CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

Faculdade de Tecnologia Baixada Santista Rubens Lara

Curso Superior de Tecnologia em Sistemas para Internet

Gabriel Lins Soares Silvestre
Larissa Lemos Lucio
Raissa Soares Lima

BAKER:
Site para Divulgar Padeiros Microempreendedores

Santos, SP
2024

Gabriel Lins Soares Silvestre
Larissa Lemos Lucio
Raissa Soares Lima

BAKER:
Site para Divulgar Padeiros Microempreendedores

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia Rubens Lara, como exigência para a obtenção do Título de Tecnólogo em Sistemas para Internet.

Orientador: Prof. Me. Davi Silvestre Moreira dos Reis

Santos, SP
2024

RESUMO

Cresce o número de padarias no Brasil, estudos apontam que 154 novos estabelecimentos dessa categoria foram abertos por dia no primeiro trimestre de 2023. A partir dessa informação, esse trabalho tem o objetivo de desenvolver uma aplicação *web* para o processo de pedidos de pães e produtos relacionados, por meio de um cardápio digital. Pesquisas foram realizadas acerca do aumento de microempreendedores nesse mercado. A aplicação oferece a oportunidade para a realização de vendas em uma plataforma específica para o ramo de panificação, levando em consideração que muitos padeiros não possuem plataforma própria e realizam vendas por meio de redes sociais. Além de oferecer uma oportunidade de alcançar novos clientes, ela também simplifica o atendimento a diferentes tipos de público através de filtros no site, abrangendo desde aqueles que possuem restrições alimentares até os que buscam uma opção mais saudável de estilo de vida a encontrarem os produtos desejados de maneira mais rápida e eficaz. Os resultados mostram que a aplicação atenderia os principais requisitos para venda e divulgação, o que ajuda tanto o padeiro que realiza a venda quanto o cliente que busca facilidade para compra.

Palavras-chaves: Microempreendedores. Negócios. Panificação.

ABSTRACT

Nowadays, the number of bakeries are growing up in Brasil, some studies shows us that 154 new establishments of this category were open per day on the first three months of the year of 2023. Based on this information, this academic work aims to develop a web application for the ordering process of breads and related products using a digital menu. Searches were realized about the microentrepreneurs increasing in this market. The application offers selling opportunities through the social midia and reaching new costumers. The web application also simplify the process of ordering for people who has dietary restrictions and who wants a healthier life through a filter in the website for different clients. The results show that the application would comply with the main requirements of selling and disclosuring, that helps the baker with sales and the costumers for finding an easy buying.

Keywords: Micro-entrepreneurs. Businesses. Bakery.

LISTA DE ABREVIATURAS E SIGLAS

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Segundo um levantamento feito pelo *SEBRAE* (2022), a partir de dados da Classificação Nacional de Atividades Econômicas foi possível verificar que no primeiro semestre de 2022 o setor de panificação alcançou o maior número de negócios formais em operação dos últimos quatro anos antecessores, registrando a marca de 273,5 mil, entre microempreendedores individuais (*MEIs*), microempresas e empresas de pequeno porte.

O *SEBRAE* (2023) a partir das informações do Cadastro Nacional de Pessoa Jurídica (*CNPJ*) da Receita Federal pode levantar que 154 novas padarias foram abertas por dia no primeiro trimestre de 2023. Desde 2019, o setor atrai empreendedores, fazendo com que o país atinja um novo recorde de padarias com 295 mil abertas, e entre o total desses estabelecimentos, 70% são de *MEIs*.

No ranking das 10 maiores cidades com padarias com o *CNPJ* ativo, segundo dados da Receita Federal, São Paulo está em primeiro lugar, seguido de Rio de Janeiro e Brasília.

Os números citados mostram que o setor de panificação tende a crescer cada vez mais no Brasil, sendo formado em sua maioria, por *MEIs*. Apesar do crescimento, nota-se a falta da tecnologia aliada ao setor, dificultando muitas vezes a renda do empreendedor por falta de divulgação do seu negócio de maneira facilitada.

Ao refletir sobre essa questão, revela-se uma necessidade da criação de um *site web* que forneça as ferramentas necessárias para auxiliar os *MEIs* e empreendedores no seu dia a dia, conectando os padeiros com os clientes.

> Formação técnica e capacidade de gestão também fazem parte do capital humano dos empreendedores. O desafio encontrado pela maior parte dos microempreendedores é conseguir gerir o seu próprio negócio com baixa escolaridade e sem conhecer ferramentas de administração de produção. Os problemas básicos de gestão são percebidos quando uma parte considerável dos empreendimentos não controla de forma alguma as contas do negócio. (Fábio, 2011, p. 14)

Assim surgiu a ideia do “*Baker*”, uma plataforma *web* que visa modernizar as vendas do setor de panificação. Oferecendo ferramentas para filtrar pelos padeiros mais próximos do código de endereçamento postal (*CEP)* do cliente, por restrição alimentar e facilitando a realização de pedidos por meio de *e-mail* através da plataforma.

O principal objetivo do site é facilitar a rotina do padeiro, proporcionando ferramentas para modernização do seu negócio, incluindo um *dashboard* de vendas e histórico de pedidos, além de cadastrar todos seus produtos com suas descrições e imagens, fornecendo assim, um cardápio completo para seus clientes visualizarem através do seu perfil na aplicação web *Baker*.

Baker é um *site* para vendas que fornece oportunidades de melhorias e tecnologias acessíveis a todos do setor de panificação, promovendo a divulgação de seus produtos, modernização e praticidade no dia a dia.

1.1 OBJETIVO

O *site* tem como principal objetivo ajudar pequenos microempreendedores na missão de modernização e tecnologia, criando um sistema acessível a todos e de fácil uso, gerando um espaço para venda e compra de maneira otimizada e intuitiva. Esse projeto busca criar um ambiente tecnológico para aprimorar a experiência do setor.

1.1.1 OBJETIVO GERAL

O objetivo do *site Baker* é promover padeiros microempreendedores, facilitando a conexão entre eles e potenciais clientes. Através de um catálogo de produtos, os usuários podem filtrar itens por restrições alimentares e adicioná-los a um carrinho de compras, permitindo a realização de pedidos.

1.1.2 OBJETIVOS ESPECÍFICOS

Objetivos específicos do projeto:

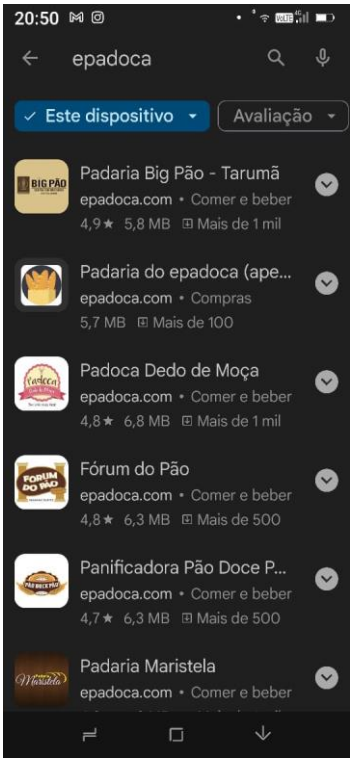
- Realizar um estudo para análise e avaliação da necessidade da aplicação web;
- Levantamento de requisitos e funcionalidades necessárias para uma aplicação do setor de panificação;
- Criar uma aplicação que traga tecnologia para auxiliar os MEIs do setor no dia a dia.

1.2 ESTADO DA ARTE

Como principais referências para esse projeto, observa-se o sistema Epadoca, o aplicativo Pão Quentinho Saindo e o site *HolySoup.* No entanto, é fundamental destacar que os dois primeiros possuem um foco direcionado para padarias, enquanto o nosso site, denominado *Baker*, tem como objetivo principal dar visibilidade aos microempreendedores padeiros. Enquanto, o *HolySoup* é um *site* de sopas instantâneas naturais, em que você seleciona o produto desejado e recebe em sua residência.

A principal funcionalidade do Epadoca que motivou o projeto de conclusão de curso reside na capacidade desse sistema de oferecer uma visibilidade ampla e eficaz para cada padaria. No entanto, um obstáculo significativo foi identificado neste sistema: a necessidade de baixar individualmente o aplicativo de cada padaria para adquirir seus produtos. Essa limitação foi a fonte de inspiração para a criação do site *Baker*, que visa unificar todos os padeiros em um único local, simplificando assim o processo de compra para os usuários.

Ilustração 1 – Tela do aplicativo Epadoca



Fonte: Epadoca

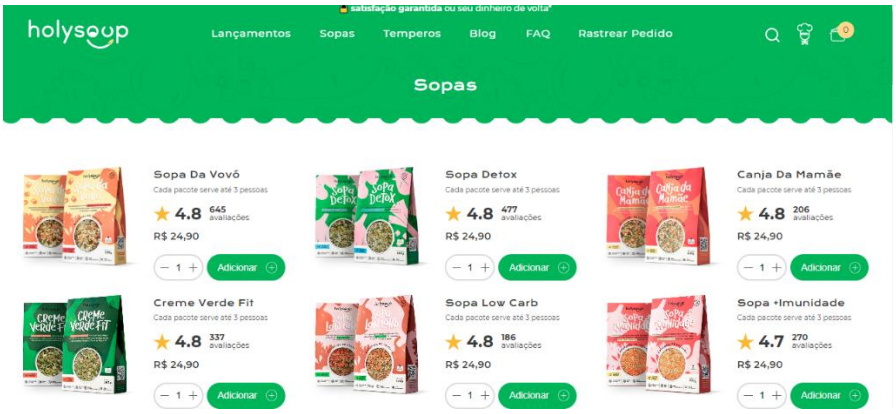
Em relação ao aplicativo Pão Quentinho Saindo destaca-se a sua capacidade de centralizar todas as padarias em um único lugar, tornando mais simples para os usuários encontrarem os produtos desejados de forma conveniente.

Ilustração 2 - Tela do aplicativo Pão Quentinho Saindo!

Fonte: Pão Quentinho Saindo!

O que foi observado de interessante no *site HolySoup* foi a facilidade de adquirir um produto em uma interface intuitiva. Necessita-se trazer para o *Baker* referente a esse sítio *web*, é a praticidade de realizar a compra dos produtos desejados a qualquer momento.

Ilustração 3- Tela do site *HolySoup*



Fonte: https://holysoup.com.br/collections/sopas, 2024

2 DESENVOLVIMENTO

O capítulo abaixo descreve as etapas de desenvolvimento do *Baker* site para divulgar padeiros microempreendedores. Os subcapítulos têm como objetivo mapear a análise do sistema, a análise de requisitos, o diagrama de caso de uso, o fluxo de eventos, o banco de dados, a camada de negócios e a camada de apresentação. Assim, cada um descrevendo as metodologias e ferramentas utilizadas para a obtenção do resultado perante o desenvolvimento do projeto.

2.1 ANÁLISE DO SISTEMA

Foi escolhida a linguagem de modelagem unificada (*UML*) para o desenvolvimento da análise do sistema, devido a sua alta utilização em projetos para modelagem de software por meio do paradigma de Orientação a Objetos. Através da análise do autor Gilleanes T. A. Guedes, a *UML* tem como objetivo principal definir as características do sistema e seus requisitos, através do seu comportamento, estrutura lógica, dinâmica de processos e necessidades físicas quando aplicável.

O site *Baker* possui ao todo 12 páginas com o propósito principal de divulgar padeiros microempreendedores, das quais 8 são destinadas aos usuários e 4 para a administração dos padeiros. As funcionalidades principais serão: a busca do padeiro pelo *CEP* inserido em um campo de pesquisa, o que resultará nos três padeiros mais próximos da localidade; a indicação de um padeiro de acordo com as restrições alimentares apresentadas pelo usuário; a opção para o usuário encaminhar o seu pedido através do e-mail do padeiro; e, por fim, uma página que proporciona a visão do padeiro para uma análise feita referente aos produtos mais pedidos, quantidade de vendas nos últimos 30 dias e gráfico mostrando a quantidade de cada produto vendido nos últimos 30 dias.

Com base nos objetivos do projeto, as linguagens de programação escolhidas para o *front-end* foram *html*, *css* e *javascript*, para o desenvolvimento *back-end* .*NET* e o banco selecionado foi o *SQL server*.

A linguagem de marcação escolhida foi a *HTML5*, pois, com os padrões estabelecidos pela comunidade *W3C* ela se torna uma potente ferramenta de grande aplicabilidade em diversas plataformas. O autor Fábio Flatschart afirma que a Linguagem de Marcação de Hipertexto é a principal utilizada na *web*, o que reforça a

escolha para implementação no site *Baker*. E, para sustentar a usabilidade do sistema, o *CSS3* também é adotado como linguagem de estilo, o que garante o desempenho responsivo para o site ser navegado.

