SAFA: UM SISTEMA DE FIDELIZAÇÃO DE CLIENTES PARA LOJAS AGRÍCOLAS

Bianca Karoline Ramos[12]
Giovanna Cerino de Oliveira Mendonça[13]

Resumo

O objetivo deste projeto é desenvolver um sistema de fidelização de clientes para lojas agrícolas, denominado Sistema de Atendimento e Fidelização Agro (SAFA). A proposta visa engajar consumidores e oferecer uma experiência de compra personalizada com base em dados de perfil e histórico de compras. O desenvolvimento justifica-se pela necessidade de modernização do comércio agrícola, dada sua relevância social e econômica. Utilizando a empresa Trevo Peças Agrícolas como estudo de caso, aplicaram-se metodologias de engenharia de *software*, como diversos diagramas, para a construção do sistema. A solução contribui para a melhoria do gerenciamento de clientes, produtos e compras, sendo uma referência para outras empresas que busquem estratégias de fidelização.

Palavras-chave: Agrícola. Clientes. Fidelização. Gestão. Vendas.

Abstract

This project aims to develop a customer loyalty system for agricultural stores, called the Agricultural Service and Loyalty System (SAFA). The proposal seeks to engage consumers and provide a personalized shopping experience based on profile and purchase history data. The development is justified by the need to modernize agricultural commerce, given it is social and economic relevance. Using Trevo Agricultural Parts as a case study, software engineering methodologies, including various diagrams, were applied to build the system. The solution enhances the management of customers, products, and purchases, serving as a reference for other companies seeking loyalty strategies.

[12] Graduanda em Análise e Desenvolvimento de Sistemas pela Fatec Dr. Thomaz Novelino – Franca/SP. Endereço eletrônico: biancakaroliner04@gmail.com.
[13] Graduanda em Análise e Desenvolvimento de Sistemas pela Fatec Dr. Thomaz Novelino – Franca/SP. Endereço eletrônico: giovannacerino.mend@gmail.com.

Keywords: *Agricultural. Customers. Loyalty. Management. Sales.*

Introdução

O presente projeto aborda o desenvolvimento de uma solução digital voltada para fortalecer o relacionamento entre lojas agrícolas e seus clientes. O objetivo principal é criar o Sistema de Atendimento e Fidelização Agro (SAFA), projetado para atender às demandas específicas desse nicho de mercado, promovendo uma experiência de compra personalizada e eficiente.

Para o desenvolvimento, foram realizados levantamentos de requisitos e análises com base em estudos realizados na empresa Trevo Peças Agrícolas, atuante no ramo, que autorizou o uso do nome neste projeto. A colaboração da empresa foi essencial para compreender as particularidades do setor, servindo como base para a definição da estrutura e funcionalidades do sistema. Assim, todas as decisões tomadas neste projeto foram alinhadas com as necessidades e desafios enfrentados pela loja em seu cotidiano.

A questão central explorada neste trabalho é como as técnicas e métodos da engenharia de *software*, bem como as tecnologias de desenvolvimento *web*, podem ser aplicadas para criar e implementar um sistema digital de fidelização de clientes, capaz de atender às diferentes preferências e necessidades de compra de uma loja agrícola, garantindo uma experiência eficiente e interativa no ambiente online.

A relevância social do projeto está relacionada ao papel estratégico no suporte aos agricultores e na produção de alimentos. Um sistema de fidelização eficaz contribui para a eficiência do fornecimento de produtos, gerando impacto positivo na segurança alimentar e no bem-estar social.

Do ponto de vista técnico, o projeto busca adotar abordagens baseadas na análise de dados e no uso de *software* de gestão. A solução proposta beneficia não apenas as lojas, mas também os agricultores, ao oferecer ofertas personalizadas que potencializam sua produção e fortalecem o setor.

O objetivo geral é desenvolver um sistema com funcionalidades que permitam categorizar clientes com base em seus padrões de compra, analisar históricos de compras, identificar grupos de clientes com demandas semelhantes e

medir a eficácia das estratégias de fidelização. Busca-se, assim, implementar um programa de fidelização eficiente, segmentando diferentes perfis, otimizando a experiência de compra e aumentando a retenção de clientes.

