MeOrganiza: SISTEMA DE GERENCIAMENTO E CONTROLE FINANCEIRO PESSOAL

Jamil Luiz da Silva Campeão[1], Sidnei Renato Silveira[2]

[1]UFSM – Universidade Federal de Santa Maria – Campus Frederico Westphalen/RS, Frederico Westphalen/RS, Brasil (jamil.campeao@acad.ufsm.br)
[21]UFSM – Universidade Federal de Santa Maria – Campus Frederico Westphalen/RS, Frederico Westphalen/RS, Brasil (sidnei.silveira@ufsm.br)

Resumo: Este artigo apresenta o desenvolvimento do *MeOrganiza*, um protótipo de sistema *web* voltado ao gerenciamento e planejamento de finanças pessoais, com o objetivo de fornecer uma ferramenta prática para a educação financeira. O sistema foi desenvolvido utilizando uma arquitetura moderna baseada em *React* com *TypeScript*, *Node.js* e *PostgreSQL*, com infraestrutura conteinerizada via *Docker* para garantir portabilidade e segurança. Um diferencial significativo do projeto é a integração de agentes de Inteligência Artificial (utilizando *n8n* e *Google Gemini*) para a geração de relatórios personalizados e previsões de saldo futuro. A validação do protótipo combinou testes automatizados de software e testes de usabilidade com usuários reais. Os resultados obtidos demonstraram alta eficácia: o sistema alcançou uma média de 91,6 pontos na escala SUS (*System Usability Scale*), classificando-se como "Excelente" em termos de usabilidade. A análise qualitativa confirmou a aceitação das funcionalidades de IA e a superioridade da ferramenta frente a métodos manuais de controle, consolidando o protótipo como uma solução robusta para a melhoria da saúde financeira dos usuários.

Palavras-chave: Educação Financeira; Gestão de Finanças Pessoais; Sistemas *Web*; Inteligência Artificial; Planejamento Financeiro.

INTRODUÇÃO

Este artigo apresenta o desenvolvimento de um protótipo de sistema *web*, denominado *MeOrganiza*, para auxiliar as pessoas no acompanhamento e planejamento de suas finanças, já que a falta de controle financeiro ainda é um problema para a maioria da população brasileira (Lanza, 2024). Esperamos, com o desenvolvimento deste protótipo, disponibilizar uma ferramenta que permita a visualização detalhada do fluxo de caixa diário, categorização e discrimininação de contas futuras, além de fornecer previsões de saldo com o suporte de IA (Inteligência Artificial).

A motivação para a elaboração deste trabalho surgiu da necessidade pessoal de contar com um sistema eficiente de controle financeiro. Durante experiências realizadas com diferentes ferramentas disponíveis no mercado tais como *Organizze* (Organnize.com.br, 2026) e *Mobills* (Mobills.com.br, 2026), percebemos que, embora a maioria oferecesse funcionalidades básicas, como categorização de receitas e despesas, muitas apresentavam deficiências na geração de relatórios, automatização de lançamentos e previsões de fluxo de caixa. Além disso, diversos sistemas demonstram problemas de usabilidade, com interfaces pouco intuitivas e recorrência de *bugs*, comprometendo a experiência do usuário.

Além da necessidade pessoal identificada, a falta de controle financeiro é um problema generalizado no Brasil. De acordo com Lanza (2024), 47% dos brasileiros não conseguem organizar o próprio orçamento. Outros 59% responderam que não sabem como fazer e 26% afirmaram que já tentaram, mas desistiram.

Neste contexto, a essência da motivação para a realização deste trabalho reside na busca de um sistema de controle financeiro que não apenas atenda às necessidades pessoais, mas que, também, contribua para a autonomia financeira dos usuários. Este projeto visa oferecer uma abordagem mais intuitiva e confiável aos usuários.

SOLUÇÃO IMPLEMENTADA: *MeOrganiza*

A solução implementada compreende o desenvolvimento de um sistema *web* de controle financeiro pessoal - denominado *MeOrganiza*, com foco em segurança e usabilidade, utilizando tecnologias modernas e escaláveis para proporcionar uma boa experiência aos usuários. O sistema foi

desenvolvido para uso em desktops, com foco na funcionalidade e desempenho do sistema.

A metodologia de pesquisa aplicada foi a Dissertação-Projeto (Ribeiro; Zabadal, 2010), pois o objetivo foi o de desenvolver um protótipo funcional do sistema, possibilitando a validação dos requisitos e funcionalidades antes de uma implementação em larga escala.

Para representar a arquitetura da solução e como as tecnologias interagem no protótipo, elaboramos um esquema gráfico (Figura 1). O esquema foi construído com o *software Draw.io* (Draw.io, 2023).

