INSTITUTO FEDERAL DE RONDÔNIA
CAMPUS PORTO VELHO CALAMA
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

IGOR VINICIUS MENDONÇA BARRETO, LUÍS GONZAGA DA SILVA NETO

EQUINOX FINANCE : UMA SOLUÇÃO FINANCEIRA COM SUPORTE A ORÇAMENTOS COMPARTILHADOS

Porto Velho/RO
2025

IGOR VINICIUS MENDONÇA BARRETO, LUÍS GONZAGA DA SILVA NETO

EQUINOX FINANCE : UMA SOLUÇÃO FINANCEIRA COM SUPORTE A ORÇAMENTOS COMPARTILHADOS

Artigo entregue como Trabalho de Conclusão de Curso ao Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO), *Campus* Porto Velho Calama, como requisito parcial para obtenção do grau de Tecnólogo, junto ao Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Leandro Ferrarezi Valiante

Porto Velho/RO
2025

Ficha catalográfica elaborada pelo Sistema Gerador de Ficha Catalográfica do IFRO.

Neto, Luís Gonzaga da Silva.
Equinox finance: uma solução inteligente para o controle e planejamento financeiro pessoal e empresarial / Luís Gonzaga da Silva Neto, Igor Vinicius Mendonça Barreto. - Porto Velho, 2025.
25 f. : il.

Orientador(a): Prof. Leandro Ferrarezi Valiante.

Trabalho de Conclusão de Curso (Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) – Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO, Porto Velho, 2025.

1. Aplicação Web. 2. Colaboração. 3. Gestão Financeira. 4. Open Finance. I. Barreto, Igor Vinicius Mendonça. II. Valiante, Leandro Ferrarezi (orient.). III. Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO. IV. Título.

Bibliotecário(a) Responsável: Miria Santana Veiga, CRB-11/898

IGOR VINICIUS MENDONÇA BARRETO, LUÍS GONZAGA DA SILVA NETO

EQUINOX FINANCE: UMA SOLUÇÃO FINANCEIRA COM SUPORTE A ORÇAMENTOS COMPARTILHADOS

A banca examinadora, abaixo listada, aprova o Trabalho de Conclusão de Curso “EQUINOX FINANCE” elaborado por “IGOR VINICIUS MENDONÇA BARRETO, LUÍS GONZAGA DA SILVA NETO” como requisito parcial para obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas, pelo Instituto Federal de Educação, Ciência e Tecnologia de Rondônia.

Porto Velho/RO, 27/11/2025

Comissão Examinadora

LEANDRO FERRAREZI VALIANTE:82694095291

Assinado de forma digital por LEANDRO FERRAREZI VALIANTE:82694095291
Dados: 2025.12.01 11:47:29 -04'00'

Prof. Leandro Ferrarezi Valiante - IFRO (Orientador)

gov.br
Documento assinado digitalmente
WESLLEY MICHEL SILVA BOLSONI
Data: 02/12/2025 19:07:50-0300
Verifique em https://validar.iti.gov.br

Prof. Weslley Michel S. Bolsoni - IFRO

ICP Brasil
Documento assinado digitalmente
WILLIANS DE PAULA PEREIRA
Data: 02/12/2025 18:42:24-0300
Verifique em https://validar.iti.gov.br

Prof. Willians de Paula Pereira - IFRO

EQUINOX FINANCE

RESUMO: O presente trabalho de pesquisa aborda a lacuna de ferramentas PFM *(Personal Financial Management)* focadas em orçamentos colaborativos, em um contexto de alta desorganização financeira no Brasil. O objetivo central foi o desenvolvimento e a implementação de um Sistema Web para Gestão Financeira com foco na segurança e na análise transacional coletiva. Utilizou-se a metodologia de Pesquisa de Design em uma arquitetura Client-Server e stack moderno (Node.js/NestJS/PostgreSQL). Os principais resultados incluem o módulo de Workspaces Colaborativos, serviços de Importação via Open Finance e a geração de relatórios analíticos em tempo real. A segurança foi garantida pela criptografia do JWT (AES-256-GCM) e o *hashing* de senhas (Argon2). O Estudo de Caso Confirmatório validou a viabilidade da solução, provando a eficácia do modelo colaborativo como um mecanismo robusto para mitigar a desorganização financeira.

PALAVRAS-CHAVE: Aplicação Web. Colaboração. Gestão Financeira. Open Finance.

ABSTRACT: This research addresses the gap in PFM (*Personal Financial Management*) tools focused on collaborative budgeting, within the context of high financial disorganization in Brazil. The central objective was the development and implementation of a Web System for Financial Management focused on security and collective transactional analysis. The Design Research methodology was employed in the construction of the artifact, utilizing a *Client-Server* architecture and a modern stack (Node.js/NestJS/Next.js/PostgreSQL). Key results include the Collaborative Workspaces module, Massive Import services via Open Finance, and the generation of real-time analytical reports. Security was ensured through JWT encryption (AES-256-GCM) for confidentiality and password hashing using Argon2. The Confirmatory Case Study validated the viability of the solution, proving the effectiveness of the collaborative model as a robust mechanism to mitigate financial disorganization.

KEYWORDS: Collaboration. Financial Management. Open Finance. Web Application.

1 INTRODUÇÃO

A gestão financeira pessoal e coletiva é um desafio crescente em todo o mundo. Estudos mostram que a falta de controle sobre gastos e receitas é uma das principais causas de endividamento e instabilidade financeira (Silva; Almeida; Carvalho Godinho, 2022).

No contexto global, uma pesquisa da S&P Global e Global Financial Literacy Excellence Center (2015) revelou que apenas 33% da população adulta mundial é financeiramente letrada. Isso significa que a maioria da população não compreende conceitos básicos de finanças, como juros e inflação, o que as torna vulneráveis a decisões financeiras ruins.

No Brasil, essa situação é especialmente crítica. Dados recentes do “Mapa de Inadimplência”, da SERASA (2025), mostram que o número de inadimplentes no país é alarmante. Em Julho de 2025, cerca de 78,2 milhões de brasileiros estavam em situação de inadimplência, tendo os brasileiros com idades entre 41 e 60 anos representando a maior fatia da população com nome restrito, com 35,3%. Na sequência estão as faixas etárias de 26 a 40 anos (34%), acima de 60 anos (19,2%) e os jovens entre 18 e 25 anos (11,5%).

