**CENTRO PAULA SOUZA**

**FACULDADE DE TECNOLOGIA DE MOGI MIRIM**

**Luiz Antonio Maria Guidini**

**Marcia Oliveira Mota Guidini**

**Thales Manjaterra**

**Desenvolvimento de Aplicativo para auxílio a idosos: CuidaDoso**

**MOGI MIRIM**

**2024**

**Luiz Antonio Maria Guidini**

**Marcia Oliveira Mota Guidini**

**Thales Manjaterra**

**Desenvolvimento de Aplicativo para auxílio a idosos: CuidaDoso**

Trabalho de Graduação apresentado ao Curso de Tecnologia em Análise e Desenvolvimento de Sistemas da Faculdade de Tecnologia de Mogi Mirim como pré-requisito para a obtenção do Título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

**Orientadora:** Profª. Ma. Rita de Cássia Catini de Macedo

**MOGI MIRIM**

**2024**

FOLHA DE APROVAÇÃO

Será entregue pela coordenação

Dedicamos este trabalho primeiramente à Deus, pois é e sempre será nosso alicerce, e a nossa família por sempre nos apoiar e fortalecer.

**AGRADECIMENTOS**

Gostaríamos de agradecer primeiramente a Deus, que nos deu forças e capacidade para conclusão não só do trabalho de graduação, mas de todo o curso; e, em segundo lugar, aos nossos pais, que sempre nos apoiaram e aconselharam em tudo: escolhas, dificuldades e medos. Tudo que vier a ser, devo a estas duas pessoas maravilhosas e batalhadoras.

À nossa querida filha, Maria Clara, pela paciência, carinho e por ser uma fonte constante de alegria e inspiração.

Aos nossos amigos, desde os tempos de escola até a faculdade, pela amizade, companheirismo e pelas valiosas trocas de conhecimento e experiências que tanto contribuíram para o nosso crescimento

Aos professores da Fatec, pela dedicação, ensinamentos e orientação durante toda nossa formação acadêmica.

À nossa orientadora, Professora Rita Catini, por sua orientação, paciência e por compartilhar seu vasto conhecimento, fundamental para a realização deste trabalho.

A todos, o nosso mais sincero agradecimento vocês são partes dessa conquista.

*“A tarefa não é tanto ver aquilo que ninguém viu, mas pensar o que ninguém ainda pensou sobre aquilo que todo mundo vê.”*

*(Arthur Schopenhauer)*

**LISTA DE FIGURAS OU ILUSTAÇÕES**

[**Figura 1** - Cronograma atividades futuras 18](#_Toc183121933)

[**Figura 2** - Desenvolvimento (a) de "dentro para fora" e (b) de "fora para dentro" 23](#_Toc183121934)

[**Figura 3** - Estrutura de Usabilidade 26](#_Toc183121935)

[**Figura 4** - Camadas da Engenharia de *Software* 28](#_Toc183121936)

[**Figura 5** - Modelo Cascata 32](#_Toc183121937)

[**Figura 6** - O paradigma da prototipação 33](#_Toc183121938)

[**Figura 7** - Modelo espiral 33](#_Toc183121939)

[**Figura 8** - Modelo de Engenharia de *Software* baseada em componentes 35](#_Toc183121940)

[**Figura 9** - Valores do Scrum 41](#_Toc183121941)

[**Figura 10** - Exemplo Diagrama de Caso de Uso 48](#_Toc183121942)

[**Figura 11** - Exemplo de Diagrama de Classe 49](#_Toc183121943)

[**Figura 12** - Exemplo de Diagrama de Sequência 50](#_Toc183121944)

[**Figura 13** - Exemplo de Diagrama de Atividade 51](#_Toc183121945)

[**Figura 14** -Estatísticas de uso de linguagens de programação para sites 54](#_Toc183121946)

[**Figura 15** - Diferentes tipos de login do Firebase *Authentication* 59](#_Toc183121947)

[**Figura 16** - Exemplo de documentos agrupados em uma coleção 60](#_Toc183121948)

**LISTA DE TABELAS**

[**Tabela 1** - Princípios dos métodos ágeis 36](#_Toc182836268)

[**Tabela 2** - Práticas da *Extreme Programming* 38](#_Toc182836269)

[**Tabela 3** - Divisão da Família Crystal 42](#_Toc182836270)

[**Tabela 4** - Princípios de boas práticas da DSDM 43](#_Toc182836271)

**LISTA DE ABREVIATURAS E SIGLAS**

ABNT – Associação Brasileira de Normas Técnicas

BaaS - *Backend as a Service*

CSS – *Cascading Style Sheets*

DSDM - Desenvolvimento de Sistemas Dinâmicos

FDD - Desenvolvimento Dirigido a Funcionalidade

HTML – *Hypertext Markup Language*

IBGE - Instituto Brasileiro de Geografia e Estatística

IHC - Interação Humano Computador

ISO - *International Organization for Standardization*

IU - Interface Usuário

PHP - *Hypertext Preprocessor*

SGBDR - Sistema de Gerenciamento de Banco de Dados Relacional

SDK - *Software Development Kit*

SQL - *Structured Query Language*

TI – Tecnologia da Informação

UML - Linguagem de Modelagem Unificada

*XP – Extreme Programming*

**RESUMO**

O envelhecimento populacional no Brasil apresenta novos desafios, principalmente em relação à saúde e à interação social dos idosos. Este trabalho tem como objetivo desenvolver o aplicativo CuidaDoso, uma plataforma móvel projetada para facilitar o acesso a serviços de saúde, promover a adesão a tratamentos médicos e o acesso a serviços de transporte especializado a esse público, além de reduzir o isolamento social. A pesquisa, de caráter exploratório e descritivo, utilizará levantamento de dados por meio de formulários e análises quantitativas e qualitativas para identificar as necessidades desse público. O projeto será realizado utilizando metodologia Scrum e tecnologias modernas, visando uma interface intuitiva e acessível. Como resultado, o CuidaDoso destaca-se como uma solução eficaz para aprimorar a qualidade de vida dos idosos, proporcionando suporte digital que combina praticidade, inclusão e promoção do bem-estar. Conclui-se que o aplicativo tem potencial para minimizar barreiras tecnológicas, contribuindo para a saúde física e mental da população idosa.

**Palavras-chave:** Aplicativos móveis; Cuidados com idosos; Envelhecimento populacional; Tecnologia e saúde.

***ABSTRACT***

*The aging population in Brazil presents new challenges, particularly regarding the health and social interaction of the elderly. This study aims to develop the CuidaDoso application, a mobile platform designed to facilitate access to healthcare services, promote adherence to medical treatments, and provide access to specialized transportation services for this demographic, while also reducing social isolation. The research, which is exploratory and descriptive in nature, will use data collection through forms and both quantitative and qualitative analyses to identify the needs of this audience. The project will be developed using the Scrum methodology and modern technologies, ensuring an intuitive and accessible interface. As a result, CuidaDoso stands out as an effective solution to improve the quality of life for the elderly, offering digital support that combines practicality, inclusion, and well-being promotion. It is concluded that the application has the potential to minimize technological barriers, contributing to the physical and mental health of the elderly population.*

*Keywords: Mobile applications; Elderly care; Population aging; Technology and health.*

Sumário

[1. INTRODUÇÃO 15](#_Toc183181666)

[1.1. Estrutura do Trabalho 16](#_Toc183181667)

[2. METODOLOGIA DA PESQUISA CIENTÍFICA 17](#_Toc183181668)

[2.1. Cronograma 18](#_Toc183181669)

[3. FUNDAMENTAÇÃO TEÓRICA 19](#_Toc183181670)

[3.1. Tecnologia e Informação 19](#_Toc183181671)

[3.2. Aplicação *Web* 20](#_Toc183181672)

[3.3. Aplicação Móvel 21](#_Toc183181673)

[3.4. Interação Humano Computador 22](#_Toc183181674)

[4. ESTUDO DE CASO 61](#_Toc183181675)

[4.1. Levantamento de Requisitos 62](#_Toc183181676)

[4.1.1. Questionário 62](#_Toc183181677)

[**Informações Gerais** 62](#_Toc183181678)

[1 - Nome do Entrevistado \* 62](#_Toc183181679)

[62](#_Toc183181680)

[**REFERÊNCIAS BIBLIOGRÁFICAS** 66](#_Toc183181681)

[**APÊNDICE (criados pelo autor)** 71](#_Toc183181682)

[**ANEXO (não criados pelo autor)** 71](#_Toc183181683)

# INTRODUÇÃO

O envelhecimento da população é uma tendência mundial que traz junto uma série de desafios e demandas crescentes, como a necessidade de mantê-la ativa e participante na sociedade.

De acordo com o Instituto Brasileiro de Geografia e Estatística (IBGE, 2023), a população com 60 anos ou mais representa mais de 33 milhões de pessoas. Em 10 anos, essa parcela da população teve um aumento de 11% para 15%. Esse crescimento é decorrente dos avanços da medicina, aumentando a expectativa de vida da população.

Com esse aumento, surge a necessidade de encontrar soluções inovadoras que possam atender às necessidades específicas dos idosos, garantindo-lhes uma vida mais saudável e independente. Nesse contexto, a tecnologia desempenha um papel crucial ao oferecer ferramentas e recursos que auxiliam os idosos em suas atividades diárias e na gestão de sua saúde.

Uma ferramenta que auxilia a população em geral, são os aplicativos móveis usados no cotidiano, como: os aplicativos de comunicação, os de entretenimento, redes sociais, saúde e bem-estar, produtividade, transporte, financeiro, entre outros.

Os aplicativos móveis passaram por uma evolução significativa desde o surgimento dos *smartphones*. Inicialmente, eram simples e focados em funcionalidades básicas. Com o tempo, tornaram-se mais diversificados e sofisticados, abrangendo uma ampla gama de serviços e usos. Atualmente, os aplicativos são uma parte essencial da vida cotidiana, oferecendo conveniência, entretenimento e produtividade para os usuários. Nesse sentido, os aplicativos móveis oferecem uma grande oportunidade para atender às necessidades específicas dos idosos, fornecendo soluções personalizadas e adaptadas às suas capacidades e preferências.

Como podemos maximizar o benefício dos dispositivos móveis para idosos, levando em conta suas necessidades exclusivas e os desafios típicos associados à idade avançada, a fim de promover sua saúde e bem-estar?

Diante do exposto, o objetivo deste trabalho é desenvolver um aplicativo para dispositivos móveis, denominado CuidaDoso, que ofereça uma plataforma móvel projetada para facilitar o acesso a serviços de saúde, promover a adesão a tratamentos médicos, estimular a prática de atividades físicas e combater o isolamento social entre os idosos.

Partindo da hipótese que o agente causador dos problemas seja a dificuldade em lidar com as tecnologias atuais e a falta de familiaridade com dispositivos móveis sejam os principais obstáculos enfrentados pelos idosos causando uma barreira no acesso a serviços de saúde, atividades físicas e interação social. Acredita-se que o aplicativo, com uma interface intuitiva e funcionalidades específicas, ajudará a melhorar a qualidade de vida dos idosos, proporcionando-lhes um meio eficaz de cuidar de sua saúde e bem-estar, em qualquer lugar, inclusive no conforto de seus lares.

A principal motivação para abordar o tema deste trabalho foi a necessidade de desenvolver um sistema que promova o cuidado abrangente dos idosos, contemplando sua saúde física, mental e emocional. O objetivo é otimizar recursos para um melhor planejamento e facilitar a interação, melhorando a qualidade de vida dos idosos por meio do acesso simplificado a serviços médicos, atividades físicas e transporte.

## Estrutura do Trabalho

O presente trabalho está estruturado em 05 capítulos:

No primeiro capítulo vê-se a introdução, apresenta o tema do trabalho, os objetivos gerais e específicos, a justificativa para a escolha do tema e a relevância do estudo no contexto do desenvolvimento.

O capítulo 2 diz respeito a metodologia da pesquisa cientifica, no qual abordamos os métodos para coleta de dados para esse projeto.

No terceiro capítulo, encontra-se a fundamentação teórica, explora os principais conceitos relacionados ao desenvolvimento web e *mobile*, incluindo a história e evolução do HTML, JavaScript e outras tecnologias envolvidas. Também aborda as características dessas linguagens e suas contribuições para o dinamismo das páginas web. A metodologia Scrum, escolhida para aplicação desse projeto.

O capítulo 4, aborda o estudo de caso, o formulário para coleta de dados e levantamento dos requisitos, os resultados obtidos.

No último capítulo apresenta a conclusão do trabalho, destacando as contribuições do estudo, limitações e sugestões futuras.

# METODOLOGIA DA PESQUISA CIENTÍFICA

Segundo Rampazzo (2005), a metodologia pode ser entendida como um conjunto de ferramentas utilizadas para estruturar dados, levantar requisitos, formular problemas, construir hipóteses, determinar a linha de pesquisa e selecionar os métodos e técnicas mais adequados. Esse processo serve de base para a aplicação da pesquisa sobre o tema em questão.