O desenvolvimento do *MeOrganiza* baseou-se em uma *stack* moderna e amplamente utilizada no mercado, priorizando desempenho, segurança e facilidade de manutenção. As principais tecnologias empregadas foram:

● *Node.js* com *Express:* Utilizado no *backend* para gerenciar as regras de negócio e a comunicação via API (*Application Program Interface*);

● *React* com *TypeScript* e *Vite*: Empregado no *frontend* para criar uma interface reativa. O uso do *TypeScript* adiciona tipagem estática, aumentando a segurança do código, enquanto o *Vite* otimiza a construção (*build*) da aplicação;

● *PostgreSQL*: SGBD (Sistema Gerenciador de Banco de Dados) relacional escolhido para a persistência segura dos dados financeiros;

● Prisma ORM: Ferramenta utilizada para facilitar a interação entre o *backend* e o banco de dados, garantindo consultas seguras;

● *Docker*: Plataforma de conteinerização utilizada para empacotar a aplicação e suas dependências, garantindo que o sistema funcione de forma padronizada em diferentes ambientes (desenvolvimento e produção);

● *Cloudflare*: Utilizado para gerenciamento de DNS (*Domain Name System*) e como camada de borda (*edge*), provendo certificação SSL (*Secure Sockets Layer*) e proteção contra ataques;

● *n8n*: Ferramenta de automação de fluxo de trabalho utilizada para orquestrar a integração com os agentes de IA (N8N, 2026);

● JWT (JSON *Web Token)* e *bcrypt*: Tecnologias de segurança para autenticação via *token* e criptografia de dados sensíveis, respectivamente (NPM, 2023).

Na Figura 1 apresentamos o esquema gráfico da interação entre estas tecnologias.

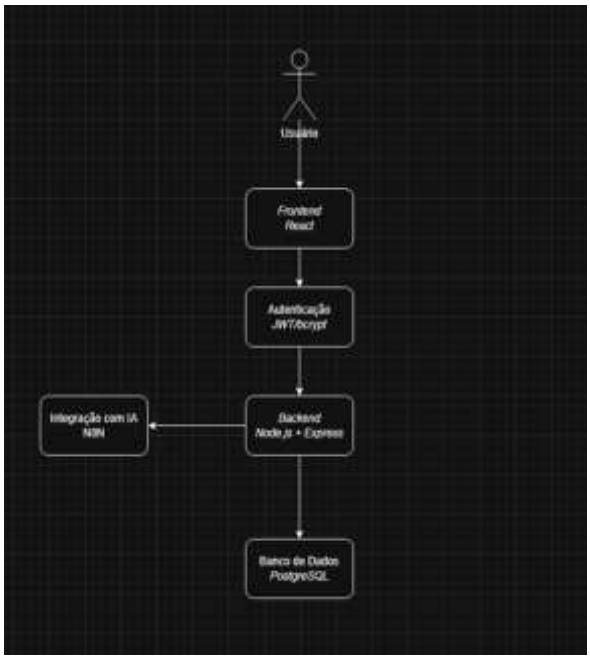

Figura 1 - Esquema gráfico de interação entre as tecnologias. Fonte: Os autores (2026)

De acordo com a Figura 1, a arquitetura de interação entre as tecnologias ocorre da seguinte forma:

● Usuário e Camada de Acesso (*Cloudflare*): O acesso inicial do usuário ao sistema é mediado pela *Cloudflare*. Esta camada gerencia o DNS e estabelece uma conexão segura (HTTPS *HyperText Transfer Protocol Secure*) via certificado SSL, protegendo o tráfego de dados entre o navegador do usuário e o servidor;

● Ambiente Conteinerizado (*Docker*): Toda a aplicação (*frontend*, *backend*, banco de dados e serviços auxiliares) é executada dentro de contêineres *Docker*, hospedados em um Servidor Privado Virtual (VPS *Virtual Private Server*). Isso isola a aplicação do sistema operacional do servidor, garantindo consistência e facilidade de implantação;

● *Frontend* (*React*): Após a camada de segurança, o usuário interage com a interface desenvolvida em *React*. O *frontend* é responsável pela experiência visual e captura das ações do usuário, comunicando-se com o servidor por meio de requisições HTTP (*HyperText Transfer Protocol*);

● *Backend* (*Node.js*) e Autenticação: O *backend* processa as requisições recebidas. Antes de executar qualquer operação sensível, o sistema valida a identidade do usuário por meio do *middleware* de autenticação, que verifica a validade do token JWT. O *bcrypt* atua na criptografia das senhas antes de sua persistência;

● Interação com Dados (Prisma e *PostgreSQL*): Para armazenar ou recuperar informações (como

transações e cadastros), o *backend* utiliza o Prisma ORM. Esta ferramenta traduz as solicitações do código para consultas SQL otimizadas, que são executadas no banco de dados *PostgreSQL*;

● Integração com IA (*n8n*): Funcionalidades específicas que requerem IA (como o assistente financeiro) acionam o serviço do *n8n*. Este serviço atua como um orquestrador, recebendo dados do *backend*, comunicando-se com os modelos de IA (LLMs[1]) e retornando a resposta processada para o sistema.

