INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ

CAMPUS PICOS

CURSO ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

GABRIEL DE SOUSA FILHO

HUBCASH: SISTEMA WEB PARA CONTROLE E PLANEJAMENTO FINANCEIRO PESSOAL COM ÊNFASE EM USABILIDADE E APOIO À TOMADA DE DECISÃO

PICOS

2026

GABRIEL DE SOUSA FILHO

HUBCASH: SISTEMA WEB PARA CONTROLE E PLANEJAMENTO FINANCEIRO PESSOAL COM ÊNFASE EM USABILIDADE E APOIO À TOMADA DE DECISÃO

Relatório Técnico apresentado como exigência para aprovação na disciplina Trabalho de Conclusão de Curso - Graduação do Curso de Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Picos.

Orientador(a): Prof. Esp. Jorge Luis da Rocha Lima

PICOS

2026

GABRIEL DE SOUSA FILHO

HUBCASH: SISTEMA WEB PARA CONTROLE E PLANEJAMENTO FINANCEIRO PESSOAL COM ÊNFASE EM USABILIDADE E APOIO À TOMADA DE DECISÃO

Projeto de pesquisa apresentado como exigência para aprovação na disciplina Trabalho de Conclusão de Curso - Graduação do Curso de Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Picos.

Aprovada em: ___/___/_____.

BANCA EXAMINADORA

_____

Prof. Esp. Jorge Luis da Rocha Lima (Orientador)

Instituto Federal do Piauí (IFPI)

_____

Prof. Me. Ciro Ferreira de Carvalho Junior

Instituto Federal do Piauí (IFPI)

_____

Prof. Esp. Denis Costa Paiva

Instituto Federal do Piauí (IFPI)

AGRADECIMENTOS

Agradeço, primeiramente, a Deus, por me conceder saúde, força, sabedoria e perseverança durante toda esta caminhada. Nos momentos de dificuldade, encontrei fé e coragem para superar os desafios, seguir em frente e concluir mais esta importante etapa da minha formação acadêmica. Cada conquista alcançada ao longo desse percurso é fruto das oportunidades e aprendizados que recebi.

À minha família, expresso minha mais sincera gratidão pelo amor, incentivo e apoio incondicional. A confiança depositada em mim, as palavras de encorajamento e a compreensão nos momentos em que precisei dedicar grande parte do meu tempo aos estudos foram fundamentais para que eu permanecesse firme em busca dos meus objetivos. Compartilho esta conquista com todos aqueles que sempre acreditaram no meu potencial.

Agradeço também aos amigos e colegas que fizeram parte dessa trajetória, tornando essa jornada mais leve e enriquecedora. As experiências compartilhadas, a troca de conhecimentos, o companheirismo e o apoio mútuo foram essenciais para enfrentar os desafios da graduação e contribuíram significativamente para meu crescimento acadêmico e pessoal.

Por fim, registro minha gratidão ao meu orientador, pela orientação, dedicação e pelos valiosos conhecimentos compartilhados ao longo do desenvolvimento deste trabalho. Agradeço, ainda, a todos os professores e às demais pessoas que contribuíram para minha formação. A todos, meu sincero muito obrigado.

RESUMO

Este trabalho apresenta o desenvolvimento do HubCash, um sistema web voltado ao controle e planejamento financeiro pessoal, com foco em usabilidade e apoio à tomada de decisão. A plataforma permite o registro e o acompanhamento de receitas e despesas, a visualização de saldos mensais, a organização das transações por categorias e a geração de relatórios em PDF. O sistema foi desenvolvido utilizando NestJS, Prisma ORM e PostgreSQL no back-end, e React, TypeScript e Vite no front-end, com implantação em Render, Vercel e Supabase. Após a implementação, foram realizados testes das principais funcionalidades, confirmando o correto funcionamento da aplicação. Como resultado, o HubCash apresenta-se como uma solução prática e intuitiva, contribuindo para a organização das finanças pessoais e para uma gestão financeira mais consciente.

Palavras-chave: Finanças pessoais; Sistema web; Controle financeiro; Planejamento financeiro; Usabilidade.

ABSTRACT

This paper presents the development of HubCash, a web-based system designed for personal financial control and planning, with a focus on usability and decision-making support. The platform enables the recording and tracking of income and expenses, the viewing of monthly balances, the organization of transactions by category, and the generation of PDF reports. The system was developed using NestJS, Prisma ORM, and PostgreSQL for the back-end, and React, TypeScript, and Vite for the front-end, with deployment on Render, Vercel, and Supabase. Following implementation, tests of the core features were conducted, confirming the application's proper functioning. Consequently, HubCash stands out as a practical and intuitive solution, contributing to organized personal finances and more conscious financial management.

Keywords: Personal finance; Web system; Financial control; Financial planning; Usability.

LISTA DE TABELAS

Tabela 1 – Requisitos Funcionais (RF) 14

Tabela 2 – Requisitos Não Funcionais (RNF) 15

Tabela 3 – Avaliação de Usabilidade do HubCash 34

LISTA DE ILUSTRAÇÕES

Figura 1 – Diagrama de Casos de Uso 16
Figura 2 – Diagrama de Entidades-Relacionamento 17
Figura 3 – Representação Gráfica da Arquitetura do HubCash 18
Figura 4 – Tela de Login da aplicação (Versão Desktop). 19
Figura 5 – Tela de Cadastro de Usuário (Versão Desktop). 20
Figura 6 – Painel Principal / Dashboard (Versão Desktop). 21
Figura 7 – Modal de Cadastro e Edição de Transações (Versão Desktop). 22
Figura 8 – Tela de Perfil do Usuário (Versão Desktop). 23
Figura 9 – Tela de Recuperação de Senha (Versão Desktop). 24
Figura 10 – Tela de Login da aplicação (Versão Mobile). 25
Figura 11 – Tela de Cadastro de Usuário (Versão Mobile). 26
Figura 12 – Painel Principal / Dashboard (Versão Mobile). 27
Figura 13 – Modal de Cadastro e Edição de Transações (Versão Mobile). 28
Figura 14 – Tela de Perfil do Usuário (Versão Mobile). 29
Figura 15 – Tela de Recuperação de Senha (Versão Mobile). 30

LISTA DE ABREVIATURAS E SIGLAS

API – Application Programming Interface

BRL – Real brasileiro

CDN – Content Delivery Network

CI/CD – Continuous Integration / Continuous Deployment

CLI – Command-Line Interface

CNC – Confederação Nacional do Comércio de Bens, Serviços e Turismo

DBaaS – Database as a Service

DER – Diagrama de Entidades-Relacionamento

DOM – Document Object Model

DTO – Data Transfer Object

E/S – Entrada/Saída (Input/Output)

Esp. – Especialista

HMR – Hot Module Replacement

HTTP – HyperText Transfer Protocol

HTTPS – HyperText Transfer Protocol Secure

ID – Identificador

IDE – Ambiente de Desenvolvimento Integrado

IFPI – Instituto Federal de Educação, Ciência e Tecnologia do Piauí

JSON – JavaScript Object Notation

JWT – JSON Web Token

Me. – Mestre

ORM – Object-Relational Mapping

PDF – Portable Document Format

Peic – Pesquisa de Endividamento e Inadimplência do Consumidor

Prof. – Professor

REST – Representational State Transfer

RF – Requisito Funcional

RNF – Requisito Não Funcional

SGBD – Sistema Gerenciador de Banco de Dados

SPA – Single Page Application

SQL – Structured Query Language

SSL/TLS – Secure Sockets Layer / Transport Layer Security

TCLE - Termo de Consentimento Livre e Esclarecido

TICs – Tecnologias da Informação e Comunicação

UC – Caso de Uso

URL – Uniform Resource Locator

VS Code – Visual Studio Code

LISTA DE SÍMBOLOS

: Proporção / Relacionamento de Cardinalidade

SUMÁRIO

1	**INTRODUÇÃO**	9
2	**OBJETIVOS**	10
3	**TECNOLOGIAS RELACIONADAS**	11
4	**MODELAGEM**	13
5	**SOFTWARE** ……..	31
6	**AVALIAÇÃO E RESULTADOS**	34
7	**CONSIDERAÇÕES FINAIS**	36
8	**REFERÊNCIAS**	37

APÊNDICE A – QUESTIONÁRIO DE AVALIAÇÃO 38

1 INTRODUÇÃO

A organização financeira pessoal constitui um fator essencial para a estabilidade econômica e para a melhoria da qualidade de vida, uma vez que possibilita o controle dos recursos disponíveis, o planejamento de despesas e a definição de metas financeiras de curto e longo prazo (Housel, 2021). Segundo Gitman (2010), o planejamento financeiro é fundamental para a gestão eficiente dos recursos. Entretanto, na prática, o elevado índice de endividamento das famílias brasileiras demonstra a necessidade de maior conscientização e controle financeiro (CNC, 2024).

Paralelamente, o avanço das Tecnologias da Informação e Comunicação (TICs) transformou a forma como as pessoas realizam diversas atividades do cotidiano, incluindo o gerenciamento das finanças pessoais. *Sistemas web* passaram a oferecer recursos para registro e acompanhamento de receitas e despesas, substituindo métodos tradicionais, como planilhas eletrônicas e anotações em papel (Gomes, 2023). Apesar disso, muitas soluções disponíveis apresentam interfaces complexas, excesso de funcionalidades ou dificuldades de utilização, fatores que comprometem a experiência do usuário e reduzem a continuidade do uso, especialmente por pessoas com pouca familiaridade com ferramentas tecnológicas.

