CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ - UNIDAVI

EDUARDO FERTIG BASTOS

PROTÓTIPO DE APLICAÇÃO *WEB* PARA IDENTIFICAÇÃO DE PONTOS COM PRODUTOS SEM GLÚTEN OU SEM LACTOSE

RIO DO SUL
2022

CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ - UNIDAVI

EDUARDO FERTIG BASTOS

PROTÓTIPO DE APLICAÇÃO *WEB* PARA IDENTIFICAÇÃO DE PONTOS COM PRODUTOS SEM GLÚTEN OU SEM LACTOSE

Trabalho de Conclusão de Curso a ser apresentado ao curso de Sistemas da Informação, da Área das Ciências Naturais, da Computação e das Engenharias, do Centro Universitário para o Desenvolvimento do Alto Vale do Itajaí, como condição parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

Prof. Orientador: M.e Fernando Andrade Bastos

RIO DO SUL
2022

CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ - UNIDAVI

EDUARDO FERTIG BASTOS

PROTÓTIPO DE APLICAÇÃO *WEB* PARA IDENTIFICAÇÃO DE PONTOS COM PRODUTOS SEM GLÚTEN OU SEM LACTOSE

Trabalho de Conclusão de Curso a ser apresentado ao curso de Sistemas da Informação, da Área das Ciências Naturais, da Computação e das Engenharias, do Centro Universitário para o Desenvolvimento do Alto Vale do Itajaí- UNIDAVI, a ser apreciado pela Banca Examinadora, formada por:

_____

Professor Orientador: M.e Fernando Andrade Bastos

Banca Examinadora:

_____

Prof. M.e Marcondes Maçaneiro

_____

Prof. M.e Jeancarlo Visentainer

Rio do Sul, 28 de novembro de 2022.

“Toda a ação é designada em termos do fim que procura atingir. ” (Nicolau Maquiavel)

Dedico este trabalho a toda a minha família, que em nenhum momento duvidou de mim, e muitas vezes acreditaram mais na minha pessoa do que eu mesmo. Dedico também ao meu avô Zelio Bastos, o qual sinto muitas saudades.

AGRADECIMENTOS

Gostaria de agradecer a Deus por ter possibilitado todo esse caminho, desde antes do ingresso à universidade, como durante os anos cursados.

Agradeço infinitamente a minha família, por sempre acreditarem em mim e terem possibilitado este processo. Por toda a confiança, paz e tranquilidade que me passaram nos momentos difíceis, momentos acadêmicos ou não. Obrigado, mãe, por sempre me ensinar e a tratar momentos complicados com leveza. Obrigado, Aninha, por ser sempre um porto seguro para qualquer desabafo, qualquer conversa, para tudo. Sei que posso contar com vocês para tudo. Amo vocês.

Agradeço demais aos meus amigos, os quais tornaram essa jornada mais alegre e divertida, alegrando os momentos ruins e tornando momentos bons em momentos de pura felicidade.

Agradeço aos meus colegas de turma, amigos também, por possibilitarem muita troca de conhecimento, incentivado a explorar novas tecnologias e compartilhar as mais diferentes teorias e práticas nos jogos de Truco.

Agradeço à Unidavi, e ao Colégio Universitário Unidavi, bem como ao Colégio Sinodal Ruy Barbosa e ao Colégio Atlântico, a todos os professores com os quais eu pude aprender diversos assuntos, sejam didáticos ou experiências de vida, os quais se tornaram um pedaço de mim.

Não tenho palavras para poder agradecer o tanto como eu gostaria ao meu orientador, coordenador e pai Me. Fernando Andrade Bastos, que me incentivou a realizar este curso, me auxiliou durante o curso e que também pode me orientar nesse trabalho de conclusão. Muito atencioso e sempre disposto a ensinar independente do momento, na universidade e em casa. Obrigado pai, você foi e é muito importante para mim, como acadêmico e como pessoa. Te amo.

RESUMO

O setor alimentício para intolerantes – glúten e lactose, evoluiu bastante nos últimos anos, porém ainda assim há enorme dificuldade em localizar novos pontos de venda em que se disponibilizem estes produtos. Este trabalho tem como objetivo desenvolver um protótipo de aplicação web responsiva procurando disponibilizar um mapa com pontos alimentícios que ofertem produtos para intolerantes. Para a metodologia, foi realizada uma pesquisa aplicada e descritiva. Para concretizar a realização de todos os objetivos foi efetuada uma revisão da literatura sobre a engenharia de software, banco de dados e as linguagens de programação utilizadas. Aplicou-se uma pesquisa para procurar soluções similares a fim de se desenvolver o estado da arte. Para separar o desenvolvimento do protótipo, foram criadas duas seções, a primeira, a análise, que envolveu o levantamento de requisitos funcionais, não funcionais e suas regras de negócio. Também foram desenvolvidos diagrama de caso de uso, diagrama de atividade e um diagrama de entidade-relacionamento. A seção de implementação relatou os detalhes sobre as técnicas e ferramentas utilizadas, além do uso e do funcionamento do protótipo. Foram apresentadas todas as telas da aplicação, mostrando as versões para computador, tablet e celular, em virtude da responsividade. Além disso, demonstrou-se as estruturas de pastas utilizadas no *back-end* e no *front-end* da aplicação e também foram mostrados os detalhes sobre alguns componentes utilizados na aplicação como o Input e o Dropzone.

Palavras-Chave: desenvolvimento web, geolocalização, alimentos para intolerantes.

ABSTRACT

The special diet food sector, has evolved a lot in recent years, but there is still some difficulty in locating new points of sale where these products are available. This work aims to develop a responsive web application prototype seeking to provide a map with food points that offer products for special diet people. For the methodology, an applied and descriptive research was carried out. In order to achieve all the objectives, a literature review was carried out on software engineering, database and the programming languages used. A research was applied to look for similar applications in order to develop the state of the art. To separate the development of the prototype, two sections were created, the first one, which is the analysis, involved gathering functional and non-functional requirements and their business rules. A use case diagram, an activity diagram and an entity-relationship diagram were also developed. The implementation section reported details about the techniques and tools used, in addition to the use and functioning of the prototype. All screens of the application were presented, showing the versions for computer, tablet and cell phone due to responsiveness. The folder structures used in the back-end and front-end of the application were presented and details about some components used in the application such as Input and Dropzone were also shown.

Keywords: web development, geolocation, food for the intolerant.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JS	JavaScript
JSON	JavaScript Object Notation
JWT	JSON Web Token
LGPD	Lei Geral de Proteção de Dados Pessoais
ORM	Object Relational Mapper
SGBD	Sistema Gerenciador de Banco de Dados
SQL	Structured Query Language
TS	TypeScript

SUMÁRIO

1. INTRODUÇÃO

É perceptível a evolução da tecnologia nas últimas décadas, tanto a nível de hardware como de *software*. A todo momento, são criados novos equipamentos ou desenvolvidos novos *software*s que modificam e facilitam a vida humana nos mais diversos campos do saber.

Do ponto de vista tecnológico, uma das áreas que mais evoluiu nos últimos anos, foi a saúde. Independente da pandemia provocada pelo COVID-19, por vários anos observa-se o incremento de máquinas digitais e *software*s com inteligência artificial ou ainda, sistemas especialistas auxiliando os profissionais médicos no processo de facilitação diagnóstica.

Porém, mesmo assim, algumas temáticas apresentam evolução aquém do esperado. A intolerância ao glúten, ou simplesmente doença celíaca, com certeza entra nesta lista. Descoberta em 1888, a doença celíaca teve avanço do diagnóstico somente após cinco décadas, de forma empírica, durante a segunda guerra mundial. No final do século passado, o diagnóstico avançou, porém até o momento não existe cura. O problema foi amenizado com o tratamento que indica a retirada do glúten da alimentação.

Por outro lado, o tratamento a intolerância à lactose se apresenta um pouco mais evoluído, pois já existem medicamentos que auxiliam no processo de melhora do paciente.

As pessoas acometidas por estas doenças necessitam de um tratamento alimentar condizente. Entretanto, possuem uma dificuldade absurda para encontrar supermercados, comércios e lanchonetes que tenham no seu *mix* de produtos uma variedade adequada para aquisição por celíacos.

Desta forma, este trabalho propõe o desenvolvimento de um protótipo *web* que facilite a procura de pontos de venda destes produtos, amenizando o problema anteriormente citado.

1.1 PROBLEMA DE PESQUISA

Como solucionar as dificuldades de se localizar produtos para celíacos e produtos para intolerantes à lactose no comércio alimentício de forma virtual?

1.2 OBJETIVOS

1.2.1 Geral

Desenvolver um protótipo de aplicação *web* com o intuito de mapear pontos que disponibilizem a venda de produtos sem glúten e sem lactose.

1.2.2 Específicos

- Definir as tecnologias utilizadas para a construção do protótipo;
- Identificar *software*s já existentes no mercado que possuem uma finalidade semelhante e explicar seu funcionamento;
- Elaborar os requisitos e os diagramas visando a construção da análise do protótipo;
- Aplicar as tecnologias selecionadas para o desenvolvimento do protótipo.

1.3 JUSTIFICATIVA

Frequentemente, para celíacos e intolerantes à lactose, há uma dificuldade em se localizar pontos de venda que disponibilizem produtos que se adequem às suas doenças. O incentivo ao avanço tecnológico, assim como a análise de diversas carências em nossa sociedade, tal como a falta de variedade no comércio de produtos sem glúten e sem lactose levam a uma grande dificuldade para encontrar esses produtos. Para descobrir se uma loja vende produtos sem glúten, faz-se necessário entrar na loja, perguntar ao atendente ou ainda, pesquisar individualmente sobre cada unidade de produto, o que torna o processo moroso e sofrível.

Uma aplicação que procura destacar em um mapa, pontos/lojas com produtos para consumo que se encaixem nos requisitos de produto sem glúten ou sem lactose ajudaria muito na relação comercial entre consumidor, empresas e fornecedores. Facilitaria bastante a procura, poupando tempo e energia por parte do usuário, além de redirecionar os clientes para o nicho desejado mais facilmente, evitando que a empresa entre no ostracismo, aumentando seus lucros.

Desta forma, foi desenvolvido um protótipo *web* que facilite a busca por locais de venda, auxiliando os consumidores a encontrar os produtos, bem como possibilitando a ampliação dos resultados financeiros das empresas, visto que conseguirão atingir um nicho mercadológico com maior facilidade.

Como mais um incentivo, que evidencia a importância desse projeto, é possível relatar que este protótipo é um facilitador na pesquisa por pontos de venda que contribui na busca de produtos para pessoas intolerantes à lactose e ao glúten. Assim, essas pessoas passam a ter uma ferramenta *web* que disponibiliza a consulta com mais rapidez, eficiência e precisão.

2. REFERENCIAL TEÓRICO

Neste capítulo estão relatados os referenciais teóricos relacionados ao desenvolvimento de *software*, engenharia de *software*, linguagens de programação, *frameworks* e bibliotecas que foram utilizadas no desenvolvimento do protótipo e que são abordadas a seguir.

2.1 DESENVOLVIMENTO DE SOFTWARE

Desenvolver um *software* é uma arte que envolve vários conhecimentos no âmbito da tecnologia, da gestão e de recursos humanos.

Do ponto de vista da tecnologia, Valente (2020) declara que durante a construção de um *software* existe uma diversidade de escolhas que devem ser tomadas, desde a definição de estrutura e como os dados serão manipulados até a escolha da linguagem, *framework* e bibliotecas que constituirão o *software*.

2.2 ENGENHARIA DE SOFTWARE

Pressman (2006, p. 9) afirma que “Independente do tamanho, complexidade ou domínio da aplicação, o *software* de computador vai evoluir com o tempo.”.

A importância da análise de sistemas e dos métodos de desenvolvimento de *software* advém do fato de que eles definem, por meio de suas notações, um canal de comunicação uniforme entre os membros da equipe de desenvolvimento. Também, os métodos estabelecem produtos de trabalho padronizados que facilitam as atividades de manutenção de *software*. Além disso, permitem que novos colaboradores sejam treinados melhorando a qualidade do *software*. Os métodos podem ser usados como referência para a escolha e aquisição de ferramentas *CASE* que podem dar maior produtividade ao processo de desenvolvimento de *software*. (HIRAMA, 2011, p. 11).

De acordo com Pressman (2006), estudos mostraram que o *software*, assim como todos os sistemas complexos, evolui durante um período de tempo e os requisitos do negócio e do produto mudam frequentemente à medida que o desenvolvimento prossegue, dificultando um caminho direto para um produto final.

Segundo Hirama (2011), a atividade de Engenharias de Sistemas tem como um de seus objetivos entender as necessidades de negócio de um cliente, a partir disso são estabelecidos requisitos de um ponto de vista sistêmico, até ser possível definir em alto nível o que será desenvolvido. Uma das tarefas é a identificação de funções, restrições e desempenho do

sistema, uma vez que o objetivo é executar um método, procedimento ou realizar o processamento de dados.

2.3 BANCO DE DADOS

"Um banco de dados é uma coleção de dados persistentes, usada pelos sistemas de aplicação de uma determinada empresa. " (DATE, 2004, p.10).

Date (2004) menciona as principais vantagens de se utilizar um banco de dados. Estas são listadas na Quadro 1.

Quadro 1 – Vantagens de bancos de dados sobre métodos tradicionais

Característica	Descrição
Densidade	Não há necessidade de arquivos de papel, possivelmente volumosos.
Velocidade	A máquina pode obter e atualizar dados com rapidez muito maior que o ser humano.
Menos trabalho monótono	Grande parte do tédio de manter arquivos à mão é eliminada. As tarefas mecânicas são sempre feitas com melhor qualidade por máquinas.
Atualidade	Informações precisas e atualizadas estão disponíveis a qualquer momento sob consulta.
Proteção	Os dados podem ser mais bem protegidos contra perda não intencional e acesso ilegal.

Fonte: Elaborado a partir de Date (2004).

2.3.1 Sistema Gerenciador de Banco de Dados (SGBD)

Date (2004), define um SGBD como um *software* que gerencia todo e qualquer acesso ao banco de dados. O funcionamento do SGBD ocorre em 4 etapas, apresentadas no Quadro 2.

Quadro 2 – Etapas de funcionamento

Etapa	Descrição
1	É feita uma requisição de acesso, por parte do usuário.
2	O SGBD captura e analisa o pedido em questão.
3	Então, o SGBD inspeciona todas as partes necessárias como: esquema externo, mapeamento externo, esquema conceitual, mapeamento interno e definição do banco de dados armazenado.
4	Por fim, o SGBD executará as operações necessárias no banco de dados.

Fonte: Elaborado a partir de Date (2004).

2.3.2 PostgreSQL

O PostgreSQL é um banco de dados cujo desenvolvimento foi realizado pelo Departamento de Ciência da Computação da Universidade da Califórnia em Berkeley e foi

inovador em diversos conceitos que atualmente são comuns nos principais bancos de dados. O PostgreSQL também é um Sistema Gerenciador de Banco de Dados Relacional (SGBDR), ou seja, é um sistema que armazena e gerencia os dados através de relações. Uma relação é um termo matemático que descreve uma tabela. (POSTGRESQL, 2022).

> Sistema Gerenciador de Banco de Dados (SGDB) – O SGDB permite que banco de dados "persistentes" sejam concorrentemente partilhados por muitos usuários e aplicações utilizando manuseio de armazenamento e estratégias de otimização. Precisamos manter os dados para executar as tarefas de leitura a partir de banco de dados, de atualização e de inserção dos dados no banco de dados, preservando a integridade de dados. (NEVES, 2002, p.17).

As funcionalidades gerais do PostgreSQL podem ser observadas no Quadro 3.

Quadro 3 – Funcionalidades gerais do PostgreSQL

Funcionalidade	Descrição
Chaves Estrangeiras	Campo ou conjunto de campos em uma tabela que associa a uma instância de outra tabela, criando um relacionamento entre as tabelas. Caracteriza-se de uma coluna que faz referência à chave primária de outra tabela.
Triggers	É uma especificação de que o banco de dados deve executar automaticamente uma função específica sempre que um determinado tipo de operação for executado. Os *triggers* podem ser definidos para serem executados antes ou depois de qualquer operação *INSERT*, *UPDATE* ou *DELETE*, uma vez por linha modificada ou uma vez por instrução SQL. Se um evento *trigger* ocorrer, a função do *trigger* será chamada no momento apropriado para lidar com o evento.
Views	São pseudo-tabelas geradas a partir de uma instrução SQL que representam um subconjunto de dados presentes em tabelas reais.
Integridade Transacional	Corresponde as quatro propriedades básicas de integridade transacional, conhecidas como ACID: Atomicidade, Consistência, Isolamento e Durabilidade.
Queries	Processo de recuperação ou o comando para recuperar dados de um banco de dados. Em SQL, o comando *SELECT* é usado para especificar essas consultas.
Tipos de dados	Suporta os tipos de dados SQL padrão *int*, *smallint*, real, *double precision*, *char* (N), *varchar* (N), date, time, *timestamp* e *interval*, assim como outros tipos de utilidade geral, e um conjunto diversificado de tipo geométricos.
Funções e Operadores	O PostgreSQL possui uma variedade ampla de funções e operadores prontos para os mais variados tipos de dados. Também é possível definir funções e operadores próprios para cada tipo de uso. Alguns dos tipos de funções e operadores do PostgreSQL são: lógicos; de comparação; matemáticos; de texto; de texto binário; de texto em bit; de data; geométricos; de e-mail e de vetores;
Funções de agregação	Calculam um único valor de resultado de um conjunto de valores de entrada. As funções de agregação internas são: *avg*; *bit_and*; *bit_or*; *bool_and*; *bool_or*; *count*; *every*; *max*; *min* e *sum*;
Métodos de Índice	São índices que permitem que a tabela seja indexada, facilitando e acelerando o processo de consulta em um banco de dados, evitando a necessidade de varrer toda a tabela para se obter um único valor.
Linguagens Procedurais	Trata-se da possibilidade que o PostgreSQL dá ao usuário de definir funções que sejam escritas em outras linguagens além do SQL e C. Essas outras linguagens são conhecidas como linguagens procedurais. O banco de dados solicita a instrução para o manipulador capacitado que analisará e executará o algoritmo retornando os resultados desejados.

Fonte: Elaborado a partir de PostgreSQL (2022).

O PostgreSQL, na sua versão 14, apresenta uma série de novos recursos e funcionalidade aos usuários, tornando o desenvolvimento mais rápido, melhorando o desempenho e geralmente tornando o PostgreSQL ainda mais utilizável.

Segundo Neves (2002, p. 17) “É um dos melhores bancos de dados para sistema operacional GNULIUX, voltado para aplicações simples ou complexas com uma robustez excepcional, suporte a várias linguagens, como C, Tcl, SQL, etc. O PostgreSQL, permite herança de tabelas e um banco relacional-objeto.”.

> A SQL é uma linguagem padrão para lidar com bancos de dados relacionais, e é aceita por quase todos os produtos existentes no mercado. Inclui operações de definição de dados e operações de manipulação de dados. A SQL é simplesmente um conjunto de instruções que manipula um banco de dados. (NEVES, 2002, p.17).

Neves (2002) afirma que o banco de dados relacional é matematicamente consistente internamente, perfeito e completo. Essas qualidades o tornam muito mais utilizado comercialmente.

2.3.3 Modelagem

Para Coronel e Rob (2011), a modelagem de dados é a primeira etapa do projeto, é criada uma modelagem de dados específica para um determinado tipo de problema. Segundo Coronel e Rob (2011) não se pode superestimar a modelagem de dados, pois os dados constituem as unidades de informação mais elementares utilizadas por um sistema.

> A modelagem de dados é um processo iterativo e progressivo. Começa-se com uma compreensão simples do domínio do problema e, conforme essa compreensão de desenvolver, o nível de detalhes do modelo também se amplia, se realizado de maneira adequada, o modelo de dados final é efetivamente uma “planta” que contém todas as instruções para a construção de um banco de dados que atenda às necessidades dos usuários finais. Essa planta é de natureza narrativa e gráfica, o que significa que contém tanto descrições de texto em linguagem direta e sem ambiguidades como diagramas úteis que ilustrem os principais elementos de dados. (CORONEL; ROB, 2011, p.31).

Barboza e Freitas (2018) explicam os tipos de entidades: entidades, entidades físicas e as entidades lógicas, conforme apresentado no Quadro 4.

