PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA E DE ARTES
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

CREFIDE: ESTUDO DE TECNOLOGIAS PARA O DESENVOLVIMENTO DE UMA APLICAÇÃO *WEB*

LUIZ AUGUSTO VIEIRA BOSCO

GOIÂNIA
2024

LUIZ AUGUSTO VIEIRA BOSCO

CREFIDE: ESTUDO DE TECNOLOGIAS PARA O DESENVOLVIMENTO DE UMA APLICAÇÃO *WEB*

Trabalho de Conclusão de Curso apresentado à Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Me. Aníbal Santos Jukemura

GOIÂNIA

2024

LUIZ AUGUSTO VIEIRA BOSCO

CREFIDE: ESTUDO DE TECNOLOGIAS PARA O DESENVOLVIMENTO DE UMA APLICAÇÃO *WEB*

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção do título de Bacharel em Ciência da Computação, e aprovado em sua forma final pela Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás em 17/06/2024.

_____

Profa. Ma. Ludmilla Reis Pinheiro dos Santos
Coordenadora de Trabalho de Conclusão de Curso

Banca Examinadora:

_____

Orientador: Prof. Me. Aníbal Santos Jukemura

_____

Prof. Me. Vicente Paulo de Camargo

_____

Profa. Dra. Carmen Cecilia Centeno

GOIÂNIA

2024

AGRADECIMENTOS

Gostaria de agradecer primeiramente a mim mesmo por ter sido capaz de chegar até aqui, apesar das circunstâncias, com todo esforço que foi posto ao longo desses anos e que me moldaram a ser quem eu sou hoje.

Agradeço aos meus pais e todo o esforço que colocaram para me ajudar. Eles colaboraram para o início da minha jornada na Universidade. Sem o apoio deles, não teria sido possível chegar aonde cheguei.

Agradeço também aos meus amigos, que estiveram comigo ao longo dessa caminhada, testemunharam e ajudaram em cada desafio e em cada conquista alcançada, em especial minha amiga Gabriela de Paula, que, através do apoio, das conversas e das trocas de ideias valiosas, fizeram todo esse percurso mais tolerável.

Estendo meus agradecimentos também a todos os meus professores que contribuíram de alguma forma para o enriquecimento do meu aprendizado e para a concretização deste trabalho.

Esse trabalho não teria sido possível sem a ajuda de todos que, mesmo que minimamente, passaram pela minha vida, deixaram sua marca e fizeram parte de tudo, mesmo sem saber.

E um último agradecimento à minha avó Maria de Carmo Bosco Alves, que infelizmente acabei perdendo ao longo dessa caminhada, mas que sempre me incentivou, sempre me motivou e teve a maior das contribuições para que eu pudesse chegar ao final dessa jornada.

RESUMO

Os programas de fidelização comercial recompensam os clientes com base em critérios estabelecidos pelos comércios, como o número de compras ou valores gastos. Entretanto, problemas como o gasto com papel, perda ou danos aos cartões são bastante comuns. Com os avanços tecnológicos, até mesmo pequenos estabelecimentos possuem computadores ou celulares com acesso à Internet, permitindo que a responsabilidade pela fidelidade seja transferida do cliente para o estabelecimento por meio de uma aplicação *Web*. Isso possibilita ao comércio gerenciar seus clientes ativos e tomar decisões de negócio baseadas nas informações obtidas. Através do uso de tecnologias modernas, como Next.JS, é possível criar uma aplicação escalável e eficaz para atender as necessidades de um sistema voltado para o gerenciamento de fidelidades comerciais.

Palavras Chave: Next.JS, Web, Desenvolvimento de Software.

ABSTRACT

Commercial loyalty programs reward customers based on criteria established by businesses, such as the number of purchases or amounts spent. However, issues like the cost of paper, loss, or damage to cards are quite common. With technological advances, even small establishments have computers or cell phones with Internet access, allowing the responsibility for loyalty to be transferred from the customer to the establishment via a web application. This enables businesses to manage their active customers and make business decisions based on the information obtained. Using modern technologies, such as Next.JS, it is possible to create a scalable and efficient web application to meet the needs of a commercial loyalty management system.

Keywords: Next.JS, Web, Desenvolvimento de Software.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

AJAX – *Asynchronous JavaScript and XML*

API – *Application Programming Interface*

CRUD – *Create, Read, Update, Delete*

CSR – *Client-Side Rendering*

CSS – *Cascading Style Sheets*

HTML – *HyperText Markup Language*

HTTP – *HyperText Transfer Protocol*

JSON – *JavaScript Object Notation*

ORM – *Object Relational Mapper*

REST – *Representational State Transfer*

SQL – *Structured Query Language*

SSR – *Server-Side Rendering*

UML – *Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

Fidelidade é a palavra utilizada nos negócios para denominar clientes que compram bens e serviços de uma empresa repetidas vezes e com exclusividade. Contudo, a fidelidade não consiste apenas em adquirir produtos, e sim preferir o produto, criar afeição pela marca e indicá-la às pessoas próximas (LOVELOCK & WIRTZ, 2006).

A fidelização entre comércios e clientes envolve criar programas ou sistemas que recompensem o cliente. Utilizando de cartões de fidelidade, os benefícios são concedidos com base em um critério estabelecido pelo próprio comércio, podendo ser baseado no número de vezes que o cliente comprou algo ou no valor que foi gasto por ele.

Segundo Barlow (1992), fidelização é uma estratégia que identifica, mantém e aumenta o rendimento dos melhores clientes numa relação de valor agregado, interativo e centrado no longo prazo.

Problemas comuns desse método incluem o gasto excessivo com papel para a impressão de cartões, a perda ou dano dos cartões, a possibilidade de rasgar, ser lavado dentro do bolso de uma peça de roupa e o esquecimento na hora de levar o cartão para ser preenchido.

Nos últimos anos, com a crescente adoção de tecnologias digitais, tem se tornado cada vez mais evidente a importância de ter uma presença de serviços digitais sólida para empresas de todos os tamanhos. Além disso, competitividade de mercado e a necessidade de fidelizar clientes são fatores que motivam os estabelecimentos a buscarem soluções inovadoras para atrair e manter seus consumidores.

Neste contexto, os programas de fidelização surgem como uma ferramenta valiosa, proporcionando benefícios tanto para os clientes quanto para os comércios. Este trabalho tem como objetivo desenvolver uma aplicação *Web* utilizando Next.JS que permita aos estabelecimentos gerenciar de forma eficiente seus programas de fidelização, eliminando a necessidade de cartões físicos e simplificando o processo tanto para o cliente quanto para o comerciante.

> (LAZUARDY, ANGGRAIN, 2022) “Nowadays web applications can do a lot of work besides just displaying static data. Modern web applications have become an important part of the daily ecosystem due to the demands of clients' needs. Therefore, development on the front-end side of the web plays an important role in

> website creation because it becomes a means of user interface with the system. A good, web front-end can attract users with all the convenience, feasibility, and features available.” (Hoje em dia aplicações *web* podem fazer muito mais do que só mostrar dados estáticos. Aplicações *web* modernas se tornaram uma importante parte do ecossistema diário devido à demanda das necessidades dos clientes. Portanto, desenvolvimento no lado do *front-end* da *web* desempenha um papel importante na criação de *websites* porque se torna um meio de interface do usuário com o sistema. Um bom *front-end* da *web* pode atrair usuários com toda a conveniência, viabilidade e recursos disponíveis.)