O projeto abordado será destinado ao uso interno da empresa, abrangendo funcionalidades para o gerenciamento de clientes, funcionários, administradores, produtos, vendas e devoluções. Administradores terão acesso exclusivo ao controle de funcionários e à gestão completa do sistema, enquanto funcionários poderão gerenciar clientes e produtos, registrar vendas e devoluções, e acessar históricos de compras. Já os clientes terão acesso restrito apenas as informações como históricos de compras, devoluções, pontuações acumuladas e produtos em promoção personalizada. O sistema também realizará a segmentação dos clientes, sugerindo produtos de interesse e viabilizando o uso de pontos acumulados em compras futuras.

Durante o planejamento e desenvolvimento, diversas ferramentas e técnicas foram empregadas para garantir a eficiência do projeto. O Termo de Abertura do Projeto (TAP) formalizou o escopo e os objetivos, enquanto o *Business Model Canvas* (BMC) estruturou o modelo de negócios. A análise SWOT identificou forças, fraquezas, oportunidades e ameaças, e o plano de ação 5W2H organizou as atividades de forma precisa. A elicitação e especificação de requisitos asseguraram que o sistema atendesse aos objetivos estabelecidos, enquanto diagramas como BPMN, Diagrama de Classes e Diagrama de Atividades auxiliaram no entendimento da estrutura e funcionalidades do sistema. Para uma visualização completa dos processos, análises e diagramas citados aqui, consulte o material de Ramos (2024).

Para o desenvolvimento, foram utilizadas tecnologias como Figma, para a criação e validação inicial do protótipo de interface; Node.js, para o *back-end*; e HTML, CSS e JavaScript, para o *front-end*. O Supabase foi adotado como *backend-as-a-service* (BaaS), integrando e gerenciando o banco de dados PostgreSQL, com suporte do Prisma para a manipulação de dados. O controle de versão foi realizado com Git e GitHub, garantindo a integridade do código durante o processo de desenvolvimento.

Este projeto pretende contribuir significativamente para a gestão de relacionamento com clientes no setor atuante, oferecendo uma solução prática e eficiente para a fidelização de clientes em lojas agrícolas.

Referencial Teórico e Trabalhos Correlatos

Referencial Teórico

A fidelização de clientes é uma estratégia amplamente utilizada em diversos setores, com o objetivo de estabelecer um relacionamento contínuo e vantajoso entre empresas e consumidores. De acordo com Larman (2005), sistemas de fidelização têm como propósito principal a retenção de clientes por meio de incentivos, como descontos, ofertas personalizadas e o acúmulo de pontos. Essa abordagem é essencial para organizações que desejam manter a competitividade em mercados saturados, promovendo, além da retenção, a satisfação e lealdade dos clientes.

No setor agrícola, contudo, poucas soluções são desenvolvidas considerando as especificidades desse nicho, como a sazonalidade das vendas e o perfil singular dos consumidores. Wiegers e Beatty (2013) ressaltam que a engenharia de requisitos desempenha um papel crucial nesse contexto, destacando que o levantamento detalhado das necessidades dos clientes e das demandas do mercado é fundamental para o sucesso de sistemas digitais. Essa etapa garante que as soluções propostas estejam alinhadas às características e aos desafios particulares do setor.

Com base nesse cenário, o SAFA foi projetado como uma solução personalizada para atender de forma eficaz às necessidades específicas do nicho. O sistema incorpora funcionalidades como acúmulo de pontos, segmentação de produtos e ofertas personalizadas com base em compras anteriores. Ao adotar conceitos tradicionais de fidelização, adaptados às demandas do contexto agrícola, o projeto em questão busca oferecer maior eficiência e valor agregado para as empresas desse segmento.

Trabalhos Correlatos

Embora o desenvolvimento do SAFA tenha sido fundamentado em requisitos específicos levantados junto à Trevo Peças Agrícolas, não foi identificado nenhum sistema de fidelização existente que atenda, de forma completa, às necessidades do setor agrícola. No entanto, algumas soluções disponíveis no

mercado para o varejo em geral, como plataformas de CRM (*Customer Relationship Management*) com módulos de fidelização, foram analisadas para compreender as limitações e funcionalidades dessas ferramentas.