Nesse contexto, torna-se relevante o desenvolvimento de sistemas que conciliem funcionalidades de controle financeiro com princípios de usabilidade, proporcionando uma interação simples, intuitiva e eficiente. Conforme destaca Elisiário (2025), aplicações desenvolvidas com foco na experiência do usuário favorecem a organização das informações financeiras, reduzem a complexidade das tarefas e contribuem para uma tomada de decisão mais consciente, permitindo que o usuário acompanhe sua situação financeira de maneira clara e organizada.

Diante desse cenário, este trabalho apresenta o desenvolvimento do HubCash, um sistema web voltado ao controle e ao planejamento financeiro pessoal. A aplicação permite o cadastro e gerenciamento de receitas e despesas, a categorização das movimentações financeiras, o acompanhamento do saldo mensal e a geração de relatórios em PDF, oferecendo uma interface intuitiva e acessível. O sistema foi desenvolvido utilizando uma arquitetura baseada em Application Programming Interface *(*API) Representational State Transfer *(*REST*)*, buscando fornecer uma solução prática, segura e eficiente para auxiliar os usuários na organização de suas finanças e no apoio à tomada de decisões financeiras.

2 OBJETIVOS

2.1 OBJETIVO GERAL

Desenvolver o HubCash, um sistema web voltado ao controle e ao planejamento das finanças pessoais, utilizando a Tecnologia da Informação como ferramenta de apoio à organização financeira individual, com ênfase em usabilidade e no apoio à tomada de decisão, de forma simples, acessível e intuitiva.

2.2 OBJETIVOS ESPECÍFICOS

1. Levantar e mapear os requisitos funcionais e não funcionais necessários para a construção de uma plataforma intuitiva de controle financeiro pessoal;
2. Projetar a arquitetura do sistema e a modelagem de dados utilizando o padrão cliente-servidor, diagramas de casos de uso e modelo entidade-relacionamento;
3. Desenvolver a API REST (Back-end) em *NestJS* e *TypeScript*, integrando a persistência de dados no banco PostgreSQL (Supabase) via *Object-Relational Mapping* (ORM) Prisma;
4. Construir a interface gráfica (Front-end) responsiva e dinâmica em *React* com *Vite*, focando em componentes interativos e navegação simplificada;
5. Implementar mecanismos de segurança para controle de acesso, incluindo autenticação por tokens *JSON Web Token* (JWT) e criptografia de senhas com o algoritmo *Bcrypt*;
6. Avaliar a usabilidade e a aceitação do sistema mediante a aplicação de testes práticos e questionário de avaliação com uma amostra de 10 (dez) usuários finais, analisando os resultados de satisfação e facilidade de uso.

3 TECNOLOGIAS RELACIONADAS

Para o desenvolvimento do sistema HubCash, foi adotada uma arquitetura desacoplada baseada no modelo cliente-servidor (*Client-Server*). Essa abordagem separa a lógica de negócios e o acesso aos dados (*Back-end*) da interface de usuário (*Front-end*), permitindo independência de desenvolvimento, facilitando a manutenção e garantindo maior escalabilidade à aplicação.

Como Ambiente de Desenvolvimento Integrado (IDE), utilizou-se o *Visual Studio Code* (VS Code), um editor de código-fonte leve e extensível desenvolvido pela *Microsoft*. O VS Code proporcionou suporte nativo e integração avançada com a linguagem *TypeScript*, integração com o Git para controle de versão, além do uso de extensões para padronização de código (*ESLint* e *Prettier*) e suporte ao ecossistema do *Prisma ORM*.

A seguir, são detalhadas as tecnologias, linguagens, *frameworks* e ferramentas utilizadas em cada camada do sistema.

3.1 BACK-END

A camada de *Back-end* do HubCash foi projetada como uma API REST, responsável por processar as regras de negócio, realizar a validação de dados, gerenciar o acesso ao banco de dados e garantir a segurança das informações trafegadas.

- **Node.js e TypeScript:** O ambiente de execução *Node.js* foi escolhido por sua alta eficiência no processamento de requisições assíncronas E/S (*Input/Output*). A linguagem TypeScript foi adotada em substituição ao *JavaScript* tradicional por adicionar tipagem estática e recursos avançados de orientação a objetos, reduzindo erros em tempo de compilação e aumentando a manutenção do código.
- **NestJS Framework:** Para a estruturação do código, utilizou-se o *framework NestJS*, desenvolvido em TypeScript. O *NestJS* fornece uma arquitetura modularizada fortemente inspirada em padrões como Injeção de Dependência (*Dependency Injection*) e Controle Invertido (*Inversion of Control*). A aplicação foi organizada em módulos distintos (*AuthModule*, *UsersModule*, *TransactionsModule*), separando claramente as responsabilidades entre *Controllers* (manipulação de requisições HTTP), *Services* (regras de negócio) e *DTOs* (*Data Transfer Objects*, para validação de entrada de dados).

- **Prisma ORM e PostgreSQL:** O armazenamento persistente dos dados utiliza o *PostgreSQL*, um Sistema Gerenciador de Banco de Dados Relacional (SGBD) de alto desempenho e confiabilidade. A comunicação entre o NestJS e o PostgreSQL foi intermediada pelo Prisma ORM (*Object-Relational Mapping*), que permite a definição do esquema do banco via arquivo declarativo (schema.prisma), além de automatizar migrações (*migrations*) e fornecer consultas estaticamente tipadas.
- **Segurança e Autenticação (Bcrypt, JWT e Guards):** A proteção dos dados dos usuários foi estruturada em duas frentes principais:
    1. *Criptografia de Senhas:* Utilização da biblioteca *Bcrypt* para aplicar algoritmos de *hash* salgado às senhas antes do armazenamento no banco de dados.
    2. *Autenticação e Autorização:* Implementação do protocolo JWT. Após o login bem-sucedido, a *API* gera um token assinado com tempo de expiração. A proteção de rotas privadas é feita por meio de *Guards* e *Strategies* do *NestJS* (com Passport), exigindo o envio do token no cabeçalho (*Header Authorization*) de cada requisição.

3.2 FRONT-END

A camada de *Front-end* constitui a interface gráfica com a qual o usuário interage. Foi projetada com foco em responsividade, clareza visual, baixo tempo de resposta e facilidade de uso.

- **React com Vite:** A interface foi desenvolvida utilizando a biblioteca React, fundamentada no conceito de componentes reutilizáveis e na atualização eficiente do *Document Object Model* (*DOM*). Para a criação e empacotamento da aplicação (*build tool*), utilizou-se o Vite, que oferece tempos de inicialização e recarregamento (*Hot Module Replacement* - HMR) extremamente rápidos durante o desenvolvimento.

- **Gerenciamento de Estado e Contextos (Context API):** Para gerenciar os estados globais da aplicação sem a necessidade de propagação excessiva de propriedades (*prop drilling*), utilizou-se a *Context* API nativa do React. Foram criados contextos específicos, como o *AuthContext* (responsável por manter o estado da sessão do usuário e o token JWT) e o *ToastContext* (para disparar notificações visuais de sucesso

ou erro).

- **Consumo de API (Axios):** A integração com o *Back-end* é realizada por meio da biblioteca *Axios*, centralizada em um serviço de API (api.ts). O *Axios* foi configurado para interceptar requisições e anexar automaticamente o token de autenticação JWT armazenado no cliente.

- **Interface e Exportação:** A interface conta com bibliotecas utilitárias para estilização modular, suporte a ícones funcionais, componentes para a visualização das finanças por categoria, além de módulos para formatação de moeda local Real brasileiro (BRL) e geração do arquivo de relatório financeiro mensal em formato PDF.

4 MODELAGEM

Neste capítulo, apresenta-se a modelagem técnica do sistema HubCash, detalhando a especificação dos requisitos funcionais e não funcionais, o mapeamento dos casos de uso, a modelagem conceitual e lógica do banco de dados, a arquitetura do software e o detalhamento das interfaces desenvolvidas.

4.1 LEVANTAMENTO DE REQUISITOS

O levantamento de requisitos define as funcionalidades e as restrições operacionais e técnicas do sistema. Os requisitos foram classificados em Requisitos Funcionais (RF), que descrevem os comportamentos esperados da aplicação, e Requisitos Não Funcionais (RNF), que definem as qualidades, padrões de segurança e premissas arquitetônicas (SOMMERVILLE, 2019).

Para a elicitação dos requisitos do HubCash, realizou-se uma análise de domínio fundamentada na revisão da literatura sobre gestão financeira pessoal e no *benchmarking* de ferramentas existentes. A partir da identificação das principais limitações dessas soluções tradicionais — como a alta carga cognitiva e a burocracia de uso —, mapearam-se as necessidades essenciais do público-alvo. Com base nesse estudo, foram definidos os fluxos prioritários da aplicação (autenticação segura, controle de movimentações, análise em *dashboard* e exportação de relatórios em PDF).

4.1.1 Requisitos Funcionais

A Tabela 1 apresenta a relação dos 21 Requisitos Funcionais identificados e implementados no HubCash.