Justifica-se criar essa aplicação uma vez que uma das formas utilizadas pelos estabelecimentos de médio e pequeno porte para controle de fidelidades é o uso de cartões com espaços para marcações de carimbo ou assinaturas a cada vez que o cliente realiza uma compra, esse formato se mostra ineficiente uma vez que é possível que o cartão se perca, rasgue ou que a pessoa esqueça de sempre estar levando ao estabelecimento para ser marcado.

A ideia de criar uma aplicação voltada para o gerenciamento de fidelidades surgiu a partir de experiências pessoais que geraram transtornos, como esquecer de levar o cartão de fidelidade para a marcação mais de uma vez, fazendo o estabelecimento ter que confiar nas próximas vezes que realmente ocorreram essas idas ao local e realizar as marcações que faltaram.

Diante desse contexto, esse projeto visa responder a seguinte questão: **- É possível criar um site para gerenciamento de fidelidades comerciais?**

1.1. Objetivos

Este trabalho tem como objetivo principal desenvolver uma aplicação web utilizando Next.JS para gestão de programas de fidelidade em estabelecimentos comerciais. Os objetivos específicos incluem:

1. Analisar as necessidades e requisitos dos estabelecimentos comerciais em relação aos programas de fidelização
2. Projetar a arquitetura da aplicação *Web*, garantindo escalabilidade e eficiência.

3. Implementar as funcionalidades básicas da aplicação, como cadastro de clientes, gerenciamento de pontos e emissão de recompensas.
4. Realizar testes e validações para assegurar o correto funcionamento da aplicação.
5. Documentar todo o processo de desenvolvimento, desde a concepção até a implementação e testes.

1.2. Organização do documento

Este documento está organizado da seguinte maneira: o Capítulo 2 trata dos procedimentos metodológicos que foram utilizados no trabalho, quanto à sua natureza, seus objetivos e procedimentos técnicos. O Capítulo 3 traz o referencial teórico do trabalho, descrevendo o conjunto de ferramentas que foram utilizadas para o desenvolvimento da aplicação e informações que são essenciais para a compreensão da proposta no Capítulo 6.

O Capítulo 4 delineia os aspectos relevantes do(s) trabalho(s) relacionado(s) que guiam o desenvolvimento desse projeto, como a escolha de ferramentas a partir da exposição de suas funcionalidades e vantagens. O Capítulo 5 traz informações sobre o ambiente usado no desenvolvimento. O Capítulo 6 trata do processo de desenvolvimento da aplicação, como o que foi feito durante a implementação e a demonstração de como as tecnologias foram utilizadas e combinadas para atender as necessidades e atingir o resultado esperado.

O Capítulo 7 traz uma discussão sobre os resultados obtidos e conclui, tratando dos trabalhos futuros que possam vir a ser desenvolvidos utilizando como base as informações e estudos aqui realizados. O Capítulo 8 conta as referências bibliográficas que serviram como ferramenta essencial de estudo. Por fim, são apresentados os resultados.

2 MÉTODOS

Neste capítulo, delineiam-se os métodos e técnicas utilizados para a realização deste trabalho. A seção 2.1, são expostos os procedimentos metodológicos empregados.

2.1. Procedimentos Metodológicos

Este trabalho, quanto à sua natureza, caracteriza-se como um resumo de assunto. “Resumos de assunto buscam apenas sistematizar uma área de conhecimento, usualmente

indicando sua evolução histórica e estado da arte” (WAZLAWICK, 2014, p. 21).

Segundo seus objetivos, é uma pesquisa exploratória, que para Gil (2010) tem como objetivo proporcionar familiaridade com os problemas visando torná-los mais explícitos ou construir hipóteses. Tendo como objetivo principal o aprimoramento de ideias e descoberta de intuições.

Com base nos seus procedimentos técnicos, trata-se de uma pesquisa bibliográfica e documental. Gil (2010, p. 27) define pesquisa bibliográfica como “desenvolvida com base em material já elaborado, constituído principalmente de livros e artigos científicos”, e pesquisa documental como “desenvolvida com base em materiais que não receberam tratamento analítico, ou que ainda podem ser reelaborados”.

3 REFERENCIAL TEÓRICO

O referencial teórico deste projeto inclui uma descrição das ferramentas e conceitos do desenvolvimento *Web* essenciais para a compreensão do tema proposto, além de informações sobre as tecnologias e *frameworks* utilizados, que ajudaram a criar uma aplicação escalável e manutenível.

3.1. *JavaScript vs TypeScript*

JavaScript é uma linguagem de programação criada com o propósito de tornar a experiência em páginas *Web* mais interativa e dinâmica, com sua característica de linguagem interpretada, podendo assim ser executada no lado do cliente. Navegadores possuem um interpretador para ela acoplado, não necessitando de comunicações com servidores para validar formulários, dar um retorno ao usuário ou atualizar o conteúdo da página dinamicamente.

No que diz respeito à sua sintaxe, é de fácil entendimento, uma vez que é parecida com linguagens conhecidas, como Java e C, sendo Java a maior fonte de inspiração para a forma como *JavaScript* funciona. Mesmo assim, ao longo do tempo, *JavaScript* tomou seu próprio rumo, consolidando-se como uma linguagem de programação independente e robusta para desenvolvimento.

Segundo Flannagan (2013), *JavaScript* tornou-se a linguagem de programação mais onipresente da história, devido à quantidade de sites e navegadores modernos que a utilizam, e a variedade de dispositivos que possuem um interpretador para ela. Sendo parte

da tríade de tecnologias para desenvolvimento *Web*, juntamente com HTML e CSS, *JavaScript* especifica o comportamento das páginas *Web*.

JavaScript possui a característica de ser uma linguagem não tipada, o que significa que não é necessário especificar sempre o tipo de variável que se está declarando. Baseado no conteúdo atribuído, o interpretador assumirá qual o tipo daquela variável. Porém, com esse tipo de liberdade, surgem algumas vulnerabilidades na linguagem, como, por exemplo, nas decisões tomadas pelo seu interpretador quando algo fora dos padrões da linguagem é feito.

Para o interpretador de *JavaScript*, se uma ação inválida é realizada, como a de tentar somar um valor inteiro armazenado em uma variável e um objeto *array*, ele tentará entender o que o programador quis fazer, aplicando diversas regras conhecidas por ele para fornecer um resultado, sempre tentando fazer o melhor que puder com o que foi fornecido. Isso pode levar a erros inesperados e até mesmo travamento total da aplicação, uma vez que esses erros só serão conhecidos em tempo de execução.

Como proposta de solução para esses possíveis problemas gerados pela fraca tipagem do *JavaScript*, em 2010 os engenheiros da Microsoft criaram, e dois anos depois disponibilizaram para o público, a linguagem de programação *TypeScript*, cuja proposta era corrigir esses erros adicionando a exigência de tipagem para as estruturas, possibilitando a IDE descobrir e alertar o programador dos possíveis erros antes que fosse executado.

Para Jansen, Vane, Wolff (2016) os objetivos que formaram o *TypeScript* como é hoje foram criar uma linguagem fortemente tipada que pudesse realizar checagens de tipos estáticos em tempo de compilação. Além disso, ela possui a sua alta compatibilidade com códigos *JavaScript* já existentes, além de prover mecanismos de estruturação para grandes partes de códigos, adicionando orientação a objeto baseado em classe, interfaces e módulos.