Pesquisas indicam que muitos aplicativos PFM (Gestão Financeira Pessoal, do inglês *Personal Financial Management*) oferecem suporte limitado para o orçamento em comparação com o rastreamento de despesas, não explorando totalmente conceitos como a Contabilidade Mental, que é a tendência de indivíduos atribuírem regras e valores diferentes ao dinheiro com base em suas origens ou propósitos, resultando em decisões financeiras irracionais (Alenazi; Sas, 2023). Essa limitação é crítica, especialmente porque a gestão familiar exige transparência e um planejamento conjunto, necessitando de ferramentas que promovam o monitoramento e o controle do orçamento compartilhado para ambos os parceiros (Santos; Cruz, 2023).

Embora o mercado apresente uma oferta crescente de aplicações de Gestão Financeira Pessoal, o que é evidenciado pela expansão do ecossistema FinTech na América Latina, que cresceu 340% entre 2017 e 2023 (Banco Interamericano de Desenvolvimento, 2024), a vasta maioria dessas soluções, incluindo “Mobills” e “Organizze”, está orientada a funcionalidades elementares e ao controle financeiro estritamente individualizado. Consequentemente, essas ferramentas falham em fornecer recursos que promovam o controle financeiro de forma verdadeiramente colaborativa e abrangente, pois tratam a gestão de orçamentos compartilhados entre famílias ou grupos de forma secundária. Desta forma, a dificuldade de visualizar a saúde financeira de maneira clara e consolidada em um contexto coletivo representa uma lacuna significativa que as soluções atuais não preenchem adequadamente.

Na Figura 1 a seguir, é demonstrado um comparativo de funcionalidades entre diferentes aplicações existentes no mercado:

Figura 1 – Comparativo dos principais apps para controle financeiro pessoal

Recursos	Mobills	Minhas Economias	Organizze	Orçamento Fácil
Controle de despesas	✅	✅	✅	✅
Controle de saldos	✅	✅	✅	✅
Gerenciamento de cartão de crédito	✅	✅	✅	✅
Planejamento financeiro	✅	✅	✅	✅
Objetivos	✅	✅	❌	❌
Investimentos	✅	❌	❌	✅
Integração com Bancos	✅	✅	❌	✅
Gamification	✅	❌	❌	❌
Conteúdos de educação	✅	✅	✅	❌
Projeção de saldos	✅	✅	✅	❌
Disponível offline	✅	✅	✅	❌
Acesso pela Web	✅	✅	✅	✅

Fonte: Terceiro (2025)

Por outro lado, além de aplicativos de PFM, também existem as planilhas eletrônicas, que apesar de serem extremamente úteis para usuários proficientes, exigem um nível de conhecimento e dedicação que a maioria das pessoas não possui. Esta dificuldade é evidenciada pela pesquisa sobre alfabetização digital e financeira da TELETIME (2023), que aponta que apenas 11,31% da população economicamente ativa do Brasil possui habilidades digitais intermediárias, proficiência na qual é exigida a capacidade de utilizar uma fórmula aritmética básica em planilhas. Portanto, a ineficiência, neste contexto, reside na falta de praticidade para o usuário médio, e não na ferramenta em si, reforçando a necessidade de uma solução digital com curva de aprendizado mais baixa e automatização de processos.

O desenvolvimento deste projeto justifica-se pela proposta de oferecer uma ferramenta mais completa e adaptável à realidade de orçamentos compartilhados, um aspecto que a maioria dos aplicativos existentes, como o “Mobills” e o “Organizze”, não contempla de maneira abrangente.

Com base nesse cenário, este trabalho visa desenvolver e implementar um sistema web para gestão financeira que capacite o usuário a organizar, analisar e visualizar suas finanças de forma individual e coletiva.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos, princípios e tecnologias que sustentam o desenvolvimento do presente trabalho. São discutidos os modelos de arquitetura, padrões de comunicação e as principais tecnologias selecionadas, estabelecendo o alicerce conceitual necessário para compreender o funcionamento e a estrutura da solução proposta.

2.1 Fundamentos de Sistemas e Arquitetura

O desenvolvimento do sistema de gestão financeira adotou o paradigma de Arquitetura Cliente-Servidor, um modelo arquitetural essencial para sistemas distribuídos e a base da *World Wide Web* (Fielding; Taylor, 2002). Nesta estrutura, as responsabilidades são claramente divididas: o cliente, também conhecido como *client-side* solicita os serviços, e o servidor, conhecido como *server-side*, processa a lógica de negócio e gerencia a persistência de dados. Essa separação promove o desacoplamento entre as camadas e garante a escalabilidade e a manutenibilidade do sistema.

A comunicação entre estas camadas é realizada por meio de uma API (Interface de Programação de Aplicações, do inglês *Application Programming Interface*), seguindo o estilo arquitetural REST (*Representational State Transfer*, ou Transferência de Estado Representacional). O REST, originalmente definido por Roy Fielding em sua tese doutoral, estabelece um conjunto de restrições que, quando respeitadas (resultando em uma “API RESTful”), conferem aos serviços web características de alta interoperabilidade e eficiência (Fielding, 2000). Os princípios centrais da arquitetura REST aplicada ao projeto incluem:

- **Separação Cliente-Servidor:** O que garante que as preocupações de interface e armazenamento de dados evoluam de forma independente.
- **Sem Estado (Stateless):** Cada requisição HTTP do cliente deve ser auto-suficiente. Esta restrição, facilitada pela utilização do protocolo HTTP, garante a simplicidade e a alta escalabilidade horizontal do servidor (APPMASTER, 2023).
- **Interface Uniforme:** O sistema utiliza os métodos padrão do protocolo HTTP (GET, POST, PUT, DELETE) para manipular os recursos do sistema (transações, categorias, *workspaces*) de forma previsível, simplificando a interação entre as partes (IBM, 2025).

2.2 Tecnologias Fundamentais do Projeto

O **JavaScript (JS)**, criado em 1995 por Brendan Eich, é uma linguagem de programação de alto nível, interpretada e multiparadigma, essencial para a interatividade na web (Flanagan, 2020). A linguagem se tornou um pilar universal, sendo regida pelas especificações **ECMAScript (ES)**. O ES atua como o padrão normativo que define as regras e a sintaxe do JS, garantindo a **interoperabilidade** entre os diferentes ambientes de execução e navegadores (ECMA International, 2024).

O **Node.js** é a plataforma de execução *server-side* baseada no motor *"V8"* da Google. Ele introduziu um modelo de processamento **não bloqueante**, orientado a eventos (*event-driven*), crucial para a escalabilidade de aplicações *web* (Node.js Foundation, 2025). Sobre esta plataforma, o *backend* foi desenvolvido utilizando o **NestJS**, um *framework* Node.js versátil e progressivo. Inspirado no Angular, o NestJS promove uma **arquitetura modularizada**, testável e de alta manutenibilidade, essencial para a construção de aplicações robustas no lado do servidor Ingila (2025).