Tabela 1 — Requisitos Funcionais (RF)

ID	Descrição do Requisito
RF01	O sistema deve permitir que o usuário se cadastre informando nome, e-mail e senha.
RF02	O sistema deve criptografar a senha do usuário antes de armazená-la no banco de dados.
RF03	O sistema deve permitir que o usuário faça login informando e-mail e senha.
RF04	O sistema deve gerar um token JSON Web Token (JWT) após a autenticação bem-sucedida.
RF05	O sistema deve proteger rotas privadas exigindo um token JWT válido no cabeçalho das requisições.
RF06	O sistema deve permitir que o usuário visualize seus dados de perfil.
RF07	O sistema deve permitir que o usuário edite seu nome, e-mail e senha.
RF08	O sistema deve permitir que o usuário exclua sua conta.
RF09	O sistema deve permitir que o usuário registre uma transação financeira informando título, valor, tipo, categoria e data.
RF10	O sistema deve classificar as transações em dois tipos: Entrada (INCOME) e Saída (EXPENSE).
RF11	O sistema deve oferecer categorias pré-definidas organizadas por tipo de transação.
RF12	O sistema deve permitir que o usuário crie categorias personalizadas.
RF13	O sistema deve permitir que o usuário edite uma transação existente.
RF14	O sistema deve permitir que o usuário exclua uma transação mediante confirmação.
RF15	O sistema deve listar as transações correspondentes ao mês selecionado.

RF16	O sistema deve exibir o resumo financeiro mensal composto por: total de entradas, total de saídas e saldo líquido.
RF17	O sistema deve permitir a navegação entre meses no dashboard.
RF18	O sistema deve permitir filtrar transações por tipo (entrada/saída) e por categoria.
RF19	O sistema deve gerar um relatório em formato PDF contendo o resumo financeiro e a listagem de transações do mês.
RF20	O sistema deve exibir notificações visuais (*toasts*) de sucesso e erro após as ações do usuário.
RF21	O sistema deve permitir que o usuário solicite a redefinição de senha informando o e-mail cadastrado.

Fonte: Elaborado pelo autor (2026).

4.1.2 Requisitos Não Funcionais

A Tabela 2 descreve os 6 Requisitos Não Funcionais que norteiam a qualidade técnica, segurança e disponibilidade da plataforma.

Tabela 2 — Requisitos Não Funcionais (RNF)

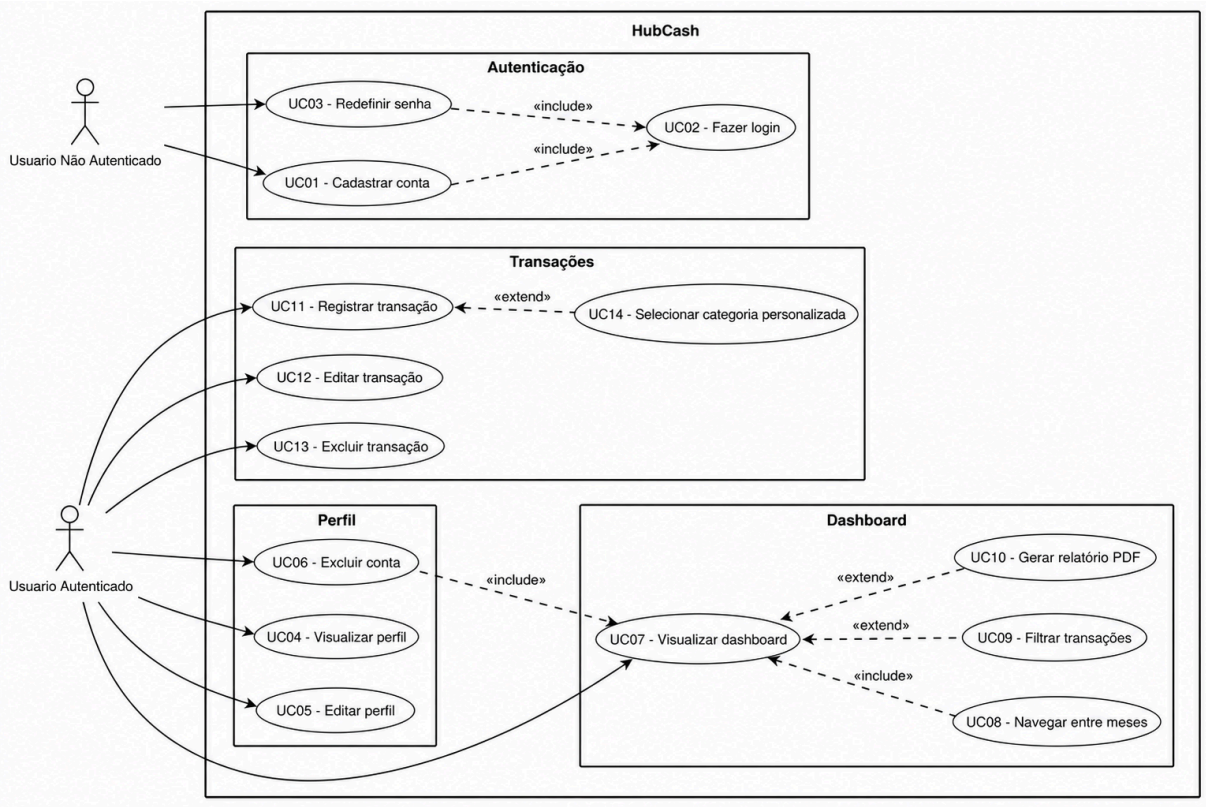
ID	Descrição do Requisito
RNF01	**Responsividade:** O sistema deve possuir interface adaptável para telas desktop e dispositivos móveis.
RNF02	**Segurança de Dados:** As senhas dos usuários devem ser armazenadas utilizando algoritmo de *hash* salgado com Bcrypt.
RNF03	**Sessão e Autenticação:** A autenticação deve utilizar tokens JWT com tempo de expiração definido em 1 dia.
RNF04	**Validação de Dados:** O sistema deve validar rigorosamente os dados de entrada no *Back-end* (via DTOs) antes de efetuar qualquer processamento ou persistência.
RNF05	**Disponibilidade:** O sistema deve estar hospedado e acessível publicamente via internet em infraestrutura em nuvem.
RNF06	**Arquitetura Desacoplada:** O *Front-end* e o *Back-end* devem constituir aplicações totalmente separadas, comunicando-se via API REST.

Fonte: Elaborado pelo autor (2026).

4.2 DIAGRAMA DE CASOS DE USO

A partir do mapeamento dos Requisitos Funcionais, foram identificados os Casos de Uso (UC) da aplicação. A Figura 1 apresenta o Diagrama de Casos de Uso do sistema HubCash, mapeando as interações dos atores com a plataforma.

Figura 1 - Diagrama de Casos de Uso



Fonte: Elaborado pelo autor (2026).

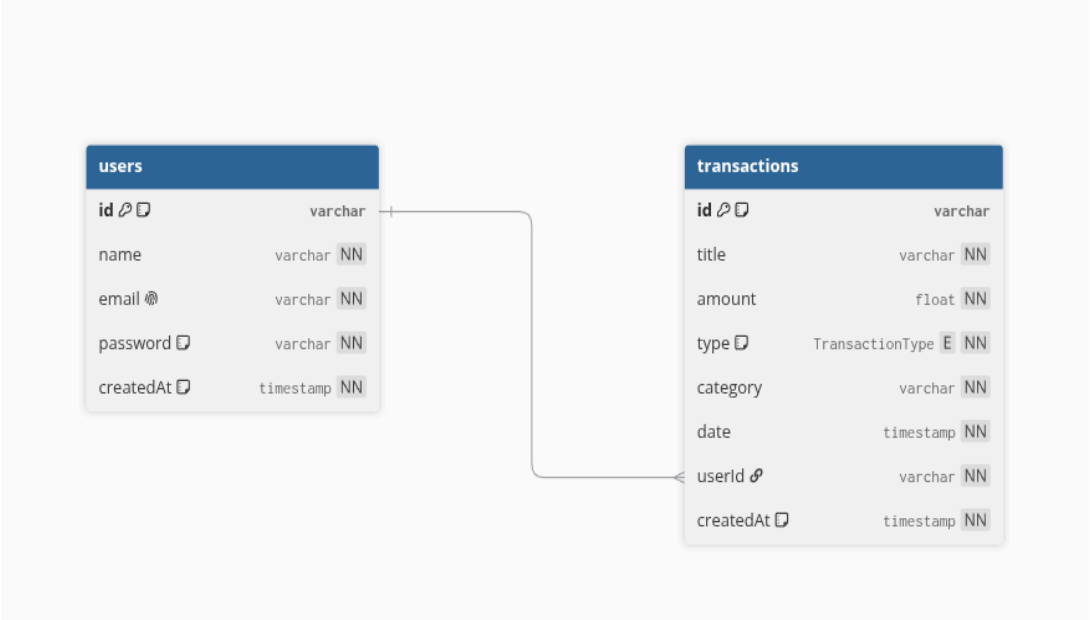
4.3 DIAGRAMA DE ENTIDADES-RELACIONAMENTO (DER)

A camada de persistência de dados do HubCash foi modelada sobre o paradigma relacional e implementada no banco de dados *PostgreSQL*, sendo gerenciada pelo Prisma ORM. A modelagem prioriza a integridade referencial, a eficiência nas consultas por usuário e a simplicidade conceitual.

4.3.1 Estrutura das Entidades e Atributos

A camada de persistência de dados do HubCash foi modelada sobre o paradigma relacional. A estrutura lógica do banco de dados está representada na Figura 2 através do Diagrama de Entidades-Relacionamento (DER).

Figura 2 - Diagrama de Entidades-Relacionamento



Fonte: Elaborado pelo autor (2026).