TypeScript se torna *JavaScript* no final para ser interpretada e utilizada para o desenvolvimento *Web*, mas suas características como uma linguagem mais organizada, menos propensa a erros e que garante o desenvolvimento de aplicações em pequena e larga escala, a consolidaram como mais apropriada do ponto de vista da manutenibilidade.

Além disso, como dito anteriormente, *TypeScript* ainda é *JavaScript*, e por conta disso, todos os frameworks e bibliotecas *JavaScript* possuem suporte para aplicações

escritas em *TypeScript*, como React.JS, uma das bibliotecas mais famosas de *JavaScript*, e Next.JS, um de seus frameworks mais conhecidos e utilizados.

3.2. Next.JS

Next.JS é um framework de desenvolvimento front-end que possibilita a criação de aplicações React com renderização do lado do servidor (SSR). *Framework* consiste em uma técnica de reutilização orientada a objetos. Ela compartilha diversas características de técnicas em geral para serem reutilizadas e técnicas orientadas a objetivo em particular (JOHNSON, 1997).

Essa capacidade torna o Next.JS uma ferramenta poderosa para construir aplicativos *Web* de alto desempenho e escaláveis. Desenvolvido pela Vercel, o Next.JS facilita a implementação de funcionalidades como geração de páginas estáticas, roteamento dinâmico e integração com APIs. A documentação oficial do Next.JS (2023) destaca sua flexibilidade e eficiência, permitindo que os desenvolvedores construam desde sites simples até aplicação complexas.

O SSR do Next.JS traz uma ideia já conhecida por quem desenvolve utilizando linguagens como PHP, onde as páginas HTML são renderizadas no servidor e só então enviadas para o navegador, diferente da ideia do CSR originalmente criada com o React, que exige do cliente baixar todo o pacote do *JavaScript* antes de conseguir carregar a página, isso faz com que toda a aplicação esteja no lado do cliente quando acessada, exigindo pouco processamento do servidor, mas ao custo de aumentar muito o do cliente.

Segundo Riva (2022), as vantagens do SSR para o CSR incluem maior segurança para os aplicativos *Web*, uma vez que renderizar a página no lado do cliente significa que ações como validação de dados acontecem no servidor e não expõe dados privados para o lado do cliente, maior compatibilidade devido à página ficar disponível mesmo que o usuário desabilite o *JavaScript* ou utilize um navegador muito antigo.

Uma das principais inovações que o Next.JS trouxe foi a ideia de *convention-over-configuration*, que visa permitir ao desenvolvedor tirar proveito de diversas funcionalidades que a *framework* tem a oferecer, sem necessitar de nenhuma configuração muito complexa. É possível especificar quais páginas devem ser renderizadas no lado do servidor e quais devem ser no lado do cliente sem ter que escrever arquivos de configurações.

Foi introduzido também uma mudança na forma de lidar com a navegação entre páginas no lado do cliente. No React, realizar esse tipo de controle exigia o uso de bibliotecas como a *react-router*, que necessitava muita configuração. Dentro do Next temos a conveniência de apenas necessitar de uma pasta chamada *pages*, onde o nome da página será o nome da pasta com um arquivo *index* dentro, por exemplo *pages/home/index.tsx.*

3.3. Banco de Dados PostgreSQL

Banco de dados consiste em um conjunto de dados organizados e armazenados que possuem uma relação entre si, contendo informações sobre algum domínio específico. Os bancos de dados podem variar entre vários tipos, os mais conhecidos são os relacionais e os não relacionais.

Em um banco de dados Relacional as informações são armazenadas em um formato de tabelas, cada linha da tabela representa um registro e cada coluna representa um campo dele, onde um campo precisa ser reservado para ser a chave primária única do registro, essa chave é usada então como referência a ele, possibilitando assim que tabelas diferentes possam se relacionar passando essa chave como uma chave estrangeira para a tabela com a qual o relacionamento será feito.

Nos bancos de dados não relacional, por outro lado, não há esse conceito de chaves primárias, eles seguem o conceito de chave-valor, onde uma chave referencia um campo do registro e junto dela fica o seu valor, fazendo com que sejam mais adequados para lidar com grande volume de dados, pois não necessitam ser tão estruturados.

Para se utilizar um banco de dados é necessário um sistema gerenciador de banco de dados (SGBD), um *software* que permite a manipulação dos dados e fornece uma interface de comunicação entre a aplicação e o banco de dados.

PostgreSQL é um sistema de gerenciamento de banco de dados objeto-relacional que enfatiza a extensibilidade e conformidade com padrões técnicos. Considerado um dos bancos de dados mais avançados, ele suporta uma ampla variedade de tipos de dados, integridade referencial, *triggers* e procedimentos armazenados. Sua capacidade de lidar com grandes volumes de dados e transações simultâneas faz dele uma escolha popular para aplicações críticas que demandam alta confiabilidade.

Nesse projeto em específico também foi optado por utilizar o Supabase, uma ferramenta BaaS (*Back-end as a Service)* grátis e de código aberto que supre as

necessidades para o desenvolvimento desta aplicação. Utilizando de diversas tecnologias, ele fornece uma interface para se utilizar o PostgreSQL e realizar as ações de CRUD (*Create, Read, Update and Delete*).

3.4. Tailwind

O Tailwind CSS é um framework de CSS utilitário que permite aos desenvolvedores aplicar estilos diretamente nos componentes HTML, facilitando a criação de designs responsivos e personalizados. Diferente dos *frameworks* tradicionais, o Tailwind não fornece componentes pré-estilizados, mas sim uma série de classes utilitárias que podem ser combinadas para construir qualquer design. Conforme apresentado pela documentação do Tailwind (2023), essa abordagem aumenta a produtividade e a consistência dos projetos de desenvolvimento *Web*.

3.5. Prisma ORM

O Prisma é uma tecnologia que busca aproximar o paradigma de orientação a objetos do desenvolvimento de aplicações com um banco de dados relacional. Ao utilizar o Prisma, desenvolvedores podem definir seu modelo de dados em um esquema intuitivo, gerando automaticamente as consultas SQL necessárias. Além disso, o Prisma oferece uma interface de programação de aplicações (API) robusta para interagir com os dados de forma segura e eficiente. A documentação do Prisma (2023) enfatiza sua integração com diversos bancos de dados, como PostegreSQL, MySQL e SQLite, tornando-o uma ferramenta versátil para projetos modernos.

Com o uso do Prisma, a interação entre a aplicação que utiliza do paradigma orientado a objetos e o banco de dados relacional se torna mais prática e mais abstrata. Fornecendo proteção contra os ataques comuns em banco de dados, como *SQL Injection* e *Session Hijacking*, resulta em um ganho de tempo para programadores.

3.6. Infraestrutura para o Next.JS

Referente a infraestrutura necessária para que uma aplicação Next.JS possa funcionar em um servidor*,* é necessário um interpretador de JavaScript, chamado V8 (Chrome V8) e desenvolvido em C++, para que o código possa ser entendido no servidor através do Node.JS, uma linguagem JavaScript para *backend (*V8, 2024).

O Next.JS também necessita da instalação de gerenciadores de pacotes, como “npm” e “yarn”, que ajudam a instalar as dependências necessárias para a aplicação funcionar. Para desenvolvimento local de um sistema que utilize o Next.JS, basta um servidor simples com o Node.JS instalado (VERCEL, 2024).