A persistência dos dados financeiros exige um **Sistema Gerenciador de Banco de Dados (SGBD)**, *software* essencial para a criação e manutenção de bancos de dados (Elmasri; Navathe, 2011). Por esta razão, optou-se pelo **PostgreSQL**, um SGBD objeto-relacional de código aberto, reconhecido por sua robustez, confiabilidade e vasta adoção em ambientes empresariais (Amit Phaujdar, 2023). O PostgreSQL é ideal para dados sensíveis, oferecendo alta compatibilidade com o padrão SQL, suporte a diversos tipos de dados e capacidade de extensibilidade (POSTGRESQL GLOBAL DEVELOPMENT GROUP, 2025).

Antes de apresentar o Next.js, é importante destacar que a camada de frontend do Equinox Finance fundamenta-se no **React**, uma biblioteca JavaScript criada para construção de interfaces baseadas em componentes reutilizáveis. O modelo declarativo e a abordagem de componentização adotados pelo React refletem princípios consolidados de engenharia de software, especialmente aqueles relacionados à modularidade e separação de responsabilidades (Gamma *et al.*, 1995). Além disso, o uso eficiente do Virtual DOM e a adoção de arquiteturas Single-Page Application (SPA) ou Aplicação de Página Única, contribuem significativamente para o desempenho e a manutenibilidade das interfaces modernas (Scott Jr, 2015).

Enquanto o React fornece a base componentizada para a construção da interface, o **Next.js** foi adotado como *framework* estrutural para entregar uma solução mais robusta e performática. Sua escolha justifica-se pela capacidade de superar as limitações inerentes às abordagens puramente *Client-Side Rendering* (CSR) do React tradicional, especialmente no que tange à renderização no lado do servidor (*Server-Side Rendering* - SSR). (Hanafi; Haq; Agustin, 2024). Através de sua arquitetura baseada no

moderno “App Router”, o Next.js facilita a criação de interfaces complexas ao permitir uma separação clara de responsabilidades: utiliza *Server Components* para processamento pesado e busca de dados no servidor — reduzindo a sobrecarga no dispositivo do cliente (Pati; Zaki, 2025) — e *Client Components* para gerenciar a interatividade, alinhando-se perfeitamente à arquitetura Cliente-Servidor adotada e oferecendo recursos nativos de roteamento e otimização (Hanafi; Haq; Agustin, 2024; Pati; Zaki, 2025).

Para garantir a consistência e a portabilidade da arquitetura Client-Server em diferentes ambientes de implantação, é fundamental compreender o Docker, que é o motor de execução que viabiliza a tecnologia de contêineres, um método de virtualização no nível do sistema operacional que empacota código e todas as suas dependências em uma unidade leve e portátil (Merkel, 2014). Ademais, o Docker Compose é a ferramenta complementar essencial que permite a orquestração e o gerenciamento de aplicações multi-contêineres definidas em um único arquivo de configuração (YAML) (Docker, Inc., 2025).

3 METODOLOGIA

Este capítulo apresenta a metodologia utilizada no desenvolvimento do presente artigo, descrevendo tanto o fluxo de trabalho adotado quanto a classificação da pesquisa. São expostos os procedimentos que orientaram a construção do sistema, as etapas executadas e o enquadramento metodológico que fundamenta o projeto. A seguir, detalham-se a classificação da metodologia e o passo a passo das atividades realizadas.

3.1 Classificação Metodológica

Com base na natureza do projeto de desenvolvimento de uma aplicação web de gestão financeira, a metodologia deste trabalho adota uma abordagem mista. Quanto à natureza, a pesquisa é inicialmente secundária, utilizada para fundamentar as escolhas conceituais e tecnológicas, culminando na pesquisa primária pela construção do sistema proposto.

Quanto aos objetivos, a presente pesquisa é classificada como “Pesquisa de Design”, pois visa criar um novo artefato tecnológico para resolver o problema prático da gestão financeira colaborativa.

Por fim, quanto aos procedimentos técnicos, este trabalho emprega a “Pesquisa Bibliográfica” para o embasamento teórico e culmina em um “Estudo de Caso Confirmatório” (ou Prova de Conceito), onde o sistema desenvolvido demonstra a viabilidade e a eficácia das arquiteturas e tecnologias selecionadas para a solução proposta. As

classificações metodológicas utilizadas, em relação à natureza, objetivos e procedimentos técnicos, seguem as diretrizes estabelecidas por Wazlawick (2021).

3.2 Passo-a-passo

1. Inicialmente, foi realizada uma revisão da literatura (Pesquisa Secundária), que constitui o passo fundamental e prévio para qualquer trabalho científico (Wazlawick, 2021). Esta etapa estabeleceu o contexto/domínio do projeto e consolidou a fundamentação teórica das tecnologias e conceitos essenciais, como a arquitetura RESTful.

2. Foi feita uma análise de mercado e das funcionalidades de concorrentes para identificar a lacuna e definir os requisitos funcionais e não-funcionais do sistema. A definição rigorosa dos requisitos é o alicerce para qualquer projeto de software (Sommerville, 2019).

3. Foi construído o Modelo de Dados (conceitual e lógico) para representar as entidades do sistema (Usuário, Transação, Workspace). A elaboração de modelos de design detalhados, como o Documento de Arquitetura (DA), é essencial para o desenvolvimento de software de qualidade e a rastreabilidade dos componentes (Pressman, 2010).

4. Foi estabelecido a estrutura de Controle de Versão (Git) e a hospedagem no GitHub. Optou-se pela criação de um repositório principal que gerencia os repositórios do *frontend* e do *backend* como **submodules**. Essa organização modular é vital para a gestão independente das dependências e o versionamento isolado de cada aplicação (Pressman, 2010).

5. O desenvolvimento da lógica de negócio e da API RESTful foi iniciado com o *framework* NestJS. Esta fase focou na implementação dos módulos de autenticação e na criação dos *endpoints* RESTful, seguindo uma abordagem iterativa e incremental típica de metodologias ágeis (Schwaber; Sutherland, 2020). Garantiu-se que o servidor cumprisse as restrições arquiteturais e a integridade transacional dos dados no PostgreSQL.