As plataformas de CRM, como destacam Hazzan e Dubinsky (2021), fornecem uma base relevante para o relacionamento com o cliente, permitindo a gestão de informações como histórico de compras e preferências. Todavia, essas ferramentas são desenvolvidas de maneira genérica, abrangendo uma ampla gama de setores, sem considerar as particularidades do setor agrícola. Esse é um diferencial essencial do sistema proposto: enquanto as soluções comerciais não segmentam ofertas nem levam em conta sazonalidades específicas, o SAFA foi projetado para atender essas questões, oferecendo produtos personalizados e promoções segmentadas com base no perfil dos clientes.

Além disso, ao avaliar estudos acadêmicos relacionados, observou-se que a maioria dos trabalhos sobre sistemas de fidelização, como o estudo de Leite e T.L.K.C.R.T. (2022), concentra-se em setores como o varejo tradicional ou o comércio eletrônico, sem abordar diretamente as demandas do nicho trabalhado nesse projeto. Essa lacuna no cenário acadêmico reforça a originalidade e a relevância do SAFA, que adapta os princípios gerais de fidelização de clientes a um contexto específico e ainda pouco explorado.

Outro aspecto importante é o uso de tecnologias modernas e práticas de desenvolvimento ágil. O projeto foi inspirado em técnicas consagradas no desenvolvimento de *software*, como a utilização de Node.js e Axios para a comunicação entre *front-end* e *back-end*, ferramentas amplamente discutidas por Tilkov e Vinoski (2010). Esses recursos tecnológicos asseguram que o sistema seja escalável e eficiente, agregando ainda mais vantagens ao público-alvo.

Dessa forma, o projeto posiciona-se como uma solução inovadora e única no mercado, ao atender uma demanda específica não contemplada plenamente por outras soluções, seja no meio acadêmico ou no mercado de *softwares* comerciais. Com foco na fidelização de clientes, o sistema apresenta uma abordagem personalizada e eficiente, preenchendo uma lacuna existente nas ferramentas de gestão de clientes disponíveis para esse nicho.

Desenvolvimento

Levantamento de Requisitos

O levantamento de requisitos foi uma etapa crítica no desenvolvimento do SAFA, pois estabeleceu as bases para a construção de um sistema que atendesse às necessidades específicas do nicho. A fim de garantir que o sistema fosse projetado de forma alinhada às expectativas dos usuários, a elicitação de requisitos foi realizada por meio de entrevistas com os *stakeholders* da loja. Essa abordagem possibilitou a coleta de informações detalhadas sobre as operações atuais e as funcionalidades desejadas no novo sistema.

Foram empregadas diversas técnicas para a elicitação de requisitos, como entrevistas com gestores e funcionários, análise de documentos existentes e observação direta das operações diárias. As informações coletadas foram organizadas em um documento de requisitos estruturado, no qual cada item foi descrito de maneira clara e específica. A categorização dos requisitos incluiu aspectos funcionais e não funcionais.

Os requisitos funcionais contemplaram diversas operações, como cadastro e gestão de clientes, permitindo aos funcionários realizar ações como cadastro, edição, exclusão e consulta de informações. Outras funcionalidades incluíram a gestão de produtos, além de processos relacionados ao registro de vendas e devoluções. Por outro lado, os requisitos não funcionais abordaram questões como segurança, usabilidade e desempenho, garantindo proteção dos dados dos clientes, uma interface amigável e um funcionamento eficiente mesmo durante picos de uso.

Além disso, foram definidas regras de negócio que estabelecem condições e restrições para o funcionamento do sistema. Um exemplo é a validação de cadastros, onde os dados precisam atender a critérios específicos para serem aprovados. Também foi definida a política de descontos, estabelecendo como e quando esses benefícios são aplicáveis (Hazzan; Dubinsky, 2021).

A elaboração de casos de uso foi uma das estratégias adotadas para descrever as interações entre os usuários e o sistema. Exemplos incluem os cenários “Efetuar Login” e “Cadastrar Cliente”, que detalham os fluxos de trabalho para cada funcionalidade. O desenvolvimento do diagrama de classes foi essencial para representar a estrutura do sistema, destacando as principais classes, atributos e métodos, como Cliente, Produto e Venda (Leite, 2022).

