intervention(s) b. Approach used to establish whether the observed outcomes were due to the intervention(s)
10. Measures	a. Measures chosen for studying processes and outcomes of the intervention(s), including rationale for choosing them, their operational definitions, and their validity and reliability b. Description of the approach to the ongoing assessment of contextual elements that contributed to the success, failure, efficiency, and cost c. Methods employed for assessing completeness and accuracy of data
11. Analysis	a. Qualitative and quantitative methods used to draw inferences from the data b. Methods for understanding variation within the data, including the effects of time as a variable
12. Ethical Considerations	Ethical aspects of implementing and studying the intervention(s) and how they were addressed, including, but not limited to, formal ethics review and potential conflict(s) of interest
Results	*What did you find?*
13. Results	a. Initial steps of the intervention(s) and their evolution over time (*e.g.*, time-line diagram, flow chart, or table), including modifications made to the intervention during the project b. Details of the process measures and outcome c. Contextual elements that interacted with the intervention(s) d. Observed associations between outcomes, interventions, and relevant contextual elements e. Unintended consequences such as unexpected benefits, problems, failures, or costs associated with the intervention(s). f. Details about missing data
Discussion	*What does it mean?*
14. Summary	a. Key findings, including relevance to the rationale and specific aims b. Particular strengths of the project

GUIA PRÁTICO PARA MELHORAR O APRENDIZADO DE PROFISSIONAIS DA ÁREA DA SAÚDE



SOFTWARES E APLICATIVOS: DA IDEIA À IMPLEMENTAÇÃO

TÁBITA MAIKA GAMA DE PAIVA
SILVANIA DA CONCEIÇÃO FURTADO
WALDIR SABINO DA SILVA

SOFTWARES E APLICATIVOS: DA IDEIA À IMPLEMENTAÇÃO – UM GUIA PRÁTICO PARA MELHORAR O APRENDIZADO DE PROFISSIONAIS DA ÁREA DA SAÚDE.

AUTORES

Esp. Tábita Maika Gama de Paiva

Especialista em Implantodontia - Faculdades Unidas do Norte de Minas, FUNORTE, Brasil.

Mestranda em cirurgia pelo programa de Pós-graduação em Cirurgia da Faculdade de Medicina da Universidade Federal do Amazonas - PPGRACI -UFAM/AM

Atualização em Cirurgia Oral Menor - UEA/AM

Profª. Dra. Silvania da Conceição Furtado

Doutora em Bases Gerais da Cirurgia - Universidade Estadual Paulista Júlio de Mesquita Filho - UNESP/SP. Professora titular do Programa de Pós-Graduação

em Cirurgia, PPGRACI - Universidade Federal do Amazonas - UFAM/AM

Especialista em Morfologia Humana - Universidade Federal do Amazonas,

UFAM/AM. Aperfeiçoamento em Cirurgia Bucal - Universidade Federal Fluminense - UFF/RJ

Prof. Dr Waldir Sabino da Silva

Doutor em Engenharia Elétrica pela Universidade Federal do Rio de Janeiro e mestre pela mesma instituição.

Graduação em Engenharia Elétrica pela Universidade Federal do Amazonas. Professor associado da Universidade Federal do Amazonas (UFAM) e professor do programa de pós-graduação (Mestrado e Doutorado) em Engenharia Elétrica da UFAM. Participa como pesquisador do Centro de Pesquisa e Desenvolvimento de Tecnologia Eletrônica e da Informação (CETELI/UFAM).

SUMÁRIO

PREFÁCIO

Vivemos em uma era em que a tecnologia está rapidamente transformando todos os aspectos da vida humana, inclusive a área da saúde. Da gestão de dados clínicos ao desenvolvimento de diagnósticos mais precisos, as soluções digitais estão moldando o presente e o futuro das práticas de saúde. No entanto, para muitos profissionais e pesquisadores da área da saúde, o mundo da tecnologia — especialmente o desenvolvimento de software e o uso de inteligência artificial — pode parecer distante, complexo e inacessível. Este guia foi criado justamente para servir de ponte entre esses dois universos.

Este guia não pretende transformar profissionais da saúde em programadores, mas oferecer as ferramentas e o conhecimento necessário para entender o processo de criação de tecnologias digitais voltadas para a saúde. Ao longo dos capítulos, desvendaremos o caminho para projetar, desenvolver e implementar softwares e aplicativos que podem solucionar problemas reais, melhorar o cuidado com o paciente, otimizar processos e aprimorar a tomada de decisão clínica.

Convido alunos de pós-graduação e profissionais da área da saúde a mergulharem neste campo inovador, onde ideias e necessidades de saúde podem ser traduzidas em soluções tecnológicas eficazes. Vamos explorar como a inteligência artificial pode ser usada para melhorar diagnósticos, prever desfechos e personalizar tratamentos, além de discutir como softwares simples e eficientes podem transformar o cotidiano de hospitais, clínicas e laboratórios.

A saúde é um campo rico em desafios e oportunidades. Com o conhecimento certo, você poderá fazer parte dessa revolução tecnológica, criando ferramentas que melhoram não apenas a vida dos pacientes, mas também a qualidade e a eficiência do trabalho dos profissionais da saúde.

Seja bem-vindo a essa jornada de aprendizado e inovação!

Silvania Furtado, Filha, Mãe, Dentista de formação, professora por paixão e, segundo pesquisa com professores brasileiros: Visionária convicta, Professora que apresenta um olhar mais otimista sobre o futuro de tecnologias na educação, menor ambivalência e atitudes negativas sobre tecnologia.

Capítulo 1 – Por que desenvolver um software na saúde?

No dia a dia de quem trabalha na área da saúde, é comum enfrentarmos desafios como: prontuários perdidos, dificuldades na marcação de consultas, demora no atendimento ou problemas na comunicação entre equipes. Muitos desses problemas podem ser resolvidos — ou pelo menos melhorados — com o uso de um bom software.

Mas afinal, **o que é um software?**

Software é um programa de computador. Ele pode ser um aplicativo no celular, um sistema no computador da clínica ou uma ferramenta online. Existem softwares para agendar consultas, armazenar prontuários eletrônicos, organizar filas de atendimento, enviar lembretes para pacientes e muito mais.

Por que um profissional da saúde deveria se envolver com isso?

Você, profissional da saúde, não precisa saber programar. Mas entender **como nasce um software** e **como ele pode ser construído pensando no seu dia a dia** é essencial para que a tecnologia realmente funcione bem e ajude — e não atrapalhe.

Quando quem está na linha de frente participa do desenvolvimento, o resultado é um sistema mais simples, útil e fácil de usar. Um bom software nasce quando os profissionais de saúde e os desenvolvedores trabalham juntos.

Exemplos práticos:

- **Agendamento Online:** Um sistema que permite ao paciente marcar consultas pelo celular. Resultado: menos ligações e menos faltas.
- **Prontuário Eletrônico:** Um lugar digital onde o histórico do paciente fica guardado com segurança, acessível a qualquer momento.
- **Controle de Estoque:** Um software que avisa quando medicamentos estão acabando ou vencendo.
- **Telemedicina:** Sistemas que facilitam atendimentos a distância, especialmente em locais de difícil acesso.

E se eu não souber nada de informática?

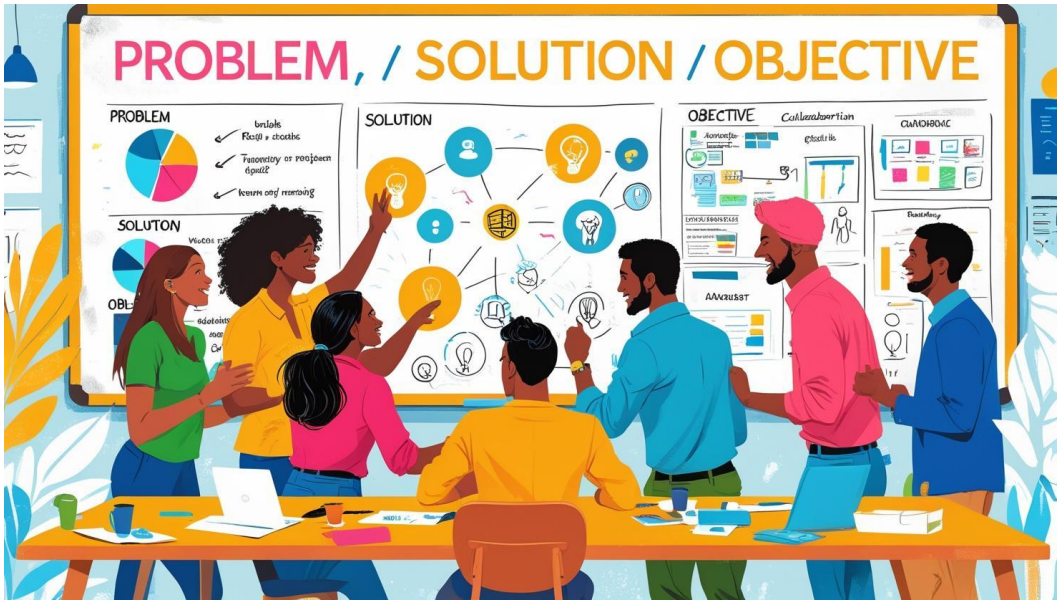
Sem problema! Este guia foi feito justamente para quem **não tem experiência com tecnologia**, mas tem interesse em aprender o básico para transformar uma boa ideia em um sistema útil.

