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

VINICIUS CASTRO DA COSTA

SISTEMA DE DELIVERY DE FARMÁCIAS (PHARMAPP)

RIO DO SUL
2021

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

VINICIUS CASTRO DA COSTA

SISTEMA DE DELIVERY DE FARMÁCIAS (PHARMAPP)

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: Dr. Marco Aurélio Butzke

RIO DO SUL
2021

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

VINICIUS CASTRO DA COSTA

SISTEMA DE DELIVERY DE FARMÁCIAS (PHARMAPP)

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: Dr. Marco Aurélio Butzke

Banca Examinadora:

_____

Prof.

_____

Prof.

Rio do Sul, xx de mês de 2021.

Tente uma, duas, três vezes e se possível tente a quarta, a quinta e quantas vezes for necessário. Só não desista nas primeiras tentativas, a persistência é amiga da conquista. Se você quer chegar aonde a maioria não chega, faça o que a maioria não faz. (Bill Gates).

Dedico esse trabalho a todos os meus familiares e também a todos que me apoiaram durante esta jornada acadêmica.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por tudo que Ele tem feito em minha vida ao longo dos últimos anos.

Agradeço a meus pais, Gil Elso Castro da Costa e Nádia Gonçalves Castro da Costa, por toda ajuda e o suporte que me deram, me apoiando em todas as minhas escolhas que eu fiz durante está jornada e fazendo tudo que fosse possível para que eu consiga finalizar este ciclo.

Em seguida gostaria de agradecer a todos os professores do curso de Sistema de Informação, pelo conhecimento que foi compartilhado e as oportunidades ao longo dessa caminhada de quatro anos. Em especial gostaria de agradecer ao Dr. Marco Aurélio Butzke, por todo o suporte fornecido desde quando este projeto era meramente uma ideia até o desenvolvimento deste trabalho.

Por fim gostaria de agradecer todos os meus colegas que fizeram parte desta jornada ao meu lado, por todas as memorias e todas as experiências que eles me proporcionaram dentro e fora da vida acadêmica.

RESUMO

O setor farmacêutico é um dos setores mais tradicionais e chegando a movimentar mais de 120 bilhões de reais nos últimos anos, segundo dados do IQVIA. Por ser um setor muito antigo e cada ano que passa a necessidade dos negócios se tornarem digitais e com a pandemia de 2020 mostrou a importância deste nicho de mercado para a sociedade, mas também mostrou a falta da possibilidade de realizar compras através da internet, com a necessidade presencial do cliente no local para realizar a compra de seus produtos. Por outro lado, podemos ver a popularização de aplicativos de *delivery* de comida como o *iFood* ou *Delivery much.* Para isso, ao longo deste trabalho foi desenvolvido um *delivery* focado no setor farmacêutico, possibilitando trazer mais organização, facilidade e também mais tecnologia para este mercado tão importante para a sociedade. A aplicação foi desenvolvida na linguagem de programação Javascript e também foi dividida em duas partes: a API feita em NodeJS e o *front-end* utilizando ReactJS que atualmente é uma das ferramentas mais utilizadas do mercado de tecnologia. Como resultado final, foi possível analisar que a plataforma desenvolvida pode contribuir para o aumento de canais de vendas para as farmácias e facilidade para realizar pedidos por parte do cliente final.

Palavras-Chave: setor farmacêutico, delivery, javascript.

ABSTRACT

Farmaceutic sector is one one the most traditional sectors and it moves more than 120 million reais (Brazilian currency) on the past years, according to IQVIA. It's such an old sector and year by year there's need of making digital business, and with the pandemic on 2020 the importance of this market was shown. Also, has showed the lack of possibility on doing online shopping by internet, considering the need of attending on the place to buy the products wanted. By other side, we can note the popularization of food delivery apps, like IFood or Delivery much. For that, in the present study, it was developed a delivery focused on pharmaceutic sector, bringing the possibility of having more organization, facility and even more technology for this market that is so important for the society. This app was developed on programing language Javascript and it was divided in two parts: API done in NodeJS and the front-end using ReactJS that it is now one of the most used technologies on the market. As final result, it was possible to analyse that the platform developed can contribute to increase the selling channels to the pharmacies and make easier to making orders as a final customer.

Keywords: Pharmaceutic sector, delivery, javascript.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ES6	ECMAScript 6
HTTP	*Hypertext Transfer Protocol*
API	*Application Programming Interface*
REST	*Representational State Transfer*
IDE	*Integrated Development Environment*
CEP	Código de Endereçamento Postal
SQL	*Structured Query Language*
CPF	Cadastro de Pessoas Físicas
CNPJ	Cadastro Nacional da Pessoa Jurídica

SUMÁRIO

1. INTRODUÇÃO

O setor farmacêutico tem sido considerado um nicho muito tradicional e importante para a sociedade, por meio desse setor viabiliza o tratamento e a cura de diversas doenças, por meio da diversa variedade de remédios ali localizado, por conta disso existe uma grande oportunidade neste setor devido à pouca tecnologia empregada, mas sendo um dos setores que mais movimenta dinheiro no brasil.

Devido à pandemia do COVID-19 iniciada final de 2019 surgiu um grande desespero para a população, ainda mais com o *lockdown* o acesso aos medicamentos e produtos farmacêuticos foi uma dificuldade encontrada, mesmo as farmácias sendo consideradas serviços essenciais, a aglomeração preocupava os clientes, algumas farmácias realizaram o *delivery* de seus produtos recebendo os pedidos via Whatsapp, devido à falta de um sistema específico que suprisse essa necessidade.

Segundo Reis (2020), ocorreu um crescimento de 8,79% no mercado farmacêutico entre o mês de janeiro a setembro de 2020, se comparado com 2019. Chegando a ser comercializados 3.529 bilhões de unidades. Uma das grandes preocupações do mercado era a crise gerada pelo covid-19, mas com base nos dados divulgados o mercado farmacêutico se demonstrou muito resiliente, devido ao próprio motivo de a crise ser relacionada à área da saúde. (MEDICINAS/A, 2020)

Tendo em vista os dados apresentados, podemos analisar que o mercado farmacêutico está em crescimento e por causa de ser muito tradicional e resiliente, ele necessita de novas tecnologias para cada vez tornar o seu processo mais ágil, seguro e prático.

Conforme uma pesquisa da Mobills (2021), startup de gestão de finanças pessoais, constatou que os gastos com os principais aplicativos focados na entrega de comida (Rappi, iFood e Uber Eats) cresceram 149% em 2020.

A proposta do projeto é o desenvolvimento de uma plataforma que possibilita às farmácias disponibilizarem seus produtos para comercialização pela internet, muito semelhante com os aplicativos de entrega de comidas, assim tornando possível seus clientes realizarem seus pedidos online, recebendo seus produtos no aconchego de seus lares. A plataforma terá um *layout* muito simples, assim sendo muito fácil para os clientes mais leigos conseguirem realizar seus pedidos, o objetivo do projeto é conseguir ajudar as farmácias a criarem mais um canal de vendas, tornando acessível os medicamentos às pessoas.

1.1 PROBLEMA DE PESQUISA

Como realizar pedidos para uma farmácia de uma maneira digital, organizada e simples?

1.2 OBJETIVOS

A seguir serão apresentados o objetivo geral e objetivos específicos, por meio de tópicos.

1.2.1 Geral

- Auxiliar no processo de vendas e entrega de produtos farmacêuticos de forma online das farmácias, possibilitando mais um canal de vendas.

1.2.2 Específicos

- Levantar os requisitos, direcionando a construção do sistema em etapas;
- Criar protótipos das telas da plataforma, facilitando o entendimento do escopo do projeto;
- Estruturar a modelagem do banco de dados, entendendo o modelo de negócio do sistema;
- Desenvolver aplicação web, para solucionar o problema da pesquisa;

1.3 JUSTIFICATIVA

Recentemente com a pandemia de 2020 causada pelo COVID 19 e por meio do *lockdown* a indústria farmacêutica teve grande dificuldade inicial para conseguir atender seus clientes e vender seus produtos. O risco de ir à farmácia e ter contato com pessoas com o possível vírus assolou o grupo de risco, que se sentiu prejudicado devido a não conseguir acessos aos produtos.

Pensando nisso surgiu a ideia de desenvolver uma plataforma que permita que os usuários consigam realizar pedidos de produtos farmacêuticos e receber no conforto de sua casa, não necessitando ir fisicamente ao estabelecimento, assim tornando mais fácil a compra destes produtos que não necessitam de prescrição médica.

Com base nisso as vantagens para as farmácias se dão a possibilidade do aumento da base de clientes, facilidade de atender várias pessoas simultaneamente. Para os clientes as vantagens seria pedir produtos de qualquer que tenha internet, tornar a compra destes produtos uma forma fácil e rápida, e também possibilitar a compra para pessoas que estão com alguma doença ou são do grupo de risco e não podem sair de casa.

Para construção do sistema optou-se por um sistema web pois torna que os celulares e também os computadores possam utilizar este sistema por meio dos navegadores de sua preferência, evitando os custos necessários gerados para se postar um aplicativo nas principais lojas de aplicativos dos celulares atuais.

2. REFERENCIAL TEÓRICO

Este capítulo apresenta uma revisão dos conceitos fundamentais para compreensão e desenvolvimento do presente trabalho. Nele é apresentado toda a pesquisa realizada para o desenvolvimento do trabalho de conclusão de curso.

2.1 Javascript

Na visão de Alves (2014, p.127) “Javascript é uma linguagem de script, desenvolvida em 1995 por Brendam Eich, da Netscape, com base em outra linguagem, denominada EMACScript. Inicialmente ela foi chamada LiveScript”.

> 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).

Segundo Alves (2014) a linguagem possibilita o desenvolvimento de rotinas desde as mais simples até as mais complexas, além disto a linguagem é orientada por objeto e executa independente do sistema operacional.

> As variáveis em Javascript podem armazenar diversos tipos de dados, valores numéricos, datas, hora, cadeias de caracteres, valores lógicos etc. Quando uma variável é declarada dentro de uma rotina, que em Javascript é denominada função, somente ela pode ter acesso aos valores. Em outras palavras, a variável somente é vista pela rotina que a declarou e não por qualquer outra. (ALVES, 2014, p.159).

Para Alves (2014) como em outras linguagens de programação, uma função é utilizada e criada para executar tarefas repetitivas, ou seja, é usada em diferentes partes dos códigos.

> O tipo fundamental de dados de JavaScript é o objeto. Um objeto é um valor composto: ele agrega diversos valores (valores primitivos ou outros objetos) e permite armazenar e recuperar esses valores pelo nome. Um objeto é um conjunto não ordenado de propriedades, cada uma das quais tendo um nome e um valor. Os nomes de propriedade são *strings*; portanto, podemos dizer que os objetos mapeiam *strings* em valores. (FLANAGAN, 2014, p.128).

Flanagan (2014, p.44) diz “Os tipos de JavaScript podem ser divididos em duas categorias: tipos primitivos e tipos de objeto. Os tipos primitivos de JavaScript incluem

números, sequências de texto (conhecidas como *strings*) e valores de verdade (conhecidos como booleanos) ”.

2.1.1 ES6

Segundo MDN (2021) ECMAScript é uma linguagem de script que é a base do Javascript, ou seja, podemos utilizar ECMAscript para nos referenciar ao Javascript, o es6 é uma versão de 2015, que trouxe várias novidades para a linguagem, dentre as principais mudanças foi a declaração de variáveis através de let e const e também a criação de *arrow functions.*

Segundo W3Schools (2021) a *arrow function* é uma forma de se escrever uma função de maneira menor, assim fazendo a mesma coisa que uma função convencional, porém utilizando menos espaço de código, podemos ver um exemplo de *arrow function* na Figura 1.

Figura 1 – Utilização de Arrow Function

```
hello = val => "Hello " + val;
```

Fonte: Elaborado a partir da documentação Javascript oferecida pela W3Schools (2021)

Para MDN (2021) a utilização do comando “let” para declaração de variável é muito parecida com o uso do comando “var”, porém a diferença do “let” é que ele é uma variável de escopo fechado, ou seja, a variável pode somente ser utilizada dentro daquele escopo, se for utilizada fora dele a variável não terá valor como se nunca tivesse sido inicializada, assim evitando o problema de o valor desta variável ser trocada em outro trecho de código, podemos ver o uso do “let” através da Figura 2.

Figura 2 – Utilização da declaração let

```
function letTest() {
  let x = 1;
  {
    let x = 2;  // different variable
    console.log(x);  // 2
  }
  console.log(x);  // 1
}
```

Fonte: Elaborado a partir da documentação Javascript oferecida pela MDN (2021)

Segundo MDN (2021) a utilização do comando “const” é utilizada na declaração de uma constante, ou seja, o valor desta variável não pode ser alterado depois de declarado, é utilizado para variável específicas que nunca vão mudar, podemos ver a declaração de uma constate conforme ilustrado na Figura 3.

Figura 3 – Utilização da declaração const

```
// define MY_FAV as a constant and give it the value 7
const MY_FAV = 7;
```

Fonte: Elaborado a partir da documentação Javascript oferecida pela MDN (2021)

2.1.2 React

Segundo React (2021) o React é uma biblioteca da linguagem Javascript, que é muito eficiente e flexível que permite ao programador criar diversas interfaces, auxiliando construir sistemas complexos a partir de pequenos e isolados códigos, conhecido como “componentes”, conforme ilustrado na Figura 4.

Figura 4 – Criação de componente

```
class ShoppingList extends React.Component {
  render() {
    return (
      <div className="shopping-list">
        <h1>Lista de compras para {this.props.name}</h1>
        <ul>
          <li>Instagram</li>
          <li>WhatsApp</li>
          <li>Oculus</li>
        </ul>
      </div>
    );
  }
}

// Exemplo de uso: <ShoppingList name="Mark" />
```

Fonte: Elaborado a partir da documentação ReactJS (2021)

No caso o *ShoppingList* é uma componente React do tipo classe, este componente recebe alguns parâmetros que são passados através das propriedades do componente e são acessados através do “this.props”, todo componente React necessita de um retorno. O método render retorna o conteúdo que será exibido em tela, sendo ele html puro ou “JSX “. (REACT, 2021)

React (2021) 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*, podemos ver um exemplo de JSX na Figura 5.

Figura 5 – Elemento JSX

```
const element = <h1>Hello, world!</h1>;
```

Fonte: Elaborado a partir da documentação ReactJS (2021)

React (2021) ressalta a importância do state e de suas duas funções principais funções *componentDidMount* e a *componentWillUnmount*, basicamente o state é uma variável global que pode ser acessada por todo o componente por meio do "this.state.nomedavariavel", o método *componentDidMount* sempre será executado quando o componente é criado, diferenciando-se do *componentWillUnmount* que é executado toda vez que o componente for destruído, podemos ver estes conceitos aplicados na Figura 6.

Figura 6 – Utilização do *state* e principais métodos

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
  }

  componentDidMount() {
    this.timerID = setInterval(
      () => this.tick(),
      1000
    );
  }

  componentWillUnmount() {
    clearInterval(this.timerID);
  }

  tick() {
    this.setState({
      date: new Date()
    });
  }

  render() {
    return (
      <div>
        <h1>Hello, world!</h1>
        <h2>It is {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  }
}

ReactDOM.render(
  <Clock />,
  document.getElementById('root')
);
```

Fonte: Elaborado a partir da documentação ReactJS (2021)

Segundo React (2021) 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* através da propriedade da *tag*, já para um componente React o comportamento de um evento é um pouco diferente ao invés de passarmos uma *string* como atributo podemos passar a própria função pois componentes JSX permitem usarmos comandos Javascript dentro de *tags* html, conforme podemos ver as diferenças entre as duas formas de utilização de eventos ilustrados na Figura 7.

Figura 7 – Diferença entre eventos Javascript Convencional X Componente React

```
// Aqui podemos ver a utilização de um evento em JavaScript Convencional
<button onclick="activateLasers()">
  Ativar lasers
</button>

// Aqui podemos ver a utilização de um evento em React
<button onClick={activateLasers}>
  Ativar lasers
</button>
```

Fonte: Elaborado a partir da documentação ReactJS (2021)

Para React (2021) 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, como podemos visualizar na figura 8.

Figura 8 – Utilização do *hook useState*

```
import React, { useState } from 'react';

function Example() {
  // Declare uma nova variável de state, a qual chamaremos de "count"
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
```

Fonte: Elaborado a partir da documentação ReactJS (2021)

React (2021) ainda diz sobre a utilização dos *hooks* na criação de *effects*, que são basicamente funções que são executadas na criação do componente muito parecido com o

método *componentDidMount* conforme visto anteriormente, porém a diferença deste hook é que quando é passado variáveis passadas num *array* de dependências quando ocorre alguma alteração nestas variáveis, a função é executada novamente, conforme ilustrado na figura 9.

Figura 9 – Utilização do *hook useEffect*

```
useEffect(() => {
  document.title = `Você clicou ${count} vezes`;
}, [count]); // Apenas re-execute o efeito quando o count mudar
```

Fonte: Elaborado a partir da documentação ReactJS (2021)

2.1.3 NodeJS

Para NodeJS (2021) 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 (2021) 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 assim viabilizando esta linguagem a ser utilizada no *backend* das aplicações.

2.1.4 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 covertido no fim para JavaScript, que é uma linguagem suportada por diversos navegadores.

Para Simas, Borges e Couto (2019) “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, podemos ver a utilização de tipos nas variáveis conforme ilustrado na Figura 10.

Figura 10 – Utilizando tipos em variáveis usando Typescript

```
let nomeVarBooleana: boolean = false;
let numInteiro: number = 23;
let numFlutuante: number = 499.99;
let numHexadecimal: number = 0xf00d;
let numBinario: number = 0b1010;
let numOctal: number = 0o744;
let nomeF: string = "Maria";
let nomeM: string = 'Joao';
let listaArray: number[] = [1, 2, 3, 4, 5];
ou
let listaArray: Array<number> = [1, 2, 3, 4, 5];
```

Fonte: Elaborado a partir de Simas, Borges e Couto (2019)

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, podemos ver a atribuição desse tipo na Figura 11.

Figura 11 – Utilizando tipo variável any usando Typescript

```
let variavel: any = 4;
variavel = "Agora eh uma string!";
variavel = true; // Agora é um booleano
```

Fonte: Elaborado a partir de Simas, Borges e Couto (2019)

2.1.4 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 um API robusta de forma rápida e simples. (EXPRESS, 2021).

2.1.5 Date-FNS

Segundo DateFNS (2021) o DateFNS é uma ferramenta muito simples para manipulação de Datas Javascript podendo ser utilizada tanto nos navegadores ou em um ambiente NodeJS, possuindo mais de 140 funções que possibilitam manipular as datas. Podemos ver um exemplo da utilização desta biblioteca na Figura 12.

Figura 12 – Utilização do DateFNS para tratamento de data

```
import { format, compareAsc } from 'date-fns'

format(new Date(2014, 1, 11), 'MM/dd/yyyy')
//=> '02/11/2014'

