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

WELLITON LUIZ BECKER

BELLEAPP: PROTÓTIPO DE APLICATIVO PARA GERENCIAMENTO DE SERVIÇOS NA ÁREA DE BELEZA E ESTÉTICA

RIO DO SUL
2023

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

WELLITON LUIZ BECKER

BELLEAPP: PROTÓTIPO DE APLICATIVO PARA GERENCIAMENTO DE SERVIÇOS NA ÁREA DE BELEZA E ESTÉTICA

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

Prof. Orientador: M.e Jullian Hermann Creutzberg

RIO DO SUL
2023

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

WELLITON LUIZ BECKER

BELLEAPP: PROTÓTIPO DE APLICATIVO PARA GERENCIAMENTO DE SERVIÇOS NA ÁREA DE BELEZA E ESTÉTICA

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

_____

Professor Orientador: M.e Jullian Hermann Creutzberg

Banca Examinadora:

_____

M.e Fernando Andrade Bastos

_____

Esp. Sandro Alencar Fernandes

Rio do Sul, 29 de novembro de 2023.

RESUMO

O mercado de beleza e estética vivenciou um crescimento extraordinário durante o período de meio e pós-pandemia, o que resultou no surgimento de muitas novas empresas. Entretanto, com o aumento de negócios no setor, agendar horários para os serviços de beleza e estética tem se tornado uma tarefa cada vez mais complexa. A maioria dos empreendedores atua de forma individual, o que acarreta dificuldades em acompanhar de maneira adequada suas redes sociais, que são frequentemente utilizadas para agendamentos. Este trabalho tem como objetivo principal desenvolver um protótipo de aplicativo para gerenciamento de serviços na área de beleza e estética. Quanto a metodologia, se caracteriza como uma pesquisa aplicada e descritiva. Para atender aos objetivos propostos, primeiramente foram realizadas pesquisas sobre as tecnologias que seriam utilizadas e que poderiam facilitar o processo de desenvolvimento, as quais estão descritas no capítulo de revisão da literatura. Para melhorar o entendimento das necessidades de desenvolvimento foram identificadas ferramentas similares, sendo que estas estão descritas no capítulo de estado da arte. Na sequência foi realizado um levantamento de requisitos e foram criados diagramas para melhor entendimento do protótipo, estes estão detalhadas na seção de análise. No capítulo de implementação estão detalhadas as rotinas que foram desenvolvidas do protótipo de aplicativo. No que se refere aos benefícios observados, o protótipo desenvolvido poderá auxiliar ao empreendedor do setor de beleza e estética a manter a sua equipe, gerir melhor o tempo além de ter acesso de forma simples aos horários que estão disponíveis ou aos horários que já estão agendados pelos clientes. Com a utilização do aplicativo mobile não será mais necessário disponibilizar um funcionário para marcar em uma agenda, sendo que o próprio usuário cliente poderá verificar os horários disponíveis e realizar o agendamento com o profissional desejado na palma da mão. Para os usuários clientes, a utilização do aplicativo mobile irá agilizar o processo de agendamentos, não ficando o mesmo com a preocupação se o funcionário realmente agendou um horário para ele. Com o aplicativo também o cliente poderá verificar a melhor data e horário, conforme as suas necessidades e disponibilidade da empresa.

Palavras-Chave: BelleApp, desenvolvimento de aplicativo, sistemas de informação.