Quadro 4 – Tipos de Entidades

Entidades	Correspondem aos objetos envolvidos em um ambiente. Esses objetos irão realizar conexões com os outros objetos para trocar informações necessárias para o correto funcionamento do sistema que será construído. As entidades podem ser divididas em físicas ou lógicas, a depender da sua existência, ou não, no mundo real.
Entidades físicas	Como exemplo de entidades físicas, temos clientes, fornecedores, produtos, colaboradores, inventário, etc. Perceba que, aqui, por se tratar de entidades físicas, todas têm existência no mundo real. Um cliente ou fornecedor pode ser uma pessoa ou ainda uma empresa; um produto pode ser uma roda de carro, um medicamento ou uma cadeira.
Entidades lógicas	As entidades lógicas são aquelas que existem geralmente "em decorrência da interação entre ou com entidades físicas, que fazem sentido dentro de um certo domínio de negócios, mas que no mundo externo/real não são objetos físicos", isto é, que ocupam lugar no espaço (RODRIGUES, 2014, documento on-line). São exemplos disso uma venda ou uma classificação de um objeto (modelo, espécie, função de um usuário do sistema). Essa necessidade se dá em virtude dos relacionamentos ou da classificação sobre as outras entidades ou conexões entre elas. Exemplos de entidades lógicas podem ser grupos de usuários do sistema, venda, relatório.

Fonte: Elaborado a partir de Barboza e Freitas (2018).

2.4 HTML

De acordo com Oliveira (2021), a linguagem HTML é a base utilizada para qualquer página *web*, trata-se de uma linguagem composta por elementos estáticos e está fortemente focada na composição e na formatação de documentos.

Oliveira (2021, p. 16) relata que o "HTML é a sigla para *HyperText Markup Language* ou Linguagem de Formatação de Hipertexto. De modo geral, são documentos de texto escritos em códigos que podem ser interpretados pelos navegadores que, por sua vez, exibem as páginas devidamente formatadas.".

A linguagem, porém, basicamente realiza a formatação de documentos, mas não apresenta os conceitos das linguagens de programação, como variáveis e estruturas de controle, entre outros. Dessa forma, para se criarem aplicações mais complexas e dinâmicas, é necessário o uso de outra linguagem de programação que complemente o HTML (OLIVEIRA, 2021, p. 19).

2.5 CSS

De acordo com Silva (2008, p.84) as "[...] folhas de estilo em cascata (ou CSS – *Cascading Style Sheets*) mudam a forma de organização das páginas. O HTML passa a ser utilizado somente como elemento para estruturar as páginas, e o CSS é utilizado na formatação da aparência das páginas.".

As Folhas de Estilo em Cascata (CSS ou *Cascading Style Sheets*) permitem customizar a formatação dos elementos HTML. Dessa forma, ampliam as possibilidades de exibição das páginas e oferecem bastante versatilidade para a diagramação das páginas e a construção da identidade visual de um site. Outra vantagem de adotar CSS em sites é a possibilidade de separar o conteúdo das páginas de sua formatação, o que facilita o processo de manutenção e atualização das páginas (OLIVEIRA, 2020).

Como relatado por Silva (2008), as folhas de estilo adicionam a possibilidade de gerar estilos personalizados para títulos, listas, imagens, ou quaisquer outros elementos, abrindo possibilidade para permitir a definição de cores, fontes, bordas, alinhamentos, entre outras características vinculadas à aparência das páginas *Web*.

2.6 JAVASCRIPT

Flanagan (2014, p. 18) afirma que "[...] JavaScript é uma linguagem de alto nível, dinâmica, interpretada e não tipada, conveniente para estilos de programação orientados a objetos e funcionais.".

> JavaScript é a linguagem de programação da *Web*. A ampla maioria dos sites modernos usa JavaScript e todos os navegadores modernos – em computadores de mesa, consoles de jogos, tablets e smartphones – incluem interpretadores JavaScript, tornando-a a linguagem de programação mais onipresente da história. JavaScript faz parte da tríade de tecnologias que todos os desenvolvedores *Web* devem conhecer: HTML, para especificar o conteúdo de páginas *Web*; CSS, para especificar a apresentação dessas páginas; e JavaScript, para especificar o comportamento delas. (FLANAGAN, 2014, p.18).

De acordo com Flanagan (2014) apresentam o JavaScript apresentou diversos nomes e versões ao longo do tempo, sendo essa estrutura formada pelo Quadro 5.

Quadro 5 - Nomes e Versões do JavaScript

Criação	JavaScript foi criada na Netscape na fase inicial da *Web* e, tecnicamente, "JavaScript" é marca registrada licenciada pela Sun Microsystems (agora Oracle), usada para descrever a implementação da linguagem pelo Netscape (agora Mozilla). A *Netscape* enviou a linguagem para a ECMA – *European Computer Manufacturer's Association* – para padronização e, devido a questões relacionadas à marca registrada, a versão padronizada manteve o nome estranho "ECMAScript". Pelos mesmos motivos ligados à marca registrada, a versão da Microsoft da linguagem é formalmente conhecida como "JScript". Na prática, quase todo mundo chama a linguagem de JavaScript. Este livro usa o nome "ECMAScript" apenas para se referir ao padrão da linguagem.
Versões	Na última década, todos os navegadores *Web* implementaram a versão 3 do padrão ECMAScript e não havia necessidade de se pensar em números de

	versão: o padrão da linguagem era estável e as implementações dos navegadores eram, na maioria, interoperáveis. Recentemente, uma importante nova versão da linguagem foi definida como ECMAScript versão 5. Às vezes, você vai ver essas versões da linguagem abreviadas como ES3 e ES5, assim como às vezes vai ver o nome JavaScript abreviado como JS.
Interpretadores	Quando falamos da linguagem em si, os únicos números de versão relevantes são ECMAScript versões 3 ou 5. (A versão 4 da ECMAScript esteve em desenvolvimento por anos, mas se mostrou ambiciosa demais e nunca foi lançada). Contudo, às vezes você também vai ver um número de versão de JavaScript, como JavaScript 1.5 ou JavaScript 1.8. Esses são números da versão do Mozilla: a versão 1.5 é basicamente a ECMAScript 3 e as versões posteriores incluem extensões não padronizadas da linguagem. Por fim, também existem números de versão vinculados a interpretadores ou "engines" de JavaScript específicos. O Google chama seu interpretador JavaScript de V8, por exemplo, e quando este livro estava sendo produzido a versão corrente do mecanismo V8 era a 3.0.

Fonte: Adaptado de Flanagan (2014).

"O uso de JavaScript em documentos *Web* normalmente deve ser controlado e moderado. O papel apropriado de JavaScript é melhorar a experiência de navegação do usuário, tornando mais fácil obter ou transmitir informações.". (FLANAGAN, 2014, p. 317).

> É interessante pensarmos nas páginas que exibimos nos navegadores Web. Algumas delas apresentam informações estáticas e podem ser chamadas de documentos. (A apresentação dessas informações pode ser bastante dinâmica – por causa de JavaScript –, mas as informações em si são estáticas.) Outras páginas Web mais parecem aplicativos do que documentos. Essas páginas podem carregar novas informações dinamicamente, conforme a necessidade, podem ser gráficas em vez de textuais e podem funcionar off-line e salvar dados de forma local, para que possam restaurar seus estados quando você visitá-las novamente. Ainda outras páginas Web ficam em algum ponto no meio desse espectro e combinam recursos de documentos e de aplicativos. (FLANAGAN, 2014, p. 314).

De acordo com Flanagan (2014), a experiência do usuário não depende totalmente do JavaScript, porém ela facilitará o processo, como criando animações e efeitos visuais que guiam o usuário para ajudar na navegação da página, ordenando colunas em uma tabela, tornando mais fácil para o usuário a disposição dos dados que mais necessita e ocultar algum conteúdo, revelando-o gradativamente à medida que o usuário se aprofunda no determinado conteúdo.

> Para realmente entender os aplicativos *Web*, é importante perceber que os navegadores *Web* foram muito além de sua função original como ferramentas para exibir documentos e se transformaram em sistemas operacionais simples. Considere o seguinte: um sistema operacional tradicional permite organizar ícones (que representam arquivos e aplicativos) na área de trabalho e em pastas. Um navegador *Web* permite organizar *bookmarks* (que representam documentos e aplicativos *Web*) em uma barra de ferramentas e em pastas. Um sistema operacional executa vários aplicativos em janelas separadas; um navegador *Web* exibe vários documentos (ou aplicativos) em guias (ou abas) separadas. Um sistema operacional define APIs de baixo nível para conexão em rede, desenho de elementos gráficos e salvamento de

> arquivos. Os navegadores *Web* definem APIs de baixo nível para conexão em rede, salvamento de dados e desenho de elementos gráficos. (FLANAGAN, 2014, p. 317).

Segundo Flanagan (2014, p. 320) "Um arquivo JavaScript contém JavaScript puro, sem marcações <script> ou qualquer outro código HTML. Por convenção, os arquivos de código JavaScript têm nomes que terminam com .js.".

> Uma marcação <script> com o atributo *src* especificado se comporta exatamente como se o conteúdo do arquivo JavaScript especificado aparecesse diretamente entre as marcações <script> e </ script>. Note que a marcação de fechamento </script> é obrigatória em documentos HTML, mesmo quando o atributo *src* é especificado, e que não há qualquer conteúdo entre as marcações <script> e </script>. Em XHTML, pode-se usar a marcação de atalho <script/> nesse caso. (FLANAGAN, 2014, p. 320).

De acordo com Flanagan (2014), existem vantagens no uso do atributo *src*, sendo estas vantagens apresentadas no Quadro 6.

Quadro 6 - Vantagens no uso do atributo src

Arquivos HTML	Ele simplifica seus arquivos HTML, permitindo remover deles grandes blocos de código JavaScript – isto é, ajuda a manter conteúdo e comportamento separados.
Páginas *Web*	Quando várias páginas *Web* compartilham o mesmo código JavaScript, o uso do atributo *src* permite que você mantenha apenas uma cópia desse código, em vez de ter de editar cada arquivo HTML quando o código mudar.
Compartilhamento	Se um arquivo de código JavaScript é compartilhado por mais de uma página, ele só precisa ser baixado uma vez, pela primeira página que o utilizar – as páginas subsequentes podem recuperá-lo da cache do navegador.
URL	Como o atributo *src* recebe um URL arbitrário como valor, um programa JavaScript ou uma página *Web* de um servidor pode empregar o código exportado por outros servidores *Web*. Muitos anúncios na Internet contam com isso.
Carregamento de scripts	A capacidade de carregar scripts de outros sites nos permite levar as vantagens do uso da cache um passo adiante: o Google está promovendo o uso de URLs padrão bem conhecidos para as bibliotecas do lado do cliente mais comumente usadas, permitindo que o navegador coloque no cache uma única cópia para uso compartilhado por qualquer site. Vincular código JavaScript aos servidores do Google pode diminuir o tempo de inicialização de suas páginas *Web*, pois é provável que a biblioteca já exista no cache do navegador do usuário; porém, você precisa estar disposto a confiar em terceiros para fornecer código fundamental para seu site.

Fonte: Adaptado de Flanagan (2014).

"Evidentemente, JavaScript é mais pertinente a aplicativos *Web* do que a documentos *Web*. Ela aprimora documentos *Web*, mas um documento bem projetado vai continuar a funcionar mesmo com JavaScript desativada. " (FLANAGAN, 2014, p. 318).

2.7 TYPESCRIPT

Segundo Simas, Borges e Couto (2019) a linguagem TypeScript é muito utilizada em aplicações *web*. Possibilitando a construção de código em mais alto nível e sendo convertido no fim para JavaScript, que é uma linguagem suportada por diversos navegadores.

Para Simas, Borges e Couto (2019, p.2) "A linguagem TypeScript é um conjunto do JavaScript que permite uma melhor usabilidade ao programador que já está acostumado a trabalhar com orientação a objetos e faz sua conversão para o JavaScript de forma automática e segura".

Segundo Simas, Borges e Couto (2019) como nas outras linguagens, os principais tipos de dados são os booleanos, *string, arrays* e números.

Conforme a fala de Simas, Borges e Couto (2019) o tipo *any* é utilizado na linguagem Typscript quando o a variável pode receber vários tipos, muito comum quando se trata no ambiente da *Web*, podendo não saber o tipo da variável vinda de uma requisição.

2.8 FRAMEWORK EXPRESS

O Express é uma ferramenta para aplicativos da *web* que utilizam o NodeJS, possibilitando a utilização de vários métodos que auxiliam o HTTP e a utilização de *middleware*, facilitando a criação de uma API robusta de forma rápida e simples. (EXPRESS, 2022).

2.9 NODE.JS

Para NodeJS (2022), o NodeJS é uma biblioteca JavaScript, que é projetado para desenvolvimento de aplicações de rede, ele foi construído a partir do motor V8 do Google, assim garantindo que a linguagem JavaScript que antes era somente utilizada nos navegadores conseguisse ser utilizada para a criação de servidores.

Conforme V8 (2022), o V8 é um projeto do Google de código aberto que foi escrito na linguagem de C++, inicialmente era somente utilizado nos navegadores permitindo-os a compreender *scripts* feitos em JavaScript, posteriormente o V8 foi desacoplado dos navegadores, permitindo a criação do NodeJS que é executado em cima do V8, assim tornando possível os diversos sistemas operacionais compreender códigos feitos em JavaScript, viabilizando esta linguagem a ser utilizada no *back-end* das aplicações.

> O Node.js é um ambiente de servidor de código aberto que possibilita a execução de aplicações escritas em JavaScript e pode atuar, em aplicações para a Internet, como uma linguagem de programação server-side. Tecnicamente, consiste em um *runtime* JavaScript desenvolvido com o Chrome's V8 JavaScript *engine*. (OLIVEIRA, 2020, p.9)

Como relatado por Oliveira (2020, p. 9) o Node.js teve sua utilização ampliada em uma escala massiva no desenvolvimento de aplicações para a Internet.

2.10 TYPEORM

Para TypeORM (2022), "o TypeORM é um ORM que pode ser executado nas plataformas NodeJS, Browser, Cordova, PhoneGap, Ionic, React Native, NativeScript, Expo e Electron e pode ser usado tanto com o TypeScript, quanto com o JavaScript".

Também conforme TypeORM (2022), a biblioteca TypeORM é um ORM (Object Relational Mapper) que permite você integrar sua aplicação JavaScript, ou, TypeScript com um banco de dados relacional.

2.11 JSON WEB TOKEN

Conforme JSON Web Token (2022), "o JSON Web Token (JWT) é um padrão aberto (RFC 7519) que define uma maneira compacta e independente de transmitir informações com segurança entre as partes como um objeto JSON. Essas informações podem ser verificadas e confiáveis porque são assinadas digitalmente.".

> Embora os JWTs possam ser criptografados para também fornecer sigilo entre as partes, nos concentraremos em tokens assinados. Os tokens assinados podem verificar a integridade das declarações contidas nele, enquanto os tokens criptografados ocultam essas declarações de outras partes. Quando os tokens são assinados usando pares de chaves pública/privada, a assinatura também certifica que apenas a parte que possui a chave privada é a que a assinou. JSON Web Token (2022).

Segundo JSON Web Token "Os JWTs podem ser assinados usando um segredo (com o algoritmo HMAC) ou um par de chaves pública/privada usando RSA ou ECDSA".

> Depois que o usuário estiver conectado, cada solicitação subsequente incluirá o JWT, permitindo que o usuário acesse rotas, serviços e recursos permitidos com esse token. O Single Sign On é um recurso que usa amplamente o JWT hoje em dia, devido à sua

> pequena sobrecarga e sua capacidade de ser facilmente usado em diferentes domínios. JSON Web Token (2022).

O padrão JWT tornou-se o padrão mais usado em autenticação de API por praticamente toda a comunidade de desenvolvimento, devido a sua praticidade, segurança e velocidade.

2.12 MULTER

Segundo Multer (2022), “o Multer é um middleware node.js para manipulação de dados multipart/form-data, que é usado principalmente para fazer upload de arquivos. Está escrito em cima do busboy para máxima eficiência”.

2.13 REACT JS

Segundo React (2022) o React é uma biblioteca da linguagem JavaScript, que é muito eficiente e flexível que permite ao programador criar diversas interfaces, auxiliando na construção de sistemas complexos a partir de códigos pequenos e isolados, que são os denominados componentes.

Um componente React recebe alguns parâmetros que são passados através das propriedades do componente e são acessados através do atributo da classe “props”. Todo componente React necessita de um retorno. A classe possui um método “render” que é responsável por retornar o conteúdo que será exibido em tela, sendo ele HTML puro ou “JSX“. (REACT, 2022).

React (2022) explica que o JSX é uma extensão de sintaxe para JavaScript, é recomendado a utilização do JSX para similar como a interface deve parecer, JSX pode lembrar muito linguagens de *template*, mas que vem com todo o poder do JavaScript.

Segundo React (2022) a manipulação de eventos também é diferente do JavaScript convencional, ao invés de chamar uma função *onClick* passando o nome da função por meio de uma *string*, passa-se a função através da propriedade da *tag*, já para um componente React o comportamento de um evento é um pouco diferente: ao invés de passar uma *string* como atributo é possível passar a própria função, pois componentes JSX permitem usar comandos JavaScript dentro de *tags* HTML.

Para React (2022) umas das melhores adições foram os *hooks*, que foi implementado na versão React 16.8. Os *hooks* são formas mais rápidas e fáceis de utilizar funções do React, tornando a criação e utilização do state muito simples.

React (2022) ainda diz sobre a utilização dos *hooks* na criação de *effects*, que são basicamente funções executadas na criação do componente. O *hook* é executado quando uma das variáveis passadas no *array* de dependências de tal *hook* é modificada.

2.14 STYLED COMPONENTS

Conforme situado por Styled Components (2022) *styled components* permite a escrita de CSS real no código JavaScript. Portanto é possível usar todos os recursos do CSS como *media queries*, todos os pseudo-seletores, aninhamento, entre outros dentro do próprio JavaScript.

Segundo Styled Components (2022) " Styled Components remove o mapeamento entre componentes e estilos. Isso significa que quando você está definindo seus estilos, você está realmente criando um componente React normal, que tem seus estilos anexados a ele".

2.15 PROTOCOLO HTTP

Segundo Simas, Borges e Couto (2019) o *Hypertext Transfer Protocol* ou HTTP é simplesmente a base da tecnologia *web*, basicamente ele é responsável pela requisição feita pelo cliente para o servidor: pega a requisição feita pelo cliente e com base nela devolve ao mesmo os dados vindos do servidor.

Apesar de se tratar de um protocolo desenvolvido em 1989, ao passar dos anos o HTTP foi aprimorado com a incorporação de tecnologias mais recentes, afirmam Simas, Borges e Couto. (2019).

O MDN (2021) define o HTTP como um conjunto de métodos de requisições, cujo os mesmos serão responsáveis por determinar qual será a ação executada para tal método, como demonstra o Quadro 7.

Quadro 7 – Métodos HTTP

Método	**Descrição**
GET	O método GET solicita a representação de um recurso específico. Requisições utilizado o método GET devem retornar apenas dados.
HEAD	O método HEAD solicita uma resposta de forma idêntica ao método GET, porém sem conter o corpo da resposta.
POST	O método POST é utilizado para submeter uma entidade a um recurso específico, frequentemente causando uma mudança no estado do recurso ou efeitos colaterais no servidor.
PUT	O método PUT substituiu todas as atuais representações do recurso de destino pela carga de dados de requisição.
DELETE	O método DELETE remove um recurso específico.

CONNECT	O método CONNECT estabelece um túnel para o servidor identificado pelo recurso de destino.
OPTIONS	O método OPTIONS é usado para descrever as opções de comunicação com o recurso de destino.
TRACE	O método TRACE executa um teste de chamada *loop-back* junto com o caminho para o recurso de destino.
PATCH	O método PATCH é utilizado para aplicar modificações parciais em um recurso.

Fonte: Elaborado a partir de MDN (2021).

2.16 LEI GERAL DE PROTEÇÃO DE DADOS (LGPD)

"A Lei Geral de Proteção de Dados Pessoais (LGPD), Lei n° 13.709/2018, foi promulgada para proteger os direitos fundamentais de liberdade e de privacidade e a livre formação da personalidade de cada indivíduo. " (BRASIL, 2021a).

"A Lei fala sobre o tratamento de dados pessoais, dispostos em meio físico ou digital, feito por pessoa física ou jurídica de direito público ou privado, englobando um amplo conjunto de operações que podem ocorrer em meios manuais ou digitais. " (BRASIL, 2021a).

Existem alguns princípios para a lei geral de proteção de dados, sendo estes princípios apresentados no Quadro 8.