const dates = [
  new Date(1995, 6, 2),
  new Date(1987, 1, 11),
  new Date(1989, 6, 10),
]
dates.sort(compareAsc)
//=> [
//   Wed Feb 11 1987 00:00:00,
//   Mon Jul 10 1989 00:00:00,
//   Sun Jul 02 1995 00:00:00
// ]
```

Fonte: Elaborado a partir da documentação ReactJS (2021)

2.1.6 Material-UI

Segundo Monteiro (2021) Material UI é a melhor biblioteca de uso geral, ela segue os padrões, componentes e ferramentas de Material Design que são do Google, possibilitando aplicativos web mais velozes.

Para Monteiro (2021) “O Material-UI começou como uma implementação do React da especificação do Material Design do Google em 2014. O objetivo era simples, dar aos desenvolvedores do React o direito de usar o Material Design. ”

2.2 BANCO DE DADOS

Segundo Ramakrishnan e Gehre (2008) o banco de dados é uma coleção de vários dados que tem como objetivo descrever as atividades de uma ou mais organizações, como por exemplo o banco de um colégio, ele vai ter as entidades como aluno e professor e os relacionamentos, como turma que seria o relacionamento de aluno e professor.

Ramakrishnan e Gehre (2008) afirma que "Uma entidade é um objeto do mundo real distinguível de outros objetos. Exemplos incluem: o brinquedo Dragonzord Verde, o departamento de brinquedos, o gerente do departamento de brinquedos, o endereço residencial do gerente do departamento de brinquedos. "

Para Ramakrishnan e Gehre (2008) "Para cada conjunto de entidades, escolhemos uma chave, que consiste em um conjunto mínimo de atributos cujos valores identificam unicamente uma entidade do conjunto.".

> Uma entidade é descrita utilizando-se um conjunto de atributos. Todas as entidades em um determinado conjunto de entidades têm os mesmos atributos; [...] Nossa escolha de atributos reflete o nível de detalhes com os quais desejamos representar as informações sobre as entidades. Por exemplo, o conjunto de entidades Funcionários poderia usar nome, código de pessoa física (CPF) e vaga de estacionamento como atributos. (GEHRE; RAMAKRISHAN, 2014, p.24).

Para Coronel e Rob (2011) o projeto de banco de dados é uma estrutura de dados desenvolvida para armazenar e gerenciar dados do usuário final. Um banco de dados bem projetado facilita o gerenciamento dos dados e gerar dados valiosos.

Segundo Coronel e Rob (2011) para compreender o projeto de banco de dados, temos que entender a diferença entre dado e informação. Dados são fatos brutos, já a informação é o resultado do processamento destes dados para obter algum significado.

> Na atual "era da informação", a produção de informações precisas, relevantes e rápidas é a chave para uma boa tomada de decisão. Por sua vez, uma boa tomada de decisão é a chave para a sobrevivência comercial no mercado global. Dizem que estamos entrando na "era do conhecimento" os dados são o fundamento das informações, que é a base do conhecimento, ou seja, do corpo de informações e fatos sobre um assunto específico. (ALVES, 2014, p.130).

Coronel e Rob (2011) afirmam que "Em geral, os dados são armazenados em um banco de dados. Para implementar um banco de dados e gerenciar se conteúdo, é necessário um sistema de gerenciamento de banco de dados (SGBD) ".

Segundo Gehre e Ramakrishnan (2008) um SGBD ou sistema de gerenciamento de banco de dados, é um programa que tem como objetivo auxiliar a manutenção e utilização de vastos conjuntos de dados.

Gehre e Ramakrishnan (2008) dizem que "A Linguagem de Consulta Estruturada (SQL *Structured Query Language*) é a linguagem de banco de dados relacional comercial mais amplamente utilizada."

2.2.1 Modelagem

Para Coronel e Rob (2011) A modelagem de dados é a primeira etapa do projeto, é criada um a modelagem de dados específico 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 desenvolve, 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) explica os tipos de entidades: entidade, entidade físicas e a entidades lógicas, conforme apresentado no Quadro 1.

Quadro 1 – 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. A entidade colaboradora se refere ao funcionário ou funcionária da corporação. Já a entidade inventário é um objeto de propriedade da empresa, como um armário, uma impressora ou uma cafeteira.

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, etc. A entidade grupos de usuários do sistema existe para regular as permissões dos usuários ao sistema; a entidade venda é necessária para relacionar a conexão de entrega de algo da entidade física produto para alguma pessoa ou empresa da entidade física cliente, guardando os dados dessa entrega, como valores, data, etc.; por fim, a entidade relatório existe para puxar informações da própria entidade lógica venda e apresentar os dados ao usuário ou gerente que a solicitou.

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

No próximo capitulo será abordado a metodologia da pesquisa, explicando um pouco do que se trata a pesquisa, definições de tecnologias e ferramentas.

3. METODOLOGIA DA PESQUISA

O presente trabalho de conclusão de curso caracteriza-se como pesquisa aplicada e descritiva, pois seu objetivo é desenvolver uma aplicação web responsiva que possua umas rotinas para usuário farmácias para controle das dos pedidos, produtos e categorias da farmácia e rotina para cliente possibilitando criar conta, realizar pedidos.

O trabalho buscou responder aos seguintes problemas: Como realizar pedidos para uma farmácia de uma maneira digital, organizada e simples?

Após a finalização do levantamento bibliográfico, foi realizado um estudo mais aprofundado sobre o desenvolvimento de aplicações web responsivas atualmente, utilizando as ferramentas e padrões de desenvolvimentos mais modernos possíveis, visando deixar a aplicação relevante por mais tempo, evitando futuros retrabalhos, migrações para novos padrões e facilitando possíveis adições futuras ao trabalho.

Com base nessa pesquisa foi definido que a aplicação seria feita na IDE Visual Studio Code, a linguagem utilizada será Javascript, usando as bibliotecas React na criação da interface e utilizando o framework Material UI para auxiliar a criação do *frontend* e a Biblioteca NodeJS utilizada na criação do servidor, utilizando a biblioteca Express para auxiliar o desenvolvimento da API, a biblioteca TypeORM para auxiliar o relacionamento com o banco de dados e a biblioteca Date-fns para auxiliar com o trabalho com datas.

O banco escolhido foi o postgres. A escolha destas bibliotecas foi devido a futuramente a aplicação ter a necessidade de se tornar um aplicativo, que será utilizado o React Native que é muito semelhante ao React utilizado na web, também por utilizar a mesma linguagem no caso o Javascript torna o desenvolvimento do *frontend* e do *backend* muito mais simples pois a sintaxe será a mesma, evitando a utilização de mais linguagem para *frontend* e *backend* onde suas sintaxes são diferentes.

4. SISTEMA DE DELIVERY DE FARMÁCIAS (PHARMAPP)

Neste capítulo serão abordados todos os passos que foram executados durante o desenvolvimento desse trabalho.

4.1 VISÃO GERAL DO SISTEMA

O objetivo principal do sistema é possibilitar que as farmácias possuam uma maneira de vender seus produtos de forma online, e que os clientes consigam realizar pedidos de produtos farmacêuticos de uma forma fácil, rápida e online. O sistema será dividido em duas partes uma dela sendo o *frontend* e a outra sendo a API representando o *backend*.

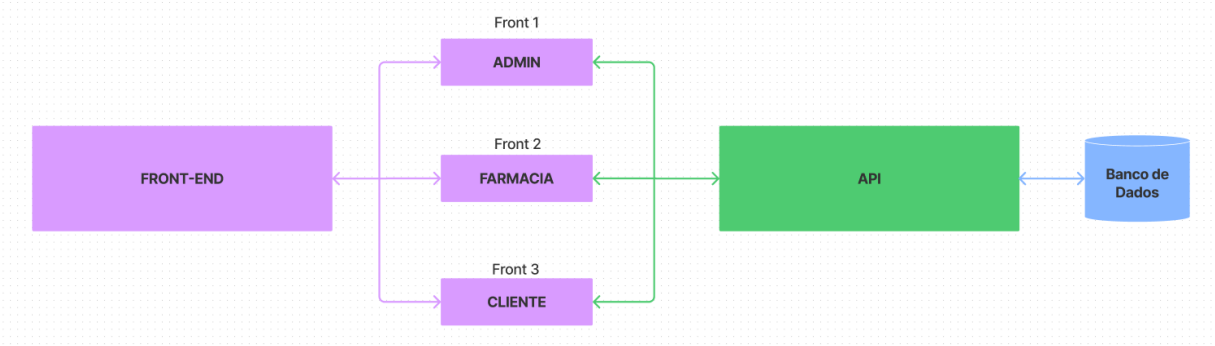
A aplicação *frontend* também foi dividida porem em três partes, a primeira parte é referente ao administrador, onde somente autenticado este usuário poderá gerenciar novas farmácias e gerenciar usuários referentes a uma farmácia especifica.

A segunda é responsável pelo controle de uma farmácia em especifico onde somente usuário que estão relacionados com elas poderão administrá-la, podendo gerenciar categorias, gerenciar e relacionar produtos com categorias e por fim gerenciar.

E por fim a última parte da aplicação se refere ao usuário cliente, que poderá se cadastrar na plataforma, conseguir visualizar todas as farmácias disponíveis, poder entrar numa farmácia e visualizar suas categorias e os produtos relacionados com ela e também realizar um pedido com os produtos selecionados.

O fluxo do sistema é basicamente um cliente servidor, onde o *frontend* terá telas que serão disponíveis somente para o administrador, outras para farmácias e para o fim outras telas somente para cliente, todas estas telas irão se comunicar com a API REST através do protocolo HTTP, quando a requisição chegar na API vai acontecer um processamento por parte do *backend* e caso não aconteça algum erro salvará os dados no banco de dados. Conforme demostrando o fluxo na figura 13.

Figura 13 – Fluxo da aplicação



Fonte: Acervo do autor.

4.2 REQUISITOS

Os requisitos são divididos em requisitos funcionais que são responsáveis pelas atividades / rotinas que serão realizadas nos sistemas, e requisitos não funcionais que demonstram características de qualidades, podendo ser sobre segurança, performance, restrições etc. No quadro 2, é apresentado os requisitos funcionais que são referentes aos usuários administradores.

Quadro 2 – Requisitos Funcionais Usuário Administrador

Número	Nome	Descrição
RF01	Login	O sistema deve permitir que para acessar as telas referentes ao usuário administrador seja feito um login com e-mail e senha.
RF02	Criar farmácias	O sistema deve dispor uma rotina para criar farmácias, tendo como os campos: nome, telefone, CNPJ, horário início do expediente e horário de fim do expediente.
RF03	Consulta de farmácias	O sistema deve dispor uma rotina para listar todas as farmácias
RF04	Alterar visibilidade da farmácia	O sistema deve dispor de uma rotina para alterar a visibilidade da farmácia, habilitando ou desabilitando a farmácia.
RF05	Alterar farmácia	O sistema deve dispor de uma rotina para alterar as informações da farmácia
RF06	Criar usuário farmácia	O sistema deve dispor uma rotina para criar usuário farmácia, tendo como os campos: nome, e-mail, senha e relação com a farmácia selecionada
RF07	Consulta de usuários farmácias	O sistema deve dispor uma rotina para listar todos os usuários farmácia
RF08	Alterar visibilidade usuário farmácia	O sistema deve dispor de uma rotina para alterar a visibilidade de um usuário farmácia, habilitando ou desabilitando a farmácia.
RF09	Alterar usuário farmácia	O sistema deve dispor de uma rotina para alterar as informações do usuário farmácia

Fonte: Acervo do autor

No quadro 3, são apresentados os requisitos funcionais referentes a usuários farmácia, que serão responsáveis pela administração da farmácia que estão relacionados

Quadro 3 – Requisitos Funcionais Usuário Farmácia

Número	Nome	Descrição
RF10	Login Usuário Farmácia	O sistema deve permitir que para acessar as telas referentes ao usuário farmácia seja feito um login com e-mail e senha.
RF11	Criar categorias	O sistema deve dispor uma rotina para criar categorias, tendo como os campo: descrição
RF12	Consulta de categorias	O sistema deve dispor uma rotina para listar todas as categorias
RF13	Alterar visibilidade da categoria	O sistema deve dispor de uma rotina para alterar a visibilidade da categoria, habilitando ou desabilitando a categoria.
RF14	Alterar categoria	O sistema deve dispor de uma rotina para alterar as informações da categoria
RF15	Criar produto	O sistema deve dispor uma rotina para criar produtos, tendo como os campos: detalhes, descrição, preço, informação adicionais, necessita receita, categoria.
RF16	Consulta de produtos	O sistema deve dispor uma rotina para listar todos os usuários farmácia
RF17	Alterar visibilidade do produto	O sistema deve dispor de uma rotina para alterar a visibilidade do produto, habilitando ou desabilitando o produto.
RF18	Alterar produto	O sistema deve dispor de uma rotina para alterar as informações do produto
RF19	Consulta de Pedidos	O sistema deve dispor de uma rotina para listar todos os pedidos com base na data selecionada e a situação do pedido também
RF20	Alterar situação de pedidos	O sistema deve dispor de uma rotina para alterar a situação do pedido

Fonte: Acervo do autor

No quadro 4, são apresentados os requisitos funcionais referentes a usuários cliente.

Quadro 4 – Requisitos Funcionais Usuário Cliente

Número	Nome	Descrição
RF21	Login Usuário Cliente	O sistema deve permitir que para acessar as telas referentes ao usuário cliente seja feito um login com e-mail e senha.
RF22	Cadastro de usuário cliente	O sistema deve dispor uma rotina para cadastro de usuário cliente, tendo como os campo: nome, e-mail, CPF, data de nascimento, telefone, senha
RF23	Consultas de usuário endereços	O sistema deve dispor uma rotina para retornar os endereços dos usuários do usuário logado.
RF24	Cadastro de Usuário Endereço	O sistema deve dispor uma rotina para cadastro de usuário cliente, tendo como os campo: CEP, estado, cidade, bairro, rua, complemento, número endereço, observação