A modelagem de processos de negócios foi realizada com a utilização do *Business Process Model and Notation* (BPMN), permitindo uma representação visual dos fluxos de trabalho da loja e a identificação de áreas passíveis de otimização. Diagramas complementares, como o diagrama de atividades, ilustraram etapas e decisões de processos-chave, como o registro de uma venda. O diagrama de estados foi utilizado para representar as diferentes condições de objetos como pedidos ao longo de seu ciclo de vida.

Adicionalmente, os diagramas de sequência descreveram interações temporais entre objetos em cenários específicos, detalhando chamadas de métodos, especialmente em processos como a finalização de uma venda. Para garantir o cumprimento integral dos requisitos definidos durante o desenvolvimento, foi criada uma matriz de rastreabilidade, facilitando o acompanhamento e a verificação de cada item.

Mais detalhes sobre os requisitos e diagramas citados estão disponíveis no material de Ramos (2024).

Modelagem do Sistema

A modelagem do sistema foi uma etapa essencial no desenvolvimento do SAFA, pois permitiu a tradução dos requisitos levantados em representações gráficas. Essas representações facilitaram a compreensão e a comunicação entre a equipe de desenvolvimento e os stakeholders. Utilizou-se o Figma para a prototipação das telas do sistema. Essa ferramenta foi fundamental na criação de um design visual e interativo, possibilitando a validação de ideias e a coleta de feedback dos usuários antes do início do desenvolvimento.

É relevante destacar que, apesar de diversas telas terem sido prototipadas, a implementação inicial concentrou-se nas principais funcionalidades. Essa priorização garantiu um desenvolvimento ágil, permitindo que o produto fosse testado e validado rapidamente.

Os modelos e protótipos desenvolvidos foram fundamentais para a validação dos requisitos e para a construção de uma arquitetura sólida. A aplicação dessas técnicas assegurou que o sistema atendesse às expectativas dos usuários de maneira eficiente e alinhada às necessidades do setor.

Tecnologias Utilizadas

Para o desenvolvimento do sistema, diversas tecnologias foram escolhidas com base na familiaridade da equipe e nas necessidades do projeto. A seguir, estão listadas as principais ferramentas utilizadas.

Figma: utilizado para a elaboração de protótipos de telas, se destacou pela interface intuitiva e pela ampla disponibilidade de recursos, como tutoriais e *templates*, que facilitam sua utilização durante o desenvolvimento.

Visual Studio Code: este editor de código foi escolhido devido à familiaridade da equipe e à vasta biblioteca de extensões, permitindo a integração com várias ferramentas e facilitando o desenvolvimento (*Visual Studio Code*, 2024).

Node.js: uma plataforma de código aberto que permite executar JavaScript no lado do servidor, o Node.js foi escolhido por sua capacidade de facilitar a manutenção do código e acelerar o desenvolvimento da API, uma vez que a *stack* utilizada para o *front-end* também é baseada em JavaScript.

PostgreSQL: um sistema de gerenciamento de banco de dados relacional de código aberto, o PostgreSQL foi escolhido por sua robustez e facilidade de uso, sendo amplamente utilizado em diversos tipos de aplicativos, desde pequenos projetos a grandes sistemas empresariais (Stonebraker, 2017).

Draw.io e Lucidchart: ambas as ferramentas foram empregadas para criar diagramas que representam processos e fluxos do sistema. O Draw.io, por ser uma ferramenta com interface intuitiva, facilitou a criação de diagramas, enquanto o Lucidchart foi utilizado para diagramas mais complexos, aproveitando sua flexibilidade e recursos colaborativos.

Supabase: como uma solução de *backend-as-a-service* (BaaS), o Supabase fornece uma plataforma robusta baseada em PostgreSQL, oferecendo suporte nativo a APIs RESTful e autenticação integrada. Essas características foram determinantes para sua escolha, especialmente em projetos que demandam escalabilidade e agilidade no *back-end* (Supabase, 2024).

Prisma: foi escolhido para facilitar a interação com o banco de dados PostgreSQL. Essa ferramenta fornece uma camada de abstração para consultas SQL, simplificando a manipulação de dados e garantindo maior segurança nas transações.

Além disso, o Prisma contribui para um código mais limpo, fácil de manter e menos propenso a erros (Prisma, 2024).

HTML, CSS e JavaScript: as tecnologias fundamentais para o *front-end*, onde HTML estrutura as páginas, CSS estiliza e garante responsividade, e JavaScript proporciona interatividade. Essas tecnologias foram escolhidas pela robustez e pela familiaridade da equipe com elas (W3C, 2020).