As pesquisas possuem dois tipos de abordagens, quantitativas e qualitativas. A abordagem quantitativa é aquela em que os resultados são obtidos por meio da quantificação dos dados. Geralmente, aplicam-se questionários com perguntas objetivas e apresenta-se os resultados através de gráficos. Já a abordagem qualitativa é conceituada como aquela que necessita de estudo interpretativa por parte do pesquisador para alcançar resultados. Normalmente, os questionários são compostos questões subjetivas e os resultados são apresentados por meio de tabelas ou resumos (CIRIBELLI, 2003).

Conforme mencionado acima, esta pesquisa caracteriza-se por ser uma abordagem quantitativa para a análise de dados, visto que o questionário possui predominantemente perguntas objetivas. Desta maneira, a interpretação dos resultados foi feita de forma dedutiva por meio de métodos estatísticos. Além disso, assume caráter qualitativo também, pois existem algumas perguntas subjetivas no questionário que exigirão a análise indutiva, ou seja, interpretativa dos autores.

As pesquisas podem ser classificadas quanto aos tipos: explicativa, descritiva e exploratória. A explicativa é aquela em que o pesquisador explica o porquê de algo e suas causas, por meio da análise de um fenômeno. A descritiva, o pesquisador regista e descreve as características dos fenômenos observados. À exploratória é aquela em que o pesquisador faz um levantamento de informações acerca do assunto por meio de outras bibliografias (PRODANOV; FREITAS, 2013).

O desenvolvimento da pesquisa será conduzido de forma criteriosa, a fim de atingir os objetivos propostos. Classifica-se como uma pesquisa exploratória, visto que a base dos conhecimentos foi em fontes bibliográficas, como livros, artigos, monografias, teses, entre outros; e descritiva, pois foi feita uma pesquisa por meio de formulários afim de observar, registrar e analisar os resultados obtidos.

## Cronograma

O cronograma consiste em uma ferramenta de planejamento e controle a serem executadas dentro de um período estimado para a conclusão de um projeto (LEÃO, 2018). É ainda um instrumento de detalhamento dos processos e das tarefas do projeto com prazos de execução e de conclusão.

Os objetivos do cronograma são, identificação e organização das sequências de tarefas, estabelecimento de prazos, facilidade na estimativa de prazo para a conclusão do projeto (LEÃO, 2018). Existem vários modelos de cronogramas utilizados, como por exemplo, Diagrama de *Gantt*, cujo é mais utilizado com maior frequência nos projetos e será a ferramenta utilizada para estimular as atividades e as metas de cada uma das tarefas que compõe o projeto.

É possível observar na Figura 1, a seguir, o cronograma das atividades futuras

**Figura 1** - Cronograma atividades futuras

Uma imagem contendo Gráfico

Descrição gerada automaticamente

**Fonte:** Próprio Autor (2024)

# FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, serão exploradas as principais abordagens e ferramentas que compõem o desenvolvimento de sistemas, com ênfase nas linguagens de modelagem, técnicas de programação e metodologias de desenvolvimento. A utilização da Linguagem de Modelagem Unificada (UML) como padrão para documentação e visualização de sistemas será abordada em detalhe, facilitando a compreensão de sua estrutura e funcionamento. Além disso, discutiremos a relevância de tecnologias como HTML, *JavaScript*, PHP, Dart e *Flutter*, que facilitam o processo de criação e implementação de aplicações. A proposta deste capítulo é fornecer uma visão abrangente das ferramentas e métodos que podem ser utilizados para otimizar o desenvolvimento de software, bem como a importância de um banco de dados eficaz, exemplificado pelo uso do *MySQL*.

## Tecnologia e Informação

De acordo com Lemos II (2011), o termo Tecnologia da Informação (TI) tem sido definido de várias formas ao longo do tempo. Inicialmente, a computação era vista como um meio de automatizar tarefas em grandes empresas e governos. Com o avanço tecnológico, as enormes máquinas foram substituídas por equipamentos menores e mais poderosos. A evolução das telecomunicações possibilitou a comunicação entre computadores, permitindo que eles não apenas automatizassem tarefas, mas também lidassem com a informação.

A informação é um conjunto de dados com um significado, que reduz a incerteza ou que aumenta o conhecimento a respeito de algo (CHIAVENATTO, 2000). Já para Beal (2004, p.12) “informação é produzida a partir de dados dotados de relevância e propósito”. Assim, entende-se que um dado, informação e conhecimento estão interligados, porém “um conjunto de dados não produz necessariamente uma informação, nem um conjunto de informações representa necessariamente um conhecimento” (BEAL, 2004, p.26), a relevância de cada um é que diz seu valor em um determinado contexto.

Segundo Veloso (2011, p. 2) “[...] O conceito de tecnologia pode ser aplicado a tudo aquilo que, não existindo na natureza, o ser humano inventa para expandir seus poderes, superar suas limitações físicas, tornar seu trabalho mais fácil e a sua vida mais agradável.”, afirma ainda que a tecnologia não se resume a instrumentos tangíveis, incluindo também elementos intangíveis como procedimentos, métodos e técnicas.

De modo geral, a tecnologia da informação tem sido um motor essencial para a inovação e transformação em diversos setores. Com o avanço da internet e das capacidades computacionais, surgiram as aplicações *web* e os aplicativos móveis, que revolucionaram a forma como interagimos, trabalhamos e consumimos informações. Essas plataformas digitais, acessíveis a partir de navegadores ou dispositivos móveis, permitem a execução de tarefas complexas e facilitam a comunicação em tempo real, tornando-se indispensáveis na era moderna. A integração entre a tecnologia e essas aplicações práticas ressalta como os avanços tecnológicos moldam e ampliam continuamente as possibilidades do mundo digital, impactando profundamente nossa vida cotidiana.

## Aplicação *Web*

A *Web* pode ser entendida como um sistema que possibilita a transferência e conexão entre hipertextos através da internet. Esses hipertextos são arquivos digitais que podem incluir textos, imagens, vídeos e sons, conectados por links que direcionam o usuário para outras páginas, também conhecidas como páginas Web (MOZILLA, 2024).

Quando foi criada em 1990, pelo físico e cientista da computação britânico Tim Berners-Lee, a Web era um ambiente estático, com a função básica de armazenar e disponibilizar informações. Naquela época, os *softwares web* não eram amplamente confiáveis e eram acessíveis apenas em redes locais dentro de organizações (CLEMENTE, 2008).

Com o avanço tecnológico e a ampliação da internet, a *Web* evoluiu e se tornou mais rápida e versátil. Surgiram então os *softwares web*, que permitiram que sistemas pudessem ser acessados diretamente por qualquer dispositivo com navegador, sem a necessidade de instalação local no cliente. Esse avanço possibilitou que muitos negócios migrassem seus sistemas para o ambiente online, transformando a internet em uma plataforma para gestão organizacional (SOMMERVILLE, 2011).

Santos (2018), define um sistema *web* como um site dinâmico que oferece uma experiência personalizada ao usuário, em contraste com sites estáticos, que apresentam conteúdo sem interação direta com o usuário. Em sistemas web, é possível realizar login, gerenciar dados e explorar funcionalidades semelhantes às de um aplicativo.

Os sistemas de informação baseados em tecnologia *web* trazem vantagens como facilidade de compartilhamento de informações, controle centralizado de atualizações e maior poder de processamento, concentrado no servidor. Diferentemente dos sistemas *desktop*, que funcionam *offline*, exigem instalação local e necessitam que atualizações sejam realizadas em cada dispositivo cliente, os sistemas *web* são acessíveis de qualquer lugar e sempre atualizados (SANTOS, 2018).

O desenvolvimento contínuo da tecnologia *web* tem sido impulsionado pela necessidade de simplificar a atualização e manutenção dos sistemas, centralizando o código-fonte em um único local para ser acessado por múltiplos usuários ao mesmo tempo. O código-fonte, composto por linhas de programação, é o núcleo que garante o funcionamento de qualquer programa ou sistema (GONÇALVES, 2007).

O ambiente web tem crescido significativamente, com diversos serviços da empresa Google, como Google Docs, Google Drive e Slides, permitindo a edição de documentos, planilhas e apresentações diretamente no navegador. Esses serviços destacam a flexibilidade, comodidade e escalabilidade oferecidas pelo ambiente web.

## Aplicação Móvel

O desenvolvimento de aplicativos móveis é o processo de criar *softwares* para dispositivos como *smartphones* e *tablets*, geralmente focado em sistemas operacionais como *Android* e *iOS*.

O crescimento da criação de aplicativos móveis reflete a necessidade de se conectarem com seus usuários de forma rápida e prática. Os aplicativos são, hoje, o principal meio de conexão com a internet, tanto para consumidores quanto para organizações. (IBM, 2024). Além disso, práticas de usabilidade e design responsivo são fundamentais para garantir uma experiência de usuário otimizada, fator decisivo no sucesso e na aceitação dos aplicativos no mercado (Nielsen e Loranger, 2013).

## Interação Humano Computador

Vivemos em um mundo de produtos de alta tecnologia e praticamente todos requerem interação humana. Para que um produto de software seja bem-sucedido, deve apresentar boa usabilidade – medida qualitativa da facilidade e eficiência com a qual o ser humano consegue empregar as funções e os recursos oferecidos. (PRESSMAN, 2011).

Entende-se por sistema de interação homem-computador a relação entre usuários e produtos que requerem a ação dos usuários para cumprir suas atividades. (ROGERS et al., 2013).

Segundo Oliveira (2015, p.11)

O conceito vai além da estética da tela (disposição de menus, cores etc.). Ele se estende a questões como corretude, tempo de resposta, grau de dificuldade de uso, rapidez no desempenho de tarefas-chave, nível de erros cometidos por usuários durante o uso, facilidade de aprendizado, fadiga produzida pelo uso prolongado, acomodação de usuários com necessidades especiais, dentre outros.

A Interação Humano Computador (IHC), no primeiro momento da construção de um sistema interativo, busca conhecer os interesses dos possíveis usuários, seus objetivos e domínios. Dessa forma, utiliza-se uma abordagem de 'fora para dentro' (Figura 1, ilustração B) para direcionar o desenvolvimento do sistema (BARBOSA; SILVA, 2010).

Já a computação tem como prioridade a qualidade da construção, isto é, conceber primeiro representações de dados, arquitetura do sistema e tudo aquilo que permite o funcionamento do sistema interativo. Pouca atenção é dedicada ao que é externo ao sistema. Trata-se de uma abordagem de “dentro para fora” (Figura 1, ilustração A). Neste caminho, os usuários têm que se adaptar ao sistema, porém o mundo externo ao sistema nem sempre se adapta de forma fácil e rápida. (BARBOSA; SILVA, 2010). Melhor dizendo, as abordagens IHC focam no uso e no contexto de uso, envolvendo os usuários no processo, e com o *design* de sistemas interativos, que se preocupa com a criação de experiências interativas, acessíveis e adequadas ao uso (BENYON, 2011). Na Figura 2, observamos melhor essa abordagem.

**Figura 2** - Desenvolvimento (a) de "dentro para fora" e (b) de "fora para dentro"

Diagrama

Descrição gerada automaticamente

**Fonte:** Barbosa e Silva (2010)

* + 1. **Interface**

“Quando o conceito de interface surgiu, ela era geralmente entendida como o hardware e o software com o qual homem e computador podiam se comunicar. A evolução do conceito levou à inclusão dos aspectos cognitivos e emocionais do usuário durante a comunicação.” (ROCHA; BARANAUSKAS, 2003, p.7)

Segundo Pressman (2011) a interface é discutivelmente o elemento mais importante. [...]é a janela para o *software,* em alguns casos ela molda a percepção do usuário quanto a qualidade. Se a “janela” estiver embaçada, quebradiça ou ondulada, ou seja, difícil para enxergar, o usuário poderá rejeitar o sistema que de outra forma seria considerado poderoso.

De acordo com Barbosa e Silva (2010) a interface é sistema interativo que compreende toda a porção do sistema com a qual o usuário mantém contato físico ou conceitual durante a interação, é o único meio de contato entre eles. Por isso, a grande maioria dos usuários acredita que o sistema é a interface com a qual entram em contato.

O contato físico na interface ocorre através do *hardware* e do *software* utilizados durante a interação. Dispositivos de entrada, que permitem ao usuário agir sobre a interface do sistema e participar ativamente da interação, já os dispositivos de saída, permitem ao usuário perceber as reações do sistema e participar passivamente da interação. O *software* determina os efeitos no comportamento do sistema decorrentes das ações do usuário sobre os dispositivos de entrada, bem como os efeitos nos dispositivos de saída decorrentes de um processamento realizado pelo sistema. (BARBOSA; SILVA, 2010).

Já o contato conceitual com a interface envolve a interpretação do usuário daquilo que ele percebe através do contato físico com os dispositivos de entrada e de saída durante o uso do sistema. Essa interpretação permite ao usuário compreender as respostas do sistema e planejar os próximos caminhos de interação (BARBOSA; SILVA, 2010). Nesse contexto a interface ela necessita de uma interação com o usuário que seja de forma clara e simples, para que não gere dificuldade no seu uso, ou seja, o fator humano é indispensável.