Com o intuito de assegurar as funcionalidades navegáveis e interativas do site desenvolvidas através do *HTML* e *CSS*, a implementação de *JavaScript* é fundamental, e segundo Preston Prescott, por essa linguagem ser classificada como *script*, não é possível ser executada por conta própria, então é necessário que se tenha um código *HTML* vinculado como base.

A partir das informações esclarecidas pelo autor Fábio Flatschart, um projeto web precisa possuir a Camada de desenvolvimento do lado do usuário, e as três ferramentas mais comuns desse grupo são: *HTML*, *CSS* e *JavaScript*. Assim, adotando essas três linguagens como pilares para o desenvolvimento *front-end* do *site Baker*.

> A organização do desenvolvimento de projetos web em camadas independentes confere flexibilidade e modularidade ao fluxo de trabalho para web integrando equipes multidisciplinares de planejamento, produção, arquitetura da informação, design e programação.
> (Fábio, 2011, p. 14)

O *framework* escolhido para produzir a parte funcional de *back-end* do site *Baker* foi *.NET*, que possui um conjunto abrangente de bibliotecas, ferramentas e modelos de desenvolvimento que permitem o desenvolvimento rápido de uma aplicação. Segundo DARSHAN, ele define um ambiente que suporta o desenvolvimento e a execução de aplicativos altamente distribuídos e baseados em componentes, permitindo que diferentes linguagens de computador trabalharem juntas com segurança, sendo um modelo de programação comum para a plataforma *Windows*.

Para o banco de dados do sistema, foi escolhido o *Microsoft Structured Query Language (SQL) server*, pois, *.NET* e *SQL Server* são ambas tecnologias da *Microsoft*, sendo altamente integradas e projetadas para funcionarem com boa compatibilidade juntas.

2.1.1 ANÁLISE DE REQUISITOS

Os fatores decisivos para o desenvolvimento de um produto são os requisitos do sistema. Segundo MACHADO (2018), eles são o ponto de partida para toda a definição de um projeto.

Requisitos Funcionais

Segundo CUNHA e RIBEIRO (2018), requisitos funcionais, como representado na Ilustração 4, são aqueles que descrevem as funcionalidades que se espera que um sistema disponibilize, de uma forma completa e consistente.

Abaixo, serão apresentados os requisitos funcionais dessa aplicação do usuário cadastrado como cliente.

Ilustração 4- Requisitos funcionais do usuário

Requisito	Descrição
RNF001	O sistema deve permitir realizar cadastro solicitando nome, e-mail, telefone, estado, endereço, cidade, CEP, senha, CNPJ no caso do padeiro ou cadastro de pessoas físicas (CPF) no caso do cliente
RNF002	O sistema deve permitir fazer *login* utilizando *e-mail* e senha
RNF003	O sistema deve permitir editar informações pessoais
RNF004	O sistema deve permitir filtrar padeiros/produtos

RNF005	O sistema deve permitir filtrar por CEP
RNF006	O sistema deve permitir achar o seu padeiro ideal através de respostas sobre restrições alimentares
RNF007	O sistema deve permitir visualizar o mapa com os padeiros mais próximos cadastrados
RNF008	O sistema deve permitir visualizar histórico de pedidos
RNF009	O sistema deve permitir adicionar produtos no carrinho
RNF010	O sistema deve permitir finalizar o pedido enviando via *e-mail*

Fonte: Elaborado pelos autores (2024)

Abaixo, serão apresentados na Ilustração 5, os requisitos funcionais dessa aplicação do usuário cadastrado como padeiro.

Ilustração 5- Requisitos funcionais do padeiro

Requisito	Descrição

RNF001	O sistema deve permitir gerenciar os produtos
RNF002	O sistema deve permitir visualizar os pedidos recebidos
RNF003	O sistema deve permitir visualizar o *dashboard* de vendas

Fonte: Elaborado pelos autores (2024)

Requisitos não funcionais

Segundo LIBA (2011) requisitos não funcionais, como representado na Ilustração 6, descrevem as qualidades requeridas para um sistema, como sua usabilidade e características de desempenho.

Abaixo, serão apresentados os requisitos não funcionais dessa aplicação.

Ilustração 6 - Requisitos não funcionais do sistema

Requisito	Descrição
RNF001	O sistema deve ser responsivo
RNF002	O sistema deve permitir visualizar o mapa com os padeiros mais próximos cadastrados

RNF003	O sistema deve permitir filtrar por CEP e/ou restrição
RNF004	O sistema deve permitir finalizar o pedido enviando via *e-mail*
RNF005	O sistema deve permitir visualizar *dashboard* de vendas

Fonte: Elaborado pelos autores (2024)

2.1.2 DIAGRAMA DE CASO DE USO

Segundo GUEDES (2018), o diagrama de caso de uso busca possibilitar a compreensão do sistema por meio da perspectiva do usuário, apresentando uma visão externa geral das funcionalidades oferecidas pelo sistema.

O Diagrama de Caso de Uso pode ser consultado no Apêndice A.

Em seguida, serão detalhados os principais casos de uso, isto é, primeiro do usuário e em seguida os específicos do cadastro do usuário como padeiro.

- Ilustração 7, apresenta o fluxo principal e os fluxos alternativos do caso de uso “Realizar cadastro do usuário”.

Ilustração 7 - Caso de Uso 1 - Realizar cadastro do usuário

UC001 – Realizar cadastro do usuário
Descrição: Realizar o cadastro do usuário no *site*
Pré-condição: -
Pós-condição: O usuário poderá utilizar o *site*, entrando com o seu e-mail e senha de cadastro
Requisitos funcionais: RF001
Fluxo Principal
Passo 1: O usuário irá abrir o site em seu dispositivo
Passo 2: Irá aparecer na tela inicial um ícone representando um usuário

Passo 3: O usuário clicará no botão e conterá na tela de “*Login*” uma mensagem direcionando para tela de “Cadastro” via *link*
Passo 4: O usuário clicará no *link* e será direcionado para um formulário contendo nome, e-mail, telefone, estado, cidade, endereço, CEP e senha para serem preenchidos
Passo 5: O usuário irá preencher o campo de CNPJ no caso de o cadastro ser como padeiro ou o campo de CPF no caso de o cadastro ser como cliente
Passo 6: O usuário irá preencher todas as informações e enviará para cadastro clicando no botão “Finalizar Cadastro” e será direcionado em seguida para a tela inicial do *site*
Fluxo Alternativo A
Passo 1: Ao preencher o campo de *e-mail* (passo 4), o usuário preenche com um *e-mail* já cadastrado no sistema
Passo 2: O usuário não consegue realizar o cadastro
Passo 3: Ao terminar de modificar o *e-mail*, o usuário irá enviar as informações clicando no botão “Finalizar Cadastro” e será direcionado em seguida para a tela inicial do *site*
Fluxo Alternativo B
Passo 1: Ao preencher o campo CNPJ (passo 4), o usuário preenche com um CNPJ inexistente
Passo 2: O usuário não consegue realizar o cadastro
Passo 3: Ao terminar de modificar o CNPJ, o usuário irá enviar as informações clicando no botão “Finalizar Cadastro” e será direcionado em seguida para a tela inicial do *site*

Fonte: Elaborado pelos autores (2024)

- Ilustração 8, apresenta o fluxo principal e o fluxo alternativo do caso de uso “Fazer *login* do usuário”.

Ilustração 8 - Caso de Uso 2 - Fazer login do usuário

UC002 – Fazer login do usuário
Descrição: O usuário irá realizar o login no *site* utilizando e-mail e senha cadastrado
Pré-condição: Ter realizado o cadastro anteriormente
Pós-condição: O usuário será direcionado para tela inicial do *site*
Requisitos funcionais: RF002
Fluxo Principal
Passo 1: O usuário irá abrir o *site* em seu dispositivo
Passo 2: Irá aparecer na tela inicial um ícone representando um usuário
Passo 3: O usuário clicará no botão e será direcionado para um formulário contendo *e-mail* e senha para serem preenchidos
Passo 4: O usuário irá preencher os campos com suas informações de cadastro e enviará as informações clicando no botão “Entrar” e será direcionado em seguida para a tela inicial do *site*
Fluxo Alternativo A

Passo 1: Ao preencher os campos (passo 4), o usuário preenche alguma informação de maneira incorreta
Passo 2: O usuário não consegue realizar o login
Passo 3: Ao terminar de modificar as informações incorretas, o usuário irá enviar as informações clicando no botão “Entrar” e será direcionado em seguida para tela inicial a do *site*

Fonte: Elaborado pelos autores (2024)

- Ilustração 9, apresenta o fluxo principal e o fluxo alternativo do caso de uso “Editar informações pessoais do usuário”.

Ilustração 9 - Caso de Uso 3 - Editar informações pessoais do usuário

UC003 – Editar informações pessoais do usuário
Descrição: O usuário poderá editar suas informações pessoais no *site*
Pré-condição: Ter realizado o *login* anteriormente
Pós-condição: O usuário terá suas informações modificadas
Requisitos funcionais: RF003
Fluxo Principal
Passo 1: O usuário irá abrir o site em seu dispositivo e realizar o *login*
Passo 2: Irá aparecer na tela inicial um *link* no menu “Configurações”
Passo 3: Clicando no *link*, o usuário será direcionado para página que contém suas informações
Passo 4: O usuário irá clicar no campo desejado e modificará as informações após digitar a senha
Passo 5: Após realizar as mudanças desejadas terá um botão “Salvar Dados” para atualizar as modificações no sistema

Fonte: Elaborado pelos autores (2024)

- Ilustração 10, apresenta o fluxo principal e o fluxo alternativo do caso de uso “Filtrar padeiros/produtos”.

Ilustração 10 - Caso de uso 4 - Filtrar padeiros/produtos

UC004 – Filtrar padeiros/produtos
Descrição: O usuário irá filtrar os padeiros/produtos do *site*
Pré-condição: -
Pós-condição: Irá aparecer as informações de padeiros/produtos
Requisitos funcionais: RF004
Fluxo Principal
Passo 1: O usuário irá abrir o *site* em seu dispositivo
Passo 2: Irá aparecer na tela inicial um link no menu chamado “Produtos do Padeiro”
Passo 3: O usuário clicará no *link* e será direcionado para a página
Passo 4: O usuário irá abrir a filtragem para selecionar o filtro de produto desejado ou o *Search* escrevendo o nome do padeiro

Passo 5: O usuário visualizará os padeiros correspondentes ao filtro aplicado
Fluxo Alternativo A
Passo 1: O usuário recebe uma mensagem na tela notificando que não foram encontrados resultados para o filtro desejado
Passo 2: O usuário seleciona novamente um novo filtro de padeiros/produtos (passo 4)
Passo 3: O usuário visualizará os padeiros correspondentes ao filtro aplicado

Fonte: Elaborado pelos autores (2024)

- Ilustração 11, apresenta o fluxo principal e o fluxo alternativo do caso de uso “Filtrar por CEP”.

Ilustração 11 - Caso de uso 5 - Filtrar por CEP

UC005 – Filtrar por CEP
Descrição: O usuário irá filtrar padeiros pelo seu CEP
Pré-condição: -
Pós-condição: Irá aparecer as informações de acordo com o filtro
Requisitos funcionais: RF005
Fluxo Principal
Passo 1: O usuário irá abrir o *site* em seu dispositivo
Passo 2: Irá aparecer na tela inicial um *link* no menu “Padeiros Próximos”
Passo 3: O usuário clicará no *link* e será direcionado para a página
Passo 4: O usuário irá buscar pelo CEP desejado
Passo 5: O usuário visualizará os padeiros mais próximos ao CEP informado
Fluxo Alternativo A
Passo 1: O usuário recebe uma mensagem na tela notificando que não foram encontrados resultados para o filtro desejado
Passo 2: O usuário seleciona novamente um novo filtro de CEP
Passo 3: O usuário visualizará os padeiros mais próximos ao CEP informado

Fonte: Elaborado pelos autores (2024)

- Ilustração 12, apresenta o fluxo principal e o fluxo alternativo do caso de uso “Filtrar por restrição”.

Ilustração 12 - Caso de Uso 6 - Filtrar por restrição

UC006 – Filtrar por restrição
Descrição: O usuário irá filtrar padeiros que trabalham com os produtos indicados para o seu tipo de restrição
Pré-condição: -
Pós-condição: Irá aparecer as informações de acordo com os requisitos selecionados
Requisitos funcionais: RF006
Fluxo Principal
Passo 1: O usuário irá abrir o *site* em seu dispositivo
Passo 2: Irá aparecer na tela inicial um *link* no menu “Padeiro Ideal”

Passo 3: O usuário clicará no *link* e será direcionado para a página
Passo 3: O usuário irá responder a algumas perguntas sobre restrições, colocará o CEP e enviará as informações
Passo 4: O usuário terá como resultado os padeiros que correspondem ao seu perfil de escolhas próximos ao CEP informado
Fluxo Alternativo A
Passo 1: O usuário receberá uma mensagem na tela notificando que não foram encontrados resultados para o filtro desejado
Passo 2: O usuário responderá novamente sobre restrições e um novo filtro de CEP
Passo 3: O usuário terá como resultado os padeiros que correspondem ao seu perfil de escolhas próximos ao CEP informado

Fonte: Elaborado pelos autores (2024)

- Ilustração 13, apresenta o fluxo principal e o fluxo alternativo do caso de uso “Mapa de padeiros próximos cadastrados”.