RF25	Busca BrasilAPI com base no CEP	O sistema deve dispor uma rotina para buscar informações do endereço com base no CEP
RF26	Consultas de farmácia	O sistema deve dispor de uma rotina para consultar todas as farmácias cadastradas que estão disponíveis
RF27	Visualizar Farmácia	O sistema deve dispor de uma rotina para trazer informações sobre a farmácias
RF28	Consulta de Categorias com base na farmácia	O sistema deve dispor uma rotina para trazer todas as categorias disponíveis com base na visualização da farmácia selecionada.
RF29	Consulta de Produtos com base na categorias	O sistema deve dispor uma rotina para trazer todas os produtos disponíveis com base na categoria selecionada.

RF30	Realizar pedidos	O sistema deve dispor de uma rotina para realizar pedidos com base nos produtos selecionados e no endereço selecionado.
RF31	Consultar pedidos	O sistema deve dispor de uma rotina para consultar todos os pedidos com base no usuário .

Fonte: Acervo do autor

Com base no próximo quadro será apresentado requisitos opcionais que não fazem parte do principal escopo do projeto, porem serão requisitos para ser desenvolvidos em versões futuras. Podendo ser visto no Quadro 5.

Quadro 5 – Requisitos Opcionais

Número	**Nome**	**Descrição**
RF32	Socket	O sistema utilizará socket para fazer atualizações em tempo real, no que se refere a realização do pedido.
RF33	Esqueci minha senha	O sistema poderá realizar a recuperação de senha através do e-mail do usuário.
RF34	Imagem do produto	O sistema permitirá vincular imagens com produto.
RF35	Rotina de recomendação de produto	O sistema permitirá analisar o perfil do usuário e recomendar produtos que combinam com o perfil.
RF36	Rotina de recomendação de farmácias	O sistema permitirá analisar o perfil do usuário e recomendar farmácias que combinam com o perfil.
RF37	Dashboard usuário administrador	O sistema permitirá ter um dashboard com informações relevantes de todas as farmácias.

Fonte: Acervo do autor

A seguir no Quadro 6, serão apresentados os requisitos não funcionais do sistema inteiro, levando em consideração a aplicação frontend e backend.

Quadro 6– Requisitos Não Funcionais do Sistema

Número	**Nome**	**Descrição**
RNF01	Autenticação Usuário Farmácia	O sistema deve gerar um *token* para o usuário farmácia com duração de 7 dias
RNF02	Autenticação Usuário Cliente	O sistema deve gerar um *token* para o usuário cliente com duração de 7 dias
RNF03	Autenticação Usuário Admin	O sistema deve gerar um *token* para o usuário admin com duração de 7 dias
RNF04	Banco de Dados	O banco de dados deve ser um banco postgres
RNF05	Criptografia	Todas as senhas devem ser criptografadas antes de ser inseridas
RNF06	Responsividade	A aplicação *frontend* deve ser responsivo para celulares, *tablets* e *desktop*
RNF07	*Frontend*	A linguagem utilizada para desenvolver deve ser Javascript, utilizando a biblioteca ReactJS
RNF08	*Backend*	A linguagem utilizada para desenvolver o *backend* deve ser o Javascript utilizando a biblioteca NodeJS

Fonte: Acervo do autor

Com base nos requisitos funcionais e requisitos não funcionais foram elaborados diagramas e também protótipos de tela que serão apresentados na próxima sessão.

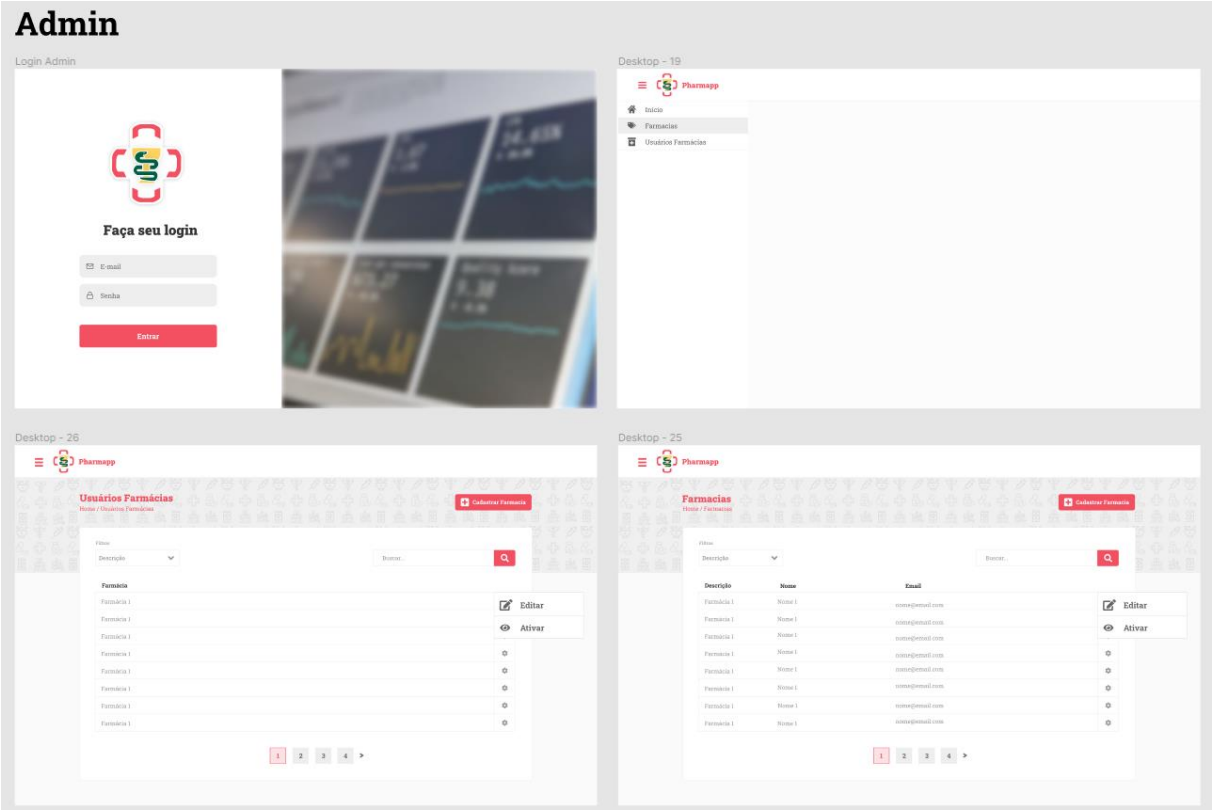
4.3 PROTÓTIPOS

Os protótipos são uma maneira simples de demostrar as principais ideias de como as telas irão se parecer, são elaborados com base nos requisitos funcionais, garantindo o cumprimento de cada requisito em sua particularidade, os protótipos de telas podem contemplar um ou mais requisitos.

4.3.1 Protótipos Aplicação usuário administrador

A seguir na Figura 14, serão apresentados todos os protótipos que se referem aos requisitos funcionais dos usuários tipo administrador.

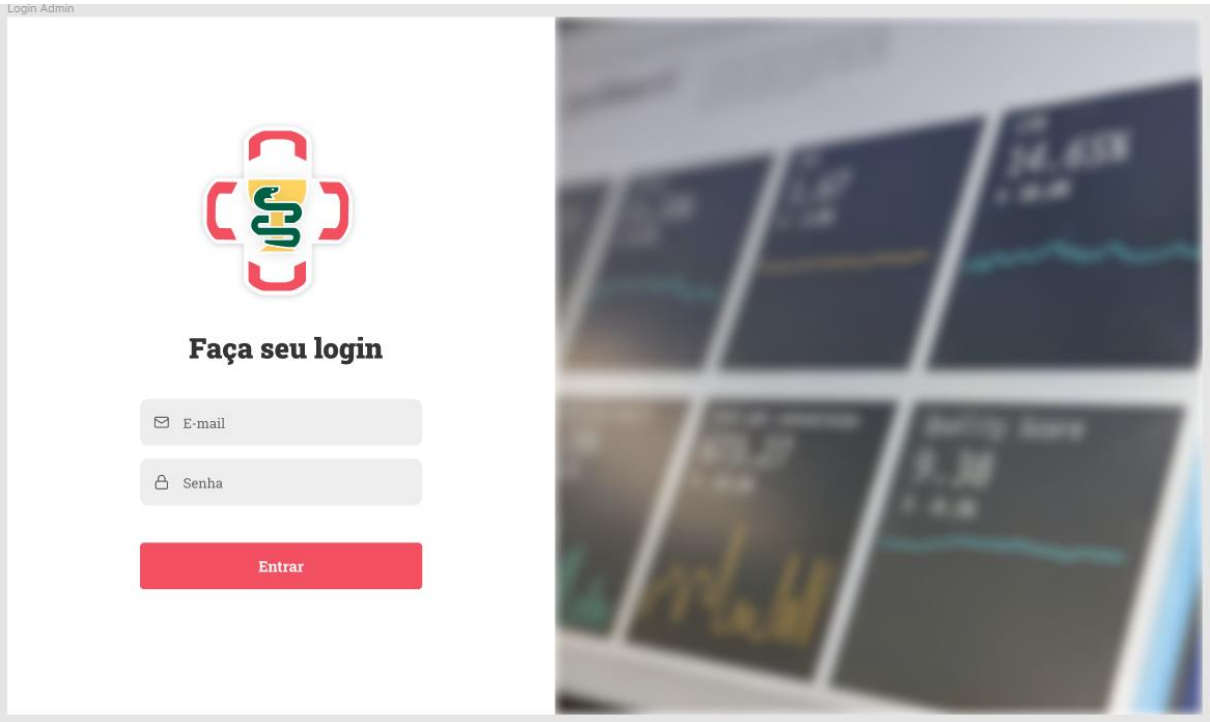
Figura 14 – Protótipo visão geral do fluxo da aplicação usuário administrador



Fonte: Acervo do autor

Na próxima Figura 15 encontra-se o protótipo de Tela de login do usuário administrador, através do e-mail e senha do usuário é realizado a autenticação do usuário, por meio desta tela o requisito RF01 será desenvolvido.

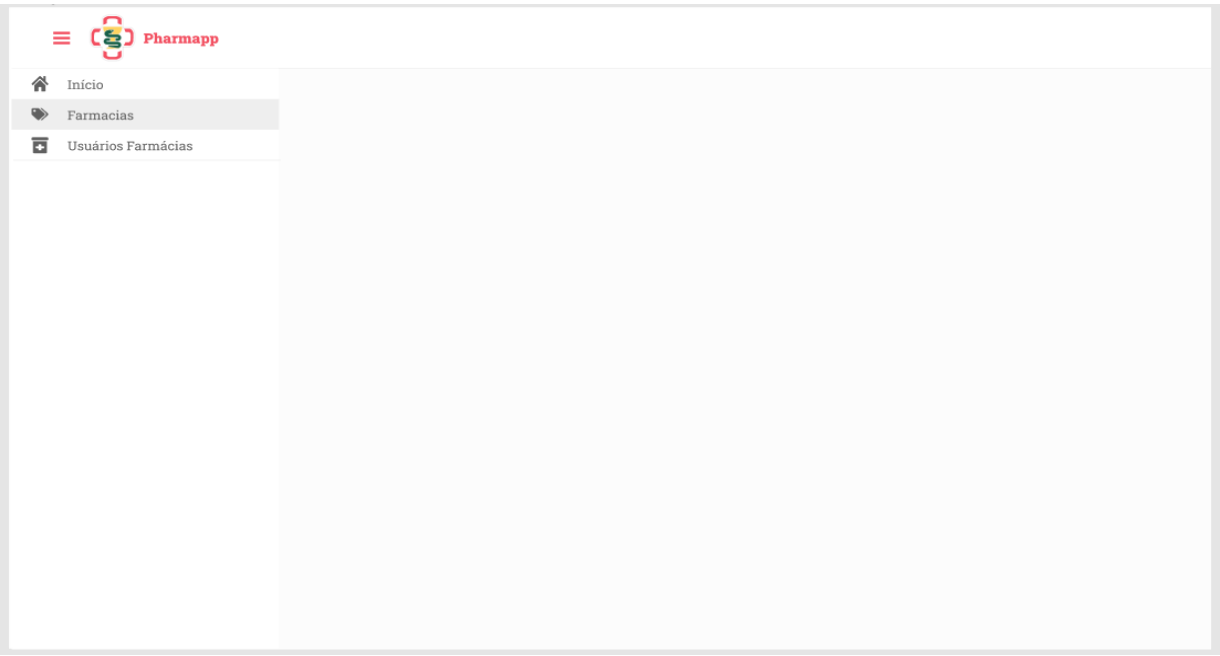
Figura 15 – Protótipo login da aplicação usuário administrador



Fonte: Acervo do autor

Na próxima Figura 16 encontra-se o protótipo de tela de *dashboard* do usuário administrador, através desta tela é possível navegar para as outras telas que implementarão os próximos requisitos funcionais.

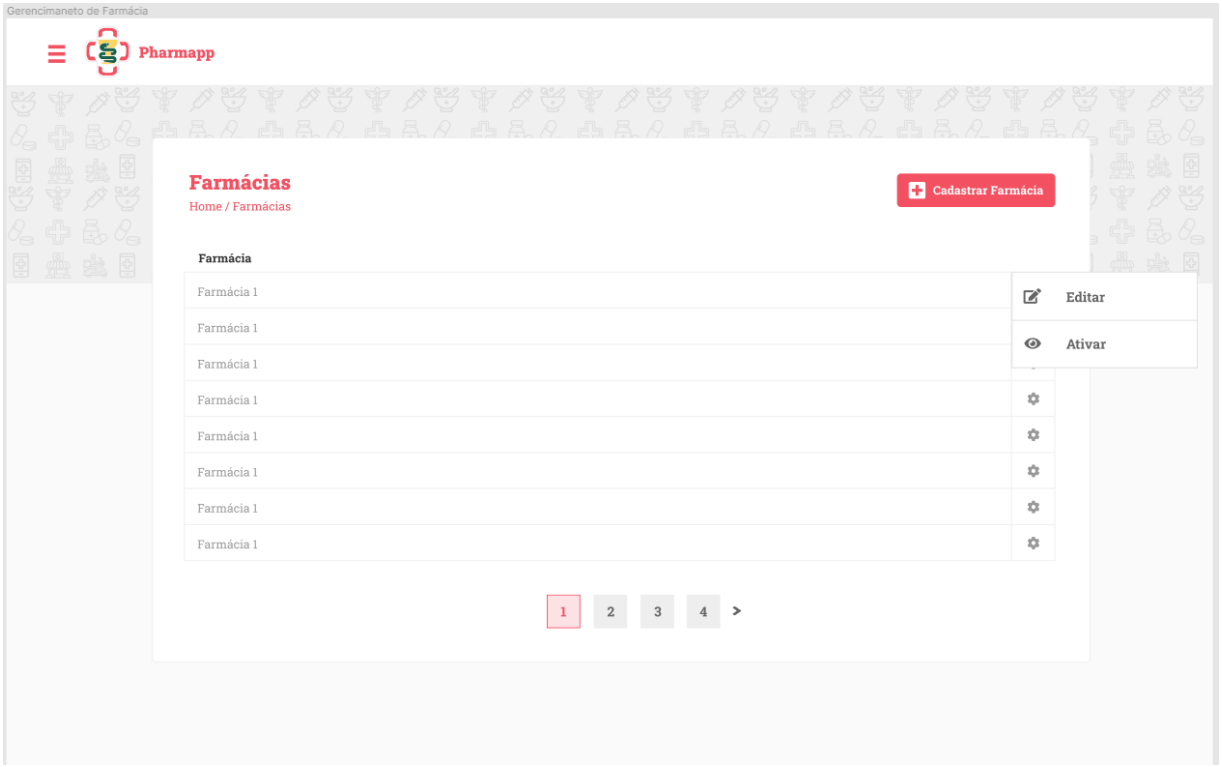
Figura 16 – Protótipo navegação aplicação usuário administrador



Fonte: Acervo do autor

Na Figura 17 encontra-se o protótipo da tela de gerenciamento de farmácia, através desta tela é possível consultar todas as farmácias existentes, cadastrar novas farmácias, editar as farmácias já existentes e alterar a visibilidade da farmácia selecionada, através deste protótipo os requisitos RF02, RF03, RF04, RF05 serão desenvolvidos.

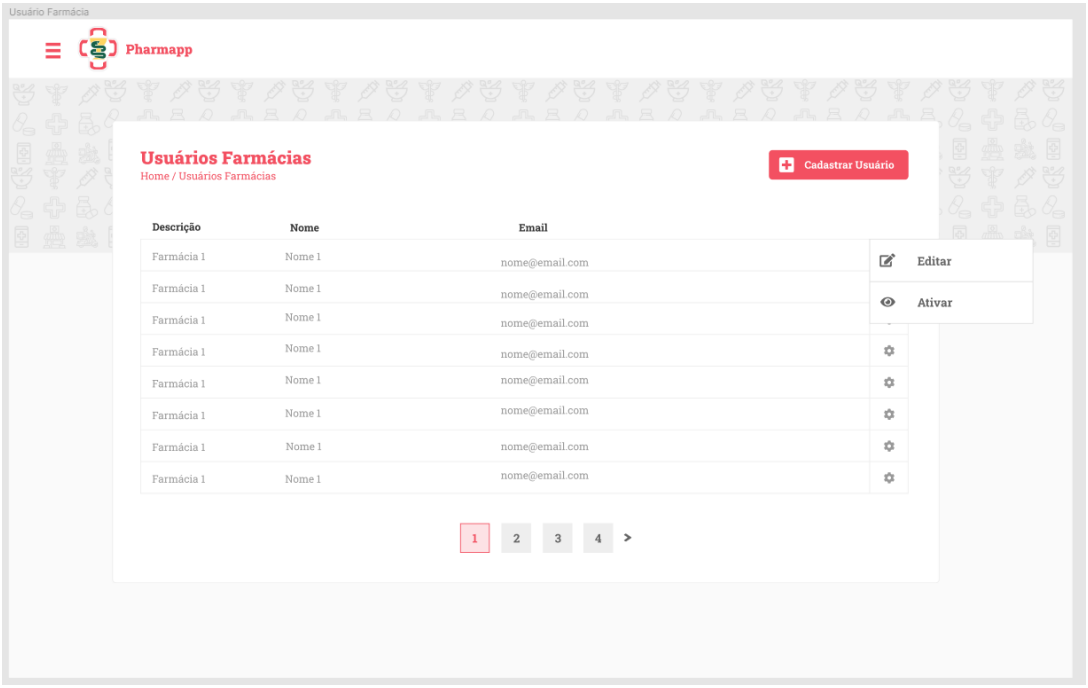
Figura 17 – Protótipo gerenciamento de Farmácia



Fonte: Acervo do autor

Na Figura 18 encontra-se também o protótipo da tela de gerenciamento do usuário farmácia, permitindo consultar todos os usuários farmácias, criar novos usuários farmácia, alterar as informações destes usuários e também alterar a visibilidade, com este protótipo os requisitos RF06, RF07, RF08, RF09 serão desenvolvidos.

Figura 18 – Protótipo gerenciamento de Usuário Farmácia

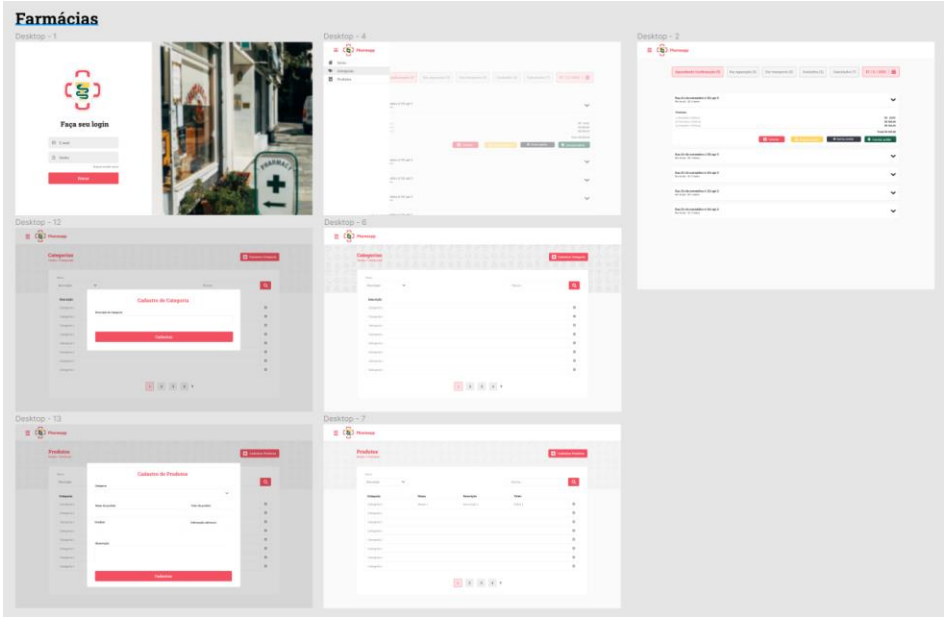


Fonte: Acervo do autor

4.3.2 Protótipos Aplicação usuário farmácia

A seguir na Figura 19, serão apresentados todos os protótipos que se referem aos requisitos funcionais dos usuários tipo farmácia.

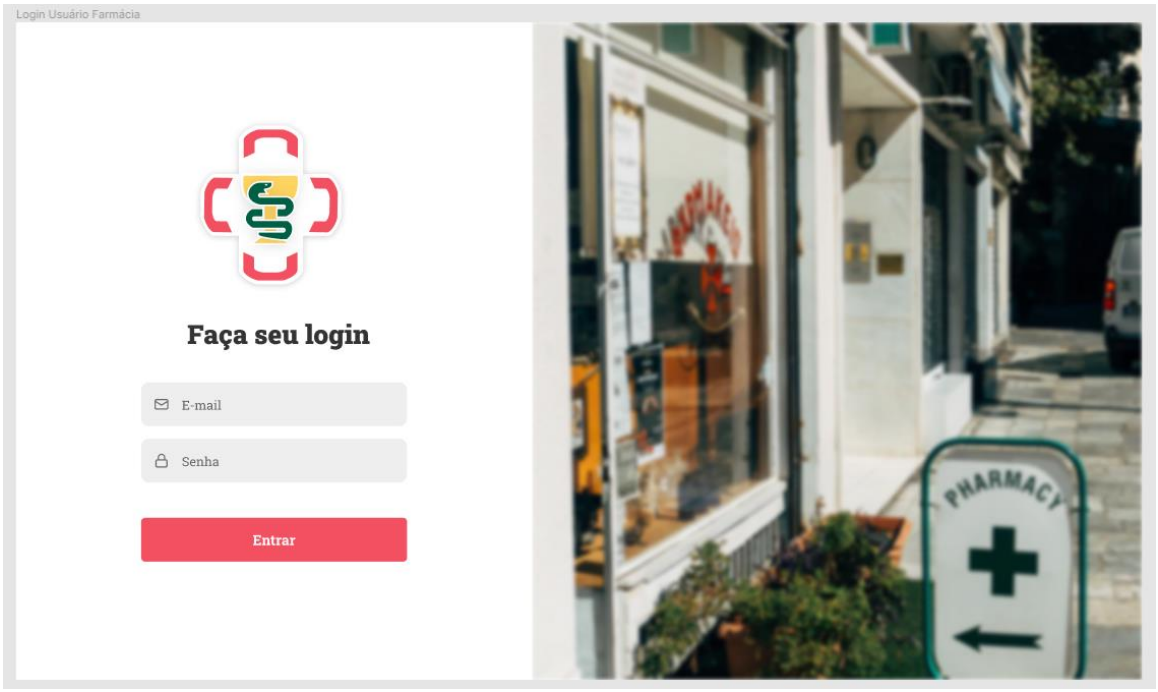
Figura 19 – Protótipo visão geral do fluxo da aplicação usuário farmácia



Fonte: Acervo do autor

Na próxima Figura 20 encontra-se o protótipo de tela de login do usuário farmácia, através do e-mail e senha do usuário é realizado a autenticação, por meio desta tela o requisito RF10 será desenvolvido.

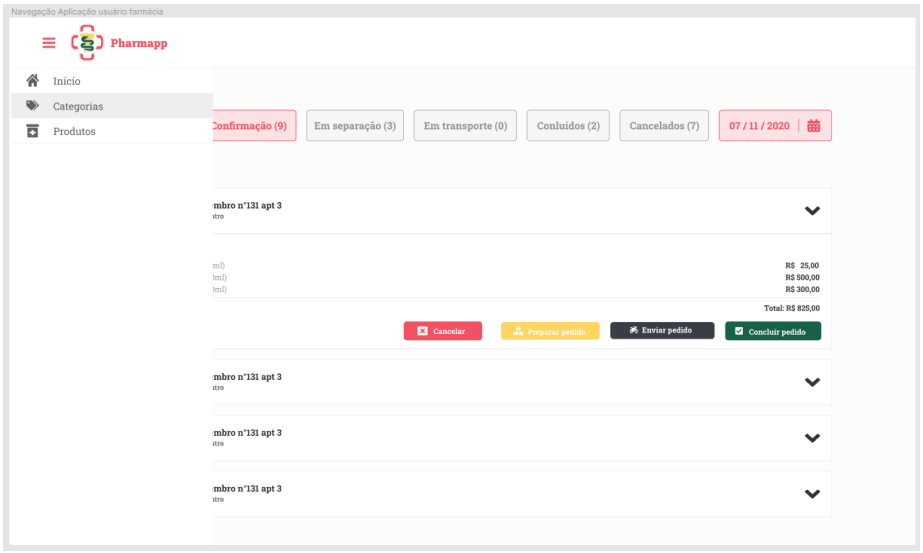
Figura 20 – Protótipo login Usuário Farmácia



Fonte: Acervo do autor

Na próxima Figura 21 encontra-se o protótipo de tela de navegação do usuário farmácia, através desta tela é possível navegar para as outras telas que implementarão os próximos requisitos funcionais.

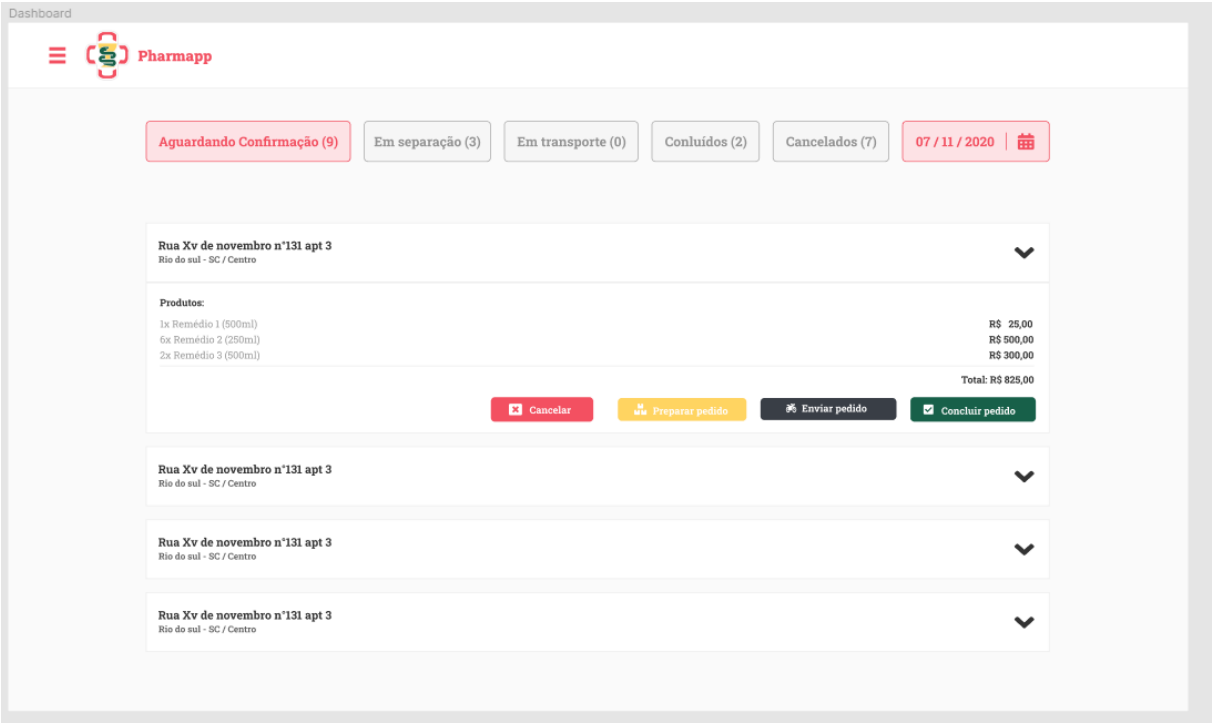
Figura 21 – Protótipo navegação aplicação usuário farmácia



Fonte: Acervo do autor

Na próxima Figura 22 encontra-se o protótipo de tela de *dashboard* do usuário farmácia, através desta tela é possível o usuário farmácia fazer o gerenciamento dos pedidos, por meio deste protótipo o requisito RF19 e RF20 será desenvolvido.

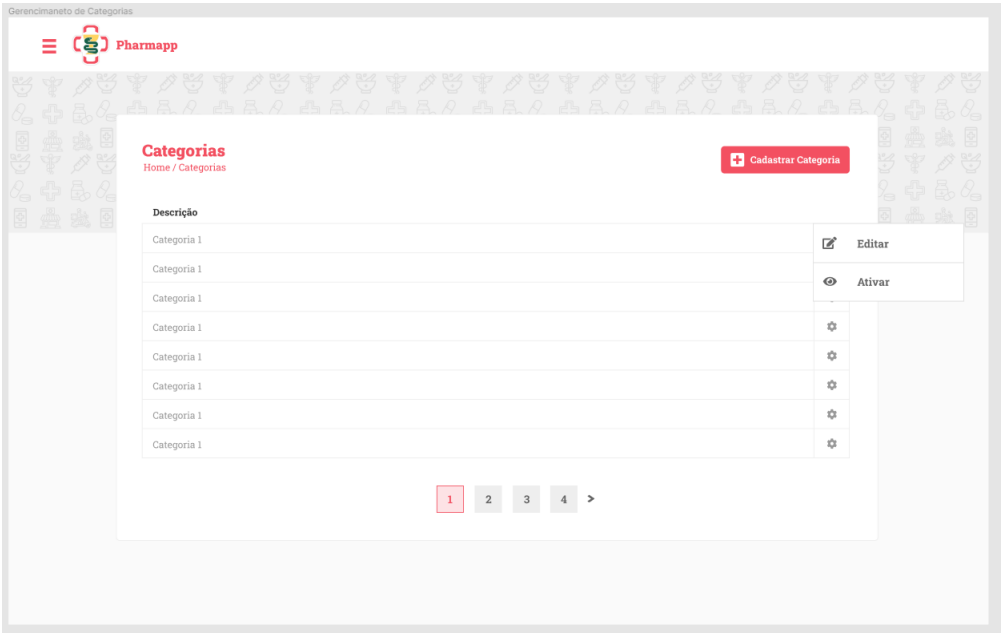
Figura 22 – Protótipo dashboard farmácia



Fonte: Acervo do autor

Na Figura 23 encontra-se o protótipo da tela de gerenciamento de categoria, permitindo consultar todas as categorias relacionadas a farmácia, criar novas categorias, alterar as informações das já existentes e também alterar a visibilidade, com este protótipo os requisitos RF11, RF12, RF13, RF14 serão desenvolvidos.

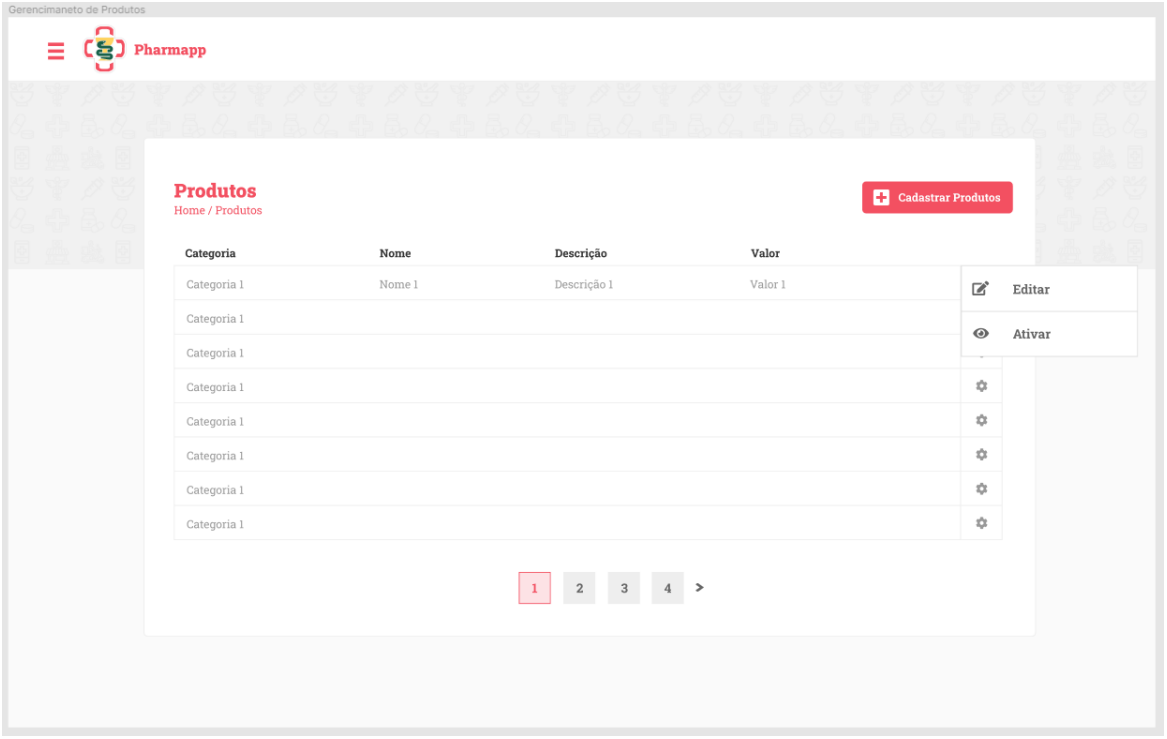
Figura 23 – Protótipo gerenciamento categoria



Fonte: Acervo do autor

Na Figura 24 encontra-se o protótipo da tela de gerenciamento de produtos, permitindo consultar todas os produtos da farmácia, criar novos produtos, alterar as informações do produto e também alterar a visibilidade, com este protótipo os requisitos RF15, RF16, RF17, RF18 serão desenvolvidos.

Figura 24 – Protótipo gerenciamento produtos

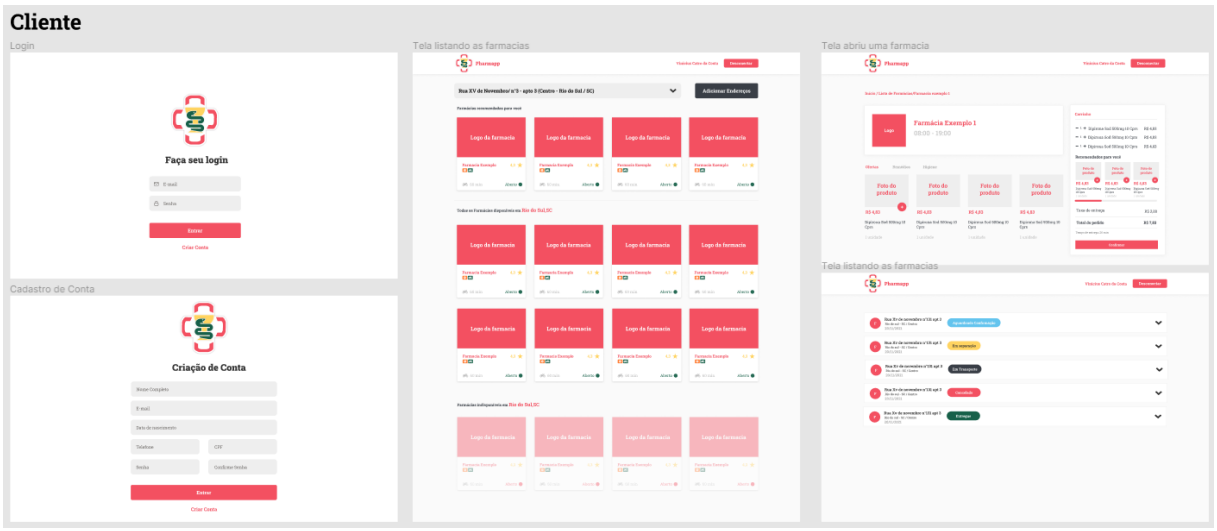


Fonte: Acervo do autor

4.3.3 Protótipos Aplicação usuário cliente

A seguir na Figura 25, serão apresentados todos os protótipos que se referem aos requisitos funcionais do usuário cliente.

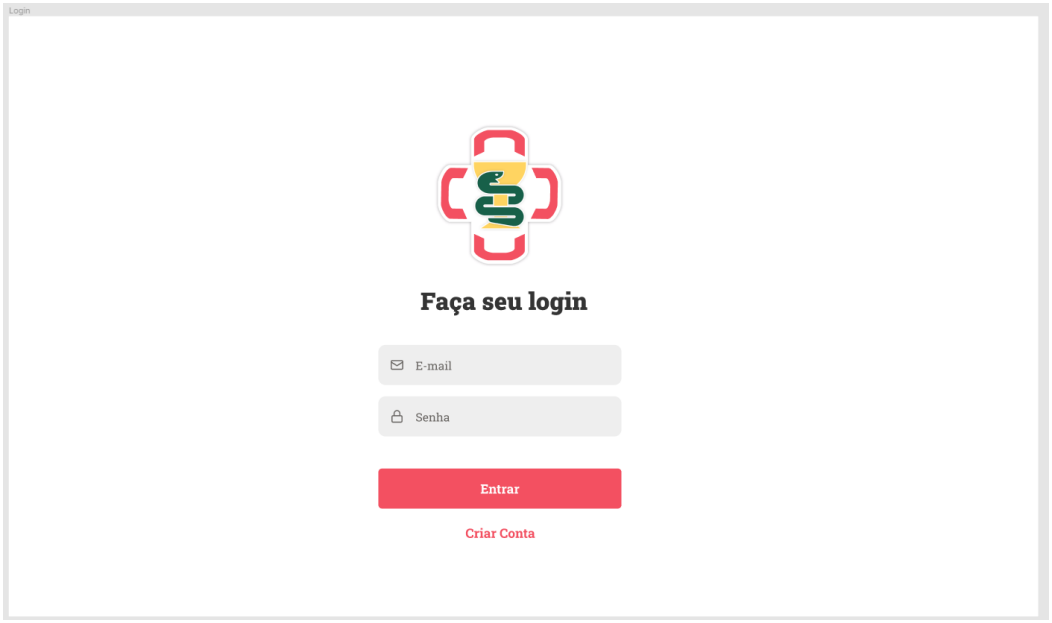
Figura 25 – Protótipo visão geral do fluxo da aplicação usuário cliente



Fonte: Acervo do autor

Na próxima Figura 26 encontra-se o protótipo de tela de login do usuário cliente, através do e-mail e senha do usuário é realizado a autenticação, por meio desta tela o requisito RF21 será desenvolvido.

Figura 26 – Protótipo login usuário cliente



Fonte: Acervo do autor

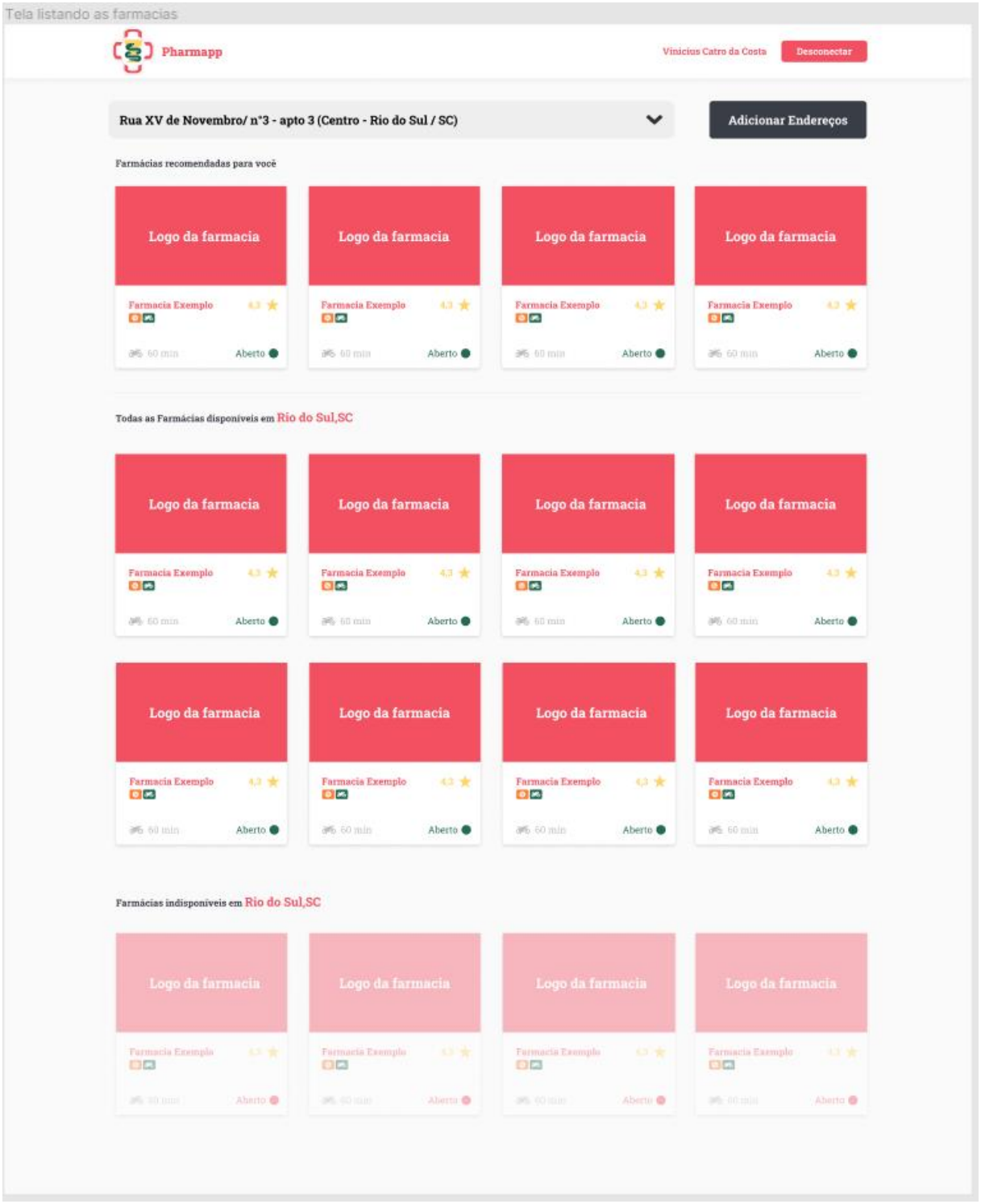
Na próxima Figura 27 encontra-se o protótipo de tela de cadastro do usuário cliente, através do nome completo, e-mail, data de nascimento, telefone, CPF, senha. Por meio desta tela o requisito RF22 será desenvolvido.

Figura 27 – Protótipo cadastro Usuário Cliente

Fonte: Acervo do autor

Na próxima Figura 28 encontra-se o protótipo de tela de listagem de farmácias, cadastro de endereços, consultas de usuário endereço. Por meio desta tela os requisitos RF23, RF24, RF25 e RF26 serão desenvolvidos.

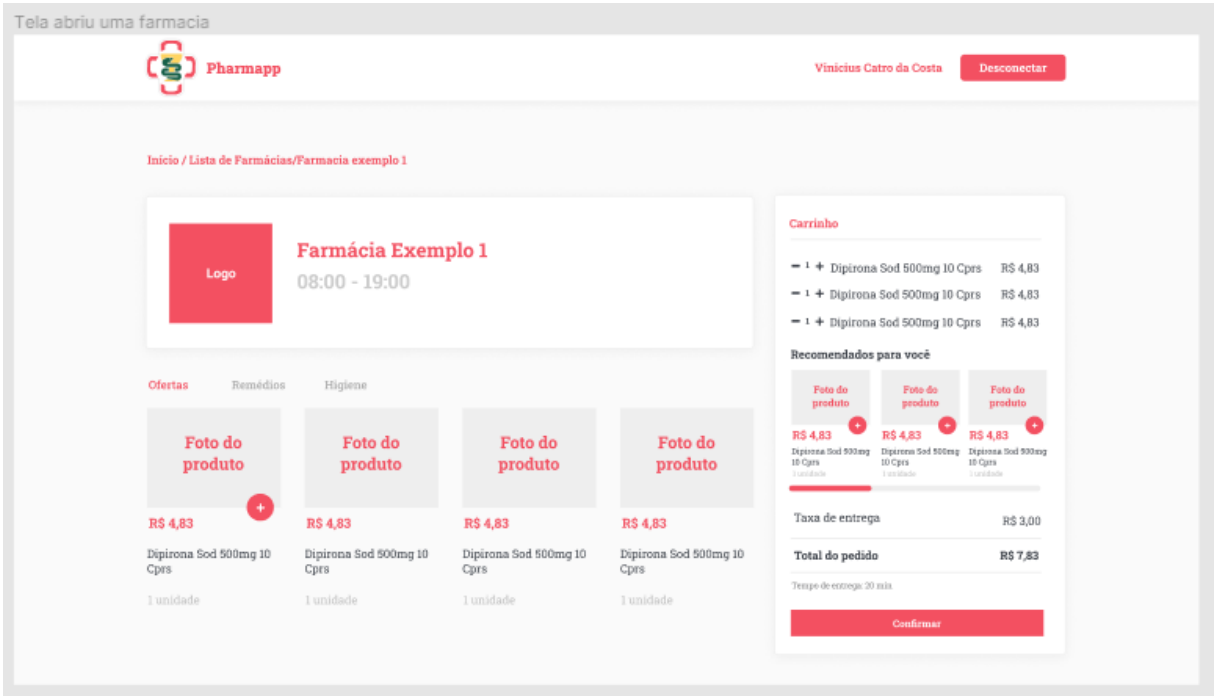
Figura 28 – Protótipo listagem de Farmácias



Fonte: Acervo do autor

Na próxima Figura 29 encontra-se o protótipo de tela de visualização de farmácia. Por meio desta tela os requisitos RF27, RF28, RF29 e RF30 serão desenvolvidos.

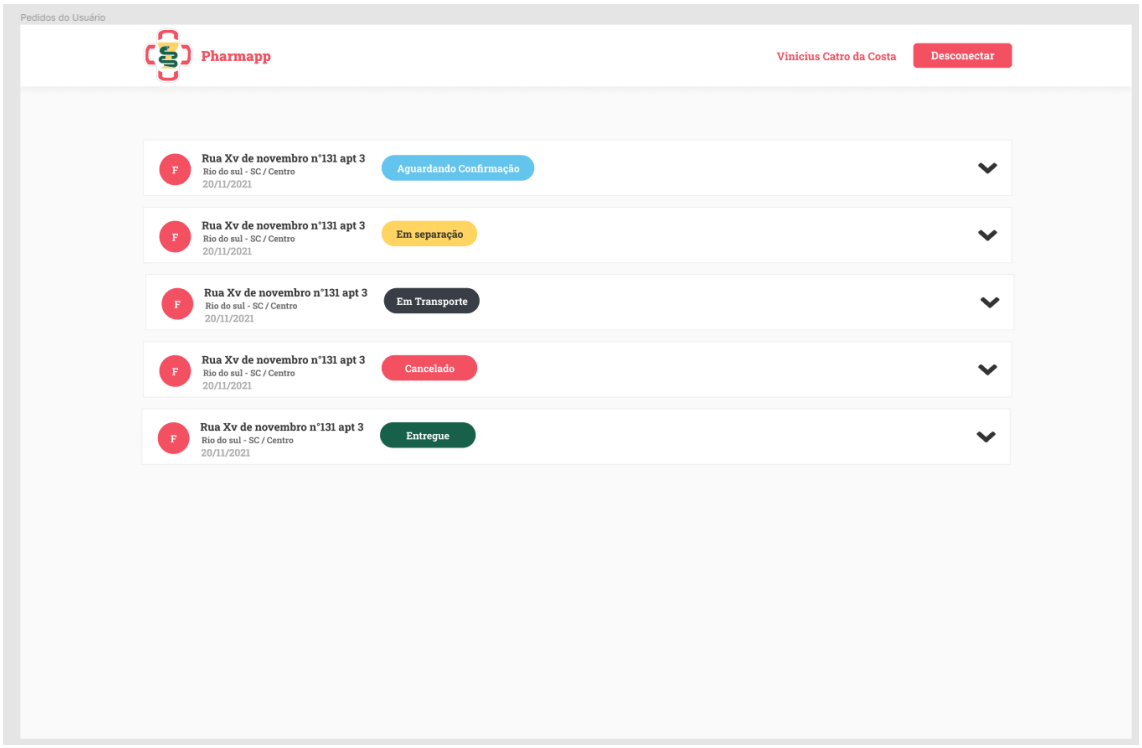
Figura 29 – Protótipo listagem de Farmácias



Fonte: Acervo do autor

Na próxima Figura 30 encontra-se o protótipo de tela de consulta de pedidos do usuário. Por meio desta tela os requisitos RF31 será desenvolvido.

Figura 30 – Protótipo listagem de pedidos do usuário

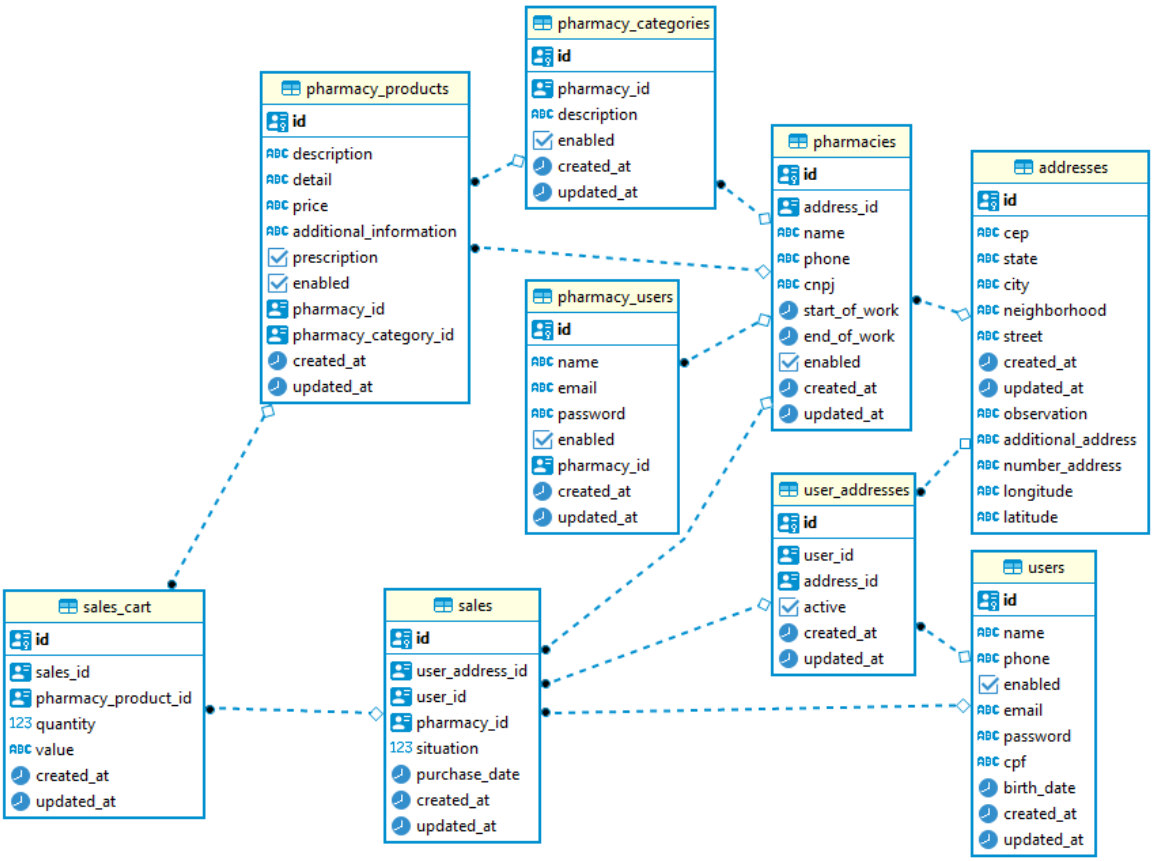


Fonte: Acervo do autor

4.4 DIAGRAMA DE ENTIDADE E RELACIONAMENTO

Na Figura 31 podemos ver o diagrama de entidade e relacionamento, que tem como objetivo demonstrar as tabelas que compõem o banco de dados, tendo como o principal relacionamento a relação do modelo de users que são o usuário cliente conforme anteriormente citado, com o pharmacy_produccts que são os produtos de uma determinada farmácia, gerando duas tabelas associativas para representar a venda a tabela sales e a tabela de sales_cart para representar os produtos que foram relacionados nesta compra.

Figura 31 – Modelagem do Banco de Dados



Fonte: Acervo do autor

4.5 IMPLEMENTAÇÂO

Nesta sessão será apresentado as tecnologias e ferramentas que fizeram parte do desenvolvimento de todo o sistema, tanto aplicação *frontend* quanto o *backend* da aplicação. Também será abortado a utilização e o funcionamento do sistema.

4.5.1 Tecnologias e ferramentas utilizadas no desenvolvimento

O ambiente de desenvolvimento foi utilizado uma máquina com sistema operacional Windows 10, também utilizando o WSL que é um subsistema do Windows para Linux, que basicamente possibilita o desenvolvedor utilize o ambiente Linux, também a maioria das ferramentas de linha de comando que este sistema operacional possui, não necessitando ter uma máquina virtual convencional ou um dual boot.

Para a elaboração dos protótipos foi utilizado o Figma, que é um software gratuito de prototipagem de projetos de design, também é um editor gráfico de vetores, ele pode ser utilizado através de um navegador convencional ou também possibilita instalar em sua máquina sendo utilizado como qualquer aplicativo.

Para a modelagem do banco de dados foi utilizado a ferramenta *SQL Power Architect*, que é um aplicativo instalável que possibilita criar um protótipo do projeto de banco de dados, podendo criar as tabelas, relacionamentos e colunas que serão um espelho do banco de dados que será utilizado no backend da aplicação.

Entrando na parte de elaboração do código, utilizou-se o editor de código fonte Visual Studio Code, que é um dos editores mais populares atualmente quando se trata de desenvolvimento de software, possibilitando instalar extensões que auxiliam o desenvolvedor.

Para auxiliar na criação da API foi utilizado o Insomnia que é uma ferramenta que ajuda o desenvolvedor a realizar testes na API, é uma ferramenta que possibilita o programador fazer qualquer requisição REST configurando ela como quiser, tornado os testes de API uma maneira rápida e fácil.

No desenvolvimento do sistema foi utilizado a linguagem de programação Javascript, que atualmente é uma das linguagens de programação mais utilizadas no mercado, devido a sua facilidade de sintaxe, métodos muito simples de utilizar e também muito dinamismo, para suprir a falta de tipos no JavaScript foi utilizado a biblioteca TypeScript que basicamente força o desenvolvedor usar tipos, facilitando a manutenções futuras, principalmente se esta manutenção for realizadas por outro desenvolvedor, pois ele pode saber os atributos de cada variável utilizada.

Para o desenvolvimento do *backend* foi realizada um API REST, foi utilizado a biblioteca NodeJS que é baseada no interpretador v8 do Google, possibilitando a utilização do JavaScript fora do navegador, também foi utilizado a biblioteca Express, podemos ver a facilidade de criar uma API utilizando Express conforme demonstrar a Figura 32.

Figura 32 – Utilização do Express