4.3.2 Cardinalidade e Relacionamento

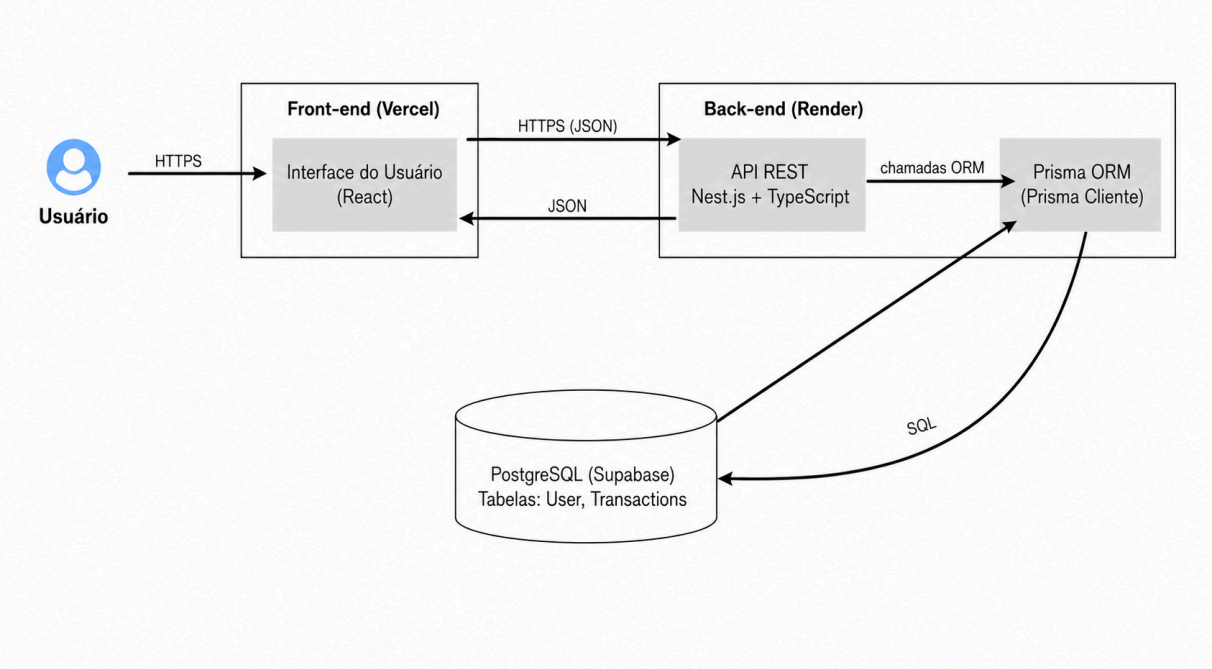
- Relacionamento Usuário vs. Transações ($1 : N$):
    a. Um usuário (User) pode registrar zero ou N transações (Transaction).
    b. Cada transação (Transaction) pertence obrigatoriamente a um único usuário (User), ligação mantida pelo campo userId.

4.4 ARQUITETURA DO SISTEMA

Para atender aos requisitos de usabilidade, desempenho e facilidade de manutenção, o sistema HubCash foi projetado utilizando o padrão de arquitetura desacoplada no modelo cliente-servidor. Essa abordagem permite a separação clara de responsabilidades entre a interface com o usuário (*Front-end*), a lógica de negócios e *APIs* (*Back-end*) e a persistência dos dados (*Banco de Dados*).

A Figura 3 ilustra a representação gráfica da arquitetura de alto nível do HubCash, mapeando os principais componentes da solução e o fluxo de comunicação entre as camadas.

Figura 3 – Representação gráfica da arquitetura do HubCash



Fonte: Elaborado pelo autor (2026)

4.4.1 Divisão das Camadas

Conforme apresentado na Figura 3, a infraestrutura e os componentes do sistema organizam-se nas seguintes camadas:

1. **Usuário e Comunicação (HTTPS/JSON):** O acesso do usuário à plataforma ocorre por meio de navegadores web via protocolo seguro HTTPS. A troca de informações entre a interface gráfica e a *API* é realizada por meio de requisições e respostas em formato padronizado *JSON* (*JavaScript Object Notation*).

2. **Camada de Front-end (React SPA):** Desenvolvida como uma *Single Page Application* (SPA) utilizando React, a interface do usuário é responsável pela renderização dos componentes visuais, gerenciamento de estado da sessão, apresentação dos gráficos financeiros e captura das interações do usuário.

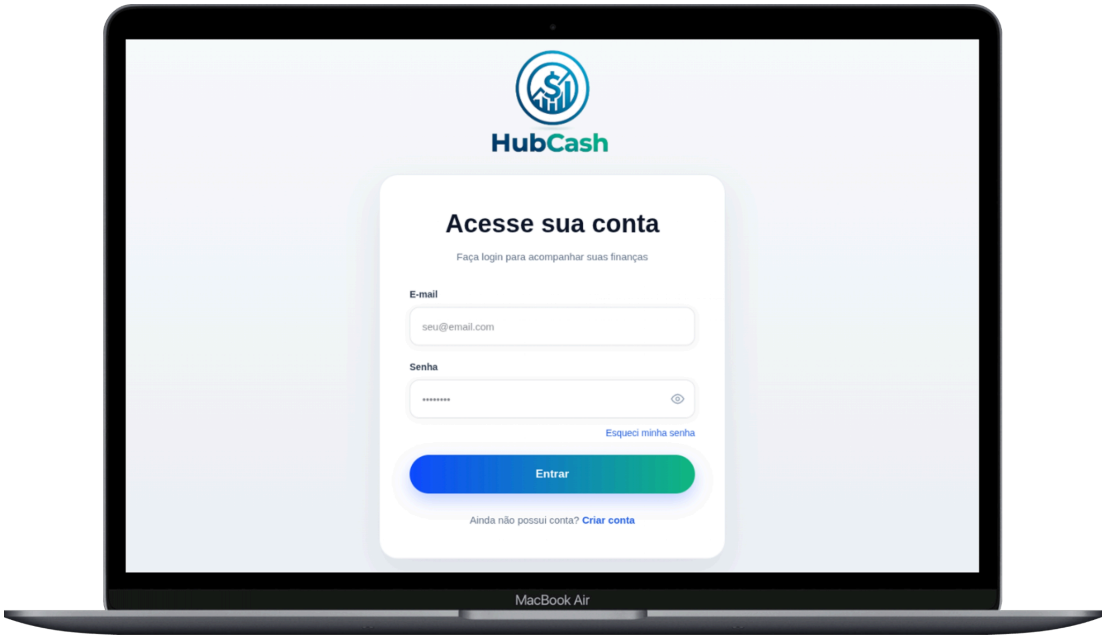
3. **Camada de Back-end (NestJS / API RESTful):** Desenvolvida em Node.js com o *framework* NestJS e linguagem *TypeScript*, a *API RESTful* centraliza as regras de negócio, a validação de dados, a autenticação de usuários via tokens *JWT* e a geração de relatórios.

4. **Camada de Persistência (Banco de Dados Relacional - Supabase/PostgreSQL):** A persistência dos dados é gerenciada por meio de um Banco de Dados Relacional (PostgreSQL) hospedado em nuvem na plataforma *Supabase*. A comunicação entre o *Back-end* e o banco de dados ocorre através do ORM Prisma, garantindo a integridade relacional entre as tabelas do sistema (User e Transaction).

4.5 INTERFACES

Nesta seção, são apresentadas as principais interfaces gráficas que compõem o HubCash. *Front-end* foi desenvolvido visando uma navegação fluida, layout limpo (*clean design*) e respostas visuais imediatas para as ações do usuário.

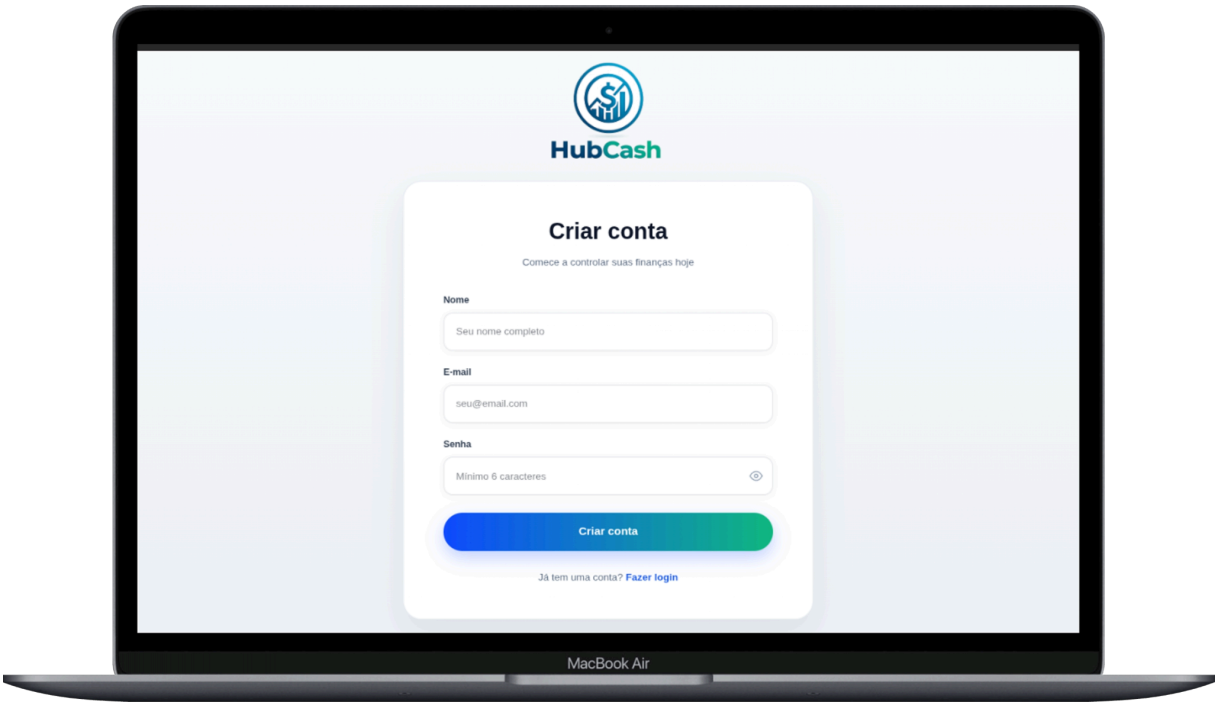
Figura 4 – Tela de Login da aplicação (Versão Desktop)



Fonte: Elaborado pelo autor (2026).