Nos próximos capítulos, você vai aprender passo a passo como pensar, planejar, testar e acompanhar o desenvolvimento de um software. Sempre com exemplos simples e explicações claras.

Você pode não ser um programador, mas é um especialista na sua área — e isso é essencial para que um software funcione bem no mundo real.

Atividade Visual: Complete a tabela abaixo:		
Problema	**Quem é impactado?**	**Como o software pode ajudar?**

Capítulo 2 – Identificação do Problema e Definição dos Objetivos



Antes de começar a desenvolver qualquer software, é preciso saber **exatamente qual problema você quer resolver**. É como quando estamos diante de um paciente: para indicar o tratamento certo, precisamos entender os sintomas e identificar a causa.

Com o software é a mesma coisa. O primeiro passo é observar, refletir e conversar com a equipe para identificar as **dificuldades do dia a dia** que poderiam ser resolvidas com uma solução digital.

Como identificar um problema?

Você pode começar se fazendo perguntas como:

- O que causa atrasos ou confusão na minha rotina?
- Em que momentos perco muito tempo com tarefas manuais?
- O que gera insatisfação entre os pacientes ou a equipe?
- O que poderia ser mais organizado?

- **Exemplo prático:**

Na recepção de uma clínica, a secretária anota os agendamentos à mão. Às vezes há erros, horários duplicados ou pacientes esquecem das consultas. Isso é um problema claro — e que um software simples de agendamento poderia ajudar a resolver.

Como definir os objetivos?

Depois de identificar o problema, você precisa definir o que espera que o sistema faça. Ou seja, seus **objetivos**.

Para facilitar, existe uma técnica muito usada chamada **SMART**. Cada letra representa um ponto importante que o objetivo deve ter:

Letra	Significado	O que isso quer dizer na prática?
S	Específico	O objetivo deve ser claro: o que exatamente você quer?
M	Mensurável	Precisa ser possível medir o resultado.
A	Alcançável	Deve ser realista, possível com os recursos que você tem.
R	Relevante	Tem que fazer sentido e realmente ajudar no seu dia a dia.
T	Temporal	Deve ter um prazo para ser feito.

Exemplo de objetivo SMART:

"Criar um sistema simples de agendamento online para pacientes da clínica, com lembrete automático por WhatsApp, para reduzir faltas em 30% em 3 meses."

Esse objetivo é:

- **Específico:** Agendamento com lembrete via WhatsApp.
- **Mensurável:** Reduzir faltas em 30%.
- **Alcançável:** Um sistema simples, sem complicação.

- **Relevante:** Diminui faltas e melhora a organização.
- **Temporal:** Prazo de 3 meses.

Dica prática

Você pode fazer uma pequena tabela ou lista com:

- O problema identificado
- A quem esse problema afeta
- Como ele impacta o trabalho
- Qual solução você imagina
- Qual seria o objetivo principal do software

Isso ajuda muito na hora de conversar com quem vai desenvolver o sistema.

Pronto! Com o problema identificado e os objetivos bem definidos, você já tem as bases para começar a planejar seu software de forma clara e organizada.

Atividade Visual: Mapa do Problema
Problema central: _____
Quem é afetado: _____
Objetivos do software: 1. _____ 2. _____ 3. _____

Capítulo 3 – Pesquisa e Análise de Mercado

Antes de investir tempo e recursos na criação de um software, é essencial realizar uma **pesquisa de mercado**. Isso permite entender se já existem soluções semelhantes, quais são suas limitações, e quais oportunidades ainda estão em aberto.

1. O que é uma pesquisa de mercado?

É o processo de **coletar e analisar informações** sobre outros softwares que já existem e que atendem (ou tentam atender) o mesmo problema que você identificou. A pesquisa ajuda a responder perguntas como:

- Já existe um software que resolve esse problema?
- Ele é fácil de usar?
- Atende bem aos usuários?
- Quanto custa?
- O que os usuários reclamam ou elogiam?

2. Tipos de pesquisa que podem ser feitos

a) Pesquisa online

Busque em sites de clínicas, hospitais, empresas de tecnologia em saúde, marketplaces de aplicativos (como Google Play e App Store). Leia avaliações e veja vídeos demonstrando o uso.

b) Entrevistas com colegas

Converse com outros profissionais da saúde que usam sistemas. Pergunte:

- O que funciona bem?
- O que atrapalha?
- O que poderia ser melhor?

c) Testes de sistemas existentes

Se possível, solicite versões de demonstração (trial) de softwares comerciais para avaliar suas funcionalidades, usabilidade e limitações.

3. Análise de concorrentes

Depois de reunir dados, liste os principais concorrentes (softwares que resolvem problemas parecidos) e analise:

Item	**Descrição**
Nome do software	Nome e empresa responsável
Funcionalidades principais	O que o sistema faz
Pontos fortes	O que ele faz bem (usabilidade, velocidade, etc.)
Pontos fracos	O que usuários reclamam (complexidade, custo, etc.)
Preço	Licença, assinatura mensal, custo por usuário
Público-alvo	Quem usa (clínicas, consultórios, hospitais, etc.)

4. Análise de requisitos

Após entender o mercado, o próximo passo é definir os **requisitos** do seu software, ou seja, **o que ele deve fazer**.

Existem dois tipos:

- **Requisitos funcionais:** São as funções básicas. Exemplo: “Registrar pacientes”, “Agendar consulta”, “Emitir receitas”.
- **Requisitos não funcionais:** São características técnicas. Exemplo: “Deve funcionar em celulares Android”, “Deve carregar as informações em até 2 segundos”.

5. Envolvimento de stakeholders

Stakeholders são todas as pessoas que usarão ou serão impactadas pelo software. Envolver esses grupos no início ajuda a identificar necessidades reais.

Exemplo de stakeholders:

- Médicos
- Enfermeiros
- Secretárias
- Gestores administrativos
- Pacientes

Aplicar questionários, entrevistas ou rodas de conversa com esses grupos ajuda a coletar sugestões e evitar falhas no planejamento.

A pesquisa e análise de mercado garantem que o software seja útil, competitivo e adaptado à realidade da saúde. Com base nesses dados, o projeto segue para o planejamento técnico e operacional.

Atividade Visual: Tabela Comparativa de Softwares	
Software	
Funcionalidades	
Pontos Fortes	
Pontos Fracos	
O que farei diferente	

Capítulo 4 – Planejamento do Projeto

Depois de entender o problema, definir os objetivos e conhecer o mercado, é hora de **organizar o plano de ação**. O planejamento é como montar o “mapa” que vai guiar todas as etapas do desenvolvimento do software, desde a ideia até a entrega final.

1. Escolhendo a metodologia de desenvolvimento

Existem diferentes formas de organizar o desenvolvimento de um software. As mais comuns são:

Metodologia Ágil (Scrum / Kanban)

Ideal para projetos que vão sendo ajustados ao longo do caminho. A equipe trabalha em ciclos curtos (chamados de “sprints”), entregando partes do sistema pouco a pouco.

Vantagens:

- Entregas mais rápidas
- Facilidade para corrigir erros no meio do caminho
- Participação ativa dos usuários

Metodologia Tradicional (Waterfall)

Segue uma ordem fixa: planejar, desenvolver, testar e entregar. Tudo é feito em sequência, como uma “cascata”.

Vantagens:

- Mais controle sobre o escopo
- Boa para projetos com poucos imprevistos

Qual escolher?

Para projetos simples ou em fase inicial, a metodologia ágil costuma ser mais flexível e eficiente.

2. Montando a equipe

Você pode não ter uma equipe de tecnologia completa, mas é importante saber **quem será responsável por cada etapa**. Algumas funções comuns:

Função	**Responsabilidade**
Profissional da saúde	Define necessidades e valida funcionalidades
Desenvolvedor	Cria o sistema (programação)
Designer	Cuida da aparência e usabilidade do sistema
Testador	Verifica se o sistema funciona como deveria

Mesmo que você contrate profissionais externos, é importante acompanhar de perto cada etapa.