MeOrganiza: FUNCIONALIDADES IMPLEMENTADAS

Nesta seção, apresentamos algumas funcionalidades do protótipo *MeOrganiza*, mostrando o fluxo de interação do usuário e os principais aspectos técnicos da implementação. O sistema foi desenvolvido com uma arquitetura de aplicação de página única (SPA *Single Page Application*[2]) (Oliveira, 2017) utilizando *React* (com *Vite*) para o *frontend*,e a linguagem *TypeScript*. A escolha pelo *TypeScript*, em detrimento do *JavaScript* puro, visou garantir a tipagem estática do código, aumentando a segurança durante o desenvolvimento e facilitando a manutenção da base de código.

No *backend*, optamos por uma API *RESTful* construída com *Node.js* e *Express*, utilizando o Prisma como ORM (*Object-Relational Mapping*) para a interação com o banco de dados *PostgreSQL*. Essa camada utiliza autenticação via JWT (JSON *Web Token*) e criptografia *bcrypt*, assegurando a proteção dos dados sensíveis dos usuários. A integração com os agentes de IA (*Google Gemini*) foi desacoplada da API principal, sendo orquestrada pela ferramenta de automação de fluxo de trabalho *n8n*, o que permitiu uma arquitetura modular e de fácil expansão.

Quanto à infraestrutura, o sistema foi implantado em um servidor privado virtual (VPS) na nuvem. Para garantir a portabilidade e a padronização do ambiente entre desenvolvimento e produção, todos os serviços — *frontend*, *backend*, banco de dados e *n8n* — foram conteinerizados utilizando *Docker*. O gerenciamento de DNS (*Domain Name System*), bem como a segurança da camada de transporte, é realizado por meio da *Cloudflare*, que provê os certificados SSL (*Secure Sockets Layer*) e gerencia os subdomínios, garantindo uma conexão segura (HTTPS) e performática para o usuário final.

O primeiro contato do usuário com o sistema é por meio das telas de *Login* e Registro. A autenticação é gerenciada via API, que utiliza JWT para criar sessões seguras. Após o *login*, o *token* do usuário é armazenado no *frontend* e enviado no cabeçalho de autorização de requisições subsequentes, permitindo que o *backend* verifique a identidade do usuário e retorne apenas os dados que lhe pertencem.

Após autenticar-se, o usuário é direcionado ao *Dashboard* (Figura 2). Esta tela serve como o ponto central da aplicação, fornecendo uma visão geral e imediata da saúde financeira do usuário. O painel exibe:

● Saldo Total: agregação dos saldos de todas as contas cadastradas;

● Gráficos de Despesas e Receitas: visualizações (gráficos de barra, *pizza*, linha, tabela ou texto puro) que sumarizam as transações por categoria ou período;

● Transações Recentes: uma lista das últimas transações registradas;

● Alertas de Contas: lembretes de contas a pagar.

Figura 2 - *Dashboard*. Fonte: Os autores (2026)

O usuário pode registrar todas as suas movimentações financeiras (receitas e despesas). O formulário de transações (Figura 3) permite ao usuário inserir:

● Descrição da transação;

● Valor (diferenciando entrada ou saída);

● Data da transação;

● A Conta de origem/destino (ex: "Carteira", "Conta Corrente").

● A Categoria (ex: "Alimentação", "Salário").

[1] LLM é a sigla para *Large Language Model* (Modelo de Linguagem Grande), um tipo de programa de inteligência artificial (IA) treinado para processar e compreender a linguagem natural. São modelos de aprendizado profundo, um subconjunto do machine learning, que utilizam redes neurais avançadas e são treinados em volumes de dados de texto massivos, muitas vezes abrangendo grande parte da internet (Gran Faculdade, 2024).

[2] A arquitetura SP é um padrão de desenvolvimento web em que a aplicação roda em uma única página HTML, atualizando o conteúdo dinamicamente por meio de código desenvolvido em linguagem de programação *JavaScript*, sem que o usuário precise recarregar a página inteira a cada navegação (Guedes, 2019).

Essa funcionalidade alimenta diretamente todos os outros módulos de relatórios e previsões.