ABSTRACT

The beauty and aesthetics market has experienced extraordinary growth during the mid- and post-pandemic period, which has resulted in the emergence of many new companies. However, with the increase in business in the sector, scheduling appointments for beauty and aesthetics services has become an increasingly complex task. Most entrepreneurs work individually, which causes difficulties in adequately following their social networks, which are often used for scheduling. The main objective of this work is to develop a prototype application for managing services in the area of beauty and aesthetics. As for the methodology, it is characterized as applied and descriptive research. To meet the proposed objectives, research was first carried out on the technologies that would be used and that could facilitate the development process, which are described in the literature review chapter. To improve the understanding of development needs, similar tools were identified, and these are described in the state of the art chapter. A requirements survey was then carried out and diagrams were created to better understand the prototype, these are detailed in the analysis section. The implementation chapter details the routines that were developed from the application prototype. With regard to the benefits observed, the prototype developed can help entrepreneurs in the beauty and aesthetics sector to maintain their team, manage time better, in addition to having simple access to available times or times that are already scheduled. by customers. Using the mobile application, it will no longer be necessary to provide an employee to schedule an appointment, as the client user will be able to check the available times and make an appointment with the desired professional in the palm of their hand. For customer users, using the mobile application will speed up the scheduling process, leaving them without worrying about whether the employee actually scheduled an appointment for themselves. With the application, the customer can also check the best date and time, depending on their needs and the company's availability.

Keywords: BelleApp, application development, information systems.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

BD	Banco de Dados
CSS	*Cascading Style Sheets*
DCL	*Data Control Language*
DDL	*Data Definition Language*
DER	Diagrama Entidade-Relacionamento
DML	*Data Manipulation Language*
DQL	*Data Query Language*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
JS	JavaScript
JWT	JSON Web Token
MIT	*Massachusetts Institute of Technology*
ODMG	*Object Data Management Group*
ORM	*Object Relational Mapper*
RF	Requisitos Funcionais
RN	Regra de Negócio
RNF	Requisitos não Funcionais
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	*Structured Query Language*
TI	Tecnologia da Informação
TS	TypeScript
WWW	*World Wide Web*

SUMÁRIO

1. INTRODUÇÃO

O mercado de beleza e estética vivenciou um crescimento extraordinário durante o período de meio e pós-pandemia, o que resultou no surgimento de muitas novas empresas. De acordo com informações divulgadas pelo renomado jornal Estadão de São Paulo (ESTADÃO, 2022), somente no mês de junho de 2022, foram registrados impressionantes 11 mil novos empreendimentos (com CNPJ), representando um aumento significativo de 28,5% em comparação ao mesmo mês do ano anterior. Essa tendência de expansão também é evidenciada pela capa do periódico, que ressalta que, ao longo de dois anos, o país testemunhou a abertura de uma notável quantidade de 343 mil salões de beleza.

Entretanto, com o aumento de negócios no setor, agendar horários para os serviços de beleza e estética tem se tornado uma tarefa cada vez mais complexa. A maioria dos empreendedores atua de forma individual, o que acarreta dificuldades em acompanhar de maneira adequada suas redes sociais, que são frequentemente utilizadas para agendamentos. Como resultado, o processo de verificação de disponibilidade de horários e confirmação de agendamentos acaba se tornando demorado e, em muitos casos, frustrante para os clientes, levando-os a desistir da marca ou profissional. Além disso, muitos empreendedores carecem de um controle efetivo dos serviços prestados e enfrentam desafios na gestão financeira, incluindo o equilíbrio entre receitas e despesas.

Diante desse cenário, torna-se imprescindível buscar soluções que aprimorem a experiência tanto para os clientes quanto para os profissionais do setor. É nesse contexto que surge a proposta deste Trabalho de Conclusão de Curso (TCC): desenvolver protótipo de aplicativo para gerenciamento de serviços na área de beleza e estética. O objetivo principal desse aplicativo é proporcionar aos clientes a comodidade de consultar e agendar serviços de forma rápida e conveniente, diretamente na palma de suas mãos, sem a necessidade de confirmação manual por parte da empresa. Além disso, o aplicativo visa otimizar o processo de prestação de serviços para os empreendedores, oferecendo recursos que permitam um controle mais preciso dos serviços prestados, além de uma gestão financeira eficiente.