Ilustração 13 - Caso de Uso 7 - Mapa de padeiros próximos cadastrados

UC007 – Mapa de padeiros próximos cadastrados
Descrição: O usuário irá visualizar um mapa com os padeiros cadastrados mais próximos
Pré-condição: Colocar o filtro do seu CEP
Pós-condição: Irá aparecer onde estão os padeiros mais próximos do seu endereço no mapa
Requisitos funcionais: RF007
Fluxo Principal
Passo 1: O usuário irá abrir o *site* em seu dispositivo
Passo 2: Irá aparecer na tela inicial um *link* no menu “Padeiros Próximos”
Passo 3: O usuário clicará no *link* e será direcionado para a página
Passo 4: O usuário irá buscar pelo CEP desejado
Passo 5: O usuário visualizará os padeiros mais próximos indicados no mapa
Fluxo Alternativo A
Passo 1: O usuário recebe uma mensagem na tela notificando que não foram encontrados resultados para o filtro desejado
Passo 2: O usuário seleciona novamente um novo filtro de CEP
Passo 3: O usuário visualizará os padeiros mais próximos indicados no mapa

Fonte: Elaborado pelos autores (2024)

- Ilustração 14, apresenta o fluxo principal do caso de uso “Histórico de pedidos do usuário”.

Ilustração 14 - Caso de Uso 8 - Histórico de pedidos do usuário

UC008 – Histórico de pedidos do usuário
Descrição: O usuário irá visualizar todos os seus pedidos realizados
Pré-condição: Ter realizado o *login* anteriormente
Pós-condição: Visualizará os pedidos

Requisitos funcionais: RF008
Fluxo Principal
Passo 1: O usuário irá abrir o *site* em seu dispositivo e realizar o *login*
Passo 2: Irá aparecer na tela inicial um botão “Minha conta”
Passo 3: O usuário clicará no botão e será direcionado para a página
Passo 4: O usuário visualizará o histórico de pedidos

Fonte: Elaborado pelos autores (2024)

- Ilustração 15, apresenta o fluxo principal do caso de uso “Adicionar produtos no carrinho”.

Ilustração 15 - Caso de Uso 9 - Adicionar produtos no carrinho

UC009 – Adicionar produtos no carrinho
Descrição: O usuário irá conseguir selecionar os produtos desejados para o carrinho de compras
Pré-condição: Ter realizado *login*
Pós-condição: Quantidade de produtos desejada no carrinho
Requisitos funcionais: RF009
Fluxo Principal
Passo 1: O usuário irá abrir o *site* em seu dispositivo
Passo 2: O usuário irá abrir o *site* em seu dispositivo e realizar o *login*
Passo 3: Irá aparecer na tela inicial um *link* no menu chamado “Produtos do Padeiro”
Passo 4: O usuário clicará no *link* e será direcionado para a página
Passo 5: Irá escolher o padeiro desejado clicando no botão
Passo 6: Visualizará os produtos disponíveis do padeiro com nome, foto e preço
Passo 7: Selecionará a quantidade desejada do produto
Passo 8: Clicará no botão de adicionar ao carinho

Fonte: Elaborado pelos autores (2024)

- Ilustração 16, apresenta o fluxo principal do caso de uso “Finalizar pedido”.

Ilustração 16 - Caso de Uso 10 - Finalizar pedido

UC010 – Finalizar pedido
Pré-condição: Ter realizado o *login* anteriormente e adicionado os produtos no carrinho
Pós-condição: Pedido enviado para o padeiro
Requisitos funcionais: RF009
Fluxo Principal
Passo 1: O usuário irá abrir o *site* em seu dispositivo e realizar o *login*
Passo 2: O usuário irá abrir a página “Meu carrinho” que contém os produtos selecionados anteriormente

Passo 3: O usuário irá enviar o pedido clicando no botão “Realizar Pedido” escolhendo a forma de envio do mesmo pelo *e-mail*
Passo 4: Aparecerá uma mensagem na tela de “Pedido realizado com sucesso!” indicando a conclusão do pedido

Fonte: Elaborado pelos autores (2024)

- Ilustração 17, apresenta o fluxo principal e o fluxo alternativo do caso de uso “Gerenciar produtos”.

Ilustração 17 - Caso de Uso 11 - Gerenciar produtos

UC001– Gerenciar produtos
Descrição: O padeiro irá visualizar a tela para cadastro/edição/exclusão de produtos
Pré-condição: Ter realizado o *login*
Pós-condição: Alterações nos produtos cadastrados realizadas
Requisitos funcionais: RF011
Fluxo Principal
Passo 1: O padeiro irá abrir o *site* em seu dispositivo e realizar o login
Passo 2: Irá aparecer na tela inicial um botão “Área do Padeiro”
Passo 3: Ao clicar no botão, o padeiro será direcionado para página que conterá um menu com a opção “Gerenciar Produtos”
Passo 3: Ao clicar na opção o padeiro será direcionado para página que conterá seus produtos cadastrados para edição/exclusão e um botão “Adicionar produto”
Passo 4: Ao clicar no botão “Adicionar produto” o padeiro preencherá as opções necessárias e clicará no botão “Finalizar”
Passo 5: Exibirá uma mensagem na tela de “Produto cadastrado com sucesso!”
Fluxo Alternativo A
Passo 1: Ao preencher os campos (passo 4) o padeiro esquece de preencher algum campo obrigatório
Passo 2: O padeiro receberá uma mensagem na tela notificando sobre a obrigação do preenchimento
Passo 3: Ao terminar o preenchimento, o padeiro irá enviar as informações clicando no botão “Finalizar”

Fonte: Elaborado pelos autores (2024)

- Ilustração 18, apresenta o fluxo principal do caso de uso “Histórico de pedidos recebidos”.

Ilustração 18 - Caso de uso 12 - Histórico de pedidos recebidos

UC002 – Histórico de pedidos recebidos
Descrição: O padeiro irá visualizar a tela de pedidos recebidos

Pré-condição: Ter realizado o *login*
Pós-condição: Visualizará os pedidos recebidos
Requisitos funcionais: RF012
Fluxo Principal
Passo 1: O padeiro irá abrir o *site* em seu dispositivo e realizar o *login*
Passo 2: Irá aparecer na tela inicial um botão “Área do Padeiro”
Passo 3: Ao clicar no botão, o padeiro será direcionado para página que conterá um menu com a opção “Histórico de Pedidos”
Passo 4: Ao clicar na opção desejada (passo 3) visualizará todo o histórico de pedidos concluídos

Fonte: Elaborado pelos autores (2024)

- Ilustração 19, apresenta o fluxo principal do caso de uso “*Dashboard* de vendas”.

Ilustração 19 - Caso de Uso 13 - *Dashboard* de vendas

UC003 – *Dashboard* de vendas
Descrição: O padeiro irá visualizar o *dashboard* com as informações de suas vendas
Pré-condição: Ter realizado o *login*
Pós-condição: O usuário será direcionado para a tela
Requisitos funcionais: RF013
Fluxo Principal
Passo 1: O padeiro irá abrir o *site* em seu dispositivo e realizar o *login*
Passo 2: Irá aparecer na tela inicial um botão “Área do Padeiro”
Passo 3: Ao clicar no botão, o padeiro será direcionado para página que conterá um menu com a opção “*Dashboard* de Vendas”
Passo 4: Ao clicar na opção desejada (passo 3) visualizará um *dashboard* referente as suas vendas realizadas

Fonte: Elaborado pelos autores (2024)

2.1.3 FLUXO DE EVENTOS

Segundo MACORRATI (2023), fluxo de evento é uma sequência de comandos que descreve as etapas de um caso de uso através de atores, mostrando o principal caminho e os possíveis desvios pré-definidos, do início ao fim do caso de uso.

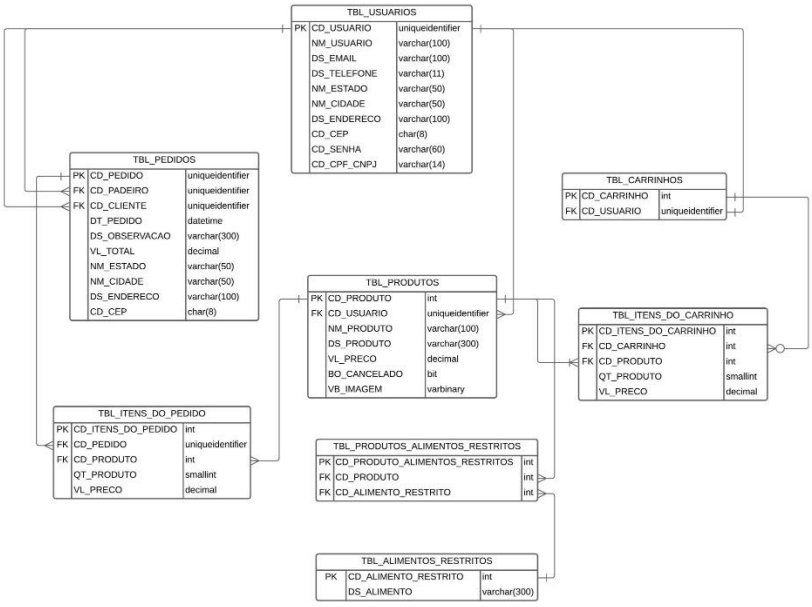
A descrição dos fluxos de eventos desse sistema pode ser consultada no Apêndice B.

2.2 BANCO DE DADOS

Banco de dados, como representado na Ilustração 20, é um sistema computadorizado para registros, cuja finalidade é facilitar o armazenamento de informações e permitir a busca e manutenção dos arquivos. Segundo DATE (2004), ele pode ser considerado como o equivalente eletrônico de um armário de arquivamento. Os usuários de um sistema podem realizar operações de acrescento, inserção, busca, exclusão, alteração de dados e remoção de arquivos existentes no banco. Os dados em questão podem ser qualquer informação valiosa para o indivíduo ou organização que ajude nas atividades que o sistema deve servir.

Optou-se pelo banco de dados da Microsoft, o SQL Server, segundo JOBSTRAIBIZER ele é reconhecido como o melhor da categoria, possuindo uma bagagem de anos de desenvolvimento e aprimoramento.

Ilustração 20 - MER banco de dados



Fonte: Elaborado pelos autores (2024)

2.3 CAMADA DE NEGÓCIO

Nesse tópico são apresentados alguns trechos de código do desenvolvimento da aplicação *Baker*.

O código representado na ilustração 21 define uma função chamada “buscarPadeirosProximos()”, que é acionada quando os usuários solicitam encontrar os três padeiros mais próximos com base em um CEP inserido, capturado e

armazenado na var CEP. Ele faz uma solicitação de busca para uma application programming interface (*API*) local, passando o CEP e a quantidade desejada de padeiros como parâmetros na uniform resource locator *(url)* do método *fetch*. Após receber a resposta da *API*, ele extrai os dados relevantes, como os códigos e nomes dos padeiros, atualiza a interface do usuário com esses dados e cria marcadores no mapa para cada padeiro encontrado através de um *loop* utilizando a instrução for. Se houver algum erro durante o processo, ele é capturado e registrado no console através da declaração *catch*.