```
import express from 'express';
const app = express();

app.get('/', (req, res) => {
  res.send('Hello World!');
});

app.listen(3333, () => {
  console.log(`Server Started on Port 3333`);
});
```

Fonte: Acervo do autor

Para criação dos modelos que são objetos que espelham as tabelas do banco, objetos que possuem as mesmas propriedades sendo elas as colunas de uma tabela, para isto foi utilizado uma famosa biblioteca ORM quando se trata de ORM utilizando TypeScript é biblioteca é a TypeORM. Na Figura 33, podemos ver a utilização da biblioteca na implementação de um modelo de Profile.

Figura 33 – Criação de um modelo utilizando TypeORM

```
import { Entity, PrimaryGeneratedColumn, Column } from 'typeorm';

@Entity()
export class Profile {
  @PrimaryGeneratedColumn()
  id: number;

  @Column()
  gender: string;

  @Column()
  photo: string;
}
```

Fonte: Acervo do autor

Na Figura 34, podemos ver como é feito os relacionamentos que são muito importantes quando se fala de banco de dados, a figura demonstra o relacionamento 1-N e N-1.

Figura 34 – Criação de um relacionamento utilizando TypeORM

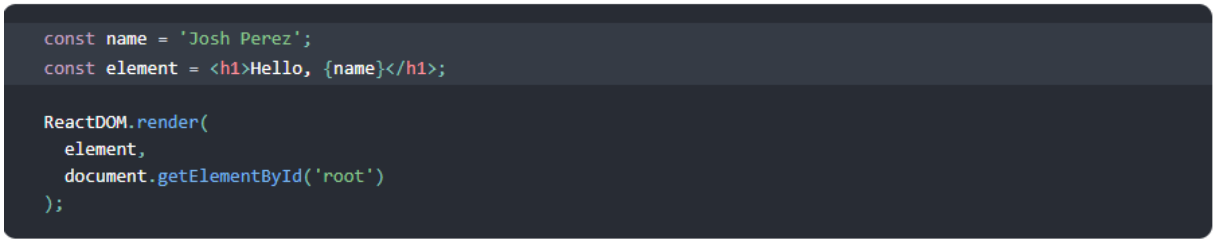
```
import { Entity, PrimaryGeneratedColumn, Column, ManyToOne } from 'typeorm';
import Profile from './Profile';

@Entity()
export default class Photo {
  @PrimaryGeneratedColumn()
  id: number;

  @Column()
  url: string;

  @ManyToOne(() => Profile, profile => profile.photos)
  profile: Profile;
}
```

```
import { Entity, PrimaryGeneratedColumn, Column, OneToMany } from 'typeorm';
import Photo from './Photo';

@Entity()
export default class Profile {
  @PrimaryGeneratedColumn()
  id: number;

  @Column()
  name: string;

  @OneToMany(() => Photo, photo => photo.profile)
  photos: Photo[];
}
```

Fonte: Acervo do autor

Na criação do *frontend* foi utilizado a linguagem de programação Javascript juntamente com a biblioteca TypeScript e também a principal biblioteca de criação de interface que é o ReactJS. O React é um *framework* tendo como seu diferencial a utilização do JSX que basicamente é a possibilita o uso de JS no meio de um HTML, conforme na Figura 35, podemos ver um exemplo da utilização do JSX.

Figura 35 – Utilização do JSX

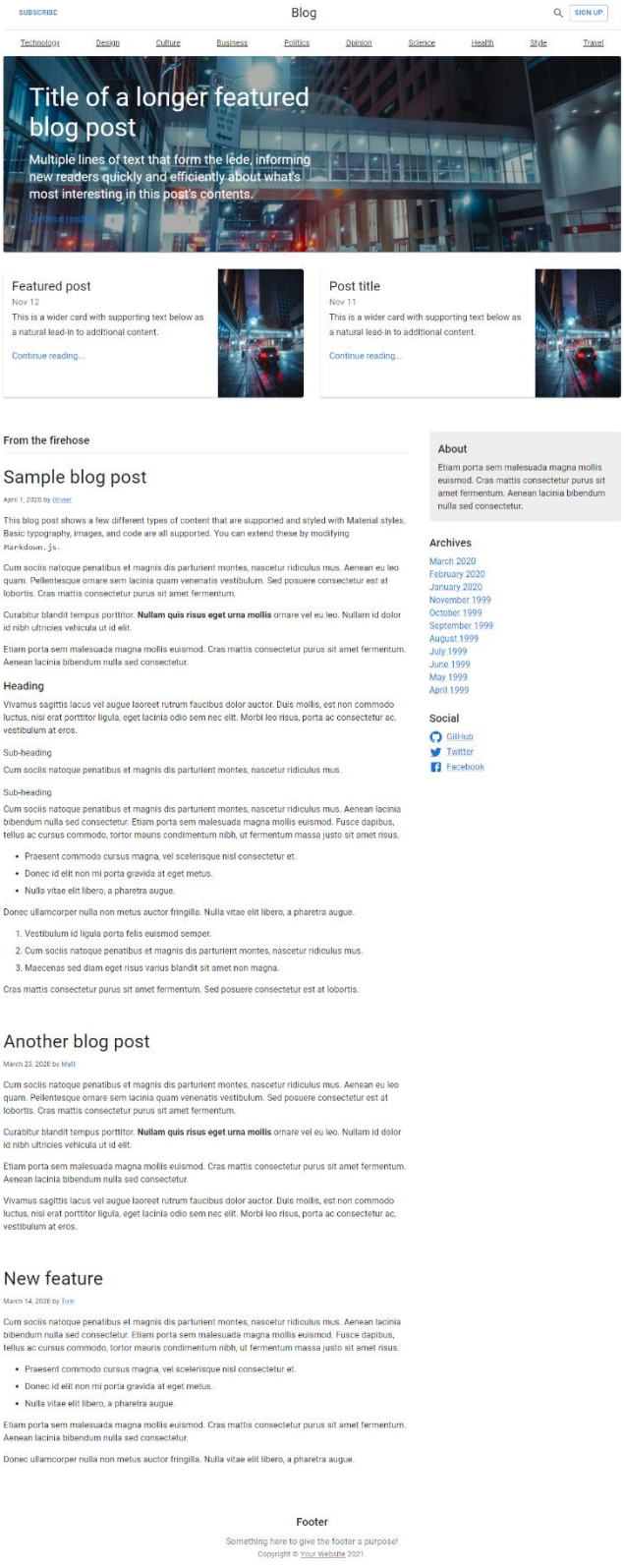


```
const name = 'Josh Perez';
const element = <h1>Hello, {name}</h1>;

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

Fonte: Elaborado a partir da documentação ReactJS (2021)

Outro *framework* que foi utilizado para criação do *frontend* é o Material UI, que é uma biblioteca que disponibiliza vários componentes já prontos, que facilita muito a criação de interfaces, também esta biblioteca segue as diretrizes do Material Design, que é um padrão de Design que auxilia na usabilidade do seu sistema e a experiência do usuário, fazendo com que ele se sinta confortável em ficar na sua aplicação, na Figura 36 podemos ver um blog que utiliza o Material UI.

Figura 36 – Exemplo de aplicação utilizando Material UI



Fonte: Elaborado a partir da documentação Material UI (2021)

No *frontend* para a realizações de requisições utilizou-se a biblioteca Axios, que consiste em uma biblioteca JS que facilita a realização de *requests* para o servidor, possibilita realizar qualquer requisição REST. Na Figura 37 podemos ver um exemplo da utilização desta biblioteca.

Figura 37 – Exemplo de utilização biblioteca Axios

```
const axios = require('axios');

// Make a request for a user with a given ID
axios.get('/user?ID=12345')
  .then(function (response) {
    // handle success
    console.log(response);
  })
  .catch(function (error) {
    // handle error
    console.log(error);
  })
  .then(function () {
    // always executed
  });
```

Fonte: Elaborado a partir da documentação Axios (2021)

Por fim o banco de dados utilizado foi escolhido o postgres, devido a melhor familiaridade com o mesmo, e também por ser gratuito e bem performático, não necessitando que seja trocado o banco caso o projeto demande de escalabilidade.

