**PÓS-GRADUAÇÃO EM DESENVOLVIMENTO FULL STACK**

**DESENVOLVIMENTO DE APLICAÇÃO DE FINANÇAS PESSOAIS**

LUCAS COIMBRA SCAGLIA

Orientador: Júlio Henrique Araújo Pereira Machado

2025



**SUMÁRIO**

[1. Contextualização da proposta 3](#_Toc203632762)

[2. Objetivos da construção da solução 4](#_Toc203632763)

[3. Elaboração da jornada do usuário 5](#_Toc203632764)

[4. Apelo mercadológico da solução 6](#_Toc203632765)

[5. Ciclo de desenvolvimento da solução 7](#_Toc203632766)

[6. Mockup da proposta de solução 8](#_Toc203632767)

[7. Arquitetura de Software 9](#_Toc203632768)

[8. Validação da solução 10](#_Toc203632769)

[9. Registros das evidências do projeto 11](#_Toc203632770)

[10. Considerações finais e expectativas 12](#_Toc203632771)

[Referências 13](#_Toc203632772)

# Contextualização da proposta

As atuais aplicações disponíveis para controle de finanças pessoais, apesar dos vastos recursos, podem acabar sendo abandonadas em pouco tempo. O motivo disto é que esperam que o usuário as use diariamente, documentando cada compra, gasto ou investimento. Essa atividade acaba sendo considerada maçante, e o esquecimento torna os dados inconsistente com o tempo. Ao perceber o trabalho gerado, os usuários muitas vezes podem optar por outros meios onde se sintam mais confortáveis para fazer este controle, ainda que mais simples e com menos recursos, como uma planilha no computador ou um caderno.

Diante desta característica, propõem-se o desenvolvimento de uma plataforma web de gestão de finanças pessoais, que visa oferecer uma praticidade de forma minimalista, onde o planejamento financeiro é realizado somente uma vez por mês. A cada novo ciclo, o usuário conseguirá perceber se foi possível cumprir o planejamento anterior e fazer os ajustes necessários para o próximo mês, numa jornada de melhoria evolutiva do seu controle financeiro.

# Objetivos da construção da solução

Atualmente, sou coordenador de engenharia de software em uma das maiores empresas de varejo do Brasil. Ao longo da minha jornada, me especializei em otimizar os processos de equipes de desenvolvimento de software com ênfase em Qualidade, DevOps e Gestão de *Releases*. Contudo, apesar de ter conhecimento em desenvolvimento de automações e pipelines, em conversas mais técnicas relacionadas a arquitetura e desenvolvimento em si, nem sempre consigo contribuir como gostaria, por falta de conhecimento prático de contribuições em aplicações.

Dado este contexto, pretendo desenvolver uma aplicação simples de ponta a ponta. O sistema terá como objetivo principal apoiar o usuário no controle financeiro mensal, permitindo que ele:

* Cadastre e gerencie contas fixas em formato de checklist, podendo marcá-las como pagas;
* Informe sua renda líquida, saldo disponível e valores destinados a despesas variáveis;
* Acompanhe um resumo mensal que mostra a proporção entre despesas fixas, variáveis e o saldo livre;

Para alcançar esses objetivos, o trabalho será estruturado em diferentes etapas complementares. Inicialmente, será definido o escopo funcional da aplicação, onde serão elaboradas as jornadas do usuário. Na sequência, serão tomadas as decisões de arquitetura da solução, envolvendo a escolha das tecnologias de *backend*, *frontend* e banco de dados. Então, será conduzido o desenvolvimento do *frontend* em React + Vite, do *backend* em Node.js + Express e a criação do banco de dados PostgreSQL.

Por fim, será realizada a integração da aplicação em um ambiente de nuvem, garantindo que a solução esteja disponível para uso e possa ser validada em condições semelhantes às de produção.

# Elaboração da jornada do usuário

Segundo Cohn (2004), histórias de usuário são descrições simples e concisas de funcionalidades escritas do ponto de vista do usuário final, geralmente estruturadas no formato “Como [usuário], quero [ação], para [benefício]”. Esse modelo facilita o entendimento comum entre a equipe de desenvolvimento e os stakeholders, além de apoiar a priorização e o refinamento dos requisitos.

Baseado nesta metodologia, apresentarei abaixo as histórias de usuário que serão utilizadas para a criação da aplicação.