Portanto, diante do contexto de crescimento acelerado do mercado de beleza e estética, aliado à demanda crescente por serviços mais eficientes e convenientes, a criação desse aplicativo de agendamentos surge como uma resposta estratégica para atender às necessidades do setor. Com o intuito de oferecer uma solução moderna e abrangente, o TCC se propõe a explorar as possibilidades tecnológicas e desenvolver uma ferramenta personalizada que agregue valor tanto para as clientes quanto para os empreendedores, visando aprimorar a

experiência de agendamento, otimizar a gestão dos serviços e impulsionar o crescimento sustentável do setor de beleza e estética.

1.1 PROBLEMA DE PESQUISA

Como melhorar o gerenciamento dos serviços em empresas da área de beleza e estética?

1.2 OBJETIVOS

1.2.1 Geral

- Desenvolver protótipo de aplicativo para gerenciamento de serviços na área de beleza e estética.

1.2.2 Específicos

- Descrever as tecnologias utilizadas para desenvolvimento do aplicativo.
- Detalhar os requisitos funcionais, não funcionais e regras de negócio do aplicativo.
- Identificar aplicações semelhantes ao protótipo proposto.
- Desenvolver o protótipo do aplicativo utilizando as tecnologias selecionadas.

1.3 JUSTIFICATIVA

Durante o período de meio e pós-pandemia, o setor de beleza e estética experimentou um crescimento notável. No entanto, essa expansão do mercado apresentou desafios consideráveis relacionados à gestão de agendamentos. Muitos empreendedores, muitas vezes atuando de forma independente, encontram dificuldades ao tentar organizar eficientemente suas agendas e interações com os clientes. O processo de agendamento, frequentemente conduzido por meio de comunicação em redes sociais, tende a ser moroso e suscetível a erros, o que resulta na insatisfação dos clientes e na perda de oportunidades de negócio. Além disso, esses profissionais muitas vezes carecem de ferramentas eficientes para a gestão dos serviços

prestados e da própria administração financeira. A falta de um controle adequado dos atendimentos, bem como o equilíbrio entre receitas e despesas, dificulta o crescimento sustentável dos empreendimentos no setor.

Diante desse contexto, faz-se necessária a implementação de soluções tecnológicas inovadoras para aprimorar a experiência tanto para os clientes quanto para os empreendedores do mercado de beleza e estética.

Espera-se que o resultado deste trabalho contribua para o desenvolvimento e aprimoramento do setor de beleza e estética, oferecendo soluções tecnológicas que proporcionem benefícios tanto para os clientes quanto para os profissionais, impulsionando o crescimento e a competitividade do mercado.

2. REVISÃO DA LITERATURA

Neste capítulo são apresentados os conceitos, ferramentas e linguagens utilizados para o desenvolvimento do protótipo.

2.1 DESENVOLVIMENTO MOBILE

O desenvolvimento *mobile* é um tipo de rotina de criação de soluções de TI (Tecnologia da Informação) voltadas para *tablets*, *smartphones* e outros dispositivos móveis. Essa área ganhou força na última década após o lançamento do iOS e do Android, as principais plataformas de sistemas operacionais móveis. Hoje, é responsável por receitas bilionárias para quem trabalha com TI, além de ser o principal mecanismo de funcionamento de vários serviços (GAEA, 2021). O autor afirma que “O desenvolvimento mobile tem dominado o mercado nos últimos anos. Com o número de aparelhos móveis ultrapassando a quantidade de computadores de uso profissional e doméstico, muitas pessoas migraram para esse mercado.” (GAEA, 2021, n.p.).

> Utilizamos os dispositivos móveis o tempo todo, seja para acessar informações rápidas, jogar, ler notícias, estudar, interagir nas redes sociais, escutar música, ver vídeos, trabalhar e muito mais. Não é à toa que a área anda bem-aquecida. Para se ter uma ideia, em 2021, tivemos um aumento de 600% na quantidade de vagas disponíveis para desenvolvimento mobile. (XPEDUCACAO, 2022, n.p.).