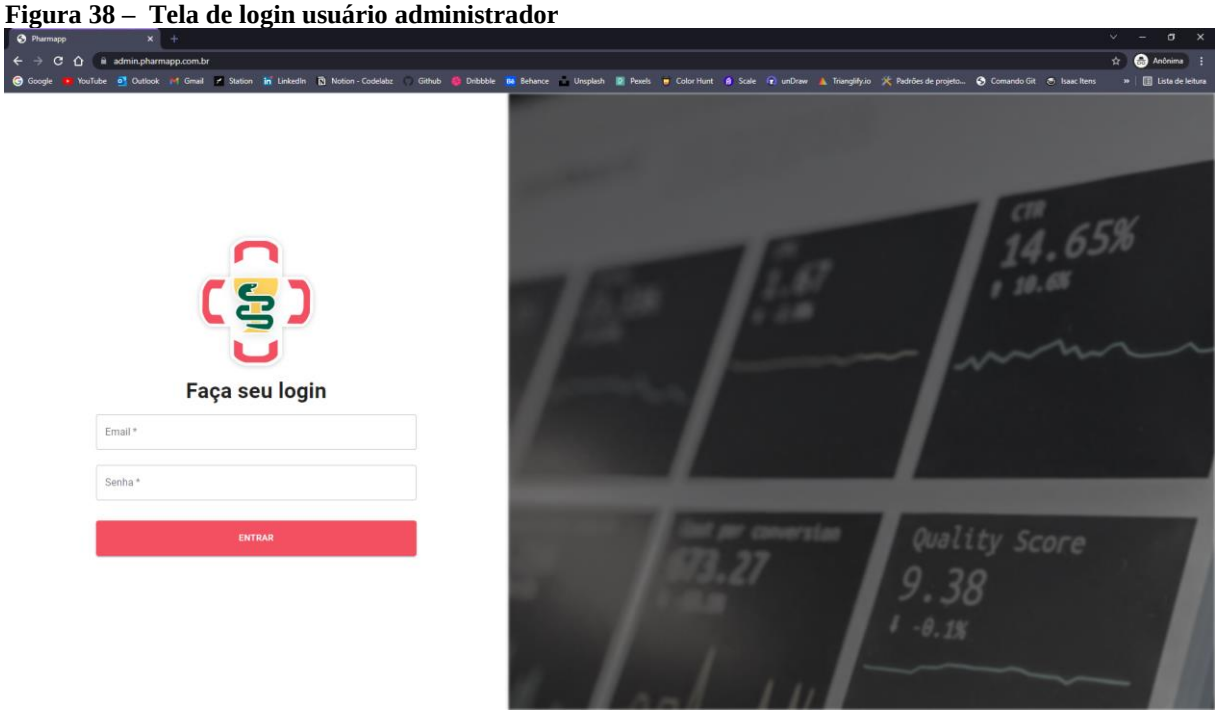
4.6 UTILIZAÇÃO E FUNCIONAMENTO

Nesta sessão será apresentado as telas desenvolvidos, a utilização do sistema e seu funcionamento, primeiramente a explicação das telas do sistema do usuário administrador, após o sistema do usuário cliente e por fim o funcionamento do sistema do usuário farmácia.

4.6.1 Utilização e funcionamento aplicação usuário administrador

Na próxima Figura 38 encontra-se a tela de login do usuário administrador desenvolvida, através do e-mail e senha do usuário é realizado a autenticação do usuário, desenvolvendo o RF01.

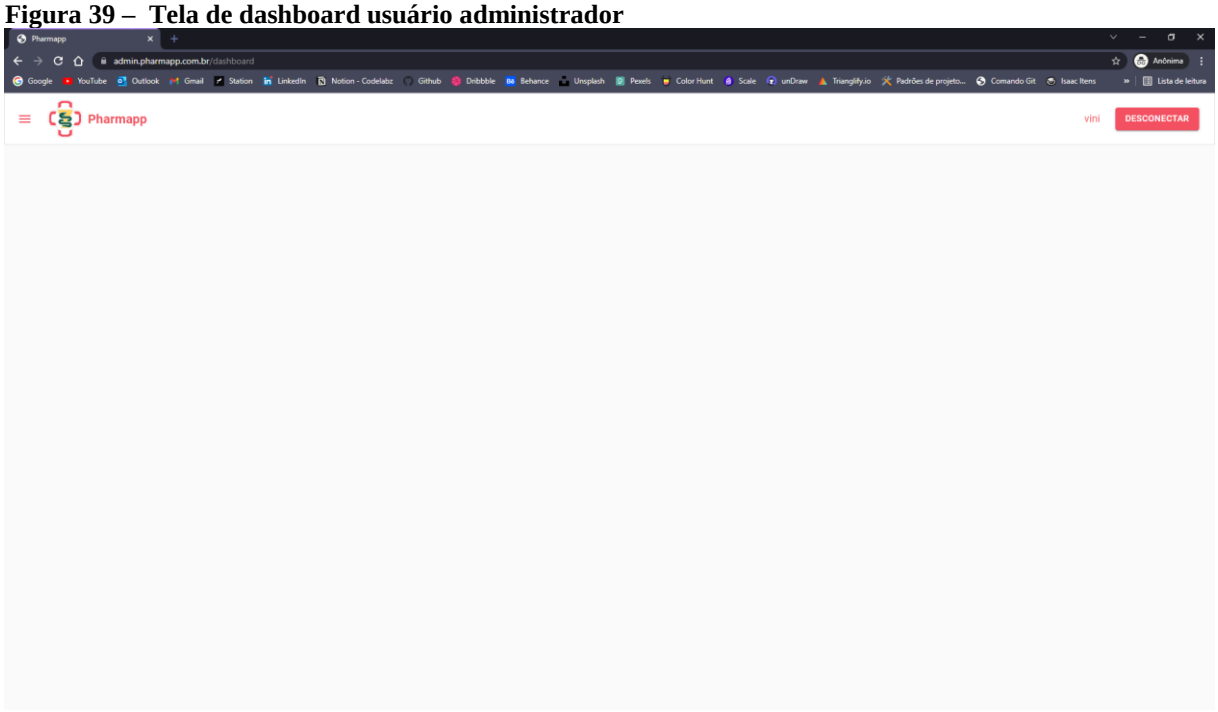
Figura 38 – Tela de login usuário administrador



Fonte: Acervo do autor

Na próxima Figura 39 pode-se ver a tela de dashboard usuário administrador desenvolvida, que futuramente haverá o desenvolvimento do RF37, permitindo trazer informações relevantes das farmácias, como por exemplo faturamento, total de vendas etc.

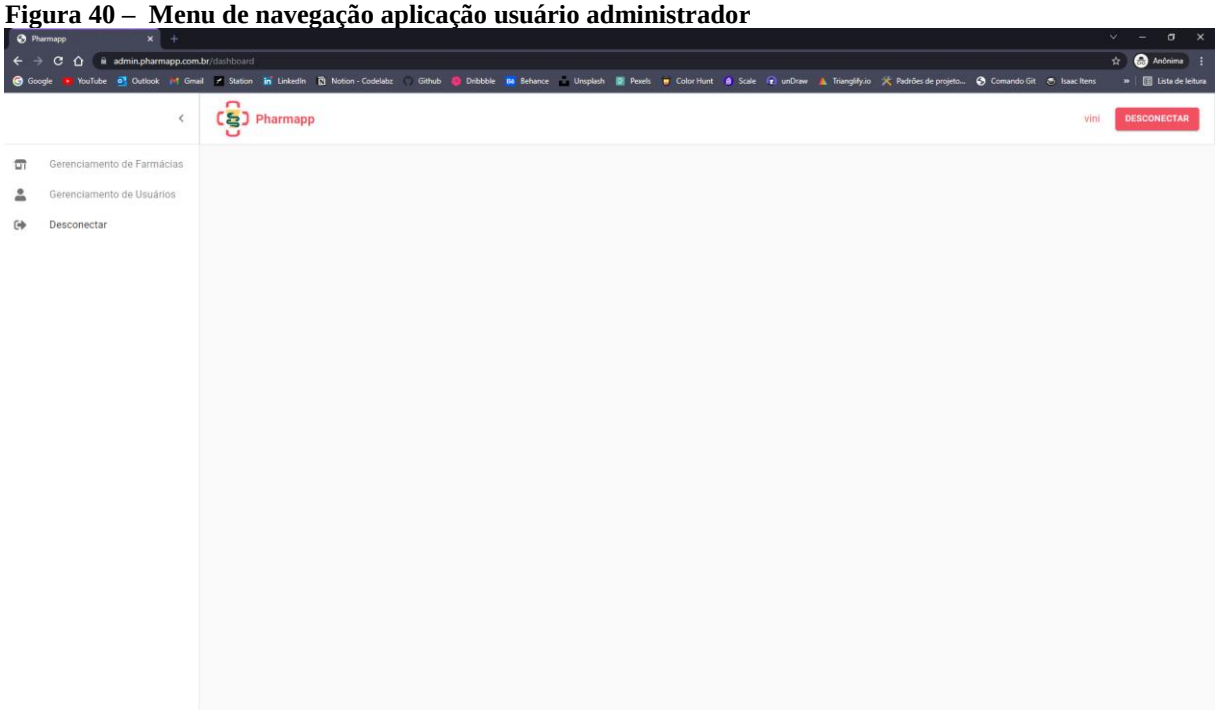
Figura 39 – Tela de dashboard usuário administrador



Fonte: Acervo do autor

Na próxima Figura 40 vemos o menu aberto, por meio dele é possível fazer a navegação para as outras telas da aplicação, como por exemplo ir ao gerenciamento de farmácias e gerenciamento dos usuários farmácias.

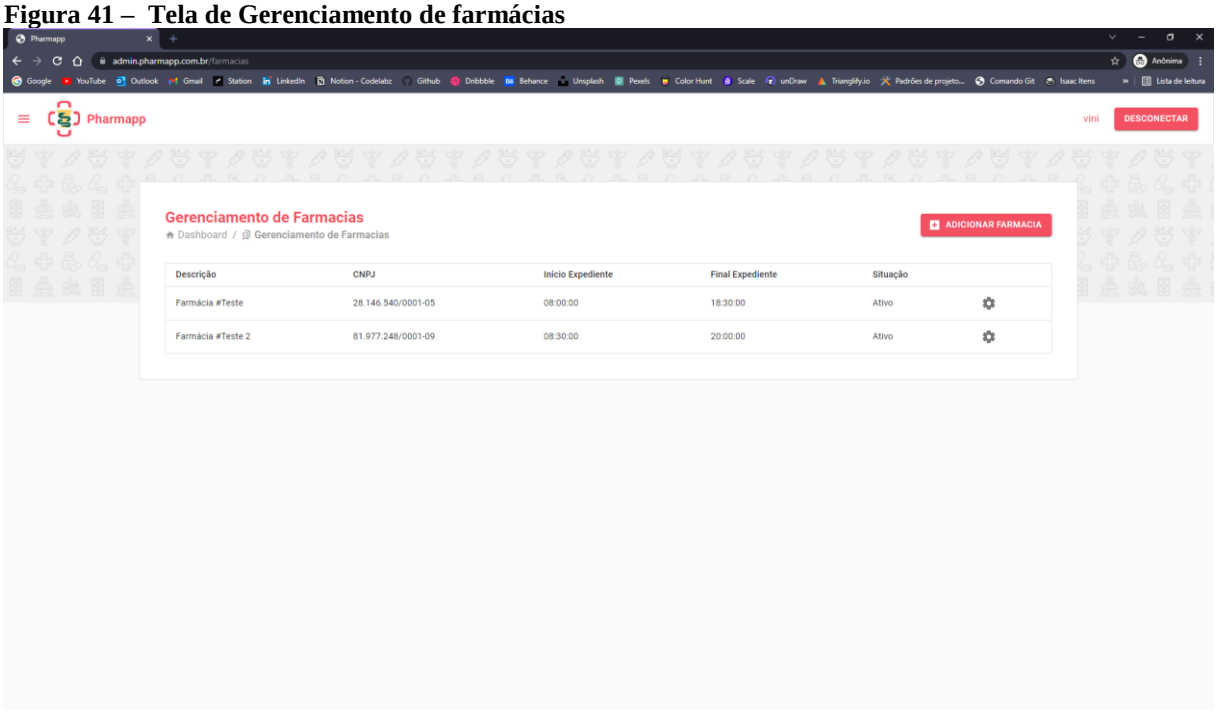
Figura 40 – Menu de navegação aplicação usuário administrador



Fonte: Acervo do autor

A Figura 41 demonstra a tela de gerenciamento de farmácias, por meio dela é possível adicionar novas farmácias, consultar todas as farmácias cadastradas, alterar a situação (Ativo / Desativo) e também permitindo alterar as informações desta farmácia, está tela contempla o desenvolvimento dos requisitos RF02, RF03, RF04, RF05.

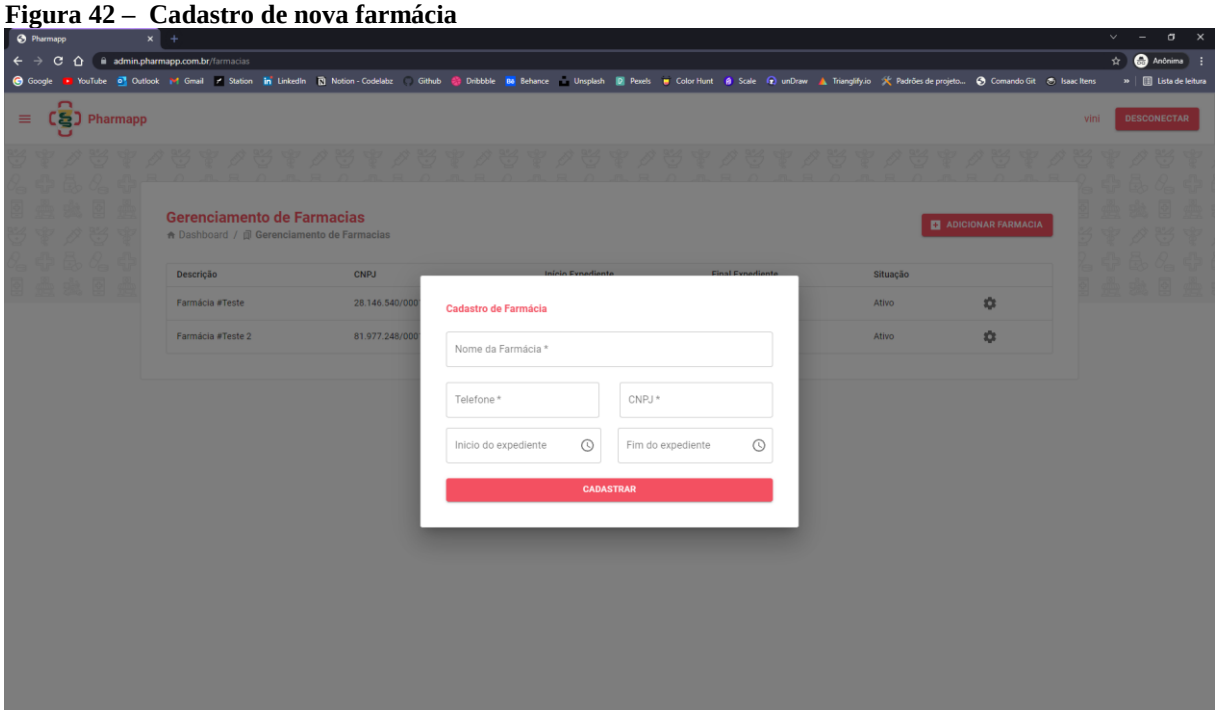
Figura 41 – Tela de Gerenciamento de farmácias



Fonte: Acervo do autor

Na próxima Figura 42 é demonstrada o formulário de criação de farmácia, é necessário informar o nome da farmácia, o telefone da farmácia, o seu CNPJ, horário do início do expediente e o horário do fim do expediente, desenvolvendo o RF02.

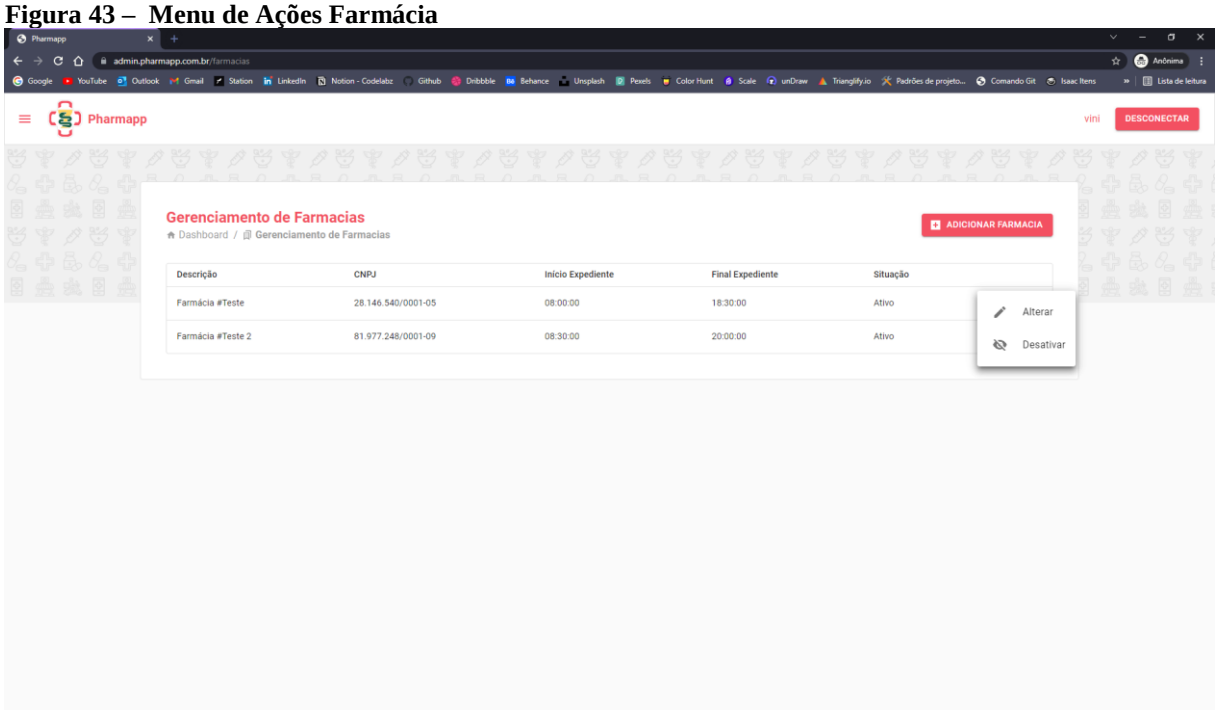
Figura 42 – Cadastro de nova farmácia



Fonte: Acervo do autor

Na próxima Figura 43 é demonstrada o menu de ações, permitindo alterar as informações da farmácia, e também alterar a visibilidade (Ativo / Desativo), desenvolvendo o RF04 e RF05.

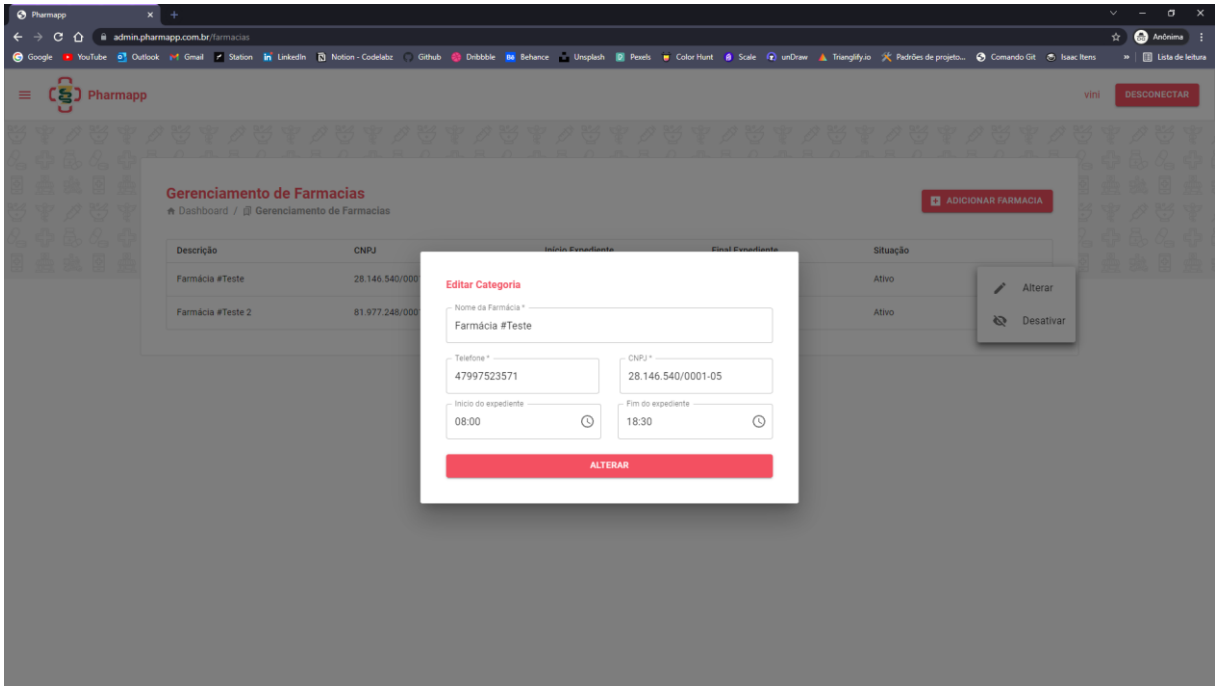
Figura 43 – Menu de Ações Farmácia



Fonte: Acervo do autor

Na próxima Figura 44 é demonstrada o formulário que permite alterar as informações da farmácia selecionada, possibilitando alterar nome da farmácia, telefone, CNPJ, horário início expediente e horário do fim expediente, desenvolvendo o RF05.

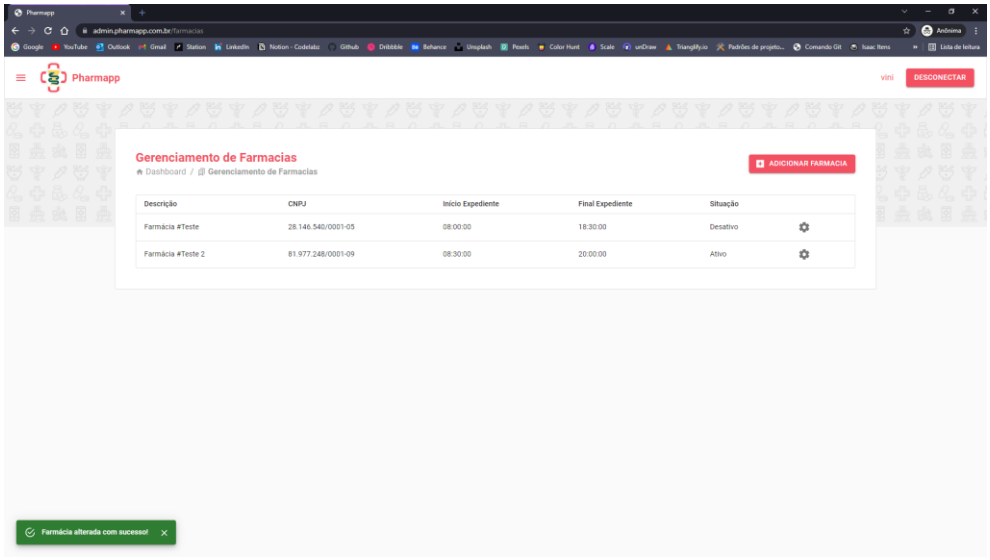
Figura 44 – Alteração da farmácia selecionada



Fonte: Acervo do autor

Na próxima Figura 45 é demonstrada o uso da ação de alterar a visibilidade da farmácia, alterando a situação de Ativo para Desativo, desenvolvendo o RF04.

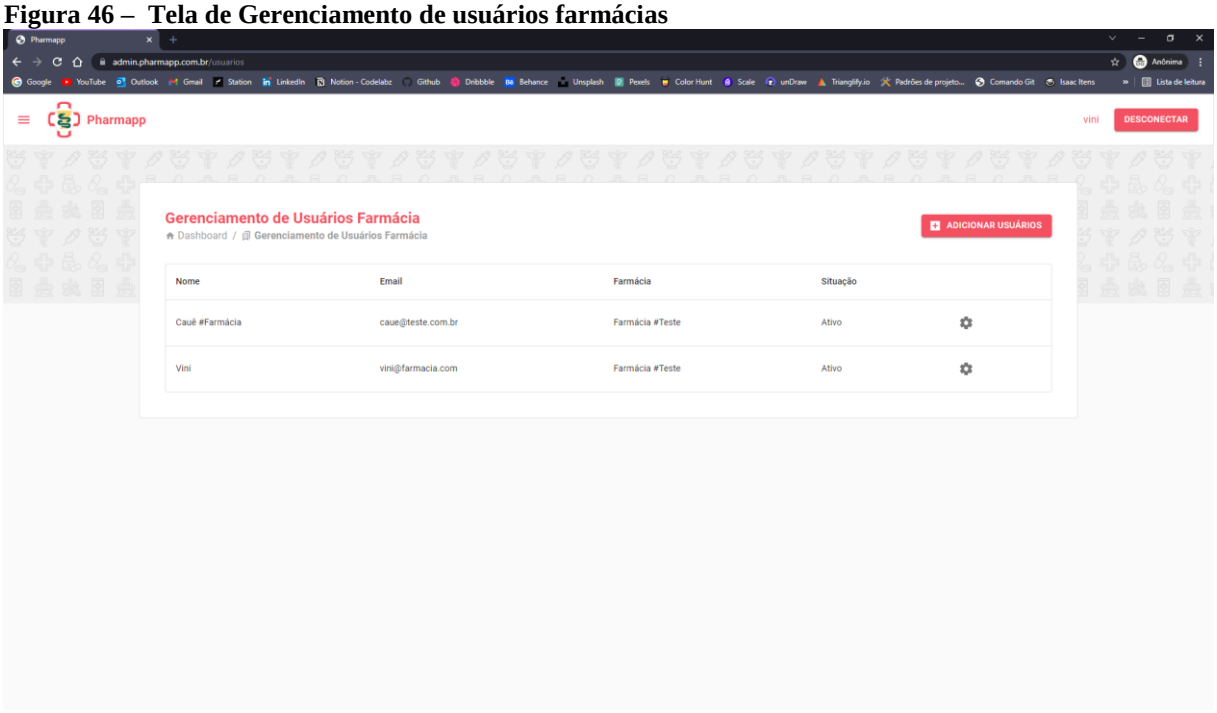
Figura 45 – Alteração da visibilidade farmácia



Fonte: Acervo do autor

A Figura 46 demonstra a tela de gerenciamento de usuários farmácias, por meio dela é possível adicionar novos usuários farmácias, consultar todos os usuários farmácias cadastrados, alterar a situação (Ativo / Desativo) e também permitindo alterar as informações do usuário selecionado, está tela contempla o desenvolvimento dos requisitos RF06, RF07, RF08, RF09.