* 1. **História 1 – Identificação do usuário**

**Como** usuário

**Quero** me autenticar no sistema com e-mail e senha

**Para** ter acesso às minhas informações financeiras de forma segura.

**Critérios de aceitação:**

* Caso o e-mail não esteja cadastrado, o usuário deve poder criar uma conta informando nome e senha.
* Caso o e-mail esteja cadastrado, o usuário deve poder informar sua senha para acessar.
* Caso a senha informada esteja errada, o usuário deve ser notificado e impedido de prosseguir.
  1. **História 2 - Encerramento da sessão de usuário**

**Como** usuário autenticado

**Quero** encerrar minha sessão de forma segura

**Para** garantir que outras pessoas não acessem meus dados

**Critérios de aceitação:**

* O usuário deve ter a opção de encerrar a sessão a partir da tela principal do sistema.
* Ao encerrar a sessão, o usuário deve ser redirecionado para a tela inicial de login.
* Após o encerramento da sessão, nenhuma informação do usuário deve permanecer acessível na aplicação.
  1. **História 4 - Tela Principal (Visão Geral do Mês)**

**Como** usuário autenticado

**Quero** visualizar em uma tela principal o planejamento do mês, incluindo saldo disponível, checklist de contas e um resumo da distribuição da renda

**Para** acompanhar rapidamente meus compromissos financeiros, organizar os pagamentos mensais e entender melhor como minha renda está distribuída.

**Critérios de aceitação:**

* O cabeçalho deve exibir automaticamente o mês e ano atuais, permitindo avançar ou retroceder.
* O usuário deve poder informar o valor de saldo disponível no início do mês.
* A qualquer momento, e em qualquer período (mês), o usuário deve poder cadastrar contas fixas, onde deverá informar a descrição, data de vencimento e valor.
* Cada conta deve ser exibida em formato de checklist, podendo ser marcada como paga.
* Deve ser possível editar ou excluir uma conta cadastrada.
* Deve ser possível copiar as contas de um mês anterior.
* O sistema deve exibir um resumo que mostre a proporção entre contas fixas, despesas variáveis e saldo livre.

# Apelo mercadológico da solução

A proposta da solução é simplificar ao máximo a experiência do usuário, permitindo um controle financeiro mensal, ao invés de interações diárias como é comumente exigido nas aplicações já disponíveis no mercado, como Minhas Economias (MINHAS ECONOMIAS, 2025), Organizze (ORGANIZZE, 2025) e Mobills (MOBILLS, 2025).

A aplicação desenvolvida será focada apenas em jornadas simples e práticas. Seu apelo é focado na relação entre o esforço do usuário e o valor percebido. Dito isso, não há a pretensão de desenvolver funcionalidades robustas como integrações bancárias, relatórios complexos e cadastros de metas financeiras.

Do ponto de vista mercadológico, o público-alvo são pessoas que já tentaram utilizar outras ferramentas, e que desistiram por falta de tempo ou disciplina. Um público que valorize a autonomia para decidir quais itens são relevantes cadastrar e que gostaria de apoio não só para organizar essas contas, mas para conseguir passar o mês com o valor que foi programado.

O desenvolvimento desta solução vai além de um exercício técnico. Ela demonstra potencial para explorar um nicho que ainda é pouco atendido.

# Ciclo de desenvolvimento da solução

O desenvolvimento da aplicação seguiu um ciclo bem definido, com cada etapa sendo tratada como um tópico independente. A seguir, detalho o que foi realizado em cada fase e as decisões tomadas.

* 1. **Ideação da solução**

O ponto de partida foi um brainstorming para definir a essência do projeto. A partir da identificação de um nicho de mercado, surgiu a ideia de criar uma ferramenta de controle financeiro minimalista, focada no planejamento mensal. Este conceito guiou todas as decisões subsequentes, buscando simplificar a jornada do usuário e evitar as complexidades comuns de aplicativos concorrentes.

* 1. **Especificação das jornadas de usuário e requisitos**

Com a ideia principal estabelecida, a próxima etapa foi a especificação das jornadas do usuário. Utilizando a metodologia de histórias de usuário, cada funcionalidade foi detalhada do ponto de vista do usuário final. Isso permitiu definir os requisitos do sistema de forma clara e objetiva, servindo como base para o desenvolvimento.

* 1. **Criação do Mockup da solução**

Para visualizar a proposta, foram criados protótipos de baixa fidelidade no aplicativo Miro (https://miro.com/). Os mockups serviram como uma representação visual do design e da usabilidade da aplicação, ajudando a validar as jornadas e a ajustar o fluxo de navegação antes de iniciar a codificação.

Os mockups elaborados estão disponíveis para consulta no Capítulo 6.