Para o servidor de produção, a aplicação pode ser hospedada em provedores de nuvem como Vercel, AWS ou Google Cloud. Com o Vercel, por exemplo, por se tratar de um servidor criado pela empresa desenvolvedora do Next.JS, também chamada Vercel, oferece uma integração nativa, que facilita a implantação e o gerenciamento da aplicação (VERCEL, 2024).

Sobre o funcionamento do *backend*, o Next.JS permite a criação de rotas de API no seu diretório “pages/api”. Essas rotas são executadas no servidor e podem ser usadas para construir um *backend* simplificado dentro da própria aplicação. Cada arquivo dentro desse diretório mapeia uma rota correspondente e exporta uma função que manipula requisições HTTP (VERCEL, 2024).

4 ANÁLISE DA LITERATURA

Este capítulo descreverá o trabalho que serviu como base de referência para a escolha das ferramentas utilizadas no desenvolvimento da aplicação proposta, sustentando uso delas. O trabalho consiste em uma aplicação *front end* que faz uso de ferramentas como Next.JS e *TypeScript* no seu desenvolvimento e como elas auxiliaram e facilitaram a criação de uma interface moderna e um sistema escalável.

4.1. Trabalho(s) Relacionado(s)

O trabalho referenciado trata de um desenvolvimento de uma aplicação *web* para otimização da alocação de recursos em instituições de ensino superior, adotando uma metodologia que envolve utilizar tecnologias como Next.JS, *Tailwind* CSS, *Typescript* e Node.JS, bem como *Fastify* e *Prisma* como ORM, com o foco em criar uma plataforma intuitiva e eficiente.

Com o Next.JS 13, foram introduzidos aprimoramentos muito significativos nos seus sistemas de roteamento, que agora, além de um roteamento baseado em arquivos, também oferece um diretório que proporciona características opcionais e avançadas, onde cada

caminho nas suas rotas está vinculado a um diretório específico e que possui um arquivo page.js servindo como ponto de entrada para o conteúdo. Essa flexibilidade permitiu a inclusão de arquivos adicionais, como *layout.js*, *loading.js* e *error.js*, que proporcionam um maior controle e modularidade sobre cada uma das rotas (CHIDERA, 2023 apud CALIXTO, 2023).

Foi utilizado um modelo de otimização linear inteira para aprimorar a distribuição de disciplinas, levando em conta vários critérios e restrições. Foram realizados testes na aplicação com diferentes disciplinas de diferentes semestres e turnos, gerando resultados promissores com eficácia e usabilidade.

Com a crescente complexidade da gestão acadêmica das instituições de ensino superior, soluções eficientes para otimização de alocação de recursos começaram a ser exigidas. Este trabalho propõe a implementação de um sistema *web* que integra um otimizador com o objetivo de simplificar e aprimorar o processo de alocação utilizando a abordagem de Programação Linear Inteira (PLI) (CALIXTO, 2023).

Tanto a aplicação quanto o estudo por volta dela contribuem bastante com a gestão acadêmica, criando uma poderosa e robusta ferramenta para coordenadores de instituições de ensino, explorando a integração de tecnologias modernas e técnicas avançadas de otimização.

Na versão mais recente que temos do Next.JS, foi introduzida a Stream Loading, uma funcionalidade que permite transmitir partes menores da interface do usuário para o cliente na medida que forem sendo geradas. Com esse recurso, que até o momento somente é suportado pelo diretório *app*, interrupções causadas por partes maiores são evitadas e se mostra benéfico principalmente para usuários que possuam uma conexão com a internet mais lenta, o que melhora significativamente a experiência do usuário (VERCEL, 2023 apud CALIXTO, 2023). Com a versão 13 do Next.JS, uma das mudanças notáveis foi a introdução do Turbopack, um empacotador *JavaScript* para substituir o Webpack. Foi desenvolvido pelos mesmos criadores do Webpack, e utilizando a linguagem Rust, afirmam ser até 700 vezes mais rápido que o Webpack e até 10 vezes mais rápido que o Vite, uma alternativa mais atual (KOPPERS & PALMER, 2022 apud CALIXTO, 2023).

5 DESENVOLVIMENTO DA SOLUÇÃO E VALIDAÇÃO DA APLICAÇÃO

Este capítulo descreve o ambiente de desenvolvimento utilizado e o processo de desenvolvimento da aplicação, incluindo a coleta de requisitos, os diagramas UML, e as etapas de validação.

5.1 Ambiente de Desenvolvimento

Para o desenvolvimento da aplicação, foi utilizado o Visual Studio Code (VSCode), um editor de código leve, mas poderoso, que pode ser executado no desktop e está disponível para Windows, macOS e Linux. Ele possui suporte integrado para linguagens como *JavaScript*, *TypeScript* e Node.JS com um rico ecossistema de extensões para outras linguagens, como C++, C#, Java, *Python*, PHP, Go e .NET (MICROSOFT, 2024).

5.2 Descrição da Aplicação

O desenvolvimento da aplicação foi conduzido por meio de um processo iterativo, seguindo os seguintes procedimentos metodológicos:

5.2.1. Recolhimento de requisitos

Os requisitos necessários serão identificados a fim de criar um sistema que melhor se adeque aos possíveis usuários. Dentre esses requisitos podemos citar:

- Autenticar Usuários: os usuários poderão criar contas e serem autenticados no sistema para possibilitar o vínculo entre os clientes e as empresas que concederão as fidelidades.
- Listar Fidelidades: os usuários de clientes deverão ser capazes de visualizar em quais empresas eles possuem fidelidades ativas e informações pertinentes a elas, como a data de início da fidelidade e quantos pontos ele possui.
- Cadastrar Produto/Serviço: os usuários de empresa poderão cadastrar seus produtos que valem pontos e atribuir a eles suas pontuações e o valor que serão vendidos
- Listar Produtos/Serviços: os usuários de empresa podem verificar atrás de uma listagem quais produtos eles possuem vinculados no sistema e informações referentes a eles, como preço e a quantidade de pontos que valem.

- Excluir Produto/Serviços: os usuários empresa poderão excluir os produtos que criaram.
- Editar Produto/Serviço: os usuários de empresa deverão ser capazes de editar os produtos ou serviços cadastrados no sistema para alterar valores, pontuações, imagem ou nome.