Sommerville (2007) destaca que alguns fatores humanos são a base dos princípios de projeto de interface e tornam aplicáveis a todos os projetos de interface. Esses fatores são nomeados como princípios gerais e normalmente são instanciados como diretrizes de projetos:

* Familiaridade de usuário: A interface deve usar termos e conceitos obtidos da experiência de pessoas que farão mais uso do sistema.
* Consistência: A interface deve ser consistente de maneira que, sempre que possível, as operações comparáveis sejam ativadas da mesma maneira.
* Surpresa mínima: Os usuários nunca devem ser surpreendidos pelo comportamento de um sistema.
* Facilidade de recuperação: A interface deve incluir mecanismos que permitam aos usuários se recuperam de erros.
* Guia do usuário: A interface deve fornecer feedback significativo quando ocorrem erros e fornecer recursos sensíveis ao contexto para ajudar o usuário.
* Diversidade do usuário: A interface deve fornecer recursos de interação adequados para tipos diferentes de usuários de sistema. (SOMERVILLE, 2007)

Pressman (2011) cita três regras de ouro para projetar uma interface, a base para um conjunto de princípios que orienta esse aspecto importante do projeto de *software*:

1ª regra: Deixar o usuário no comando:

* Defina modos de interação para não forçar o usuário a realizar ações desnecessárias ou indesejadas;
* Proporcione interação flexível;
* Possibilite que a interação de usuário possa ser interrompida de desfeita;
* Simplifique a interação à medida que os níveis de competência avançam e permita que a interação seja personalizada;
* Oculte os detalhes técnicos de funcionamento interno do usuário casual;
* Projete para interação direta com objetos que aparecem em tela;

2ª regra: Reduzir a carga de memória do usuário:

* Reduza a demanda de memória recente;
* Estabeleça *defaults* significativos;
* Defina atalhos intuitivos;
* O *layout* visual da interface deve se basear na metáfora do mundo real;
* Revele as informações de maneira progressiva.

3ª regra: Tornar a interface consistente.

* Permita ao usuário inserir a tarefa atual em um contexto significativo;
* Mantenha a consistência ao longo de uma família de aplicações;
* Se modelos interativos anteriores tiverem criado expectativa nos usuários, não faça alterações a menos que haja uma forte razão para isso.

Ambos os autores convergem na importância de colocar o usuário no centro do processo de design, garantindo que a interface seja intuitiva, adaptável e coerente, a fim de facilitar a interação e maximizar a usabilidade do sistema.

* + 1. **Usabilidade**

De acordo com a ISO 9241-11 (2002), a usabilidade é a capacidade que um produto tem de oferecer ao seu usuário, em um contexto específico de uso, a realização das tarefas e objetivos específicos com eficácia, eficiência e satisfação. A eficácia é a “acurácia e completude com as quais os usuários alcançam objetivos específicos”. A eficiência se refere aos “recursos gastos em relação à acurácia e abrangência com as quais usuários atingem os objetivos”. Já a satisfação se refere a “ausência do desconforto e presença de atitudes positivas com o uso de um produto”.

Segundo Brasil (2010) A usabilidade é o estudo ou a aplicação de técnicas para prover facilidade de uso de um sistema, visa garantir que qualquer usuário consiga usar este sistema e que, também, funcione da forma esperada pelo usuário, cita ainda, que a usabilidade tem por objetivo a facilidade de uso e de aprendizado; memorização e produtividade na execução de tarefas; prevenção de erros; e por último, satisfazer o usuário.

Pressman (2011) afirma que usabilidade não é derivada da estética, mecanismos de interação de última geração ou de inteligência incorporada a interfaces, mas, quando a interface atende as necessidades do usuário.

Para especificar ou medir a usabilidade, conforme a ISO 9241-11 (2002) “é necessário identificar os objetivos e decompor eficácia, eficiência e satisfação e os componentes do contexto de uso em subcomponentes com atributos mensuráveis e verificáveis”. Na Figura 3 observa-se como esses componentes se relacionam entre si.

**Figura 3** - Estrutura de Usabilidade

**Diagrama

Descrição gerada automaticamente**

**Fonte:** NBR ISO, 9241-11 (2002)

A usabilidade é um aspecto central na Engenharia de *Software,* pois um sistema eficaz não se resume a funcionalidades e desempenho, mas também deve ser intuitivo e agradável para o usuário. A Engenharia de *Software* estabelece princípios e práticas que visam criar sistemas de fácil uso e interação, projetando a experiência do usuário como um componente essencial no processo de desenvolvimento.

* 1. **Engenharia de *Software***

A engenharia de *software* é um campo que se concentra na especificação, desenvolvimento, manutenção e criação de *software*, aplicando técnicas e práticas de gerenciamento de projetos, com o objetivo de organização, produtividade e qualidade. Essas práticas incluem linguagens de programação, bancos de dados, ferramentas, plataformas, bibliotecas, padrões de projeto de *software*, processos de *software* e qualidade de *software*. Além disso, a engenharia de *software* deve fornecer mecanismos para planejar e gerenciar o processo de desenvolvimento de sistemas computacionais de alta qualidade que atendam às necessidades dos solicitantes de *software.* (PRESSMAN,2011)

Segundo Sommerville (2007, p.5) “A Engenharia de *Software* é uma disciplina de engenharia relacionada com todos os aspectos da produção de *software*, desde os estágios iniciais de especificação do sistema até a sua manutenção, depois que ele entrou em operação.” Já para Pressman (2011) “a engenharia de *software* engloba processos, métodos e ferramentas que possibilitam a construção de sistemas complexos baseados em computador dentro do prazo e com qualidade.” [...]é uma tecnologia em camadas.

Na Figura 4 observamos como são apresentadas as camadas da Engenharia de *Software*.

**Figura 4** - Camadas da Engenharia de *Software*

![Uma imagem contendo Gráfico de funil

Descrição gerada automaticamente]()

**Fonte:** Presmann (2011, p.39)

* **Ferramentas:** As ferramentas de apoio automatizado facilitam o processo de desenvolvimento de *software*, integrando informações entre si. Isso constitui a engenharia de *software* com o auxílio do computador.
* **Métodos:** Fornecem as informações técnicas para desenvolver *software*. Envolvem uma gama de tarefas, que incluem: comunicação, análise de requisitos, modelagem de projeto, construção de programa, teste e suporte, baseadas em princípios básicos.
* **Processos:** É a que mantem as camadas da tecnologia coesas e possibilita o desenvolvimento do *software* de forma racional e dentro do prazo, constitui a base para o controle de gerenciamento de projetos e determina o contexto no qual são aplicados métodos técnicos.
* **Foco na qualidade:** a gestão da qualidade promove uma cultura de aperfeiçoamento contínuo dos processos, levando abordagens cada vez mais efetivas. É a pedra fundamental que sustenta a engenharia de *software*. (PRESMANN, 2011)

Pressman e Maxim (2021) cita que um conjunto de cinco atividades estruturais são aplicáveis a todos os projetos, estabelece uma metodologia para a prática da engenharia de *software:*

* **Comunicação:** Antes que qualquer trabalho técnico possa começar, é de importância fundamental se comunicar e colaborar com o cliente (e outros envolvidos). A intenção é entender os objetivos para o projeto e reunir requisitos que ajudem a definir as funções de um *software*.
* **Planejamento:** Define o trabalho de engenharia de *software*, descrevendo as tarefas técnicas a serem conduzidas, os riscos prováveis, os recursos necessários, os produtos resultantes a ser produzidos em um cronograma de trabalho.
* **Modelagem:** Cria-se um “esboço” para que se possa ter uma ideia do todo – qual será o seu aspecto em termos de arquitetura, como as partes constituintes se encaixarão e várias outras características. Se necessário, refina-se com mais detalhes, para uma melhor compreensão do problema e como resolvê-lo.
* **Construção:** Combinação de geração de código (manual ou automatizada) e testes necessários para revelar erros na codificação.
* **Entrega:** O *software* (como entidade completa ou parcialmente efetivado) é entregue ao cliente, que avalia e fornece o *feedback*, baseado na avaliação.

E para complementar essas atividades estruturais aplicáveis, Presmann (2011), afirma, que independente do seu nível de enfoque, princípios ajudam a estabelecer um modo de pensar para a prática segura da engenharia de *software* - esta é a razão por que são importantes. Nesse contexto, propôs sete princípios:

* **Primeiro Princípio: a razão de existir**

Todo sistema de *software* deve agregar valor para seus usuários e todas as decisões devem ser orientadas por esse princípio.

* **Segundo Princípio: *KISS* (*Keep It Simple, Stupid!* ou seja: não complique)**

Busque a simplicidade no projeto de *software*, mantendo-o fácil de compreender e manter, sem torná-lo simplista.

* **Terceiro Princípio: mantenha a visão**

Uma visão clara é essencial para o sucesso do projeto. Comprometer a visão arquitetural pode enfraquecer o sistema.

* **Quarto Princípio: o que um produz outros consomem**

Sempre projete considerando que outros precisarão entender, manter ou ampliar o sistema. Facilite o trabalho dessas pessoas para agregar valor ao sistema.

* **Quinto Princípio: esteja aberto para o futuro**

Projetar sistemas que possam se adaptar a mudanças e evoluções futuras é essencial para sua longevidade e sucesso.

* **Sexto Princípio: planeje com antecedência, visando a reutilização**

Planeje com antecedência visando a reutilização de componentes e projetos, o que economiza tempo e esforço no desenvolvimento.

* **Sétimo Princípio: pense!**

Analise cuidadosamente antes de agir, pois o pensamento claro e reflexivo leva a melhores resultados. Pensar é crucial para aplicar os outros princípios com eficácia. (PRESMANN,2011).

Na Engenharia de *Software*, essa ideia reflete-se nas metodologias adotadas para o desenvolvimento, as tradicionais garantem um processo bem estruturado e controlado, enquanto as metodologias ágeis focam em flexibilidade e adaptação rápida, permitindo entregas mais dinâmicas e colaborativas. Cada abordagem contribui para a criação de um software eficiente, embora de maneiras diferentes.

* + 1. **Metodologias Tradicionais**

Segundo Pressman (2011, p.59) “Originalmente, modelos de processos descritivos são, algumas vezes, conhecidos como modelos de processos “tradicionais”.

Sommerville (2007) diz que é uma representação abstrata de um processo de *software*, fornecendo somente informações parciais sobre esse processo.

De acordo com (PRESMANN, MAXIM, 2021, p.88)

“Definem como um conjunto de elementos de processo e um fluxo de trabalho de processo previsível, concentra-se em estruturar e ordenar o desenvolvimento de *software*, ocorrendo sequencialmente as atividades e tarefas, com diretrizes e progressos definidos. [...] prescrevem um conjunto de elementos de processos – atividades metodológicas, ações de engenharia de *software*, tarefas, artefatos, garantia de qualidade e mecanismos de controle de mudanças para cada projeto[...]”

Todos os modelos de processo de *software* podem acomodar as atividades estruturais metodológicas genéricas, mencionado no tópico de Engenharia de *Software*, porém, cada um deles dando ênfase diferente.

Dentro das Metodologias Tradicionais, temos 3 modelos de processos destacados por Sommerville (2007) que são:

1. Modelo em cascata;

2. Desenvolvimento incremental;

3. Engenharia de *software* baseada em componentes.

* + - 1. **Modelo Cascata**

“O primeiro modelo de processo de desenvolvimento de *software* publicado” (SOMMERVILLE, 2007, p. 44). O modelo cascata, algumas vezes chamado de ciclo de vida clássica, sugere uma abordagem sequencial e sistemática para o desenvolvimento de software, começando com a especificação dos requisitos do cliente, avançando pelas fases do planejamento, modelagem, construção e entrega. (PRESMAN, MAXIM, 2021, p.89)

De acordo com Sommerville (2011) o modelo em cascata possui estágios principais que correspondem diretamente às atividades essenciais do desenvolvimento.

* Análise e definição de requisitos. Os serviços, restrições e objetivos do sistema são determinados em colaboração com os usuários. Depois, são detalhados minuciosamente e servem como uma especificação para o sistema
* Projeto de sistema e *software*. O processo distribui os requisitos para hardware e software, definindo uma arquitetura geral, inclui a identificação e descrição das principais abstrações do sistema de *software* e seus relacionamentos
* Implementação e teste unitário. Durante essa fase, é elaborado como um conjunto de programas ou unidades de programa do *software.* O teste unitário verifica se cada unidade cumpre suas especificações
* Integração e teste de sistema. As unidades individuais do programa são integradas e testadas como um sistema completo garantindo que os requisitos do software foram atendidos, após a essa fase, o *software* é entregue ao cliente.
* Operação e manutenção. Esta é a fase mais longa do ciclo de vida, embora nem sempre. A instalação é feita e entra em operação. A manutenção do *software* inclui a correção de erros que não foram identificados nas etapas iniciais, além de melhorias na implementação das unidades do sistema e expansão de seus serviços para atender a novos requisitos descobertos.

Na Figura 5 observa-se o processo dessa metodologia.

**Figura 5** - Modelo Cascata

Diagrama

Descrição gerada automaticamente

**Fonte:** Presmann (2011, p.60)

* + - 1. **Desenvolvimento Evolucionário**

O *software* evolui ao longo do tempo, como todos os sistemas complexos. Conforme o desenvolvimento do projeto avança, os requisitos do negócio e do produto frequentemente mudam, tornando inadequado seguir um planejamento em linha reta de um produto final. (PRESSMAN, 2011).

Modelos evolucionários são iterativos, com características que possibilitam desenvolver versões cada vez mais completas. (PRESSMAN,2011).