* 1. **Definição da arquitetura e escolha das tecnologias**

A arquitetura da solução foi projetada para ser simples, escalável e de fácil manutenção. As tecnologias escolhidas para o desenvolvimento foram:

* *Backend*: Node.js com o framework Express, ideal para criar APIs robustas.
* *Frontend*: React com Vite, para uma interface reativa e de alta performance.
* Banco de Dados: PostgreSQL, um banco de dados relacional confiável e amplamente utilizado no mercado.
  1. **Desenho conceitual e modelagem do banco de dados**

Com as tecnologias definidas, foi realizada a modelagem conceitual do banco de dados no aplicativo Draw IO (https://www.drawio.com/). A estrutura das tabelas, campos e relacionamentos foi definida para garantir a integridade dos dados.

A modelagem conceitual do banco de dados está disponível para consulta no Capítulo 7.

* 1. **Criação dos repositórios e estruturação do projeto**

O código-fonte foi armazenado no GitHub (https://github.com/lucascoimbra-lab/finance-app). Está organizado como um monorepo, onde as principais pastas são:

* *Backend*: Desenvolvido em Node.js/Express
* *Frontend*: Desenvolvido em React/Vite
* Evidências: artefatos que evidenciam a criação da aplicação desde sua concepção, conforme solicitado nas instruções de desenvolvimento deste trabalho.
  1. **Configuração da aplicação para rodar em containers**

Para garantir um ambiente de desenvolvimento e produção consistente, toda a aplicação foi conteinerizada usando Docker. Um arquivo docker-compose.yml foi configurado para orquestrar os containers do *backend*, *frontend* e banco de dados, simplificando a inicialização e a integração de todas as partes do sistema.

* 1. **Desenvolvimento Full-Stack**

Esta foi a etapa central do projeto. O desenvolvimento do *frontend* e do *backend* ocorreu em paralelo, com a criação das telas, a implementação das funcionalidades, a configuração dos endpoints da API e a conexão com o banco de dados. Tudo isso seguindo as jornadas de usuário e os mockups definidos previamente.

* 1. **Publicando da solução em nuvem**

Para automatizar a entrega e a implantação da aplicação, foram utilizadas as soluções Vercel (https://vercel.com/) para o *frontend* e o Railway (https://railway.com/) para o *backend* e o banco de dados. Em ambas, foram configurados gatilhos no ramo principal do repositório (branch main), a fim de gerar nova compilação e publicação a cada alteração.

A aplicação pode ser acessada no link: <https://finance-app-one-eta.vercel.app/>

# Mockup da proposta de solução

As telas a seguir foram feitas utilizando o Miro e foram criadas como protótipo do design e usabilidade da aplicação. O material aqui proposto não representa o design final da aplicação e a jornada final após o desenvolvimento.

* 1. **Tela de Login**

Interface gráfica do usuário, Diagrama, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Figura 1 - Tela de login

* + 1. *Identificação*

Na tela inicial do sistema, será solicitado o e-mail do usuário.

Ao avançar, será persistido se o e-mail está cadastrado:

* Se negativo, será necessário realizá-lo informando seu nome e uma senha.
* Se positivo, vai solicitar a senha do usuário.

Ao avançar, será exibida a tela principal do sistema.

* + 1. *Recuperação da senha*

Na tela de identificação, deverá ser possível recuperar a senha através da informação do e-mail cadastrado.

* + 1. *Dicas de utilização*

Serão implementadas em botões ao lado de pontos chaves do sistema, para que o usuário o utilize corretamente.

## *Tela Principal (Visão Geral do Mês)*

Interface gráfica do usuário, Tabela

O conteúdo gerado por IA pode estar incorreto.

Figura 2 - Tela de Planejamento Mensal

Uma imagem contendo Linha do tempo

O conteúdo gerado por IA pode estar incorreto.

Figura 3 - Utilização da tela de Planejamento

A tela principal será dividida por sessões, que serão detalhadas a seguir:

### Cabeçalho do Mês

Ao entrar na tela, ela mostrará automaticamente os dados do mês atual, informando no cabeçalho o nome do mês e o ano.

Ainda neste cabeçalho, será possível avançar ou recuar o mês.

### Saldo disponível

Neste campo o usuário deverá informar o valor que ele tem disponível para efetuar os pagamentos do mês.

### Checklist de Contas

Nesta sessão, o usuário poderá cadastrar contas, informando uma descrição, dia de vencimento e o valor.