6. Para estabelecer um ambiente de execução e testes rigoroso e repetível, o sistema foi “containerizado” utilizando as ferramentas Docker e Docker Compose. A capacidade de replicar o ambiente de desenvolvimento e testes é um requisito fundamental do método científico para validar os resultados empíricos do artefato (Wazlawick, 2021).

7. Concomitantemente, foi desenvolvido o *frontend* utilizando Next.js, com foco na experiência do usuário (UX), abrangendo aspectos como usabilidade, acessibilidade e clareza visual, bem como no design da Interface (UI). Foram implementadas as interfaces para a interação com os *endpoints* da API, sendo as funcionalidades liberadas em pequenos ciclos de desenvolvimento e testes, conforme preconizado por práticas ágeis (Schwaber; Sutherland, 2020).

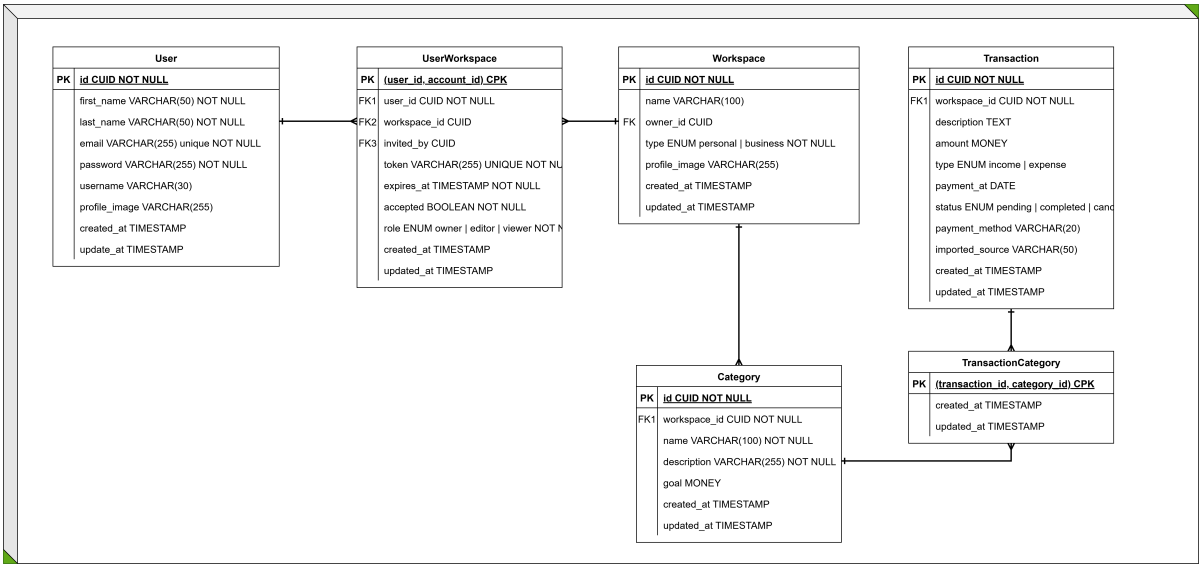
4 DESENVOLVIMENTO

Este capítulo descreve o processo de construção do sistema, detalhando a implementação técnica das soluções propostas. O desenvolvimento foi dividido em duas camadas principais: o *backend* (servidor e banco de dados) e o *frontend* (interface do usuário), que se comunicam através de uma arquitetura RESTful. As seções a seguir detalham a implementação da camada de *frontend*.

4.1 *Backend*

Antes de entrarmos diretamente no contexto de código do servidor, é importante definirmos a estrutura do banco de dados. Na Figura 2 é demonstrado a primeira versão do Modelo Lógico Relacional.

Figura 2 – Modelo Lógico Relacional



Fonte: Autoria própria.

A partir disso, foi construído o Esquema Físico do banco de dados. Para essa finalidade foi utilizado o ORM *(Object-Relational Mapping) Prisma.js*, que tem como função intermediar a comunicação entre o JavaScript e o banco de dados.

Tendo claro a estrutura do banco de dados podemos partir para os módulos do servidor.

4.1.1 Autenticação

O sistema de autenticação utiliza o *JSON Web Token (JWT)* para delegar a validação da identidade à assinatura criptográfica do token. A segurança é reforçada por camadas específicas: a **Confidencialidade** é garantida pela criptografia do payload utilizando o algoritmo **AES-256-GCM** (*Advanced Encryption Standard*), superando a vulnerabilidade padrão de legibilidade do JWT. Para a **Proteção de Credenciais**, o *hashing* das senhas é realizado pelo **Argon2**. Por fim, a **Segurança de Acesso** é complementada pela Verificação em Duas Etapas (2FA) para acessos de máquinas desconhecidas.

4.1.2 Permissões

O módulo de autenticação é complementado por um serviço dedicado à Autorização, que estabelece as ações que um usuário autenticado pode executar dentro de um *workspace* específico. Essa lógica é implementada através do modelo de **Controle de Acesso Baseado em Papéis (*Role-Based Access Control* - RBAC)**.

A Figura 3 ilustra a estrutura de configuração das permissões, que mapeia cada papel (ex: *Viewer*, *Editor*) a um conjunto granular de privilégios (ex: criar transação, excluir categoria). O Serviço de Permissões é ativado em cada requisição crítica, sua função primária é verificar, em tempo de execução, se o usuário solicitante possui a permissão necessária.

4.1.3 *Workspaces*

O módulo de Gestão de Workspaces é o componente mais crítico do sistema, pois estabelece a estrutura de dados relacional que interliga e segrega todas as informações financeiras.

A funcionalidade de **Compartilhamento de Workspace** é implementada através de um *endpoint* que orquestra a comunicação e a persistência do convite. Para iniciar o processo, são requeridos o identificador (ID) do usuário proprietário do workspace, o ID do workspace a ser compartilhado, o e-mail do usuário destinatário e o seu papel neste workspace.

Com base nestes dados, o sistema executa o seguinte processo de validação:

Figura 3 – Configuração de Permissões

```
export const RoleAccess: Record<
    ModuleType,
    Record<OperationType, RoleType[]>
> = {
    [ModuleType.WORKSPACE]: {
        [OperationType.DELETE]: ['OWNER'],
        [OperationType.CREATE]: ['OWNER', 'EDITOR', 'VIEWER'],
        [OperationType.UPDATE]: ['OWNER'],
        [OperationType.READ]: ['OWNER', 'EDITOR', 'VIEWER'],
    },
    [ModuleType.TRANSACTION]: {
        [OperationType.DELETE]: ['OWNER'],
        [OperationType.CREATE]: ['OWNER', 'EDITOR'],
        [OperationType.UPDATE]: ['OWNER', 'EDITOR'],
        [OperationType.READ]: ['OWNER', 'EDITOR', 'VIEWER'],
    },
```

Fonte: Autoria própria.