Existem dois tipos de desenvolvimento evolucionário:

1. Prototipação: é uma abordagem útil quando os objetivos do software não estão claros ou quando há incerteza sobre como implementar certas funcionalidades. Essa técnica pode ser usada como uma maneira de identificar requisitos de software. A Figura 6 ilustra essa abordagem.

**Figura 6** - O paradigma da prototipação

Forma

Descrição gerada automaticamente

**Fonte:** Pressman (2016, p.45)

1. Modelo Espiral: é um modelo de processo de *software* evolucionário que acopla a natureza iterativa da prototipação com os aspectos sistemáticos e controlados do modelo cascata. Cada ciclo da espiral representa uma fase do processo de desenvolvimento de software. A parte interna está relacionada à avaliação da viabilidade do sistema, seguida pela definição de requisitos, projeto do sistema e etapas subsequentes (SOMMERVILLE,2007). Observa-se na Figura 7 esse modelo.

**Figura 7** - Modelo espiral

Gráfico, Diagrama

Descrição gerada automaticamente

**Fonte:** Sommerville (2011, p.33)

* + - 1. **Engenharia de *software* baseada em componentes**

Sommerville (2007) cita que a maioria dos projetos de *software* utilizam algum reuso de *software,* ocorrendo de maneira informal, pois as pessoas que trabalham no projeto, conhecem os códigos similares, os modificam e incorporam ao sistema. Essa abordagem de reuso depende de uma grande base de componentes de *software* reusáveis e algum *framework* de integração desses componentes. A engenharia de *software* baseada em componentes tem a vantagem de reduzir os custos e riscos e entrega mais rápida do *software.* Os estágios em um processo orientado a reuso são:

1. Análise de componentes: Diante da especificação de requisitos, realiza-se uma busca por componentes para implementar essa especificação. Geralmente, não há uma correspondência exata, e os componentes disponíveis fornecem apenas parte da funcionalidade necessária.

2. Modificação de requisitos: Essa etapa, os requisitos são analisados com base nas informações dos componentes encontrados, logo, serão ajustados para se adequarem aos componentes disponíveis. Caso não seja possível fazer as modificações necessárias, a atividade de análise dos componentes pode ser reintroduzida na busca por soluções alternativas.

3. Projeto do sistema com reuso: Durante esta fase, o *framework* do sistema é projetado ou é feito uso de algo existente para reutilização, tendo em mente os componentes que serão reutilizados e organizados no *framework* para facilitar essa reutilização. Se esses componentes não estiverem disponíveis, pode ser necessário desenvolver novos *softwares*.

4. Desenvolvimento e integração: *Softwares* que não podem ser adquiridos externamente são desenvolvidos internamente, a integralização dos componentes e sistemas são feitos para criar o novo sistema, podendo ser incorporada ao processo de desenvolvimento, em vez de ser uma atividade a parte.

Na Figura 8 podemos observar esse modelo.

**Figura 8** - Modelo de Engenharia de *Software* baseada em componentes

Diagrama, Texto

Descrição gerada automaticamente

**Fonte:** Sommerville (2007, p.46)

* + 1. **Metodologias Ágeis**

“A insatisfação com as abordagens de desenvolvimento pesada levou um número de desenvolvedores de *software* na década de 1990 a propor novos métodos ágeis”. (SOMMERVILLE, 2007, p.262)

Pressman (2011) diz que os métodos ágeis se desenvolveram em um esforço para sanar fraquezas reais e perceptíveis do tradicional. O desenvolvimento ágil oferece benefícios, mas não é indicado para todos os projetos, produtos, pessoas e situações.

De acordo com Highsmith (2001) um encontro entre 17 líderes que trabalhavam no contrafluxo dos padrões da indústria de *software,* foi discutido formas de trabalho, objetivando chegar a uma nova metodologia de produção de software, que pudesse ser usada por todos eles e em outras empresas, substituindo os modelos tradicionais de desenvolvimento. O grupo chegou ao consenso de que alguns princípios eram determinantes para a obtenção de bons resultados. O resultado deste encontro foi a criação do *Agile Alliance* - nome dado ao grupo de pensadores, a identificação de 12 princípios e a publicação do Manifesto Ágil de Desenvolvimento de *Software* que os representa com quatro valores:

* Indivíduos e iterações são mais importantes do que processos e ferramentas;
* *Software* funcionando é mais importante do que documentação completa;
* Colaboração com o cliente é mais importante do que negociação de contratos;
* Adaptação a mudanças é mais importante do que seguir o plano inicial;

Esses valores são concretizados nos 12 princípios que a *Agile Alliance* estabelece, conforme destaca Pressman (2011)

1. A maior prioridade é satisfazer o cliente com entrega adiantada e contínua de *software* funcionando.

2. Aceite bem os pedidos de alterações, mesmo com o desenvolvimento adiantado. Os processos ágeis se aproveitam das mudanças para a vantagem competitiva do cliente.

3. Entregue *software* em funcionamento frequentemente, de algumas semanas a alguns meses, dando preferência a intervalos mais curtos.

4. O pessoal do comercial e os desenvolvedores devem trabalhar em conjunto diariamente ao longo de todo o projeto.

5. Construa projetos em torno de pessoas motivadas. Dê a elas o ambiente e o apoio necessários e acredite que elas farão o trabalho corretamente.

6. O método mais eficiente e efetivo de transmitir informações para e dentro de uma equipe de desenvolvimento é uma conversa aberta, presencial.

7. *Software* em funcionamento é a principal medida de progresso.

8. Os processos ágeis promovem desenvolvimento sustentável. Proponentes, desenvolvedores e usuários devem estar aptos a manter um ritmo constante indefinidamente.

9. Atenção contínua para com a excelência técnica e para com bons projetos aumenta a agilidade.

10. Simplicidade a arte de maximizar o volume de trabalho não realizado, é essencial.

11. As melhores arquiteturas, requisitos e projetos surgem de equipes auto-organizadas.

12. Em intervalos regulares, a equipe se avalia para ver como pode se tornar mais eficiente, então, sintoniza e ajusta seu comportamento de acordo.

Ainda nesse contexto de princípios Sommerville (2007) afirma que os métodos ágeis compartilham um conjunto de princípios e tem muito em comum, mas propõem processos diferentes entre si.

Esses princípios são apresentados a seguir, na Tabela 1

**Tabela 1** - Princípios dos métodos ágeis

|  |  |
| --- | --- |
| Princípio | Descrição |
| Envolvimento do cliente | Clientes devem ser profundamente envolvidos no processo de desenvolvimento, com o papel de fornecer e priorizar novos requisitos do sistema e avaliar as iterações do sistema. |
| Entrega Incremental | O software é desenvolvido em incremento e o cliente especifica os requisitos a serem incluídos em cada incremento |
| Pessoas, não processos | As habilidades da equipe de desenvolvimento devem ser reconhecidas e exploradas. Os membros da equipe devem desenvolver suas próprias maneiras de trabalhar sem processos prescritivos. |
| Aceite as mudanças | Tenha em mente que os requisitos do sistema vão mudar, por isso projete o sistema para acomodar essas mudanças. |
|  |  |
| Mantenha a simplicidade | Concentre-se na simplicidade do *software* que está sendo desenvolvido e do processo de desenvolvimento. Sempre que possível, trabalhe ativamente para eliminar complexidade do sistema. |

**Fonte:** Adaptado de Sommerville (2007, p.263)

Pressman (2011) enfatiza que nem todo modelo de processo ágil aplica esses 12 princípios de forma igualitária, alguns modelos preferem ignorar (ou pelo menos subestimam) a importância de um ou alguns desses princípios. Contudo, os princípios definem um espírito ágil mantido em cada um dos modelos de processo.

Os métodos ágeis utilizam uma abordagem iterativa e incremental, permitindo a especificação, desenvolvimento e entrega de software de forma contínua (SOMMERVILLE, 2007).

Segundo Sommerville (2007), entre as metodologias ágeis, a mais conhecida é a *Extreme Programming* (XP), outras metodologias existentes incluem Scrum, Crystal, Método de Desenvolvimento de Sistemas Dinâmicos (DSDM), Desenvolvimento Dirigido a Funcionalidade (FDD), entre outras.

* + - 1. ***Extreme Programming* (XP)**

Conforme Pressman (2011), o XP é uma metodologia ágil que enfatiza o desenvolvimento rápido do projeto e visa garantir a satisfação do cliente, além de favorecer o cumprimento das estimativas. As práticas e valores da XP proporcionam um agradável ambiente de desenvolvimento de *software* para os seus adeptos, que são conduzidos por cinco valores:

* Comunicação: A finalidade desse valor é manter o melhor relacionamento possível entre clientes e desenvolvedores, preferindo conversas pessoais a outros meios de comunicação. A comunicação entre os desenvolvedores e o gerente do projeto também é encorajada;
* Simplicidade: Permitir a criação de código simples que não deve possuir funções desnecessárias. Por código simples entende-se implementar o *software* com o menor número possível de classes e métodos. Outra ideia importante da simplicidade é projetar apenas para as necessidades imediatas, ao invés de se preocuparem com as necessidades futuras, se o projeto precisar ser melhorado, ele poderá ser refatorado.
* *Feedback*: A prática do *feedback* constante implica em fornecer ao programador informações regulares provenientes tanto do código quanto do cliente. Testes frequentes fornecem *feedback* sobre erros no código, enquanto o cliente recebe regularmente partes do *software* funcionais para avaliação, permitindo sugerir novas características. Essa abordagem facilita a identificação rápida e correção de erros, resultando em um produto alinhado às expectativas do cliente.
* Coragem: Adotar a prática do XP exige coragem, ou melhor, disciplina, pois esse valor dá suporte aos outros. Uma equipe XP ágil deve ter disciplina para projetar para hoje, reconhecendo que as necessidades podem mudar drasticamente, consequentemente um retrabalho venha acontecer, em relação ao projeto e o desenvolvimento do código. Além disso, é preciso coragem para obter *feedback* constante do cliente.
* Respeito: Nesse valor a equipe recomenda respeito entre os membros, entre os outros envolvidos e, indiretamente ao próprio *software. C*onforme conseguem entregar com sucesso incrementos *do software,* a equipe desenvolve cada vez mais respeito pelo processo. (PRESSMAN, 2011**)**

Já Sommerville (2007) afirma que o XP é uma abordagem desenvolvida pelo avanço da boa prática, desenvolvimento iterativo e o envolvimento do cliente em níveis extremos e que envolve um número de práticas que se enquadram aos métodos ágeis. Na Tabela 2 observamos essas práticas resumidas.

**Tabela 2** - Práticas da *Extreme Programming*

|  |  |
| --- | --- |
| Princípio ou Prática | Descrição |
| Planejamento | consiste em decidir o que é necessário ser feito e o que pode ser adiado no projeto. A XP baseia-se em requisitos atuais para desenvolvimento de *software*, não em requisitos futuros. Procura evitar os problemas de relacionamento entre clientes e a área de desenvolvimento, ambos devem cooperar para o sucesso do projeto. |
| Pequenos *Releases* | O conjunto mínimo útil de funcionamento que agrega valor ao negócio é desenvolvido primeiro. *Realeses* são frequentes e adicionam funcionalidade incrementalmente. |
| Projeto simples | É realizado um projeto suficiente para atender aos requisitos atuais e nada mais. |
| Testes | Foca na validação do projeto durante todo o processo de desenvolvimento. Os programadores desenvolvem o *software* criando primeiramente os testes. |
| Refatoração | Focaliza o aperfeiçoamento do projeto do *software* e está presente em todo o desenvolvimento. A refatoração deve ser feita apenas quando é necessário, ou seja, quando um desenvolvedor da dupla, ou os dois, percebe que é possível simplificar o  módulo atual sem perder nenhuma funcionalidade. |
| Programação em pares | Os desenvolvedores trabalham em pares, um verificando o trabalho do outro e fornecendo apoio para realizar sempre um bom trabalho. Uma grande vantagem desse tipo de programação é a possibilidade dos desenvolvedores estarem continuamente aprendendo um com o outro. |
| Propriedade coletiva | O código do projeto pertence a todos os membros da equipe, ou seja, qualquer pessoa que percebe que pode adicionar valor a um código, mesmo que ele próprio não o tenha desenvolvido, pode fazê-lo, desde  que faça a bateria de testes necessária. Isto é possível porque todos são responsáveis pelo *software* inteiro. Uma grande vantagem desta prática é que, caso um membro da equipe deixe o projeto antes do fim, a equipe consegue continuar o projeto com poucas dificuldades, pois todos conhecem todas as partes do *software*, mesmo que não seja de forma detalhada. |
| Integração continua | Quando uma tarefa é concluída, este é integrado ao sistema como um todo. Depois de qualquer integração, todos os testes unitários devem ser realizados. |
| Ritmo sustentável | Grandes quantidades de horas extras não são consideradas aceitáveis, pois, no médio prazo, há uma redução na qualidade do código e na produtividade. |
| Cliente presente | É fundamental a participação do cliente durante todo o  desenvolvimento do projeto. O cliente deve estar sempre disponível para sanar todas as dúvidas de requisitos, evitando atrasos e até mesmo construções  erradas. Uma ideia interessante é manter o cliente como parte integrante da equipe de desenvolvimento. |

**Fonte:** Adaptado de Sommerville (2007, p.264)

* + - 1. **Scrum**

De acordo com Pressman (2011) a metodologia Scrum foi desenvolvida no início de 1990, é um método de desenvolvimento ágil, seus princípios são consistentes com o manifesto ágil e são usados para orientar as atividades de desenvolvimento dentro de um processo.