Git e GitHub: o Git foi utilizado para controle de versão do código fonte, enquanto o GitHub serviu como repositório. Essas ferramentas foram essenciais para documentar alterações e deixar o código fonte mais seguro.

Desenvolvimento de funcionalidades

Neste tópico, apresentamos a implementação de funcionalidades relevantes do sistema SAFA, com trechos de código que ilustram a lógica por trás de cada operação.

O método *create* presente na Figura 1 permite a criação de um novo cliente. Este método recebe uma requisição contendo os dados de um novo cliente no corpo da requisição *req.body*. Utilizamos o método *create* do Prisma para inserir esses dados na tabela cliente. Se a operação for bem sucedida, retornamos uma resposta 201. Em caso de erro, um retorno 500 é registrado na tela.

Figura 1 – Método *create*

```
5   controller.create = async function (req, res) {
6     try {
7       await prisma.cliente.create({ data: req.body })
8
9       res.status(201).end()
10    }
11    catch(error) {
12      console.log(error)
13
14      res.status(500).end()
15    }
16  }
```

Fonte: autores (2024)

O sistema também implementa uma lógica similar para gerenciar clientes, funcionários e produtos. Assim como no módulo de clientes, as operações para funcionários permitem a criação, visualização, edição e exclusão dos

funcionários, garantindo um controle completo sobre o quadro de colaboradores. Da mesma forma, a gestão de produtos segue a mesma abordagem.

Também é relevante demonstrar o funcionamento do registro de uma venda no sistema. Este trecho de código realiza a associação de uma venda a um cliente e calcula os pontos de fidelidade correspondentes. A função é assíncrona devido à necessidade de realizar diversas operações com o banco de dados, como consultas e inserções, de forma eficiente e sem bloquear a execução do programa.

Primeiro, os dados da requisição são extraídos, incluindo o tipo de venda, a data, a forma de pagamento, os produtos, o ID do cliente e a pontuação resgatada. Em seguida, o cliente é buscado no banco de dados. Se o cliente não for encontrado, a função retorna um erro. Caso o cliente exista, a pontuação acumulada é obtida.

Depois, os produtos da venda são validados e processados. Cada produto é buscado pelo seu ID no banco de dados e o valor total de cada item é calculado. Todo esse processo está especificado na Figura 2.

Figura 2 – Criar venda parte 1

```
controller.create = async function (req, res) {
  try {
      const { tipo, data, formaPagamento, produtos, clienteId, pontuacaoResgatada = 0 } = req.body;

      const cliente = await prisma.cliente.findUnique({
          where: { id: clienteId },
      });

      if (!cliente) {
          return res.status(404).send({ error: 'Cliente não encontrado' });
      }

      const pontuacaoAtual = parseFloat(cliente.pontuacaoAcumulada || 0);

      const vendaProdutosData = await Promise.all(
          produtos.map(async (produto) => {
              const produtoData = await prisma.produto.findUnique({
                  where: { id: produto.produtoId },
              });
              if (!produtoData) {
                  throw new Error(`Produto com ID ${produto.produtoId} não encontrado`);
              }
              const valorUnitario = parseFloat(produtoData.preco.replace(",", "."));
              return {
                  quantidade: produto.quantidade,
                  valorUnitario: valorUnitario,
                  valorTotal: valorUnitario * produto.quantidade,
                  produtoId: produto.produtoId,
              };
          })
      );
```