3. Definindo prazos e etapas

Divida o projeto em **fases**. Isso ajuda a manter o controle e a motivação. Exemplo de cronograma simplificado:

Fase	Duração estimada
Levantamento de requisitos	1 semana
Design da interface (protótipos)	1 a 2 semanas
Desenvolvimento inicial	2 a 4 semanas
Testes e ajustes	1 a 2 semanas
Lançamento	1 semana

Lembre-se de **reservar tempo** para imprevistos e ajustes.

4. Recursos necessários

Considere os recursos que você vai precisar:

- **Financeiros:** quanto vai custar o desenvolvimento?
- **Humanos:** quem vai participar?
- **Tecnológicos:** computadores, internet, licenças de software, ferramentas de design ou programação

Mesmo que o projeto seja simples, é importante **ter clareza dos custos e da disponibilidade de tempo da equipe**.

5. Ferramentas que ajudam no planejamento

Aqui vão algumas ferramentas fáceis de usar e que podem ajudar no planejamento:

- **Trello ou Notion:** Para organizar tarefas e acompanhar o progresso
- **Google Planilhas:** Para fazer cronogramas e orçamentos
- **Canva:** Para criar protótipos ou rascunhos da interface
- **Chat com a equipe:** WhatsApp, Slack ou Google Chat para comunicação rápida

Planejar bem é essencial para que o projeto siga no caminho certo. Um bom planejamento **evita desperdícios**, facilita a comunicação e aumenta as chances de sucesso.

4. Análise de requisitos

Após entender o mercado, o próximo passo é definir os **requisitos** do seu software, ou seja, **o que ele deve fazer**.

Existem dois tipos:

- **Requisitos funcionais:** São as funções básicas. Exemplo: “Registrar pacientes”, “Agendar consulta”, “Emitir receitas”.
- **Requisitos não funcionais:** São características técnicas. Exemplo: “Deve funcionar em celulares Android”, “Deve carregar as informações em até 2 segundos”.

5. Envolvimento de stakeholders

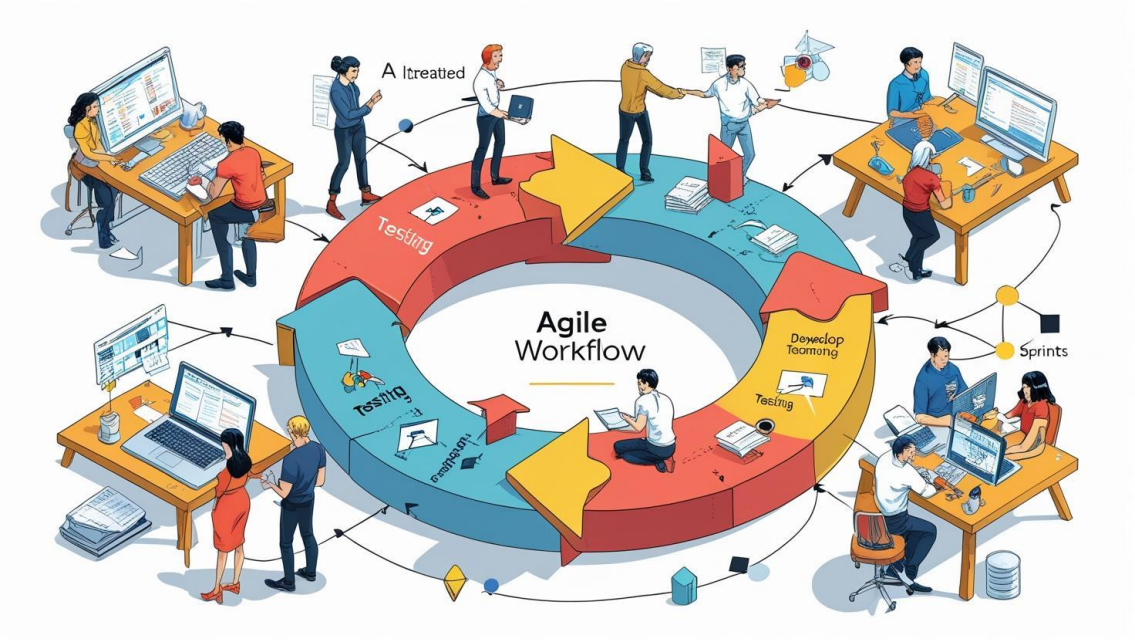
Stakeholders são todas as pessoas que usarão ou serão impactadas pelo software. Envolver esses grupos no início ajuda a identificar necessidades reais.

Exemplo de stakeholders:

- Médicos
- Enfermeiros
- Secretárias
- Gestores administrativos
- Pacientes

Aplicar questionários, entrevistas ou rodas de conversa com esses grupos ajuda a coletar sugestões e evitar falhas no planejamento. A pesquisa e análise de mercado garantem que o software seja útil, competitivo e adaptado à realidade da saúde. Com base nesses dados, o projeto segue para o planejamento técnico e operacional.

Capítulo 4.1 – Entendendo a Metodologia Ágil



Desenvolver software pode parecer complicado, mas existem formas organizadas de tornar esse processo mais leve e adaptável. Uma dessas formas é a **metodologia ágil**, que tem sido muito usada em projetos modernos, inclusive na área da saúde.

O que é metodologia ágil?

A metodologia ágil é uma forma **flexível e colaborativa** de desenvolver software. Ao invés de tentar planejar tudo com perfeição antes de começar, o projeto é dividido em **pequenas etapas**, chamadas de **sprints**. A cada sprint, o time entrega uma parte funcional do sistema, que já pode ser testada e avaliada.

Por que usar a metodologia ágil na saúde?

Porque ela **permite mudanças no caminho**, algo comum quando lidamos com realidades clínicas e hospitalares. À medida que o software vai sendo construído, os profissionais da saúde podem testar, dar feedback e sugerir melhorias. Isso evita que o sistema seja lançado com problemas ou que não atenda bem os usuários.

Como funciona na prática?

A metodologia ágil pode ser organizada com base no **Scrum**, que é um dos modelos mais usados. Veja os principais componentes:

1. Sprint

É um ciclo de trabalho curto (geralmente de 1 a 2 semanas), onde o time trabalha em tarefas específicas. Ao final, entrega-se uma parte do software funcionando.

2. Backlog

É uma lista com tudo que o software precisa fazer (funcionalidades, correções, melhorias). Antes de cada sprint, escolhem-se os itens mais importantes dessa lista para trabalhar.

3. Reuniões rápidas (Daily)

Todos os dias, o time faz uma reunião de 10-15 minutos para alinhar o que cada um está fazendo, identificar dificuldades e garantir que tudo está no caminho certo.

4. Papéis principais

Papel	**Função**
Product Owner	Representa o cliente/usuário (pode ser você, como profissional da saúde). Define prioridades.
Scrum Master	Ajuda o time a seguir a metodologia, resolve obstáculos.
Time de desenvolvimento	Pessoas que vão construir o software (programadores, designers etc.).

5. Review e Retrospectiva

Ao final de cada sprint:

- **Review:** o time apresenta o que foi feito e coleta feedback.
- **Retrospectiva:** o time discute o que funcionou bem e o que pode melhorar no próximo ciclo.

Exemplo prático – Aplicando ágil em um software de agendamento médico

1. **Semana 1 – Sprint 1**
    - Tarefa: Criar tela de login e cadastro
    - Entrega: Versão simples com campos de e-mail e senha
    - Feedback: Os profissionais pedem botão de "Esqueci minha senha"
2. **Semana 2 – Sprint 2**
    - Tarefa: Tela de agendamento e botão de recuperar senha
    - Entrega: Usuário pode escolher especialidade, data e hora
    - Feedback: Médicos sugerem filtros por tipo de convênio
3. **Semana 3 – Sprint 3**
    - Tarefa: Ajustar filtros e melhorar calendário
    - Entrega: Interface mais intuitiva, pronta para testes reais

Ferramentas que facilitam o uso da metodologia ágil

- **Trello ou Jira:** para organizar o backlog e acompanhar sprints
- **Google Meet ou Zoom:** para reuniões rápidas e revisões
- **Miro:** para mapas de ideias, fluxogramas e brainstorming visual
- **WhatsApp ou Slack:** para comunicação direta entre os envolvidos

Vantagens da metodologia ágil

- **Maior participação dos profissionais da saúde**
- **Rapidez na entrega de resultados**
- **Facilidade para corrigir erros e ajustar o sistema**
- **Mais qualidade no produto final**

Atividade Visual: Linha do Tempo do Projeto

Crie uma linha com as fases principais e marque quem será responsável.

Use uma régua, papel, ou ferramenta digital

Exemplo:

Ideia → Pesquisa → Planejamento → Design → Desenvolvimento → Testes → Lançamento

Sugestão de leitura: “A arte de fazer o dobro do trabalho na metade do tempo” de Jeff Sutherland.