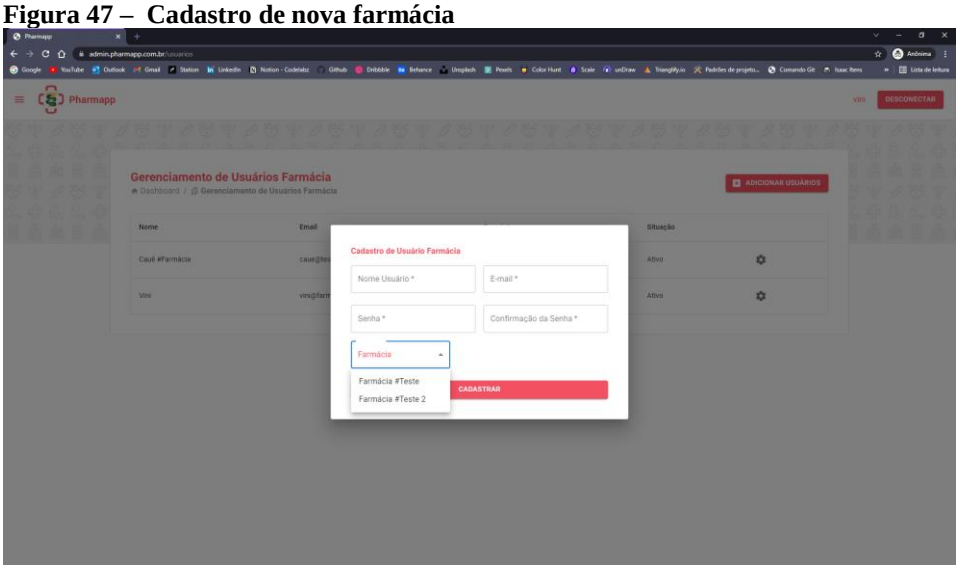
Figura 46 – Tela de Gerenciamento de usuários farmácias



Fonte: Acervo do autor

Na próxima Figura 47 é demonstrada o formulário de criação de usuário farmácia, é necessário informar o nome do usuário, o e-mail, a senha e a confirmação desta senha, desenvolvendo o RF06.

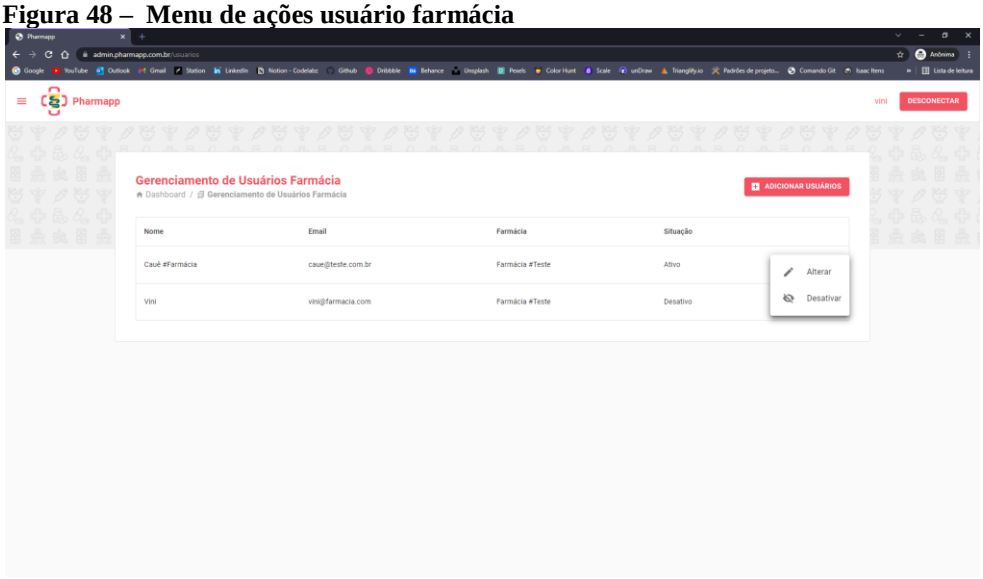
Figura 47 – Cadastro de nova farmácia



Fonte: Acervo do autor

Na próxima Figura 48 é demonstrada o menu de ações, permitindo alterar as informações do usuário, e também alterar a visibilidade (Ativo / Desativo), desenvolvendo o RF08 e RF09.

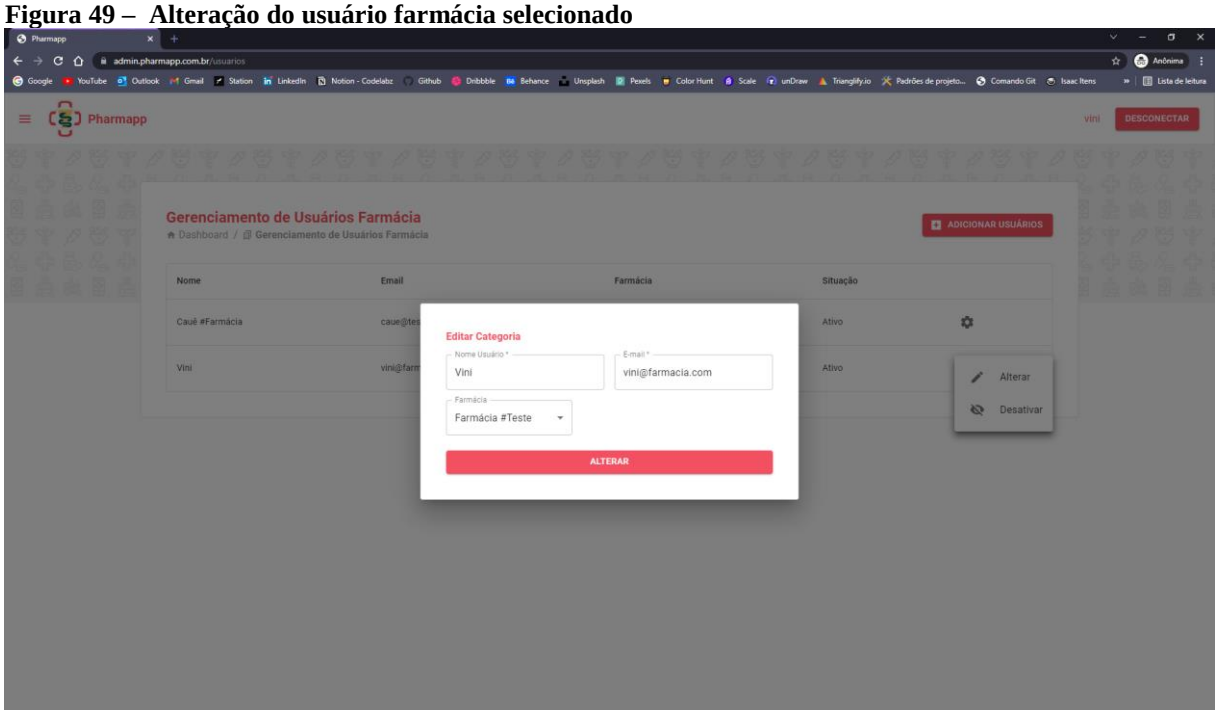
Figura 48 – Menu de ações usuário farmácia



Fonte: Acervo do autor

Na próxima Figura 49 é demonstrada o formulário que permite alterar as informações do usuário selecionado, possibilitando alterar o nome do usuário, e-mail e relacionamento da farmácia, desenvolvendo o RF09.

Figura 49 – Alteração do usuário farmácia selecionado



Fonte: Acervo do autor

Na próxima Figura 50 é demonstrada o uso da ação de alterar a visibilidade do usuário, alterando a situação de Ativo para Desativo, desenvolvendo o RF08.

Figura 50 – Alteração da visibilidade usuário farmácia

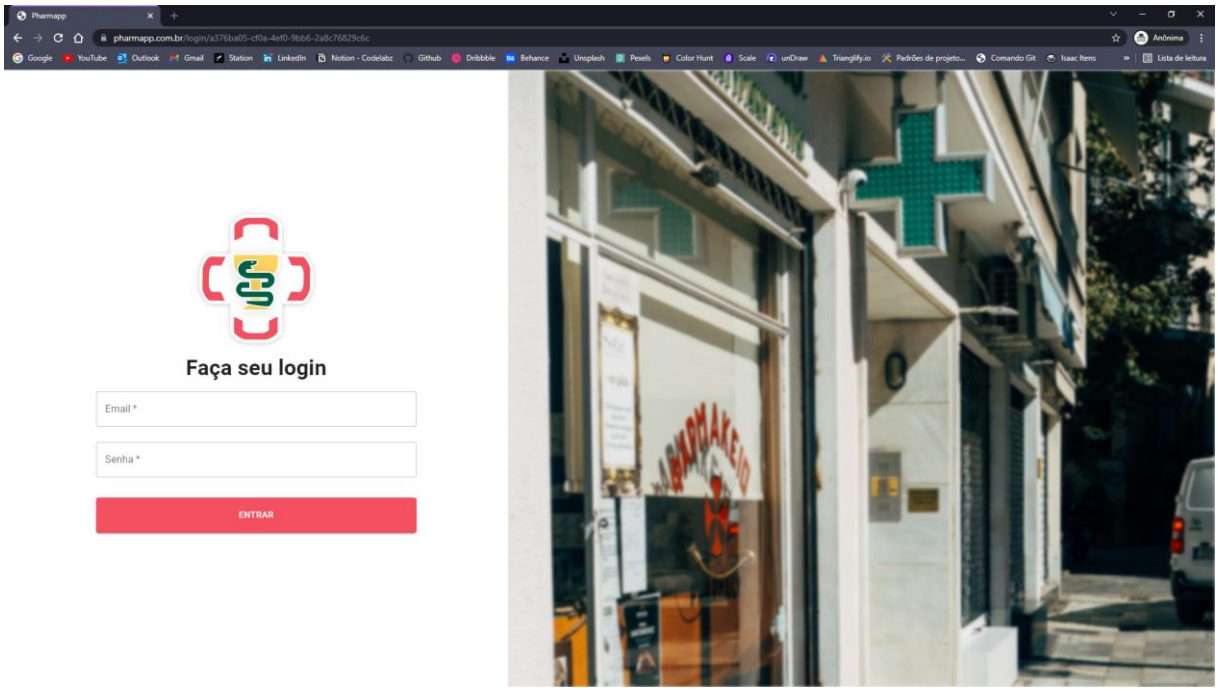
Fonte: Acervo do autor

Na próxima sessão vai ser abordado as telas e a utilização da aplicação do usuário farmácia, este usuário é utilizado para gerenciamento de categorias da farmácia, gerenciamento dos produtos e gerenciamento de pedidos, está farmácia e este usuário são os mesmo que o usuário administrador cadastrou.

4.6.2 Utilização e funcionamento aplicação usuário farmácia

Na próxima Figura 51 encontra-se a tela de login do usuário farmácia, através do e-mail, senha do usuário e na url passar o UUID da farmácia que quer fazer o login, é realizado a autenticação do usuário, desenvolvendo o RF10.

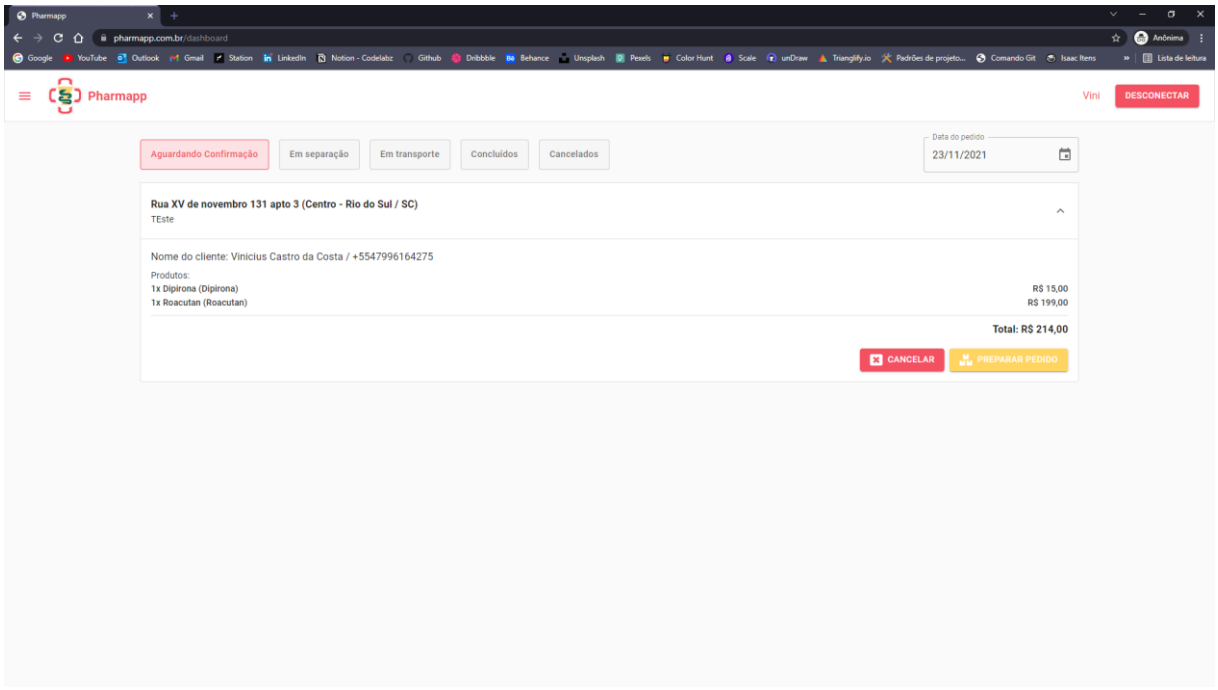
Figura 51 – Login da aplicação usuário farmácia



Fonte: Acervo do autor

Na próxima Figura 52 pode-se ver a tela de dashboard do usuário farmácia, trazendo todos os pedidos referente ao dia selecionado e também o tipo do pedido, por exemplo se o pedido está aguardando confirmação, já foi para separação, está em transporte, se foi concluído ou se já foi cancelado, também cada pedido possui a ação de cancelar o pedido ou mandar para próxima etapa até chegar na etapa de concluído ou cancelado, com isso concluindo o requisito RF19 e RF20, também ficou como desenvolvimento futuro é a implementação de socket nesta tela permitindo realizar a mudanças de etapas em tempo real, desenvolvendo o RF32.

Figura 52– Dashboard usuário farmácia pedidos em aguardo de confirmação



Fonte: Acervo do autor

Na próxima Figura 53 pode-se ver a tela de dashboard do usuário farmácia com o filtro de “Em separação”, assim trazendo todos os pedidos que estão nesta situação e no dia selecionado, os pedidos trazidos possui a ação de cancelar pedido ou mandar para a próxima (Enviar Pedido) etapa no caso mandar para transporte, ainda continua desenvolvendo o RF19 e RF20.

Figura 53 – Dashboard usuário farmácia pedidos em separação

Fonte: Acervo do autor

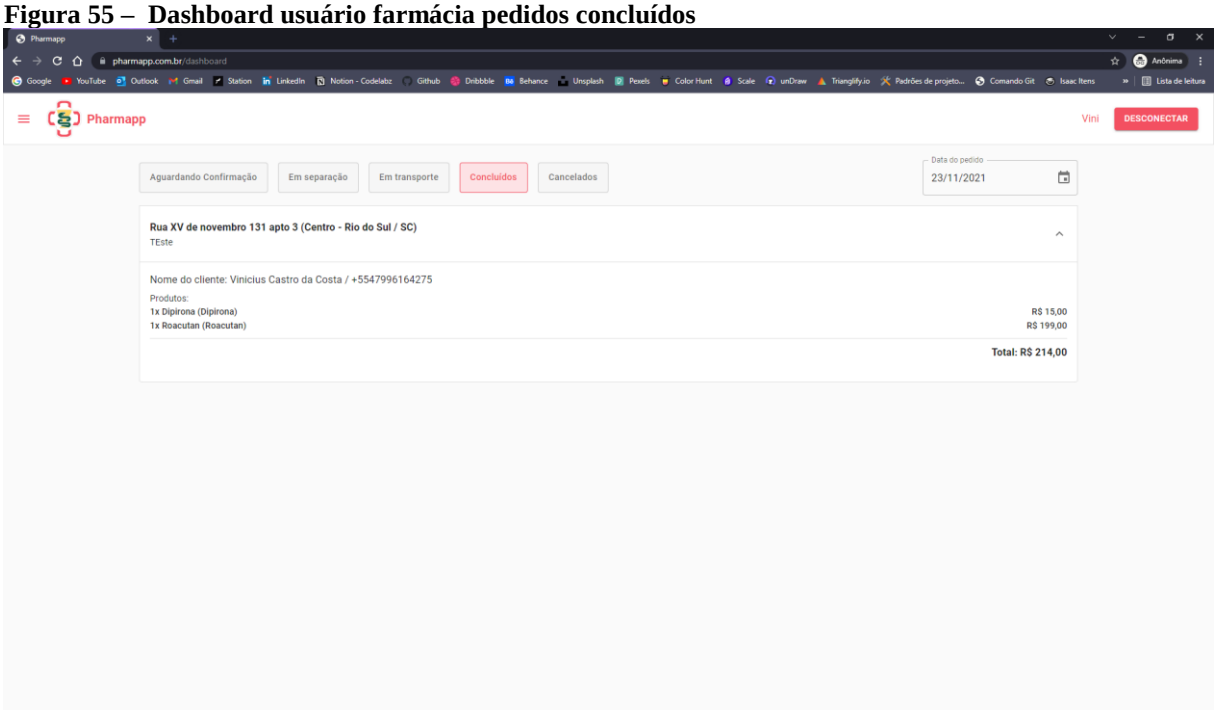
Na próxima Figura 54 pode-se ver a tela de dashboard do usuário farmácia com o filtro de “Em transporte”, assim trazendo todos os pedidos que estão nesta situação e no dia selecionado, os pedidos trazidos possui a ação de cancelar pedido ou mandar para a próxima etapa (Concluir pedido) no caso mandar para transporte, ainda continua desenvolvendo o RF19 e RF20.

Figura 54 – Dashboard usuário farmácia pedidos em transporte

Fonte: Acervo do autor

Na próxima Figura 55 pode-se ver a tela de dashboard do usuário farmácia com o filtro de “Concluídos”, assim trazendo todos os pedidos que estão nesta situação e no dia selecionado, os pedidos trazidos desta vez não possuem as ações para alterar a situação do pedido, somente desenvolvendo o RF19.

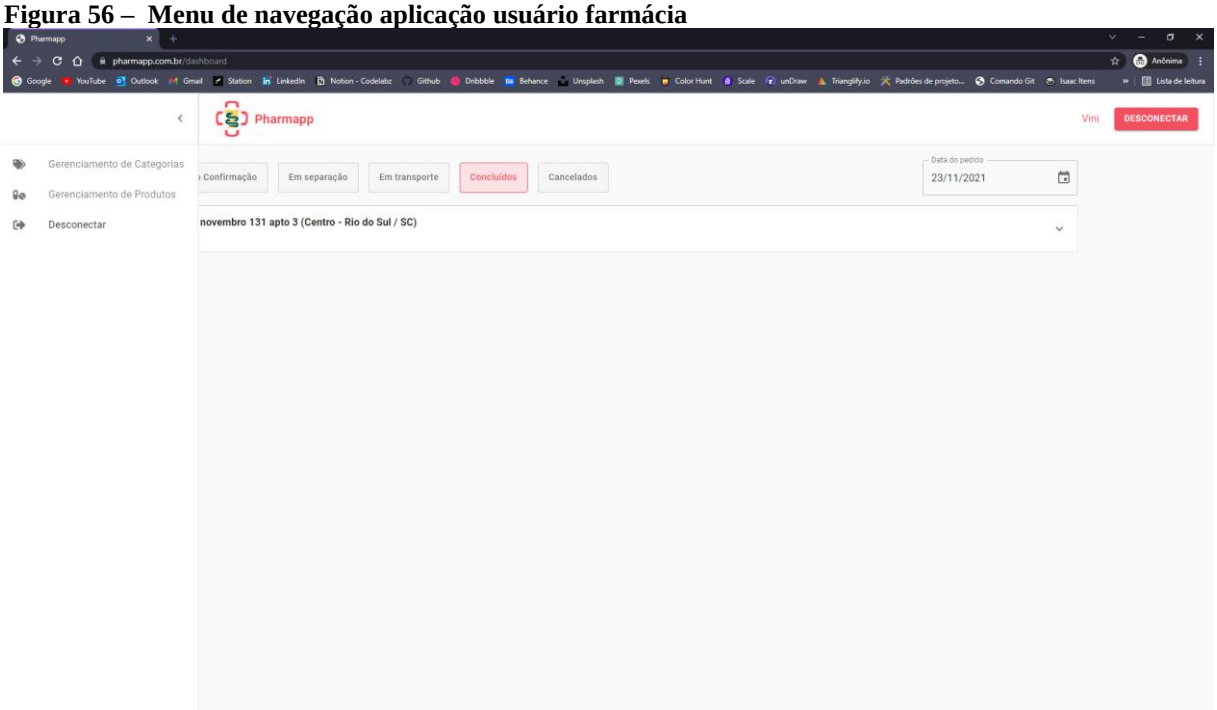
Figura 55 – Dashboard usuário farmácia pedidos concluídos



Fonte: Acervo do autor

Na próxima Figura 56 vemos o menu aberto, por meio dele é possível fazer a navegação para as outras telas da aplicação, como por exemplo ir ao gerenciamento de categorias e gerenciamento de produtos.

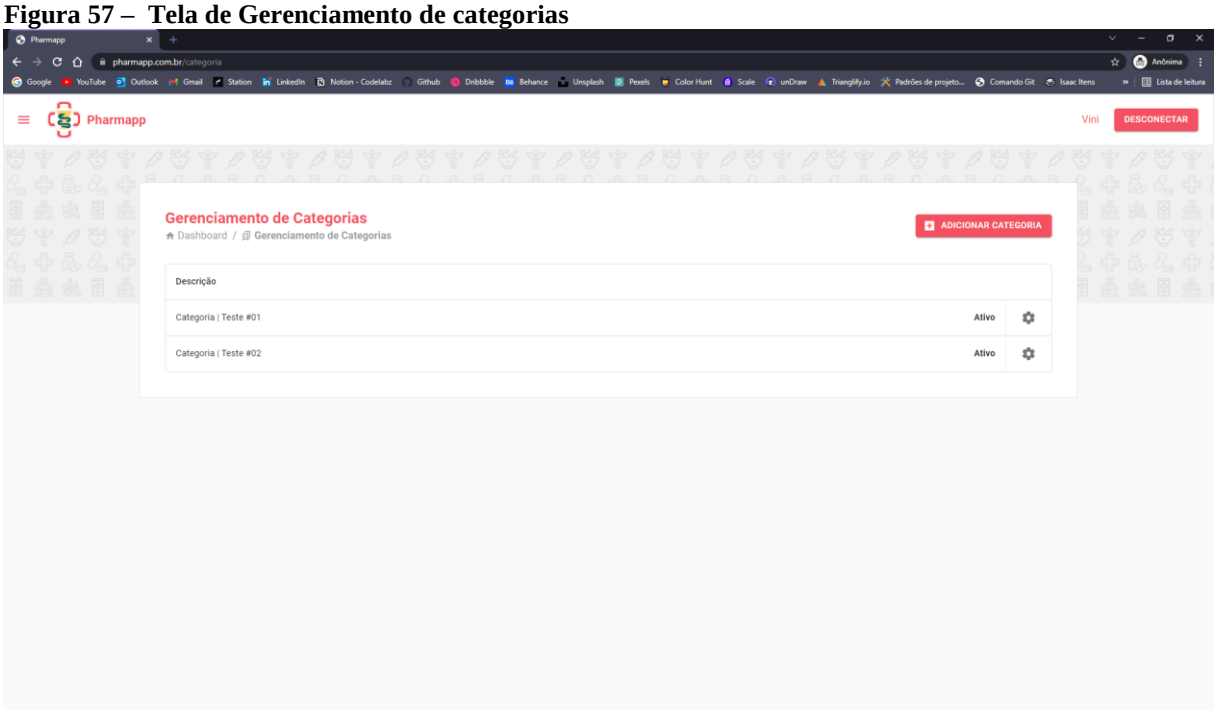
Figura 56 – Menu de navegação aplicação usuário farmácia



Fonte: Acervo do autor

A Figura 57 demonstra a tela de gerenciamento de categorias, por meio dela é possível adicionar novas categorias da farmácia, consultar todas as categorias cadastradas, alterar a situação (Ativo / Desativo) e também permitindo alterar as informações da categoria selecionada, está tela contempla o desenvolvimento dos requisitos RF11, RF12, RF13, RF14.