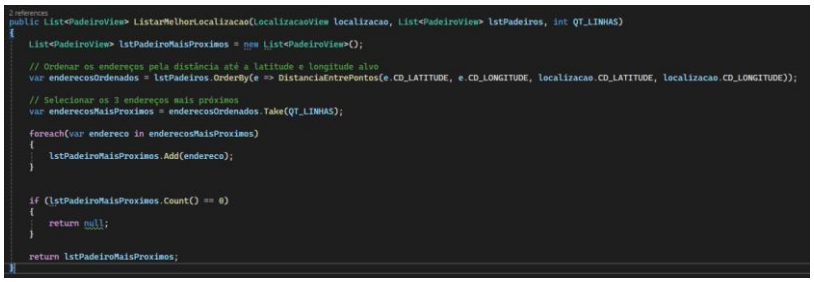
Ilustração 21 - Implementação do filtro padeiros próximos

```
function buscarPadeirosProximos() {

    var cep = document.getElementById('padeiros_proximos-search').value;
    var quantidade = 3;

    fetch(`https://localhost:7023/ListarPadeirosProximos?CEP_CLIENTE=${cep}&QT_LINHAS=${quantidade}`, {
        headers:{
            "Content-Type":"application/json",
            "Accept":"*/*"
        },
        mode:"cors"
    })
    .then(response => {
        if (!response.ok) {
            throw new Error('Erro ao fazer a requisição');
        }
        return response.json();
    })
    .then(data => {
        var dados = data.data;
        var mensagem = data.mensagem;
        var stacktrace = data.stacktrace;

        CD_PADEIRO1 = dados[0].cD_USUARIO;
        NM_PADEIRO1 = dados[0].nM_USUARIO;
        CD_PADEIRO2 = dados[1].cD_USUARIO;
        NM_PADEIRO2 = dados[1].nM_USUARIO;
        CD_PADEIRO3 = dados[2].cD_USUARIO;
        NM_PADEIRO3 = dados[2].nM_USUARIO;

        document.querySelector('#padeiro1').textContent = NM_PADEIRO1;
        document.querySelector('#padeiro2').textContent = NM_PADEIRO2;
        document.querySelector('#padeiro3').textContent = NM_PADEIRO3;

        let somaLat = dados.reduce((total, dados) => total + dados.cD_LATITUDE, 0)
        let mediaLat = somaLat / dados.length

        let somaLong = dados.reduce((total, dados) => total + dados.cD_LONGITUDE, 0)
        let mediaLong = somaLong / dados.length

        map.setCenter({ lat: mediaLat, lng: mediaLong })

        for (var i = 0; i < Math.min(dados.length, 3); i++) {
            var padeiro = dados[i];
            var marker = new google.maps.Marker({
                position: { lat: padeiro.cD_LATITUDE, lng: padeiro.cD_LONGITUDE },
                map: map,
                title: padeiro.nM_USUARIO
            });
        }
    })
    .catch(error => {
        console.error('Erro:', error.data);
    });
}
```

Fonte: Elaborado pelos autores (2024)

A ilustração 22 demonstra como o método “ListarMelhorLocalizacao()” procede ao receber uma localização de referência, uma lista de padeiros com suas respectivas localizações e a quantidade de padeiros que se deseja encontrar. Ele calcula a distância entre cada padeiro na lista e a localização de referência e, em seguida, classifica os padeiros pela distância. Os três padeiros mais próximos são selecionados através do método *Take*() e retornados em uma lista. Se não houver padeiros na lista de entrada ou nenhum padeiro for encontrado dentro da quantidade especificada, retorna-se *null*.

Ilustração 22 - Implementação do filtro padeiros próximos



```
2 references
public List<PadeiroView> ListarMelhorLocalizacao(LocalizacaoView localizacao, List<PadeiroView> lstPadeiros, int QT_LINHAS)
{
    List<PadeiroView> lstPadeiroMaisProximos = new List<PadeiroView>();

    // Ordenar os endereços pela distância até a latitude e longitude alvo
    var enderecosOrdenados = lstPadeiros.OrderBy(e => DistanciaEntrePontos(e.CD_LATITUDE, e.CD_LONGITUDE, localizacao.CD_LATITUDE, localizacao.CD_LONGITUDE));

    // Selecionar os 3 endereços mais próximos
    var enderecosMaisProximos = enderecosOrdenados.Take(QT_LINHAS);

    foreach(var endereco in enderecosMaisProximos)
    {
        lstPadeiroMaisProximos.Add(endereco);
    }


    if (lstPadeiroMaisProximos.Count() == 0)
    {
        return null;
    }

    return lstPadeiroMaisProximos;
}
```

Fonte: Elaborado pelos autores (2024)

A ilustração 23 demonstra como é realizado operações básicas de criar, ler, atualizar, deletar (*CRUD*) no banco de dados relacionado aos produtos. A classe ProdutoRepository contém métodos para inserir, atualizar, excluir e listar produtos no banco de dados. Cada método se comunica com o banco de dados por meio de um objeto *Helper*, que é uma classe auxiliar responsável por executar operações de banco de dados.

O método *Insert* insere um novo produto no banco de dados, enquanto o método *Update* atualiza um produto existente. O método *Delete* remove um produto com base em seu código único (“CD_PRODUTO”). O método *List* retorna uma lista de produtos associados a um determinado usuário, identificado por seu código único (“CD_USUARIO”).

Ilustração 23 - Implementação produtos do padeiro

```
0 references
public class ProdutoRepository
{
    const string dbName = "DB_BAKER";

    1 reference
    public void Insert(ProdutoModel produto)
    {
        Helper helper = new Helper();
        helper.ExecuteScalar(dbName, "dbo.spINSProduto", new
        {
            produto.CD_PRODUTO,
            produto.CD_USUARIO,
            produto.NM_PRODUTO,
            produto.DS_PRODUTO,
            produto.VL_PRECO,
            produto.VB_IMAGEM,
            produto.LS_ALIMENTOS_RESTRITOS
        });
    }

    1 reference
    public void Update(ProdutoModel produto)
    {
        Helper helper = new Helper();
        helper.ExecuteScalar(dbName, "dbo.spUPDProduto", new
        {
            produto.CD_PRODUTO,
            produto.CD_USUARIO,
            produto.NM_PRODUTO,
            produto.DS_PRODUTO,
            produto.VL_PRECO,
            produto.VB_IMAGEM
        });
    }

    1 reference
    public void Delete(int CD_PRODUTO)
    {
        Helper helper = new Helper();
        helper.ExecuteScalar(dbName, "dbo.spDELProduto", new
        {
            CD_PRODUTO
        });

    }

    1 reference
    public List<ProdutoModel> List(Guid CD_USUARIO)
    {
        Helper helper = new Helper();
        return helper.ExecuteList<ProdutoModel>(dbName, "dbo.spLSTProduto", new
        {
            CD_USUARIO
        });

    }
}
```

Fonte: Elaborado pelos autores (2024)

A ilustração 24 demonstra como é feita a busca de vendas de um padeiro. Este método *Report* retorna uma lista de modelos de relatório “Relatorio*Model*”

relacionados às vendas de um determinado usuário, identificado pelo seu código único “CD_USUARIO”. Ele utiliza um objeto *Helper* para executar uma consulta ao banco de dados, invocando o procedimento armazenado “spRPTVendasPadeiros” na base de dados “db*Name*”. Este procedimento está responsável por recuperar dados específicos relacionados às vendas de padeiros vinculados ao usuário fornecido. Os resultados dessa consulta são então mapeados para objetos “Relatorio*Model*” e retornados como uma lista para serem utilizados na aplicação.

Ilustração 24 - Implementação vendas do padeiro

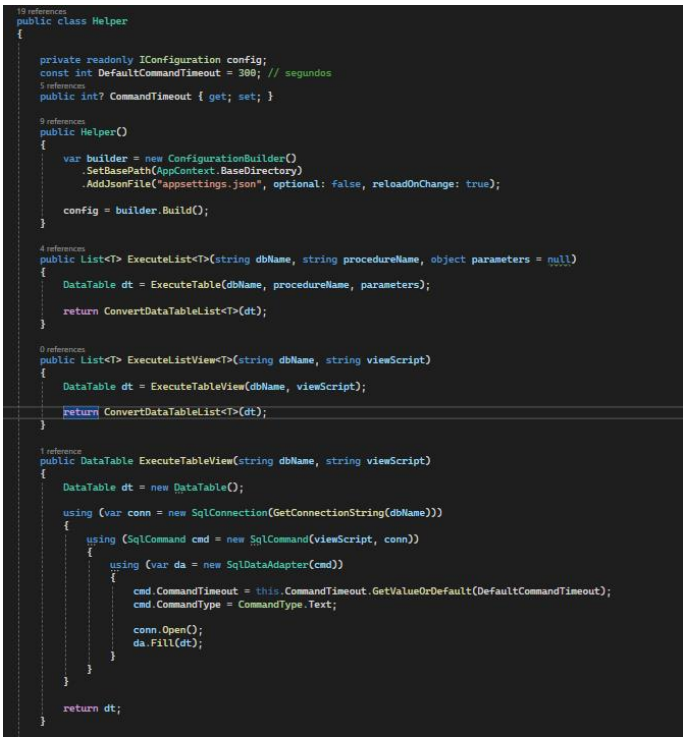
```
1 reference
public List<RelatorioModel> Report(Guid CD_USUARIO)
{
    Helper helper = new Helper();
    return helper.ExecuteList<RelatorioModel>(dbName, "dbo.spRPTVendasPadeiros", new
    {
        CD_USUARIO
    });

}
```

Fonte: Elaborado pelos autores (2024)

A ilustração 25 demonstra como a classe *Helper*, responsável por fornecer métodos para executar consultas em um banco de dados, foi utilizada no projeto. A função “*ExecuteListView*<T>” é usada para executar consultas em uma visualização específica do banco de dados e retornar os resultados como uma lista de objetos do tipo especificado. A função “*ExecuteTableView*” executa uma consulta direta a uma tabela no banco de dados e retorna os resultados como uma *DataTable*. Dentro da função “*ExecuteTableView*”, uma nova conexão é estabelecida com o banco de dados usando “*SqlConnection*”. Em seguida, um comando SQL é criado usando “*SqlCommand*”, onde o texto da consulta é especificado como *viewScript*. Um adaptador de dados “*SqlDataAdapter*” é usado para preencher uma *DataTable* com os resultados da consulta.

Antes de executar a consulta, o tempo limite do comando *CommandTimeout* é configurado com um valor padrão de 300 segundos, a menos que um valor diferente seja fornecido explicitamente. Isso evita que a execução da consulta demore demais e causa um *timeout*. Após a execução da consulta e o preenchimento da *DataTable*, os resultados são retornados para serem processados posteriormente. Este trecho de código demonstra como a classe *Helper* facilita a execução de consultas em tabelas e visualizações do banco de dados, simplificando o processo de acesso aos dados.

Ilustração 25 - Implementação objeto Helper



```
19 references
public class Helper
{

    private readonly IConfiguration config;
    const int DefaultCommandTimeout = 300; // segundos
    5 references
    public int? CommandTimeout { get; set; }

    9 references
    public Helper()
    {
        var builder = new ConfigurationBuilder()
            .SetBasePath(AppContext.BaseDirectory)
            .AddJsonFile("appsettings.json", optional: false, reloadOnChange: true);

        config = builder.Build();
    }

    4 references
    public List<T> ExecuteList<T>(string dbName, string procedureName, object parameters = null)
    {
        DataTable dt = ExecuteTable(dbName, procedureName, parameters);

        return ConvertDataTableList<T>(dt);
    }

    0 references
    public List<T> ExecuteListView<T>(string dbName, string viewScript)
    {
        DataTable dt = ExecuteTableView(dbName, viewScript);

        return ConvertDataTableList<T>(dt);
    }

    1 reference
    public DataTable ExecuteTableView(string dbName, string viewScript)
    {
        DataTable dt = new DataTable();

        using (var conn = new SqlConnection(GetConnectionString(dbName)))
        {
            using (SqlCommand cmd = new SqlCommand(viewScript, conn))
            {
                using (var da = new SqlDataAdapter(cmd))
                {
                    cmd.CommandTimeout = this.CommandTimeout.GetValueOrDefault(DefaultCommandTimeout);
                    cmd.CommandType = CommandType.Text;

                    conn.Open();
                    da.Fill(dt);
                }
            }
        }

        return dt;
    }
```

Fonte: Elaborado pelos autores (2024)

As funções “desabilitaAlimentoRestrito()” e “desabilitaAlimentoCaseiros()” são responsáveis por desabilitar os *checkboxes* relacionados quando outros *checkboxes* são marcados. Por exemplo, “desabilitaAlimentoRestrito()” é chamada quando o *checkbox* relacionado à opção "Caseiros" é marcado, enquanto “desabilitaAlimentoCaseiros()” é chamada quando qualquer outra opção de alimento restrito é marcada. A função “buscarPadeiroIdeal()” é responsável por buscar padeiros ideais com base nas opções selecionadas pelo usuário (alimentos restritos e CEP). Os valores dos *checkboxes* (glúten, lactose, lowCarb e caseiros) são obtidos com base nos valores marcados ou não marcados dos *checkboxes* relacionados. Uma *string* Extensible Markup Language (*XML*) é construída com base nos valores dos *checkboxes*. Cada *checkbox* marcado é representado por um item <ITEM> no *XML*. Os valores dos *checkboxes* são codificados com os códigos correspondentes (por exemplo, "gluten" é representado pelo código "1"). A *URL* da *API* é construída com base no CEP fornecido pelo usuário, na *string XML* construída e em outros parâmetros necessários, como a quantidade de padeiros desejados. A função “*fetch*()” é usada para fazer uma requisição *Hypertext Transfer Protocol (HTTP)* para a *URL* da *API*, passando os parâmetros adequados, como o cabeçalho e o modo de acesso (no caso,

"cors"). Essas são as principais funcionalidades do código. Ele permite que o usuário selecione suas preferências de alimentos restritos e faça uma solicitação para encontrar padeiros ideais com base nessas preferências e no CEP fornecido. Representado abaixo na ilustração 26.