Segundo (SCHWABER; SUTHERLAND, 2020), Essa metodologia emprega uma abordagem iterativa e incremental para otimizar a previsão e controlar o risco, envolve grupos de pessoas que, em conjunto, possuem todas as habilidades e conhecimentos necessários para fazer o trabalho e compartilhar ou adquirir essas habilidades conforme necessário.

“Scrum é baseado no empirismo e *lean thinking*. O empirismo afirma que o conhecimento vem da experiência e da tomada de decisões com base no que é observado. O *lean thinking* reduz o desperdício e se concentra no essencial.” (SCHWABER; SUTHERLAND, 2020, p.4)

Os autores citam ainda, sobre os pilares empíricos do Scrum, que são:

* Transparência: O Scrum preza pela visibilidade do processo e do trabalho, essencial para tomar decisões com base nos artefatos formais. A falta de transparência pode levar a escolhas prejudiciais, diminuindo o valor e aumentando o risco.
* Inspeção: A inspeção frequente dos artefatos do Scrum e do progresso em direção às metas é crucial para detectar variações ou problemas. Essa prática permite a adaptação, essencial para promover mudanças e evitar problemas maiores.
* Adaptação: Quando algum aspecto do processo ou do produto não está dentro dos limites aceitáveis, ajustes devem ser feitos rapidamente para evitar desvios maiores. A adaptação é mais fácil quando as equipes são capacitadas e autogerenciadas, permitindo que respondam efetivamente às novas aprendizagens. (SCHWABER; SUTHERLAND, 2020)

Além disso, Schwaber e Sutherland (2020, p. 5) enfatizam que o sucesso do Scrum depende da proficiência das pessoas em viver cinco valores fundamentais: Compromisso, Foco, Abertura, Respeito e Coragem.

Na Figura 9, podemos visualizar esses valores.

**Figura 9** - Valores do Scrum

Forma

Descrição gerada automaticamente

**Fonte:** Próprio autor

**Compromisso**: O valor Scrum do comprometimento é crucial para criar uma cultura ágil. As equipes colaboram para alcançar a meta do produto e as metas de sprint, confiando umas nas outras para cumprir o que prometem, se houver dúvidas sobre o progresso do trabalho, os membros da equipe perguntam. As equipes se comprometem apenas com tarefas que acreditam poder concluir, evitando assumir responsabilidades excessivas.

**Foco**: Cada membro da equipe se dedica ao trabalho atual para contribuir com a meta da *sprint*.

**Abertura**: As equipes Scrum buscam continuamente novas ideias e oportunidades de aprendizado, sendo honestas sobre suas necessidades de ajuda e transparentes com a equipe e partes interessadas sobre os desafios enfrentados.

**Respeito**: Os membros da equipe Scrum respeitam uns aos outros, ao proprietário do produto, às partes interessadas e ao Scrum Master. Eles sabem que sua força está na colaboração, valorizam as contribuições distintas de cada um, respeitam ideias, apoiam-se em momentos difíceis e reconhecem as realizações.

**Coragem**: O valor coragem é crucial para o sucesso da equipe, devem estar seguras para dizer não, pedir ajuda, tentar novas abordagens e questionar o status quo quando este impede seu sucesso. (SCRUM ALLIANCE, 2024)

* + - 1. **Crystal**

Segundo Pressman (2011) a família Crystal de métodos ágeis visa conseguir uma abordagem de desenvolvimento *de software* que priorize a adaptabilidade, comunicação próxima e melhoria contínua com o objetivo de entregar *software* útil e em funcionamento.

Sobre adaptabilidade Pressman (2011) cita que foi definido um conjunto de metodologias com elementos essenciais e comuns a todos, o que diferencia entre eles são os papeis, padrões de processos, produtos de trabalho e práticas. Para uma melhor compreensão a divisão foi feita em cores; quanto mais escuro, mais crítico o sistema seria. Cada cor tem um propósito diferente e são separadas de acordo com a criticidade, as letras (C *confort*, D *discret*, E *essencial*, L *life)* representam o nível de risco do projeto, ou seja, as perdas acarretadas caso haja falhas do sistema. (SBROCCO; MACEDO, 2013)

“A intenção é possibilitar que equipes ágeis selecionem o membro da família Crystal mais apropriado para seu projeto e seu ambiente” (PRESSMAN, 2011, p.97).

A Tabela 3 mostra essa divisão por cores.

**Tabela 3** - Divisão da Família Crystal

|  |  |  |  |
| --- | --- | --- | --- |
| Núcleo | Desenvolvedores | | Em caso de falha... |
| Crystal *Clear* | | 1 - 6 | Perdem dinheiro, mas recuperam facilmente. |
| Crystal *Yellow* | | 7 - 20 | Perdem dinheiro discretamente |
| Crystal *Orange* | | 21 -40 | Perdem dinheiro substancialmente |
| Crystal Red | | 41 - 100. | Há perda substancial de dinheiro, e, possivelmente, de vida humana. |

**Fonte:** Adaptado de (SBROCCO; MACEDO, 2013)

Segundo Sbrocco e Macedo (2013) a família Crystal possui sete princípios básicos que independentemente das cores devem ser seguidos:

1. Trabalho face a face com o cliente: considera que envolver o cliente nas iterações e nas decisões é muito mais produtivo.
2. Peso significa custo: quanto maior a complexidade, maior o custo.
3. Usar metodologias diferenciadas para equipes maiores.
4. Mais cerimônias maior criticidade: quanto mais diálogos com os envolvidos melhor.
5. Comunicação eficiente (realimentação) é melhor que entregas que não funcionam.
6. Habitabilidade: tolerância em lidar com seres humanos.
7. Eficiência no desenvolvimento.
   * + 1. **Método de Desenvolvimento de Sistemas Dinâmicos (DSDM)**

“É uma metodologia de desenvolvimento de projetos de *software* centrada em estabelecer os recursos e o tempo fixo para o desenvolvimento de um projeto, ajustando suas funcionalidades de maneira a atender os prazos estipulados.” (SBROCCO; MACEDO, 2013, p.111)

Pressman (2011) Acrescenta que é um processo de *software* iterativo, onde em cada iteração apenas parte do trabalho suficiente é requisitada para cada incremento, facilitando o movimento para o próximo. O restante pode ser concluído posteriormente, quando outros requisitos do negócio forem conhecidos ou alterações forem solicitadas e acomodadas.

Sbrocco e Macedo (2013) ressaltam que DSDM baseia-se em nove princípios, que são considerados boas práticas de utilização dessa metodologia.

Na Tabela 4 observamos esses princípios de boas práticas:

**Tabela 4** - Princípios de boas práticas da DSDM

|  |  |
| --- | --- |
| Princípio | Descrição |
| Participação ativa dos usuários e *stakeholders*: | Todos os participantes do projeto e aqueles familiarizados com o processo devem monitorar o progresso para assegurar que tudo seja entregue dentro do prazo e conforme o especificado |
| Abordagem cooperativa e compartilhada: | Todas as partes interessadas devem colaborar e se comprometer com as entregas dos componentes do software, além de decidir a ordem de sua implementação. Essas decisões devem ser tomadas em consenso. |
| Equipes com poder de decisão: | As pessoas envolvidas no desenvolvimento devem ter o conhecimento e a autonomia necessários para decidir o destino do sistema. Não é aceitável esperar por decisões durante longos períodos. |
| Entregas contínuas que fazem a diferença: | É um critério básico da DSDM tentar trazer um retorno ao cliente do projeto desde o início, a fim de promover a validação do que está sendo feito. |
| Desenvolvimento iterativo e incremental: | O objetivo é alinhar o sistema com as necessidades do negócio e melhorá-lo continuamente em um processo iterativo, identificando e corrigindo os problemas o mais rapidamente possível. |
| *Feedback:* | O foco está nas frequentes entregas de produtos de *software,* permitindo ao usuário colocar suas opiniões e solicitar modificações. |
| Alterações possíveis durante o processo devem ser reversíveis: | É necessário permitir que o impacto das modificações seja testado e, caso não produzam os efeitos desejados, que as alterações possam ser revertidas. |
| Fixar os requisitos essenciais: | Os requisitos principais devem ser identificados no início para estabelecer os objetivos gerais. Os requisitos específicos serão orientados pelos principais e poderão ser ajustados, mas sempre com foco nos objetivos. |
| Teste em todo o ciclo de vida: | Devido ao prazo geralmente apertado, não podemos deixar a fase de testes apenas para o final da implementação. Os testes devem ser realizados em todas as fases e componentes do projeto. O teste de regressão é frequentemente utilizado pela DSDM devido ao desenvolvimento incremental. |

**Fonte**: Adaptado de Sbrocco e Macedo 2013

Além desses princípios Sbrocco e Macedo (2013) reforçam que a DSDM possui conceitos específicos como o '*Timeboxing'* que envolve dividir o projeto em partes com orçamentos próprios e prazos definidos, trata-se do encapsulamento do tempo e a técnica '*MoSCoW*' (*Must have, Should have, Could have, Won't have*) usada para priorizar requisitos. Estes conceitos ajudam na definição de prioridades durante a análise do projeto. Além disso, a prototipagem é crucial para identificar problemas e limitações antecipadamente.

Traduzindo a técnica *MoSCow*:

* *Must have*: TEM que ter isto;
* *Should have:* DEVE ter isto e se for possível completamente;
* *Could have*: PODE ter isto se não afetar o resto;
* *Won't have*: NÃO VAI ter isto agora, mas SERIA bom ter no futuro.

Pressman (2011) acrescenta que essa metodologia possui um modelo de processos ágeis, conhecido como Ciclo de Vida DSDM:

* Iteração de modelos funcionais — Produz protótipos incrementais que demonstram funcionalidades ao cliente, com a intenção de evoluí-los para a aplicação final. Durante o ciclo iterativo, busca-se reunir requisitos adicionais por meio do feedback dos usuários ao testarem os protótipos.
* Iteração de projeto e desenvolvimento - Revisa os protótipos desenvolvidos durante as iterações de modelos funcionais para garantir que tenham passado por um processo de engenharia, visando proporcionar valor operacional aos usuários. Em alguns casos, as iterações de modelos funcionais e de projeto e desenvolvimento podem ocorrer simultaneamente.
* Implementação - Coloca a última versão do incremento de *software*, um protótipo 'operacionalizado', no ambiente operacional. É importante notar que o incremento pode não estar completamente finalizado ou que alterações podem ser solicitadas após a implantação. Em ambos os casos, o trabalho de desenvolvimento do DSDM continua voltando à iteração do modelo funcional. (PRESSMAN, 2011)
  + - 1. **Desenvolvido Dirigido a Funcionalidade (FDD)**

De acordo com Pressman (2011) FDD foi concebido como um modelo de processos prático para a engenharia de *software* orientada a objetos, um processo ágil adaptativo que pode ser aplicado a projetos moderados e a projetos maiores.

Como outras abordagens ágeis a FDD oferece um método de trabalho que satisfaz todas as partes envolvidas no projeto, propondo interações e controles simples e eficientes. Os desenvolvedores se sentem confortáveis durante a implementação, graças a um conjunto de regras compreensíveis e com resultados rápidos, o que também traz benefícios ao cliente: (SBROCCO; MACEDO, 2013)

* Benefício ao cliente por meio de trabalho significativo;
* Atende equipes pequenas, médias ou grandes;
* *Software* de qualidade;
* Entrega de resultados frequentes, tangíveis e funcionais;
* Permite o acompanhamento do progresso do desenvolvimento do projeto.

Sbrocco e Macedo (2013) ressaltam as práticas dessa metodologia:

* Modelagem de objetos do domínio: é uma atividade colaborativa realizada pela equipe para estudar, analisar e modelar um sistema. O modelo visa fornecer uma visão geral do projeto, demonstrar o que será produzido e orientar a equipe. Geralmente, resulta na produção de um documento de requisitos, obtido por meio de métodos como entrevistas, questionários ou formulários de pesquisa, auxiliados por diagramas da UML para facilitar a implementação.
* Desenvolvimento por funcionalidade (*feature*): prática sugerida a partir dos requisitos, construir uma lista de funcionalidades sobre o domínio do negócio. Categorizada em três níveis:

a) Áreas de Negócio (*Business Areas, Major Feature Sets*)

b) Atividades de Negócio (*Business Activities, Feature Sets*)

c) Passos da Atividade de Negócio (*Activity Steps, Features*)

* Entregas regulares (*builds*): Isso garante que os desenvolvedores atualizem continuamente o *software* com novas funcionalidades e modificações. Permitindo que todos trabalhem com a versão mais recente do projeto, beneficiando tanto os desenvolvedores quanto os clientes. É essencial testar as mudanças e manter um controle de versão para lidar com problemas potenciais de impacto.
* Formação da Equipe do Projeto: Equipe de *Feature*
* Posse individual do código: Cada funcionalidade do código deve ter um proprietário designado, isso garante uma posse individual clara do código e permite que o programador responda por problemas de implementação relacionados à sua funcionalidade. (SBROCCO; MACEDO, 2013).