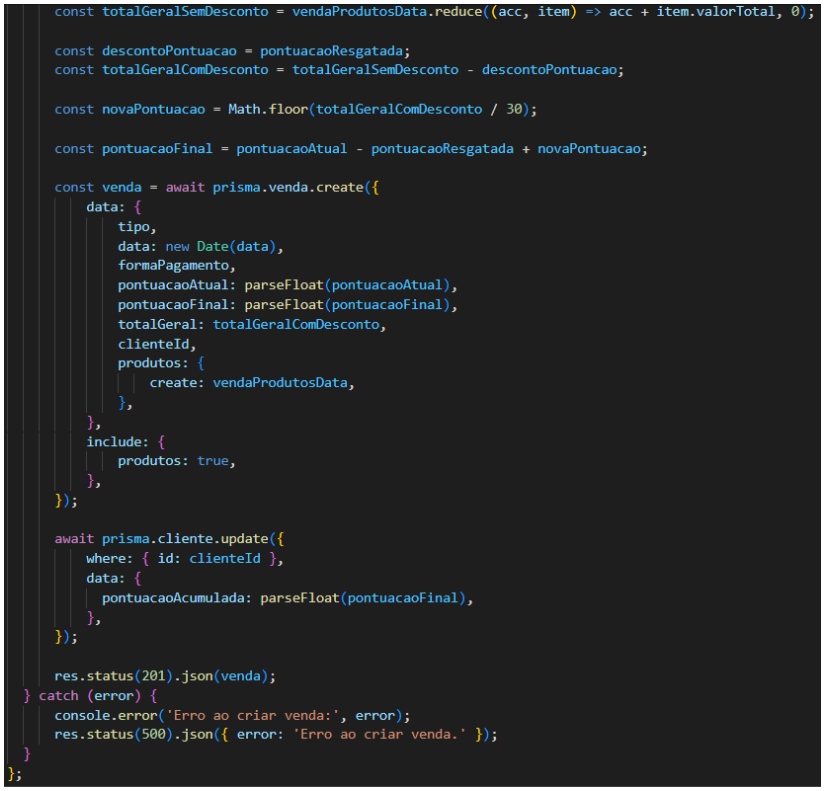
Fonte: autores (2024)

Na Figura 3 é mostrado a continuação dessa lógica, onde o total da compra é somado e o desconto, que equivale à pontuação resgatada, é aplicado. A

nova pontuação do cliente é calculada com base no valor total da compra, e a venda é registrada no banco de dados com todos os detalhes.

Por fim, em caso de erro ao concluir o processo, uma mensagem é apresentada na tela.

Figura 3 – Criar venda parte 2



```
    const totalGeralSemDesconto = vendaProdutosData.reduce((acc, item) => acc + item.valorTotal, 0);

    const descontoPontuacao = pontuacaoResgatada;
    const totalGeralComDesconto = totalGeralSemDesconto - descontoPontuacao;

    const novaPontuacao = Math.floor(totalGeralComDesconto / 30);

    const pontuacaoFinal = pontuacaoAtual - pontuacaoResgatada + novaPontuacao;

    const venda = await prisma.venda.create({
      data: {
        tipo,
        data: new Date(data),
        formaPagamento,
        pontuacaoAtual: parseFloat(pontuacaoAtual),
        pontuacaoFinal: parseFloat(pontuacaoFinal),
        totalGeral: totalGeralComDesconto,
        clienteId,
        produtos: {
          create: vendaProdutosData,
        },
      },
      include: {
        produtos: true,
      },
    });

    await prisma.cliente.update({
      where: { id: clienteId },
      data: {
        pontuacaoAcumulada: parseFloat(pontuacaoFinal),
      },
    });

    res.status(201).json(venda);
  } catch (error) {
    console.error('Erro ao criar venda:', error);
    res.status(500).json({ error: 'Erro ao criar venda.' });
  }
};
```

Fonte: autores (2024)

Integração de componentes

A integração entre o *front-end* e o *back-end* do sistema foi realizada utilizando a biblioteca Axios, que permite a realização de requisições *Hypertext Transfer Protocol* (HTTP). Essa ferramenta permite o envio e recebimento de dados do servidor sem a necessidade de recarregar a página, o que proporciona uma experiência mais fluida e satisfatória para o usuário.

Nas telas desenvolvidas em HTML, o Axios foi empregado para executar operações como o envio de dados ao servidor e a busca dinâmica de informações, conforme demonstrado na Figura 4. Por meio dessa biblioteca, foi possível implementar a comunicação entre o *front-end* e o *back-end* de forma ágil.

Essa integração foi aplicada em todas as telas do sistema, permitindo operações rápidas e em tempo real. Tal abordagem contribuiu significativamente para a melhoria da usabilidade do sistema, garantindo maior responsividade e eficiência no acesso aos dados.