Figura 3 - Formulário de Transações. Fonte: Os autores (2026)

Para permitir um controle detalhado, o sistema modulariza as fontes financeiras, dividindo-as em:

● Contas: permite ao usuário cadastrar suas contas bancárias, carteiras digitais ou saldo em espécie. Cada conta possui um saldo independente que é atualizado automaticamente com base nas transações vinculadas;

● Cartões de Crédito: o usuário cadastra seus cartões, definindo limite e data de fechamento. As transações feitas no "crédito" são lançadas em uma fatura que, ao ser paga, gera uma transação de débito na conta selecionada;

Com base nos dados inseridos, o sistema oferece módulos avançados de análise e assistência, que utilizam IA para fornecer insights ao usuário, indo além dos relatórios tradicionais.

● Relatórios (Tradicionais): este módulo permite ao usuário gerar relatórios financeiros convencionais, como mostra a Figura 4. É possível filtrar transações por período, categoria (ex: "Alimentação") ou conta (ex: "Conta Corrente"), obtendo gráficos e somatórios que ajudam a entender de forma clara para onde o dinheiro está indo;

Figura 4 - Relatórios. Fonte: Os autores (2026)

● Relatórios com IA: esta é uma funcionalidade avançada. Ao invés de apenas mostrar os dados, o usuário pode solicitar que a IA monte um relatório ou gráfico, de acordo com a solicitação/pergunta do usuário. Funciona da seguinte forma: O usuário escreve sua pergunta/solicitação, como por exemplo: "Gere um gráfico de barras com as 5 maiores despesas que tive no mês de Outubro de 2025". O sistema então encaminha essa solicitação para o *backend*, onde é executada sua lógica e devolve o relatório solicitado. A Figura 5 mostra a tela de relatórios com IA;

Figura 5 - Relatórios com IA Fonte: Os autores (2026)

● Previsão de Saldo (com IA): A funcionalidade de Previsão de Saldo (Figura 6) aplica um modelo de projeção para estimar o saldo futuro do usuário. O *backend* analisa o histórico de transações, a recorrência de contas a pagar cadastradas e a média de gastos variáveis para alimentar um modelo de *forecast*. O resultado é um gráfico que demonstra a tendência do saldo, permitindo ao usuário se antecipar a possíveis meses de aperto ou planejar investimentos com maior segurança;

Figura 6 - Previsão de Saldo com IA Fonte: Os autores (2026)

● Assistente Financeiro (*Chat*): É o recurso de IA mais interativo do protótipo. Funciona como um *chatbot* especializado nas finanças do usuário, como mostra a Figura 7. O usuário pode fazer perguntas em linguagem natural (ex: "Quanto gastei com restaurantes este mês?" ou "Qual foi minha maior despesa em outubro?"). O *backend* interpreta a pergunta, busca os dados específicos no banco de dados e utiliza a IA generativa para formular uma resposta precisa e contextualizada, facilitando o acesso à informação sem a necessidade de navegar por telas de relatórios.

Figura 7 - Assistente de IA Fonte: Os autores (2026)

A IA do protótipo está implementada em três funcionalidades centrais: Relatórios com IA, Previsão de Saldo com IA e o Assistente IA. O modelo de linguagem generativa (LLM) empregado em todos os agentes é o *Gemini*, da *Google*.

Para esta solução, optamos por uma arquitetura de micro-serviços, desacoplando a lógica de IA do *backend* principal da aplicação (*Node.js*). A implementação utiliza a plataforma de automação *n8n* para orquestrar os fluxos de IA.

Nesta arquitetura, o *backend* do *MeOrganiza* se comunica com o *n8n* por meio de requisições a *endpoints* de *webhook* dedicados para cada funcionalidade. O *n8n*, por sua vez, executa um fluxo de nós que envolve agentes de IA , ferramentas de consulta ao banco de dados (*PostgreSQL*) e lógicas de roteamento.

A geração de relatórios inteligentes é orquestrada por um fluxo de automação no *n8n*, acionado pelo *backend* quando o usuário realiza uma solicitação. A lógica de processamento pode ser sintetizada em três etapas principais:

1. Triagem e Classificação: Ao receber a pergunta do usuário, o fluxo utiliza o modelo *Gemini* para interpretar a intenção e determinar qual a melhor visualização para os dados (por exemplo: gráfico de barras, gráfico de pizza, tabela ou apenas texto);

2. Interpretação e Busca de Dados: Um agente de IA analisa a solicitação em linguagem natural e a traduz em instruções precisas para uma ferramenta de banco de dados, que executa a consulta e retorna o histórico financeiro relevante;

3. Formatação e Resposta: Com os dados brutos recuperados, o fluxo direciona a informação para um nó de formatação específico (baseado na classificação da etapa 1. A IA estrutura a resposta final em um formato JSON padronizado, que é então enviado ao *frontend* para a renderização visual do relatório.