“A FDD é uma metodologia ágil robusta e muito utilizada nos dias atuais”. (SBROCCO; MACEDO, 2013, p.99)

Para o desenvolvimento deste trabalho, será adotada a metodologia ágil, o Scrum adaptado, adequado para o projeto, com equipe menor, sprints com prazo maiores, para garantir flexibilidade e entregas incrementais, permitindo ajustes contínuos conforme o *feedback*. Para estruturar o sistema, utilizaremos a UML (Linguagem de Modelagem Unificada), que facilitará a modelagem e comunicação dos requisitos de forma clara e visual. A combinação dessas abordagens assegura um desenvolvimento dinâmico, mas bem estruturado, alinhado aos objetivos do projeto.

* + 1. **Linguagem de Modelagem Unificada – UML**

“É uma linguagem visual utilizada para modelar *softwares* baseados no paradigma de orientação a objetos. É uma linguagem de modelagem de propósito geral que pode ser aplicada a todos os domínios de aplicação.” (GUEDES, 2011, p.19)

De acordo com Pressman (2011, p.727) “é uma linguagem-padrão para descrever/documentar projeto de software. A UML pode ser usada para visualizar, especificar, construir e documentar os artefatos de um sistema de *software* intensivo”.

Segundo Guedes (2011), a UML nasceu da fusão de três métodos de modelagem orientada a objetos populares na década de 1990: o método de Booch, o OMT de Jacobson e o OOSE de Rumbaugh. A *Rational Software* apoiou e financiou essa união. O esforço inicial resultou no Método Unificado, que combinava *Booch* e OMT. Posteriormente, Rumbaugh se juntou à equipe e seu método OOSE foi incorporado. Conhecidos como 'Os Três Amigos', Booch, Jacobson e Rumbaugh lançaram a primeira versão da UML em 1996.

Portanto, em 1997, a linguagem de modelagem unificada foi adotada pela Object Management Group (OMG), que tem como objetivo a padronização de notações orientadas a objetos (PRESSMAN, 2011).

* + - 1. **Tipos de Diagrama UML**

Dentre tantos diagramas existentes, serão abordados aqueles mais relevantes para o contexto específico do projeto, visando uma compreensão clara e uma representação eficaz de suas necessidades e processos.

* **Diagrama de Caso de Uso**

O diagrama de casos de uso é uma representação ampla e informal da UML, frequentemente usado nas fases iniciais de levantamento e análise de requisitos, embora seja consultado em todo o processo de modelagem. Ele oferece uma linguagem simples para compreensão, permitindo aos usuários entenderem como o sistema irá operar. Identifica os atores (usuários, sistemas externos etc.) e os serviços (ou casos de uso) que o sistema oferecerá a esses atores. (GUEDES, 2011).

Na Figura 10, observa-se um exemplo desse tipo de diagrama.

**Figura 10** - Exemplo Diagrama de Caso de Uso

Diagrama

Descrição gerada automaticamente

**Fonte:** Guedes (2011, p.32)

* **Diagrama de Classe**

O diagrama de classes, um dos mais importantes e amplamente utilizado, é essencial para a maioria dos demais diagramas. Ele descreve a estrutura das classes do sistema, especificando seus atributos e métodos, além de definir os relacionamentos e interações entre as classes. (GUEDES, 2011)

Na Figura 11, apresenta um exemplo desse tipo de diagrama.

**Figura 11** - Exemplo de Diagrama de Classe

Diagrama

Descrição gerada automaticamente

**Fonte:** Guedes (2011, p.33)

* **Diagrama de Sequência**

O diagrama de sequência é um diagrama comportamental que registra a ordem temporal das mensagens trocadas entre os objetos em um processo específico. Geralmente, é baseado em um caso de uso e utiliza o diagrama de classes para identificar os objetos envolvidos. Ele detalha o evento inicial do processo, o ator responsável e como o processo acontece, incluindo a chamada de métodos por meio de mensagens entre os objetos. (GUEDES, 2011).

Na Figura 12, observa-se um exemplo para esse diagrama.

**Figura 12** - Exemplo de Diagrama de Sequência

Diagrama

Descrição gerada automaticamente

**Fonte:** Guedes (2011, p.36)

* **Diagrama de Atividades**

O diagrama de atividade representa o comportamento dinâmico de um sistema, mostrando o fluxo de controle entre ações. Utiliza nós de ação para representar tarefas e setas para indicar o fluxo entre elas. O início da atividade é marcado por um ponto preto cheio, enquanto o fim é marcado por um ponto dentro de um círculo preto. O fork representa a divisão de atividades em múltiplas tarefas concorrentes. (PRESSMAN, 2011).

Guedes (2011) afirma que o diagrama de atividade descreve os passos necessários para concluir uma atividade específica, que pode variar em complexidade e ser representada por um método, algoritmo ou processo completo. Ele se concentra na representação do fluxo de controle da atividade.

Na figura 13, é apresentado um exemplo para esse diagrama.

**Figura 13** - Exemplo de Diagrama de Atividade

Diagrama

Descrição gerada automaticamente

**Fonte:** Guedes (2011, p.39)

Os diagramas apresentados serão utilizados no desenvolvimento do projeto, servindo como ferramentas essenciais para a modelagem e compreensão do sistema, garantindo uma abordagem clara e estruturada.

* + 1. **Desenvolvimento**

Para a construção deste projeto, foi essencial a utilização de diversas tecnologias e linguagens de programação que viabilizam uma aplicação dinâmica, funcional e responsiva. Cada tecnologia escolhida, como HTML, CSS, *JavaScript*, PHP, *Dart* e *Flutter*, contribui de maneira única para a estruturação, apresentação, funcionalidade e desempenho da aplicação. Complementando essas tecnologias, o banco de dados *MySQL* foi adotado para o armazenamento seguro e eficiente das informações. A seguir, são apresentadas as principais características e papéis dessas tecnologias no desenvolvimento do sistema.

* + - 1. **HTML**

De acordo com Silva (2008, p. 26), “HTML é a sigla em inglês para *Hyper Text Markup Language*, que em português significa linguagem para marcação de hipertexto”. É o componente básico da web, utilizado para definir a estrutura e o significado do conteúdo online. Apesar de outras tecnologias contribuírem para a apresentação e funcionalidade de uma página, o HTML é fundamental para conectar hipertextos por meio de links, permitindo a navegação entre páginas de um mesmo site ou entre sites diferentes. Assim, ao publicar conteúdo na Internet e vinculá-lo a outras páginas, o usuário se torna um participante ativo da *World Wide Web* (MOZILLA HTML, 2024).

Além disso, por ser uma linguagem de marcação, o HTML aceita formatação semântica de apresentação, permitindo que se defina a maneira como a informação será exibida ao usuário, o que contribui para uma melhor organização e acessibilidade do conteúdo (FALSTCHART, 2011). A linguagem HTML é composta por *tags* (marcadores) delimitadas por sinais como < > e </>, que identificam a função e o conteúdo de cada elemento, incluindo cabeçalhos, corpo de página, títulos, imagens e rodapés (SILVA, 2008).

Segundo o W3 (2024), a história do HTML começou em 1989, quando Tim Berners-Lee inventou a *World Wide Web* (WWW). Em 1991, ele desenvolveu a primeira versão do HTML. Desde então, a linguagem evoluiu com atualizações significativas, culminando na versão HTML 5.2, lançada em 2017, que trouxe melhorias na integração com CSS (*Cascading Style Sheets*) e *JavaScript*.

Silva (2008), ressalta que o HTML é responsável por estruturar os elementos de uma página, enquanto o CSS auxilia na criação de estilos visuais, e o J*avaScript* permite programar o comportamento e a lógica dos elementos.

* + - 1. **Java Script**

De acordo com Noleto (2024), *JavaScript* foi criado em 1995 por Brendan Eich, inicialmente com os nomes *Mocha* e *LiveScript,* antes de ser rebatizado como *JavaScript*, em uma estratégia de marketing que aproveitou o sucesso da linguagem Java. Sua implementação trouxe dinamismo à *web*, permitindo interações diretas com os usuários, como animações e ações automatizadas. O *Netscape Navigator* foi o primeiro navegador a dar suporte à linguagem, marcando um avanço significativo na navegação online.

Segundo a Mozilla JavaScript (2024), a linguagem é reconhecida por sua leveza e flexibilidade, sendo amplamente utilizada em páginas web e em ambientes como Node.js. JavaScript é uma linguagem multiparadigma, permitindo a adoção de estilos de programação orientada a objetos, funcional e baseada em eventos (NEGRINO, 2000).

Sua sintaxe simples possibilita até mesmo iniciantes implementarem interatividade em páginas HTML, como menus dinâmicos, validação de formulários e personalização da experiência do usuário com base nas características do navegador. Além disso, JavaScript desempenha um papel essencial na criação de interfaces modernas e dinâmicas. Por ser uma linguagem de alto nível, suas instruções se assemelham à linguagem humana e são processadas diretamente pelos navegadores no lado cliente, consolidando-se como um pilar indispensável do desenvolvimento web (NOLETO, 2024).

A linguagem permite a inclusão de interatividade diretamente em páginas HTML, controlando elementos que respondem a ações do usuário, como menus, janelas e painéis de conteúdo dinâmico. Ela também pode configurar ações automáticas, como o carregamento de uma página ou interações com componentes específicos, além de ser capaz de ler e modificar o conteúdo dos elementos HTML, validar dados de formulários antes do envio e identificar as características do navegador, redirecionando o usuário para páginas compatíveis quando necessário (FLASTCHART, 2011).

* + - 1. **PHP - *Hypertext Preprocessor***

É uma linguagem de programação para *web* de *script* *open-source*, mundialmente utilizada, trabalha mesclado ao HTML (Hypertext Markup Language), tornando mais fácil a geração de páginas *Web*, é executado no lado servidor, o que possibilita que o site seja dinâmico (SOARES, 2013).

Essa linguagem teve sua criação em 1994 por Rasmus Lerdorf, formada por um conjunto de *scripts* voltados a criação de páginas dinâmicas para monitorar acesso ao seu currículo na internet (DALL’OGLIO,2009).

O PHP, inicialmente criado para uso em páginas pessoais, evoluiu significativamente ao longo do tempo. À medida que as demandas aumentaram, novas versões foram lançadas, incorporando recursos avançados como um interpretador de comandos SQL (*Structured Query Language*) uma linguagem de consulta estruturada, suporte à orientação a objetos, herança, polimorfismo, apontadores de memória, entre outros (PHP.NET, 2024). Atualmente, PHP é amplamente utilizado, sendo uma das linguagens preferidas para desenvolvimento web em grande escala (W3Techs, 2024).

Na Figura 14 podemos observar essa utilização:

**Figura 14** -Estatísticas de uso de linguagens de programação para sites

Interface gráfica do usuário

Descrição gerada automaticamente com confiança média

**Fonte:** W3Techs.com (2024)

Essa linguagem também disponibiliza suporte a um grande número de bases de dados, inclusive *MySQL* o qual será utilizado no contexto desse trabalho.

Segundo Bento (2016), alguns motivos de usar o PHP e o *MySQL* são: PHP nasceu para a *web* e sua integração com servidores *web* é simples, tem uma curva de aprendizado suave, comparada a outras linguagens, PHP e MySQL são tecnologias livres, é fácil de encontrar serviços de hospedagem que oferecem PHP e MySQL. Serviços de hospedagem que as oferecem são mais baratos que serviços semelhantes para outras tecnologias e *MySQL* é leve e rápido, mesmo para quantidades razoavelmente grandes de dados.

* + - 1. **CSS - *Cascading Style Sheets***

O *Cascading Style Sheets* (CSS), permite descrever qual o modo da apresentação de páginas na *web*, com uma linguagem de estilo simples e legível e compreensível por humanos e com codificações compactas na maneira de apresentar o conteúdo comparado a imagens ou ficheiros de áudio, que são frequentemente usados, de modo a obter efeitos de renderização; as

folhas de estilo geralmente diminuem o tamanho do conteúdo, também facilitam na simplificação da manutenção do site de maneira a manter uma aparência, tal como os elementos HTML são apresentados, em uma página, possibilitando a escolha de cores, posicionamento no *layout*, fontes, bordas, entre outros, sendo possível descrever código CSS diretamente dentro de um documento HTML consistente em todo o sistema ou site (MCFARLANDY, 2012).

Para gerir a apresentação e a formatação de um documento HTML usa-se o CSS devido ao seu estilo e linguagem; com o CSS podemos aplicar todos os estilos de modo a formatar os documentos HTML, como seja, inserir cores, posicionamentos, tamanhos, entre outros tipos de formatação (MEYER, 2000).

* + - 1. **Dart**

Dart é uma linguagem projetada para criar aplicativos rápidos e eficientes em diversas plataformas, com foco no desenvolvimento ágil e de alta qualidade para o usuário. Essa linguagem é a base do *Flutter*, *framework* também desenvolvido pelo *Google*, e inclui diversas ferramentas para formatação, análise e teste de código, proporcionando uma experiência versátil que abrange desde a *web* até dispositivos móveis e *desktops* (DART.DEV, 2024).