Ilustração 26 - Implementação padeiro ideal

Fonte: Elaborado pelos autores (2024)

O código acima lida com a solicitação de listar padeiros próximos com base no CEP do cliente e em possíveis restrições alimentares. Quando um cliente faz uma solicitação *GET* para a rota /ListarPadeirosProximos, este código é executado. Ele espera três parâmetros: CEP_CLIENTE, que é o CEP do cliente, QT_LINHAS, que define o número de padeiros a serem retornados, e LS_ALIMENTOS_RESTRITOS (opcional), que especifica quaisquer restrições alimentares que o cliente possa ter. Primeiro, ele instancia um objeto *GoogleMaps* para lidar com operações relacionadas ao *Google Maps* e um objeto *RetornoView* para preparar a resposta da *API*. Em seguida, ele tenta obter a localização do cliente com base no CEP fornecido. Se houver algum erro durante essa etapa, ele retorna uma mensagem de erro. Depois de obter a localização do cliente com sucesso, ele utiliza essa informação para buscar uma lista de padeiros na mesma cidade do cliente. Se não encontrar nenhum padeiro, ele retorna uma mensagem informando que nenhum padeiro foi encontrado. Em seguida, ele obtém as localizações de cada padeiro na lista e determina os padeiros mais próximos da localização do cliente. Por fim, ele retorna uma resposta *HTTP* Ok (200) contendo os dados dos padeiros mais próximos encontrados, ou uma

mensagem de erro caso algo tenha falhado durante o processo. É possível visualizar na ilustração 27 abaixo.

Ilustração 27 - Implementação padeiro ideal

```
[HttpGet("ListarPadeirosProximos")]
0 references
public IActionResult ListarPadeirosProximos(string CEP_CLIENTE, int QT_LINHAS, string? LS_ALIMENTOS_RESTRITOS = null)
{
    GoogleMaps maps = new GoogleMaps();
    RetornoView retorno = new RetornoView();

    try
    {
        // Pesquisa Latitude/Longitudo do Cliente pelo CEP
        LocalizacaoView localizacao = GoogleMaps.ZipCodeSearch(CEP_CLIENTE);

        if (!string.IsNullOrEmpty(localizacao.DS_MENSAGEM))
        {
            retorno.Mensagem = localizacao.DS_MENSAGEM;
            return BadRequest(retorno);
        }

        // Lista de padeiros por Cidade
        Padeiros padeiro = new Padeiros();
        List<PadeiroView> lstPadeiros = padeiro.ListarPadeiros(localizacao.NM_CIDADE, LS_ALIMENTOS_RESTRITOS);


        if (lstPadeiros == null || lstPadeiros.Count == 0)
        {
            retorno.Mensagem = "Nenhum padeiro foi encontrado nas proximidades!";
            return BadRequest(retorno);
        }

        // Retornar a localização de cada padeiro próximo do cliente
        padeiro.ListarLocalizacaoPadeiro(localizacao, lstPadeiros);


        // Retornar os 3 padeiros mais próximos da localização do cliente
        retorno.Data = padeiro.ListarMelhorLocalizacao(localizacao, lstPadeiros, QT_LINHAS);

        if (retorno.Data == null)
        {
            retorno.Mensagem = "Nenhum registro encontrado!";
        }

        return Ok(retorno);
    }
    catch (Exception ex)
    {
        retorno.Mensagem = "Erro de Sistema";
        retorno.StackTrace = ex.Message + "/n" + ex.StackTrace;
        return BadRequest(retorno);
    }
}
```

Fonte: Elaborado pelos autores (2024)

2.4 CAMADA DE APRESENTAÇÃO

Nesse tópico são apresentadas as principais telas do aplicativo, criadas para atender os requisitos propostos.

Segundo MAGALHÃES o objetivo da camada de apresentação é encapsular toda a lógica de apresentação usada para servir os clientes.

Ilustração 28 - Tela inicial

Fonte: Elaborado pelos autores (2024)

A ilustração 28 apresenta a tela inicial da aplicação, que contém um menu superior com as páginas disponíveis para navegação, um botão direcionando para a tela de login e um para o carrinho de compras. É possível visualizar um breve texto com informações sobre o site e ícones com legendas que representam as especialidades de pães que o usuário poderá encontrar a venda.

Além disso, contém dois containers que direcionam para as páginas de localizar o padeiro pelo cep e dar match com o padeiro ideal do usuário.

Para finalizar a página, observa-se um carrossel apresentando os produtos do primeiro padeiro cadastrado na aplicação, contendo imagem e valor dos produtos cadastrados, possibilitando através de um botão que o usuário adicione ao carrinho o produto desejado.

A interface busca facilitar o entendimento sobre todo o conteúdo do site na primeira navegação, proporcionando uma experiência intuitiva e positiva.

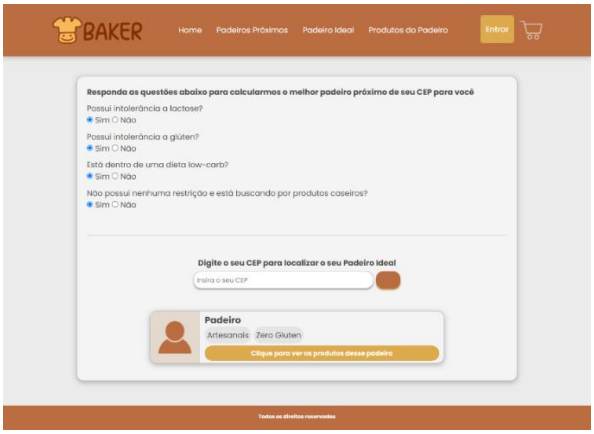
Ilustração 29 - Tela de login

Fonte: Elaborado pelos autores (2024)

A ilustração 29 demonstra a tela de login da aplicação, que contém os campos para inserir o e-mail do usuário e a senha, caso o usuário não tenha conta, é possível iniciar uma nova clicando em cadastrar-se que se encontra abaixo do botão de entrar.

A tela desempenha um papel fundamental para o uso completo do sistema.

Ilustração 30 - Tela padeiro ideal

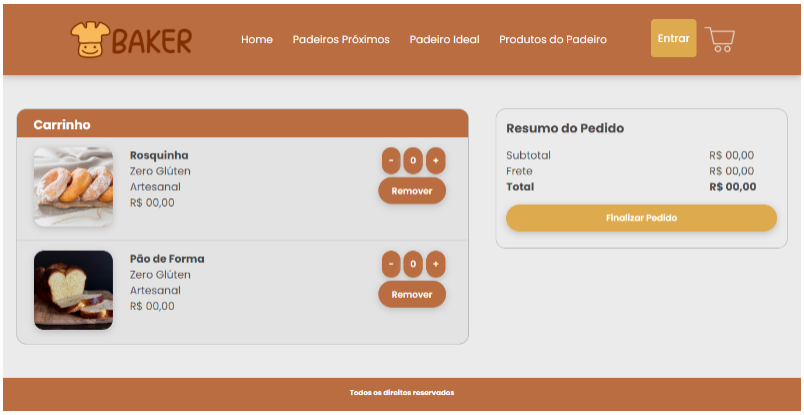


Fonte: Elaborado pelos autores (2024)

Na ilustração 30 apresenta a tela de padeiro ideal, contendo os campos onde o usuário poderá escolher as especialidades que deseja que os produtos tenham, como se possuí lactose, glúten e farinha. Após utilizar os filtros, ele terá que digitar o CEP e clicar em enviar, obtendo como resultado o padeiro mais próximo do CEP informado que possuí as especialidades que ele filtrou inicialmente, sendo utilizado como critério de escolha da plataforma o valor mais baixo em critério de desempate.

A tela tem o objetivo de realizar de forma dinâmica a busca do padeiro mais compatível com o usuário, de maneira clara e objetiva, facilitando para todos os usuários que poderiam encontrar dificuldades em realizar os filtros.

Ilustração 31 - Tela carrinho



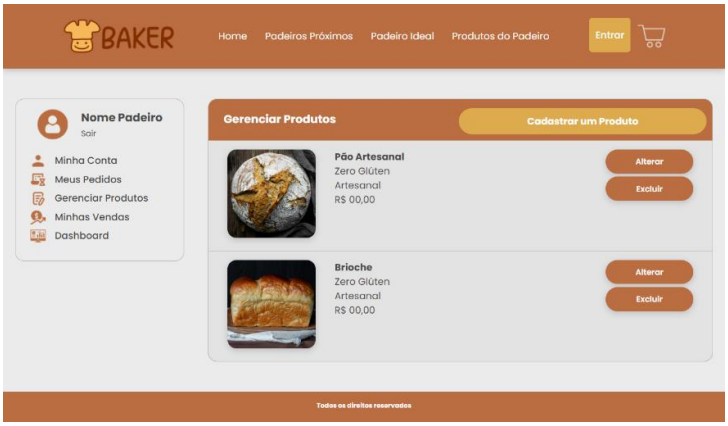
Fonte: Elaborado pelos autores (2024)

A ilustração 31 apresenta a tela de carrinho, contendo os produtos escolhidos anteriormente pelo cliente que podem ser modificados no próprio carrinho aumentando a quantidade, diminuindo ou até mesmo, excluindo. Podemos visualizar também o subtotal dos produtos, frete e total da compra.

O botão “Finalizar Pedido” levará o usuário para a tela de conclusão onde poderá enviar o pedido por meio do e-mail para o padeiro automaticamente através do sistema.

O objetivo da tela é proporcionar uma experiência dinâmica para o usuário, facilitando a alteração dos itens do pedido antes do fechamento da compra, o que proporciona uma experiência positiva ao usuário.

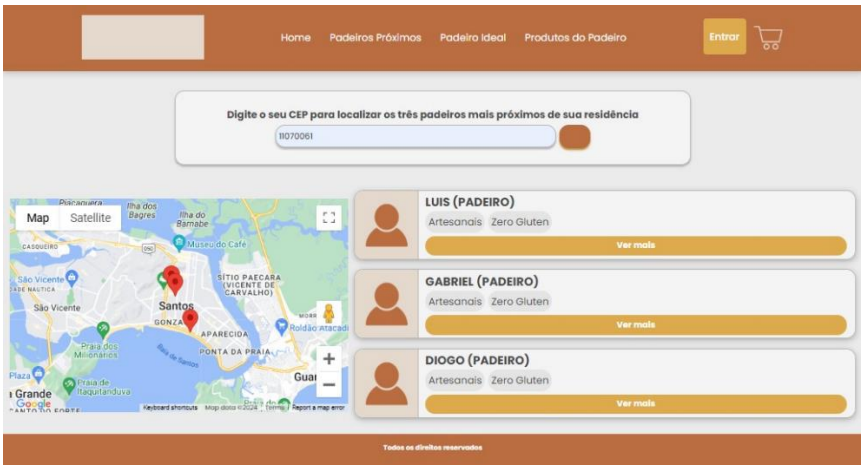
Ilustração 32 - Tela gerenciar produtos



Fonte: Elaborado pelos autores (2024)

A ilustração 32 representa a tela de gerenciamento de produtos do padeiro, sendo possível cadastrar um novo produto que contém os campos nome do produto, preço do produto, imagem e escolhas de especificação do produto por meio de *checkbox*. Um produto já cadastrado poderá ser alterado ou excluído, possibilitando uma experiência completa para o padeiro.

Ilustração 33 - Tela gerenciar produtos



Fonte: Elaborado pelos autores (2024)

A ilustração 33 representa a tela de padeiros mais próximos por CEP, sendo possível digitar um CEP no campo indicado e ter como resultado os três padeiros mais próximos, sendo possível visualizar suas localizações no mapa e os seus nomes, especialidades e um botão “Ver mais” para entrar no perfil de produtos de cada padeiro resultante. Dessa forma, o usuário consegue ter uma experiência completa de busca por localização.

3 RESULTADO

Foram realizados testes com oito usuários nas duas últimas semanas de abril, sendo divididos em dois grupos, quatro participantes realizaram os testes como clientes e quatro participantes realizaram os testes como padeiros. Feitos na máquina em que foi desenvolvido o projeto, em um ambiente silencioso para concentração dos participantes. Após finalizarem os testes, foram direcionados para um questionário *Google Forms* para avaliarem a experiência.

Tarefas passadas para os clientes:

- Realizar cadastro no sistema como cliente;
- Realizar *login* no sistema;
- Editar *e-mail*;
- Responder o questionário para encontrar o padeiro ideal;
- Filtrar padeiro mais próximo por CEP;
- Adicionar produtos de um padeiro ao carinho.
- Finalizar pedido enviando por *e-mail*.