Figura 4 – Uso do Axios nos arquivos HTML

```
<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
```

Fonte: autores (2024)

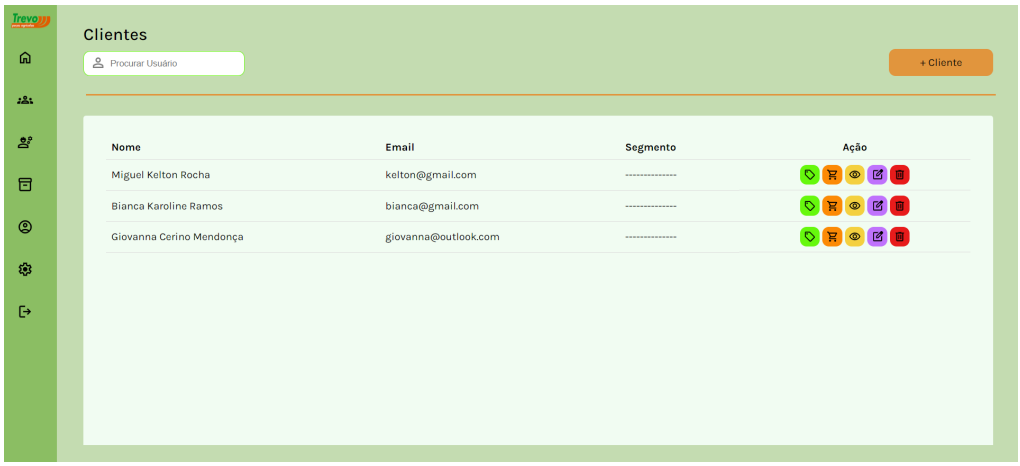
Resultados e Discussão

A aplicação SAFA adota uma abordagem de navegação estruturada e eficiente, utilizando um menu lateral que permite aos usuários acessar rapidamente qualquer funcionalidade do sistema a partir de qualquer tela. Na tela de "Clientes" (Figura 5), são exibidos os dados dos clientes cadastrados, como nome, e-mail e segmento. Além disso, os usuários podem realizar diversas ações, como adicionar uma venda ao cliente, visualizar vendas existentes, consultar detalhes, editar informações ou remover o cliente. Esse *layout* simples e objetivo facilita a gestão de clientes pelos funcionários da loja.

A tela de "Produtos" (Figura 6), essencial para o desempenho do sistema, segue o mesmo padrão visual da tela de "Clientes", garantindo consistência no design e uma experiência de usuário fluida. Ambas as telas incluem recursos de busca, permitindo que os usuários filtrem rapidamente clientes ou produtos.

A interface de "Registrar Venda" (Figura 7) destaca-se como uma das mais importantes do sistema, diretamente ligada ao objetivo principal do projeto. Nela, os funcionários podem registrar uma nova venda de maneira prática, selecionando os produtos correspondentes. O sistema também permite que o funcionário adicione ou remova itens da venda de forma dinâmica, atualizando o total em tempo real. Essa tela foi projetada com foco na funcionalidade e usabilidade, incluindo botões claros para as ações de adicionar ou cancelar a venda.

Figura 5 – Tela de clientes

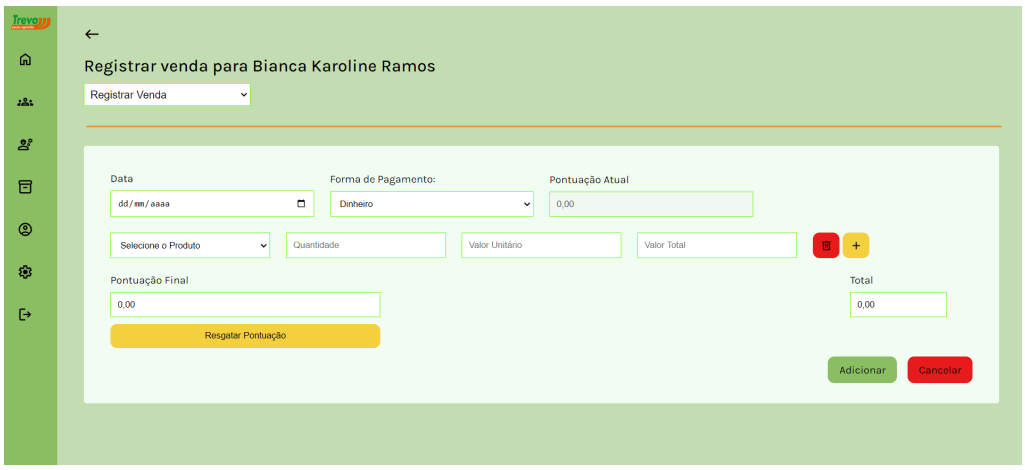


Fonte: autores (2024)

Figura 6 – Tela de produtos

Fonte: autores (2024)

Figura 7 – Tela de registrar venda



Fonte: autores (2024)

Todas as telas de gerenciamento foram desenvolvidas com ênfase na simplicidade e usabilidade, visando agilizar as operações do dia a dia e melhorar a experiência dos usuários no controle das atividades relacionadas à fidelização de clientes.