A Figura 4 apresenta a tela de autenticação do sistema, contendo formulário centralizado com validação de campos em tempo real e indicação visual de carregamento durante a verificação do login.

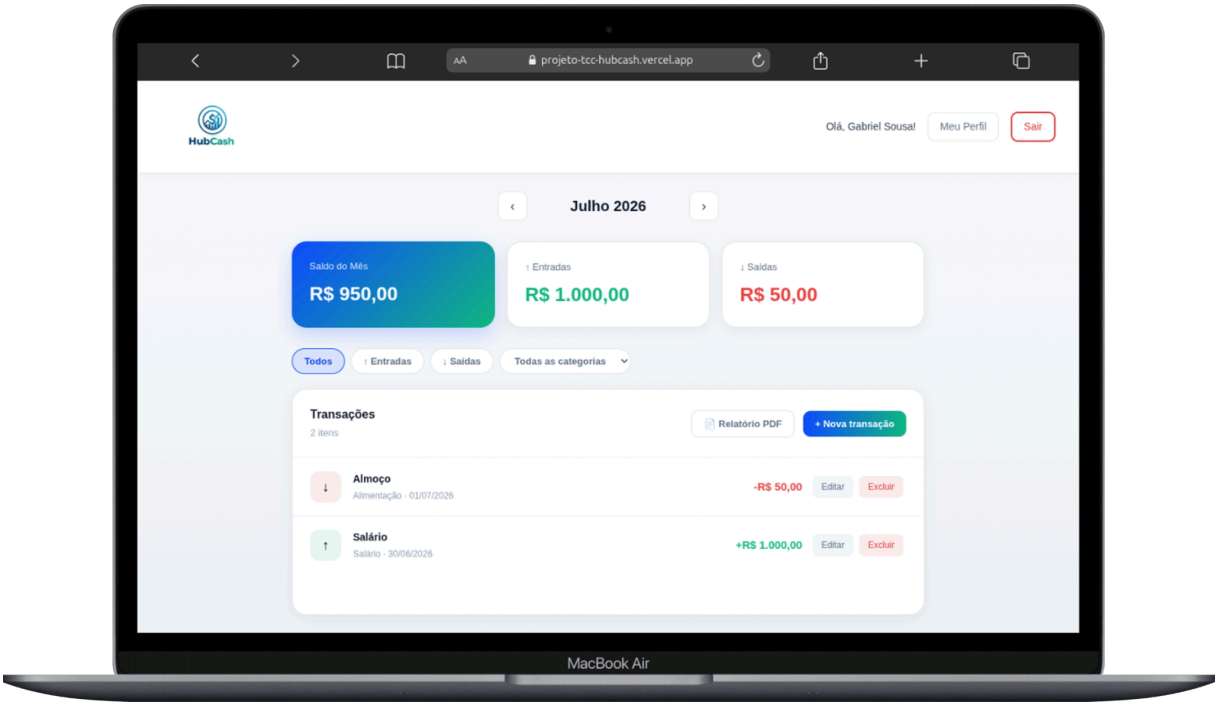
Figura 5 – Tela de Cadastro de Usuário (Versão Desktop)



Fonte: Elaborado pelo autor (2026).

A Figura 5 ilustra a interface de registro de novos usuários, onde são solicitados nome, e-mail e senha. As informações são validadas na submissão e salvas de forma segura no banco de dados com senha criptografada.

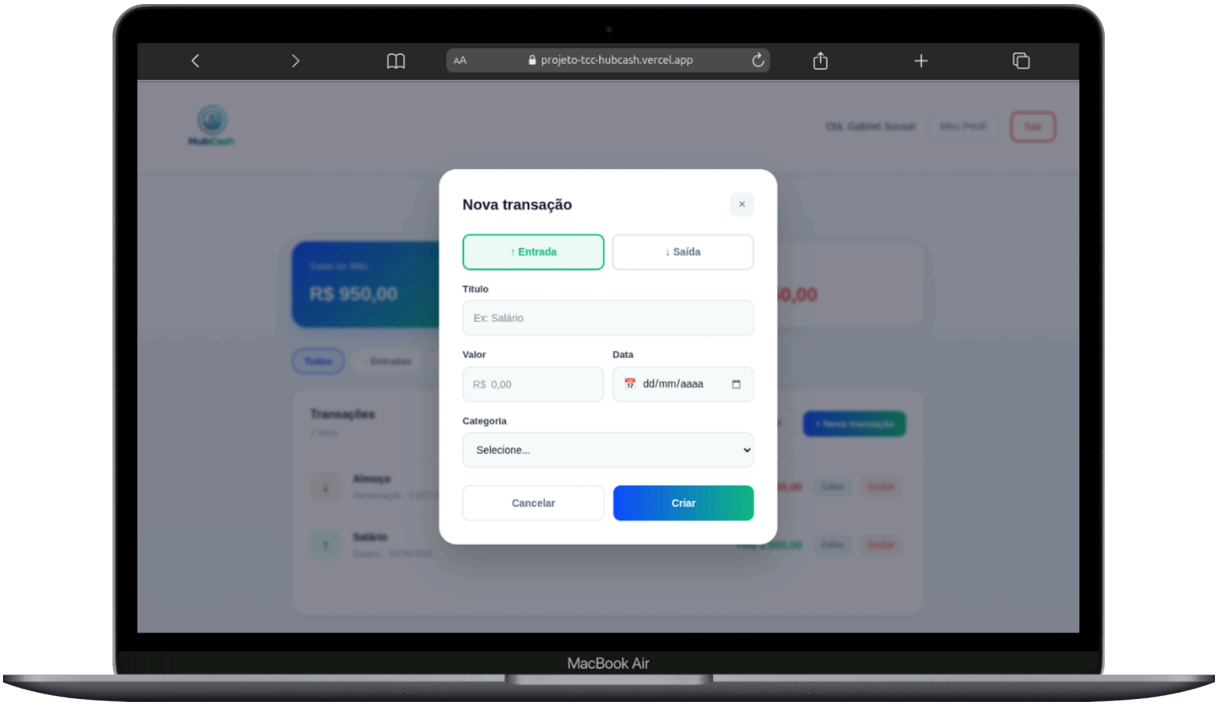
Figura 6 – Painel Principal / Dashboard (Versão Desktop)



Fonte: Elaborado pelo autor (2026).

A Figura 6 exibe o *Dashboard* da aplicação, que consolida os cartões de saldo e totais de entradas/saídas, filtros de despesas ou receitas por categoria e o histórico das últimas movimentações.

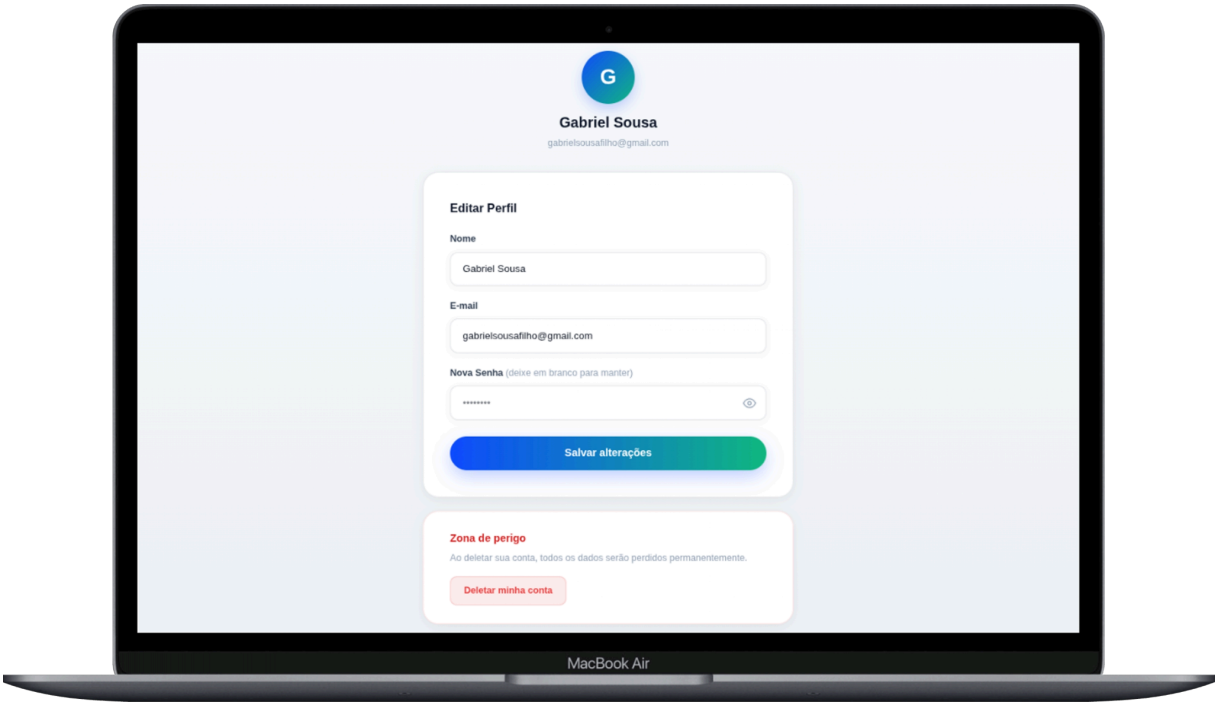
Figura 7 – Modal de Cadastro e Edição de Transações (Versão Desktop)



Fonte: Elaborado pelo autor (2026).