Quadro 8 - Princípios da LGPD

Finalidade	A realização do tratamento deve ocorrer para propósitos legítimos, específicos, explícitos e informados ao(à) titular, sem possibilidade de tratamento posterior de forma incompatível com essas finalidades.
Adequação	A compatibilidade do tratamento deve ocorrer conforme as finalidades informadas ao(à) titular, de acordo com o contexto do tratamento.
Necessidade	O tratamento deve se limitar à realização de suas finalidades, com abrangência dos dados pertinentes, proporcionais e não excessivos em relação às finalidades do tratamento de dados.
Livre acesso	É a garantia dada aos(às) titulares de consulta livre, de forma facilitada e gratuita, à forma e à duração do tratamento, bem como à integralidade de seus dados pessoais.
Qualidade dos dados	É a garantia dada aos(às) titulares de exatidão, clareza, relevância e atualização dos dados, de acordo com a necessidade e para o cumprimento da finalidade de seu tratamento.
Transparência	É a garantia dada aos(às) titulares de que terão informações claras, precisas e facilmente acessíveis sobre a realização do tratamento e os respectivos agentes de tratamento, observados os segredos comercial e industrial.
Segurança	Trata-se da utilização de medidas técnicas e administrativas qualificadas para proteger os dados pessoais de acessos não autorizados e de situações acidentais ou ilícitas de destruição, perda, alteração, comunicação ou difusão.
Prevenção	Compreende a adoção de medidas para prevenir a ocorrência de danos por causa do tratamento de dados pessoais.
Não discriminação	Sustenta que o tratamento dos dados não pode ser realizado para fins discriminatórios, ilícitos ou abusivos.

Responsabilização e prestação de contas	Demonstração, pelo Controlador ou pelo Operador, de todas as medidas eficazes e capazes de comprovar o cumprimento da lei e a eficácia das medidas aplicadas.

Fonte: Adaptado de Brasil (2021b).

A “Lei estabelece uma estrutura legal de direitos dos titulares de dados pessoais. Esses direitos devem ser garantidos durante toda a existência do tratamento dos dados pessoais realizado pelo órgão ou entidade. ” (BRASIL, 2021b).

3. METODOLOGIA DA PESQUISA

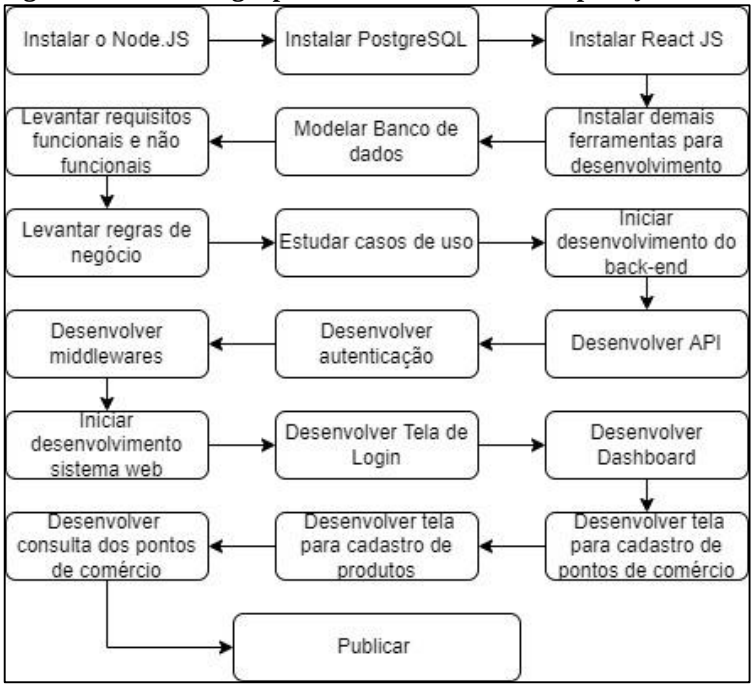
O presente trabalho se caracteriza como pesquisa aplicada e descritiva, pois foi desenvolvido um protótipo da aplicação para auxiliar o usuário na busca por estabelecimentos fornecedores de produtos sem glúten e sem lactose.

O trabalho buscou responder o seguinte problema: É possível desenvolver um aplicativo para auxiliar o usuário evidenciado estabelecimentos que fornecem produtos sem glúten ou produtos sem lactose?

Após a finalização do levantamento bibliográfico, foi realizado um estudo mais aprofundado sobre o desenvolvimento de aplicações *web* atualmente, utilizando os padrões de desenvolvimento e ferramentas mais modernas possíveis, buscando manter a aplicação relevante por mais tempo, evitando grande número de manutenções, migrações para novos padrões e facilitando a inserção de novos recursos.

Com base nessa pesquisa foi definido que a aplicação *web* seria feita utilizando a biblioteca React JS, mundialmente conhecida e amplamente utilizada. A aplicação usufrui de um *back-end* construído em Node.JS, onde se envia e recupera os dados através de uma API RESTful. Essas escolhas foram feitas por serem linguagens consolidadas no mercado e que possuem uma comunidade enorme, extremamente ativa e em evolução no número de usuários. O Fluxograma da aplicação está detalhado na Figura 1.

Figura 1 – Metodologia para desenvolvimento da aplicação.



Fonte: acervo do autor (2022)

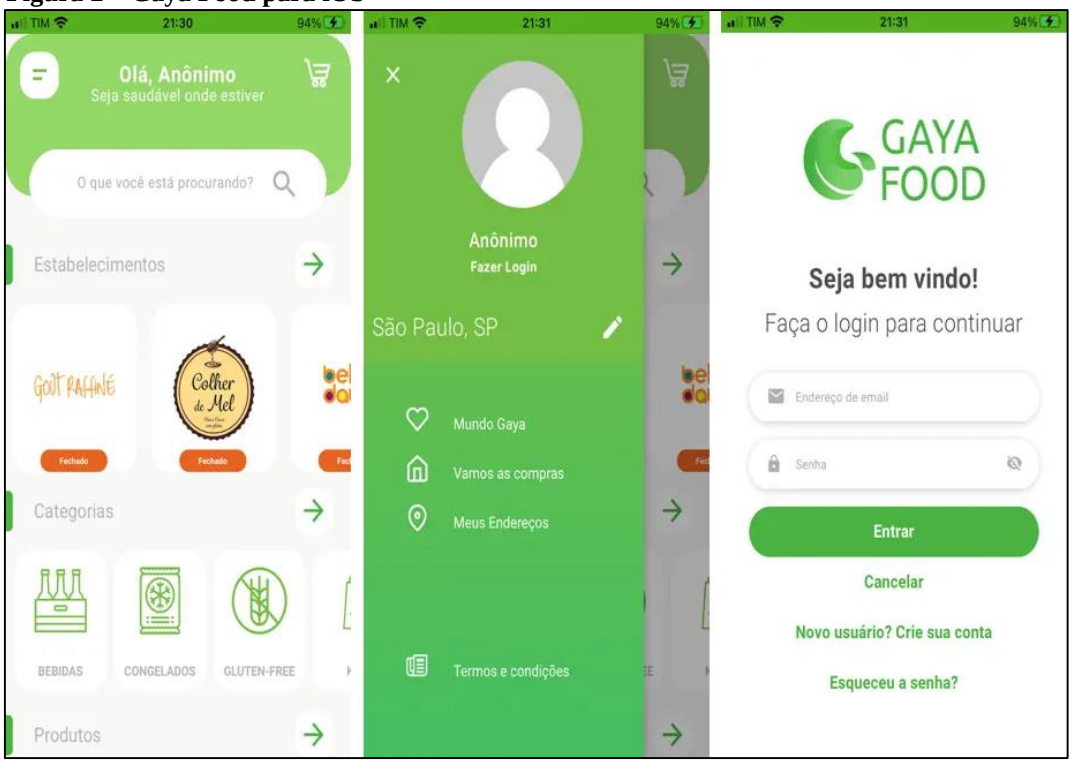
3.1 ESTADO DA ARTE

Nesta seção são descritas duas ferramentas já existentes no mercado que se assemelham às propostas do desenvolvimento desse projeto. As vantagens e as desvantagens dessas ferramentas são listadas na sequência com a finalidade de demonstrar algumas aplicações já existentes.

3.1.1 Gaya Food

O aplicativo Gaya Food é uma aplicação similar ao protótipo projetado. Oferece cadastro de restaurantes e o cadastro de produtos que se encaixam em grupos de intolerância. Possui a proposta de ser um delivery. Foi criado em 2018 e atualmente consta com apenas um estabelecimento cadastrado. A aplicação possui versões para Android e iOS, abrangendo a maioria dos dispositivos móveis disponíveis no mercado,

Figura 2 – Gaya Food para iOS

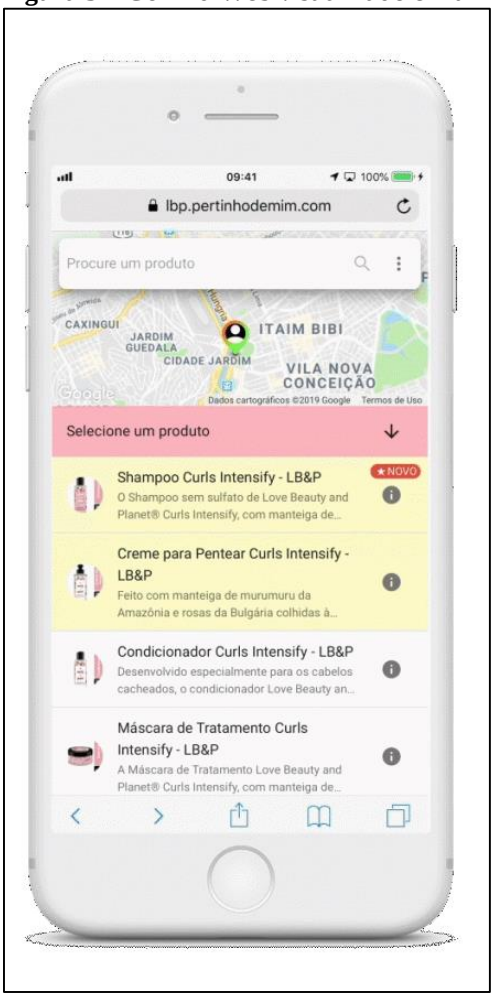


Fonte: Gaya Food (2022).

3.1.2 GoFind

A aplicação Gofind é um localizador de produtos *omnichannel* - estratégia de vendas que integra diferentes canais de comunicação e divulgação - que mapeia e mostra a disponibilidade de produtos em lojas físicas, online e *marketplaces* através de conexões com o Google Meu Negócio. Foi criada em 2016 e possui mais de 900.000 de estabelecimentos e 550.000 produtos cadastrados. A aplicação possui uma versão web e uma versão para Android, onde possui o nome de “OndeTem”. Não possui versão para iOS.

Figura 3 – GoFind Web visualizado em um iOS



Fonte: GoFind (2022).

3.1.3 Comparação do Protótipo com o Estado da Arte

Em consistência com o desenvolvimento deste projeto, no quadro 9 são apresentadas as diferenças dos recursos entre o projeto desenvolvido e as aplicações já listadas nesse tópico.

Quadro 9 - Recursos nos sistemas avaliados

Funcionalidade	Projeto Atual	Gaya Food	GoFind
Cadastro de Estabelecimentos.	X	X	X
Cadastro de Produtos.	X	X	X
Rastreamento de informações.			X
Mapa para visualização.	X		X
Autenticação para consulta.		X	
Aplicação Web.	X		X
Específico para alimentos inclusivos.	X	X	

Fonte: Acervo do autor (2022)

4. PROTÓTIPO DE APLICAÇÃO *WEB* PARA IDENTIFICAÇÃO DE PONTOS COM PRODUTOS SEM GLÚTEN OU SEM LACTOSE

Neste capítulo são abordados assuntos referentes a construção do protótipo, como a elaboração de requisitos, a análise do projeto e a sua implementação.

4.1 ANÁLISE

Conforme mencionado na revisão da literatura, a engenharia de software tem o propósito de ajudar a compreender as necessidades de negócio de um projeto. Com base nesse conceito, evidencia-se a importância de realizar a análise sobre o projeto e assim estruturar o seu processo de desenvolvimento, bem como compreender as regras de negócios e requisitos que existem no sistema, ajudando a minimizar os problemas e atuando como método de prevenção e facilitar a correção dos mesmos.

4.1.1 Visão Geral do Sistema

Este protótipo de aplicação tem como objetivo facilitar a divulgação de produtos para pessoas com intolerância como celíacos, intolerantes à lactose, diabéticos etc. Atualmente, intolerantes encontram dificuldade ao tentar localizar os produtos que desejam.

A aplicação foi desenvolvida com o propósito anteriormente exposto. Com um simples aplicativo o usuário consegue ter acesso às lojas e aos produtos que deseja, agilizando a busca, poupando tempo do usuário e a certeza de que encontrará o produto adequado, no local desejado.

Além da questão do melhor aproveitamento do tempo do usuário em relação aos métodos tradicionais de busca de produto, a aplicação tem um forte enredo baseado na acessibilidade, possibilitando o fácil acesso ao produto de quem tem uma alimentação inclusiva com uma dieta rígida e necessária.

O projeto consiste em uma aplicação web onde é possível cadastrar o estabelecimento, seus dados e sua geolocalização. Também é possível cadastrar os produtos com várias fotos, em determinadas categorias.

A aplicação disponibiliza um mapa onde se pode visualizar os estabelecimentos cadastrados na aplicação. Ao clicar na loja é possível visualizar os dados sobre a loja e todos

os seus produtos. É possível visualizar todos os produtos sem ordem pré-definida, ou selecionados por categoria.

4.1.2 Requisitos de Software

Nesta seção são documentadas as especificações de requisitos de software que estabelecem a base de como a aplicação deve funcionar.

4.1.2.1 Requisitos Funcionais

Neste tópico são documentadas as funcionalidades do sistema, para que o mesmo execute as operações como foi planejado. O Quadro 10, representa os requisitos funcionais do protótipo.

Quadro 10 - Requisitos Funcionais

Número	Nome	Descrição	Regras de Negócio
RF01	Login do Sistema	A aplicação web deve exigir o login do usuário para acesso ao sistema, sendo necessário informar o email e a senha.	
RF02	Logout do Sistema	A aplicação web deve permitir o usuário fazer logout na aplicação.	
RF03	Menu Principal	A aplicação web deve conter um menu, onde são apresentados os botões para acessos as páginas de dashboard, edição de perfil, site institucional do sistema, visualização de produtos, cadastro de produto e um botão para logout	
RF04	Tela de visualização do Mapa	A aplicação web deve ter uma tela onde é disponibilizado um mapa, no qual são carregados os estabelecimentos a um raio de 10km da localidade do usuário.	RN03
RF05	Tela Dashboard – Aplicação Web	A aplicação web deverá ter uma tela principal onde são apresentados todos os produtos que o usuário (estabelecimento) possui.	RN05
RF06	Tela de Login – Aplicação Web	A aplicação web deve ter uma tela onde permitirá o usuário realizar o login.	
RF07	Tela de Cadastro de Estabelecimento – Aplicação Web	A aplicação web deve ter uma tela onde permite o usuário realizar o cadastro do estabelecimento.	RN01, RN04, RN06, RN07, RN08
RF08	Tela de Cadastro de Produto – Aplicação Web	A aplicação web deve ter uma tela onde permite o estabelecimento realizar o cadastro do produto.	RN02, RN09, RN10
RF09	Tela institucional	O sistema deve possuir uma tela institucional para apresentar o sistema e suas funcionalidades para o usuário.	

RF10	Cadastro de Estabelecimentos	O sistema deve permitir o cadastro de estabelecimentos. Os campos necessários para o cadastro de usuários são: nome (obrigatório), CNPJ (obrigatório), senha (obrigatório), e-mail (obrigatório), avatar (não obrigatório), seu endereço (obrigatório) e telefones (pelo menos 1 telefone deve ser informado). O endereço é vinculado a uma tabela de endereços. Os telefones são vinculados a uma tabela de telefones.	RN01, RN06, RN07, RN08
RF11	Cadastro de Produtos	O sistema deve permitir o cadastro de produtos. Os campos necessários para o cadastro de produtos são: título (obrigatório), preço (obrigatório), categoria (obrigatório), tipo de produto (obrigatório), marca (obrigatório), estabelecimento (obrigatório) e fotos. A categoria é vinculada a uma tabela de categorias. O tipo de produto é vinculado a uma tabela de tipos de produtos. A marca é vinculada a uma tabela de marcas. O ponto é vinculado a uma tabela de pontos, sendo definido através do Token de autenticação. As fotos são vinculadas a uma tabela de fotos por produto.	RN09
RF12	Cadastro de Marcas	O sistema deve permitir o cadastro de marcas. O campo necessário para o cadastro é o seguinte: descrição (obrigatório).	
RF13	Cadastro de Categorias	O sistema deve permitir o cadastro de categorias. O campo necessário para o cadastro é o seguinte: descrição (obrigatório).	
RF14	Cadastro de Tipos de Produtos	O sistema deve permitir o cadastro de tipos de produtos. O campo necessário para o cadastro é o seguinte: descrição (obrigatório).	RN10
RF15	Cadastro de Endereços	O sistema deve permitir o cadastro de endereços. Os campos necessários para o cadastro de usuários são: cep (obrigatório), descrição (obrigatório), bairro (obrigatório), complemento (não obrigatório), número (obrigatório), latitude e longitude (obrigatórios se o usuário for um estabelecimento, senão não são obrigatórios) e cidade (obrigatório). A cidade é vinculada a uma tabela de cidades.	RN07
RF16	Cadastro de Cidades	O sistema deve permitir o cadastro de cidades. Os campos necessários para o cadastro são o seguinte: nome (obrigatório) e estado. O estado é vinculado a uma tabela de estados.	RN08
RF17	Cadastro de Estados	O sistema deve permitir o cadastro de estados. Os campos necessários são os seguintes: nome (obrigatório), sigla (obrigatório) e país. O país é vinculado a uma tabela de países.	
RF18	Cadastro de Países	O sistema deve permitir o cadastro de países. Os campos necessários são os seguintes: nome (obrigatório), sigla (obrigatório).	

RF19	Cadastro de Telefones	O sistema deve permitir o cadastro de telefones. Os campos necessários são os seguintes: código internacional (obrigatório), código regional (obrigatório) e número de telefone (obrigatório).	
RF20	Cadastro de fotos por produtos.	O sistema deve permitir o cadastro de fotos por produtos. Os campos necessários são os seguintes: produto (obrigatório) e foto (obrigatório). O produto é vinculado a uma tabela de produtos.	RN09
RF21	Cadastro de telefones por estabelecimentos.	O sistema deve permitir o cadastro de telefones por estabelecimento. Os campos necessários são os seguintes: telefone (obrigatório) e estabelecimento (obrigatório). O telefone é vinculado a uma tabela de telefones. O estabelecimento é vinculado a uma tabela de estabelecimentos.	

Fonte: Acervo do autor (2022)

4.1.2.2 Requisitos Não Funcionais

No Quadro 11, são descritos os requisitos não funcionais da aplicação, ou seja, os requisitos relacionados ao uso da aplicação e suas características.

Quadro 11 - Requisitos não funcionais

Número	**Descrição**
RNF01	O back-end deve ser desenvolvido com a versão 16 do Node.js ou superior.
RNF02	O front-end deve ser desenvolvido com a versão 18.2 do React JS ou superior.
RNF03	O sistema deve ser desenvolvido com a versão 14 do PostgreSQL ou superior.
RNF04	O sistema deve ser a prova de SQL *Injection*.
RNF05	O sistema deve ser responsivo, sendo acessível pelos principais navegadores do mercado.
RNF06	O sistema não pode possuir rotinas onde a requisição demore mais de 20 segundos para ser finalizada.
RNF07	O sistema deve retornar ao funcionamento normal em no máximo 4 horas em caso de falhas.
RNF08	O sistema deve possuir princípios de segurança da informação e estar aderente a LGPD.

Fonte: Acervo do autor (2022)

4.1.2.3 Regras de Negócio

No Quadro 12, estão descritas as regras de negócio da aplicação, ou seja, as regras que precisam ser implementadas para o correto desenvolvimento dos requisitos.