Capítulo 5 – Design e Prototipagem

Depois de planejar e organizar o projeto, chega o momento de começar a pensar na **aparência e na experiência do sistema**, ou seja, como ele será visualmente e como os usuários irão utilizá-lo. Essa etapa é chamada de **design e prototipagem**.

O que é o design de um software?

O design de um software envolve a **criação das telas, botões, menus e caminhos de navegação** que o usuário verá ao usar o sistema. É como projetar o layout de um consultório: você pensa onde vai colocar cada item para facilitar a vida de quem vai trabalhar ali.

Dois conceitos importantes: UI e UX

1. UI (Interface do Usuário)

É o “visual” do sistema – como ele aparece para o usuário. Isso inclui:

- Cores
- Fontes
- Ícones
- Botões
- Organização da tela

Um bom UI precisa ser **limpo, bonito e fácil de entender**.

2. UX (Experiência do Usuário)

É a **sensação** que a pessoa tem ao usar o sistema. O foco é tornar tudo **simples, rápido e confortável**.

Exemplo:

Se um paciente precisa de 10 cliques para agendar uma consulta, a UX está ruim. Mas se consegue agendar com 3 cliques, a UX está boa.

O que é um protótipo?

Um **protótipo** é como um rascunho interativo do sistema. Ele simula as telas e botões, mas ainda **não funciona de verdade**. Serve para mostrar como o sistema será antes de programá-lo.

Você pode criar protótipos com ferramentas simples e gratuitas, como:

- **Figma** (recomendado, fácil de usar e gratuito)
- **Adobe XD**
- **Balsamiq**
- **Pen & papel** (sim! até desenhos feitos à mão servem para prototipar)

Por que é importante prototipar?

- Evita retrabalho: é mais fácil corrigir um rascunho do que refazer um sistema pronto.
- Ajuda a testar ideias com os usuários antes de programar.
- Facilita a comunicação entre profissionais da saúde e os desenvolvedores.

Exemplo prático – Tela de agendamento

Imagine que você está desenvolvendo um sistema de agendamento de consultas para uma clínica. Veja como você poderia prototipar:

1. Tela de login
    - Campos: e-mail e senha
    - Botão: “Entrar” e “Esqueci minha senha”
2. Tela inicial do médico
    - Opções: “Ver consultas de hoje”, “Agendar nova consulta”, “Perfil”

3. Tela de agendamento
    - Campos: nome do paciente, especialidade, data, horário
    - Botão: “Confirmar agendamento”

Com isso pronto, você pode **mostrar para colegas da clínica** e perguntar:

- Está fácil de usar?
- Os botões estão claros?
- A sequência faz sentido?

Esse retorno é valioso e pode evitar muitos erros mais adiante.

Dica: Teste com pessoas reais

Antes de seguir para a fase de desenvolvimento, **mostre os protótipos para pessoas que realmente vão usar o sistema**:

- Médicos
- Enfermeiros
- Secretárias
- Pacientes (se for um app voltado a eles)

Peça que tentem fazer uma tarefa simples, como marcar uma consulta. Observe se conseguem sem dificuldades. Se tiverem dúvidas, anote e melhore o protótipo.

Resumo do Capítulo

- O design cuida da aparência (UI) e da experiência (UX) do software.
- Protótipos ajudam a visualizar e testar o sistema antes de programar.
- Ferramentas como Figma e desenhos manuais podem ser usados.
- Testes com usuários ajudam a identificar melhorias ainda no início.

Atividade Visual: Desenhe sua tela inicial!
Use esta moldura ou uma folha em branco: Você pode usar papel, Canva, PowerPoint ou Figma.

Nome do Software	
Campo principal 1:	
Campo principal 2:	
Botão Salvar	Botão Cancelar

Capítulo 6 – Desenvolvimento

Agora que você já sabe **o que o sistema deve fazer** e como ele deve **aparecer para o usuário**, chegou a hora de transformar tudo isso em realidade. Essa etapa é chamada de **desenvolvimento**, ou seja, **programar o sistema para que ele funcione de verdade**.

Mesmo que você não vá programar diretamente, é importante entender o que acontece nessa fase para poder **acompanhar e orientar os desenvolvedores**.

O que acontece no desenvolvimento?

O time de desenvolvimento escreve o **código-fonte**, que é como o "DNA" do sistema. Esse código é o que permite que:

- os botões funcionem;
- os dados sejam salvos no sistema;
- os usuários façam login;
- os relatórios sejam gerados;
- e muito mais.

Quais tecnologias são usadas?

Aqui estão algumas ferramentas comuns que os desenvolvedores podem usar:

O que é feito	Exemplos de ferramentas
Linguagens de programação	Java, JavaScript, Python, Swift (para iOS), Kotlin (para Android)
Frameworks (aceleram o desenvolvimento)	React Native, Flutter, Angular, Laravel
Banco de dados (guarda os dados)	MySQL, PostgreSQL, MongoDB
Controle de versão (para guardar mudanças)	Git (com GitHub ou GitLab)

Você não precisa saber programar, mas é bom conhecer esses termos para poder conversar com os desenvolvedores e fazer boas escolhas.

Boas práticas no desenvolvimento

Mesmo que o software esteja nas mãos dos programadores, sua participação continua sendo essencial! Veja como ajudar:

1. Organize as funcionalidades em prioridades

Nem tudo precisa ser feito de uma vez. É melhor começar com as **funções mais importantes** (como agendamento, cadastro de pacientes) e deixar o restante para depois.

2. Participe das revisões

Peça para ver versões parciais do sistema. Mesmo que ainda não esteja pronto, você pode **testar e sugerir melhorias**.

3. Garanta o uso de boas práticas

Você pode sugerir ao time:

- **Usar controle de versão (Git)** para registrar todas as alterações.

- **Escrever código limpo**, organizado e documentado (isso facilita manutenção).
- **Fazer revisões entre programadores** (chamado de *code review*).

Exemplo prático: Desenvolvimento de um sistema de prontuário eletrônico
Imagine que sua clínica deseja um sistema para registrar atendimentos médicos. As funcionalidades básicas podem incluir:

- Cadastro de pacientes
- Registro de consultas
- Prescrição de medicamentos
- Geração de atestados

O time de desenvolvimento começa a programar essas funcionalidades, uma por uma. A cada nova entrega, você testa e dá feedback.
Exemplo de progresso:

- Semana 1: Login funcionando
- Semana 2: Cadastro de paciente
- Semana 3: Registro de consulta

Segurança é prioridade
Na área da saúde, **dados dos pacientes são extremamente sensíveis**. Por isso, é obrigatório que o software:

- **Tenha login com senha forte**
- **Criptografe os dados**
- **Evite acesso indevido**
- **Esteja em conformidade com a LGPD** (Lei Geral de Proteção de Dados)

Converse com o time sobre essas exigências desde o início.

Resumo do Capítulo

- A fase de desenvolvimento transforma ideias em um sistema real.
- Envolve linguagens de programação, bancos de dados e outras ferramentas.
- É importante priorizar funcionalidades, revisar entregas e garantir segurança.
- Você, como profissional da saúde, deve acompanhar e colaborar com o time técnico.

Atividade Visual: Fluxograma Simples da Funcionalidade

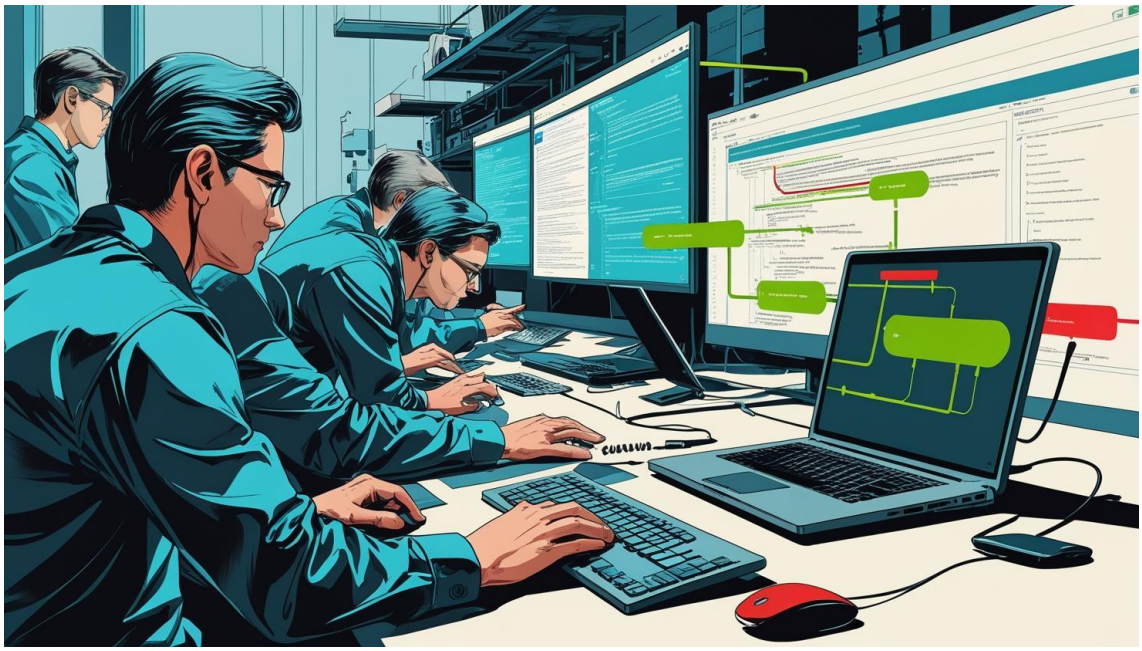
Desenhe a sequência de passos para uma ação, como “Cadastrar paciente”

Exemplo:

[Início] → [Preencher dados] → [Clique em Salvar] → [Dados armazenados com sucesso]

Use caixas e setas!