Tarefas passadas para os padeiros:

- Realizar cadastro no sistema como padeiro;
- Realizar *login* no sistema;
- Editar *e-mail*;
- Adicionar um produto em seu perfil;
- Editar o valor do produto adicionado anteriormente;
- Visualizar histórico de pedidos recebidos;
- Visualizar *dashboard* de vendas.

3.1 ANÁLISE DOS RESULTADOS DOS TESTES

Nesse tópico será descrito o perfil dos participantes e comentários sobre a participação deles.

Os oito participantes são moradores da baixada santista, das cidades de Santos e São Vicente, com idades entre 20 e 57 anos. Todos obtiveram excito em realizar todas as tarefas propostas, com uma média de 9 minutos para realização completa.

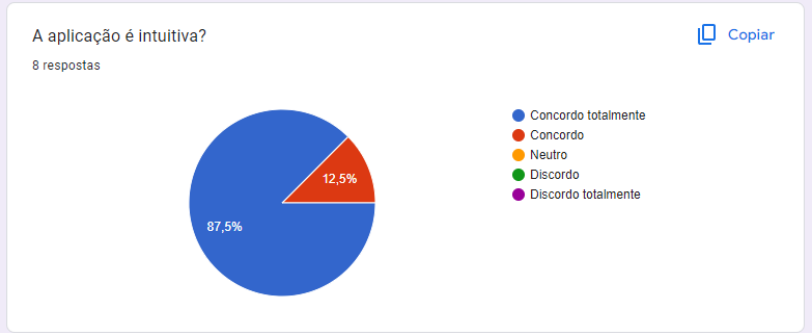
Ao fim da realização das tarefas, foram recebidos feedbacks dos usuários, a maioria relatou que não encontrou nenhuma dificuldade, porém, dois participantes relataram problemas para encontrar a tela da edição de informações, sugerindo deixá-la mais visível. A experiência foi avaliada como positiva por todos que participaram, relatando que recomendariam o aplicativo para familiares e amigos, por ser intuitivo e inovador.

3.2 ANÁLISE DO QUESTIONÁRIO

Todos os participantes responderam ao questionário após o fim da experiência, sendo realizado pelo *Google Forms*, segundo ANDRES, por ser uma ferramenta online com diversas funcionalidades, se for executada de maneira correta trará resultados positivos, como a agilidade, praticidade e sustentabilidade.

O questionário contém 4 perguntas alternativas e 1 dissertativa para avaliação da experiência durante o uso da plataforma *web*.

Ilustração 34 - Gráfico da primeira pergunta



Fonte: Elaborado pelos autores (2024)

A ilustração 34 representa o gráfico de resultado da primeira pergunta, revelando que 87,5% dos participantes concordam totalmente e 12,5% concordam que é intuitiva.

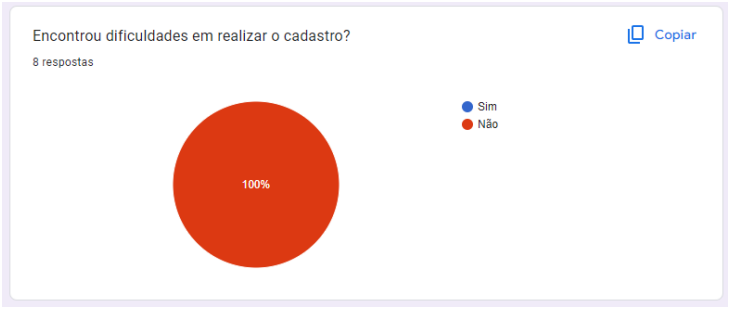
A análise revela que a aplicação proporciona uma experiência positiva para os usuários, confirmando a pergunta.

Ilustração 35 - Gráfico da segunda pergunta

Fonte: Elaborado pelos autores (2024)

A ilustração 35 representa o gráfico de resultados da segunda pergunta, revelando 75% dos participantes concordam totalmente e 25% concordam sobre a facilidade de navegação no site, sendo um resultado positivo.

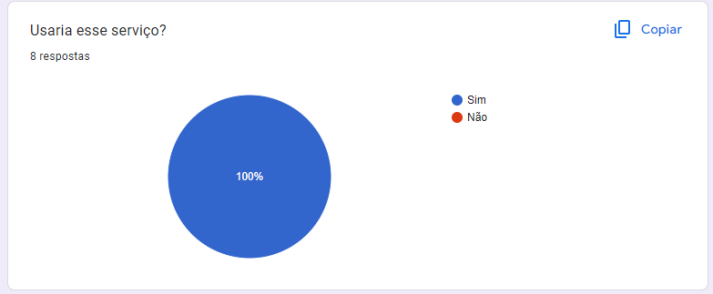
Ilustração 36 - Gráfico da terceira pergunta



Fonte: Elaborado pelos autores (2024)

A ilustração 36 representa o gráfico de resultados da terceira pergunta, revelando que todos os participantes não encontraram dificuldades para a realização do cadastro no site.

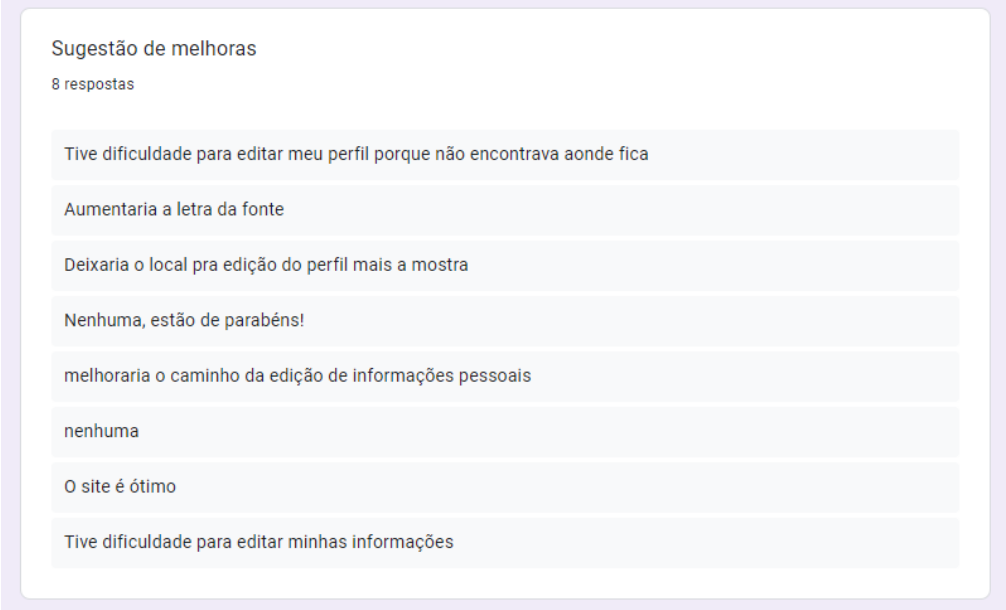
Ilustração 37 - Gráfico da quarta pergunta



Fonte: Elaborado pelos autores (2024)

A ilustração 37 representa o gráfico de resultados da quarta pergunta, revelando que todos os participantes usariam esse serviço, mostrando com excito a satisfação dos participantes.

Ilustração 38 - Gráfico da quinta pergunta



Fonte: Elaborado pelos autores (2024)

Na ilustração 38, foram validadas as melhorias sugeridas pelos usuários, sendo a mais predominante a opção de avaliação dos padeiros para facilitar a compra por novos usuários.

3.3 CONCLUSÃO

Como resultado desse projeto, foi possível verificar o crescimento dos padeiros microempreendedores nos últimos anos. Segundo ROVAROTO (2023), cerca de 70%

desse crescimento vem através da classificação microempreendedor individual, que são pessoas que trabalham como pequenos empresários no Brasil.

Desse modo, surgiu a ideia de desenvolver uma plataforma para divulgar esse setor que possui crescimento contínuo ao longo dos anos. Segundo BINI e CARAVAGGI (2022), o Sindipan-MT informou que 76% dos brasileiros consomem pão no café da manhã, além de em 2021, o mercado de panificação e confeitaria ter faturado R$ 105,85 bilhões no país. Todos esses dados fizeram com que criasse um entusiasmo para facilitar a comunicação entre cliente e padeiro, o que deu surgimento ao *Baker*.

O site *Baker* foi pensado para atender tanto pessoas que buscam por uma alimentação mais saudável no seu dia a dia, quanto para realizar a divulgação desses pequenos empresários padeiros. Dando a possibilidade de localizar o padeiro mais próximo do endereço de sua residência e disponibilizar o seu cardápio para solicitar um pão caseiro.

Toda a estrutura do projeto foi realizada com *HTML*, *CSS*, *JavaScript* e o *framework .NET* para elaboração de suas funcionalidades. Com o resultado desse protótipo, foram realizados testes entre os clientes finais e os padeiros para validar a usabilidade do sistema, e se de fato ele pode alcançar o objetivo de divulgação dos produtos do padeiro.

Em decorrência a esses testes, após analisar um público-alvo de pessoas entre 20 e 57 anos nas cidades de Santos e São Vicente, de maneira geral, foram estabelecidos comentários positivos referente a navegabilidade do *site*. Um dos pontos levantados foi a respeito da localização das configurações do usuário, em que o caminho não estava claro. Com relação a essa classificação, será realizada uma análise referente ao comentário e posteriormente uma adequação sobre esse acesso.

Especialistas docentes na área de tecnologia da Faculdade de Tecnologia da Baixada Santista Rubens Lara foram consultados para realizarem uma análise no sistema. Embora tenham sido levantados diversos fatores, alguns não puderam ser aplicados devido às restrições de tempo no desenvolvimento do *site Baker*.

Um dos elementos propostos foi a inclusão de uma área de comentários nos pedidos e a capacidade de os clientes classificarem os padeiros. Essa adição tinha o potencial de facilitar a busca dos consumidores por novos empreendedores e produtos, promovendo uma maior interação e *feedback* na plataforma.

Outro ponto discutido foi a implementação de um sistema de validação para a busca de padeiros próximos. Além de apresentar os três primeiros resultados, em casos em que houvesse mais opções, era necessário estabelecer critérios para desempatar os resultados. Isso visava aprimorar a experiência do usuário, garantindo que as recomendações fossem relevantes e úteis.

Um terceiro elemento, identificado como uma possível melhoria futura, era a criação de um usuário administrador *(*ADM*).* Esse usuário teria a responsabilidade de validar atividades inadequadas tanto de clientes quanto de padeiros, além de monitorar o uso eficiente das contas na plataforma. Essa função de supervisão poderia contribuir significativamente para a segurança e integridade do sistema, garantindo uma comunidade online saudável e confiável.

Um quarto elemento, se trata da melhoria do dashboard utilizando o *Power BI* da *Microsoft* para uma visualização robusta, contendo elementos visuais para melhoria da função, tornando-a mais dinâmica.

Embora essas sugestões tenham sido consideradas valiosas, o tempo necessário para desenvolver e implementar cada uma delas foi um fator limitante. Portanto, essas ideias permanecem como possíveis melhorias a serem exploradas em futuras iterações do sistema.

Site Baker possui diversos benefícios, além da divulgação do padeiro, de fato ele garante e facilita uma alimentação saudável na vida dos clientes finais, em que em tempos atuais, se torna uma tarefa difícil devido à alta demanda na vida das pessoas. Sem contar com o fato do crescimento por busca de alimentos caseiros e valorização desses pequenos empreendedores que atuam na execução desse ramo.

A partir disso, por conseguinte, fica evidente que, melhoras precisam ser implementadas no *site Baker* para que ele possa atingir um alto potencial. Entretando, os resultados da primeira versão apresentaram bons resultados e fizeram com que pudesse ser visualizado êxito nos objetivos propostos para o projeto.

REFERÊNCIAS

ANDRES, F. C. et al **A utilização da plataforma Google forms em pesquisa acadêmica: relato de experiência**. Research, Society and Development, v. 9, n. 9, 1–7, 2020.

BINI T.; CARAVAGGI D. (ED.). **Quem quer pão? Setor de panificação movimenta R$ 105 bilhões por ano no Brasil**. Disponível em: <https://www.cnnbrasil.com.br/viagemegastronomia/gastronomia/quem-quer-pao-setor-de-panificacao-movimenta-r-105-bilhoes-por-ano-no-brasil/>. Acesso em: 02 de mar. 2024.

CUNHA H.; RIBEIRO, D.; **Engenharia de Requisitos de Sistemas**, editora Horacio Ribeiro, Brasil, 2018.

DARSHAN P. **Dot Net Programming using C#(as per NEP Syllabus)**, Editora Blue Rose Publishers, 2023.

FLATSCHART, F.; **HTML 5 - Embarque Imediato**; 1ª. Edição; Editora Brasport, Rio de Janeiro, 2011.

FONTES, A.; PERO V.; **“Desempenho dos microempreendedores no Brasil.”** EconomiaA v. 12, n. 3, p. 635-665, 2011.