1. **Identificação e Validação de Destinatário:** É efetuada uma busca no banco de dados para validar a existência do usuário de destino através do e-mail fornecido.
2. **Validação de Auto-Convite:** É verificada a condição de que o ID do usuário proprietário e o ID do usuário de destino não sejam iguais.
3. **Controle de Duplicidade e Prevenção de Spam:** É executada uma consulta para determinar se já há um convite ativo ou enviado nos últimos minutos, mitigando o envio abusivo.

Se todas as regras de validação forem satisfeitas, o convite é registrado na base dados e, em seguida, uma notificação via e-mail é disparada ao usuário destinatário. O processo de aceitação do convite é iniciado quando o usuário destinatário acessa a URI de aceitação. Ao clicar no link, o endpoint de aceitação no servidor é acionado, executando as seguintes etapas de validação:

1. **Recuperação do Convite:** O registro do convite é recuperado da base de dados através do token fornecido na URL.
2. **Validação de Expiração:** É efetuada a validação do prazo de expiração, que é rigorosamente definido para uma (1) hora a partir do momento do envio, garantindo a segurança e o controle de acesso temporário.
3. **Atualização de Status:** Uma vez que todas as condições de validade são satisfeitas, o registro correspondente na base dados tem o valor de seu atributo "*accepted*" alterado para verdadeiro.

Assim que solidificado o aceite do convite, ao requisitar a lista de workspaces, o servidor executa uma consulta que abrange dois critérios de acesso:

1. **Workspaces de Propriedade:** São recuperados todos os workspaces onde o usuário figura como o proprietário principal (dono).
2. **Workspaces de Membro Convidado:** É realizada uma busca complementar para identificar todos os workspaces nos quais o usuário foi convidado e aceitou o convite (ou seja, onde o atributo "*accepted*" é verdadeiro).

Essa lógica de busca bidimensional assegura que o usuário tenha acesso completo e imediato a todos os ambientes colaborativos e individuais, promovendo a clareza e a transparência na gestão de suas finanças.

O sistema também implementa dois mecanismos essenciais para a flexibilidade e a governança de dados:

1. **Transferência de Conteúdo:** Funcionalidade para migrar dados internos (transações e categorias) entre *workspaces*. O sistema realiza uma operação transacional atômica para reassociar registros, manipulando a integridade referencial dos dados.
2. **Transferência de Propriedade:** Permite a migração do *workspace* para outro usuário. O processo replica o fluxo de convite e aceite, com regras de validação e expiração. Essa abordagem garante o consentimento explícito do novo proprietário, sendo fundamental para a não-repúdio, segurança e auditoria em sistemas de dados sensíveis.

4.1.4 Serviço de E-mail

O módulo de e-mail atua centralizando a gestão de *templates* e a governança de segurança para convites. O serviço orquestra o envio de convites de *workspace* por meio da validação rigorosa de regras de negócio. Isso inclui a configuração de um **Tempo de Vida (*TTL*)** para o *token* criptográfico e uma **Regra de Anti-Spam** para controlar o reenvio.

4.1.5 Categorias

O módulo de Gestão de Categorias embora conceitualmente simples, sua importância é fundamental, pois estabelece a estrutura para a classificação e organização de todas as entradas e saídas financeiras do sistema.

A importância deste módulo é maximizada em duas áreas:

1. **Análise e Relatórios:** As categorias servem como critério de agrupamento de dados. Nos relatórios, elas permitem ao usuário agrupar conjuntos de transações por uma temática específica, possibilitando uma análise detalhada dos fluxos de gastos e receitas.
2. **Integridade de Dados:** A categorização coerente é crucial para que os relatórios financeiros sejam informativos e reflitam com precisão a saúde financeira do workspace.

4.1.6 Transações

A Transação constitui o dado fundamental e indivisível dentro da aplicação, representando o registro contábil de toda movimentação financeira. Sua estrutura robusta é projetada para capturar o contexto completo do evento, sendo primariamente classificada pelo seu tipo (entrada ou saída), descrição e valor monetário. O atributo *status* é crucial para o controle de fluxo de caixa, permitindo a gestão de transações pendentes, canceladas e finalizadas. Adicionalmente, o campo *payment_at* registra a temporalidade do evento. Por fim, a classificação é garantida pela propriedade *categories_id*, que permite a associação a múltiplas categorias, transformando o registro em informação para a inteligência analítica do sistema.

O módulo de Transações é complementado por serviços dedicados à importação massiva de registros, o que aumenta a usabilidade e a velocidade de alimentação de dados na aplicação.

O primeiro serviço aceita arquivos *CSV* em formato padronizado (campos predefinidos) para processamento automático. Para lidar com variações externas de dados (como extratos bancários), o serviço suporta o Mapeamento Dinâmico. Neste método, o *frontend* envia um objeto **De/Para** que instrui o servidor a normalizar os campos do *CSV* para a estrutura de transação.

O segundo mecanismo de importação utiliza a **Pluggy**, um serviço especializado que integra a aplicação ao ecossistema *Open Finance*. A migração dos dados ocorre de maneira segura: o servidor recebe os dados transacionais da Pluggy, mediante o consentimento explícito do usuário, processa-os e os normaliza para o formato interno da entidade Transação.

4.1.7 Relatórios

O módulo de Relatórios atua como a camada de *Business Intelligence* da aplicação, com a função primária de processar dados transacionais e de categorias para gerar *insights* que auxiliem a **tomada de decisão financeira**. O sistema oferece três relatórios analíticos essenciais: **Resumo por Período** (cálculo de saldo em intervalos definidos); **Resumo por Categoria** (análise granular de gastos por agrupamento); e **Resumo por Status** (visão operacional do fluxo de caixa ao contabilizar transações *PENDING*, *COMPLETED* e *CANCELLED*).

4.2 *Frontend*

A interface do usuário foi desenvolvida utilizando o *framework* Next.js. Esta escolha arquitetural permitiu a implementação de um modelo híbrido de renderização, fundamental para o desempenho da aplicação. Diferentemente de aplicações de página única (SPA) tradicionais, o Equinox Finance utiliza componentes de servidor para processar a busca de dados e a renderização inicial, enviando ao navegador apenas o necessário para interatividade

Essa abordagem trouxe benefícios diretos à segurança, mantendo a lógica sensível e os tokens de autenticação no lado do servidor, e à performance, otimizando o tempo de carregamento inicial percebido pelo usuário. A separação de responsabilidades entre cliente e servidor garante uma interface reativa sem sacrificar a eficiência.