Lançada em 2011, a linguagem inicialmente tinha como objetivo substituir o *JavaScript* como principal opção para desenvolvimento em navegadores (ANDRADE, 2019). No entanto, ao longo dos anos, Dart evoluiu, tornando-se uma linguagem multi-paradigma, embora com características robustas de orientação a objetos (GUEDES, 2019). Uma linguagem madura para interfaces de usuário, destacando-se por: (a) sua otimização para desenvolvimento de interfaces, proporcionando um desempenho aprimorado para o usuário final; (b) o recurso de "*Hot Reload*", que facilita a produtividade durante o desenvolvimento; e (c) sua capacidade de compilar para código de máquina ARM (*Advanced RISC Machine*) e x64, o que permite a execução eficiente em diversas plataformas, além da compilação para *JavaScript* para sistemas *web* (PEREIRA, 2019).

* + - 1. **Flutter**

Segundo Alura (2023), *Flutter* é um *framework* de desenvolvimento multiplataforma criado pela *Google*, focado principalmente em dispositivos móveis, mas também capaz de criar aplicações para desktop (Linux, *Windows* e macOS). Utilizando a linguagem Dart, também da *Google*, permite escrever um único código que executa tanto em *Android* quanto em *iOS*, oferecendo vantagens como versatilidade e agilidade. Entre seus recursos estão:

* *Hot reload*, que atualiza rapidamente o código;
* Executar *templates* em navegador e
* *Null safety,* que melhora a adaptabilidade do código ao permitir valores nulos sem quebrar a aplicação, o que o torna ideal para quem buscam otimizar o desenvolvimento em múltiplas plataformas.

Flutter é um SDK (*Software Development Kit*) multiplataforma criado para desenvolver aplicativos com alta performance e interface nativa a partir de um único código (FLUTTER, 2019). Segundo Abranches (2018), ele compila o código em ARM, otimizando a velocidade e o desempenho dos apps, e utiliza *Widgets* para definir o layout, eliminando a necessidade de arquivos separados como *JavaScript*, HTML e CSS. Com performance similar à de aplicativos nativos, permite interfaces personalizadas e animações, além de facilitar o acesso aos recursos do aparelho (PEREIRA, 2019). De fácil instalação e configuração, com um desenvolvimento mais rápido que outros *frameworks*, como o *React Native* (CAPPELLI, 2018).

* + - 1. **Banco de Dados**

De acordo com Elmasri e Navathe (2011), banco de dados é uma coleção logicamente coerente de dados com algum significado próprio, contudo uma variedade de dados aleatórios não pode ser apontada como um banco de dados. O banco de dados é projetado, construído e alimentado com dados para uma finalidade específica.

Segundo Ferrari (2007), banco de dados é um local no qual é possível armazenar informações, para consulta ou utilização, quando necessário. Todos os bancos de dados são constituídos por três elementos básicos: campos, registros e tabelas. Cada banco de dados possui complexas estruturas internas de funcionamento, uma diferente da outra, a fim de facilitar o acesso aos elementos do banco de dados.

O SQL (Structured Query Language), criado e distribuído pela IBM em 1981, é uma linguagem de consulta estruturada amplamente utilizada para acessar bancos de dados. É funcional, articulada e compatível com diferentes arquiteturas computacionais, sendo a base para o MySQL (FERRARI, 2007).

* + - * 1. **MySQL e Firebase**

O MySQL e o Firebase são escolhidos neste projeto por suas distintas e complementares vantagens, oferecendo uma solução robusta e eficiente para o desenvolvimento de sistemas de banco de dados.

O MySQL é um sistema de gerenciamento de banco de dados relacional (SGBDR) com licença dupla, incluindo uma opção de software livre. Inicialmente projetado para aplicações de pequeno e médio porte, atualmente ele suporta grandes aplicações com vantagens competitivas (MILANI, 2006). Um banco de dados relacional organiza informações em relações predefinidas, onde os dados são armazenados em tabelas compostas por colunas e linhas, facilitando a visualização e compreensão de como diferentes conjuntos de dados se inter-relacionam. Os relacionamentos são conexões lógicas entre diferentes tabelas, baseadas na interação entre elas (GOOGLE CLOUD, 2024).

Reconhecido por sua robustez, o MySQL é considerado o banco de dados *open source* que possibilita a entrega econômica de aplicações de banco de dados confiáveis, de alto desempenho e escaláveis, com base na web e incorporadas (ORACLE, 2024).

Segundo Milani (2006), o MySQL foi desenvolvido como uma extensão da linguagem de programação SQL, baseada no modelo relacional proposto por Edgar F. Codd. A empresa sueca MySQL AB, fundada por David Axmark, Allan Larsson e Michael Widenius, lançou o MySQL em 1995. O nome "MySQL" combina o nome da filha de Michael Widenius, "My", com "SQL", que significa Structured Query Language. Em 2008, a Sun Microsystems adquiriu a MySQL AB, e, posteriormente, em 2010, a Oracle Corporation adquiriu a Sun Microsystems, tornando-se a atual proprietária do MySQL.

As operações suportadas por esse banco são: inclusão, exclusão, atualização de registros e alterações de estrutura. Além de seu excelente desempenho, o MySQL possui compatibilidade e fácil integração com a linguagem de programação PHP (BENTO, 2016).

Esses fatores foram motivos para a escolha do MySQL, compatível com PHP, uma solução rápida, confiável, escalável e fácil de usar. Ele pode ser executado confortavelmente em um computador, ao lado de outros aplicativos e servidores web. Isso o torna uma escolha prática, simplificando o gerenciamento e a operação de sistemas de banco de dados.

Além disso, o *Firebase* é um *Backend as a Service* (BaaS) criado pelo *Google* em 2004, oferecendo uma estrutura completa de *backend* pronta para integração com aplicativos móveis e web, incluindo configuração de servidor, banco de dados, notificações e segurança integrada (FIREBASE, 2024). Ele é ideal para protótipos, pequenas aplicações e produtos simples e acessíveis, como o proposto neste projeto. Os serviços se dividem em três categorias. No grupo de desenvolvimento, estão recursos como o *Cloud Firestore* e o *Realtime Database* para armazenamento e sincronização de dados, além do *Cloud Storage* para arquivos e *Authentication* para autenticação segura (FIREBASE, 2024). No grupo de melhoria, encontram-se o *Crashlytics*, que fornece relatórios detalhados de erros, e o *Performance Monitoring*, que monitora o desempenho do aplicativo (FIREBASE, 2024). Já o grupo de crescimento inclui o *Google Analytics*, para análise de uso, *Cloud Messaging*, para envio de notificações, e *Remote Config*, que permite modificar o app sem a necessidade de novas atualizações (FIREBASE, 2024).

O *MySQL* e o *Firebase* se complementam ao combinar a robustez e confiabilidade do *MySQL* para gestão de grandes volumes de dados estruturados e transações complexas, com a agilidade e funcionalidades avançadas do *Firebase* para desenvolvimento rápido, autenticação de usuários e sincronização de dados em tempo real. Essa integração oferece uma base sólida e flexível para atender às necessidades do projeto, aproveitando as melhores características de cada tecnologia para criar uma solução eficaz e escalável.

***Firebase Authentication***

O *Firebase Authentication* é um produto do *Firebase* que fornece serviços de *back-end*, SDKs fáceis de usar e bibliotecas de IU (Interface Usuário) prontas para autenticar usuários no aplicativo. Além disso, oferece serviços de autenticação utilizando usuário e senha, números de telefone, conta do *Google, Facebook, Twitter*, entre outros (FIREBASE, 2024). Na Figura 15, podemos observar os tipos de autenticação.

**Figura 15** - Diferentes tipos de login do *Firebase* *Authentication*

Interface gráfica do usuário

Descrição gerada automaticamente com confiança média

Fonte: FireBase, 2024

***Firebase Cloud Firestore***

*Cloud Firestore* é um banco de dados flexível e escalável para o desenvolvimento de aplicações móveis, *Web* e servidores, fornecido pelo Firebase e *Google Cloud Platform*. Trata-se de um banco de dados *NoSQ*L baseado em nuvem, acessível por iOS, *Android*, Node.js, Java, Python, Go e *Web* por meio de diferentes SDKs nativos (FIREBASE, 2024).

O *Firestore* segue o modelo *NoSQL* e armazena dados em formato *JSON*. Os dados são organizados em coleções *(collections)* e documentos *(documents)*. Cada documento funciona como um objeto com chaves e valores, que podem ser booleanos, inteiros, *floats, strings, arrays*, datas, *null*, entre outros. As coleções reúnem documentos, sendo que coleções não podem conter outras coleções, e documentos não podem conter outros documentos (FERNANDES, 2017). A Figura 16 mostra o exemplo de uma coleção (retângulo amarelo) e seus respectivos documentos (retângulos azuis).

**Figura 16** - Exemplo de documentos agrupados em uma coleção

**Interface gráfica do usuário

Descrição gerada automaticamente**

**Fonte:** FireBase, 2024

# ESTUDO DE CASO

O envelhecimento populacional é um fenômeno global que tem desafiado os sistemas de saúde e sociais em todo o mundo. No Brasil, assim como em muitos outros países, a proporção de idosos na população está em constante crescimento, o que requer soluções inovadoras para garantir que essa parcela significativa da sociedade tenha acesso adequado a serviços de saúde, mobilidade e entretenimento, garantindo-lhes uma qualidade de vida plena e independente. Nesse contexto, o presente estudo de caso se propõe a explorar a concepção e implementação de um aplicativo móvel voltado para atender às múltiplas necessidades dos idosos, bem como promover melhorias para sua saúde física e emocional, mobilidade e bem-estar geral.

No âmbito da saúde, os idosos muitas vezes enfrentam dificuldades para marcar consultas médicas e acessar serviços especializados. A mobilidade pode ser um desafio, especialmente para aqueles que não têm acesso fácil a transporte adequado. Além disso, a solidão e o isolamento social são problemas comuns entre os idosos, o que pode ter um impacto negativo em sua saúde física e mental.

Diante desses desafios, a tecnologia emerge como uma aliada poderosa na busca por soluções inovadoras e acessíveis. O desenvolvimento de um aplicativo móvel específico para os idosos visa abordar essas questões de forma abrangente, proporcionando acesso fácil a serviços de saúde, mobilidade, entretenimento e comunicação, além de promover um estilo de vida ativo e saudável.

## Levantamento de Requisitos

Como o aplicativo será desenvolvido diretamente com foco no público idoso, desenvolveu-se um questionário para descobrir as principais dificuldades e necessidades do público-alvo.

Aplicação desse questionário será no próximo mês para coletarmos as informações necessárias para o levantamento de requisitos do aplicativo.

## Questionário

**Entrevista/Questionário sobre Aplicativo para Idosos**

**Aplicativo CuidaDoso**

\* Indica uma pergunta obrigatória

Uma imagem contendo Texto

Descrição gerada automaticamente

**Informações Gerais**

1 - Nome do Entrevistado \*

2 - Idade: \*

Marcar apenas uma opção.

* entre 18 e 40 anos
* entre 40 e 60 anos
* mais de 60 anos

3 - Relação com o público idoso: \*

Marcar apenas uma opção.

* Idoso
* Cuidador
* Profissional de saúde
* Familiar
* Outros.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4- Sexo:

Marcar apenas uma opção.

* Feminino
* Masculino
* Prefiro não dizer

5- Localidade (cidade/estado): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6- Frequência de uso de dispositivos móveis (smartphone, tablet, etc.):

Marcar apenas uma opção.

* Todos os dias
* Algumas vezes por semana
* Uma vez por semana
* Raramente
* Nunca

7- Conhecimento prévio sobre aplicativos voltados para idosos:

Marcar apenas uma opção:

* Sim, tenho conhecimento e já utilizei esses aplicativos anteriormente.
* Sim, tenho conhecimento, mas nunca utilizei esses aplicativos.
* Não, não tenho conhecimento sobre aplicativos voltados para idosos.

**SOBRE O APLICATIVO:**

8- Você já ouviu falar de aplicativos específicos para idosos antes?  
Marcar apenas uma opção

* Sim
* Não

9- Quais funcionalidades você considera mais importantes em um aplicativo voltado para idosos? \* (Marque todas as opções que se aplicam)

* Marcação de consultas médicas
* Solicitação de transporte para consultas e compromissos
* Acesso a profissionais de saúde e serviços de exercícios físicos
* Notificações para usuários e familiares
* Dicas de exercícios e saúde
* Receitas saudáveis
* Jogos e atividades de entretenimento
* Botão de pânico/emergência
* Cadastro de medicamentos e lembretes de horário
* Outro

10- Você acredita que um aplicativo como esse poderia melhorar a qualidade de vida dos idosos? \*  
Marque apenas uma opção

* Sim
* Não
* Talvez

11- Quais são os principais desafios que você vê na utilização de aplicativos por parte dos idosos? Marque todas que se aplicam.