A funcionalidade de previsão de saldo é executada por um fluxo de automação específico, acionado pelo servidor sempre que uma nova análise é solicitada. O processo de inteligência preditiva ocorre em três etapas fundamentais:

1. Coleta de Contexto Financeiro: Um agente especializado inicia o processo invocando a ferramenta de banco de dados para recuperar o histórico completo de transações do usuário, garantindo que a análise seja baseada em dados reais de receitas e despesas;
2. Análise de Padrões e Sazonalidade: O modelo generativo examina o histórico recuperado para identificar comportamentos de consumo recorrentes. O diferencial desta etapa é a detecção de sazonalidades, onde a IA pondera despesas atípicas de períodos específicos (tais como Natal ou impostos anuais) para calcular uma estimativa de saldo mais precisa para o mês seguinte;

3. Estruturação da Resposta: Para garantir a integridade da interface, a saída da IA (tanto o resumo textual da análise quanto o valor numérico previsto) é forçada a seguir um esquema JSON antes de ser enviada de volta ao *frontend*.

O Assistente IA, apresentado na Figura 7, constitui a interface conversacional do sistema. Diferentemente dos módulos anteriores que executam tarefas pontuais, este componente foi arquitetado como um sistema multiagentes. O fluxo de automação gerencia a interação por meio de quatro etapas lógicas:

1. Gestão de Contexto e Roteamento: Ao receber uma mensagem, o sistema verifica se ela pertence a uma sessão ativa. Para novas interações, um modelo classificador analisa a intenção do usuário e define o tópico da conversa (Dívidas, Investimentos ou Dúvidas Gerais). Para conversas em andamento, o

sistema preserva a continuidade, mantendo o usuário conectado ao especialista definido anteriormente;

2. Atuação dos Agentes Especialistas: Com base no roteamento, a solicitação é processada por um de três agentes independentes, cada um configurado com diretrizes específicas:

○ Assistente Financeiro (Generalista): Responsável por questões operacionais, como consulta de saldo, análise de gastos passados e dúvidas sobre o orçamento;

○ Especialista em Dívidas: Configurado com uma persona empática e objetiva, focado em explicar estratégias de quitação (tais como os métodos Avalanche e Bola de Neve[3]) e analisar a capacidade de pagamento do usuário;

○ Especialista em Investimentos: Atua com um perfil estritamente educacional. É instruído a analisar o perfil de risco e objetivos do usuário, explicando classes de ativos sem fornecer recomendações de compra ou venda (consultoria regulamentada);

3. Isolamento de Memória: Para garantir a consistência, cada agente opera de forma isolada, possuindo sua própria memória de conversação e acesso a ferramentas de banco de dados específicas. Isso impede, por exemplo, que o contexto de uma negociação de dívidas interfira em uma orientação sobre investimentos;

4. Consolidação da Resposta: A resposta gerada pelo agente ativo é capturada e padronizada pelo sistema antes de ser enviada de volta à interface de chat, garantindo uma experiência fluida para o usuário.

TESTES E VALIDAÇÃO DO PROTÓTIPO

A estratégia de garantia de qualidade do *MeOrganiza* foi estruturada em duas etapas complementares: a verificação técnica, realizada por meio de testes automatizados de software, e a validação de uso, realizada por meio de testes de usabilidade com usuários finais. Para assegurar a confiabilidade e a estabilidade do sistema, foram implementadas rotinas de testes automatizados tanto no *backend* quanto no *frontend*, utilizando ferramentas modernas e padrões de mercado.

No âmbito do *Backend* (API), optou-se pela implementação de testes de integração, com o objetivo de validar o fluxo completo das requisições HTTP, desde a entrada nos controladores (*Controllers*) até a persistência no banco de dados. Para isso, foram utilizadas as ferramentas *Jest* como *framework* de execução e *Supertest* para a simulação de requisições programáticas. O ambiente de testes foi configurado com um banco de dados dedicado gerenciado pelo ORM Prisma (Prisma.io, 2025), garantindo que os dados de teste não interferissem no ambiente de desenvolvimento.

Todos os cenários propostos foram executados com sucesso, confirmando a integridade das regras de negócio no servidor.

No *frontend*, a estratégia de qualidade focou em garantir a estabilidade da interface e a integridade dos componentes visuais. Foram adotadas as ferramentas *Vitest*, pela sua alta performance e integração nativa com o *build* do *Vite* (Vite.dev, 2025), e a RTL (*React Testing Library*) (Dodds, 2025), que permite testar os componentes focando no comportamento do usuário e na acessibilidade.

Após a verificação técnica das funcionalidades (testes de integração e unitários), foram realizados testes de usabilidade com foco na experiência do usuário (*User Experience* - UX). Estes testes tiveram como objetivo validar a eficácia das funcionalidades principais e garantir que o sistema atenda às necessidades de gestão financeira do público-alvo de forma intuitiva.