Quadro 12 - Regras de Negócio

Número	Descrição
RN01	A senha para acesso deve conter uma letra minúscula, uma letra maiúscula, um carácter especial e um número no mínimo.
RN02	O estabelecimento logado no sistema só pode cadastrar produtos do próprio estabelecimento.
RN03	O mapa deve carregar inicialmente os estabelecimentos localizados a 10km de distância da localização do usuário.
RN04	Ao informar o CEP no cadastro de estabelecimentos, a aplicação deve realizar uma requisição na API dos correios e carregar o logradouro, complemento, bairro, cidade e UF.
RN05	No *dashboard* do estabelecimento são apresentados todos os produtos cadastrados pelo próprio.
RN06	Não poderá haver dois estabelecimentos cadastrados com o mesmo e-mail.
RN07	Não poderá haver dois estabelecimentos cadastrados com o mesmo CNPJ.
RN08	O estabelecimento deve ter uma foto cadastrada.
RN09	O produto pode ter várias fotos cadastradas.
RN10	Após cadastrar um produto, o formulário deverá ser limpo para permitir inserir um novo produto em seguida.

Fonte: Acervo do autor (2022)

4.1.4 Diagramas

Nesta seção são apresentados os diagramas relacionados à utilização do sistema e sua engenharia de software.

4.1.4.1 Casos de Uso

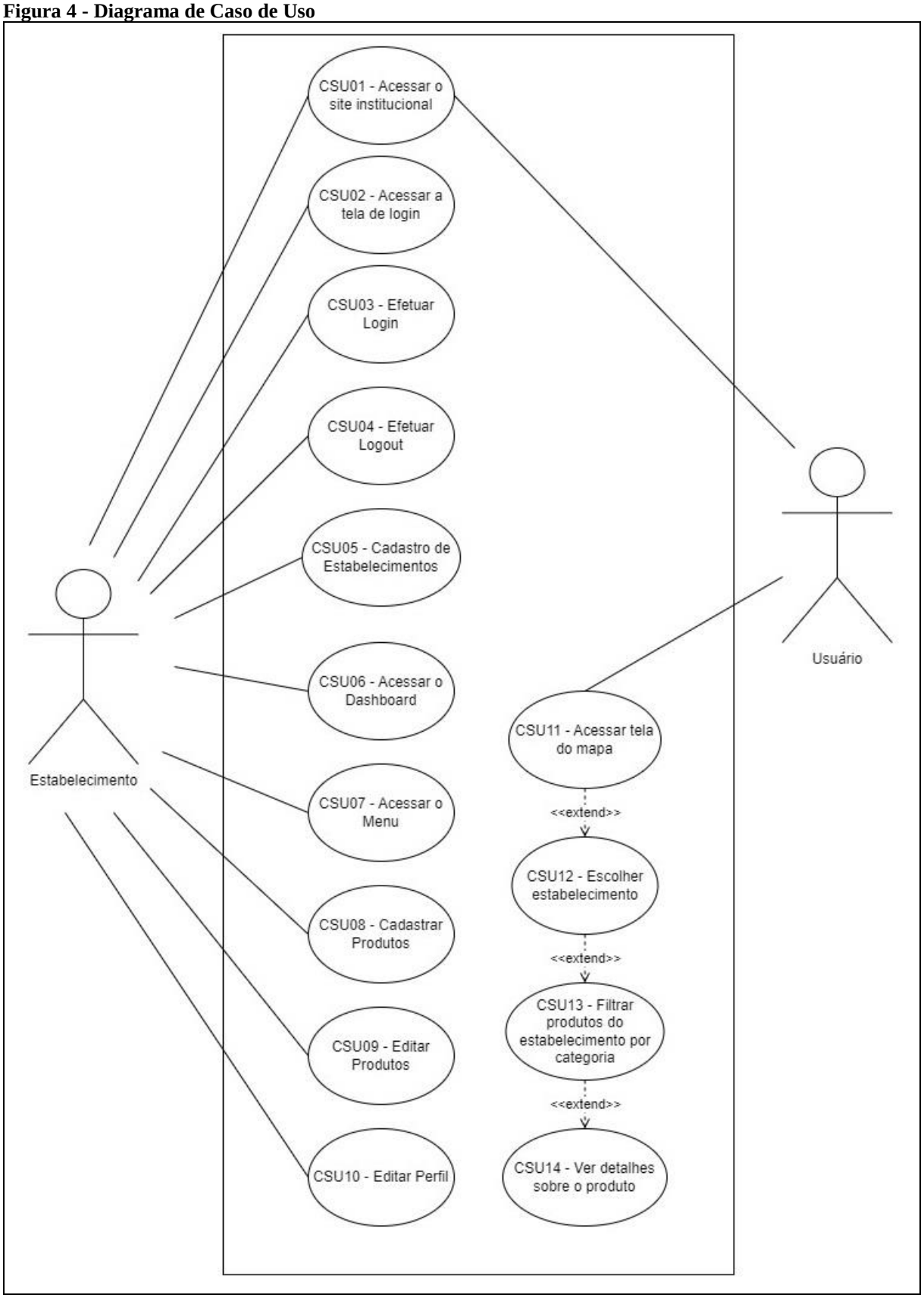
Na Figura 4 é apresentado o diagrama de caso de uso, onde o objetivo é demonstrar as funcionalidades do sistema, inclusive o que cada ator poderá executar dentro do protótipo.

Ao observar o diagrama de caso de uso se percebe que a aplicação disponibiliza login e logout para os estabelecimentos. Estes permitem acesso ao sistema, podendo realizar o seu cadastro para que o estabelecimento fique localizado no mapa e também possa cadastrar os seus produtos. Apenas os estabelecimentos têm acesso ao sistema, podendo editar seu perfil, utilizar do menu e acessar o seu dashboard.

Os usuários têm acesso ao mapa podendo se mover, observar e escolher os estabelecimentos. Ao clicar em um estabelecimento, são mostrados os produtos em ordem alfabética, que podem ser filtrados por categorias, nome e preço. Ao clicar em um produto é possível de ver os detalhes do produto.

Ambos usuários e estabelecimentos poderão acessar o site institucional.

Figura 4 - Diagrama de Caso de Uso

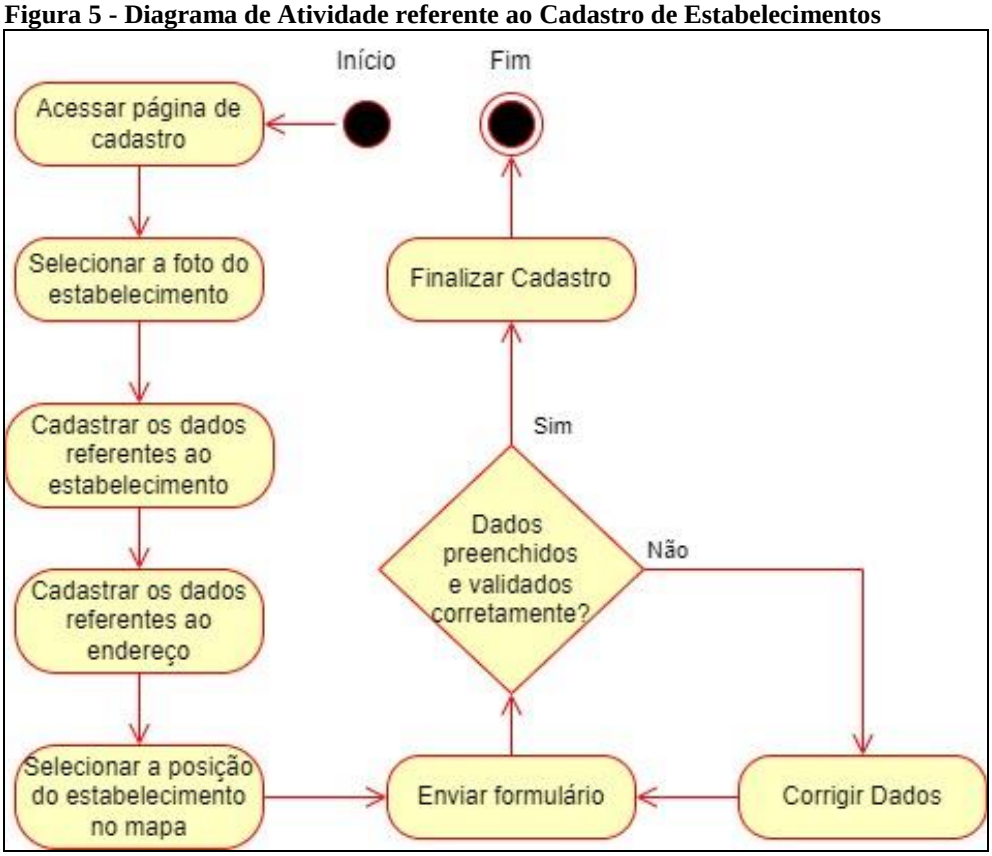


Fonte: Acervo do autor (2022)

4.1.4.2 Diagramas de Atividade

Na Figura 5 é apresentado o diagrama de atividade referente ao processo de cadastro de estabelecimentos. Para cadastrar o estabelecimento não é necessário login, apenas acessar a página de cadastro. Após estar na página, deve-se selecionar uma foto para o perfil, preencher os dados vinculados ao estabelecimento e seu endereço. Também é necessário selecionar sua localização no mapa. Ao enviar o formulário, é realizada uma validação dos dados preenchidos. Caso haja algum dado inválido, é necessária a correção. Após o envio dos dados devidamente preenchidos e validados, o cadastro é finalizado.

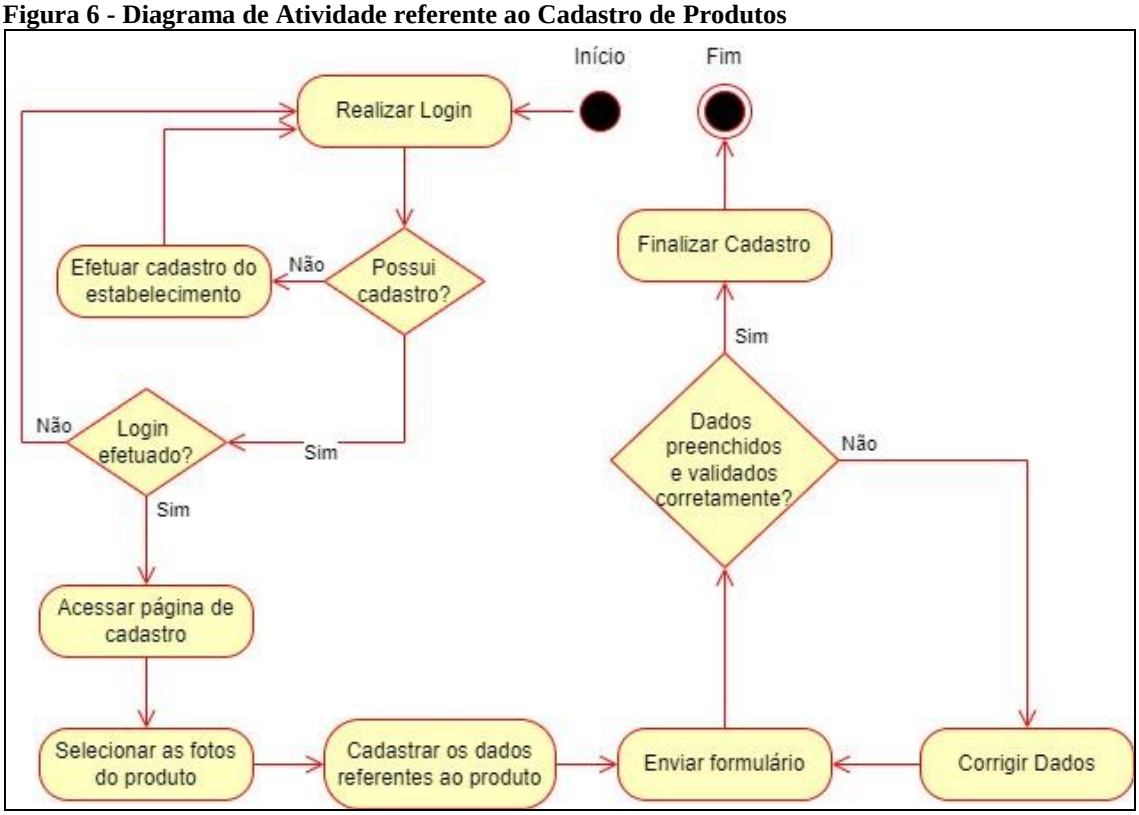
Figura 5 - Diagrama de Atividade referente ao Cadastro de Estabelecimentos



Fonte: Acervo do autor (2022)

A seguir, na Figura 6, é apresentado o diagrama de atividade referente ao processo de cadastro de produtos. Para cadastrar o produto é necessário efetuar login no sistema. Caso o usuário não possua um cadastro, deverá efetuá-lo. Após estar na página, deve-se selecionar uma ou mais fotos para o perfil do produto e então preencher os dados. Ao enviar o formulário, é realizada uma validação dos dados preenchidos. Caso haja algum dado inválido, a correção é necessária. Após o envio dos dados devidamente preenchidos e validados, o cadastro é finalizado.

Figura 6 - Diagrama de Atividade referente ao Cadastro de Produtos

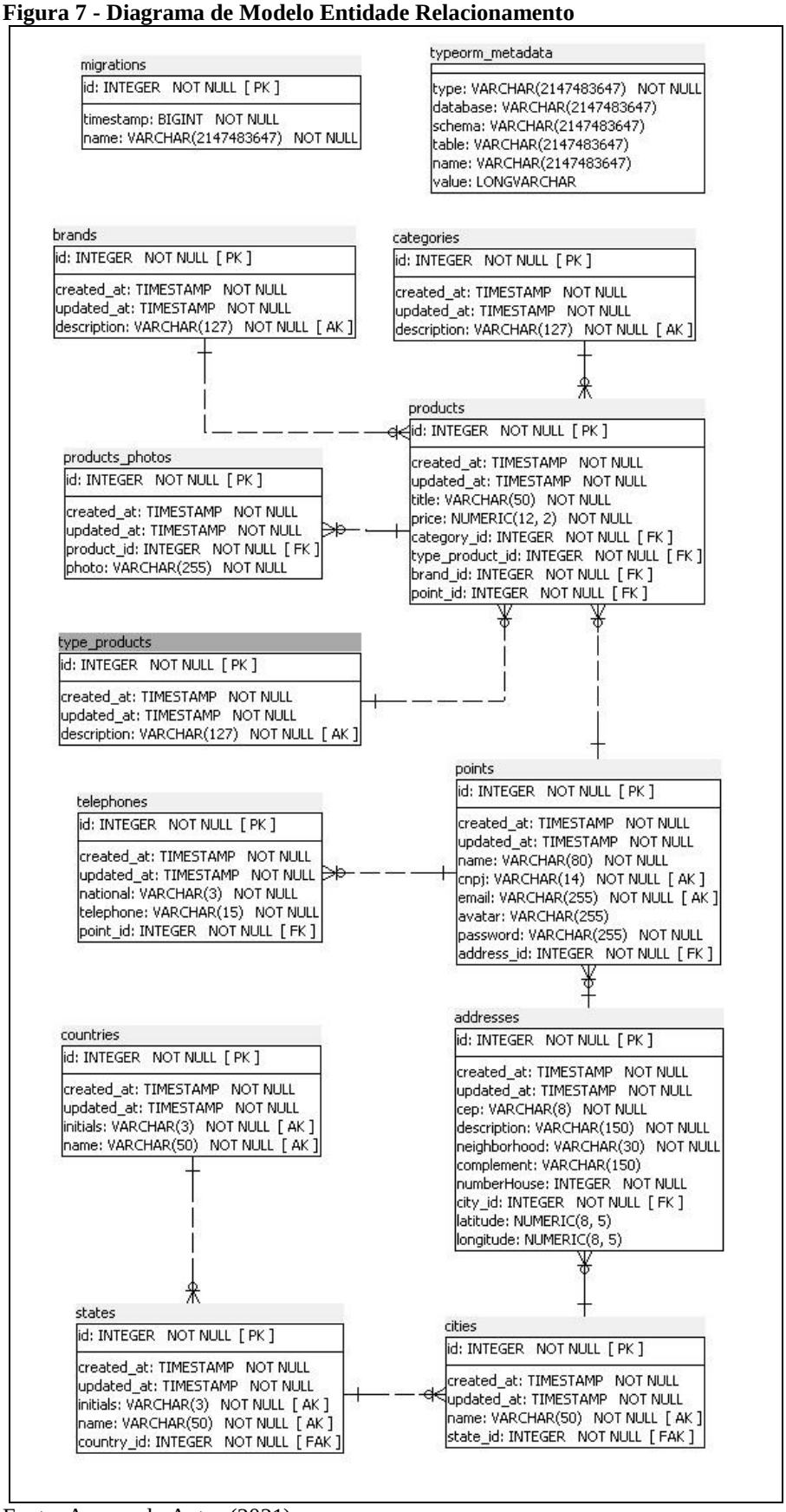


Fonte: Acervo do autor (2022)

4.1.4.3 Diagramas de Entidade Relacionamento

O diagrama de entidade relacionamento (Modelo Entidade Relacionamento), Figura 7, detalha as tabelas utilizadas no sistema, colunas e seus tipos de dados, e seus relacionamentos vigentes no banco de dados.

Figura 7 - Diagrama de Modelo Entidade Relacionamento



Fonte: Acervo do Autor (2021).

O projeto possui 14 tabelas. A tabela “migrations” tem como propósito facilitar o versionamento das *migrations* utilizadas. A tabela “typeorm_metadata” é utilizada para refletir o *metadata* utilizado na biblioteca TypeORM. As tabelas “countries”, “states”, “cities” e “adresses” foram elaboradas para constituir a parte de endereçamento do sistema. A tabela “points” se refere aos estabelecimentos. A tabela “telephones” aos telefones, possuindo uma chave estrangeira que a relaciona ao estabelecimento, pois este pode ter vários telefones. As tabelas “brands”, “categories” e “type_products” se referem as marcas, categorias e tipos do produto, respectivamente. Há também a tabela “products” para armazenar os registros de produtos, e a tabela “products_photos”, pois um produto pode ter várias fotos, assim realizando um relacionamento de 1 para N.

4.2 IMPLEMENTAÇÃO

Nesta seção são aprofundados os porquês da utilização das ferramentas e tecnologias utilizadas durante o desenvolvimento do protótipo, bem como também são apresentadas as rotinas implementadas.

4.2.1 Técnicas e Ferramentas Utilizadas

Para realizar a análise foram utilizadas as ferramentas *Draw.io* para a construção dos diagramas de caso de uso, e de atividades. Para o desenvolvimento do diagrama entidade relacionamento foi utilizado o programa *SQL Power Architect*.

O protótipo caracteriza-se por ser uma aplicação web responsiva, na qual são realizados todos os cadastros acessados e suas consultas. Para a programação foram utilizadas tecnologias de desenvolvimento web. A principal linguagem de desenvolvimento foi o JavaScript, justificando-se por ser uma linguagem muito versátil, podendo ser utilizada no back-end e no front-end, além da fácil integração com outras tecnologias como o HTML e CSS.

Para marcação de texto no front-end foi utilizado o HTML versão 5. Foi utilizado por ser suportado em praticamente todos os navegadores e também devido à sua enorme praticidade e facilidade de sua implementação em uma aplicação web.

Para a estilização das páginas foi utilizado o CSS versão 3. Essa tecnologia possibilita a criação de estilos personalizados para distintos elementos, permitindo a criação de uma tela com uma interface bonita, usável, intuitiva e acessível para o usuário.

O JavaScript foi utilizado como uma linguagem *cliente-side*. Sua escolha se deve ao fato de que o JavaScript permite a manipulação de elementos e criação de comportamentos, executados no navegador do cliente, já que com a linguagem é possível receber informações e carregá-las em tela em tempo de execução para o usuário. Devido a estes comportamentos, o JavaScript possibilita que o front-end do usuário seja muito mais dinâmico e interativo, proporcionando uma melhor experiência para o usuário.

Para agilizar e facilitar o desenvolvimento foi empregada a biblioteca React JS, que facilita e proporciona um desenvolvimento mais ágil e flexível, a base de componentes, permitindo uma ampla reutilização de código simples e intuitiva.

Para o desenvolvimento do back-end, JavaScript também foi a linguagem selecionada. Foi utilizado a framework Express para facilitar o desenvolvimento em Node.js, tornando assim o desenvolvimento mais sólido e robusto. O back-end foi projetado para ser uma aplicação RESTful, recebendo as requisições somente via API através de métodos HTTP.

Com a utilização do TypeORM foi realizada a manipulação do banco de dados. O uso dessa biblioteca possibilita um fácil e intuitivo uso na manipulação de registros e traz uma enorme facilidade ao criar as *migrations* baseados nos *models* elaborados.