As principais linguagens utilizadas no desenvolvimento *mobile* são: HTML (*Hypertext Markup Language*), CSS (*Cascading Style Sheets*), JS (JavaScript), Java, PHP, C, C++, Objective-C, LUA, Kotlin, Swift. Eles afirmam que existem várias outras linguagens e muitas delas são específicas para cada sistema operacional, por exemplo o Kotlin é mais direcionada para o Android e o Objective-C é voltado para o IOS (XPEDUCACAO, 2022).

2.2 HTML

Conforme explica o Mozilla (2023a), o HTML é o bloco de construção mais básico da web. Ele é responsável por definir o significado e a estrutura do conteúdo presente na página. Outras tecnologias, como o CSS, geralmente são usadas para definir a aparência/apresentação, enquanto o JS é utilizado para definir a funcionalidade/comportamento de uma página da web. O termo "Hipertexto" refere-se aos links que conectam as páginas da web, tanto dentro de um mesmo site quanto entre diferentes sites. Esses links são uma parte fundamental da web,

permitindo que o conteúdo seja carregado na Internet e vinculado a outras páginas criadas por outros usuários, fazendo com que o usuário se torne um participante ativo na WWW (*World Wide Web*).

Já para o W3schools (2023b, n.p.), o HTML que "significa *Hyper Text Markup Language*, é uma linguagem de marcação padrão para criar páginas da Web, descreve a estrutura de uma página da Web, consiste em uma série de elementos que informam ao navegador como exibir o conteúdo".

> HTML não é uma linguagem de programação; é uma linguagem de marcação, usada para definir a estrutura do seu conteúdo. HTML consiste em uma série de elementos, que você usa para delimitar ou agrupar diferentes partes do conteúdo para que ele apareça ou atue de determinada maneira. (MOZILLA, 2023a, n.p.).

No Quadro 1 podemos ver as principais partes de um elemento HTML, conforme destaca o Mozilla:

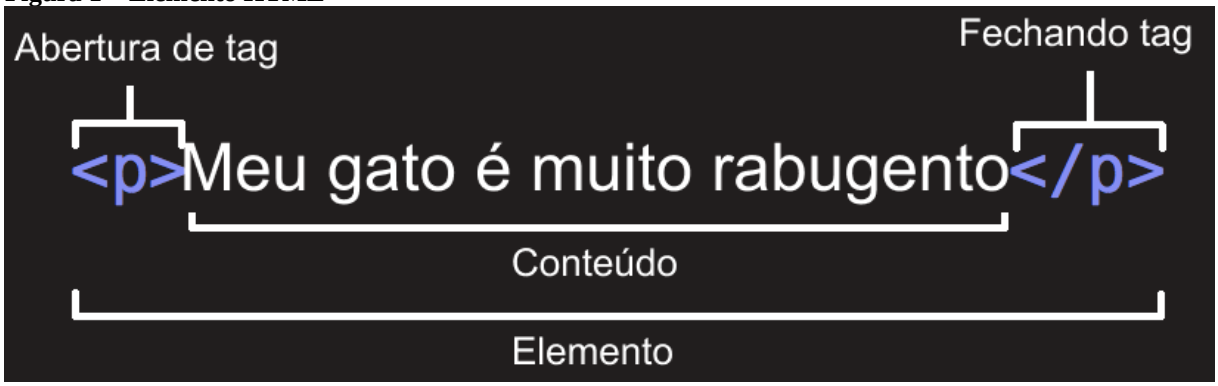
Quadro 1 – Elemento HTML

***Tag* de abertura**	Consiste no nome do elemento, envolvido em parênteses angulares de abertura e fechamento. Isso demonstra onde o elemento começa, ou onde seu efeito se inicia.
***Tag* de fechamento**	Isso é a mesma coisa que a *tag* de abertura, exceto que inclui uma barra antes do nome do elemento. Isso demonstra onde o elemento acaba. Esquecer de incluir uma *tag* de fechamento é um dos erros mais comuns de iniciantes e pode levar a resultados estranhos.
O conteúdo	Esse é o conteúdo do elemento.
O elemento	A *tag* de abertura, a de fechamento, e o conteúdo formam o elemento.