Para validar a usabilidade do protótipo, foram conduzidos testes com um grupo de 5 usuários voluntários, selecionados entre a comunidade acadêmica e pessoas interessadas no projeto. Segundo Nielsen (2000), 5 usuários são suficientes para verificar a usabilidade de um sistema. A metodologia aplicada consistiu em duas etapas: a execução de tarefas guiadas (baseadas em cenários de teste predefinidos) e a avaliação subjetiva por meio do questionário SUS (*System Usability Scale*).

Para a execução prática, foi elaborado um roteiro de testes contendo 13 cenários fundamentais que cobrem o ciclo principal de uso do sistema, desde o cadastro inicial até o uso das funcionalidades avançadas de IA. Os participantes foram instruídos a realizar as tarefas descritas no Quadro 1, sem auxílio prévio, para simular uma interação real.

Quadro 1 – Casos de Teste

ID do Cenário	Módulo/ Página	Tarefa do Usuário (*Script*)	Passos para Execução
TST-001	*Dashboard*	Analisar a situação financeira inicial	1. Ao logar, identifique seu Saldo Total. 2. Verifique o total de Receitas e Despesas do mês. 3. Localize o alerta de contas pendentes.

[3] O método da avalanche de dívidas concentra-se em pagar primeiro as dívidas com as taxas de juros mais altas para reduzir os custos totais com juros. Já o método da bola de neve para dívidas visa primeiro os saldos menores, para ganhar impulso por meio de conquistas iniciais (WESTERN & SOUTHERN FINANCIAL GROUP, 2025)

			4. Identifique sua maior despesa por categoria no gráfico.
TST-002	*Transações (CRUD - Create)*	Registrar gastos do dia-a-dia	1. Vá para 'Transações'. 2. Crie uma nova 'Despesa' (Ex: 'Almoço', R$ 35,00, Categoria 'Almoços fora de casa', 'Conta Corrente'). 3. Crie uma nova 'Receita' (Ex: 'Aula Particular', R$ 200,00, Categoria '*Freelancer*', 'Conta Corrente').
TST-003	Categorias (CRUD)	Organizar suas categorias pessoais	1. Vá para 'Categorias'. 2. Crie uma nova categoria de Despesa: 'Pets'. 3. Edite a categoria 'Outros Lazer' para 'Lazer e *Hobbies*'. 4. Tente excluir a categoria 'Telefone, Internet e TV' (que possui transações).
TST-004	Contas e Cartões (CRUD)	Adicionar uma nova conta e um novo cartão	1. Vá para 'Contas'. 2. Adicione uma 'Nova Conta' (Ex: 'Conta Investimento', Tipo 'Investimento', Saldo Inicial R$ 1.000). 3. Vá para 'Cartões'. 4. Adicione um 'Novo Cartão' de 'Crédito' (Ex: 'Cartão Reserva', Limite R$ 2.000, vinculado à 'Conta Corrente').
TST-005	*Transações (CRUD - Edit/Delete)*	Corrigir e remover lançamentos	1. Vá para 'Transações'. 2. Encontre a transação 'Almoço' (R$ 35,00) criada no TST-002. Edite o valor para R$ 38,50. 3. Encontre a transação 'Compra em Supermercado' e a exclua.
TST-006	Contas a Pagar	Pagar uma conta pendente	1. Vá para o Dashboard. 2. Localize a conta 'Conta de Luz' (status 'Pendente'). 3. Realize a ação de 'Pagar' esta conta.
TST-007	Dívidas (CRUD)	Registrar um novo financiamento	1. Vá para 'Dívidas'. 2. Adicione uma 'Nova Dívida' (Ex: 'Financiamento Moto', Tipo 'Financiamento de Veículo', Valor Inicial R$ 15.000, Saldo Devedor R$ 10.000).
TST-008	Investimentos (CRUD)	Registrar um novo investimento	1. Vá para 'Investimentos'. 2. Adicione um 'Novo Investimento' (Ex: 'PETR4', Tipo 'Ação', Quantidade 10, Valor R$ 1.600).
TST-009	Relatórios (Filtro)	Analisar gastos de um período específico	1. Vá para 'Relatórios'. 2. Use o filtro de data para selecionar o 'Mês Passado' (Outubro/2025, com base no seed). 3. Clique em 'Gerar Relatório'. 4. Verifique os gráficos de Receita x Despesa e Despesas por Categoria.
TST-010	Contas/Cartões (Inativação)	Inativar um cartão que não usa mais	1. Vá para 'Cartões'. 2. Encontre o 'Cartão Reserva' criado no TST-004. 3. Use a função de 'Inativar' (ícone de 'Power'). 4. Verifique se o cartão é exibido como inativo. 5. Ative o cartão novamente.