O *PostgreSQL* foi o banco de dados selecionado. É um banco de dados relacional, amplamente difundido na comunidade de desenvolvedores. Somados a isso, leva-se em conta o fator de que o *PostgreSQL* é gratuito e possui todos os recursos necessários que possibilitam um melhor controle do banco de dados.

A ferramenta de desenvolvimento utilizada foi o *Visual Studio Code*, um editor de texto que possui o apoio de uma comunidade muito extensa e que possui uma variedade imensa de extensões, sugerindo opções através de *snippets*. Possui temas personalizáveis e um terminal integrado.

O método para mensurar a qualidade do software consistiu em realizar diversos testes nas rotinas desenvolvidas, simulando o uso diário de um usuário. Dessa forma foi possível encontrar alguns erros e inconsistências nas rotinas.

Para garantir a segurança na autenticação do usuário, foi realizada o uso de criptografia nas senhas dos usuários. A senha jamais é descriptografada, portanto, o único método para checar se a senha existente é a mesma que a senha informada consiste em criptografar a senha enviada e comparar com a senha criptografada já existente. A mesma senha, ou também a sua criptografia, jamais é retornada pelo back-end. A autenticação ocorre através do uso de JSON Web Token.

4.2.2 Utilização e Funcionamento

O protótipo se refere a uma aplicação responsiva para dois tipos de usuários. O primeiro possui a possibilidade de realizar o cadastro de estabelecimentos e de produtos para intolerantes. O segundo usuário, que não necessitará de cadastro no sistema, poderá visualizar os estabelecimentos cadastrados e seus produtos próximo a sua localidade.

Neste tópico são apresentados as telas e os seus funcionamentos. Todas as telas são apresentadas, sendo visualizas em um Notebook com tamanho de 1536x713 pixels, em um iPad Air (Tablet) com tamanho de 820x1180 pixels e em um iPhone 12 Pro (Celular) com tamanho de 390x844 pixels.

4.2.2.1 Tela Inicial

A tela inicial (Figura 8 e 9), consiste em apresentar a aplicação ao usuário, contendo a logo do sistema, e dois botões, onde o primeiro redireciona o usuário para a tela de login, na qual o usuário pode cadastrar um estabelecimento. O segundo botão redireciona o usuário a tela de consulta dos estabelecimentos próximos a sua localização.

Figura 8 – Tela Inicial - Computador

Fonte: Acervo do autor (2022)

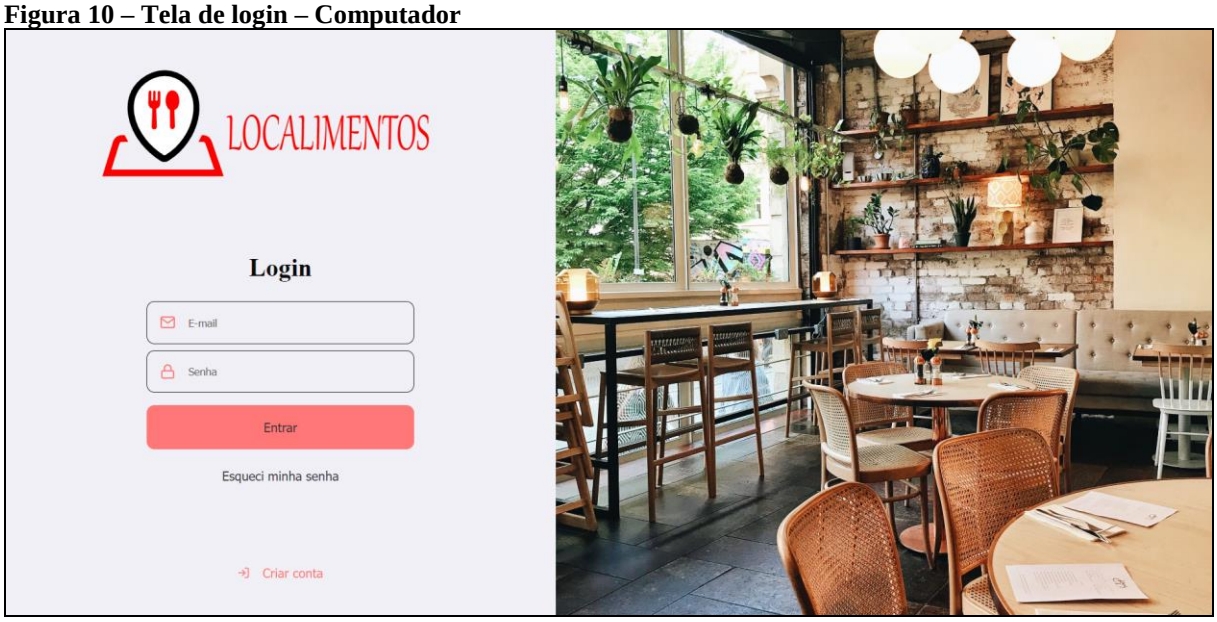
Figura 9 – Tela Inicial - Celular e Tablet respectivamente

Fonte: Acervo do autor (2022)

4.2.2.2 Tela de Login

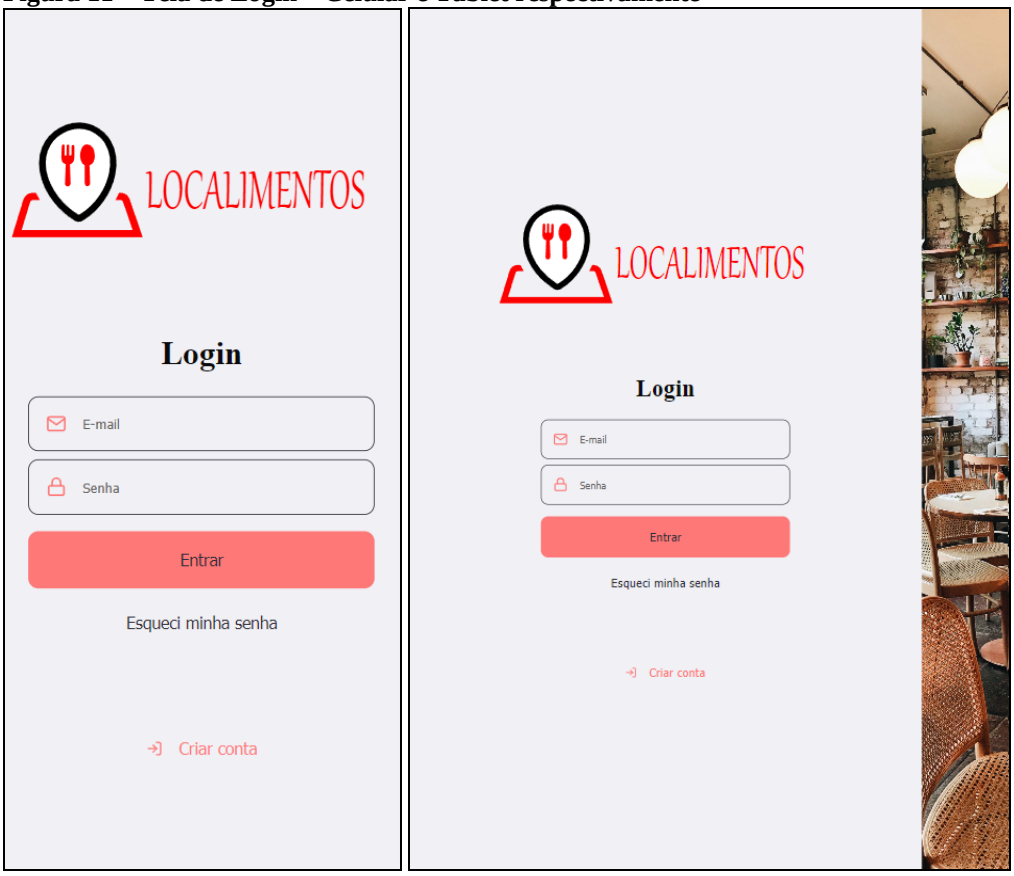
A Figura 10 representa a tela de login, a qual possibilita que o usuário efetue o login, através de um email e de uma senha, para poder acessar todas as funcionalidades do sistema voltadas aos estabelecimentos. Também possibilita o redirecionamento do usuário a tela de cadastro de estabelecimentos ao clicar no botão inferior “Criar Conta”. A Figura 11 possui a mesma funcionalidade, operando em tablets e celulares.

Figura 10 – Tela de login – Computador



Fonte: Acervo do autor (2022)

Figura 11 – Tela de Login – Celular e Tablet respectivamente



Fonte: Acervo do Autor (2022)

4.2.2.3 Cadastro de Estabelecimentos

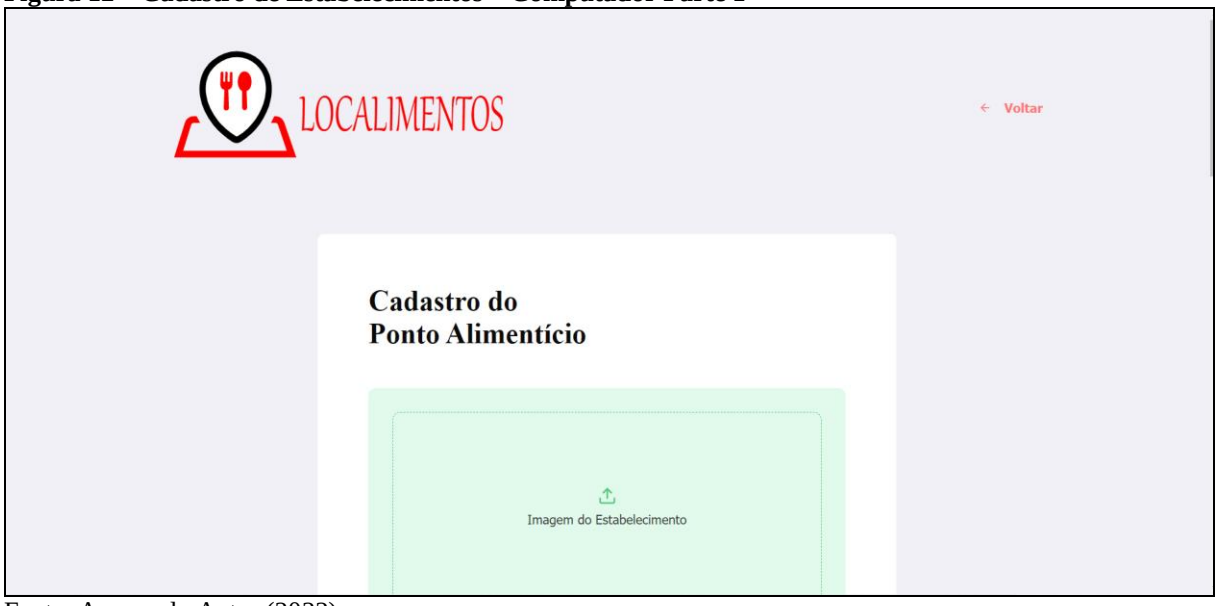
A tela de cadastro de estabelecimentos é uma das telas mais complexas do sistema. A mesma possui três seções principais, sendo apresentadas nas Figuras 12 a 20 e descritas na sequência.

A primeira seção possibilita ao usuário informar uma foto de perfil para o seu estabelecimento, utilizando um componente de Dropzone, podendo clicar e selecionar a imagem, ou arrastá-la do seu computador ao componente.

A segunda seção corresponde aos dados do estabelecimento. Permite ao usuário informar dados como CNPJ, email, nome, senha e telefone.

A terceira seção corresponde ao endereço do usuário havendo duas subdivisões principais. Possibilita ao usuário selecionar a sua localização no mapa através do componente de Map. Permite ainda ao usuário informar os dados do endereço, como: CEP, descrição, bairro, número, complemento. Uma importante funcionalidade do sistema consiste em que ao informar o CEP (primeiro campo dessa seção), a aplicação realiza uma pesquisa via API dos Correios para buscar os dados de tal CEP e preenchê-lo na aplicação, a fim de agilizar o processo para o usuário.

Figura 12 – Cadastro de Estabelecimentos – Computador Parte 1



Fonte: Acervo do Autor (2022)

Figura 13 – Cadastro de Estabelecimentos – Computador Parte 2

Fonte: Acervo do autor (2022)

Figura 14 – Cadastro de Estabelecimentos – Computador Parte 3

Fonte: Acervo do autor (2022)

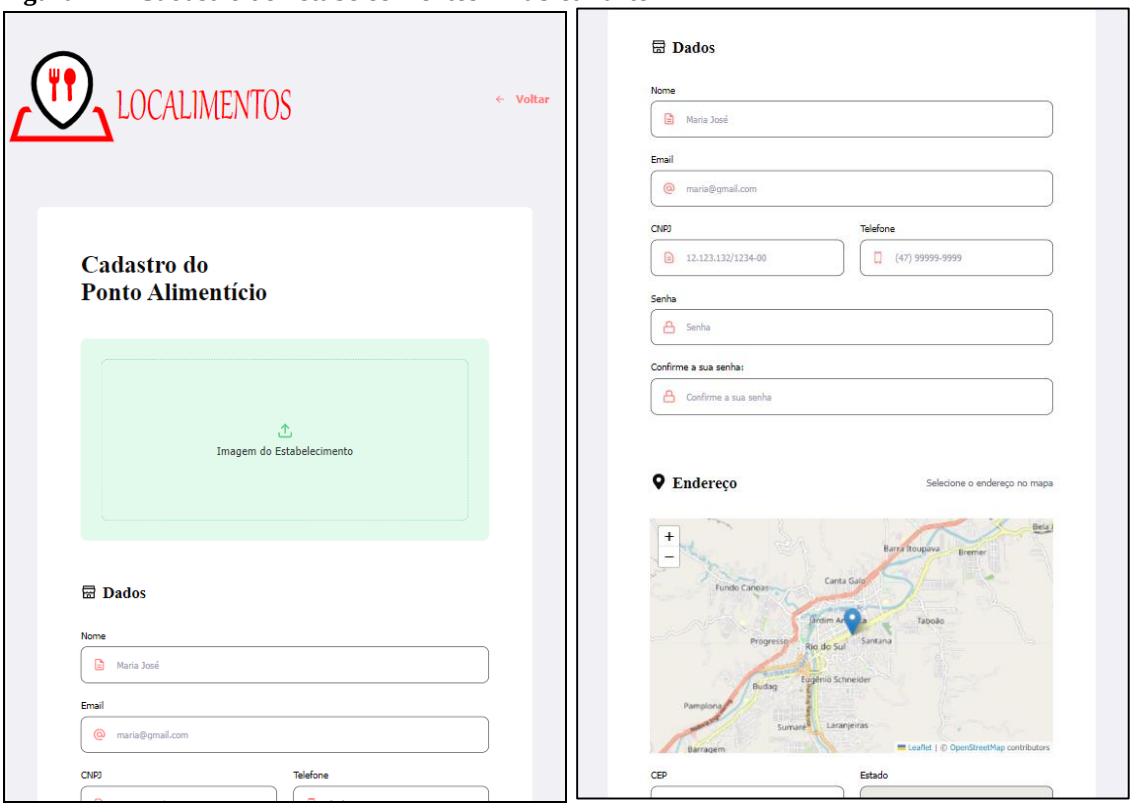
Figura 15 – Cadastro de Estabelecimentos – Computador Parte 4

Fonte: Acervo do autor (2022)

Figura 16 – Cadastro de Estabelecimentos – Computador Parte 5

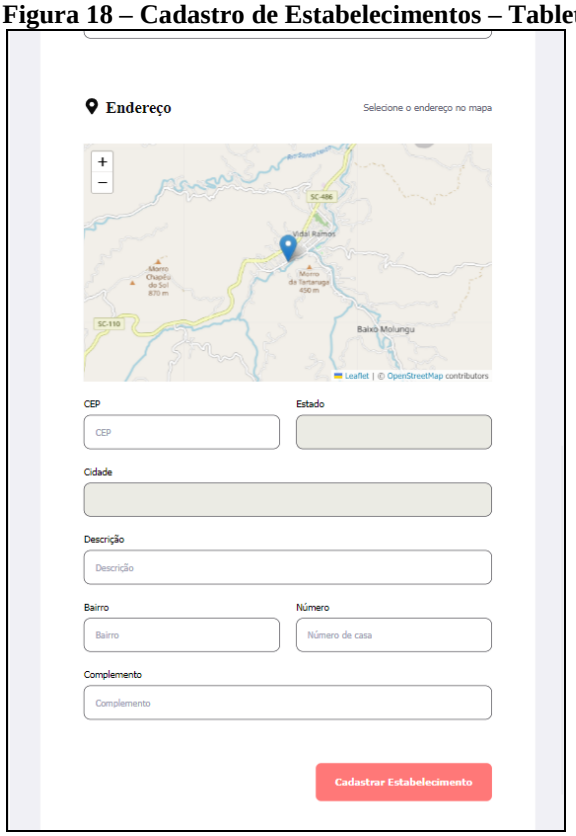
Fonte: Acervo do autor (2022)

Figura 17 – Cadastro de Estabelecimentos – Tablet Parte 1



Fonte: Acervo do autor (2022)

Figura 18 – Cadastro de Estabelecimentos – Tablet Parte 2

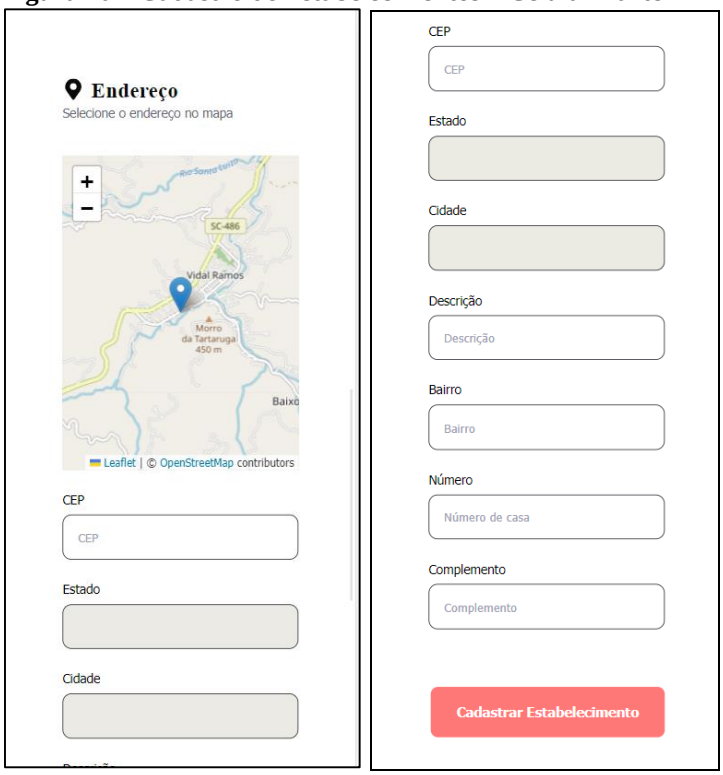


Fonte: Acervo do Autor (2022)

Figura 19 – Cadastro de Estabelecimentos – Celular Parte 1

Fonte: Acervo do autor (2022)

Figura 20 – Cadastro de Estabelecimentos – Celular Parte 2

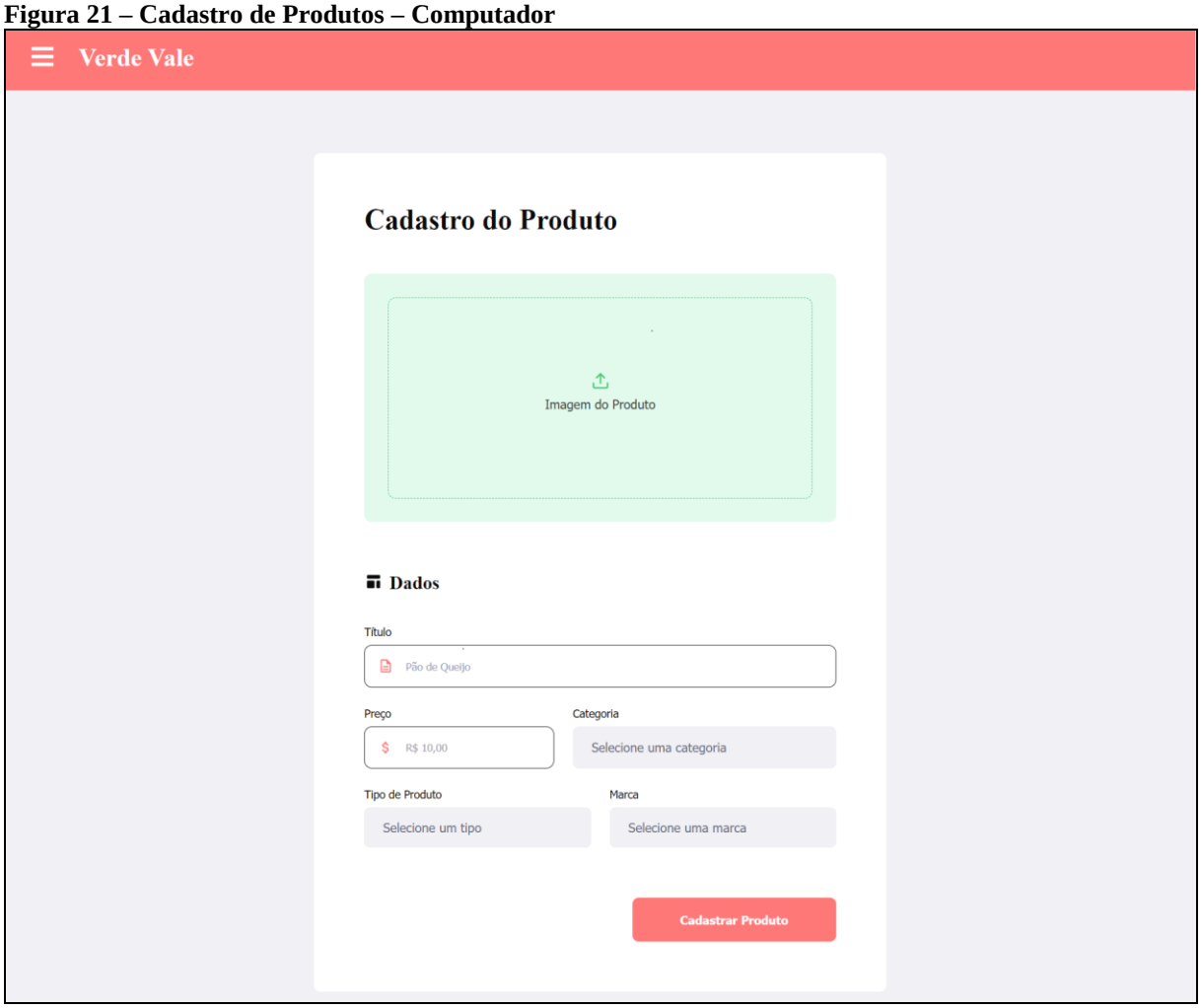


Fonte: Acervo do autor (2022)

4.2.2.4 Cadastro de Produtos

A tela de cadastro de produtos que pode ser visualizada nas figuras 21 a 23 permite ao usuário informar até 6 (seis) fotos para o produto, utilizando um componente de Dropzone, podendo clicar e selecionar a imagem, ou arrastá-la do seu computador ao componente. Também permite ao estabelecimento informar os dados do produto, como seu nome, preço, categoria (se é produto sem glúten ou sem lactose, por exemplo), marca (se é comprado ou se é caseiro, por exemplo) e tipo de produto (se consiste em uma massa, bolacha, *fast food*).