Fonte: Elaborado a partir de Mozilla (2023b, n.p.)

Na Figura 1 podemos ver a estrutura de um elemento de parágrafo em HTML. Note que as *tags* possuem a sua abertura e o seu fechamento, e tudo o que estiver entre elas é considerado o conteúdo dela.

Figura 1 – Elemento HTML



Fonte: Mozilla (2023a, n.p.)

O W3schools (2023a) ainda explica algumas partes do HTML, conforme mostra o Quadro 2:

Quadro 2 – Exemplos de elementos HTML

<!DOCTYPE html>	Declaração define que este documento é um documento HTML5.
<html>	Elemento é o elemento raiz de uma página HTML.
<*head*>	Elemento contém metainformações sobre a página HTML.
<*title*>	Elemento especifica um título para a página HTML (que é mostrado na barra de título do navegador ou na guia da página).
<*body*>	Elemento define o corpo do documento e é um recipiente para todos os conteúdos visíveis, como cabeçalhos, parágrafos, imagens, hiperlinks, tabelas, listas etc.
<h1>	Elemento define um título de primeiro nível.
<p>	Elemento define um parágrafo.

Fonte: Elaborado a partir de W3schools (2023a, n.p.)

2.3 CSS

O CSS "significa *Cascading Style Sheets*, descreve como os elementos HTML devem ser exibidos na tela, no papel ou em outras mídias, economiza muito trabalho, pode controlar o layout de várias páginas da Web de uma só vez e folhas de estilo externas são armazenadas em arquivos CSS" (W3SCHOOLS, 2023b, n.p.).

Segundo explica a documentação do Mozilla (2023b, n.p.), "CSS (*Cascading Style Sheets* ou Folhas de Estilo em Cascata) é uma linguagem de estilo usado para descrever a apresentação de um documento escrito em HTML ou em. O CSS descreve como elementos são mostrados na tela, no papel, na fala ou em outras mídias.".

Gonçalves (2022, n.p.) afirma que o "CSS foi desenvolvido pelo W3C (*World Wide Web Consortium*) em 1996, por uma razão bem simples. O HTML não foi projetado para ter *tags* que ajudariam a formatar a página. Você deveria apenas escrever a marcação para o site." O autor ainda complementa que "a relação entre HTML e CSS é bem forte. Como o HTML é uma linguagem de marcação (o alicerce de um site) e o CSS é focado no estilo (toda a estética de um site), eles andam juntos." (GONÇALVES, 2022, n.p.).

Na Figura 2 podemos ver um exemplo demonstrado no site do W3schools de como é a sintaxe do CSS;

Figura 2 – Exemplo de Sintaxe CSS

```css
body {
  background-color: lightblue;
}

h1 {
  color: white;
  text-align: center;
}

p {
  font-family: verdana;
  font-size: 20px;
}
```

Fonte: W3schools (2023a, n.p.)

Veja que ao escolher um elemento HTML, tudo o que está dentro das chaves {} será aplicado ao elemento em questão para todo o escopo em que esse arquivo for chamado.

2.4 JAVASCRIPT

"Javascript é uma linguagem de programação interpretada criada em 1995 por Brendan Eich da Netscape como uma extensão do HTML para o *browser Navigator* 2.0.". (ROCHA, 1999, p.12).

Segundo Flanagan (2002), o JS é uma linguagem de programação para web, que está na grande maioria dos sites, pois qualquer navegador possui interpretadores de JS tornando assim a linguagem mais onipresente da história. Ela é uma linguagem de programação de alto nível, interpretada e não é tipado, no entanto existem algumas ferramentas e *frameworks* que conseguem lidar com o JS de forma tipado. E como a maioria das linguagens de hoje ela aceita orientação a objetos. Por mais que o seu nome possa parecer que ela é derivada do Java, o JS é completamente diferente e já deixou de ser de ser uma linguagem de *script* e como uma linguagem complementar a outras linguagens e tornou se uma linguagem de uso geral robusta e eficiente sendo utiliza sozinha para sites e aplicativos modernos.