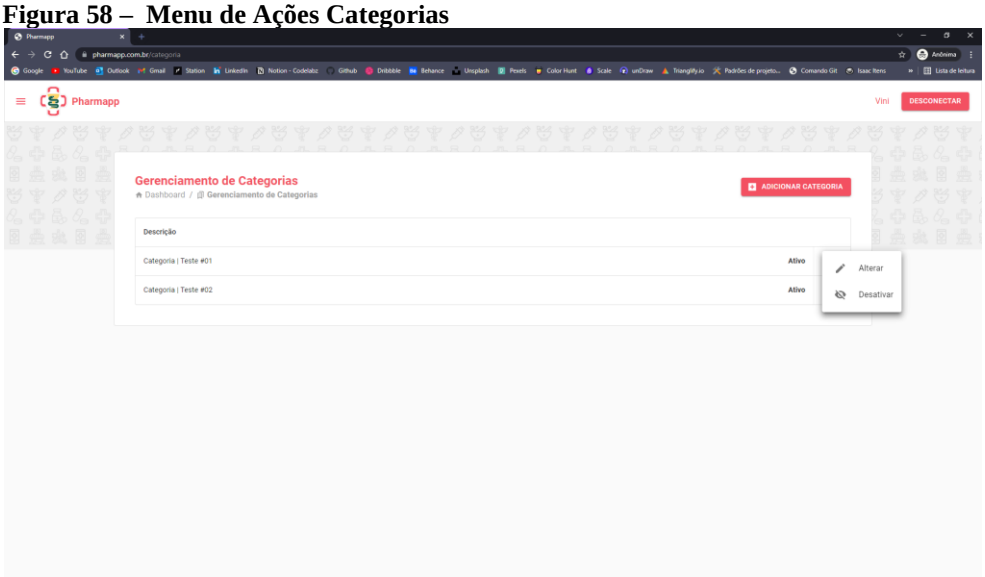
Figura 57 – Tela de Gerenciamento de categorias



Fonte: Acervo do autor

Na próxima Figura 58 é demonstrada o menu de ações, permitindo alterar as informações da categoria, e também alterar a visibilidade (Ativo / Desativo), desenvolvendo o RF13 e RF14.

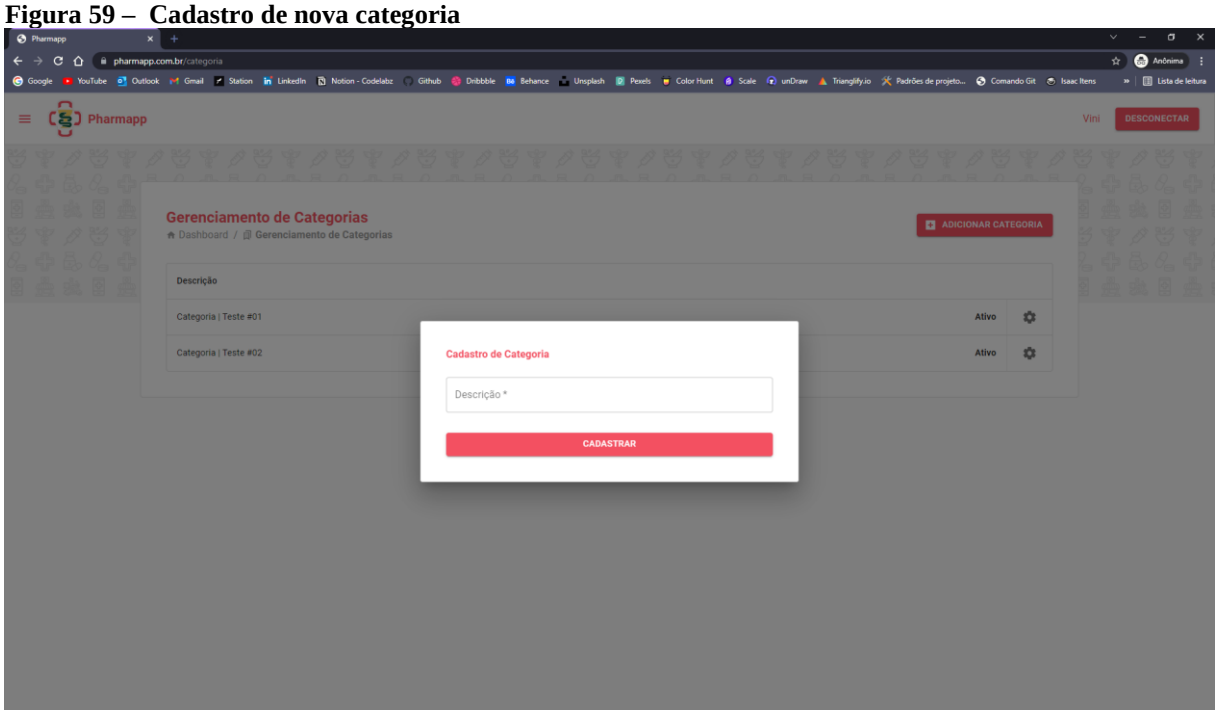
Figura 58 – Menu de Ações Categorias



Fonte: Acervo do autor

Na próxima Figura 59 é demonstrada o formulário de criação de categoria, é necessário informar somente a descrição, desenvolvendo o RF11.

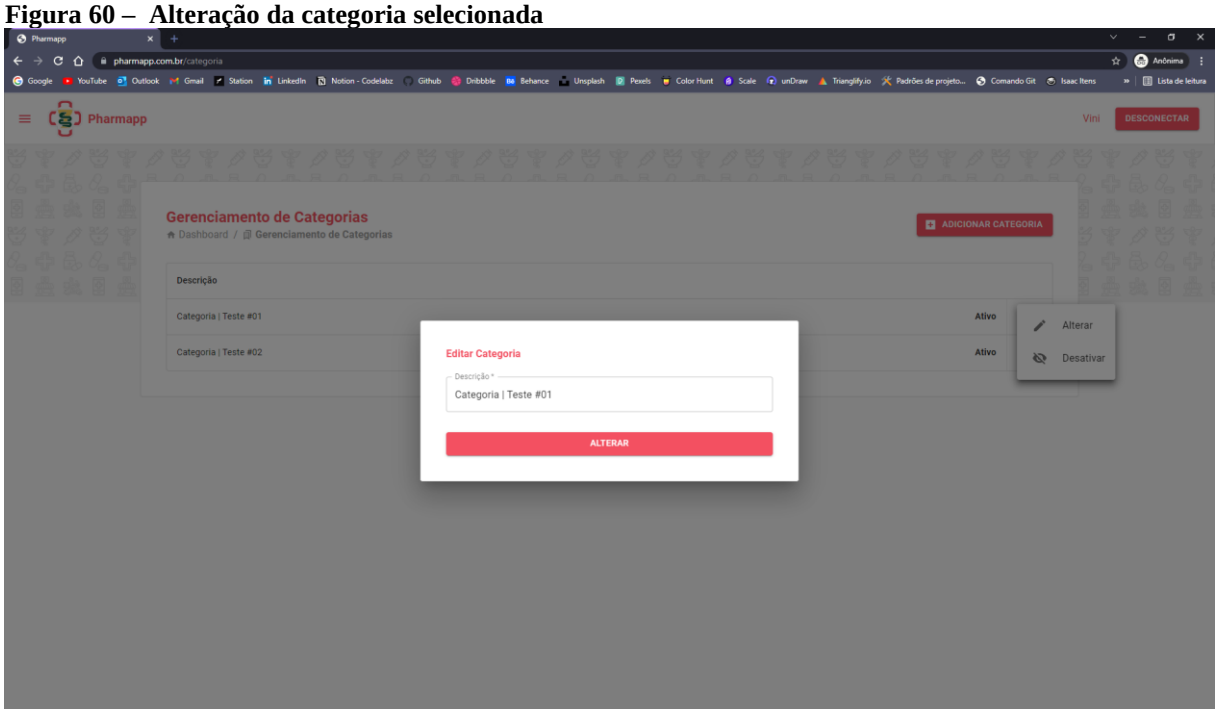
Figura 59 – Cadastro de nova categoria



Fonte: Acervo do autor

Na próxima Figura 60 é demonstrada o formulário que permite alterar as informações da categoria selecionada, possibilitando alterar a descrição, desenvolvendo o RF14.

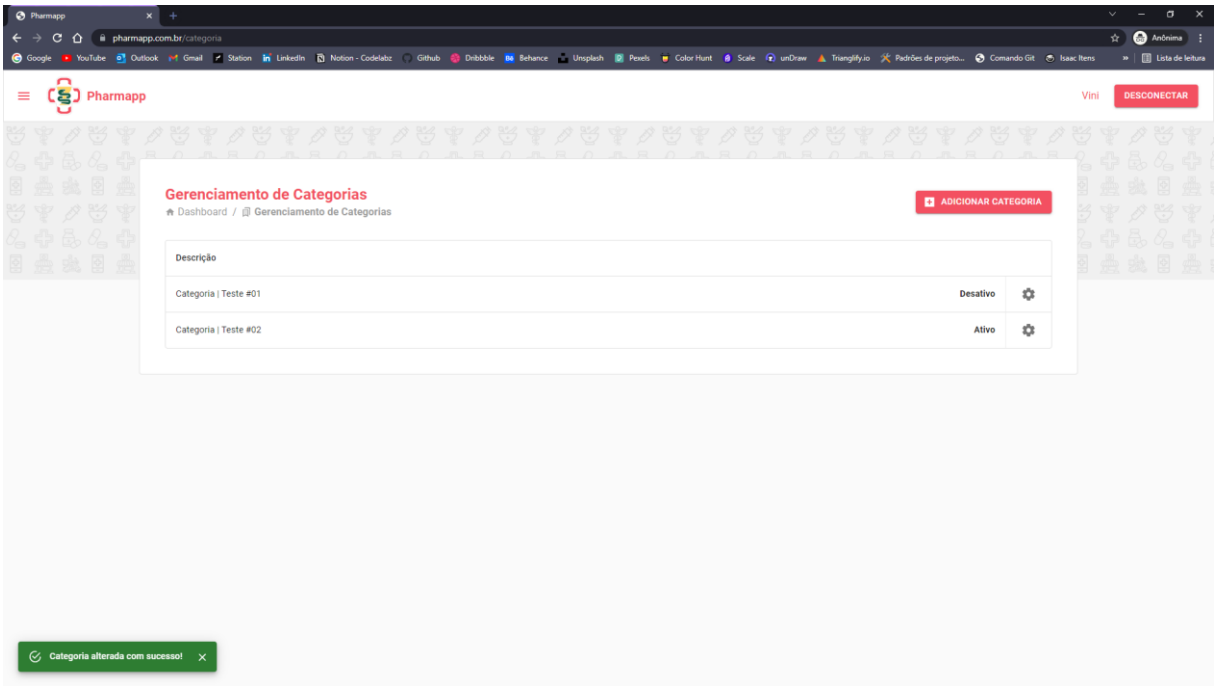
Figura 60 – Alteração da categoria selecionada



Fonte: Acervo do autor

Na próxima Figura 61 é demonstrada o uso da ação de alterar a visibilidade da categoria, alterando a situação de Ativo para Desativo, desenvolvendo o RF08.

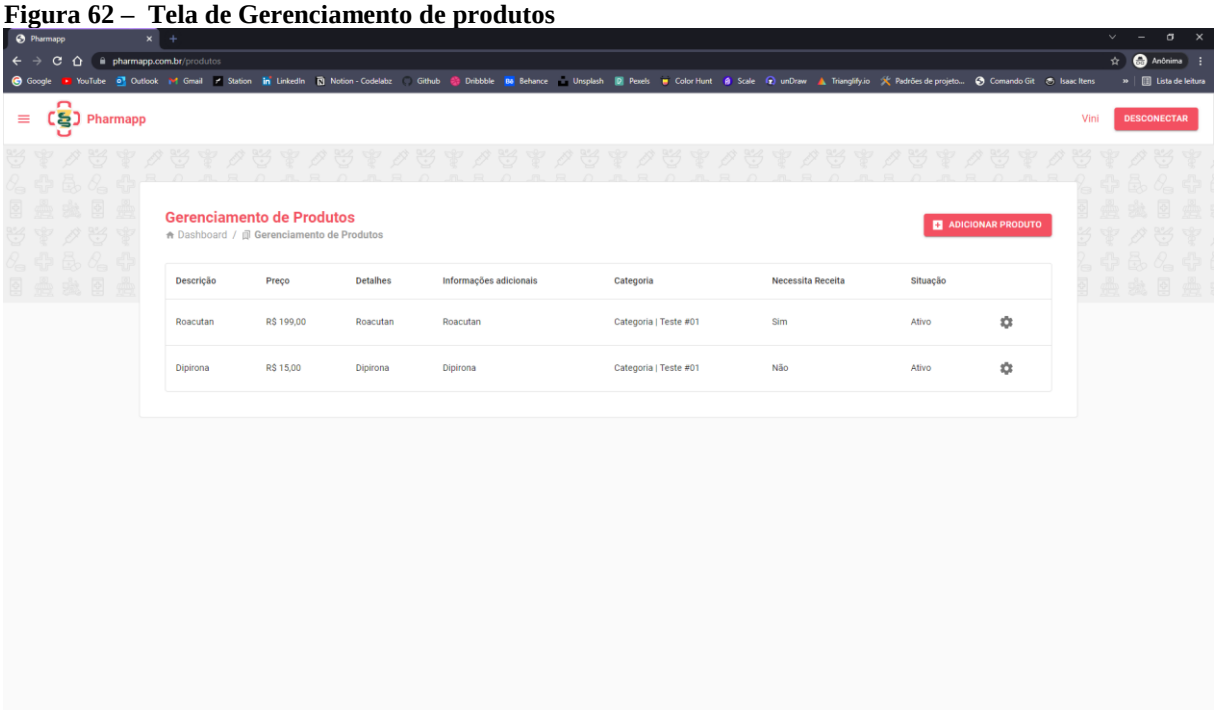
Figura 61– Alteração da visibilidade farmácia



Fonte: Acervo do autor

A Figura 62 demonstra a tela de gerenciamento de produtos, por meio dela é possível adicionar novos produtos da farmácia, consultar todos os produtos cadastrados, alterar a situação (Ativo / Desativo) e também permitindo alterar as informações do produto selecionado, está tela contempla o desenvolvimento dos requisitos RF15, RF16, RF17, RF18.

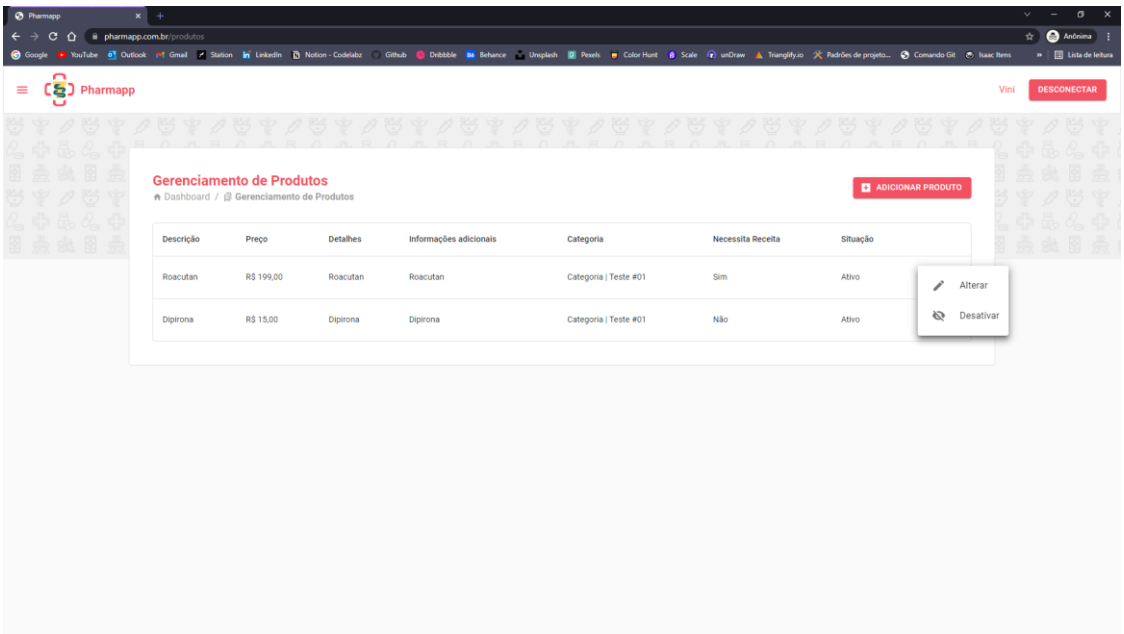
Figura 62 – Tela de Gerenciamento de produtos



Fonte: Acervo do autor

Na próxima Figura 63 é demonstrada o uso da ação de alterar a visibilidade do produto, alterando a situação de Ativo para Desativo, desenvolvendo o RF17.

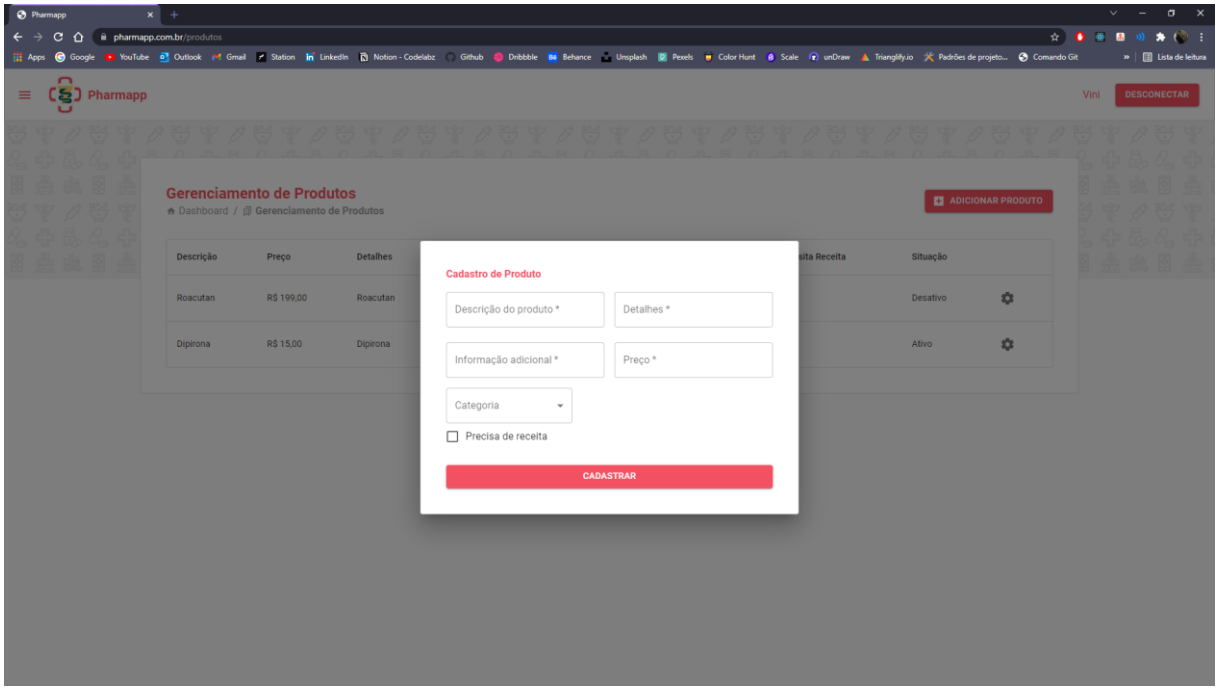
Figura 63 – Alteração da visibilidade do produto



Fonte: Acervo do autor

Na próxima Figura 64 é demonstrada o formulário de criação de produto, é necessário informar a descrição do produto, o detalhe do produto, informações adicionais, o preço, selecionar uma categoria para relacionar e se precisa de receita ou não, desenvolvendo o RF15.

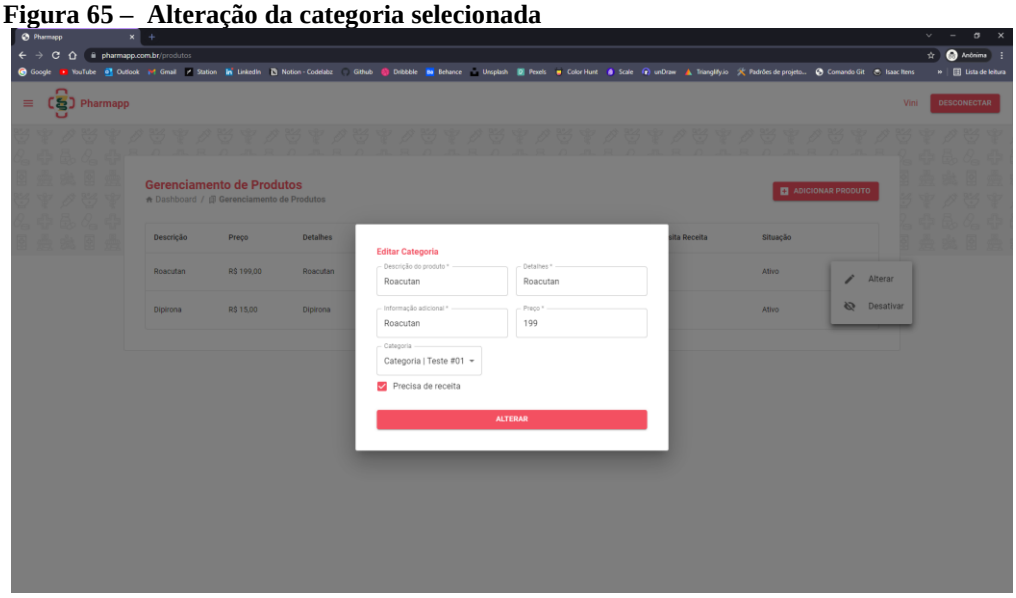
Figura 64 – Cadastro de nova categoria



Fonte: Acervo do autor

Na próxima Figura 65 é demonstrada o formulário que permite alterar as informações da categoria selecionada, possibilitando alterar a descrição do produto, detalhes, informação adicional, preço, categoria e se necessita de receita, desenvolvendo o RF18.