A Figura 7 demonstra o modal suspenso para adição e edição de movimentações financeiras, permitindo ao usuário registrar entradas e saídas com atualização instantânea dos totais sem recarregar a página.

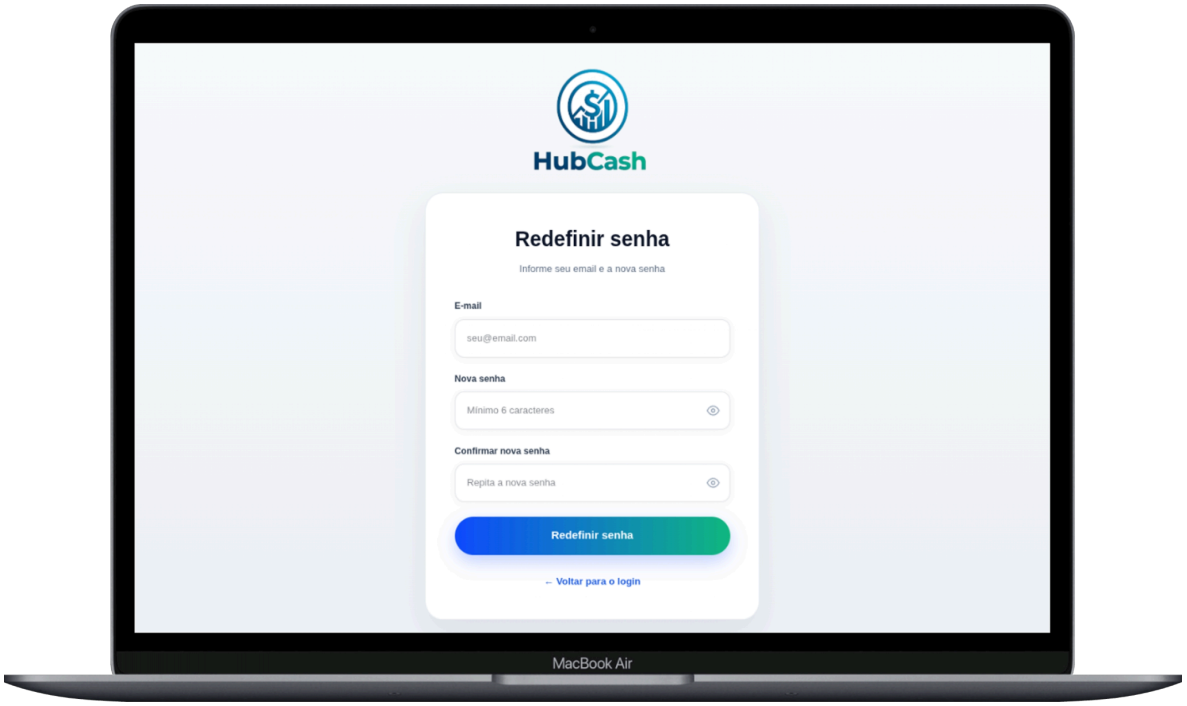
Figura 8 – Tela de Perfil do Usuário (Versão Desktop)



Fonte: Elaborado pelo autor (2026).

A Figura 8 apresenta a interface de gerenciamento do perfil, onde o usuário pode visualizar e atualizar os seus dados cadastrais (como nome e e-mail), alterar a senha de acesso, deletar sua conta e gerenciar as configurações da sua conta de forma simples e segura.

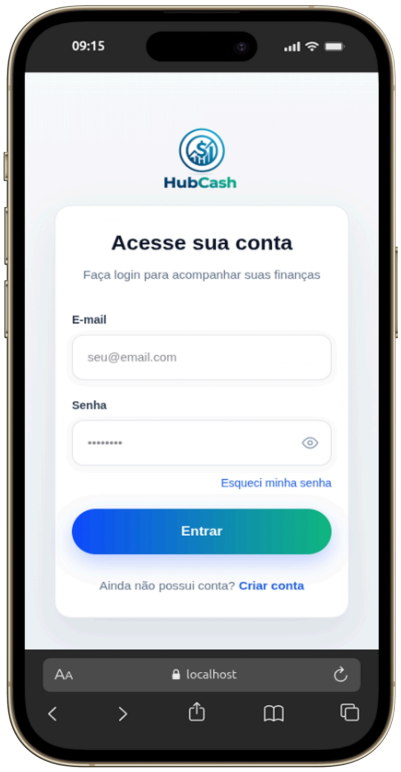
Figura 9 – Tela de Recuperação de Senha (Versão Desktop)



Fonte: Elaborado pelo autor (2026).

A Figura 9 ilustra a interface de redefinição/recuperação de senha, onde o usuário insere o e-mail cadastrado para a redefinição de acesso ao sistema.

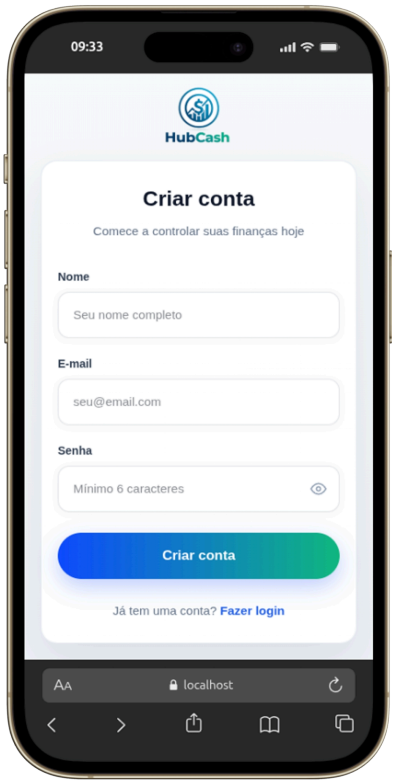
Figura 10 – Tela de Login da aplicação (Versão Mobile)



Fonte: Elaborado pelo autor (2026).

A Figura 10 exibe a interface de autenticação adaptada para telas menores, mantendo os campos de e-mail e palavra-passe organizados verticalmente para facilitar o toque e o acesso rápido.

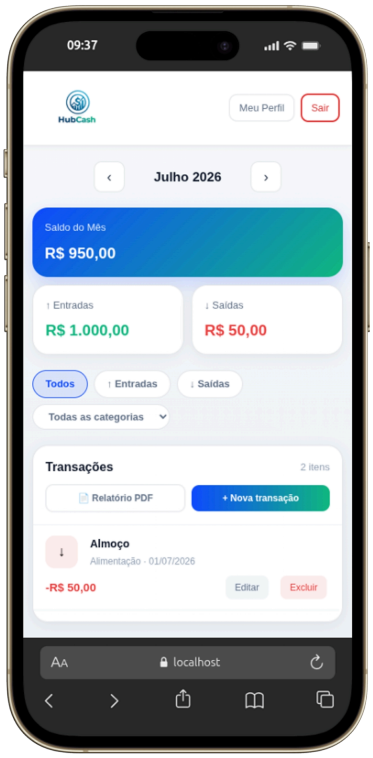
Figura 11 – Tela de Cadastro de Usuário (Versão Mobile)



Fonte: Elaborado pelo autor (2026).

A Figura 11 ilustra a tela de registo de novos utilizadores no telemóvel, ajustando o formulário para um formato fluido e de fácil preenchimento em dispositivos sensíveis ao toque.

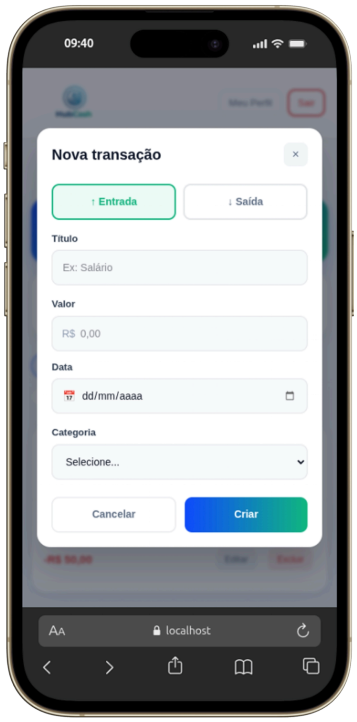
Figura 12 – Painel Principal / Dashboard (Versão Mobile)



Fonte: Elaborado pelo autor (2026).

A Figura 12 apresenta o *Dashboard* em versão móvel, onde os cartões de saldo e atalhos operacionais são reordenados em uma única coluna vertical, garantindo leitura clara e navegação simplificada.