4.2.1 Interface Visual e Experiência do Usuário

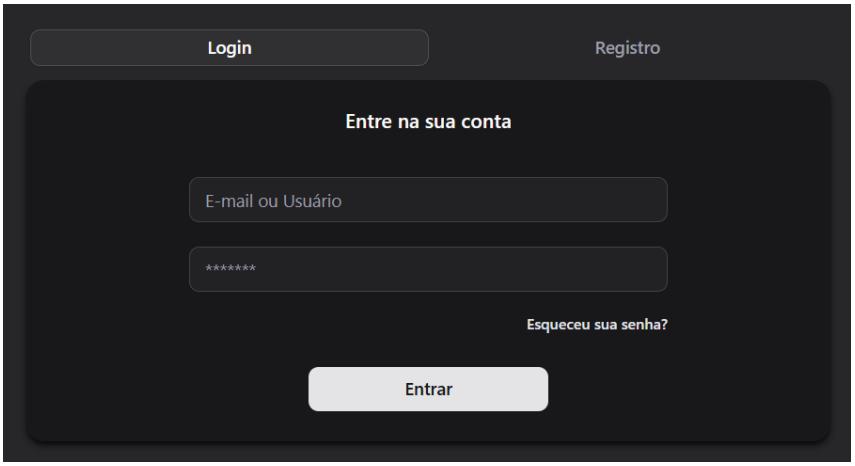
Para materializar a arquitetura proposta, a interface visual foi construída com foco na responsividade e na padronização. O design adotado utiliza componentes acessíveis e personalizáveis, assegurando que a aplicação seja navegável por teclado e leitores de tela. Além da estética, a camada de interface implementa rigorosos esquemas de validação em todos os formulários, garantindo que dados inconsistentes sejam tratados antes mesmo do envio ao servidor.

4.2.2 Autenticação

O primeiro módulo implementado foi o sistema de autenticação, essencial para a segurança e personalização da experiência do usuário. Para otimizar a experiência do usuário e reduzir o tempo de navegação, optou-se por unificar as interfaces de Login e Registro em uma única rota, conforme visualizado nas figuras 4 e 5. Essa estratégia

de renderização condicional elimina a latência de navegação entre páginas distintas, proporcionando uma transição imediata e mantendo o contexto visual do usuário.

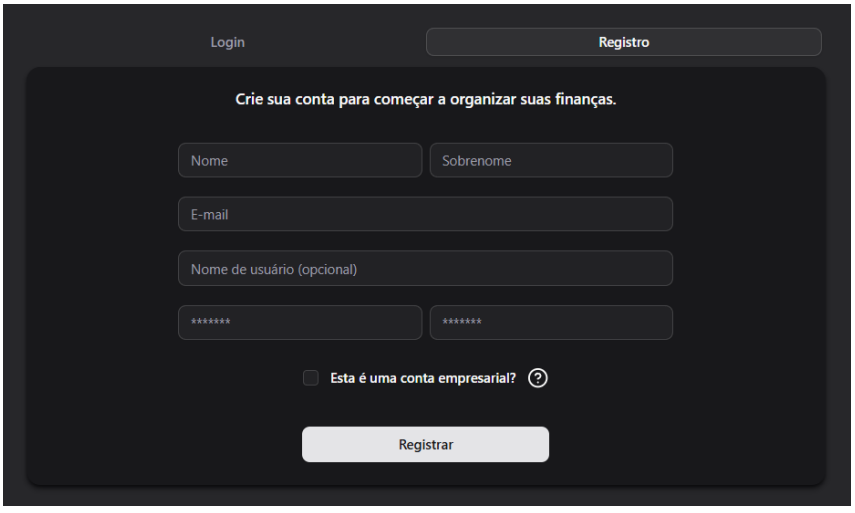
Figura 4 – Formulário de Login



Fonte: Autoria própria.

Visualmente, a interface utiliza componentes de abas com botões de alternância, garantindo clareza sobre qual ação está sendo realizada.

Figura 5 – Formulário de Registro

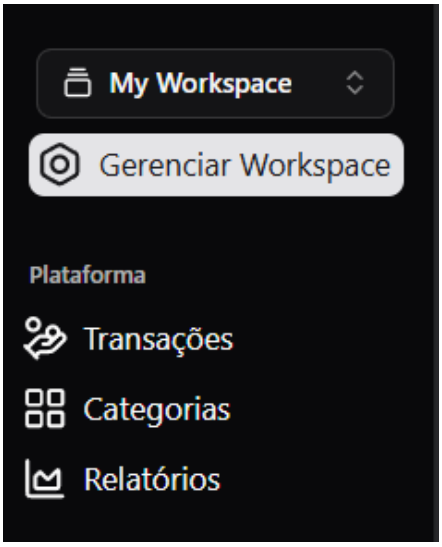


Fonte: Autoria própria.

4.2.3 *Dashboard* e Navegação

O *dashboard* do sistema foi concedido como um ponto central de navegação, a interface é estruturada em torno de um menu lateral (*sidebar*), demonstrado na figura 6, que garante o acesso rápido aos módulos da aplicação e permite a gestão de múltiplos *workspaces*.

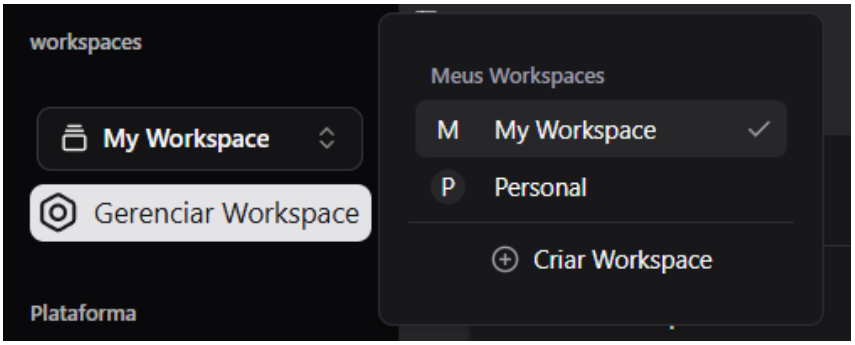
Figura 6 – Menu Lateral



Fonte: Autoria própria.