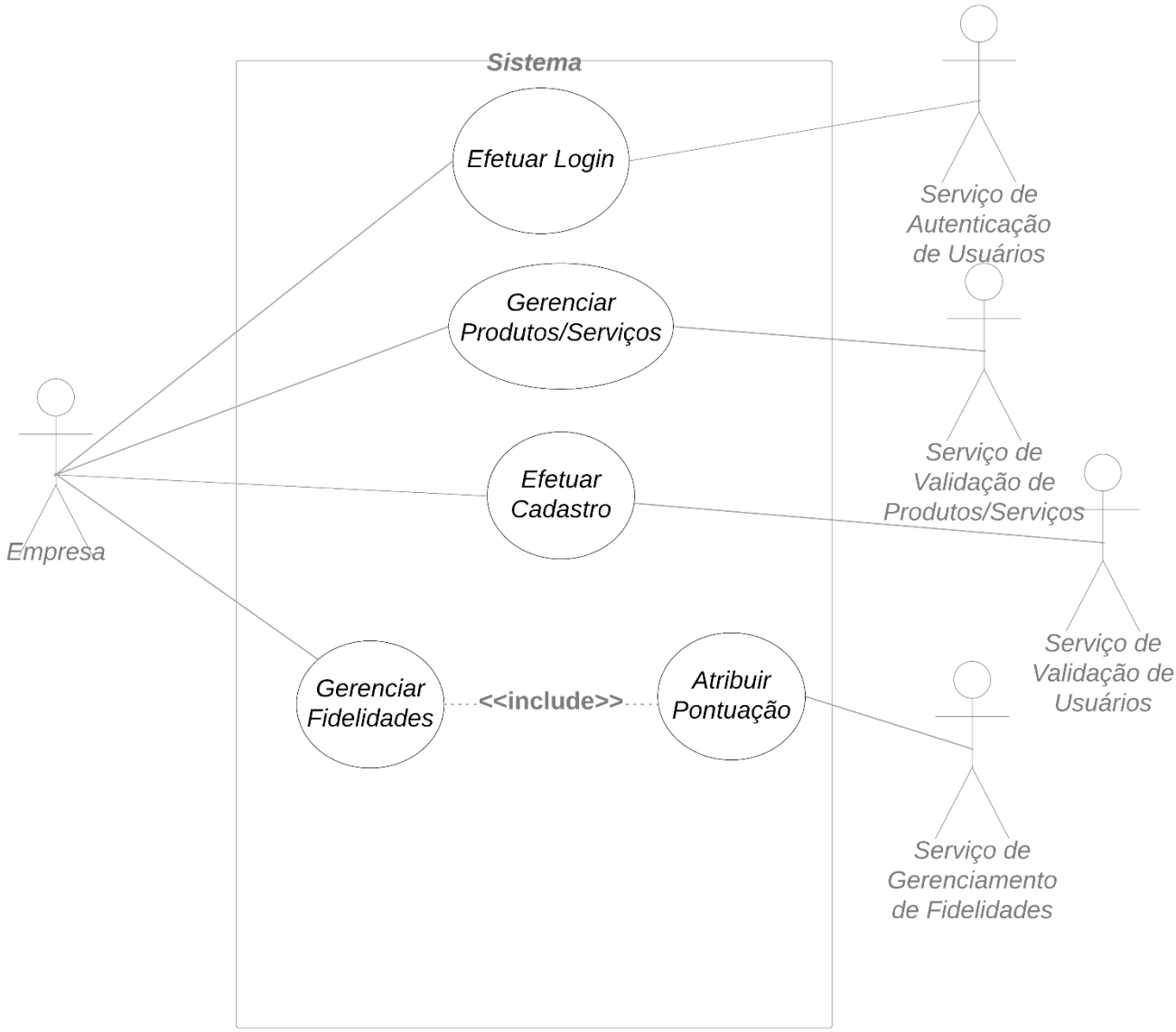
5.2.2. Diagramas UML

Os diagramas UML são uma forma de visualizar sistemas e *softwares* e ajudam a manter controle das hierarquias dentro do código, e embora alguns tenham semelhanças a árvores e decisão e fluxogramas, possuem atributos únicos.

5.2.3. Diagrama de Casos de Uso

Com um diagrama UML de casos de uso (Figura 1) conseguimos resumir os detalhes de cada usuário dentro do sistema e as interações que possuem com o sistema.

Figura 1 - Diagrama de Casos de Uso



Fonte: elaborado pelo autor (2024)

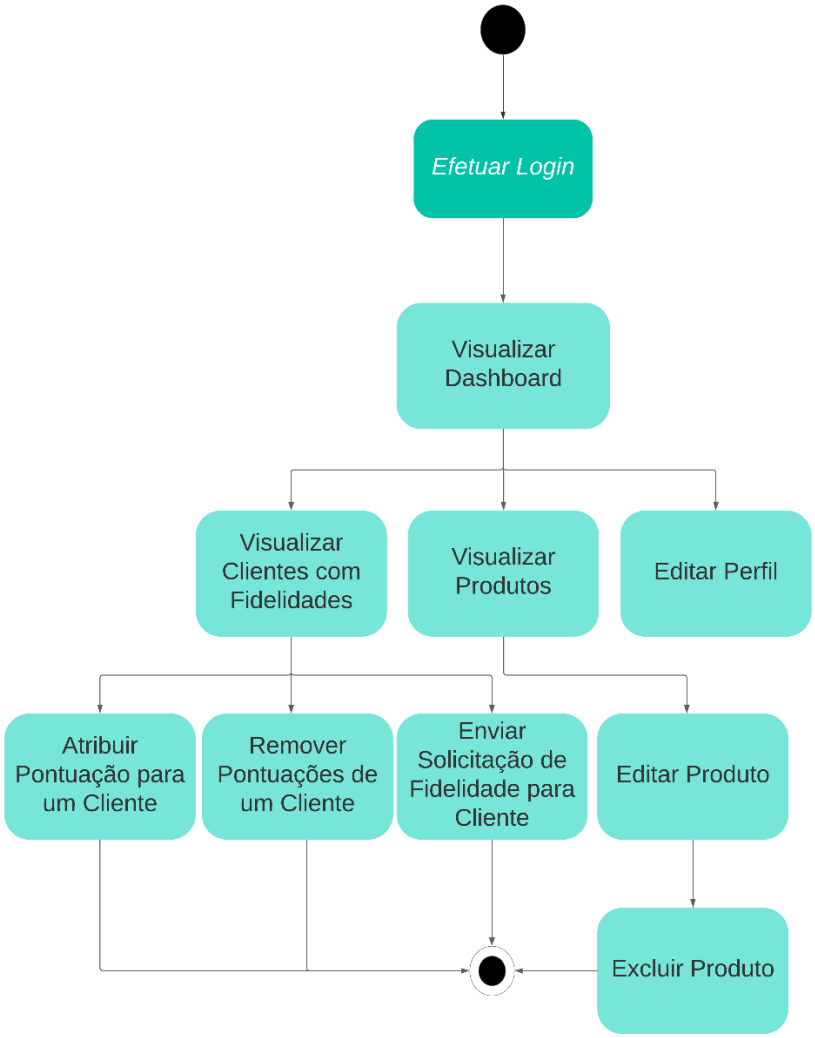
Conforme apresentado na Figura 1, é possível perceber quais atividades o ator (empresa) possui dentro do sistema. A empresa ficará responsável por gerenciar seus produtos e as fidelidades.

5.2.4. Diagrama de Atividades

Um diagrama UML de atividades (Figura 2) consiste em um gráfico de fluxo que mostra o fluxo de controle de uma atividade para outra e podem ser utilizados para modelar aspectos dinâmicos de um sistema, demonstrando principalmente etapas sequenciais.

Como mostra a Figura 2, após o processo de login, a empresa seguirá um fluxo partindo da sua tela inicial de *dashboard*.