Figura 21 – Cadastro de Produtos – Computador

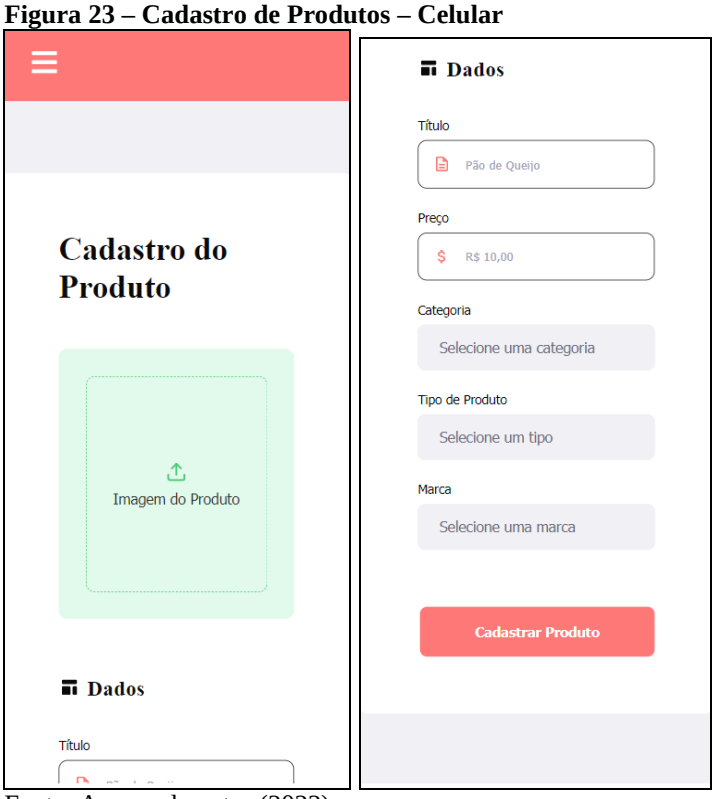


Fonte: Acervo do autor (2022)

Figura 22 – Cadastro de Produtos – Tablet

Fonte: Acervo do autor (2022)

Figura 23 – Cadastro de Produtos – Celular

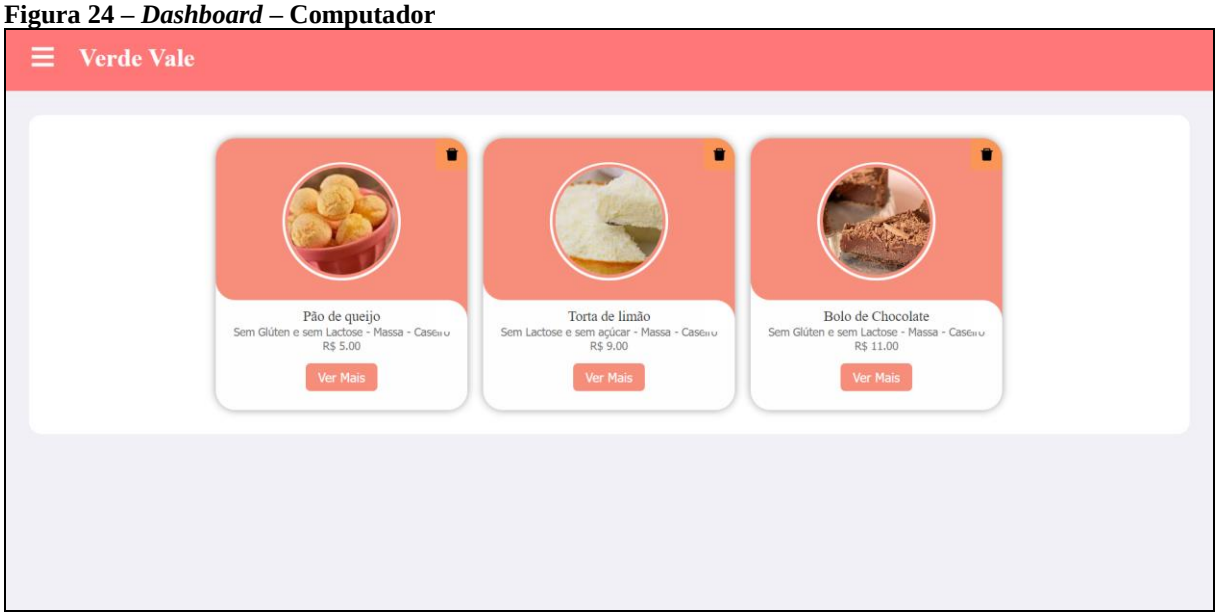


Fonte: Acervo do autor (2022)

4.2.2.5 *Dashboard*

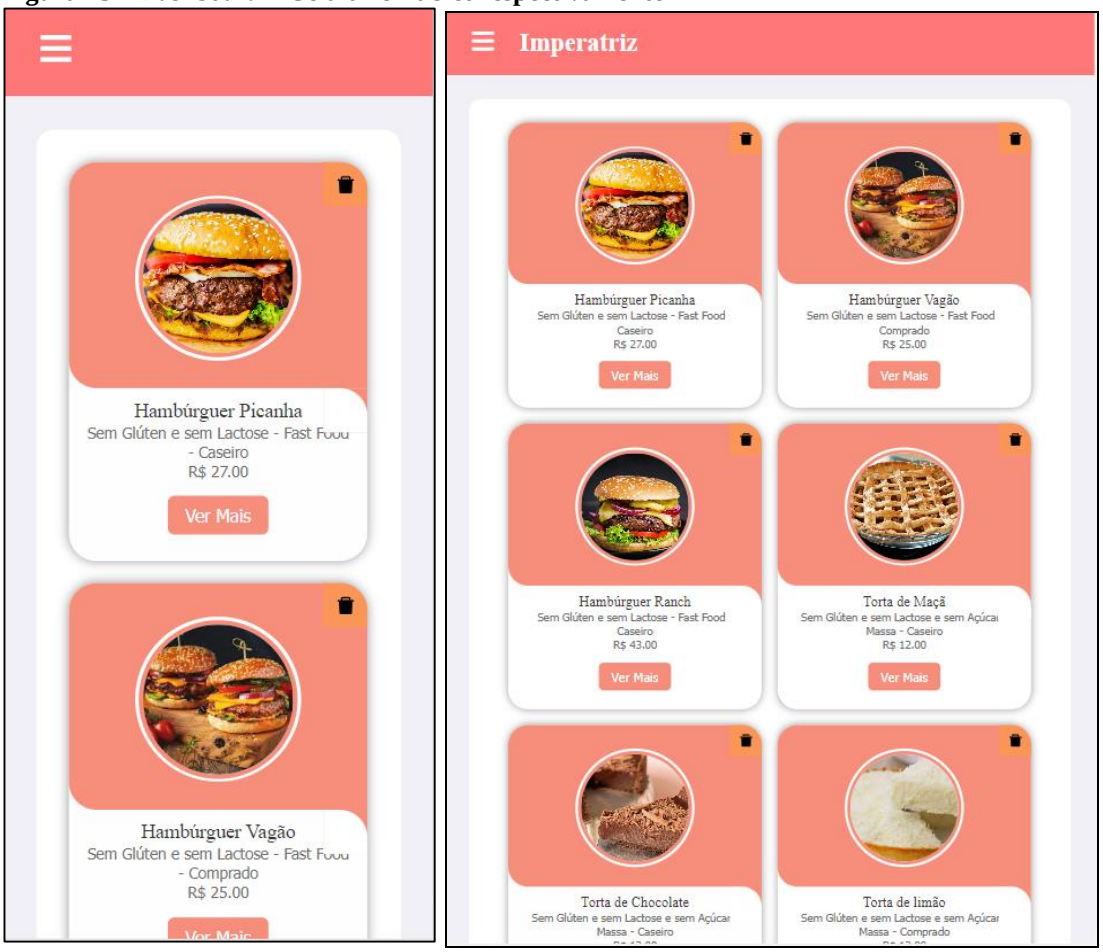
O *dashboard* (Figuras 24 e 25) consiste em mostrar ao usuário os produtos vinculados ao seu sistema em forma de *Cards*, apresentando todos os seus detalhes. O *card* possui um botão de clique duplo que permite ao usuário deletar o produto e também um botão de "Ver Mais", para possibilitar a edição do produto para o usuário.

Figura 24 – *Dashboard* – Computador



Fonte: Acervo do autor (2022)

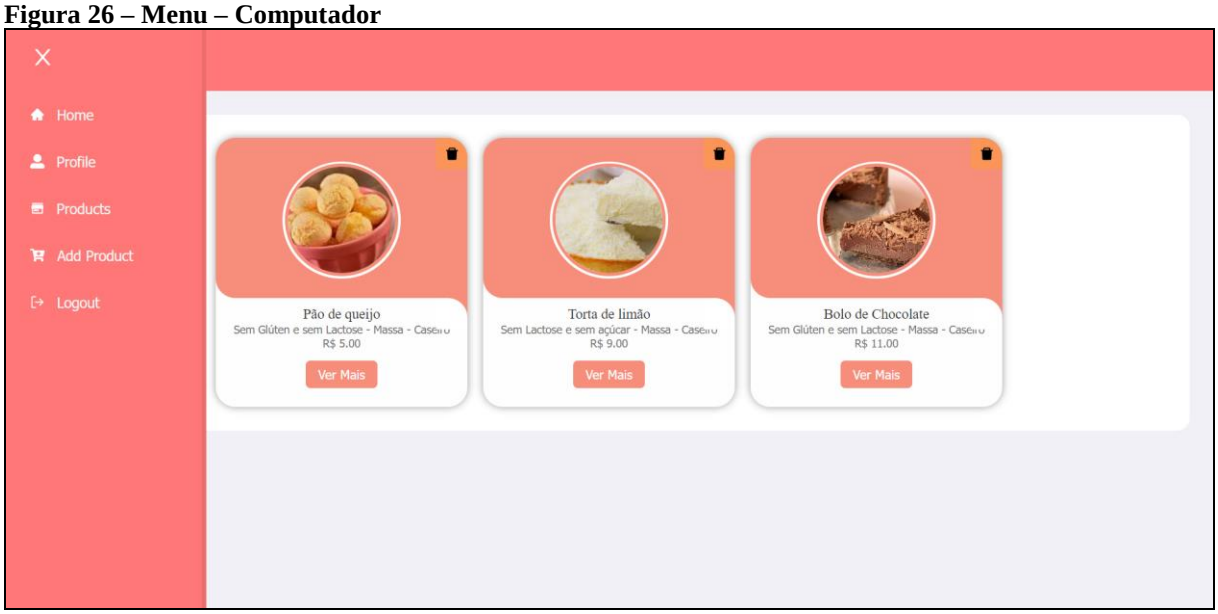
Figura 25 – *Dashboard* – Celular e Tablet respectivamente



Fonte: Acervo do autor (2022)

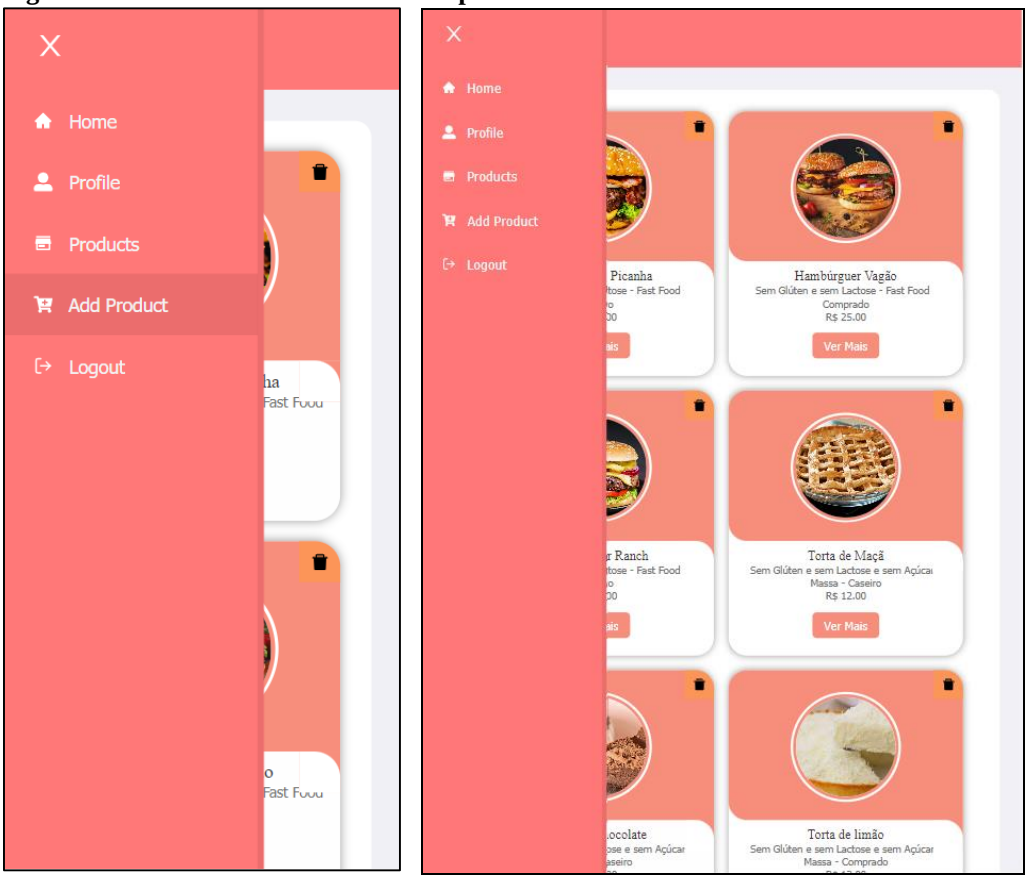
O *dashboard* possui um menu (Figuras 26 e 27), o qual ao ser clicado de forma intuitiva, abre o menu lateral que apresenta as possibilidades de redirecionamento ao *dashboard*, redirecionamento a tela de cadastro de produtos e também possui um botão de *logout*, para permitir ao usuário sair do sistema.

Figura 26 – Menu – Computador



Fonte: Acervo do autor (2022)

Figura 27 – Menu – Celular e Tablet respectivamente



Fonte: Acervo do autor (2022)

4.2.2.6 Mapa

A tela do mapa é a principal da aplicação e é representada da Figura 28 à 32. Essa tela consiste em abrir um mapa centralizado na localização do usuário, através do uso da geolocalização permitida pelo usuário. São carregados todos os estabelecimentos em um raio de distância de até 10 quilômetros do usuário em cards, disponibilizando foto e nome do estabelecimento, na localização cadastrada pelo próprio estabelecimento.

Ao clicar em um estabelecimento, abre-se um modal que possui 2 seções. A primeira disponibiliza um card que mostra todos os detalhes do estabelecimento, tais como sua foto, seu nome, seu endereço, e-mail e telefones. A segunda seção consiste em apresentar todos os produtos cadastrados no estabelecimento, separados por categorias. Os dados apresentados dos produtos consistem em uma foto, seu nome, preço, tipo de produto e marca. É possível filtrar os produtos pela categoria designada através de um campo select, localizado no topo desta mesma seção. Ao selecionar a categoria, a tela é “rolada” até a categoria informada.

Figura 28 – Mapa - Computador

Fonte: Acervo do autor (2022)

Figura 29 – Mapa – Celular e Tablet respectivamente

Fonte: Acervo do autor (2022)

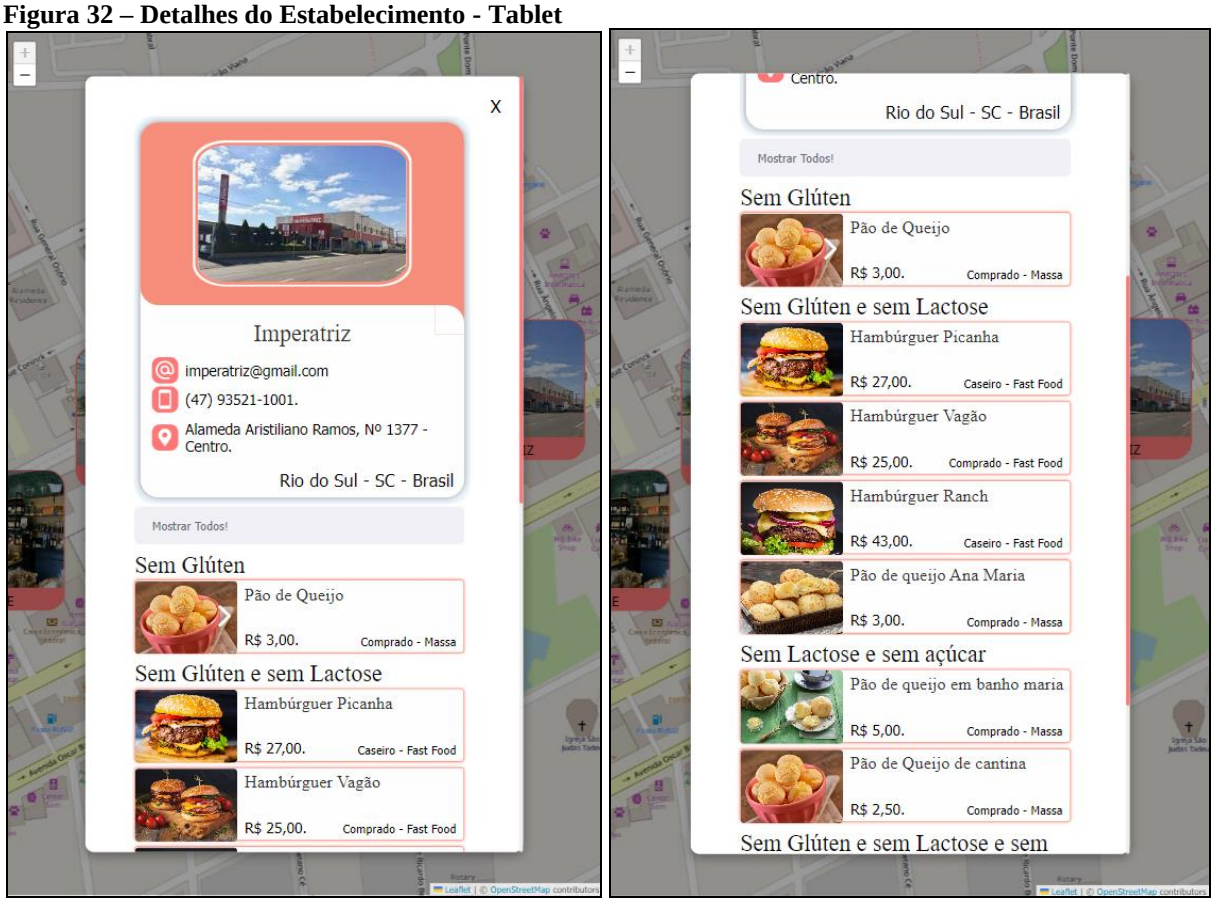
Figura 30 – Detalhes do Estabelecimento - Computador

Fonte: Acervo do autor (2022)

Figura 31 – Detalhes do Estabelecimento - Celular

Fonte: Acervo do autor (2022)

Figura 32 – Detalhes do Estabelecimento - Tablet



Fonte: Acervo do autor (2022)

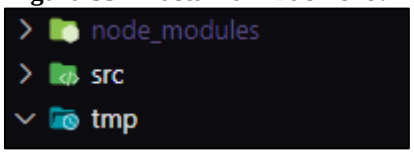
4.2.3 Estrutura de Pastas

Nesse tópico são apresentadas as estruturas de pastas utilizadas no desenvolvimento do projeto e a finalidade desta organização.