TST-011	Previsão de Saldo	Gerar uma previsão de saldo	1. Vá para 'Previsão com IA' 2. Clique em 'Gerar previsão'
TST-012	Relatórios IA	Gerar um relatório com IA	1. Vá para 'Relatórios I.A' 2. Digite uma pergunta, como por exemplo: 'Gere um gráfico de barras com as minhas 5 maiores despesas' 3. Realize o envio
TST-013	Assistente IA	Conversar com o assistente financeiro	1. Faça uma pergunta ou uma solicitação para o assistente financeiro, como por exemplo: 'Qual é o meu saldo total atual na minha conta', 'Lançe uma transação de despesa no valor de 20 reais na categoria Supermercado na conta Principal')

Fonte: Os autores (2026)

Durante as sessões, observou-se que os usuários conseguiram completar a maioria das tarefas sem erros críticos. As funcionalidades de IA (TST-012 e TST-013) destacaram-se como diferenciais, onde os usuários relataram surpresa positiva com a facilidade de gerar gráficos apenas descrevendo o que desejavam em texto.

Ao final da execução dos cenários, aplicou-se o questionário *System Usability Scale* (SUS), uma ferramenta padronizada para mensurar a usabilidade de sistemas. O questionário é composto por 10 afirmações com respostas em escala *Likert* (de 1 a 5).

Os resultados obtidos foram compilados e o score final foi calculado conforme a metodologia padrão do SUS (Homem Máquina, 2024), que gera uma pontuação de 0 a 100. O protótipo *MeOrganiza* atingiu uma média de 91,6 pontos. De acordo com a escala de classificação, sistemas com pontuação acima de 80 são considerados "Excelentes" em termos de usabilidade (Bangor; Kortum; Miller, 2008).

Este resultado valida a hipótese de que a interface desenvolvida em R*eact*, aliada aos componentes de UI modernos, proporcionou uma experiência fluida e intuitiva. A alta aceitação sugere que o sistema não apresenta barreiras significativas de entrada, permitindo que novos usuários compreendam e utilizem as funcionalidades de gestão financeira e inteligência artificial com eficiência.

Além da avaliação quantitativa via escala SUS, os participantes responderam a perguntas abertas opcionais ao final do teste, permitindo uma análise qualitativa mais aprofundada sobre a percepção de valor e as dificuldades de interação. As respostas foram categorizadas em três eixos principais:

1. Pontos Fortes e Aceitação: Os usuários destacaram a automatização e a abrangência das funcionalidades como diferenciais competitivos em relação a outras soluções de mercado. O módulo de cartão de crédito e a fluidez das transações foram citados nominalmente como aspectos positivos. Um dos participantes enfatizou a viabilidade do sistema para uso contínuo, declarando: "Gostei de tudo no geral, usaria no meu dia a dia", o que corrobora a utilidade prática do protótipo;

2. Dificuldades de Interação (Usabilidade): Foi identificada uma falha de usabilidade crítica nos formulários de edição. Os participantes relataram que "os campos de edição não ficam pré-preenchidos com as informações atuais", obrigando o usuário a registrar novamente todos os dados apenas para alterar um campo específico. Além disso, questões de interface (UI), como a identificação de botões clicáveis e o refinamento do layout, foram apontadas como barreiras menores que impactam a fluidez da navegação;

3. Sugestões de Melhoria e Funcionalidades Ausentes: A ausência de uma barra de pesquisa global foi a lacuna mais citada, com os usuários sentindo falta de um mecanismo para localizar transações antigas rapidamente sem a necessidade de filtros manuais. Outra sugestão relevante foi a inclusão de um módulo específico para lançar rendimentos de investimentos, demonstrando que o público-alvo demanda recursos avançados de gestão patrimonial.

CONSIDERAÇÕES FINAIS

Acreditamos que os objetivos definidos para este trabalho foram majoritariamente alcançados. O objetivo principal era o de desenvolver um protótipo de sistema web para auxiliar no acompanhamento e planejamento de finanças pessoais. Este objetivo foi atingido por meio da implementação de um sistema funcional que permite ao usuário registrar transações financeiras; categorizar receitas e despesas; gerar relatórios financeiros e utilizar ferramentas que auxiliam na previsão de saldo futuro. Os objetivos específicos, tais como contribuir para a educação financeira e proporcionar ao usuário uma visão clara de seus gastos, são consequências diretas do uso da ferramenta. Ao centralizar e visualizar seus dados financeiros de forma intuitiva, o usuário é capacitado

a tomar decisões mais informadas sobre seus gastos e investimentos.