Um componente fundamental nesta estrutura de *sidebar* é o "Seletor de *Workspaces*", destacado na figura 7, que permite ao usuário criar (figura 8) e alternar instantaneamente entre diferentes contextos financeiros (como pessoal ou empresarial). Ao alterar o *workspace* ativo, a aplicação atualiza automaticamente o escopo de dados exibidos em todas as telas subsequentes.

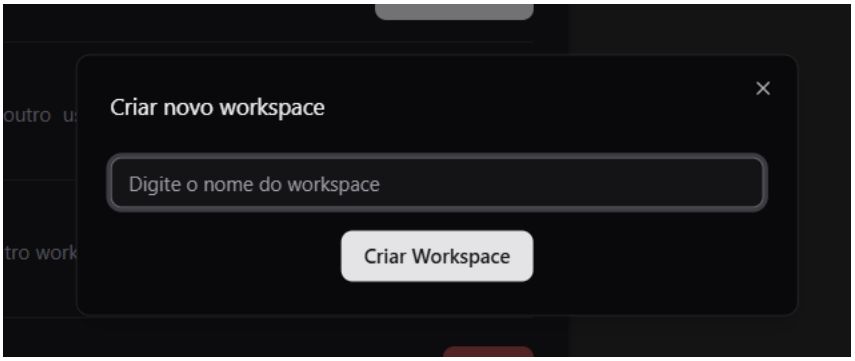
Figura 7 – Seletor de *Workspaces*



Fonte: Autoria própria.

Logo abaixo do seletor, o menu de navegação organiza o acesso às páginas funcionais do sistema. Esta estrutura permite ao usuário gerenciar as configurações do espaço de trabalho atual, incluindo a administração de membros e propriedades do *workspace*, além de fornecer acesso direto às tabelas de gestão financeira e de categorias personalizadas. Por fim, o menu disponibiliza a navegação para o painel de relatórios, centralizando a análise visual dos dados.

Figura 8 – Modal para Criação de Workspace



Fonte: Autoria própria.

4.2.4 Página de Transações

A página de transações centraliza o controle financeiro, permitindo a visualização e manipulação de receitas e despesas. A interface foi otimizada para lidar com grandes volumes de dados através de paginação no lado do servidor, o que reduz o consumo de memória no dispositivo do usuário ao carregar registros sob demanda.

Além das operações manuais de cadastro e edição, o sistema integra um recurso de automação via *Open Finance*. Esta funcionalidade permite que o usuário conecte suas contas bancárias de forma segura, autorizando a leitura de dados sem compartilhar credenciais diretamente com a aplicação. Após a conexão, o sistema sincroniza e categoriza as movimentações bancárias automaticamente, eliminando a necessidade de inserção manual.

4.2.5 Página de Categorias

Para garantir a organização eficiente dos registros, a página de categorias permite a criação de classificações personalizadas. A interface deste módulo segue estritamente os padrões visuais e de usabilidade estabelecidos na página de transações, garantindo consistência na navegação e reduzindo a curva de aprendizado do usuário ao interagir com diferentes partes do sistema.

4.2.6 Página de Relatórios

O módulo de Relatórios transforma os dados transacionais em visualizações estratégicas para a tomada de decisão. A apresentação dos dados foi segmentada em três perspectivas complementares:

- **Relatório por Períodos:**Demonstra a evolução financeira histórica. Através de gráficos de área, as curvas de receitas e despesas são sobrepostas, permitindo a identificação visual imediata de períodos de *superávit* ou *déficit*.
- **Relatórios por Categoria:** Permite identificar o destino dos recursos. Utiliza gráficos de barras horizontais para acomodar rótulos extensos, facilitando a leitura e a comparação entre diferentes grupos de gastos
- **Relatório por Status:** Foca na saúde operacional do fluxo de caixa. Diferente dos gráficos, esta visualização utiliza cartões de sumário com indicadores semânticos para destacar valores pendentes, concluídos ou cancelados, permitindo o monitoramento rápido de pendências financeiras.

5 CONCLUSÃO

O presente Trabalho de Conclusão de Curso atingiu seu objetivo central ao desenvolver e implementar um sistema web para gestão financeira focado na problemática da desorganização e no gerenciamento de orçamentos colaborativos. A pesquisa foi motivada pela alta taxa de endividamento no Brasil e pela identificação de uma lacuna no mercado de PFM *(Personal Financial Management)*, onde a maioria das soluções é voltada para o uso estritamente individual. A metodologia de Pesquisa de Design utilizada culminou na entrega de um artefato funcional que valida a viabilidade e a relevância social da solução proposta.

O sistema desenvolvido atingiu os objetivos propostos ao entregar uma plataforma unificada, segura e intuitiva. A arquitetura Cliente-Servidor adotada, fundamentada no uso do NestJS para o *backend* e do Next.js para o *frontend*, provou-se robusta e escalável. A implementação de uma “API RESTful” garantiu a integridade e a segurança dos dados, enquanto a interface web híbrida (SSR/CSR) proporcionou uma experiência de usuário fluida e performática, validando as escolhas tecnológicas discutidas na fundamentação teórica.

A principal contribuição reside na implementação do mecanismo de *Workspaces*, que permite a segregação e o compartilhamento seguro de finanças entre múltiplos usuários, na automação via *Open Finance* e na geração de relatórios analíticos.

Contudo, o sistema foi focado em Gestão Financeira Pessoal (PFM) e Colaborativa, excluindo funcionalidades de contabilidade gerencial complexas (como depreciação ou gestão de estoques) que seriam necessárias para o uso estritamente empresarial.

Como caminhos para a continuidade, propõe-se a expansão para Inteligência Analítica com *Machine Learning* (ML/IA) para previsões orçamentárias preditivas e o

Desenvolvimento Móvel Nativo para otimizar a experiência do usuário e a captura de transações em tempo real.

REFERÊNCIAS

ALENAZI, Mariam Zaid; SAS, Corina. Evaluating Budgeting Apps: Limited Support for Budgeting Compared to Tracking. **ResearchGate**, 2023. Disponível em: https://www.researchgate.net/publication/377242109_Evaluating_Budgeting_Apps_Limited_Support_for_Budgeting_Compared_to_Tracking.

AMIT PHAUJDAR. **PostgreSQL vs MySQL: Explore suas 12 Diferenças Críticas**. 2023. Publicado em: 2023. Disponível em: Kinsta Blog. Disponível em: https://kinsta.com/pt/blog/postgresql-vs-mysql/.