4.2.3.1 Back-End

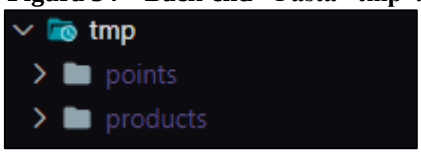
Na pasta raiz são apresentadas 3 pastas (Figura 33). A pasta "node_modules" contém todas as dependências instaladas no projeto. A pasta "tmp", Figura 34, contém os arquivos temporários, sendo que neste projeto contém todas as imagens enviadas pelos usuários. Para facilitar a organização, foi categorizado em uma pasta de "points" para imagens direcionadas aos estabelecimentos e uma pasta "products" para imagens direcionadas aos produtos.

Figura 33 – Pasta Raiz Back-end.



Fonte: Acervo do autor (2022)

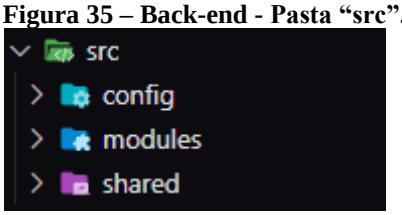
Figura 34 – Back-end - Pasta “tmp”.



Fonte: Acervo do autor (2022)

A pasta “src”, Figura 35, contém o código fonte. Dentro dessa pasta há outras 3 pastas. A primeira é a pasta "config" que possui arquivos de configurações. Para as outras duas pastas foi utilizado o princípio de módulos distintos e compartilhados. A pasta "modules" contém diversos módulos, onde um módulo basicamente não se comunicará com o outro. Os módulos são detalhados posteriormente. A pasta "shared", por outro lado, possui todos os arquivos que podem ser compartilhados entre diversos módulos.

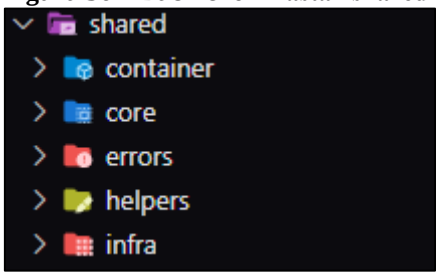
Figura 35 – Back-end - Pasta “src”.



Fonte: Acervo do autor (2022)

Dentro da pasta "shared", Figura 36, há a pasta "container" que possui a funcionalidade de registrar os repositórios no container fornecido pelo Tsyringe (biblioteca que permite injeção de dependência nos métodos construtores em TypeScript). Possui a pasta "core", que possui o padrão de algumas classes como “ControllerPadrao”, “ServicePadrao” e afins. Há a pasta "errors" que possui todos os arquivos vinculados a erros. Há também a pasta "helpers" que possui todos os arquivos helpers (classes que provém assistência ou funcionalidades específicas). Por último, tem a pasta "infra" que possui as partes relacionadas a infraestrutura.

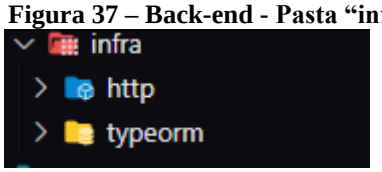
Figura 36 – Back-end - Pasta “shared”.



Fonte: Acervo do autor (2022)

Dentro da pasta “infra”, representada na Figura 37, tem a pasta "http" que possui o arquivo que inicializa a aplicação, suas rotas e a pasta de "middlewares" que possui todos os middlewares que podem ser relacionados a quaisquer módulos do sistema. E por último há a pasta "typeorm" que possui todos os arquivos que devem ser compartilhados relacionados a infraestrutura da biblioteca TypeORM para relacionamento com banco de dados.

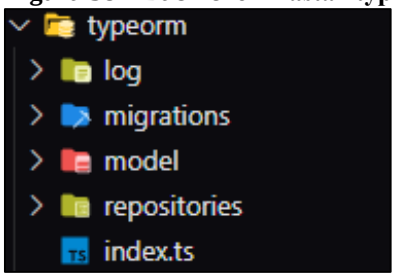
Figura 37 – Back-end - Pasta “infra”.



Fonte: Acervo do autor (2022)

Dentro da pasta “typeorm” existe a pasta "log" para guardar os logs do sistema (Figura 38). Há também a pasta "migrations" que possui todas as migrations para versionamento do banco de dados. Há também variados models padrão dentro da pasta de "model", dependendo do tipo de entidade que se deseja ter, e a pasta de "repositories" que possui um arquivo de configuração específico ao TypeORM (por isso se encontra localizado nessa pasta e não na pasta "config"), e um repositório padrão. O último item desta pasta é o arquivo index.ts que inicializa a conexão com o banco de dados.

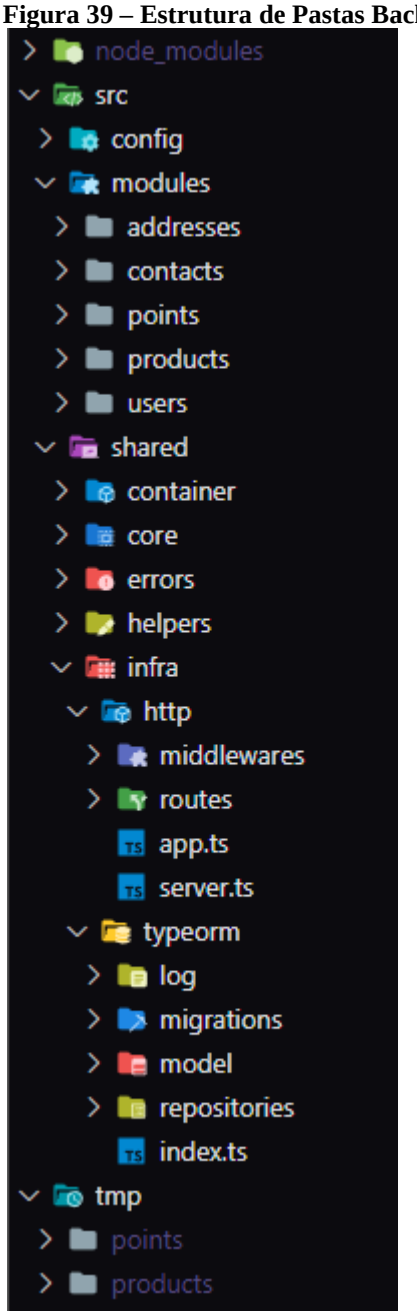
Figura 38 – Back-end - Pasta “typeorm”.



Fonte: Acervo do autor (2022)

A Figura 39 demonstra uma representação geral das pastas do back-end.

Figura 39 – Estrutura de Pastas Back-end

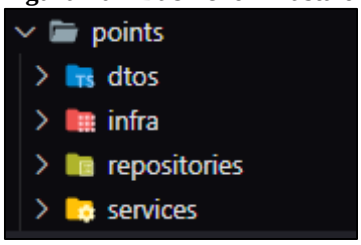


Fonte: Acervo do autor (2022)

A pasta “modules” possui vários módulos. Todos os módulos possuem as mesmas conjunturas de pastas. As pastas de módulos se constituem de 4 subpastas (Figura 40). A primeira é a pasta "dtos" que possui todos os DTOs (Data Transfer Object, ou Objeto de Transferência de Dados) relacionadas ao módulo. A segunda pasta é a pasta de “repositories”, que possuirá um ou mais arquivos de interface para repositórios. Esse recurso foi utilizado para possibilitar uma implementação do Princípio de Substituição de Liskov, afinal, caso seja necessário trocar de ORM no futuro da aplicação, os services não terão que tratar com os

repositórios fornecidos pelos ORMs, e sim por um repositório próprio da aplicação, onde todos os repositórios possuem uma mesma interface, logo mesmos métodos com os mesmos parâmetros, facilitando imensamente a troca. A terceira pasta é a pasta de "services", onde se encontram todos os services da aplicação. A última pasta é a pasta "infra", que é bem semelhante a pasta "infra" da pasta "shared", porém aqui todos os arquivos são unicamente relacionados ao módulo.

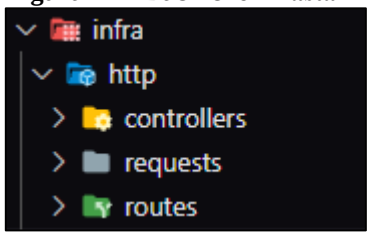
Figura 40 – Back-end – Pasta de módulos.



Fonte: Acervo do autor (2022)

A pasta "infra" representada na Figura 41 possui uma pasta "http". Esta pasta possui a pasta "controllers" que possui os controllers deste módulo. Dentro de "http" há a pasta "routes", que possui as rotas criadas para atender à este módulo e à pasta "requests". Essa última possui arquivos de validação, que são utilizados para validar os dados das requests, repassados dos controllers para os services. O nome "requests" e não "validation" foi inspirado na pasta de "requests" que o Laravel (Framework PHP) fornece.

Figura 41 – Back-end - Pasta “http”.

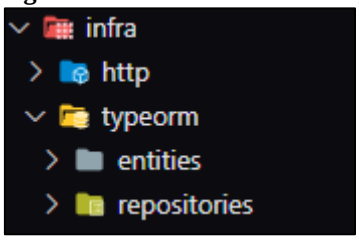


Fonte: Acervo do autor (2022)

Dentro de "infra" (Figura 42) também há a pasta "typeorm". Dentro desta pasta há a pasta "entities" que possui todas as entidades do módulo. Esta pasta se encontra dentro de "typeorm", pois as entidades utilizam de diversos decorators fornecidos pela biblioteca TypeORM. Também dentro da pasta "typeorm" há os repositórios. Esses repositórios implementam as interfaces criadas na pasta "repositories", localizadas no diretório do módulo,

onde são implementados os métodos existentes na interface, realizando o envio e a recuperação de dados com o banco de dados.

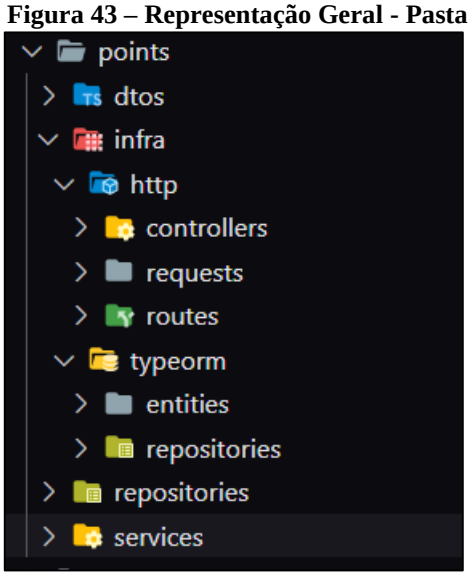
Figura 42 – Back-end - Pasta “typeorm”.



Fonte: Acervo do autor (2022)

A Figura 43 demonstra a representação geral das pastas de módulos.

Figura 43 – Representação Geral - Pasta de Módulos

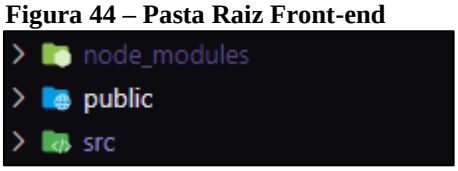


Fonte: Acervo do autor (2022)

4.2.3.1 Front-End

Na pasta raiz são apresentadas 3 pastas, conforme ilustrado na Figura 44. A pasta "node_modules" contém todas as dependências instaladas no projeto. A pasta "public" contém o arquivo index.html, assim como outros arquivos importantes, tais como o favicon.ico e o manifest.json. A pasta "src" contém o código fonte.

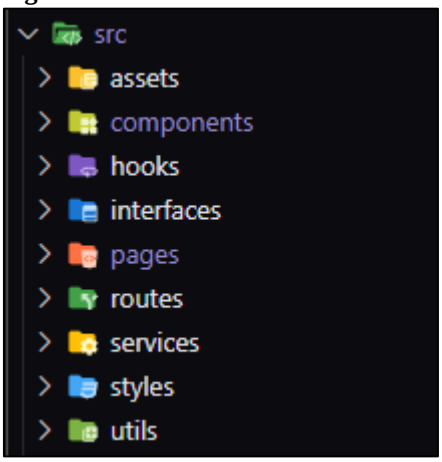
Figura 44 – Pasta Raiz Front-end



Fonte: Acervo do autor (2022)

Dentro de "src", Figura 45, é possível encontrar a pasta "assets", que possui todos os arquivos de mídia que são utilizados diretamente do front-end. A pasta "components" possui todos os componentes criados na aplicação. A pasta "hooks" possui todos os hooks criados, incluindo providers e arquivos de contexto (que salvam e recuperam informações no localStorage do navegador). Há a pasta "interfaces" totalmente destinada às interfaces, para evitar com que uma mesma interface seja escrita em diversos componentes ou páginas diferentes, possibilitando assim um maior reuso do código. A pasta "pages" possui todos os arquivos relacionados às páginas da aplicação. Existe também a pasta "routes", onde são criadas todas as rotas no front-end, as quais cada rota abre uma página diferente. Há a pasta "services", que possui alguns arquivos de APIs diferentes que foram utilizados na aplicação, tais como a API dos Correios, e a API da própria aplicação. Há a pasta "styles" que disponibiliza alguns estilos globais e também certos estilos que foram reutilizados em diversos componentes. Por último a pasta "utils", que disponibiliza vários métodos facilitadores para o desenvolvimento, semelhante a pasta de "Helpers" no back-end.

Figura 45 – Pasta src Front-End



Fonte: Acervo do autor (2022)

4.2.4 Componentes

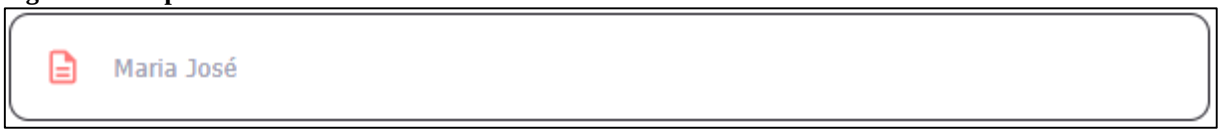
Neste tópico são apresentados alguns componentes existentes no sistema, abrangendo suas funcionalidades e todas as suas possíveis formas.

4.2.4.1 Input

O componente Input é utilizado em todos os formulários do sistema. Possui diversas formas, havendo estilo para estado inicial, para estado focado, para estado preenchido e um

estado de erro. Ao ser inicializado, o componente é apresentado, conforme ilustrado na Figura 46.

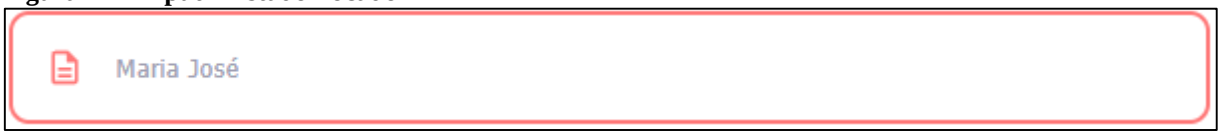
Figura 46 – Input – Estado Inicial



Fonte: Acervo do autor (2022)

Ao clicar no Input, com a intuição de informar algum dado, o componente recebe uma borda rosa, com o intuito de separar o Input selecionado dos demais, exemplificado na Figura 47.

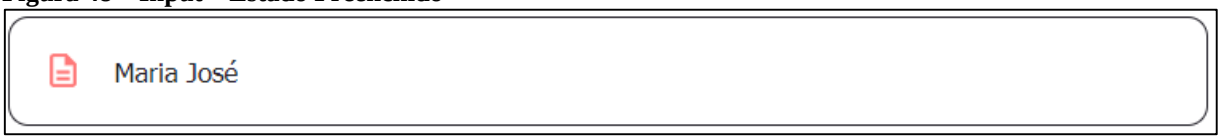
Figura 47 – Input – Estado Focado



Fonte: Acervo do autor (2022)

Após o dado ser informado, o Input perde a sua borda, seja uma borda de indicação de foco, ou de indicação de erro, como mostra a Figura 48.

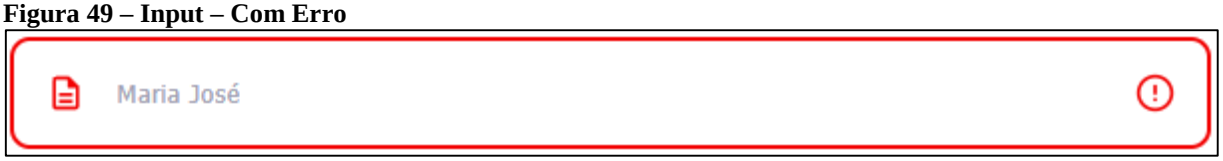
Figura 48 – Input – Estado Preenchido



Fonte: Acervo do autor (2022)

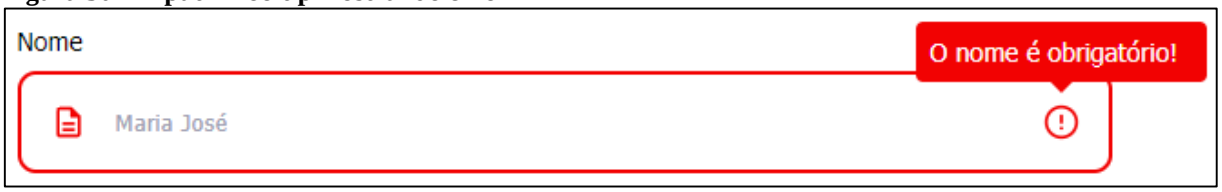
Ao receber um erro, como comumente acontece ao enviar um formulário, o componente recebe uma borda vermelha, o ícone adquire a mesma cor (Figura 49). Além disso, é adicionado um ícone indicando um erro na parte direita do componente. Ao se passar o mouse por cima, é mostrado um Tooltip onde é apresentada a devida mensagem de erro relacionada ao Input, mostrado na Figura 50.