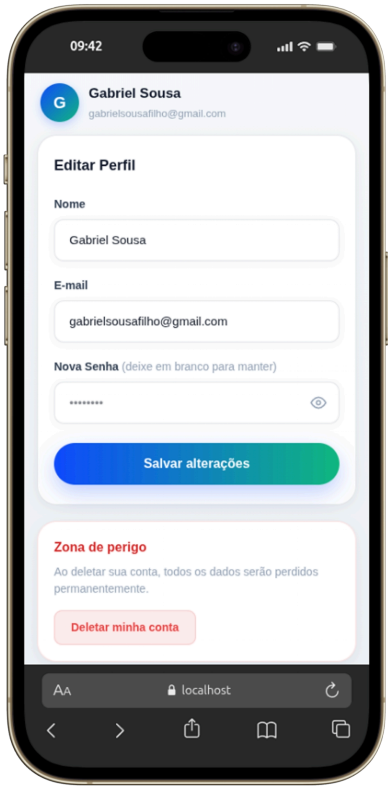
Figura 13 – Modal de Cadastro e Edição de Transações (Versão Mobile)



Fonte: Elaborado pelo autor (2026).

A Figura 13 demonstra o formulário suspenso para adição e edição de movimentações ajustado à tela móvel, permitindo a seleção rápida de valores, datas e categorias com teclado nativo do dispositivo.

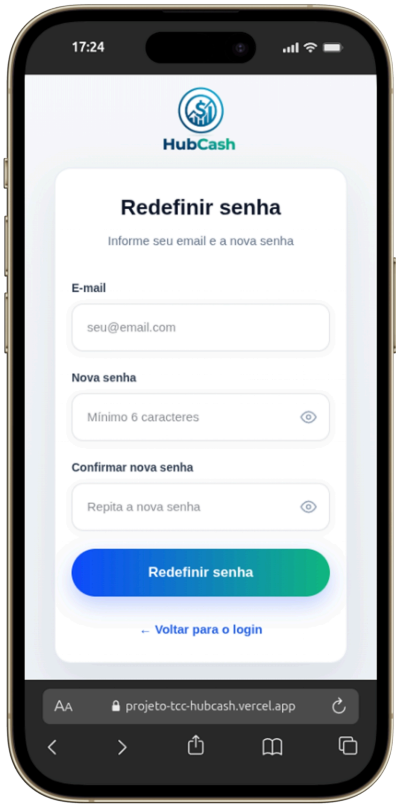
Figura 14 – Tela de Perfil do Usuário (Versão Mobile)



Fonte: Elaborado pelo autor (2026).

A Figura 14 ilustra a página de gestão do perfil na versão móvel, permitindo ao utilizador atualizar os seus dados pessoais e alterar a palavra-passe de forma prática e segura.

Figura 15 – Tela de Recuperação de Senha (Versão Mobile)



Fonte: Elaborado pelo autor (2026).

A Figura 15 ilustra a interface de redefinição/recuperação de senha, onde o usuário insere o e-mail cadastrado para a redefinição de acesso ao sistema.

5 SOFTWARE

Neste capítulo, são apresentados os procedimentos adotados para a implementação (*deploy*) da aplicação HubCash em ambiente de produção, bem como a metodologia e os resultados dos testes executados para validação das funcionalidades e da segurança da plataforma.

5.1 IMPLEMENTAÇÃO

A implementação do HubCash foi estruturada utilizando o modelo de hospedagem em nuvem com arquitetura distribuída, visando garantir alta disponibilidade, segurança no tráfego de dados e automação no fluxo de atualização do código-fonte.

5.1.1 Infraestrutura em Nuvem e Plataformas de Hospedagem

Dada a arquitetura desacoplada do sistema, a camada de *Front-end*, a API *Back-end* e o Banco de Dados foram implantados em provedores independentes especializados em suas respectivas categorias:

- **Hospedagem do Front-end (Vercel):** A aplicação SPA desenvolvida em *React* com *Vite* foi implantada na plataforma *Vercel*. A *Vercel* foi escolhida por oferecer otimização nativa para ativos estáticos, distribuição global via rede de entrega de conteúdo *Content Delivery Network* (CDN) e provimento automático de certificados de segurança SSL/TLS (HTTPS).

- **Hospedagem do Back-end (Render):** A API REST desenvolvida no *framework NestJS* foi hospedada na plataforma Render. O serviço provê um ambiente de execução gerenciado para *Node.js* em produção, tratando o gerenciamento de processos, reconexão automática e terminação SSL/TLS para garantir comunicação segura com o cliente.

- **Banco de Dados em Nuvem (Supabase PostgreSQL):** A persistência dos dados foi alocada no *Supabase*, uma plataforma de banco de dados PostgreSQL totalmente gerenciada em nuvem Database as a Service (DBaaS). O *Supabase* oferece conexões criptografadas, alta disponibilidade e recursos avançados de segurança, garantindo o isolamento e a integridade para as tabelas do sistema (User e Transaction).

O *front-end*[1] e o *back-end*[2] do sistema encontram-se disponíveis em ambiente de produção para fins de demonstração, permitindo a utilização das principais funcionalidades da plataforma. Além disso, o código-fonte[3] também está disponível para consulta, contribuindo para a transparência e a reprodutibilidade do projeto.

[1] Disponível em: <projeto-tcc-hubcash.vercel.app> Acesso em 18 de agosto de 2026

[2] Disponível em: <https://hubcash-backend.onrender.com> Acesso em 18 de agosto de 2026

[3] Disponível em: <https://github.com/gabrielsousa06/projeto-tcc-hubcash> Acesso em 18 de agosto de 2026

5.1.2 Pipeline de Integração e Implantação Contínua (CI/CD)

Para garantir agilidade e confiabilidade na publicação de correções e novas funcionalidades, foi configurado um fluxo de CI/CD integrado aos repositórios remotos do projeto no GitHub:

- **Gatilhos de Deploy Automático:** Tanto a *Vercel* quanto o Render foram conectados diretamente aos seus respectivos repositórios no GitHub. A cada *commit* ou *merge* efetuado na ramificação principal (main), os gatilhos das plataformas são acionados automaticamente.

- **Processo de Build do Front-end:** A Vercel executa o script de compilação do *Vite* (npm run build), validando os tipos em *TypeScript* e gerando os arquivos estáticos otimizados para publicação imediata.

- **Processo de Build do Back-end:** O Render baixa as dependências do projeto, executa o compilador *TypeScript* do *NestJS* (nest build) e inicia a execução da API em modo de produção (node dist/main.js).

5.1.3 Gerenciamento de Variáveis de Ambiente e Migrações

A segurança das credenciais e a flexibilidade do ambiente de produção foram mantidas por meio da segregação estrita de Variáveis de Ambiente e scripts automatizados de migração:

- **Variáveis de Ambiente (*Environment Variables*):** Informações sensíveis e parâmetros de configuração não foram inseridos no código-fonte. Na API (*Back-end*), foram configuradas variáveis no painel da plataforma Render para gerenciar o endereço do banco de dados (DATABASE_URL), a chave secreta para assinatura dos tokens (JWT_SECRET) e a porta de escuta (PORT). No *Front-end*, a URL base da API REST foi injetada em tempo de compilação na Vercel (VITE_API_URL).

- **Execução de Migrações do Banco de Dados:** A sincronização do esquema da base de dados em nuvem foi realizada por meio do Prisma Command-Line Interface (CLI). Durante a esteira de implantação do *Back-end*, o comando npx prisma migrate deploy é executado para aplicar automaticamente no PostgreSQL do Supabase qualquer nova migração pendente, garantindo a consistência das tabelas sem perda de dados existentes.

6 AVALIAÇÃO E RESULTADOS

Neste capítulo, são apresentados e discutidos os resultados obtidos a partir da avaliação de usabilidade e aceitação do sistema HubCash. A condução desta etapa consolida o cumprimento do **sexto objetivo específico** deste trabalho, focado em avaliar a facilidade de uso e a satisfação da aplicação mediante testes práticos e questionário aplicado a uma amostra de 10 (dez) usuários finais.

6.1 AVALIAÇÃO DE USABILIDADE E METODOLOGIA DE TESTES

Para avaliar a percepção de usabilidade, a clareza da interface e a satisfação geral com o HubCash, foi realizada uma pesquisa quantitativa e qualitativa com uma amostra de 10 (dez) usuários finais. A avaliação baseou-se na aplicação de questionários estruturados baseados na escala Likert de 5 pontos (variando de 1 – Discordo Totalmente a 5 – Concordo Totalmente).

Os participantes receberam o link de acesso da aplicação em produção (hospedada na plataforma Vercel) e executaram um roteiro de tarefas pré-definido, composto pelos seguintes passos:

1. Realizar o cadastro de uma nova conta de usuário;
2. Efetuar o *login* no sistema;
3. Cadastrar movimentações financeiras (entradas e saídas);
4. Analisar os indicadores gerados no *Dashboard*;
5. Utilizar os filtros da tabela e exportar o relatório em formato PDF.

Após a realização completa do roteiro, os usuários responderam ao questionário de avaliação de usabilidade de forma totalmente anônima, garantindo a imparcialidade das respostas. Todos os participantes formalizaram a concordância voluntária mediante o Termo de Consentimento Livre e Esclarecido (TCLE) disponibilizado digitalmente.

6.2 ANÁLISE DOS RESULTADOS

A Tabela 3 sintetiza o desempenho do HubCash em cada uma das dez dimensões avaliadas, apresentando as médias numéricas (de 1,00 a 5,00) e a taxa de concordância positiva (percentual de respostas "Concordo Parcialmente" e "Concordo Totalmente").