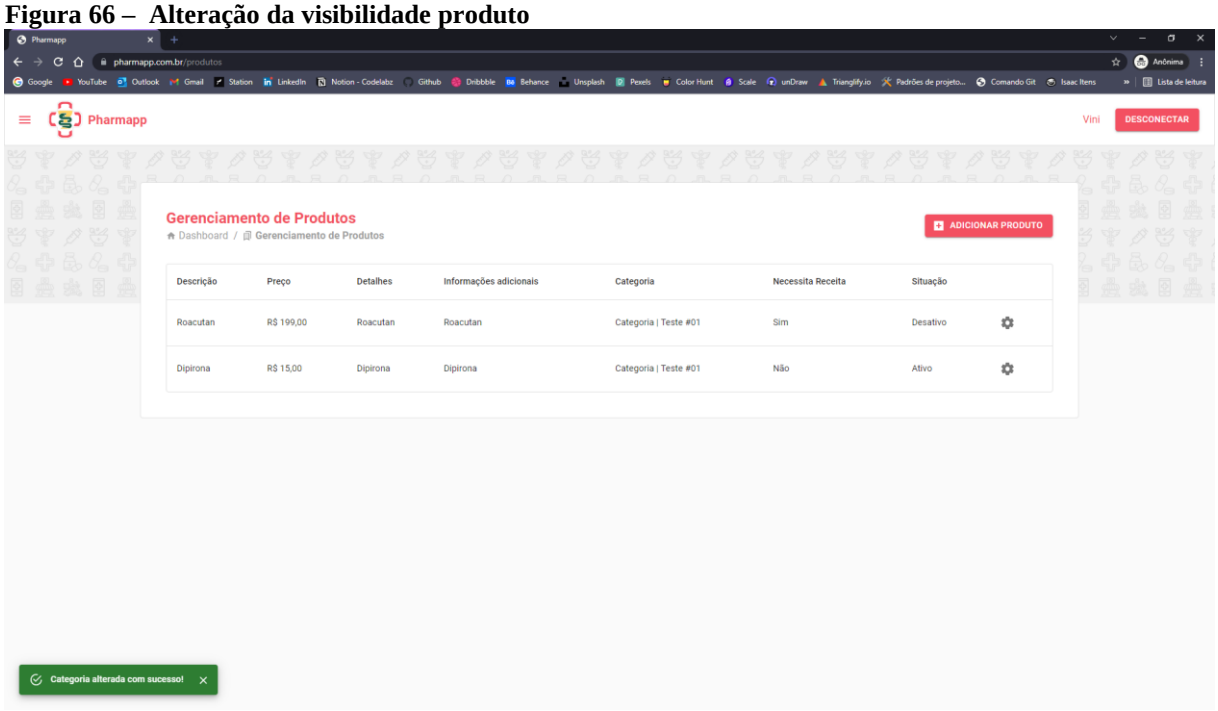
Figura 65 – Alteração da categoria selecionada



Fonte: Acervo do autor

Na próxima Figura 66 é demonstrada o uso da ação de alterar a visibilidade do produto, alterando a situação de Ativo para Desativo, desenvolvendo o RF17.

Figura 66 – Alteração da visibilidade produto



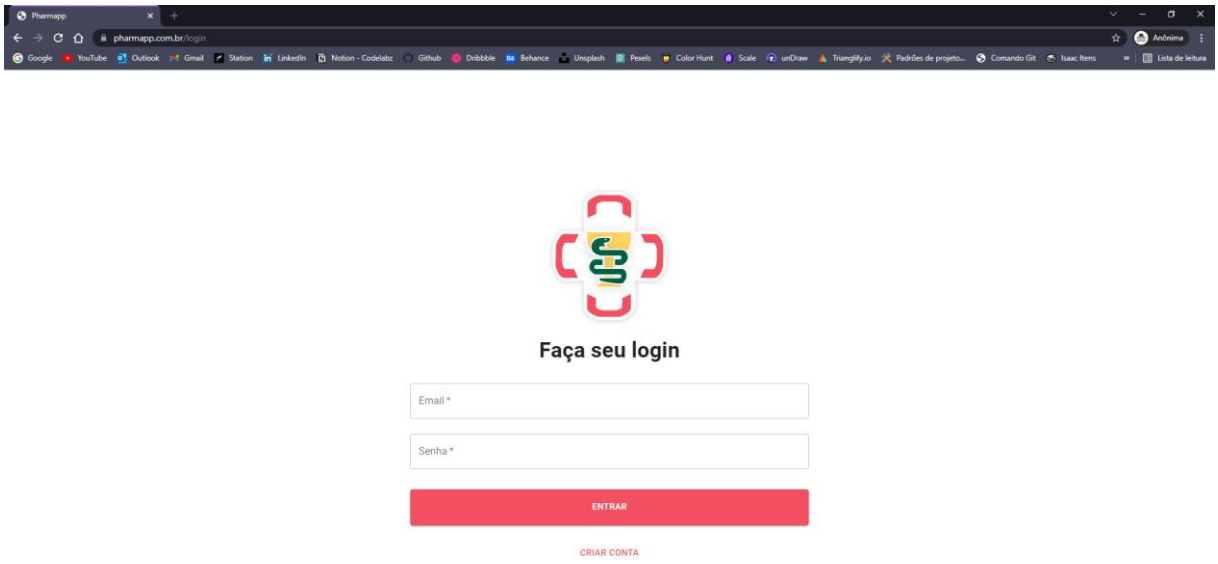
Fonte: Acervo do autor

Na próxima sessão vai ser abordado as telas e a utilização da aplicação do usuário cliente, este usuário é utilizado para conseguir ver as farmácias, visualizá-las, cadastrar seus endereços, realizar pedidos com base nos produtos cadastrados pela farmácia.

4.6.3 Utilização e funcionamento aplicação usuário cliente

Na próxima Figura 67 encontra-se a tela de login do usuário cliente, através do e-mail, senha do usuário, também possuindo um botão de navegação para a tela de cadastro do usuário, é realizado a autenticação do usuário, desenvolvendo o RF21.

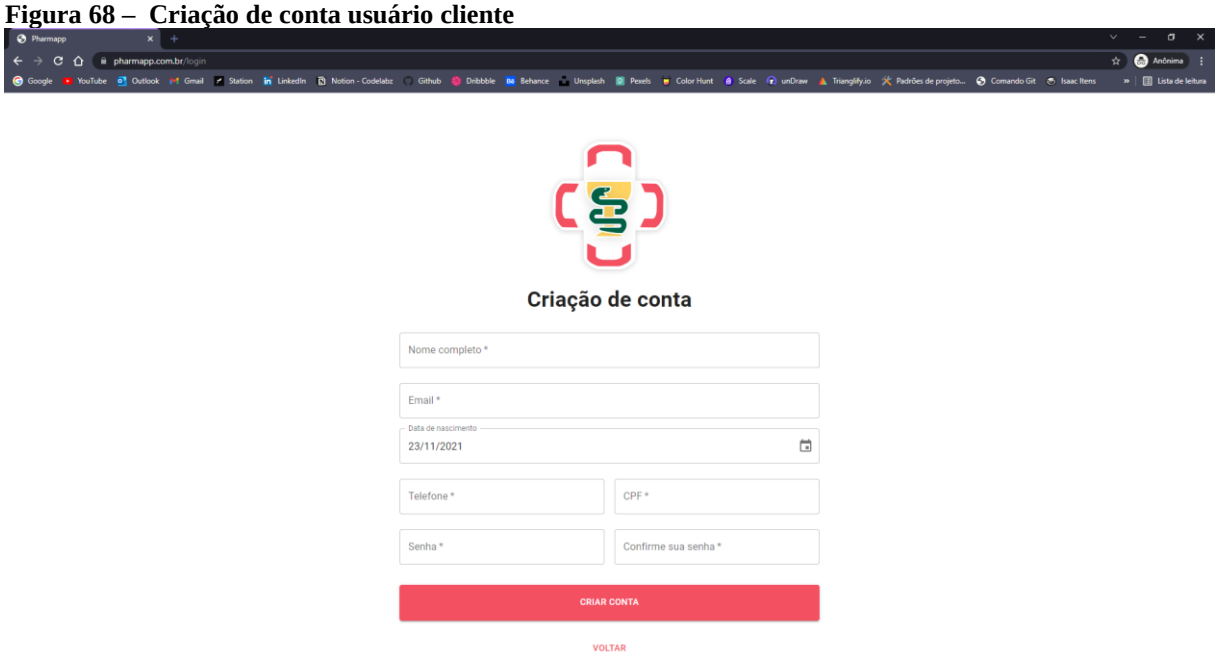
Figura 67 – Login da aplicação usuário cliente



Fonte: Acervo do autor

Na próxima Figura 68 é demonstrada o formulário que permite alterar as informações da categoria selecionada, possibilitando alterar a descrição do produto, detalhes, informação adicional, preço, categoria e se necessita de receita, desenvolvendo o RF18.

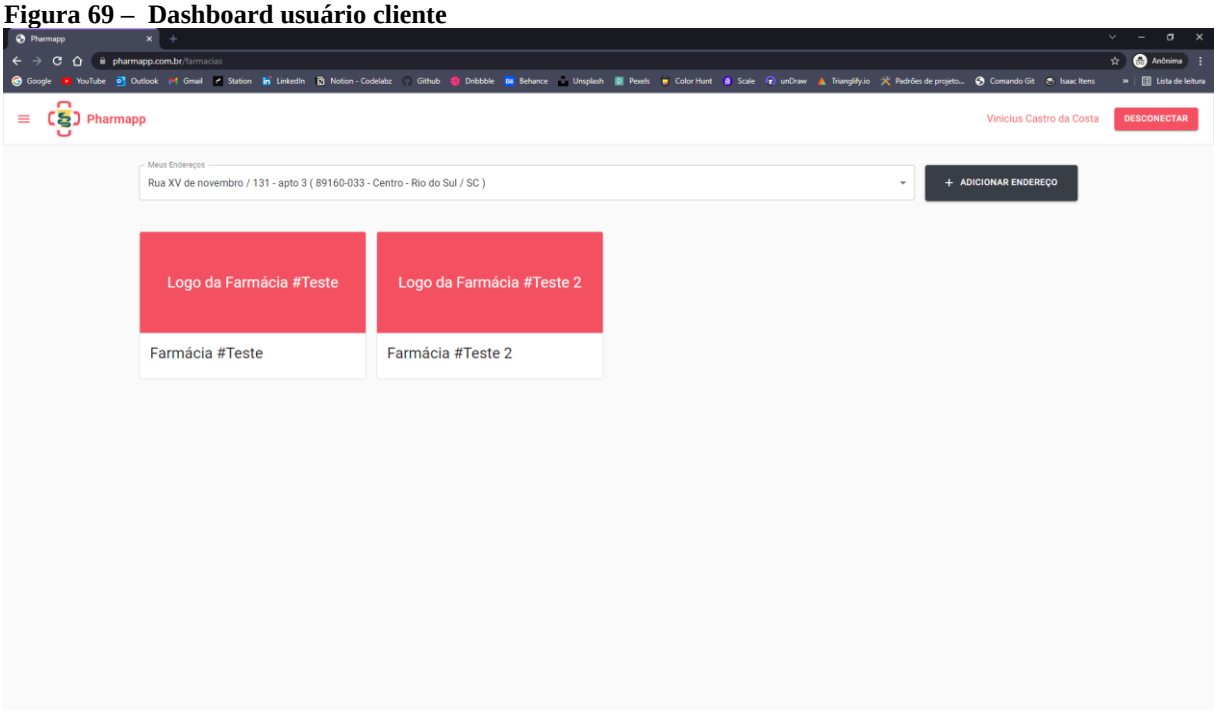
Figura 68 – Criação de conta usuário cliente



Fonte: Acervo do autor

Na Figura 69 demonstra a tela que você será redirecionado após o login do usuário, está tela lista todas as farmácias, também a listagem de todos os endereços do usuário e por fim, permite a criação de novos usuários, com esta tela foi desenvolvido o RF23, RF24, RF25, RF26.

Figura 69 – Dashboard usuário cliente



Fonte: Acervo do autor

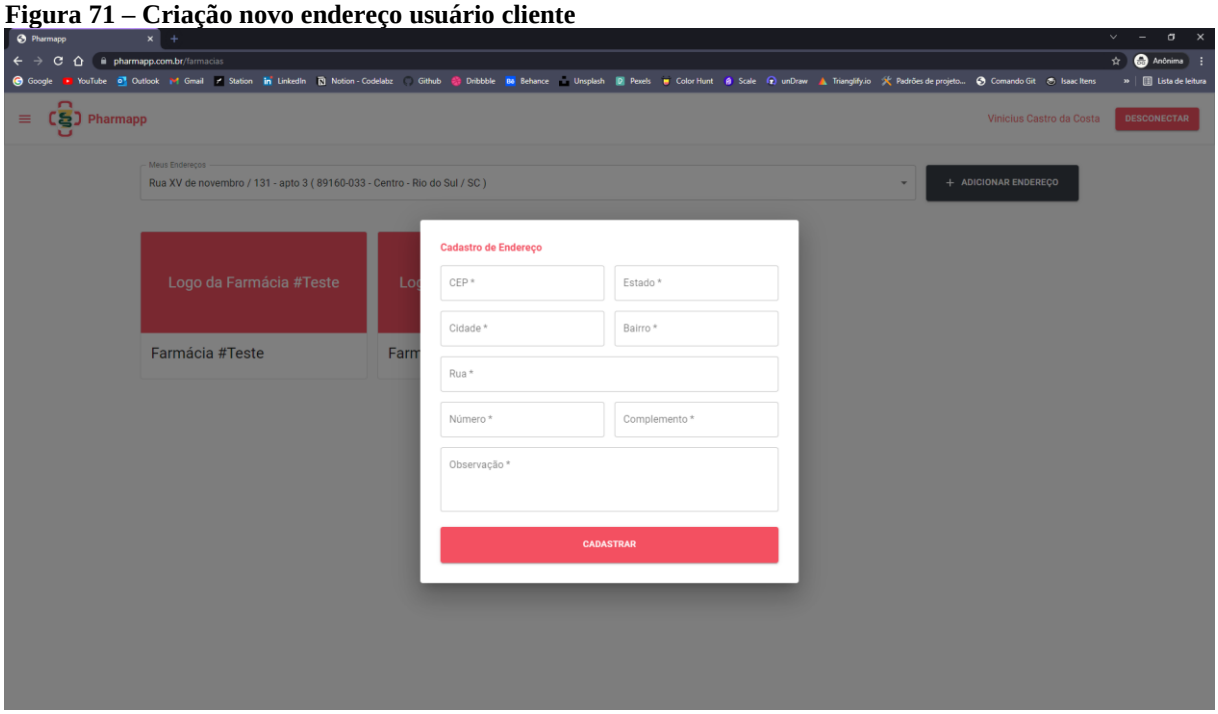
Na próxima Figura 70 vemos o menu aberto, por meio dele é possível fazer a navegação para as outras telas da aplicação, como por exemplo ir aos pedidos do usuário ou as farmácias que estão cadastradas.

Figura 70 – Menu de navegação aplicação usuário cliente

Fonte: Acervo do autor

Na próxima Figura 71 é demonstrada o formulário que permite criar endereços com base no usuário logado, esta tela ao preencher o CEP é feita uma requisição para o BrasilAPI, para buscar as informações com base no CEP, com base nesta tela é desenvolvido o RF24 e RF25.

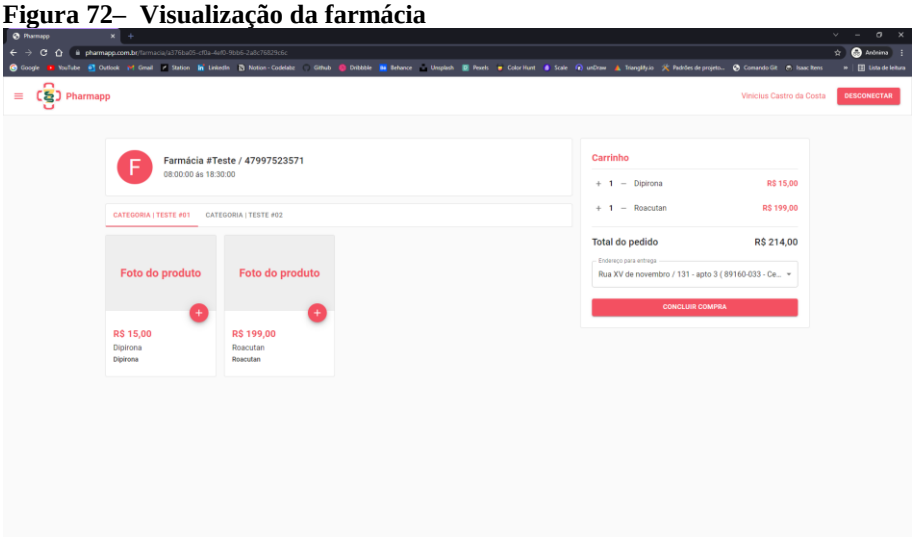
Figura 71 – Criação novo endereço usuário cliente



Fonte: Acervo do autor

Na próxima Figura 72 é demonstrada a visualização de farmácia, trazendo todas as categorias cadastradas e os produtos conforme a categoria selecionada, também permitindo adicionar os produtos no carrinho e por fim realizar o pedido, com base nesta tela é desenvolvida o RF27, RF28, RF29 e RF30.

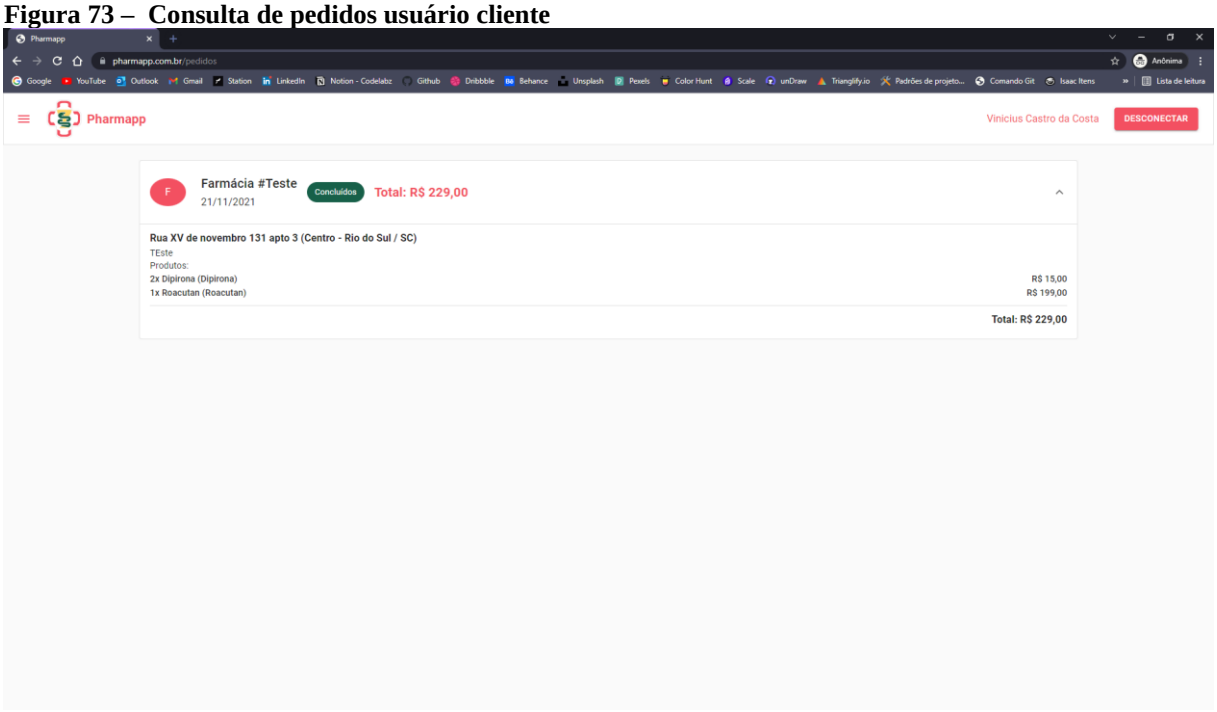
Figura 72– Visualização da farmácia



Fonte: Acervo do autor

Na próxima Figura 73 é demonstrada a tela de pedidos do usuário cliente, por meio desta tela é possível consultar todos os pedidos feitos por aquele usuário, podendo ver todos os produtos relacionados com um determinado pedido, está tela desenvolve o RF31.

Figura 73 – Consulta de pedidos usuário cliente



Fonte: Acervo do autor

No próximo capitulo será abordado as conclusões finais sobre o presente trabalho, apresentando se os objetivos foram concluídos e que tipos de conclusões pode-se tirar a partir do trabalho realizado.

5. CONCLUSÃO

Com o desenvolvimento do presente trabalho permitiu demonstrar que é possível fazer um sistema que torne a realização de pedidos farmacêuticos de forma online,tornado um jeito muito simples e organizado, permitindo que as farmácias além dos benefícios já citados também consigam ampliar o seu canal de vendas, saindo de vendas somente físicas e entrando na venda digital também.

Na etapa de desenvolvimento dos requisitos tornou-se possível o entendimento do negócio que se tratava, pode-se organizar a ordem de desenvolvimento e também o que cada aplicação deveria possuir para que o sistema visto de forma total se mantivesse um sistema solido e completo.

A criação dos protótipos da aplicação permitiu que houvesse um maior entendimento de como o sistema ia se parecer quando estivesse completo, permitindo entender o fluxo geral do sistema, a navegação entre telas como seria realizada, como os componentes deveriam ser distribuídos nas telas para que as aplicações fossem as mais intuitivas possível, fazendo que usuários leigos consigam entender como utilizar o sistema sem precisar ficar quebrando a cabeça, para achar as funcionalidades disponíveis.

Por causa do negócio ser basicamente um N-N relacionando vários clientes com várias farmácias os relacionamentos associativos foram imprescindíveis para o desenvolvimento do projeto do banco de dados, possibilitando que farmácias possam possuir diversas categorias e essas categorias diversos produtos relacionados e por parte do cliente ele pode elaborar diversos pedidos entre N farmácias, adicionando diversos produtos em seu carrinho.

Após entendimento de todo negócio, protótipos prontos, requisitos elaborados, analisando todo escopo do projeto e a necessita de futuramente este sistema se tornar um aplicativo a escolha da linguagem JavaScript foi realizada pois com a tecnologia React Native que é uma das principais bibliotecas utilizadas para criação de aplicativos na atualidade e se assemelha muito com a biblioteca React mudando muito pouco a sintaxe da linguagem assim podendo reaproveitar muitas coisas já desenvolvidas, também para que a linguagem de *backend* fosse a mesma do *frontend* foi escolhido o NodeJS, estes critérios foram utilizados para escolha da linguagem.

5.1 SUGESTÃO DE TRABALHO FUTURO

Conforme anteriormente citado uma ideia para o presente trabalho é que o Pharmapp se tornar um aplicativo para celulares, para maior praticidade para o usuário final, permitindo com

que se torne cada vez mais rápido e pratico a realização de pedidos farmacêuticos através da internet, evitando a dependência de abrir um navegador para poder realizar suas compras.

Conforme citado nos requisitos opcionais também ficou para desenvolvimento futuro rotinas de inteligência, permitindo com que o sistema crie um perfil do usuário e indique farmácias que possuem produtos que ele costuma comprar que estão com um preço melhor, e pelo outro lado também ao finalizar um compra, acima do botão de concluir compra mostre uma lista de produtos daquela farmácia que o usuário tem mais chance de adicionar no carrinho, também com base no perfil do usuário, assim possibilitando com que o usuário coloque cada vez mais produtos em seu carrinho, aumentando a quantidade de produtos vendidos por farmácia.

E por fim a última ideia de desenvolvimento futuro é a adição de rotinas que usem socket, para que a utilização do sistema se torne mais fluido, fazendo com que as rotinas escolhidas atualizem em tempo real, por exemplo: o dashboard da farmácia atualize sozinho quando um novo pedido for realizado ou também quando a situação de um pedido for alterada o cliente receba um aviso da atual situação em tempo real.

REFERÊNCIAS

ALVES, William Pereira. **Projetos de sistemas Web:** conceitos, estruturas, criação de banco de dados e ferramentas de desenvolvimento. São Paulo: Erica, 2019. *ebook*

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

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

EXPRESS. **Documentação**. Disponível em: https://pt-br.reactjs.org/docs/getting-started.html. Acesso em: 21 jun. 2021.

FLANAGAN, David. **JavaScript**: o guia definitivo. 6. Porto Alegre: Bookman, 2014. *ebook*

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

MEDICINASA. **Mercado farmacêutico cresce 13,6% impulsionado por associativismo**. Disponível em: https://medicinasa.com.br/mercado-farmaceutico-cresce/. Acesso em: 21 jun. 2021.

MDN. **Documentação**. Disponível em: https://developer.mozilla.org/en-US/. Acesso em: 21 jun. 2021.

MONTEIRO, Allan. **Material-UI agora é MUI** Disponível em: https://pingback.com/desenvbr/material-ui-agora-e-mui Acesso em: 19 nov. 2021.

NODEJS. **Documentação**. Disponível em: https://nodejs.org/pt-br/docs/. Acesso em: 21 jun. 2021.

REACTJS. **Documentação**. Disponível em: https://expressjs.com/pt-br/. Acesso em: 19 nov. 2021.

VAREJOS.A. **Gastos com delivery aumentam em 187% no Brasil**. Disponível em: https://cndl.org.br/varejosa/gastos-com-delivery-aumentam-em-187-no-brasil/. Acesso em: 21 jun. 2021.

W3SChools. **Documentação**. Disponível em: https://developer.mozilla.org/en-US/. Acesso em: 21 jun. 2021.