Cada conta cadastrada será mostrada em formato de checklist, onde ele poderá editá-la, excluí-la ou sinalizá-la como paga.

Para facilitar a rotina do usuário, deverá ser possível copiar os saldos e contas cadastrados no mês anterior, evitando ter que preencher tudo novamente.

### Resumo

Nesta sessão, o usuário deverá informar o valor que ele precisa reservar para contas variáveis. Ou seja, aquelas que não aparecem no Checklist, pois não são contas fixas.

Com este dado, a sessão de resumo deverá informar o percentual do seu salário que é destinado para Contas (itens do Checklist), Despesas Variáveis (valor do saldo informado nesta sessão) e a sobra, que seria o “Saldo não comprometido”. Ou seja, um valor que ele poderia usar livremente.

### Logout

Deverá ter um botão de sair, para encerrar a sessão do usuário.

# Arquitetura de Software

Este capítulo detalha a arquitetura da aplicação desenvolvida, que reflete as escolhas técnicas e a abordagem que adotei para o projeto. A estrutura foi pensada para ser moderna e, ao mesmo tempo, objetiva, dividindo a solução em três camadas principais. Essa organização foi fundamental para garantir que o desenvolvimento pudesse ser feito de forma organizada e escalável.

A figura a seguir ilustra a arquitetura da aplicação, baseada no Modelo C4. Essa abordagem facilita a compreensão dos diferentes elementos do sistema, organizando-os em camadas que mostram a relação entre o usuário, os contêineres de software (frontend e backend) e o banco de dados.

Diagrama

O conteúdo gerado por IA pode estar incorreto.

Figura 7.1 - Arquitetura da Aplicação – Modelo C4

O fluxo da solução começa com o usuário final, que interage com o *Frontend* através de um navegador web.

Quando o usuário realiza uma ação, o *Frontend* envia uma requisição para o *Backend*, que executa a lógica do negócio, como a validação dos dados de uma conta, conectando-se com o banco de dados, e retornando a resposta para o *Frontend*. Esse ciclo de comunicação é o que garante que a aplicação esteja sempre sincronizada, refletindo as ações do usuário em tempo real.

As tecnologias escolhidas para o *Frontend* (React + Vite) e *Backend* (Node.js + Express) foram escolhidas por terem sido apresentadas nas aulas pelos professores do curso, já que são amplamente difundidas no mercado.

Minha decisão para a camada de dados foi o PostgreSQL. Fui com o modelo relacional porque, para um sistema que lida com finanças, a integridade dos dados e a confiabilidade são prioridade.

Diagrama, Esquemático

O conteúdo gerado por IA pode estar incorreto.

Figura 7.2 – Modelagem conceitual do banco de dados

Para a hospedagem do *Frontend*, utilizei a Vercel. Sua integração nativa com o React e a automação de deploy com o GitHub fizeram com que a publicação de cada nova alteração fosse um processo praticamente instantâneo. Já para o *Backend* e o Banco de Dados, a Railway se mostrou a solução ideal. Ela simplificou toda a complexidade de gerenciar a infraestrutura, me dando a liberdade de focar apenas no desenvolvimento.

A figura a seguir dá uma visão final da arquitetura, considerando sua hospedagem citada anteriormente.

Diagrama

O conteúdo gerado por IA pode estar incorreto.

Figura 7.3 - Arquitetura da Aplicação

# Validação da solução

Para garantir que o ambiente de desenvolvimento seja o mais próximo possível ao de produção, toda a solução foi estruturada em containers. Um arquivo Docker-Compose.yml foi configurado na raiz do projeto, garantindo a orquestração dos containers do *backend*, *frontend* e banco de dados.

Para esta fase inicial do projeto, escopo deste trabalho, serão utilizados apenas testes fim-a-fim. Os cenários de testes serão descritos devem ser executados a cada nova atualização do sistema.

## Cenários de Testes na Tela de Login

CT001 (Positivo): Acessar a tela de login, inserir um e-mail já cadastrado, inserir a senha correta e avançar. Resultado esperado: A tela principal deve ser exibida.

CT002 (Negativo): Acessar a tela de login, inserir um e-mail já cadastrado, inserir a senha incorreta e avançar. Resultado esperado: Uma mensagem de erro de senha incorreta deve ser exibida, sem avançar para a tela principal.

CT003 (Positivo): Acessar a tela de login, inserir um e-mail não cadastrado, avançar. Em seguida, fazer o cadastro informando o nome e cadastrando uma senha. Resultado esperado: A tela principal deve ser exibida e o novo usuário deve ser criado.