Capítulo 7 – Testes e Validação



Depois que o software está programado, ele **precisa ser testado antes de começar a ser usado no dia a dia**. Essa etapa garante que tudo funcione como esperado e que **nenhum erro prejudique os usuários ou comprometa os dados dos pacientes**.

Por que testar é tão importante?

Imagine lançar um sistema de agendamento e os pacientes não conseguirem marcar consultas. Ou então, o prontuário não salvar corretamente os dados. Isso pode causar **problemas graves**, especialmente em ambientes de saúde.

Testar serve para **evitar falhas**, garantir a **qualidade** e aumentar a **confiança** dos profissionais e pacientes no sistema.

Tipos de testes que você precisa conhecer

Você não precisa realizar os testes técnicos, mas é importante saber que eles existem e como cada um contribui para a segurança e o bom funcionamento do sistema:

Tipo de Teste	O que verifica	Exemplo prático
Teste unitário	Verifica se cada parte do sistema funciona isoladamente	Um botão de "Salvar Consulta" realmente salva os dados?
Teste de integração	Verifica se as partes do sistema funcionam bem juntas	O cadastro do paciente aparece corretamente na agenda?
Teste de usabilidade	Verifica se o sistema é fácil de usar	Usuários entendem como agendar uma consulta?
Teste de aceitação	Verifica se o sistema atende os objetivos definidos	O sistema permite gerar atestados e imprimir relatórios?

A sua participação nos testes

Mesmo que os testes técnicos fiquem por conta dos desenvolvedores, **os testes de usabilidade e aceitação devem ser feitos com você e sua equipe**, porque só vocês conseguem avaliar se o sistema está adequado à realidade da clínica ou hospital.

Exemplo de teste de usabilidade:

- **Objetivo:** Verificar se um recepcionista consegue agendar uma consulta em menos de 3 minutos.
- **Passo a passo:**
    1. Abrir o sistema
    2. Clicar em “Novo agendamento”
    3. Escolher o médico e o horário
    4. Confirmar o paciente
- **Resultado esperado:** O agendamento deve ser concluído com facilidade e rapidez.

Exemplo de teste de aceitação:

- “Cadastrar um novo paciente e conferir se os dados aparecem corretamente no prontuário.”
- “Gerar um relatório de atendimentos e conferir se ele contém os dados certos.”

Você pode usar **listas de verificação (checklists)** para garantir que todas as funções foram testadas.

O que fazer se encontrar erros?

Se algum problema for encontrado:

1. **Anote exatamente o que aconteceu**, com prints (capturas de tela) se possível.
2. **Informe o responsável pelo desenvolvimento.**
3. **Acompanhe a correção** e teste novamente após a atualização.

Repetir os testes é normal. Às vezes, corrigir um erro causa outro. Por isso, o ciclo de testes pode acontecer várias vezes até tudo estar pronto.

Resumo do Capítulo

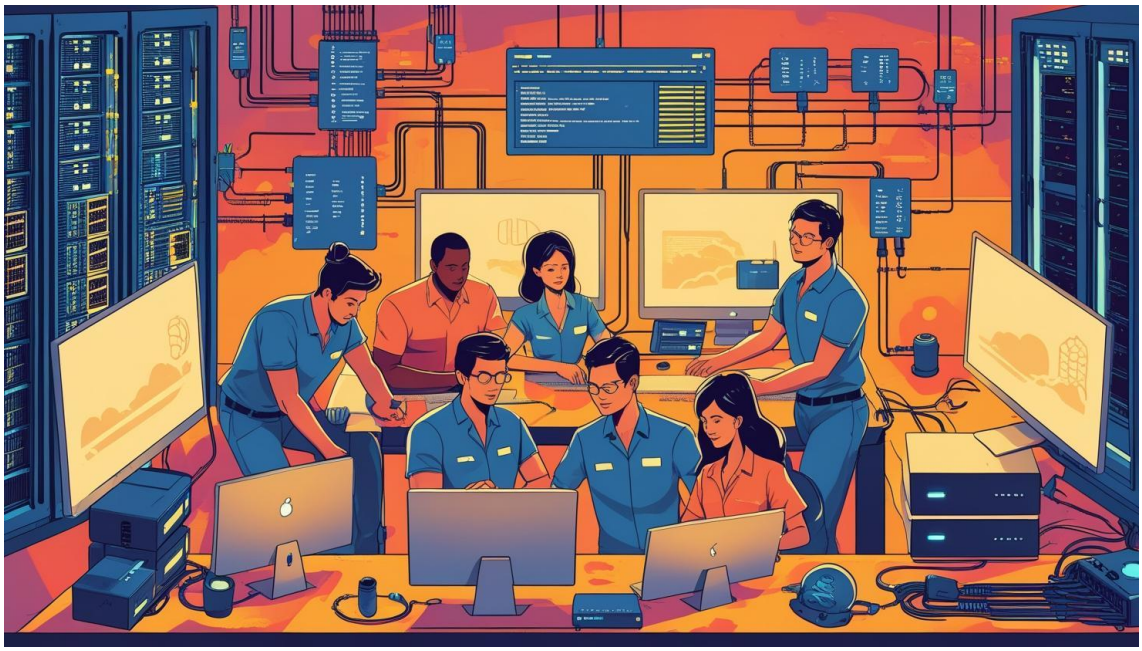
- Testar o sistema antes de lançar é essencial para garantir qualidade e segurança.
- Existem diferentes tipos de testes (unitário, integração, usabilidade e aceitação).
- A equipe de saúde deve participar dos testes com foco na experiência real de uso.
- Encontrar e corrigir falhas antes do lançamento evita prejuízos e aumenta a confiança.

Atividade Visual: Checklist de Testes

Marque com ✓ se estiver funcionando:

Teste	OK?	Observações
Ex: O botão "Salvar" funciona?	☐	
	☐	
	☐	
	☐	

Capítulo 8 – Implantação



O que é a implantação?

Implantar um software significa **preparar o ambiente, instalar o sistema e garantir que ele comece a ser usado com segurança e eficiência**. Essa etapa precisa ser feita com cuidado, pois é quando o sistema passa a fazer parte da rotina de trabalho da equipe.

Etapas principais da implantação

1. Preparar o ambiente

Antes de usar o sistema, é necessário configurar os computadores, internet, servidores e garantir que tudo esteja pronto. Se o software for online (em nuvem), esse processo é mais simples.

Exemplo prático:

- Verificar se os computadores da recepção têm acesso à internet.
- Instalar impressoras para atestados e receitas.
- Configurar o sistema para o fuso horário correto.

2. Implantação gradual

Não é preciso começar usando todas as funções do sistema de uma vez. Uma boa prática é **implantar aos poucos**, testando com um grupo pequeno de usuários.

Exemplo:

- Na primeira semana, usar o sistema só para agendamentos.

- Depois, adicionar prontuários e prescrições.
- Por fim, utilizar os relatórios e ferramentas de gestão.

Essa abordagem ajuda a **identificar problemas com menos risco** e permite que a equipe vá se adaptando aos poucos.

3. Fase de observação

Durante as primeiras semanas de uso, é importante **monitorar o comportamento do sistema e ouvir os usuários**. Assim, ajustes podem ser feitos rapidamente.

Dicas:

- Crie um grupo no WhatsApp ou e-mail só para dúvidas e sugestões.
- Tenha um técnico ou desenvolvedor de plantão nos primeiros dias.
- Faça reuniões rápidas com a equipe para saber o que está funcionando bem ou não.

Como lidar com dificuldades na implantação?

É comum que apareçam dúvidas ou pequenas falhas. O mais importante é:

- Manter a calma;
- Registrar os problemas;
- Comunicar ao suporte ou equipe técnica com clareza;
- Evitar mudanças bruscas na rotina até o sistema estar estável.