Tabela 3 – Avaliação de Usabilidade do HubCash

Nº	Critério / Funcionalidade Avaliada	Média (1 a 5)	Concordância (Notas 4 e 5)
Q1	Facilidade de primeiro contato e aprendizado inicial	**5,00**	100%
Q2	Navegação fluida e intuitiva entre telas	**4,90**	100%
Q3	Design visual limpo, organizado e agradável	**5,00**	100%
Q4	Disposição dos cartões no *Dashboard*	**4,80**	100%
Q5	Procedimento de inclusão, edição e exclusão de transações	**5,00**	100%
Q6	Eficiência da busca e filtros aplicados à tabela	**5,00**	100%
Q7	Clareza e utilidade do relatório financeiro exportado em PDF	**5,00**	100%
Q8	Desempenho operacional e rapidez de resposta do sistema	**4,60**	90%
Q9	Confiabilidade no cadastro, login e encerramento de sessão	**4,60**	90%
Q10	Atendimento às necessidades financeiras e potencial de adoção	**5,00**	100%

-	**MÉDIA GERAL DE SATISFAÇÃO DO SISTEMA**	**4,89**	**97,8%**

Fonte: Elaborado pelo autor (2026).

Como demonstra a Tabela 3, o sistema alcançou uma Média Geral de Satisfação de 4,89, o que corresponde a um índice de aprovação global de 97,8%.

Destaca-se que seis dos dez quesitos avaliados (Q1, Q3, Q5, Q6, Q7 e Q10) obtiveram nota máxima absoluta (5,00 com 100% de aprovação). Esse resultado evidencia a eficiência na criação de uma interface limpa, ágil e livre de sobrecarga visual.

Os quesitos relativos à navegação fluida (Q2 = 4,90) e à disposição dos *cards* de resumo (Q4 = 4,80) mantiveram 100% de percepção positiva, validando a arquitetura SPA adotada.

As dimensões de desempenho operacional (Q8 = 4,60) e confiabilidade do fluxo de acesso (Q9 = 4,60) obtiveram 90% de concordância positiva. A ligeira variação registrada na questão Q8 deve-se a oscilações pontuais de latência na comunicação via API com os serviços em nuvem (*backend* no Render e banco de dados PostgreSQL no Supabase).

Com a realização dos testes práticos e a compilação detalhada dos indicadores quantitativos e qualitativos, o sexto objetivo específico do trabalho foi concluído com sucesso, comprovando que o HubCash atinge níveis elevados de usabilidade, aceitação e eficiência na gestão de finanças pessoais.

7 CONSIDERAÇÕES FINAIS

O presente trabalho apresentou o desenvolvimento do HubCash, um sistema web voltado ao controle e ao planejamento financeiro pessoal, projetado com foco na usabilidade, clareza visual, desempenho e segurança da informação. A motivação para a construção da aplicação pautou-se na constatação de que grande parte dos indivíduos enfrenta dificuldades no gerenciamento do orçamento diário, cenário frequentemente agravado pela complexidade de ferramentas tradicionais ou excessivamente burocráticas.

Com a conclusão das etapas de modelagem, desenvolvimento, implementação e validação descritas ao longo deste relatório, constata-se que os objetivos propostos foram

plenamente alcançados. A adoção de uma arquitetura desacoplada no modelo cliente-servidor permitiu a construção de uma aplicação escalável, em que o *Back-end* em NestJS e PostgreSQL assegurou o processamento eficiente das regras de negócio, enquanto o *Front-end* em React proporcionou uma interface moderna, ágil e intuitiva.

Sob a ótica da experiência do usuário, os testes empíricos de usabilidade demonstraram que o HubCash é um ótimo caminho para o controle financeiro pessoal, alcançando uma média geral de satisfação de 4,89 (97,8% de aprovação global). A aplicação reduziu com êxito a alta carga cognitiva presente em planilhas e em sistemas com excesso de funcionalidades, entregando uma interface limpa e organizada que facilita o registro de receitas e despesas e contribui diretamente para uma tomada de decisão mais consciente.

Como limitações, destacam-se a ausência do desenvolvimento de um aplicativo móvel e a falta de recursos avançados de automação. Como trabalhos futuros, propõem-se a verificação de e-mail durante o processo de cadastro de usuários, implementação de alertas e metas de gastos, a inclusão de gráficos estatísticos (como gráficos de pizza e de barras) para facilitar a visualização, e a utilização de "caixinhas" de metas e reservas financeiras.

Dessa forma, conclui-se que o HubCash atende aos objetivos propostos, apresentando-se como uma contribuição relevante para o desenvolvimento de soluções voltadas ao controle financeiro pessoal, com foco na usabilidade e no apoio à tomada de decisões**.**

8 REFERÊNCIAS

CONFEDERAÇÃO NACIONAL DO COMÉRCIO DE BENS, SERVIÇOS E TURISMO (CNC). Pesquisa de Endividamento e Inadimplência do Consumidor (Peic). Brasília: CNC, 2024. Disponível em: https://www.portaldocomercio.org.br. Acesso em: 10 fev. 2026.

ELISIÁRIO, Ricardo Medeiros. Desenvolvimento de uma aplicação web para simulação e planejamento financeiro. 2025.

ELISIÁRIO, Ricardo Medeiro. *Desenvolvimento de uma aplicação web para simulação e planeamento financeiro.* 2025. Projeto (Mestrado em Engenharia Informática) – Instituto Politécnico de Tomar, Escola Superior de Tecnologia de Tomar, 2025.

GITMAN, Lawrence J. *Princípios de administração financeira*. 12. ed. São Paulo: Pearson, 2010.

GOMES, Júlia da Silva. *A tecnologia da informação como apoio ao planejamento financeiro pessoal.* 2023. Trabalho de Graduação (Tecnologia em Análise e Desenvolvimento de Sistemas) – Faculdade de Tecnologia de Franca – "Dr. Thomaz Novelino", Centro Paula Souza, Franca, 2023.

HOUSEL, Morgan. *A psicologia financeira: lições atemporais sobre fortuna, ganância e felicidade*. Tradução de Maria Luiza X. de A. Borges. Rio de Janeiro: HarperCollins, 2021.

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

APÊNDICE A – QUESTIONÁRIO DE AVALIAÇÃO

1. TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO (TCLE)

Você está sendo convidado(a) a participar da pesquisa de avaliação de usabilidade do HubCash, um sistema web desenvolvido para auxílio no controle e planejamento financeiro pessoal. A sua participação é voluntária e anônima. Os dados coletados serão utilizados exclusivamente para fins acadêmicos no âmbito do Trabalho de Conclusão de Curso (TCC) do Curso de Bacharelado em Análise e Desenvolvimento de Sistemas do Instituto Federal do Piauí (IFPI) – Campus Picos.

2. ROTEIRO DAS TAREFAS PRÉ-DEFINIDAS

Antes de responder ao questionário, solicitamos que acesse a aplicação em produção no endereço hospedado na Vercel e realize a seguinte sequência de ações:

1. Cadastro: Criar uma nova conta de usuário preenchendo nome, e-mail e senha.
2. Autenticação: Realizar o *login* com as credenciais cadastradas.
3. Gestão de Transações: Cadastrar pelo menos duas movimentações financeiras (uma entrada/receita e uma saída/despesa).
4. Análise de Dados: Acompanhar a atualização dos cartões de indicadores (*KPIs*) e dos gráficos resumo no *Dashboard*.
5. Filtragem e Exportação: Aplicar filtros na tabela de histórico e gerar o relatório consolidado em formato PDF.

3. QUESTIONÁRIO DE AVALIAÇÃO (ESCALA LIKERT DE 5 PONTOS)

Escala de Avaliação: 1 – Discordo Totalmente | 2 – Discordo Parcialmente | 3 – Neutro | 4 – Concordo Parcialmente | 5 – Concordo Totalmente.

- Q1: O primeiro contato com a aplicação HubCash demonstrou ser simples, permitindo a rápida compreensão e o aprendizado de suas funcionalidades básicas.
- Q2: A navegação entre as diferentes seções da aplicação (Autenticação, Dashboard e

Histórico de Transações) é fluida, clara e intuitiva.

- Q3: O design visual da interface gráfica apresenta-se organizado, limpo e agradável, sem sobrecarga visual de informações.
- Q4: A disposição dos cartões de resumo e gráficos no *Dashboard* possibilita uma análise rápida e precisa do estado financeiro.
- Q5: O procedimento para inclusão, alteração e remoção de lançamentos (receitas e despesas) ocorre de maneira ágil e eficiente.
- Q6: Os filtros aplicados à tabela facilitam a localização de transações específicas.
- Q7: A funcionalidade de geração e exportação do relatório financeiro em formato PDF é satisfatória e apresenta os dados estruturados de forma legível.
- Q8: O sistema demonstra elevado desempenho operacional, respondendo com rapidez aos comandos efetuados, sem a ocorrência de lentidão ou interrupções.
- Q9: A aplicação transmite confiabilidade e clareza durante as etapas de cadastramento de conta, autenticação (*login*) e encerramento de sessão (*logout*).
- Q10: De modo geral, a solução desenvolvida atende adequadamente às necessidades de gestão financeira e possui elevado potencial de adoção pelo público.

4. CAMPO ABERTO PARA FEEDBACK QUALITATIVO

Comentários e Sugestões: Deixe aqui suas considerações, críticas construtivas ou sugestões de novas funcionalidades para o aprimoramento do HubCash.