GUEDES G.; **UML 2 - Guia Prático**; 2ª Edição; Editora Novatec, Brasil, 2014.

GUEDES, G. T. A. **UML 2 - Uma Abordagem Prática - 3ª Edição**; Editora Novatec, Brasil, 2018.

JOBSTRAIBIZER F. **Guia Profissional Microsoft SQL Server 2008**, Editora Universo dos Livros, Brasil, 2008.

LIBA. **Um guia para o Corpo de Conhecimento de Análise de Negócios(TM) (Guia Babok®)**, Editora International Institute of Business Analysis, 2011.

MACHADO, F. N. R. **Análise e Gestão de Requisitos de Software – Onde nascem os sistemas**; 3ª Edição; Editora Saraiva Educação S.A, Brasil, 2018.

MACORATTI, J. **UML - Casos de Uso - Conceitos (revisão)**. Disponível em: <https://www.macoratti.net/11/10/uml_rev1.htm>. Acesso em: 31 mar.

MAGALHÃES, R. **Modelo de arquitetura de aplicações em camadas baseado em padrões de projeto**.

O que é MEI - Microempreendedor Individual? Quem pode ser MEI?. Disponível em <https://www.gov.br/empresas-e-negocios/pt-br/empreendedor/perguntas-frequentes/o-que-e-o-microempreendedor-individual-mei/o-que-e-mei>. Acesso em: 02 de mar. 2024.

Pão quentinho: negócios de panificação registram alta no número de empreendedores formais. Disponível em <https://agenciasebrae.com.br/cultura-empreendedora/pao-quentinho-negocios-de-panificacao-registram-alta-no-numero-de-empreendedores-formais/>. Acesso em: 20 de mar. 2024.

PRESCOTT. P. **Programando em JavaScript,** Editora Babelcube Incorporated., 2016.

ROVAROTO. I. (ED.). **154 novas padarias são abertas por dia no Brasil — e mercado é dominado por MEIs.** Disponível em: < https://exame.com/negocios/154-novas-padarias-sao-abertas-por-dia-no-brasil-e-mercado-e-dominado-por-meis/>. Acesso em: 02 de mar. 2024.

APÊNDICE A

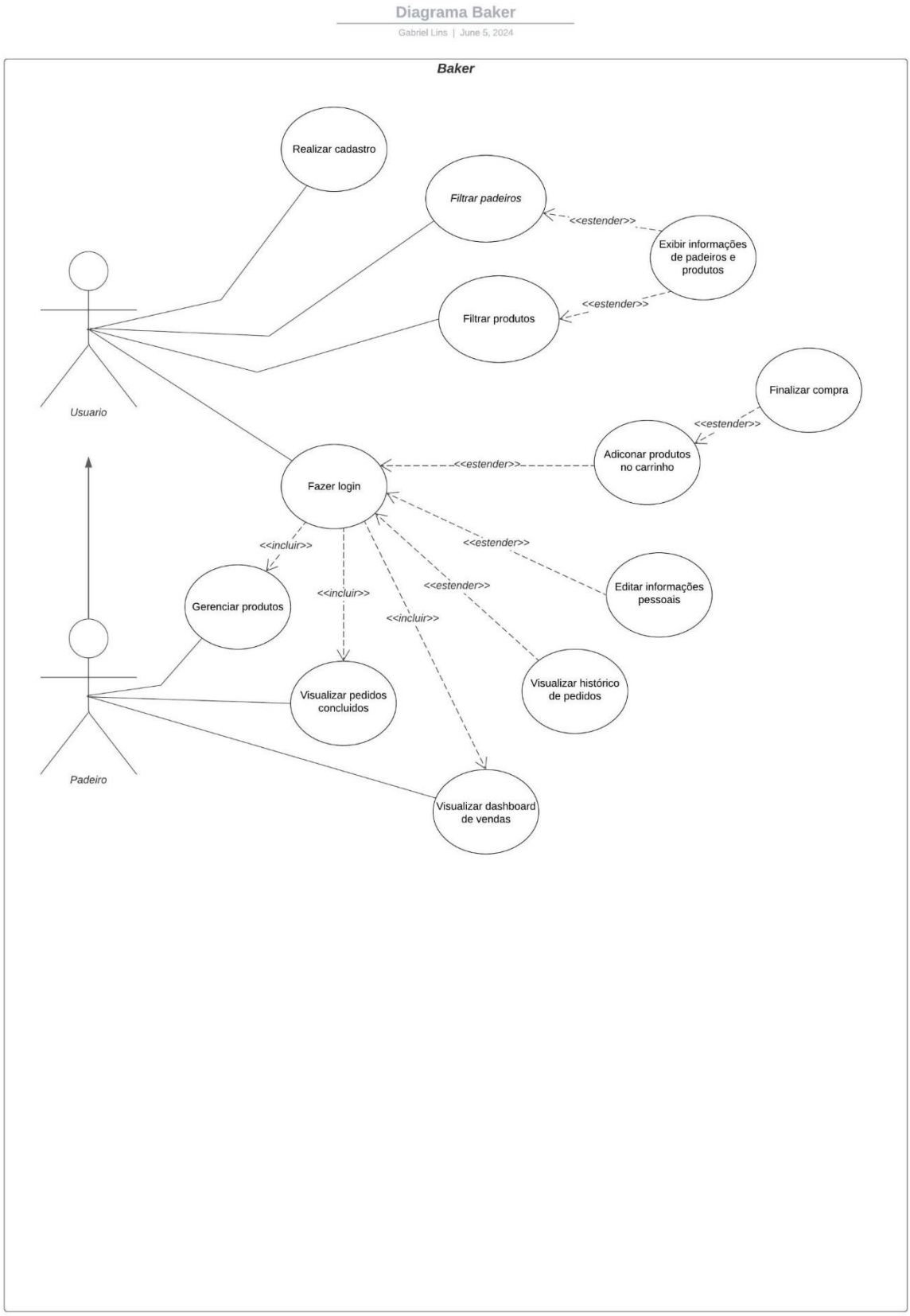
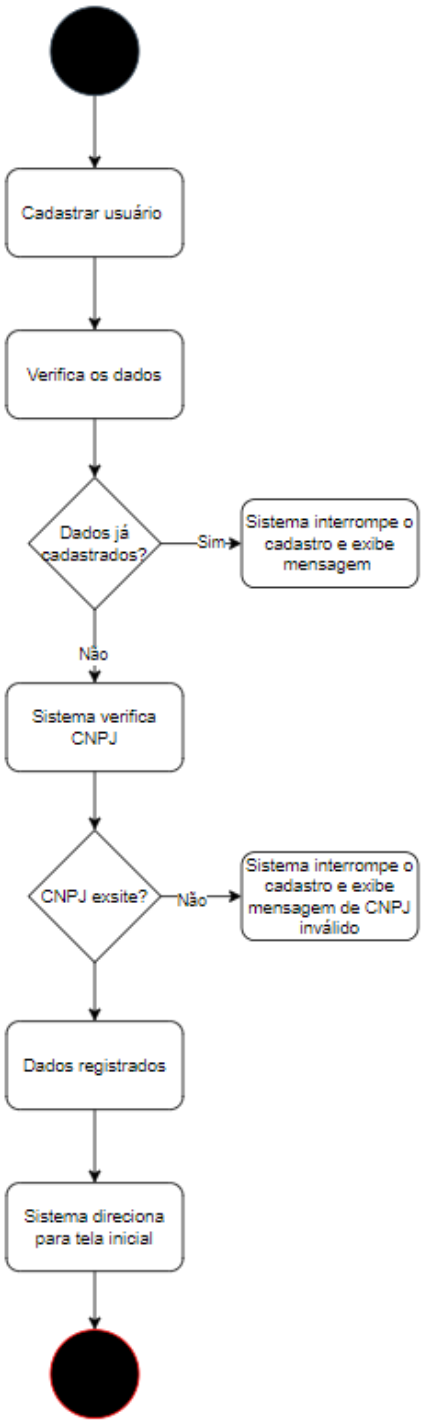


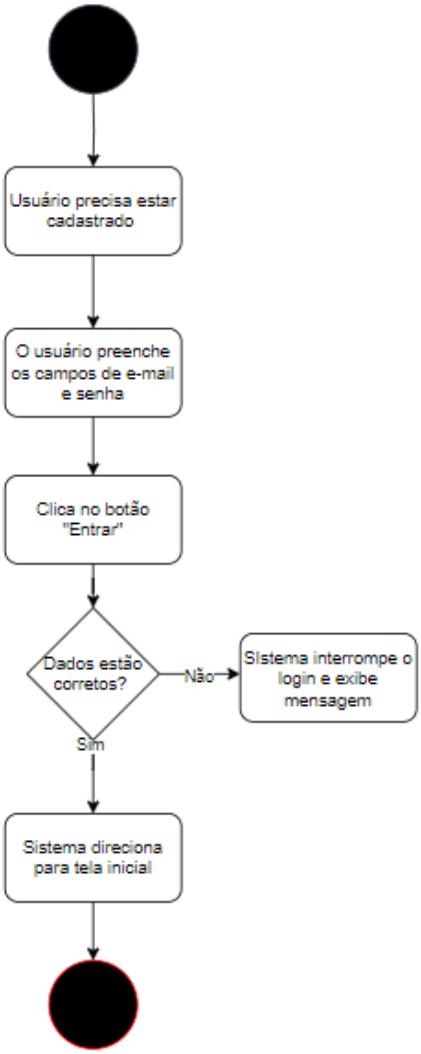
Diagrama Baker
Gabriel Lins | June 5, 2024
Baker
Realizar cadastro
Filtrar padeiros
<<estender>>
Exibir informações de padeiros e produtos
Filtrar produtos
<<estender>>
Usuario
Finalizar compra
<<estender>>
Adiconar produtos no carrinho
<<estender>>
Fazer login
<<incluir>>
<<estender>>
Editar informações pessoais
Gerenciar produtos
<<incluir>>
<<estender>>
<<incluir>>
Visualizar histórico de pedidos
Visualizar pedidos concluidos
Padeiro
Visualizar dashboard de vendas