Resumo do Capítulo

- A implantação é a etapa de colocar o sistema em uso real.
- É preciso preparar o ambiente e fazer a transição com calma.
- Começar com funções básicas ajuda a reduzir riscos.
- O acompanhamento contínuo evita falhas e melhora a adaptação da equipe.

Atividade Visual: Plano de Implantação Simplificado

Desenhe este esquema:

[Local de uso] → [quem começa a usar] → [primeiros dias: o que observar?]

Exemplo: Clínica ABC → Recepcionistas → Verificar se há lentidão ou travamentos

Capítulo 9 – Treinamento e Suporte

Depois que o sistema foi implantado, **o próximo passo é garantir que todos saibam usá-lo da melhor forma possível**. Um bom treinamento evita erros, aumenta a produtividade e **deixa os profissionais mais confiantes** para usar a tecnologia no dia a dia.

Por que o treinamento é essencial?

Mesmo que o sistema seja fácil de usar, é comum surgirem dúvidas no início. O treinamento serve para:

- Ensinar como usar cada função do sistema;
- Evitar erros que podem comprometer o atendimento;
- Aumentar a eficiência e a segurança no uso do software.

Treinamento por perfil de usuário

Cada profissional da equipe usa o sistema de um jeito. Por isso, o ideal é **personalizar o treinamento** de acordo com as funções de cada grupo:

Grupo	Foco do Treinamento
Médicos e enfermeiros	Prontuário eletrônico, prescrição digital, exames, histórico do paciente
Recepcionistas	Agendamento de consultas, cadastro de pacientes, organização da agenda
Administradores	Relatórios, controle financeiro, acompanhamento de atendimentos e indicadores de gestão

Formas de oferecer o treinamento

Você pode combinar diferentes formas de ensino para facilitar o aprendizado da equipe. Aqui vão algumas opções práticas:

1. Videoaulas

- Curtas e diretas, podem ser assistidas no tempo livre.
- Úteis para rever sempre que surgir uma dúvida.

2. Tutoriais e manuais simples

- Arquivos PDF com imagens e passo a passo.
- Devem ficar disponíveis no computador da recepção ou em uma pasta online.

3. Treinamento presencial ou online ao vivo

- Ideal para explicar pontos mais complexos e tirar dúvidas na hora.
- Pode ser feito com pequenos grupos para melhor aproveitamento.

4. Acompanhamento no uso real

- Ter alguém ajudando durante os primeiros dias de uso real do sistema.
- Isso evita que pequenos erros causem grandes transtornos.

Suporte contínuo: como funciona?

Mesmo com um bom treinamento, dúvidas e problemas podem surgir. Por isso, é fundamental garantir que a equipe tenha **suporte sempre que precisar**.

Formas de suporte comuns:

- **Help Desk ou Chat Online:** atendimento direto com alguém da equipe técnica.
- **Base de conhecimento e perguntas frequentes (FAQ):** site com respostas para dúvidas comuns.
- **Suporte por e-mail ou telefone:** para casos que exigem mais explicação ou urgência.

Dicas para um treinamento eficaz

- Use **linguagem simples**, sem termos técnicos complicados.
- **Mostre exemplos reais**, do dia a dia da clínica ou hospital.
- Incentive os profissionais a **fazerem perguntas e testarem** o sistema durante o treinamento.
- Mantenha um **clima leve**, sem pressão. O aprendizado é mais eficiente assim.

Resumo do Capítulo

- Treinamento é essencial para que todos usem o sistema com segurança e eficiência.
- Cada grupo de usuários deve receber orientações específicas.
- Videoaulas, tutoriais e acompanhamentos práticos ajudam muito no aprendizado.
- O suporte contínuo é indispensável para resolver dúvidas e evitar interrupções no trabalho.

<table>
<tr><th colspan="2">Atividade Visual: Mini guia Visual (Passo a Passo)</th></tr>
<tr><td colspan="2">Crie uma cartilha com ícones ou ilustrações.

Exemplo de layout:</td></tr>
<tr><td>1. Clique em “Novo Paciente”
2. Preencha os dados
3. Clique em “Salvar”

Pronto! O paciente foi cadastrado.</td><td></td></tr>
</table>

Capítulo 10 – Monitoramento e Manutenção

Depois que o sistema está funcionando e os profissionais já sabem usá-lo, **o trabalho ainda não acabou**. Para garantir que tudo continue rodando bem, é necessário **monitorar e manter o software em boas condições** — como se fosse uma revisão periódica.

O que é monitoramento?

Monitorar significa **acompanhar o funcionamento do sistema em tempo real**, para garantir que tudo está certo. Isso inclui:

- Verificar se o sistema está rápido;
- Identificar falhas antes que causem prejuízos;
- Medir o uso de recursos como memória e internet.

Ferramentas que ajudam:

- **Zabbix** e **Prometheus**: usadas por técnicos para monitorar servidores, redes e sistemas.
- **Alertas automáticos**: avisam quando algo sai do normal (ex: sistema fora do ar ou uso de disco cheio).

O que é manutenção?

Manutenção é o conjunto de ações feitas para que o sistema **continue funcionando bem ao longo do tempo**. Inclui:

1. Correção de erros (bugs)

Mesmo com testes, alguns erros só aparecem no uso real. Eles devem ser corrigidos o mais rápido possível.

2. Atualizações

São melhorias feitas no sistema, como:

- Nova função (ex: prontuário com histórico gráfico);
- Melhorias visuais (ex: tela mais limpa e intuitiva);
- Reforços de segurança (proteger dados dos pacientes).

3. Ajustes com base no feedback

Quando os usuários dão sugestões ou relatam dificuldades, o sistema pode ser ajustado para ficar melhor.

Como planejar a manutenção

Assim como consultórios e hospitais têm escalas e agendas, **a manutenção do sistema também deve seguir um plano**:

Frequência	**Ação Recomendada**
Semanal	Verificar desempenho e corrigir pequenos erros
Mensal	Revisar relatórios, ajustar funções usadas com frequência
Trimestral ou Semestral	Atualizar o sistema, implementar melhorias e rever a segurança

Dicas práticas para manter o sistema saudável

- **Tenha um responsável técnico** (interno ou contratado) para cuidar do sistema.
- **Use sistemas de backup automático** para não perder dados importantes.
- **Comunique falhas e melhorias rapidamente** à equipe de suporte.

- **Ouça os usuários** — muitas vezes, pequenos ajustes fazem grande diferença na prática.

Resumo do Capítulo

- Monitorar o sistema é garantir que ele funcione bem o tempo todo.
- A manutenção corrige erros, atualiza funções e mantém o sistema seguro.
- Um bom planejamento de manutenção evita problemas maiores.
- Escutar os usuários e agir rápido é essencial para a continuidade do bom uso.

Atividade Visual: Simulando Problemas e Soluções

Problema Possível	Como eu resolveria?	Tempo estimado

Capítulo 11 – Avaliação e Feedback

Depois de todo o esforço para planejar, desenvolver, implantar e treinar a equipe, é hora de entender **como o sistema está funcionando na prática**. Isso é feito por meio da **avaliação e do feedback dos usuários**.

Esse processo ajuda a perceber o que está dando certo, o que pode melhorar e quais resultados o software está trazendo para o dia a dia da clínica, hospital ou consultório.

O que é feedback?

Feedback é a opinião dos usuários sobre o sistema. Ouvir quem usa o software todos os dias é essencial para:

- Identificar dificuldades que precisam ser corrigidas;
- Descobrir ideias de melhorias;
- Confirmar se o sistema está ajudando na rotina de trabalho.

Formas simples de coletar feedback

Você pode escolher uma ou mais formas de escutar os profissionais que usam o sistema:

1. Pesquisas rápidas

- Um formulário com perguntas simples (ex: “Você está satisfeito com o sistema?”, “O que pode melhorar?”).
- Pode ser feito com papel impresso ou online.

2. Entrevistas

- Conversas diretas com médicos, enfermeiros e recepcionistas.
- Ótimo para entender melhor os detalhes das dificuldades ou sugestões.

3. Caixa de sugestões

- Fisicamente, em um local visível da clínica.
- Ou digital, dentro do próprio sistema.

4. Observação no uso

- Acompanhar o uso do sistema por alguns minutos ajuda a perceber pontos confusos ou funcionalidades pouco utilizadas.

Como fazer a avaliação do impacto do sistema?

Além do que as pessoas dizem, é importante **medir os resultados práticos** do sistema. Algumas formas de fazer isso incluem:

1. Métricas de uso

- Quantas vezes os profissionais acessam o sistema;
- Quais funções são mais utilizadas;
- Tempo médio para realizar tarefas (ex: agendar uma consulta).

2. Eficiência operacional

- O sistema ajudou a reduzir o tempo em tarefas administrativas?
- Houve aumento na produtividade da equipe?