> JavaScript nos permite fazer *scripts* do conteúdo HTML e da apresentação CSS de documentos em navegadores Web, mas também nos permite definir o comportamento desses documentos com rotinas de tratamento de evento. Uma rotina de tratamento de evento é uma função JavaScript que registramos no navegador e que este chama

> quando ocorre algum tipo de evento especificado. O evento de interesse pode ser um clique de mouse ou um pressionamento de tecla (ou, em um *smartphone*, pode ser um gesto de algum tipo, feito com dois dedos). Ou então, uma rotina de tratamento de evento pode ser ativada quando o navegador termina de carregar um documento, quando o usuário redimensiona a janela do navegador ou quando o usuário insere dados em um elemento de formulário HTML. (FLANAGAN, 2002, p. 10).

Já para Rocha (1999), o JS é uma linguagem de programação baseada em objetos e trata suas estruturas básicas como as propriedades do *browser* e elementos de uma página HTML como objetos e que permitem serem manipuladas através de eventos programáveis que possuem interação com o usuário, recursos que faltam no HTML e permitem a criação de sites interativos e dinâmicos, interpretados pelo *browser* sem a necessidade de uma requisição ao servidor.

2.4.1 Eventos

Conforme explica Flanagan (2002, p. 10), “Os programas JavaScript do lado do cliente usam um modelo de programação dirigido por eventos assíncronos. Nesse estilo de programação, o navegador Web gera um evento onde acontece algo interessante no documento, no navegador ou em algum elemento ou objeto associado a ele.”.

> Uma rotina de tratamento de evento ou ouvinte de evento é uma função que manipula ou responde a um evento. Os aplicativos registram suas funções de tratamento de evento no navegador Web, especificando um tipo e um alvo de evento. Quando ocorre um evento do tipo especificado no alvo especificado, o navegador chama a rotina de tratamento. Quando as rotinas de tratamento de evento são chamadas para um objeto, às vezes dizemos que o navegador “ativou”, “disparou “ou “despachou” o evento. Um objeto evento é um objeto associado a um evento em especial e contém detalhes sobre esse evento. Cada tipo de evento define um conjunto de propriedades para seu objeto evento associado. O objeto associado a um evento de mouse contém as coordenadas do cursor do mouse, por exemplo, e o objeto associado a um evento de teclado contém detalhes sobre a tecla que foi pressionada e sobre as teclas modificadoras que foram mantidas pressionadas. (FLANAGAN, 2002, p. 10).

Em seu livro Rocha (1999) fala sobre 13 tipos de eventos que são explicados no Quadro 3.

Quadro 3 – Eventos JavaScript

onclick	Quando um objeto é clicado pelo mouse. Elementos: <a>, <*input*>
onselect	Quando um objeto é selecionado. Elementos: <*input type=text*>, <*textarea*>
onchange	Quando uma seleção ou campo de texto tem seu conteúdo modificado. Elementos: <*input type=text*>, <*textarea*>, <*select*>
onfocus	Quando um componente de formulário ou janela se torna ativa. Elementos: <*textarea*>, <*body*>, <*form*>, <*input*>, <*select*>, <*option*>
onblur	Quando um componente de formulário ou janela se torna inativa. Elementos: <*textarea*>, <*body*>, <*form*>, <*input*>, <*select*>, <*option*>
onmouseover	Quando o mouse está sobre um link. Elementos: <a>, <*area*>
onmouseout	Quando o mouse deixa um link. Elementos: <a>, <*area*>
onsubmit	Antes de enviar um formulário. Elementos: <*input type=submit*>
onreset	Antes de limpar um formulário. Elementos: <*form*>
onload	Após carregar uma página, janela ou frame. Elementos: <*body*>
onunload	Ao deixar uma página, janela ou frame. Elementos: <*body*>
onerror	Quando um erro ocorre durante a carga de uma imagem ou página. Elementos: <*img*>, <*body*>
onabort	Quando a carga de uma imagem é abortada. Elementos: <*img*>