Figura 2 - Diagrama de Atividades

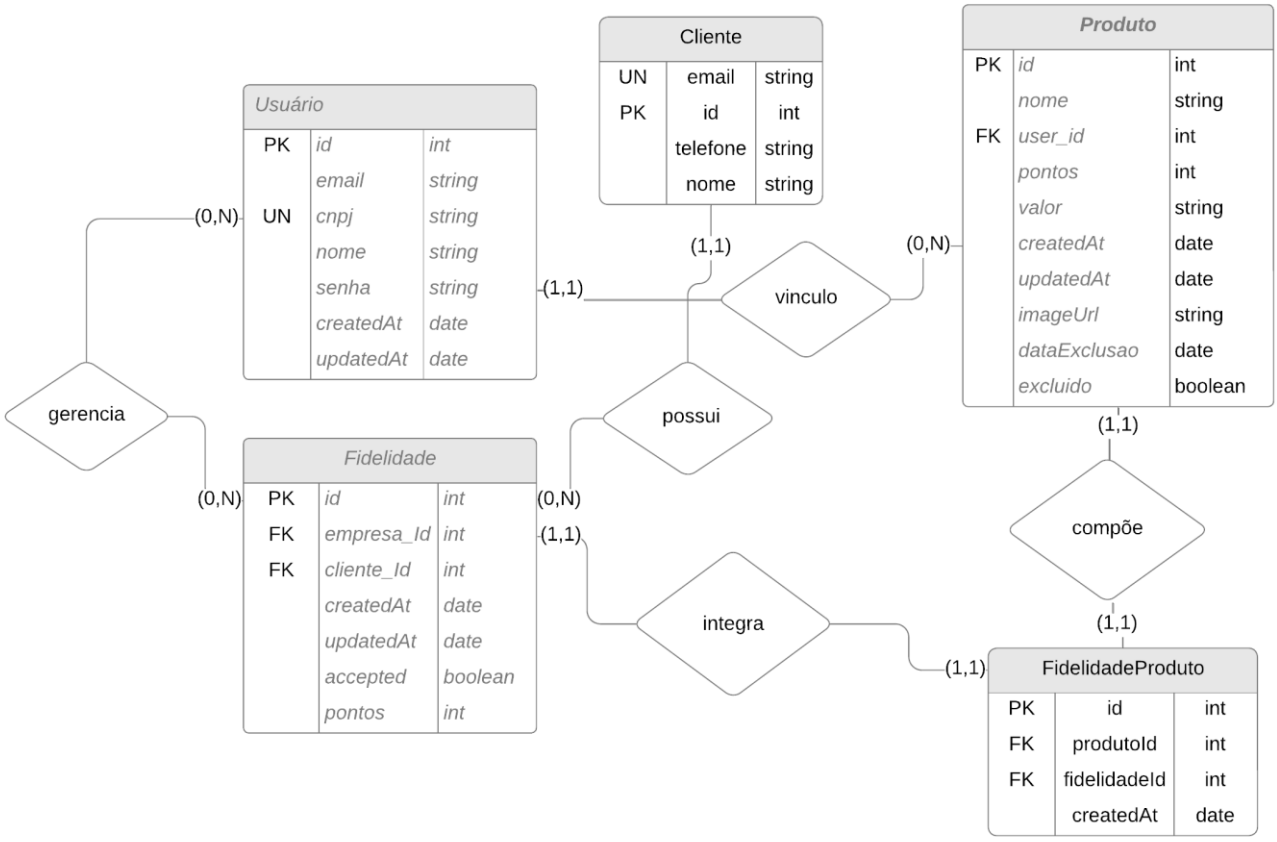


Fonte: elaborado pelo autor (2024)

5.2.5. Diagrama Entidade Relacionamento

Um diagrama entidade relacionamento (ER) é um tipo de fluxograma utilizado como forma de ilustrar “entidades”, por exemplo pessoas ou objetos, e a forma como se relacionam dentro de um sistema, sendo mais utilizados para projetar bancos de dados relacionais.

Figura 3 - Diagrama Entidade Relacionamento



Fonte: elaborado pelo autor (2024)

O diagrama entidade relacionamento, ilustrado na Figura 3, demonstra como o banco de dados da aplicação está sendo projetado. A entidade apresentada como usuário, por exemplo, possui uma tabela que demonstra seus atributos e sua relação com as fidelidades e seus produtos.

5.2.6. Descrição das Telas

Uma importante questão levantada durante o planejamento da aplicação foi na forma como as telas deveriam ser apresentadas. Para decidir o rumo a ser tomado, é necessário se pensar primeiramente em quem são os usuários alvos da aplicação e em como é sua interação com tecnologias e a forma como lida com elas. Como essa aplicação teria como foco principal empresas de pequeno, médio e grande porte que buscam implementar um sistema para gerenciar suas fidelidades, além comerciantes individuais que também buscam uma forma simples e prática de gerenciar seus clientes fiéis, é necessário tornar a

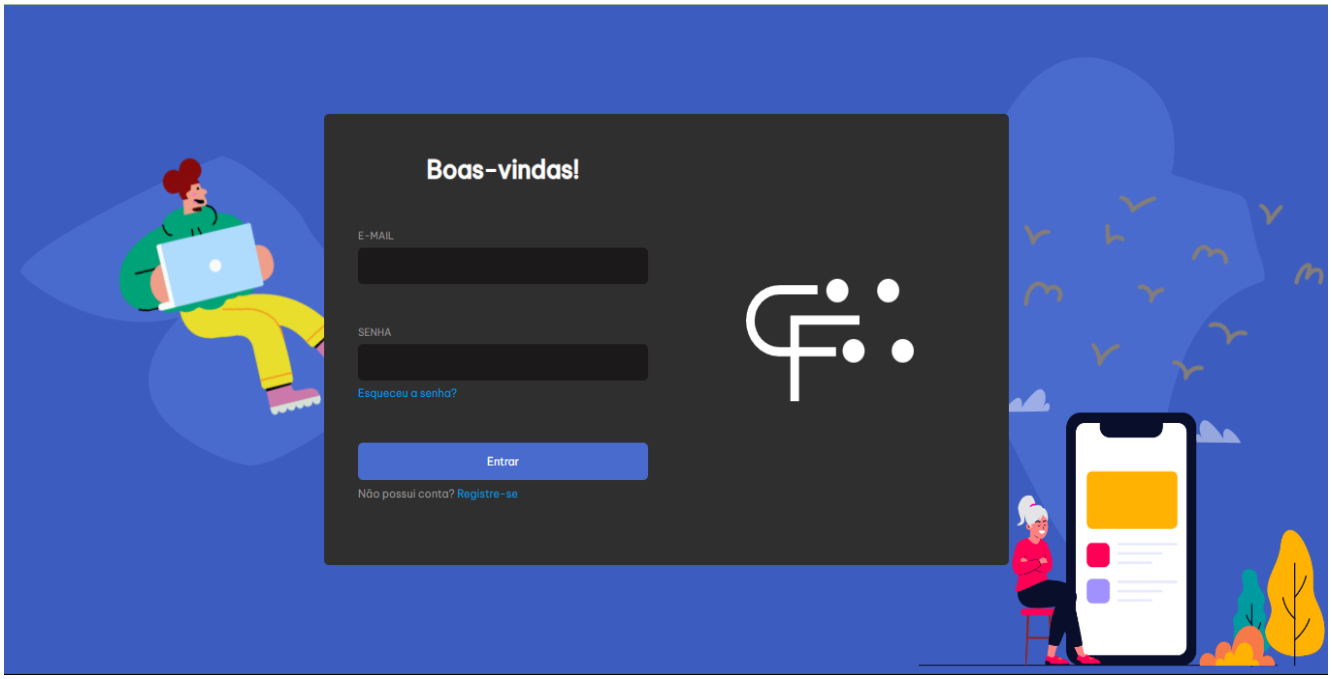
experiência dos usuários a mais simples e objetiva possível, e que o sistema seja capaz de lidar com todas as exceções que possam vir a acontecer.