3. Satisfação dos usuários

- Os profissionais sentem que o sistema facilitou o trabalho?
- Os pacientes percebem mais organização e agilidade?

4. Segurança e confiabilidade

- Houve falhas ou perdas de dados?
- O sistema ficou fora do ar em algum momento importante?

O que fazer com as informações coletadas?

Depois de ouvir os usuários e analisar os dados, é hora de **tomar decisões**:

- Corrigir o que não está funcionando;

- Investir nas melhorias mais solicitadas;
- Planejar novas atualizações com base nas necessidades reais da equipe.

Resumo do Capítulo

- Avaliar o sistema e ouvir os usuários garante a melhoria contínua.
- Feedback pode ser feito por pesquisas, entrevistas, caixas de sugestão ou observação direta.
- Medir o impacto do software ajuda a comprovar seus benefícios e identificar ajustes necessários.
- Escutar e agir sobre o feedback é um dos passos mais importantes após a implantação.

Atividade Visual: Formulário Simples para Usuários	
Faça um formulário com emojis ou notas:	
Exemplo: • De 1 a 5, o sistema é fácil de usar? • O que mais gostou? • O que melhoraria? • Voltaria a usar? ( ) Sim ( ) Não • Comentários extras:	

Capítulo 12 – Segurança, Privacidade e LGPD

Quando usamos um sistema de software na área da saúde, lidamos com **informações pessoais e sensíveis de pacientes**, como nome, endereço, exames, diagnósticos e tratamentos. Proteger esses dados é uma **obrigação ética e legal**.

Neste capítulo, você vai entender, de forma simples, o que é necessário para garantir a segurança e a privacidade das informações, respeitando a **LGPD – Lei Geral de Proteção de Dados (Lei nº 13.709/2018)**.

O que é segurança da informação?

Segurança da informação significa proteger os dados contra:

- **Acessos não autorizados** (por pessoas que não deveriam ver);
- **Perdas ou exclusões acidentais**;
- **Ataques cibernéticos**, como vírus e hackers.

Práticas básicas de segurança:

- Usar **senhas fortes** e trocá-las com frequência;
- Controlar quem pode acessar cada parte do sistema;
- Manter o sistema **atualizado e protegido**;
- Fazer **backups regulares** dos dados.

O que é privacidade de dados?

Privacidade significa respeitar o direito das pessoas de saber como os seus dados estão sendo usados. O paciente precisa estar ciente e, na maioria dos casos, **autorizar o uso de seus dados**.

O que é a LGPD?

A LGPD é a lei brasileira que **protege os dados pessoais de todos os cidadãos**. Ela vale para qualquer empresa ou profissional que **coleta, armazena ou compartilha informações pessoais**, inclusive na saúde.

A LGPD exige que:

1. **O paciente saiba** que seus dados estão sendo coletados e para quê;
2. **Seja informado** sobre como e onde os dados serão armazenados;
3. **Tenha direito de acessar, corrigir ou excluir** seus dados quando desejar;
4. **Tenha os dados protegidos** contra vazamentos e acessos indevidos.

Como a LGPD se aplica à saúde?

Na área da saúde, os dados são considerados **sensíveis**, ou seja, exigem cuidados redobrados. A LGPD determina que:

- **O consentimento do paciente** é necessário na maioria dos casos;
- Só devem ser coletados **os dados estritamente necessários**;
- Profissionais devem ter **acesso restrito**, apenas ao que for essencial para seu trabalho;
- A clínica, hospital ou consultório deve **manter registros** de como os dados estão protegidos.

O que acontece se a LGPD for desrespeitada?

Se uma instituição ou profissional não cumprir a LGPD, pode sofrer:

- **Multas** que chegam a milhões de reais;
- Perda de confiança dos pacientes;
- Problemas legais, judiciais e de imagem.

Como estar em conformidade com a LGPD?

Aqui vão algumas dicas práticas para clínicas e profissionais de saúde:

Ação	Descrição
Políticas claras	Tenha documentos simples explicando como os dados dos pacientes são tratados.
Consentimento	Peça autorização por escrito para coletar e usar os dados.
Controle de acesso	Apenas pessoas autorizadas devem acessar informações sensíveis.
Sistemas seguros	Use softwares com criptografia e ferramentas de segurança.
Treinamento da equipe	Ensine todos os colaboradores sobre a importância da privacidade.

O que é a ANVISA?

A **Agência Nacional de Vigilância Sanitária (ANVISA)** é o órgão responsável por regular produtos e serviços que afetam a saúde da população, garantindo sua **segurança, eficácia e qualidade**.

Isso inclui medicamentos, alimentos, cosméticos, equipamentos médicos e, mais recentemente, **softwares com finalidades médicas**.

ANVISA – Quando o Software é um Produto Regulado

A ANVISA (Agência Nacional de Vigilância Sanitária) regula softwares que têm função diagnóstica, terapêutica ou de monitoramento direto da saúde, ou seja, softwares como dispositivos médicos.

Quando seu software deve seguir as normas da ANVISA?

Se ele:

- Faz cálculos clínicos que afetam decisões de tratamento;
- **Monitora sinais vitais**;
- Serve como apoio à decisão médica (ex: alerta de risco de AVC);

- Se conecta com equipamentos médicos.

Nesses casos, o software precisa de registro ou notificação junto à ANVISA, conforme a **RDC nº 751/2022** (que trata de dispositivos médicos).

Boas práticas recomendadas mesmo para softwares não regulados:

- Documentar as funcionalidades do sistema;
- Ter um manual de uso claro e acessível;
- Garantir rastreabilidade de alterações feitas no sistema (logs).

Quando um software é considerado um "dispositivo médico"?

Segundo a RDC nº 751/2022, um software é considerado um Dispositivo Médico (Software as a Medical Device – SaMD) quando ele:

Tem **finalidade médica** (diagnóstico, prevenção, monitoramento, tratamento ou reabilitação);
Atua **de forma autônoma** (sem estar embutido fisicamente em um equipamento);
Pode influenciar diretamente decisões clínicas.

Exemplos de Softwares Regidos pela ANVISA:

- Um app que **analisa eletrocardiogramas e gera alertas automáticos**;
- Um sistema que **recomenda doses de medicamentos** com base em dados clínicos;
- Softwares de **avaliação de risco cirúrgico**;
- Softwares de **interpretação de imagens médicas** (como ressonâncias ou tomografias). **Softwares que não são regulados pela ANVISA:**
- Sistemas de **gestão de clínicas e consultórios (ERP)**;
- Agendas, prontuários eletrônicos ou sistemas administrativos **sem atuação clínica direta**;
- Apps de **educação em saúde** sem função diagnóstica;

- Ferramentas para **telemedicina básica**, quando não envolvem análise clínica automatizada.

Importante: Mesmo que seu software não exija registro na ANVISA, ele **ainda deve seguir a LGPD e boas práticas de segurança digital.**

Resumo do Capítulo

- Segurança e privacidade são fundamentais no uso de sistemas de saúde.
- A LGPD protege os dados dos pacientes e exige responsabilidade de quem coleta e usa essas informações.
- É essencial obter consentimento, proteger os dados e treinar a equipe.
- Respeitar a LGPD é proteger o paciente, o profissional e a instituição.
- Mesmo que seu software não exija registro na ANVISA, ele **ainda deve seguir a LGPD e boas práticas de segurança digital.**

Atividade Visual: Cartaz de Boas Práticas com Dados
Escreva 5 frases curtas, como se fossem regras coladas na parede:
Exemplo: “Não compartilhe informações de pacientes sem autorização”
1. 2. 3. 4. 5.

Glossário de Termos Tecnológicos para Profissionais da Saúde

API (Interface de Programação de Aplicações)

Uma espécie de “ponte” que permite que dois sistemas diferentes conversem entre si. Exemplo: um app de consultas pode se conectar com o sistema do laboratório para receber exames.

Back-end

Parte “invisível” do sistema, responsável pelo funcionamento interno, como armazenamento de dados, regras de negócio e processamento.

Banco de Dados

Local onde todas as informações do sistema são guardadas, como prontuários, agendamentos e dados de pacientes.

Bug

Erro ou falha no sistema que pode causar mau funcionamento. Exemplo: um botão que não responde quando clicado.

Cloud Computing (Computação em Nuvem)

Forma de armazenar e acessar dados pela internet, sem depender de servidores físicos locais. Exemplo: Google Drive, Dropbox.

Criptografia

Técnica para proteger dados, transformando informações em códigos que só podem ser lidos por quem tem permissão.

Deploy (Implantação)

Ato de colocar o sistema pronto para uso em funcionamento, geralmente em um servidor ou na internet.

Design de Interface (UI – User Interface)

Parte visual do sistema, como botões, telas e menus. Deve ser bonita, organizada e fácil de usar.