## Cenários de Testes da Tela Principal (Visão Geral do Mês)

CT005 (Positivo): Acessar a tela principal. Resultado esperado: O cabeçalho deve exibir o mês e o ano atual.

CT006 (Positivo): Clicar no botão para avançar o mês e de recuar o mês. Resultado esperado: O cabeçalho deve mudar para o mês seguinte e para o anterior.

CT007 (Positivo): Informar um valor válido no campo "Saldo disponível". Resultado esperado: O valor deve ser salvo e exibido corretamente.

CT008 (Positivo): Cadastrar uma nova conta com descrição, dia de vencimento e valor. Resultado esperado: A conta deve ser exibida na lista em formato de checklist.

CT009 (Positivo): Clicar no botão "Editar" de uma conta. Resultado esperado: Os campos da conta devem ser editáveis e as alterações salvas após a confirmação.

CT010 (Positivo): Clicar no botão "Excluir" de uma conta. Resultado esperado: A conta deve ser removida da lista.

CT011 (Positivo): Marcar uma conta como paga. Resultado esperado: A conta deve ser visualmente sinalizada como paga (checkbox marcado e texto taxado).

CT012 (Positivo): Clicar no botão "+ Copiar todas as contas e saldos do mês anterior”. Resultado esperado: As contas e o saldo do mês anterior devem ser replicados para o mês atual.

CT013 (Positivo): Preencher o campo de "Saldo para despesas variáveis". Resultado esperado: A sessão de resumo deve ser atualizada para mostrar os percentuais de “Total em Contas”, “Saldo para despesas variáveis” e “Saldo não Comprometido”.

CT014 (Positivo): Verificar se o cálculo de percentuais é feito de forma precisa e que a soma total é igual a 100%.

CT015 (Positivo): Clicar no botão "Sair". Resultado esperado: A sessão deve ser encerrada e o usuário redirecionado para a tela de login.

# Registros das evidências do projeto

O trabalho está disponível no GitHub: <https://github.com/lucascoimbra-lab/finance-app>.

Toda a evolução dos *commits* permitem uma visão detalhada da jornada, desde a estruturação inicial dos *frameworks*, desenvolvimento das telas, correções de *bugs*, melhorias visuais e adaptações para publicação em nuvem.

Além disso, na pasta “evidências” localizada na raiz do código, é possível visualizar:

* Imagens do Mockup feito no Miro.
* Desenho do modelo conceitual do banco de dados.
* Script de criação utilizado para modelar o banco de dados.
* *Collection* do *Postman* para testes dos *endpoints* da aplicação.
* Arquivo de PowerPoint utilizado para o desenho da arquitetura.
* O Word utilizado para escrever este documento.

# Considerações finais e expectativas

O curso me permitiu absorver os fundamentos de conceitos essenciais e observar diferentes abordagens. As aulas mantiveram meu interesse, e alguns tópicos me cativaram de maneira especial. Um exemplo foi o tema de Design de UI, que eu não esperava aprofundar, mas que agregou um valor significativo à minha atuação profissional. Hoje, aplico os conhecimentos adquiridos diretamente na minha rotina de implementação de soluções com a equipe de desenvolvimento.

Não foi necessário esperar a conclusão do curso para perceber minha evolução nos refinamentos técnicos junto à equipe do meu trabalho. Deste então, consigo reconhecer os conceitos do curso nas discussões técnicas e melhorar minha absorção e contribuição.

A construção deste trabalho de conclusão me desafiou a entrar a fundo nos conceitos vistos, estudar os detalhes por conta própria e tirar dúvidas que, até então, eu nem sabia que tinha. De fato, não há melhor forma de aprender do que botar em prática.

Minha perspectiva para o futuro é de seguir construindo meu diferencial através de um perfil *Generalist Specialist*, conciliando a busca acompanhar o mercado dinâmico da tecnologia com o desafiador mundo executivo de gestão de equipes.

# Referências

COHN, M. User stories applied: for agile software development. Boston: Addison-Wesley, 2004.

MINHAS ECONOMIAS. Disponível em: https://www.minhaseconomias.com.br/. Acesso em: 25 ago. 2025.

ORGANIZZE. Disponível em: https://www.organizze.com.br/. Acesso em: 25 ago. 2025.

MOBILLS. Disponível em: https://www.mobills.com.br/. Acesso em: 25 ago. 2025.

BROWN, Simon. C4 model for visualizing software architecture. Disponível em: https://c4model.com/. Acesso em: 22 set. 2025.