O desenvolvimento de um sistema *full-stack* como o *MeOrganiza*, mesmo em nível de protótipo, apresentou desafios significativos. As principais dificuldades foram:

● Modelagem de Dados Financeiros: definir a arquitetura do banco de dados foi um desafio, especialmente em relação à lógica de faturas de cartão de crédito. Garantir que o pagamento de uma fatura refletisse corretamente como um débito na conta e quitasse os lançamentos do cartão exigiu uma modelagem relacional cuidadosa;

● Gerenciamento de Estado no *frontend*: A natureza reativa do sistema, onde uma transação no *Dashboard* deve atualizar o saldo da Conta e o gráfico de Relatórios simultaneamente, tornou o gerenciamento de estado complexo. A utilização de ferramentas tais como *React Query* foi essencial para manter a consistência dos dados na interface sem sobrecarregar a API com requisições desnecessárias;

● Configuração do Ambiente de Produção: Embora o foco tenha sido o protótipo, a configuração do ambiente de *deploy* com *Docker* e o servidor *web Nginx* para servir o *frontend React* e redirecionar para a API *Node.js* exigiu um esforço considerável de infraestrutura (*DevOps*).

O *MeOrganiza* foi implementado como um protótipo, o que abre um vasto leque de possibilidades para trabalhos futuros e melhorias, tais como:

● Integração com *Open Banking*: A melhoria mais significativa seria a conexão com APIs de bancos reais (por meio do *Open Banking* Brasil) para importar automaticamente extratos bancários e faturas de cartão, eliminando a necessidade de lançamento manual;

● Aplicativo Móvel: O desenvolvimento de uma versão móvel (utilizando *React Native* ou *Flutter*) para facilitar o registro de transações no dia-a-dia;

● Módulo de Orçamento: Implementação de uma funcionalidade onde o usuário possa definir metas de gastos por categoria (ex: "R$ 500,00 para Alimentação") e o sistema o alerte quando estiver próximo de atingir o limite.

REFERÊNCIAS

Bangor, A.; Kortum, P.; Miller, J. A. The System Usability Scale (SUS): An Empirical Evaluation, International Journal of Human-Computer Interaction, 24(6), 2008.

Dodds, K C. React Testing Library. Disponível em: https://testing-library.com/docs/react-testing-library/intro/. Acesso em 28 nov. 2025.

Draw.io. Security-first diagramming for teams. 2023. Disponível em: https://www.drawio.com/. Acesso em: 03 maio 2025.

Gran Faculdade. Large Language Model (LLM) em TI? saiba o que é! 2024. Disponível em: https://faculdade.grancursosonline.com.br/blog/llm-large-language-model/. Acesso em: 27 nov 2025.

Guedes, M. O que são aplicações SPA? 2019. Disponível em: https://www.treinaweb.com.br/blog/o-que-sao-aplicacoes-spa. Acesso em: 15 nov. 2025.

Homem Máquina. System Usability Scale (SUS) e os indicadores de usabilidade. 2024. Disponível em: https://www.homemmaquina.com.br/system-usability-scale-sus-e-os-indicadores-de-usabilidade/. Acesso em: 26 maio 2025.

Lanza, L. Educação Financeira: 59% dos brasileiros não sabe como organizar o orçamento. 2024. Disponível em: https://einvestidor.estadao.com.br/educacao-financeira/educacao-financeira-pesquisa-onze-orcamento/. Acesso em: 15 mar. 2025.

Mobills.com.br. Mobills: finanças e cartões. Disponível em: https://www.mobills.com.br/. Acesso em: 24 mar. 2025.

Nielsen, J.. Why You Only Need to Test with 5 Users. 2000. Disponível em: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/. Acesso em: 01 dez. 2025.

NPM. Bcrypt. 2023. Disponível em: https://www.npmjs.com/package/bcrypt. Acesso em: 10 maio. 2025.

N8N. N8N. Disponível em: https://n8n.io/. Acesso em: 10 mar. 2026.

Oliveira, D. J. Uma proposta de arquitetura para Single-Page Applications. Recife/PE: Universidade Federal de Pernambuco, 2017. Trabalho de Graduação em Ciência da Computação. Disponível em: https://www.cin.ufpe.br/~tg/2017-2/djo-tg.pdf. Acesso em: 12 nov. 2025.

Organizze.com.br. Organizze: controle financeiro pessoal. Disponível em: https://www.organizze.com.br/. Acesso em: 24 mar. 2025.

Prisma.io. Prisma. Disponível em: https://www.prisma.io/. Acesso em: 10 maio. 2025.

Ribeiro, V. G.; Zabadal, J. R. Pesquisa em Computação. Porto Alegre: UniRitter, 2010.

Sá, H. Teste A/B: O que é e como fazer?. 2023. Disponível em: https://www.salesforce.com/br/blog/teste-ab/. Acesso em: 22 jun. 2025.

Vite.dev. Vite: the unified toolchain for the web. Disponível em: https://vite.dev/guide/build. Acesso em 28 nov. 2025.

Western & Southern Financial Group. Debt Avalanche vs. Debt Snowball: Which Method Fits Your Strategy? Disponível em: https://www.westernsouthern.com/personal-finance/debt-avalanche-vs-debt-snowball. Acesso em: 20 nov. 2025.