APÊNDICE B

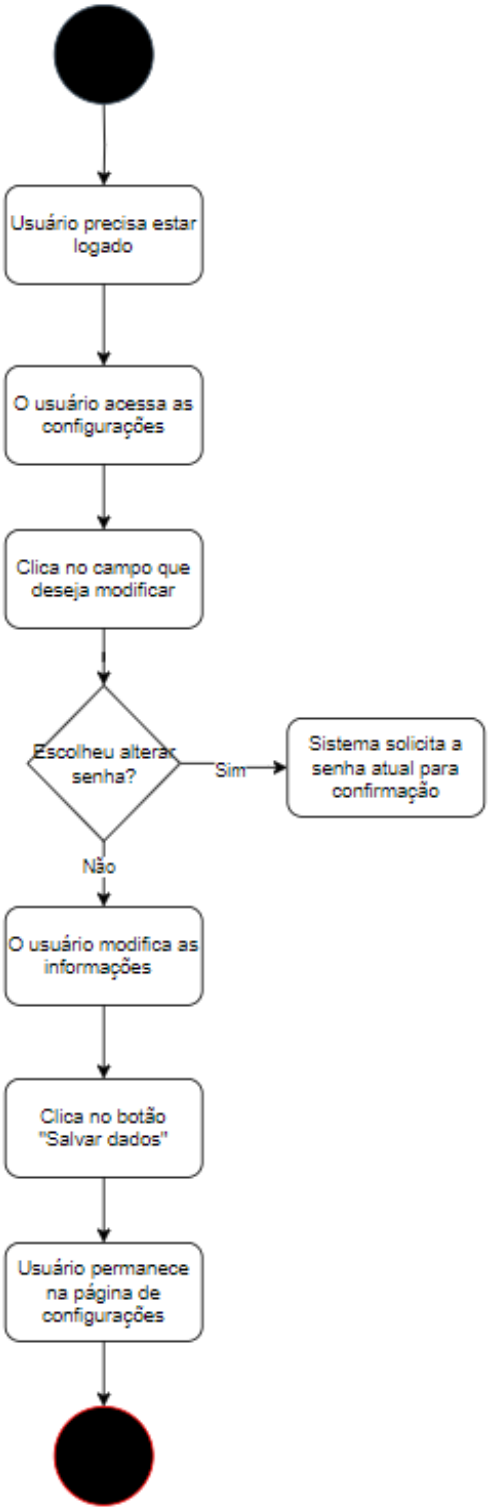
Cadastro Usuário



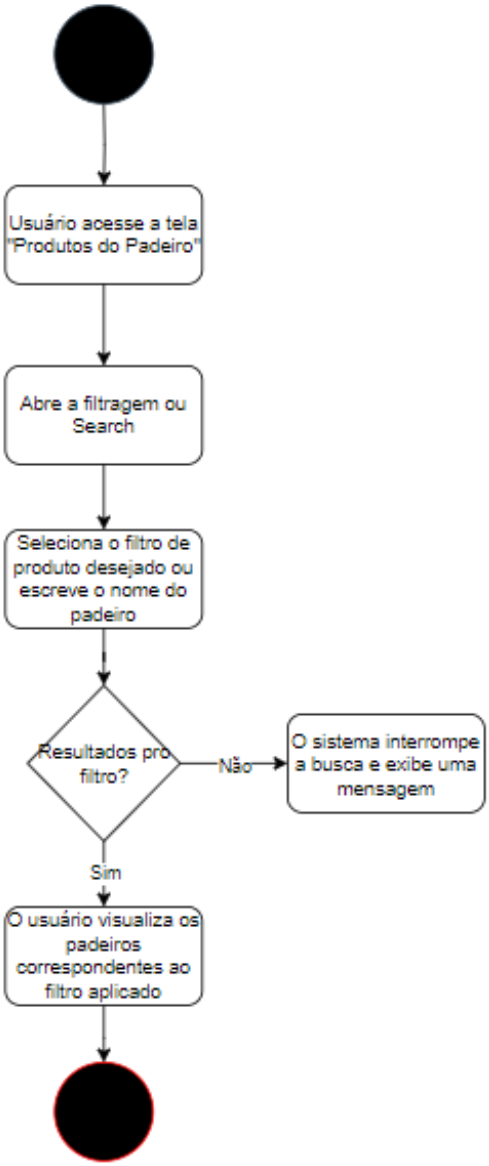
Login Usuário



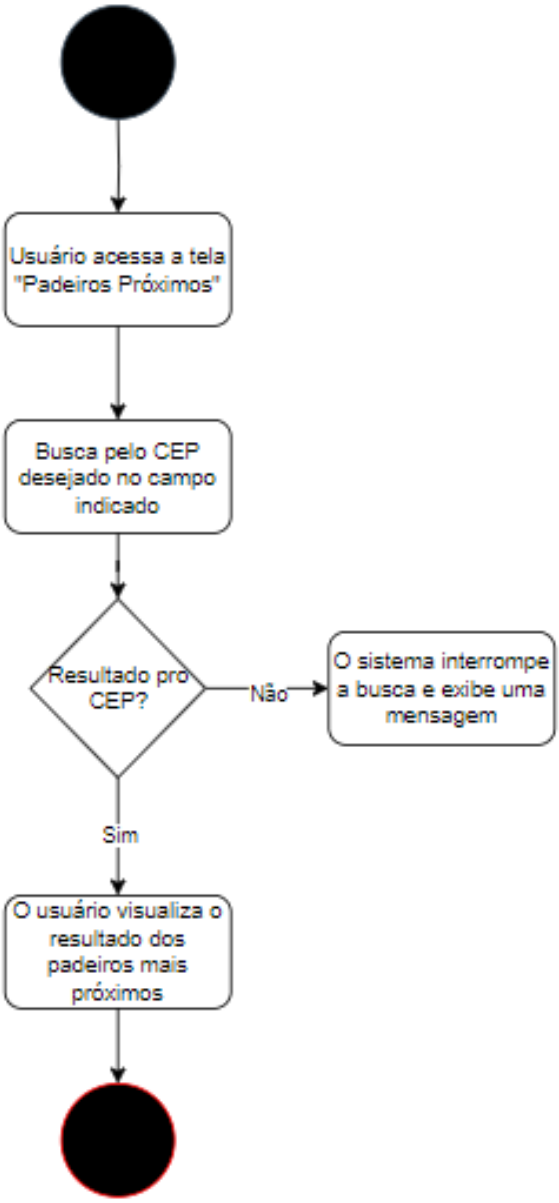
Editar Informações



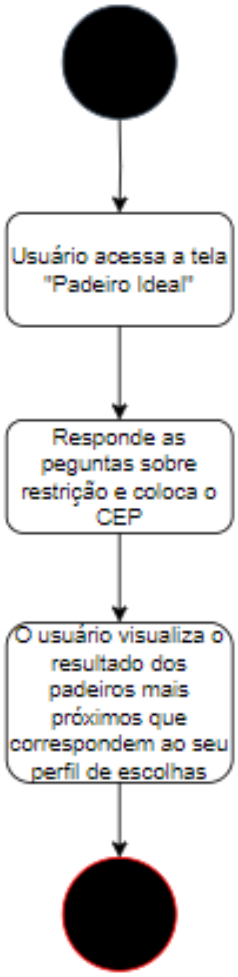
Filtrar produto/padeiro



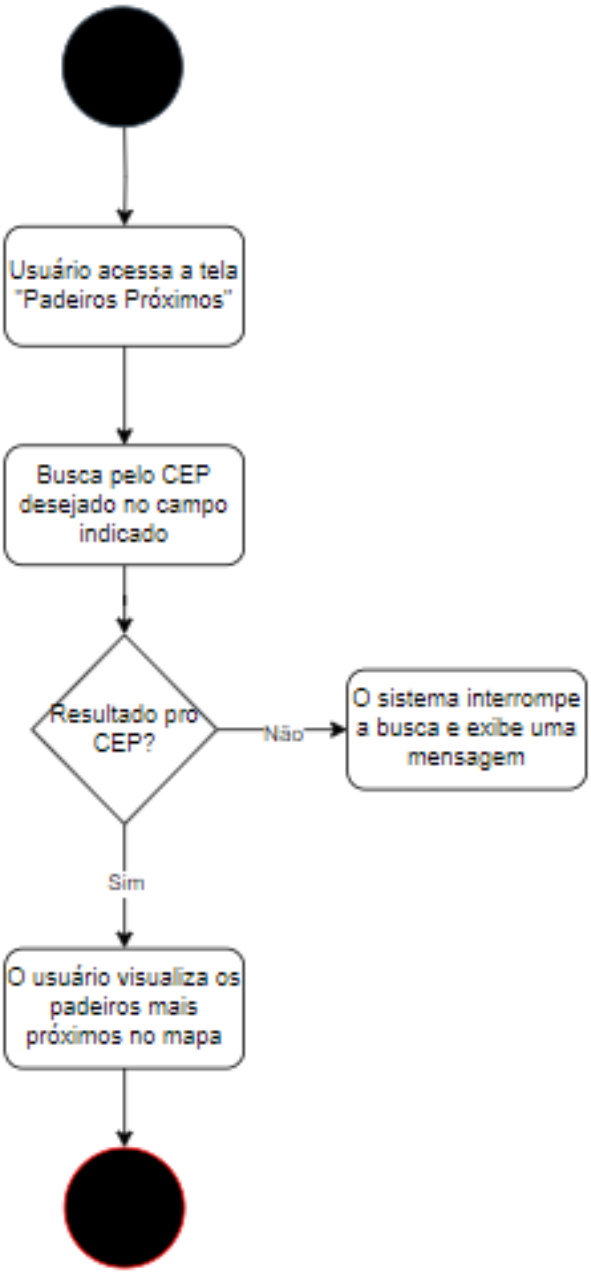
Filtrar por CEP



Filtrar por restrição

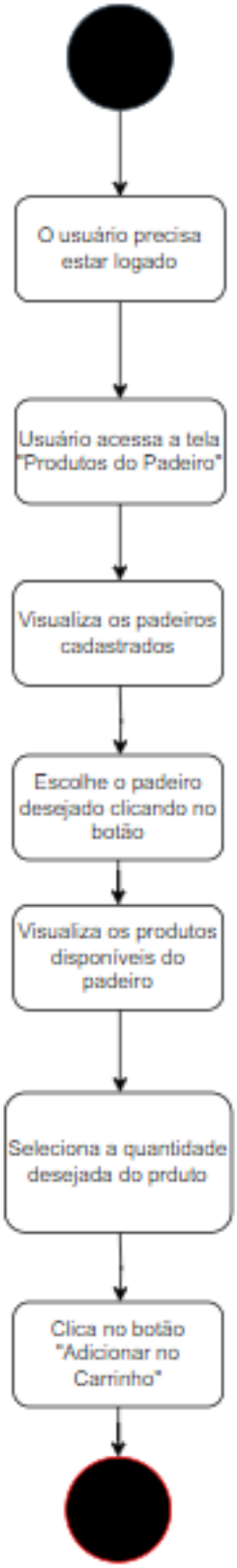


Mapa de padeiros próximos

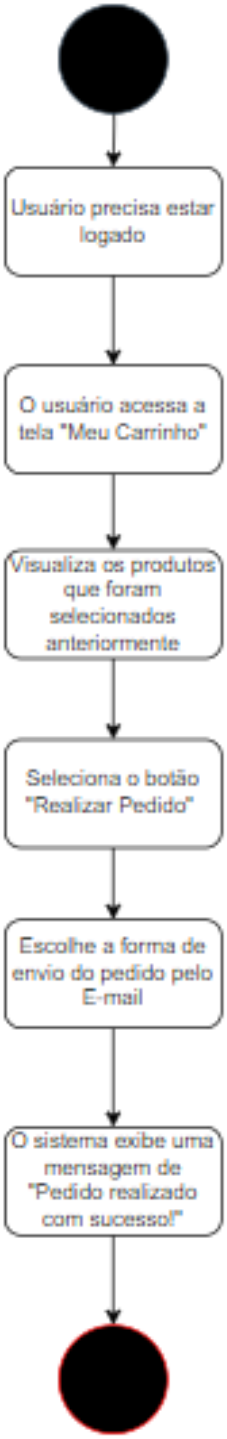


Histórico de pedidos

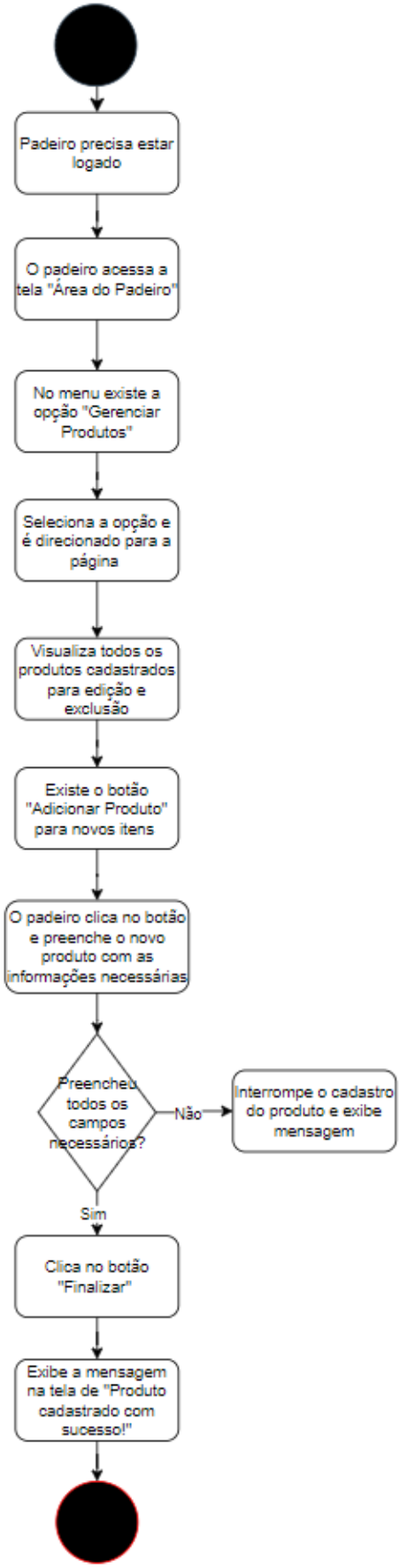
Adicionar no carrinho



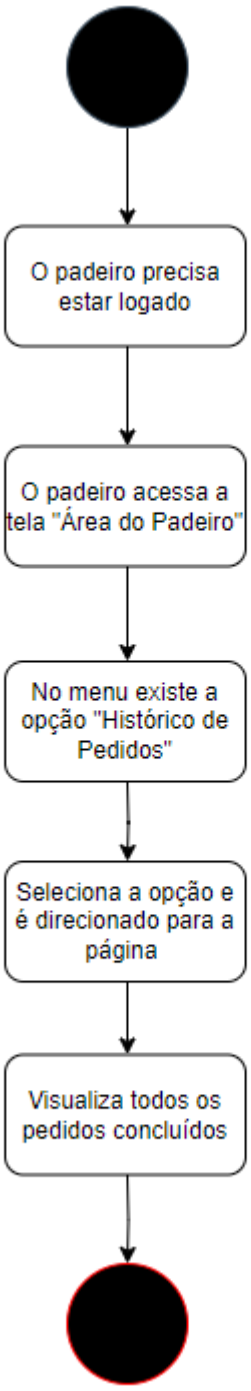
Finalizar pedido



Gerenciar produtos



Histórico de pedidos concluídos



Dashboard de vendas