Figura 4 - Tela Inicial da Aplicação

Fonte: elaborado pelo autor (2024)

A Figura 4 ilustra a tela inicial da aplicação, sendo esse o ambiente em que os usuários da aplicação terão seu primeiro contato com ela. Nelas o usuário poderá ter uma breve descrição do que esperar da aplicação, caso ele venha a ser empresa que busca implementar um sistema de fidelidade para si.

Figura 5 - Tela de *Login*



Fonte: elaborado pelo autor (2024)

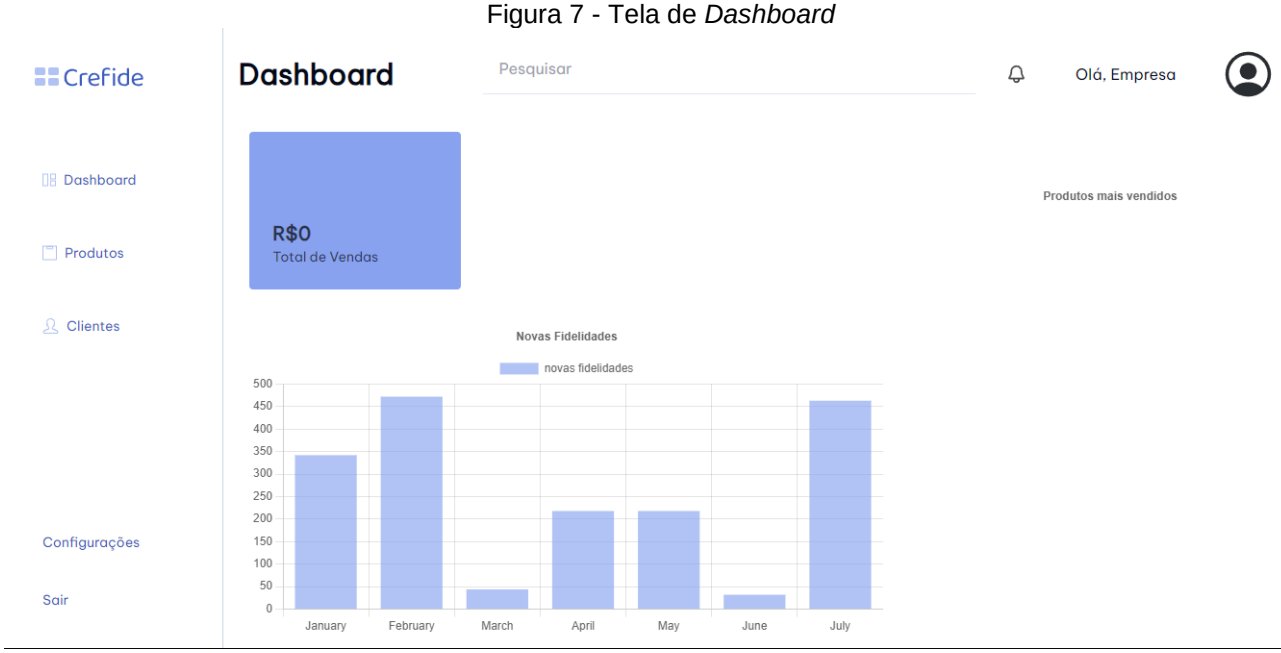
Na tela de *login* (Figura 5) o usuário verá somente as informações relevantes para melhor entendimento, com textos bem descritivos do que ele pode fazer para evitar que fique estacionado sem saber como proceder em alguma ação.

Figura 6 - Tela de Cadastro

Fonte: elaborado pelo autor (2024)

Na tela de cadastro (Figura 6) o usuário poderá se registrar no sistema fornecendo informações relevantes para ser categorizado como empresa e para que sua dinâmica dentro do sistema seja mais personalizada a partir de informações como seu nome fantasia.

Figura 7 - Tela de *Dashboard*



Fonte: elaborado pelo autor (2024)

A Figura 7 demonstra como é composta a primeira tela vista pelo usuário empresa ao fazer *login*, nela vai ser possível visualizar os gráficos informativos com novas fidelidades, os produtos mais vendidos pela empresa e o valor total adquirido das vendas por exemplo.

Figura 8 - Tela de Produtos

Fonte: elaborado pelo autor (2024)

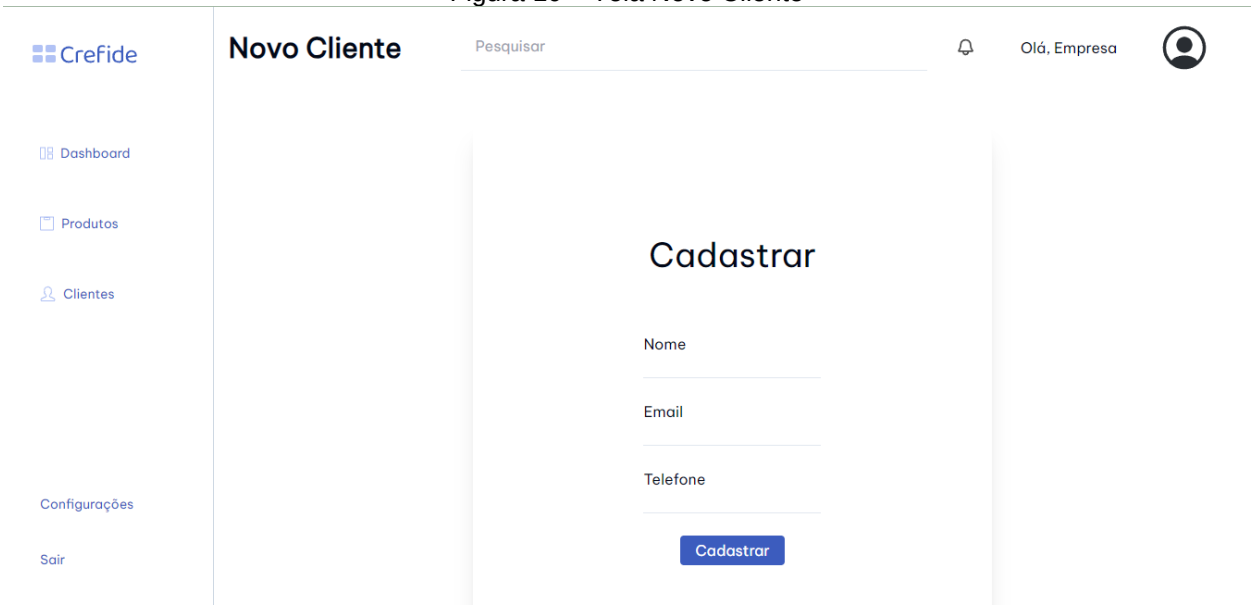
Na tela de produtos (Figura 8), o usuário poderá visualizar uma prévia dos produtos que possui cadastrados, o valor deles e a sua pontuação, será possível editar o produto ou adicionar um novo.