* Dificuldade em aprender a usar novas tecnologias.
* Problemas de visão que dificultam a leitura na tela do dispositivo
* Dificuldade em digitar em teclados pequenos ou sensíveis ao toque.
* Falta de familiaridade com o vocabulário e os conceitos tecnológicos.
* Insegurança em relação à segurança e privacidade dos dados pessoais.
* Barreiras de acessibilidade, como falta de legendas ou áudio para pessoas com deficiência visual ou auditiva.
* Resistência à mudança ou aversão à tecnologia.
* Outro: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

12- Você tem alguma sugestão ou recurso adicional que gostaria de ver em um aplicativo voltado para idosos? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**SOBRE A EXPERIENCIA DO USUARIO:**

13- Como você avalia a facilidade de uso de aplicativos em geral?  
Marque apenas uma opção

* Muito fácil
* Fácil
* Neutro
* Difícil
* Muito difícil

14 - Quais são os principais aspectos que você considera importantes para garantir que um aplicativo seja fácil de usar para os idosos?  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

15- Você tem alguma experiência pessoal ou observação sobre como os idosos lidam com a tecnologia e aplicativos móveis?  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**REFERÊNCIAS BIBLIOGRÁFICAS**

ABNT - Associação Brasileira De Normas Técnicas. NBR ISO 9241-11. **Requisitos ergonômicos para trabalho de escritórios com computadores Parte 11- Orientações sobre usabilidade**. Brasil: ABNT, 2002. Disponível em: https://www.inf.ufsc.br/~edla.ramos/ine5624/\_Walter/Normas/Parte%2011/iso9241-11F2.pdf Acessado em 30 de abril 2024

ABRANCHES, J. **Conhecendo um pouco mais do Flutter.** 2018. Disponível em: <https://imasters.com.br/framework/conhecendo-um-pouco-mais-flutter>. Acesso em: 08 de novembro de 2024.

ALURA. **Flutter**. Disponível em: <https://www.alura.com.br/artigos/flutter>. Acesso em 05 de novembro de 2024.

ANDRADE, K. O. **Introdução a linguagem de programação Dart**. 2019. Disponível em: https://medium.com/flutter-comunidade-br/introdução-a-linguagem-de-programação-dart-b098e4e2a41e. Acesso em: 08 de novembro de 2024.

BARBOSA, S. D. J; SILVA.B. S; **Interação humano-computador**. 380 págs. Rio de Janeiro: Elsevier, 2010. Disponível em: <https://www.academia.edu//41857244/INTERAÇÃO_HUMANO_COMPUTADOR>

Acessado em 10 de maio 2024

BEAL, A. **Gestão estratégica da informação: como transformar a informação**

**e a tecnologia da informação em fatores de crescimento e alto desempenho nas**

**organizações**. 138 págs. São Paulo: Atlas, 2004.

BECK, K. et al. **O Manifesto para o Desenvolvimento Ágil de Software.** 2001. Disponível em: <https://agilemanifesto.org> Acessado em 15 de maio de 2024

BENYON, D. **Interação humano-computador**. 2ª ed. 442 págs. São Paulo: Pearson Prentice Hall, 2011.

BENTO, E. Junior. **Desenvolvimento web com PHP e MySQL**. 419 págs. São Paulo: Casa do Código, 2016

BRASIL. Ministério do Planejamento, Orçamento e Gestão. Secretaria de Logística e Tecnologia da Informação. **Padrões Web em Governo Eletrônico: Cartilha de Usabilidade**, Brasília: MP, SLTI, 2010. Acessado em 30 de abril de 2024.Disponível em:<https://egov.ufsc.br/portal/sites/default/files/padroes_brasil_e-gov_-_cartilha_de_usabilidade_v12.pdf#Padrões%20Brasil%20e-GOV> Acessado em 28 de abril de 2024

CAPPELLI, E. **Desenvolvimento Híbrido com Flutter: Prós e Contras**. 2018. Disponível em: <https://medium.com/@devmob/desenvolvimento-híbrido-com-flutter-prós-e-contras-6f3f422c480c>. Acesso em 08 de novembro de 2024.

CHIAVENATO, I. **Administração: Teoria, Processo e Práticas**. 406 págs. 3ª ed. São Paulo: Makron Books, 2000. Disponível em:

<https://www.academia.edu/92950485/Administração_Teoria_Processo_e_Prática_Idalberto_Chiavenato> Acessado em 05 de junho de 2024

CIRIBELLI, M.C. **Como elaborar uma dissertação de mestrado através da pesquisa científica**. Rio de Janeiro: 7Letras, 2003.

DALL’OGLIO, P. **PHP: programando com orientação a objetos**. 2.ed. 574 págs. São Paulo, Novatec Editora, 2009.

DART.DEV. **Visão Geral do Dart**. 2024. Disponível em: <https://dart.dev/overview>. Acesso em 05 de novembro de 2024.

ELMASRI, R.; NAVATHE, S. B. **Sistemas De Banco De Dados**. 810 págs. 6. ed. São Paulo: Person Addison Wesley, 2011. Disponível em: <https://archive.org/details/navathe-sistemas-de-banco-de-dados-6a/page/4/mode/2up>. Acesso em 05 de novembro 2024.

FERRARI, F. A. **Crie banco de dados em MySQL**. 128 págs. São Paulo: Digerati Books,2007

FERNANDES, R. P. **O que é Cloud Firestore?** 2017. Disponível em: <https://medium.com/android-dev-moz/firestore-8d225062ffe8>. Acesso em 08 de novembro 2024.

FIREBASE, **Documentação. Firebase Authentication**. 2020. Disponível em: <https://firebase.google.com/docs/auth?hl=pt-br#top_of_page>. Acesso em: 08 novembro 2020.

FIREBASE, **Documentação. Cloud Firestore**. 2020. Disponível em: <https://firebase.google.com/docs/firestore?hl=pt-br>. Acesso em: 08 de novembro 2024.

FLASTCHART, F. **HTML 5: embarque imediato**. 233 págs. Rio de Janeiro: Brasport, 2011

GOOGLE CLOUD. 2024. **O que é MySQL?** Disponível em: https://cloud.google.com/mysql?hl=pt-BR. Acesso em: 21 de novembro de 2024.

GUEDES, G.T. A. **UML 2: uma abordagem prática.** 479 págs.2ª ed. São Paulo: Novatec, 2011.

GUEDES, M. **O que é Dart?** 2019. Disponível em: <https://www.treinaweb.com.br/blog/o-que-e-dart/>. Acesso em: 08 de novembro 2024

HIGHSMITH, J. **O Manifesto Ágil.** 2001. Disponível em: <https://agilemanifesto.org> Acessado em 15 de maio de 2024.

IBM. **O que é desenvolvimento de aplicativos móveis?** 2024. Disponível em: <https://www.ibm.com/br-pt/topics/mobile-application-development>. Acesso em 09 de novembro de 2024

LEMOS II, D. L. **Tecnologia da informação**. 86 págs. 2ª ed. Florianópolis: Instituto Federal de Educação, Ciências e Tecnologia - Santa Catarina, 2011. Disponível em: <https://educapes.capes.gov.br/bitstream/capes/206391/2/CST%20GP%20-%20Tecnologia%20da%20informação%20-%20MIOLO.pdf> Acessado em 05 de junho de 2024

MCFARLANDY. D.S, **CSS3: The Missing Manual**, O'Reilly, 2012.

MEYER. E. A, **Cascading Style Sheets: The Definitive Guide**, O’Reilly, 2000.

MILANI A. **MYSQL: guia do programador**. 336 págs. São Paulo: Novatec Editora, 2006

MOZILLA. **HTML: Linguagem de Marcação de Hipertexto**. 2024. Disponível em:

<https://developer.mozilla.org/pt-BR/docs/Web/HTML> Acesso em 30 de outubro de 2024.

MOZILLA. **Introdução Web**. 2024. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Learn/Getting_started_with_the_web> Acesso em 03 de novembro 2024.

MOZILLA. **JavaScript**. 2024. Disponível em: <https://developer.mozilla.org/ptBR/docs/Web/JavaScrip>. Acesso em 30 de outubro de 2024.

NEGRINO, T; SMITH, D. **JavaScript para World Wide Web**; tradução da [3ª ed. Original] Vanderberg Dantas de Souza, Rio de Janeiro: Campus 2000.

NIELSEN, J; LORANGER, H. (2007). **Usabilidade na Web**. 406 págs. Rio de Janeiro: Elsevier, 2007.

NOLETO, C. **Javascript: o que é, aplicação e como aprender a linguagem JS.** 2024. Disponível em: https://blog.betrybe.com/javascript/. Acesso em 20 de novembro 2024.

OLIVEIRA, F.C.M.B; OLIVEIRA, F.A.M.B. **Interação Humano Computador**. 2ª ed. 93 págs. Fortaleza, Ed. UECE (Editora Universidade Estadual Ceara) 2016. <https://www.uece.br/cct/wp-content/uploads/sites/28/2021/07/Interacao-Homem-Computador.pdf> Acessado em: 08/05/2024

ORACLE. **O que é um banco de dados?** 2024. Disponível em: <https://www.oracle.com/br/database/what-is-database/#mysql>.

Acesso em 05 novembro 2024.

PEREIRA, M. **Flutter: porque você deveria apostar nesta tecnologia**. 2019. Disponível em: <https://medium.com/tableless/flutter-porque-você-deveria-apostar-nesta-tecnologia-94a510fffd18>. Acesso em 08 de novembro de 2024.

PEREIRA, M. **Iniciando com Flutter**. 2019. Disponível em: <https://medium.com/manacespereira/iniciando-com-flutter-aba6158635e2>. Acesso em 08 de novembro de 2024.

PHP.NET. **História do PHP.** Disponível em: <https://www.php.net/manual/pt_BR/history.php.php>.Acesso em: 04 de novembro 2024.

PRESSMAN, S.R; MAXIM, R.B. **Engenharia de Software: uma abordagem profissional**. 9ª ed. Porto alegre. AMGH Editora, 2021. Disponível em <https://ww.academia.edu/89376481/PRESSMAN_Engenharia_de_software_Uma_Abordagem_Profissional_9a_Ed> Acessado em 29 de abril 2024

PRESSMAN, S.R; MAXIM, R.B **Engenharia de Software: uma abordagem profissional**. 8ª ed. 968 págs . Porto alegre. AMGH Editora,2016. Disponível em: <https://archive.org/details/pressman-engenharia-de-software-uma-abordagem-profissional-8a/page/n1/mode/2up> Acessado em 29 de abril 2024

PRESSMA, S.R; **Engenharia de Software: uma abordagem profissional**. 7ª ed. 771 págs. Porto alegre. AMGH Editora,2011

PRODANOYV, C. C.; FREITAS, E. C. **Metodologia do trabalho científico: Métodos e Técnicas da Pesquisa e do Trabalho acadêmico**. 2. ed. — Novo Hamburgo: Feevale, 2013

RAMPAZZO, L. **Metodologia Científica: Para alunos dos cursos de graduação e pós-graduação**. 3. ed. São Paulo: Edições Loyola, 2005.

ROCHA, H.V; BARANAUSKAS, M.C.C, **Design e Avaliação de Interfaces Humano -Computador**; 244 págs., Editora Nied, 2003. Disponível em: <https://www.nied.unicamp.br/biblioteca/design-e-avaliacao-de-interfaces-humano-computador/> Acessado em 14 de maio 2024

ROGERS, Y.; SHARP, H; PREECE, J. **Design da Interação: Além da Interação humano computador**. 3.ed. 585 pags. Porto Alegre: Bookman, 2013

SCHWABER.K, SUTHERLAND.J. **O Guia do Scrum. O Guia Definitivo para o Scrum: As Regras do Jogo.** 2020. 16 págs. Disponível em: <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-PortugueseBR-3.0.pdf> Acessado em 19 de maio 2024

SCRUMM ALLIANCE. **Valores do Scrum**. 2024. Disponível em: <https://www.scrumalliance.org/about-scrum#values> Acessado em 20 de maio 2024

SILVA, M.G da. **Informática – Terminologia: Microsoft Windows Vista, Internet, Segurança, Microsoft Office Word 2007, Microsoft Office Excel 2007, Office Acess 2007, Microsoft Office Power Point 2007**. 3ª ed. São Paulo: Editora Erica, 2011.

SILVA, M. S. **Criando Sitee com HTML: sites de alta qualidade com HTML e CSS.** 427 págs. São Paulo: Novatec Editora, 2008

SOARES, W. **PHP: Programação e Integração com Banco de Dados**. 7.ed. 528 págs. São Paulo, Erica, 2013.

SBROCCO, J.H.T.C, MACEDO, P.C**., Metodologias Ágeis: engenharia de software sob medida**. São Paulo. Editora Érica, 2013

SOMMERVILLE, I. **Engenharia de Software**. 8.ed. 552 págs. São Paulo, Person, 2007.

SOMMERVILLE, I. **Engenharia de Software**. 9.ed. 529 págs. São Paulo, Person, 2011.

VELOSO, R. **Tecnologias da Informação e Comunicação: desafios e perspectivas.** 144 págs. São Paulo: Saraiva, 2011

W3 Schools. **Introdução HTML**. 2024. Disponível em: https://www.w3schools.com/html/html\_intro.asp. Acesso em: 19 de novembro de 2024.

W3TECHS. **Estatísticas de uso de linguagens de programação do lado do servidor para sites.** Disponível em:<https://w3techs.com/technologies/overview/programming_language/>. Acesso em 05 de novembro 2024

.

**APÊNDICE (criados pelo autor)**

**ANEXO (não criados pelo autor)**