Considerações finais

Ao iniciar este projeto, o objetivo principal foi desenvolver um sistema digital de fidelização de clientes que atendesse às necessidades específicas de lojas agrícolas. Buscou-se otimizar a experiência de compra e fortalecer o relacionamento entre a loja e seus clientes. Para alcançar esse objetivo, foi criado o Sistema de Atendimento e Fidelização Agro (SAFA), uma solução personalizada e eficiente, que utiliza o histórico de compras e as preferências dos usuários para gerenciar todo o processo de forma integrada.

Durante o desenvolvimento, enfrentaram-se desafios significativos, como a adaptação do sistema às particularidades do nicho em questão, a manipulação eficiente de grandes volumes de dados e a criação de uma interface amigável e intuitiva, adequada tanto para administradores e funcionários quanto para os clientes. A escolha criteriosa das ferramentas e tecnologias, juntamente com a prototipagem e os feedbacks obtidos ao longo do processo, foi determinante para superar esses obstáculos e garantir o sucesso do projeto.

Quanto ao futuro do sistema, diversas possibilidades de evolução se apresentam. Entre elas, destaca-se a integração de relatórios gráficos mais detalhados, permitindo uma análise mais aprofundada do comportamento e das preferências dos clientes. Além disso, a incorporação de funcionalidades que conectem o sistema a outras soluções de gestão do agronegócio representa uma oportunidade de aumentar a eficiência no setor. Outra perspectiva promissora é a expansão da plataforma para incluir suporte a aplicativos móveis, proporcionando uma experiência ainda mais acessível e dinâmica para os usuários.

Dessa forma, conclui-se que o sistema desenvolvido não apenas atendeu ao problema inicialmente proposto, como também abriu espaço para investimentos futuros. Acredita-se que esse projeto pode contribuir para o avanço do setor, especialmente no contexto de fidelização de clientes e otimização de vendas em lojas agrícolas.

Referências Bibliográficas

HAZZAN, O.; DUBINSKY, Y. Research in Software Engineering: A Comprehensive Guide to Software Engineering Research. Springer, 2021.

LARMAN, Craig. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. 3. ed. Prentice Hall, 2005.

LEITE, J. C. S.; T. L. K. C. R. T. Requirements Engineering: A Comprehensive Survey. IEEE Transactions on Software Engineering, v. 48, n. 1, p. 1-25, 2022.

MICROSOFT. Visual Studio Code Documentation. Disponível em: https://code.visualstudio.com/docs. Acesso em: 17 nov. 2024.

PRISMA. Prisma Documentation. Disponível em: https://www.prisma.io/docs. Acesso em: 17 nov. 2024.

RAMOS, B. SAFA. 2024. Disponível em: https://drive.google.com/drive/folders/12Gans3LXVMEHQ_RUvoS2i3sLF7dPy5uo?usp=sharing. Acesso em: 07 out. 2024.

STONEBRAKER, Michael. PostgreSQL: A Brief History and Future Directions. [S.l.], 2017. Disponível em: https://www.postgresql.org. Acesso em: 17 nov. 2024.

SUPABASE. Supabase Documentation. Disponível em: https://supabase.com/docs. Acesso em: 17 nov. 2024.

TILKOV, S.; VINOSKI, S. Node.js: Using JavaScript to Build High-Performance Network Applications. IEEE Internet Computing, v. 14, n. 6, p. 80-83, 2010.

W3C. HTML5: A Vocabulary and Associated APIs for HTML and XHTML. W3C, 2020. Disponível em: https://www.w3.org/TR/html5/.

WIEGERS, Karl; BEATTY, Joy. Software Requirements. 3. ed. Microsoft Press, 2013.