APPMASTER. **Princípios de Design REST**. 2023. Publicado em: 2023. Disponível em: AppMaster Blog. Disponível em: https://appmaster.io/pt/blog/resto-dos-principios-de-design.

BANCO INTERAMERICANO DE DESENVOLVIMENTO. **Estudo: Ecossistema fintech na América Latina e no Caribe supera 3.000 startups**. 2024. Disponível em: https://www.iadb.org/pt-br/noticias/estudo-ecossistema-fintech-na-america-latina-e-no-caribe-supera-3000-startups.

DOCKER, INC. **Overview of Docker Compose**. Documentação oficial. 2025. Disponível em: https://docs.docker.com/compose/overview/.

ECMA INTERNATIONAL. **ECMAScript: The Standard for JavaScript**. Padrão mantido pelo Comitê Técnico TC39. 2024. Disponível em: https://www.ecma-international.org/technical-committees/tc39/.

ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de Banco de Dados**. 6. ed. São Paulo: Pearson Education do Brasil, 2011. Disponível em: https://www.kufunda.net/publicdocs/Sistemas%20de%20Banco%20de%20Dados%20(Ramez%20Elmasri,%20Shamkant%20B.%20Navathe).pdf.

FIELDING, Roy T. **Architectural Styles and the Design of Network-based Software Architectures**. 2000. Tese de Doutorado – University of California, Irvine, Irvine, CA. Disponível em: https://fenix.tecnico.ulisboa.pt/downloadFile/1689468335707305/S02-A03-03%20-fielding_dissertation-ch5.pdf.

FIELDING, Roy T.; TAYLOR, Richard N. Principled Design of the Modern Web Architecture. **ACM Transactions on Internet Technology**, ACM, v. 2, n. 2, p. 115–150, 2002. Disponível em: https://ics.uci.edu/~taylor/documents/2002-REST-TOIT.pdf.

FLANAGAN, David. **JavaScript: The Definitive Guide, Canada**. O'Reilly Media, Inc, 2020.

GAMMA, Erich *et al.* **Design patterns: elements of reusable object-oriented software**. Pearson Deutschland GmbH, 1995.

HANAFI, Roy; HAQ, Abd; AGUSTIN, Ninik. Comparison of web page rendering methods based on Next. js framework using page loading time test. **Teknika**, v. 13, n. 1, p. 102–108, 2024.

IBM. **O que é PostgreSQL?** 2025. Disponível em: IBM Think Topics. Disponível em: https://www.ibm.com/br-pt/think/topics/postgresql.

INGILA. NestJS: A Progressive Node.js Framework for Building Robust Server-Side Applications. **Stackademic (Medium)**, fev 2025. Disponível em: https://blog.stackademic.com/nestjs-a-progressive-node-js-framework-for-building-robust-server-side-applications-e7b3b93b6fd3.

MERKEL, Dirk. Docker: lightweight Linux containers for consistent development and deployment. **Linux Journal**, 2014. Disponível em: https://www.linuxjournal.com/content/docker-lightweight-linux-containers-consistent-development-and-deployment.

NODE.JS FOUNDATION. **About Node.js**. Página oficial de descrição do projeto. 2025. Disponível em: https://nodejs.org/en/about.

PATI, Swostik; ZAKI, Yasir. Evaluating the Efficacy of Next. js: A Comparative Analysis with React. js on Performance, SEO, and Global Network Equity. *In:* COMPANION Proceedings of the ACM on Web Conference 2025. 2025. p. 1239–1243.

POSTGRESQL GLOBAL DEVELOPMENT GROUP. **About PostgreSQL**. 2025. Disponível em: PostgreSQL Official Website. Disponível em: https://www.postgresql.org/about/.

PRESSMAN, Roger S. **Engenharia de Software: Uma Abordagem Profissional**. 7. ed. São Paulo: McGraw-Hill, 2010.

S&P GLOBAL; GLOBAL FINANCIAL LITERACY EXCELLENCE CENTER. **S&P Global FinLit Survey: Financial Literacy Around the World**. 2015. Relatório de pesquisa. Data da Publicação: 18 de nov. de 2015. Disponível em: https://gflec.org/wp-content/uploads/2015/11/Finlit_paper_16_F2_singles.pdf.

SANTOS, João Alves dos; CRUZ, Regiane Silva Ribeiro da. Gestão Financeira Familiar: aplicativo compartilhado para planejamento, monitoramento e controle do orçamento. **Revista de Informática Aplicada**, 2023. Disponível em: https://ric.cps.sp.gov.br/bitstream/123456789/19812/1/informaticanegocios_2023_2_joaoalvesdossantos_gestaofinanceirafamiliaraplicativocompartilhadop.pdf.

SCHWABER, Ken; SUTHERLAND, Jeff. **The Scrum Guide: Guia Definitivo para Scrum**. 2020. Disponível em: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-PortugueseBR-3.0.pdf.

SCOTT JR, Emmit A. **SPA Design and Architecture: Understanding single-page web applications**. Simon e Schuster, 2015.

SERASA. **Mapa da inadimplência e renegociação de dívidas no Brasil**. 2025. Disponível em: Serasa Blog. (Acesso: 27 out. 2025). Disponível em: `https://www.serasa.com.br/limpa-nome-online/blog/mapa-da-inadimplencia-e-renogociacao-de-dividas-no-brasil/`.

SILVA, Allan Gabriel Cunha; ALMEIDA, Vinicius Cesar Oliveira; CARVALHO GODINHO, Luiz Antônio de. Gestão financeira: um estudo sobre finanças pessoais como ferramenta para evitar o endividamento. **LIBERTAS: Revista de Ciênciais Sociais Aplicadas**, v. 12, n. 2, 2022.

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

TELETIME. **Brasil tem índice de habilidades digitais abaixo da média na América Latina**. 2023. Publicado em: 26 de abr. de 2023. Disponível em: `https://teletime.com.br/26/04/2023/brasil-tem-indice-de-habilidades-digitais-abaixo-da-media-na-america-latina/`.

TERCEIRO, Carlos. **Apps de controle financeiro**. Acesso em: 17 nov. 2025. 2025. Disponível em: `https://www.mobills.com.br/blog/aplicativos/apps-de-controle-financeiro/`.

WAZLAWICK, Raul Sidnei. **Metodologia de Pesquisa para Ciência da Computação**. 3. ed. Rio de Janeiro: Elsevier Brasil, 2021. Disponível em: `https://tsxvpsbr.dyndns.org/arquivos/UFFS/Metodologia%20De%20Pesquisa%20CienciaDaComputacao%20_%20TCC1.pdf`.