Figura 9 - Tela de Clientes

Fonte: elaborado pelo autor (2024)

A tela de clientes (Figura 9) será possível visualizar a lista de clientes que estão no programa de fidelidade da empresa, através dos filtros e buscas por nome e *e-mail* o usuário poderá buscar algum cliente específico e abrir a sua tela de fidelidade para gerenciar, também é possível criar uma fidelidade, onde o usuário empresa enviará um convite para o cliente solicitando a participação no programa de fidelidade.

Figura 10 – Tela Novo Cliente



Fonte: elaborado pelo autor (2024)

A tela de cadastrar novos clientes (Figura 10) possibilita à empresa cadastrar seus clientes para que possam ser vinculados nas fidelidades e assim permitir que ela atribua as pontuações.

6 CONCLUSÃO E TRABALHOS FUTUROS

O sistema proposto visa aprimorar os programas de fidelidade que utilizam cartões marcados, oferecendo uma interface eficaz e intuitiva. Através da metodologia iterativa, o desenvolvimento do “CREFIDE” passou por várias iterações, refinando a interface e o protótipo para otimizar o fluxo de atividades dos usuários.

Do ponto de vista prático, o sistema fornece uma interface amigável e intuitiva, que poderá simplificar a vida dos clientes que frequentemente enfrentam problemas com cartões de fidelidades, transferindo a responsabilidade do gerenciamento para a empresa que administra o programa. Isso, por sua vez, aumentará o controle da empresa sobre os clientes fiéis.

O “CREFIDE” também se destaca pela possibilidade de eliminar o uso dos cartões físicos, que geralmente requerem muita impressão em papel e acabam sendo descartados, seja por perda ou descuido. Migrar a dinâmica dos programas de fidelidade para o ambiente eletrônico, que tende a crescer e se tornar mais acessível, permitirá também que mais empresas de pequeno porte ou simples prestadores de serviços individuais tenham maior controle sobre seu negócio com o mínimo de gastos e de forma mais sustentável.

Quanto às limitações e dificuldades, não foram encontradas durante o desenvolvimento, já que todas as funcionalidades necessárias do sistema foram eficazmente atendidas pelas tecnologias empregadas. Com o uso do Next.JS, o desenvolvimento dos componentes tornou-se mais prático e ágil, enquanto as tecnologias integradas contribuíram para a melhoria das interfaces e da comunicação com o *backend* aprimorando o desempenho e capacidade de lidar com um grande volume de acessos e processos.

Para aprimorar o sistema, algumas melhorias e adições podem ser consideradas, visando proporcionar uma experiência mais confortável para os usuários. Entre elas, está a criação de um aplicativo móvel integrado ao sistema, que facilite o acesso dos usuários-clientes por meio de seus celulares. Para os usuários-empresas, podem ser incluídas novas maneiras de gerenciar as fidelidades e novos gráficos que auxiliem no gerenciamento de seus programas de fidelidade, além de novos métodos de atribuição de pontos.

Em resumo, o “CREFIDE” representa um avanço significativo para o gerenciamento de programas de fidelidades comerciais, combinando tecnologias modernas e princípios sólidos de programação robusta. Sua originalidade é notável tanto no contexto acadêmico quanto social, oferecendo contribuições práticas e teóricas, abrindo caminho para novas ideias e pesquisas a contínua melhoria do processo de gerenciamento de fidelidades comerciais.

7 REFERÊNCIAS BIBLIOGRÁFICAS

BARLOW, R. **Relationship Marketing – The ultimate in costumer services, Retail Control**, 1992. Acessado em junho de 2024.

CALIXTO, L M. S. **UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE ESCOLA DE CIÊNCIAS E TECNOLOGIA GRADUAÇÃO EM CIÊNCIA E TECNOLOGIA ÊNFASE EM COMPUTAÇÃO APLICADA. Front end para um sistema de otimização de alocação de horários de aulas Natal, RN, 12 de dezembro de 2023**. [s.l: s.n.]. Disponível em: <https://repositorio.ufrn.br/bitstream/123456789/56439/2/FrontendPara_Calixto_2023.pdf>. Acessado em outubro de 2023.

Desenvolvimento Web: conceito, vantagens e carreira. Voomp, 2021 Disponível em: https://blog.voomp.com.br/graduacao/tecnologia/desenvolvimentoweb-conceito-vantagens-e-carreira. Acesso em outubro de 2023.

FARIZ, M.; LAZUARDY, S.; ANGGRAINI, D. Modern Front End Web Architectures with React.Js and Next.Js. **International Research Journal of Advanced Engineering and Science**, v. 7, n. 1, p. 132–141, 2022. Acesso em fevereiro de 2024.

FIGMA. Figma: the collaborative interface design tool. Disponível em: <https://www.figma.com/>. Acesso em novembro de 2023.

GIL, A. C. **Como elaborar projetos de pesquisa**, 5. ed. São Paulo: Atlas, 2010. Acessado em setembro de 2023.

JANSEN, R. H.; VANE, V.; WOLFF, I. G. DE. ***TypeScript: Modern JavaScript Development***. [s.l.] Packt Publishing Ltd, 2016. Acessado em maio de 2024.

JOHNSON, R. E. Components, frameworks, patterns. **Proceedings of the 1997 symposium on Software reusability - SSR ’97**, 1997. Acessado em junho de 2024.

LOVELOCK, Christopher; WIRTZ, Jochen. . **Marketing de serviços: pessoas, tecnologia e resultados**. 5. ed. São Paulo: Prentice Hall, 2006. 412p. Acessado em junho de 2024.

RIVA, M. ***Real-World Next.js: Build scalable, high-performance, and modern web applications using Next.js, the React framework for production***. [s.l.] Packt Publishing Ltd, 2022. Acessado em junho 2024.

SUPABASE. ***The Open Source Firebase Alternative***. Disponível em: <https://supabase.com/>. Acesso em junho de 2024.

V8 JavaScript engine. Disponível em: <https://v8.dev/>.

VERCEL. ***Next.js by Vercel - The React Framework***. Disponível em: <https://nextjs.org/>. Acesso em setembro de 2023.

WAZLAWICK, R. S. **Metodologia de pesquisa para ciência da computação.** Rio de Janeiro: Elsevier, 2014. Acessado em novembro de 2023.

ZHAO, Z. ***Build A Live News Application With Next.js 13***. Disponível em: <https://urn.fi/URN:NBN:fi:amk-202304155335>. Acessado em junho de 2024.

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
GABINETE DO REITOR

Av. Universitária, 1069 ● Setor Universitário
Caixa Postal 86 ● CEP 74605-010
Goiânia ● Goiás ● Brasil
Fone: (62) 3946.1000
www.pucgoias.edu.br ● reitoria@pucgoias.edu.br

RESOLUÇÃO n° 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O(A) estudante LUIZ AUGUSTO VIEIRA BOSCO
do Curso de Ciência da Computação, matrícula 2018.2.0028.0027-6, telefone: (62) 99947-7666 e-mail luizaugusto1299@gmail.com, na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do Autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado CREFIDE: ESTUDO DE TECNOLOGIAS PARA O DESENVOLVIMENTO DE UMA APLICAÇÃO *WEB*, gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto(PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 24 de Julho de 2024.

Assinatura do autor: Luiz Augusto Vieira Bosco

Nome completo do autor: LUIZ AUGUSTO VIEIRA BOSCO

Assinatura do professor–orientador: _____

Nome completo do professor-orientador: ANIBAL SANTOS JUKEMURA