Desenvolvimento Ágil

Metodologia que divide o desenvolvimento em etapas curtas, com entregas frequentes e adaptação rápida a mudanças. Exemplo: Scrum.

Framework

Conjunto de ferramentas e códigos prontos que ajudam os desenvoledores a criar softwares mais rápido. Exemplo: React, Laravel.

Front-end

Parte “visível” do sistema, com a qual o usuário interage, como telas, formulários e botões.

Git

Sistema para controle de versões do código. Ajuda os desenvolvedores a organizarem mudanças e colaborar em equipe.

Hackers

Pessoas que tentam acessar sistemas de forma ilegal. Nem sempre com má intenção, mas geralmente representa risco à segurança.

Interface

Forma como o usuário interage com o sistema. Pode ser visual (telas, botões) ou por voz (assistentes virtuais).

LGPD (Lei Geral de Proteção de Dados)

Lei brasileira que protege os dados pessoais e sensíveis de cidadãos, exigindo consentimento, segurança e transparência no uso das informações.

Login e Senha

Forma de garantir que apenas pessoas autorizadas tenham acesso ao sistema. Cada usuário tem sua própria conta.

Metodologia Ágil

Conjunto de práticas para desenvolver software com mais flexibilidade, interação com o cliente e foco em entregas rápidas. Exemplo: Scrum, Kanban.

Plataforma

Ambiente onde um software é executado. Pode ser um sistema operacional (como Android, iOS, Windows) ou um serviço online.

Prototipagem

Criação de um modelo inicial do sistema, com visual e funções simuladas, para testes e validações antes do desenvolvimento final.

Requisitos Funcionais

Funções que o sistema precisa cumprir. Exemplo: permitir agendamento de consultas, gerar relatórios médicos.

Requisitos Não Funcionais

Características técnicas do sistema, como desempenho, segurança e facilidade de uso.

Scrum

Um tipo de metodologia ágil que organiza o trabalho em ciclos curtos (chamados de “sprints”), com reuniões frequentes para ajustes e progresso.

Servidor

Computador ou sistema que armazena e entrega os dados e serviços do software aos usuários.

Sistema

Conjunto de programas e componentes que funcionam juntos para resolver um problema ou realizar uma tarefa. Exemplo: sistema de gestão hospitalar.

Teste de Usabilidade

Avaliação para saber se o sistema é fácil de usar para os usuários finais. Envolve observação e coleta de feedback.

UI (User Interface)

Aparência e estrutura das telas com as quais o usuário interage. Envolve layout, botões, cores, fontes, etc.

UX (User Experience)

Experiência geral do usuário ao usar o sistema. Inclui facilidade de uso, satisfação e eficiência.

Wireframe

Desenho simples e básico da interface do sistema, feito para planejar a organização dos elementos antes de criar o visual final.

CONCLUSÃO

Se você chegou até aqui, parabéns! Isso já mostra que você está dando um passo importante rumo à inovação na saúde.

Ao longo deste guia, mostramos que desenvolver um software não é algo exclusivo para quem é da área de tecnologia — **é, sim, um caminho possível e poderoso para quem vive a saúde na prática**. Você, que conhece os desafios dos consultórios, hospitais e pacientes, tem um olhar valioso que pode transformar realidades através da tecnologia.

Você aprendeu como identificar necessidades reais, planejar soluções inteligentes, trabalhar em equipe, escolher ferramentas, testar, melhorar e proteger os dados de quem mais importa: as pessoas. E o mais incrível? **Você não precisa saber programar para começar**. Basta ter visão, curiosidade e vontade de melhorar o que já existe.

A tecnologia só faz sentido quando serve à vida. E quem entende de cuidar da vida são vocês, os profissionais da saúde.

Então, leve esse conhecimento adiante. Compartilhe, questione, crie. Participe do desenvolvimento de soluções mais humanas, mais acessíveis e mais eficazes.

Siga o link para responder nosso questionário de satisfação.
https://forms.gle/4YYQqZJnFuQHTgeF9

A inovação na saúde precisa de você. E o futuro começa agora.

“Não precisamos prever o futuro, precisamos criá-lo.”
— Peter Drucker

REFERÊNCIAS

BOEHM, Barry. *A Spiral Model of Software Development and Enhancement*. ACM SIGSOFT Software Engineering Notes, v. 11, n. 4, p. 14–24, 1986.

BRASIL. *Lei Geral de Proteção de Dados Pessoais – LGPD*. Lei nº 13.709, de 14 de agosto de 2018. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/L13709.htm. Acesso em: 28 maio 2025.

BRASIL. Ministério da Saúde. Diretrizes metodológicas: elaboração de diretrizes clínicas. Brasília: Ministério da Saúde, 2016.

BRASIL. Ministério da Saúde. Diretrizes metodológicas: ferramentas para adaptação de diretrizes clínicas. Brasília: Ministério da Saúde, 2014. BVSMS

CANTÚ, Leandro. *UX Design e Usabilidade: Como aplicar boas práticas de UX no dia a dia*. São Paulo: Casa do Código, 2022.

CYBIS, Walter et al. *Ergonomia e Usabilidade: Conhecimentos, métodos e aplicações*. Rio de Janeiro: Elsevier, 2010.

LARMAN, Craig. *Utilizando UML e Padrões: Uma Introdução à Análise e ao Projeto Orientados a Objetos e ao Processo Unificado*. Porto Alegre: Bookman, 2007.

PRESSMAN, Roger S.; MAXIM, Bruce R. *Engenharia de Software: uma abordagem profissional*. 8. ed. Porto Alegre: AMGH, 2016.

SOMMERVILLE, Ian. *Engenharia de Software*. 10. ed. São Paulo: Pearson Education do Brasil, 2019.

STELLMAN, Andrew; GREENE, Jennifer. *Scrum: a arte de fazer o dobro do trabalho na metade do tempo*. Rio de Janeiro: Sextante, 2015.

Questionário de Satisfação – Guia "Softwares e Aplicativos: Da Ideia à Implementação"

Instruções:
Para cada afirmação, marque o número que melhor representa sua opinião, considerando:
1 – Discordo totalmente | 2 – Discordo parcialmente | 3 – Neutro/Indiferente | 4 – Concordo parcialmente | 5 – Concordo totalmente
* Indica uma pergunta obrigatória

Nome completo (Opcional): _____

Instituição / profissão (área de atuação) *: _____

_____

Seção 1 – Clareza e Organização

Nº	Afirmação	1	2	3	4	5
1	O guia apresenta o conteúdo de forma clara e objetiva.	☐	☐	☐	☐	☐
2	A sequência dos capítulos facilitou meu entendimento sobre o desenvolvimento de software na saúde.	☐	☐	☐	☐	☐
3	A linguagem utilizada é acessível para profissionais da saúde, mesmo sem conhecimento prévio em tecnologia.	☐	☐	☐	☐	☐

Seção 2 – Relevância e Aplicabilidade

Nº	Afirmação	1	2	3	4	5
4	O conteúdo é relevante para minha prática profissional.	☐	☐	☐	☐	☐
5	Os exemplos práticos	☐	☐	☐	☐	☐

	apresentados ajudaram a compreender melhor o assunto.					
6	As atividades visuais propostas contribuíram para fixar o aprendizado.	☐	☐	☐	☐	☐
7	O guia fornece informações que podem ser aplicadas diretamente no meu contexto de trabalho.	☐	☐	☐	☐	☐

Seção 3 – Qualidade Visual e Estrutura

Nº	Afirmação	1	2	3	4	5
8	O design e a diagramação do guia são atraentes e facilitam a leitura.	☐	☐	☐	☐	☐
9	As ilustrações e tabelas ajudaram na compreensão do conteúdo.	☐	☐	☐	☐	☐

Seção 4 – Abrangência dos Tópicos

Nº	Afirmação	1	2	3	4	5
10	O guia abordou todas as etapas essenciais do desenvolvimento de software.	☐	☐	☐	☐	☐
11	O capítulo sobre Segurança, Privacidade e LGPD foi útil para compreender a importância da	☐	☐	☐	☐	☐

	proteção de dados.					
12	A explicação sobre ANVISA esclareceu dúvidas sobre a regulamentação de softwares na saúde.	☐	☐	☐	☐	☐

Seção 5 – Utilidade Geral

Nº	Afirmação	1	2	3	4	5
13	O guia atendeu às minhas expectativas.	☐	☐	☐	☐	☐
14	Este material me deixou mais confiante para participar do desenvolvimento de software na área da saúde.	☐	☐	☐	☐	☐
15	Eu recomendaria este guia para outros profissionais da área.	☐	☐	☐	☐	☐

Comentários Finais

Pontos positivos:

_____
_____
_____

Sugestões de melhoria:

_____
_____
_____
_____