Figura 49 – Input – Com Erro



Fonte: Acervo do autor (2022)

Figura 50 – Input – Tooltip mostrando erro

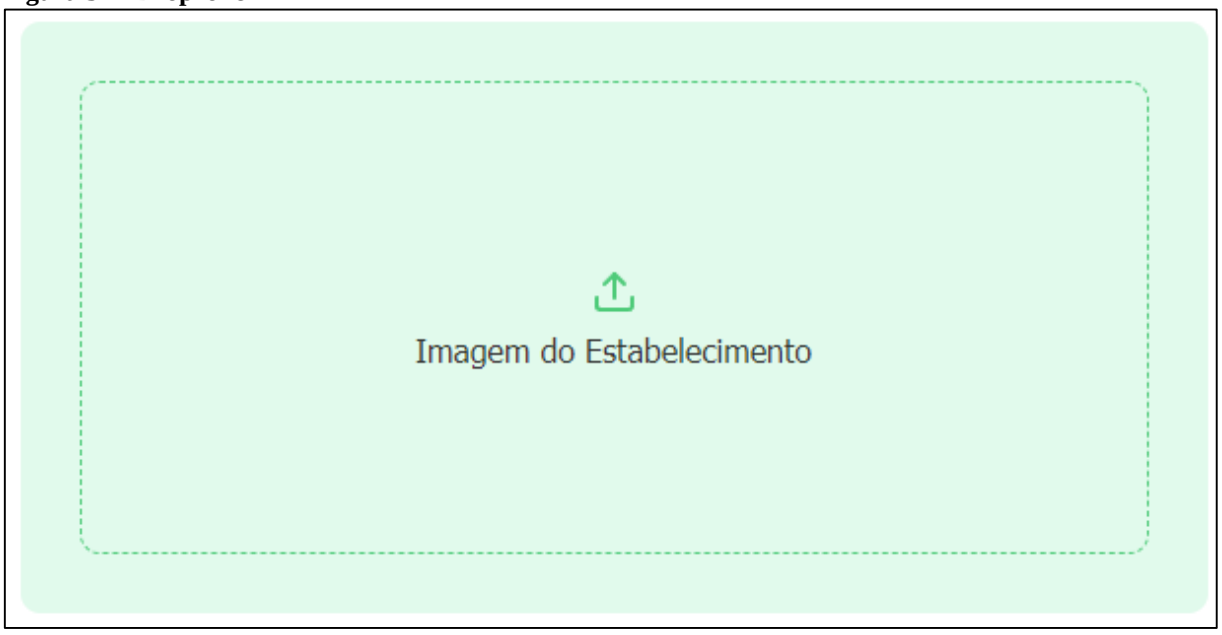


Fonte: Acervo do autor (2022)

4.2.4.2 Dropzone

O componente Dropzone é utilizado nos formulários de cadastros de estabelecimentos e nos cadastros de produtos. É possível visualizar as fotos selecionadas no componente, ver as previamente carregadas e as duas simultaneamente. O componente também apresenta uma forma de mostrar o erro. Ao ser inicializado o componente é apresentado, conforme demonstrado na Figura 51.

Figura 51 - Dropzone



Fonte: Acervo do autor (2022)

Ao selecionar uma ou mais imagens, arrastando do computador ou clicando no componente, a imagem selecionada aparece abaixo do Dropzone na seção “Imagens Selecionadas”, exemplificado na Figura 52.

Figura 52 – Dropzone – Imagem Selecionada

Fonte: Acervo do autor (2022)

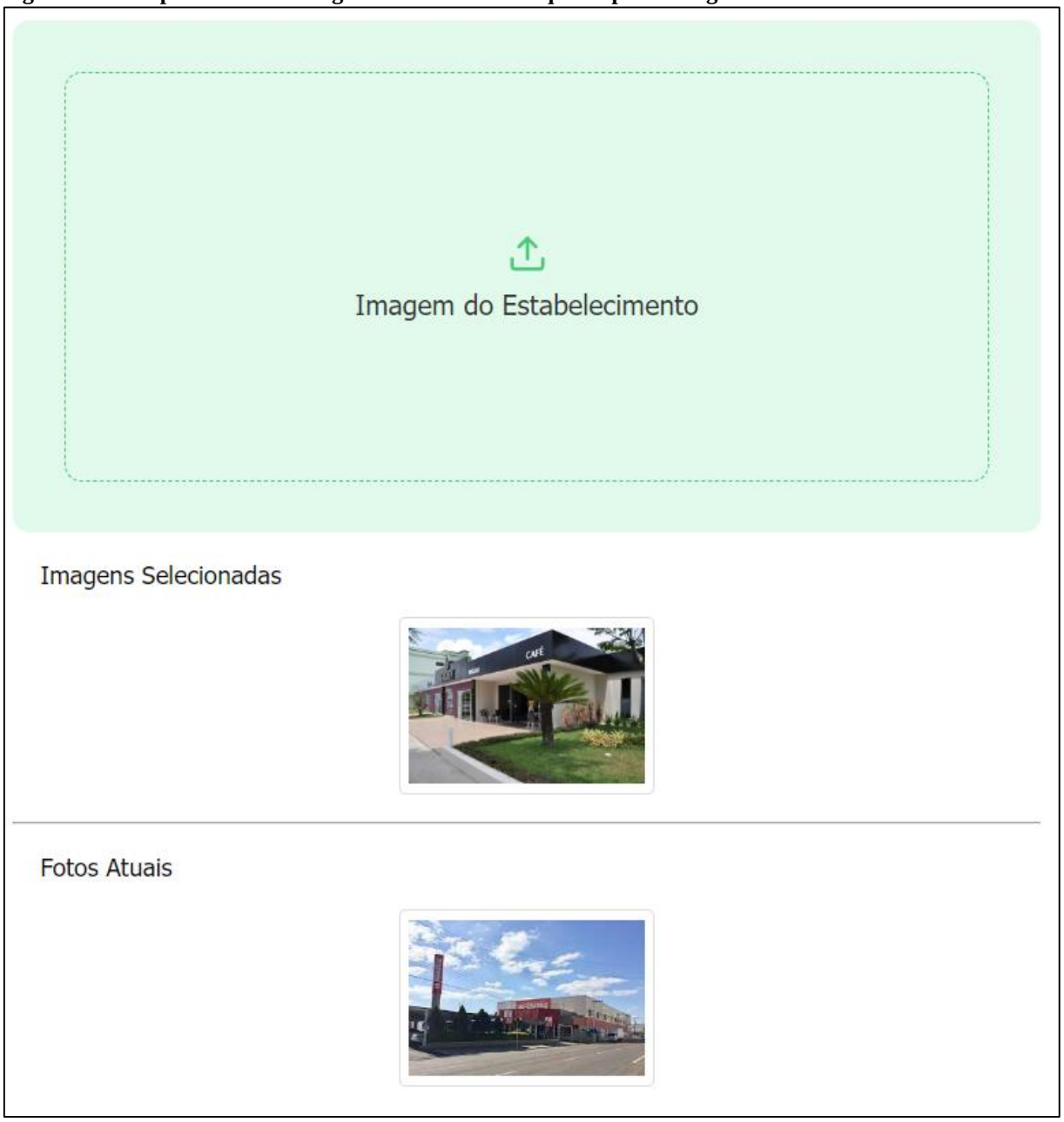
Caso seja necessário carregar previamente uma ou mais imagens vinculadas ao Dropzone, elas são mostradas abaixo do Dropzone na seção “Fotos Atuais”, como é observado na Figura 53.

Figura 53 – Dropzone – Arquivos pré-carregados.

Fonte: Acervo do autor (2022)

Caso haja uma ou mais imagens previamente carregadas e vinculadas ao Dropzone e o usuário selecione uma foto, a seção “Imagens Selecionadas” é exibida logo abaixo do Dropzone e a seção “Fotos Atuais” será exibida mais abaixo, com uma leve divisória para indicar a separação entre as seções, como demonstrado na Figura 54.

Figura 54 – Dropzone – Com imagem selecionada e Arquivos pré-carregados.



Fonte: Acervo do autor (2022)

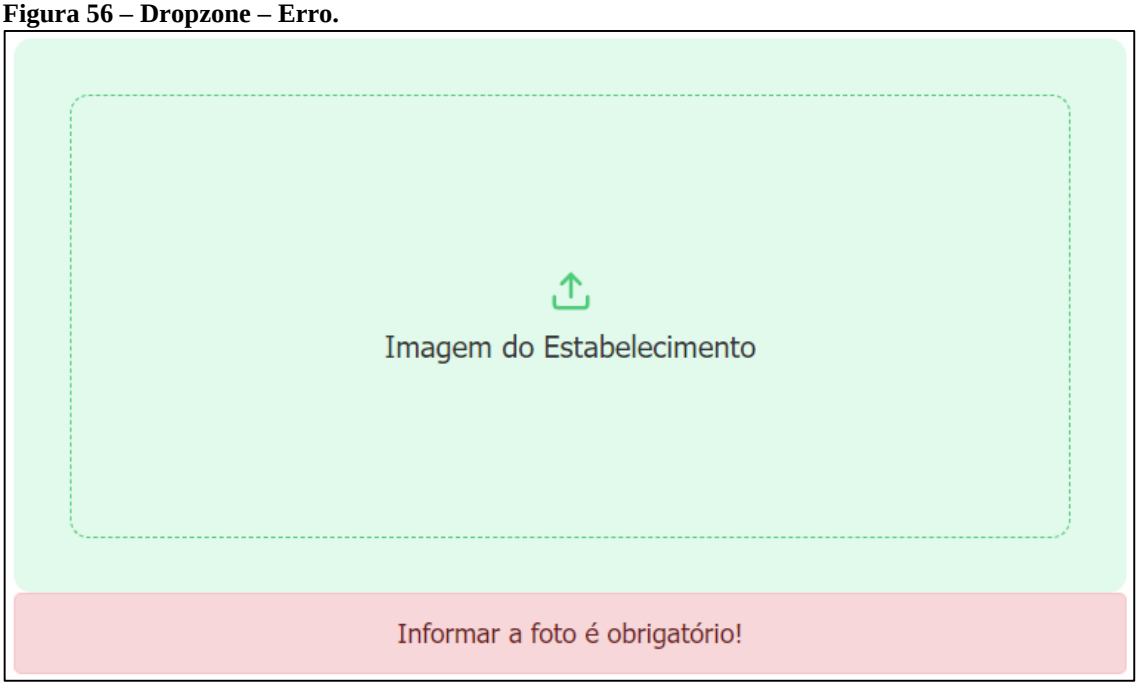
Caso o usuário queira visualizar qualquer imagem das seções “Imagens Selecionadas” ou “Fotos Atuais”, basta passar o mouse por cima de qualquer uma das imagens. A imagem em foco será exibida em alta definição sob o campo Dropzone, como pode ser visto na Figura 55.

Figura 55 – Dropzone – Visualização da foto selecionada.

Fonte: Acervo do autor (2022)

Caso haja um equívoco com a imagem, o componente apresenta imediatamente abaixo do Dropzone uma mensagem informando o erro que ocorreu, conforme demonstrado na Figura 56.

Figura 56 – Dropzone – Erro.



Fonte: Acervo do autor (2022)

5. CONCLUSÃO

Este trabalho teve o propósito de desenvolver um protótipo de uma aplicação web. Havia o intuito de possibilitar o cadastro de pontos que disponibilizassem a venda de produtos para intolerantes e na consulta de produtos carregar todos os pontos próximos ao usuário em um raio de 10 quilômetros, para permitir com que o usuário descubra novos pontos ou mesmo consulte os produtos em estabelecimentos já conhecidos.

As tecnologias utilizadas, como ferramentas ou linguagens de programação, foram decisivas para possibilitar o processo de desenvolvimento do protótipo. Para o back-end o Node.js utilizado através do f*ramework* Express foi essencial. A utilização desse *framework* possibilitou a criação de uma API RESTful, onde todos os acessos para a aplicação via rotas foram devidamente tratados e protegidos.

Para o desenvolvimento do código foram utilizados os princípios de orientação a objeto, como heranças, encapsulamento e abstração de classes como controllers, models, repositories e services. Para facilitar a comunicação com o banco de dados em um ambiente back-end escrito com JavaScript foi utilizado o ORM TypeORM, que possibilitou uma fácil e rápida comunicação com o banco de dados. O Princípio de Substituição de Liskov foi usado durante a implementação para facilitar a manutenção, caso seja necessário trocar de ORM no futuro da aplicação.

Para o *front-end* a utilização da biblioteca React JS, que possibilitou a criação de componentes e de muita dinamicidade ao projeto, integrado ao uso de HTML e ao CSS possibilitou um desenvolvimento responsivo para toda a aplicação. Para proporcionar a garantia de que todos os dados sejam íntegros e estejam seguros, foi utilizado o PostgreSQL como banco de dados.

Com relação aos objetivos específicos do trabalho, o primeiro objetivo, que pretendia definir as tecnologias utilizadas durante o desenvolvimento do protótipo, foi concretizado através da revisão da literatura.

O segundo objetivo, ao qual correspondia identificar softwares já existentes no mercado que possuem uma finalidade semelhante e explicar seu funcionamento, foi alcançado através do desenvolvimento do estado da arte, onde foram encontrados os aplicativos Gaya Food e GoFind.

O terceiro objetivo, ao qual propunha elaborar os requisitos e os diagramas visando a construção da análise do protótipo, foi elaborado na seção de análise, onde houve o levantamento de requisitos funcionais, requisitos não funcionais e regras de negócio para o

devido funcionamento e uso do sistema. Também foram elaborados os diagramas de caso de uso, os diagramas de atividades referentes ao cadastro de estabelecimento e cadastro de produtos e ainda o diagrama de entidade-relacionamento, onde foram explicados todos os relacionamentos entre entidades das tabelas construídas no banco de dados.

O último objetivo visava a aplicação das tecnologias selecionadas para o desenvolvimento do protótipo, o qual foi concluído e apresentado no capítulo referente ao desenvolvimento da aplicação. Foram apresentadas e detalhadas as ferramentas utilizadas na construção do protótipo e na sua implementação, além de explicar todas as telas do sistema e exibi-las para computador, tablet e celular. Foram apresentados dois componentes (Input e Dropzone) presentes nas duas telas de cadastro, além de toda a estrutura de pastas utilizada tanto no back-end como no front-end.

O protótipo desenvolvido disponibiliza ao usuário a oportunidade de poder consultar pontos alimentícios com produtos para intolerantes próximos a ele. É um facilitador para o usuário que possui alguma intolerância alimentícia, pois agiliza o processo de encontrar novos estabelecimentos ou mesmo de consultar a variedade e os preços dos produtos que deseja consumir de forma rápida e prática. Da mesma forma, ajuda a divulgar os estabelecimentos para os usuários que não os conhecem, sendo uma relação benéfica para ambos.

5.1 RECOMENDAÇÕES PARA TRABALHOS FUTUROS

Para o desenvolvimento futuro do trabalho, uma recomendação que poderia ser aplicada seria a possibilidade de filtrar os estabelecimentos por categoria, exibindo apenas os pontos alimentícios que possuem produtos de determinada categoria.

Para tornar o protótipo em um sistema mais robusto, seria necessário aplicar um método para recuperação de senha do estabelecimento, além de haver mais segurança no cadastro do mesmo, através do envio de tokens temporários via e-mail.

Também poderia haver integração com as APIs do Facebook ou do Google para possibilitar o login e o cadastro de estabelecimentos, sendo um grande facilitador para os que que já possuem contas nesses sistemas.

Por último, seria interessante a aplicação de um sistema de pagamentos, realizando cobrança mensal, trimestral ou anual para os estabelecimentos. Esta seria uma forma de monetização para garantir uma receita e assim auxiliar no custeio da manutenção do sistema.

REFERÊNCIAS

ALVES, William Pereira. **Banco de dados**. São Paulo Erica 2014. *Ebook*

BOOTSTRAP. ***Build fast, responsive sites with Bootstrap***. 2021. Disponível em: https://getbootstrap.com/. Acesso em: 27 mar. 2022.

BARBOZA, Fabrício Felipe Meleto. Modelagem e desenvolvimento de banco de dados. Porto Alegre SAGAH 2018 1 recurso online ISBN 9788595025172.

BORGES, Olimar. T.; COUTO, Júlia.M. C; SIMAS, Victor. L., et. **Desenvolvimento para dispositivos móveis. Volume 2.** Porto Alegre: SAGAH, 2019. Ebook

BRASIL. **Princípios da LGPD**. 2021a. Disponível em: https://www.gov.br/cidadania/pt-br/acesso-a-informacao/lgpd/principios-da-lgpd. Acesso em: 23 mar. 2022

BRASIL. **Lei Geral de Proteção de Dados Pessoais (LGPD)**. 2021b. Disponível em: https://www.gov.br/cidadania/pt-br/acesso-a-informacao/lgpd/principios-da-lgpd. Acesso em: 23 mar. 2022.

CORONEL, Carlos; ROB, Peter. **Sistemas de banco de dados:** projeto, implementação e gerenciamento. São Paulo: Cengage Learning, 2011. Ebook

DATE, C. J. **Introdução a sistemas de bancos de dados.** Rio de Janeiro: Elsevier, 2004. 864 p.

EXPRESS. **Framework *web* rápido, flexível e minimalista para Node.js**. Disponível em: https://expressjs.com/pt-br/. Acesso em: 27 mar. 2022.

FLANAGAN, David. **JavaScript.** O guia definitivo. Porto Alegre Bookman 2014. *Ebook*

GAYA FOOD. **Gaya Food - Comida Saudável**. Disponível em: https://apps.apple.com/pt/app/gaya-food-comida-saud%C3%A1vel/id1485497615. Acesso em: 01 set. 2022.

GEHRKE, Johannes Coautor; RAMAKRISHNAN, Raghu. **Sistemas de gerenciamento de banco de dados**. Porto Alegre: AMGH, 2008. *Ebook.*

GOFIND. GoFind. Disponível em: https://www.gofind.online/. Acesso em: 02 set. 2022.

HIRAMA, Kechi. **Engenharia de *Software***: Qualidade e Produtividade com Tecnologia. Rio de Janeiro: Grupo GEN, 2011. *Ebook*

JSON WEB TOKEN. ***Documentation*.** Disponível em: https://jwt.io. Acesso em: 19 set. 2022.

MDN. **Métodos de requisição HTTP**. 2021. Disponível em: https://developer.mozilla.org/pt-BR/docs/*Web*/HTTP/Methods. Acesso em: 10 jun. 2022.

MULTER. ***Documentation*.** Disponível em: https://github.com/expressjs/multer. Acesso em: 19 set. 2022.

MILETTO, Evandro Manara. **Desenvolvimento de *software* II**: introdução ao desenvolvimento *web* com html, css, JavaScript e php. Porto Alegre Bookman 2014. *Ebook*

NODEJS. **Documentação de referência de API**. Disponível em: https://nodejs.org/pt-br/docs/. Acesso em: 27 mar. 2022.

NEVES, Denise Lemes Fernandes. **PostgreSQL.** Conceitos e aplicações**.** São Paulo: Érica, 2002.

OLIVEIRA, Cláudio Luís Vieira. **JavaScript descomplicado.** Programação para a Web, IoT e dispositivos móveis. São Paulo. Erica 2020.

OLIVEIRA, Cláudio Luís Vieira. **Node.js.** Programe de forma rápida e prática**.** São Paulo Expressa 2021 1

PÁDUA, P.F.W. D. **Engenharia de *Software***. Projetos e Processos. Vol. 2. Rio de Janeiro: Grupo GEN, 2019. *Ebook*

POSTGRESQL. ***Documentation*.** Disponível em: https://www.postgresql.org/docs/. Acesso em: 12 mar. 2022.

PRESSMAN. Roger S, **Engenharia de Software.** São Paulo, McGraw-Hill, 2006, 6ª.edição

REACTJS. ***A JavaScript library for building user interfaces***. Disponível em: https://reactjs.org/. Acesso em: 26 mar. 2022.

SAÚDE, Atlas da. **História da doença celíaca.** Disponível em: https://www.atlasdasaude.pt/publico/content/historia-da-doenca-celiaca. Acesso em: 27-03-2022

SCHÖNIG, Hans-Jürgen. ***Mastering* PostgreSQL 12**. Packt Publishing Limited, 2019. E-book.

SILVA, Maurício Samy. **Criando sites com HTML.** Sites de alta qualidade com HTML e CSS. São Paulo: Novatec, 2008.

STYLED COMPONENTS. ***Documentation*.** Disponível em: https://styled-components.com/docs. Acesso em: 19 set. 2022.

TERUEL, Evandro Carlos. **HTML 5.** Guia Prático. São Paulo: Érica, 2014. *Ebook.*

TERADA, Routo. **Segurança de dados.** Criptografia em rede de computador. São Paulo Blucher 2008. *Ebook.*

TYPEORM. ***Documentation*.** Disponível em: https://typeorm.io. Acesso em: 19 set. 2022.

VALENTE, Marco Tulio. Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade. 2020. 502 p. *Ebook.*

V8. ***Documentation*.** Disponível em: https://v8.dev/docs. Acesso em: 19 set. 2022.

W3SCHOOLS. ***Learn to code***. Disponível em: https://www.w3schools.com/. Acesso em: 26 mar. 2022.