Fonte: Elaborado a partir de Rocha (1999, p. 8)

2.4.2 TypeScript

Conforme explica a documentação do TypeScript (2023), ele mantém uma relação incomum com o JS. O TS, sigla do TypeScript oferece todos os recursos do JS, mas com uma camada adicional, a tipagem. Com o TS o JS fica tipado, assim se uma variável for declarada como inteiro, ela só poderá receber um valor inteiro. O principal benefício do uso do TS, é que destaca comportamentos inesperados diminuindo a chances de erros.

A documentação do TypeScript (2023, n. p.) ainda complementa, “Ao entender como o JavaScript funciona, o TypeScript pode criar um sistema de tipos que aceita código JavaScript, mas possui tipos. Isso oferece um sistema de tipos sem a necessidade de adicionar caracteres extras para tornar os tipos explícitos em seu código.”.

2.5 FRAMEWORKS E BIBLIOTECAS

Esta seção trará informações sobre as *frameworks* e bibliotecas utilizados no protótipo.

2.5.1 TailwindCss

Dos Santos (2023), explica que o TailwindCss é um *framework* CSS de código aberto conhecido por sua abordagem singular de design utilitário. Em vez de oferecer estilos pré-definidos para elementos específicos, ele disponibiliza um conjunto de classes utilitárias que podem ser aplicadas diretamente a elementos HTML para personalizar sua aparência. Essas

classes representam propriedades CSS individuais, como margens, preenchimentos, cores, tamanhos de fonte, alinhamentos e diversas outras opções de estilização.

Na Figura 3 temos um exemplo da utilização do TailwindCss em um botão.

Figura 3 – Exemplo TailwindCss

Fonte: Dos Santos (2023, n.p.).

2.5.2 Node.js

Conforme citado por Simas (2019, p.76), “O Node.js é baseado em alguns componentes, como o motor ou engine V8, um interpretador JavaScript criado pelo Google e utilizado no navegador Chrome;”.

> No final de 2009, Ryan Dahl com a ajuda inicial de 14 colaboradores criou o Node.js. Esta tecnologia possui um modelo inovador, sua arquitetura é totalmente *non-blocking thread* (não-bloqueante), apresentando uma boa performance com consumo de memória e utilizando ao máximo e de forma eficiente o poder de processamento dos servidores, principalmente em sistemas que produzem uma alta carga de processamento. Usuários de sistemas Node estão livres de aguardarem por muito tempo o resultado de seus processos, e principalmente não sofrerão de *dead-locks* no sistema, porque nada bloqueia em sua plataforma e desenvolver sistemas nesse paradigma é simples e prático. (PEREIRA, 2014, p. 2).

Pereira (2014, p.2) ainda continua, “Esta é uma plataforma altamente escalável e de baixo nível, pois você vai programar diretamente com diversos protocolos de rede e internet. O Javascript é a sua linguagem de programação, e isso foi possível graças à *engine* Javascript V8, a mesma utilizada no navegador Google Chrome.”, por utilizar o JS como linguagem, muitos programadores de *front-end*, que utilizam o JS, são capazes de escrever códigos no *server-side* sem precisar aprender uma linguagem nova.

2.5.3 React.js

Em 2015, a equipe interna do Facebook resolveu a dificuldade de desenvolver para duas plataformas distintas, Android e iOS, ao criar o React Native. Essa solução permitiu o desenvolvimento de um único aplicativo que pudesse ser executado em ambas as plataformas,

eliminando a necessidade de criar aplicativos separados para cada uma delas. (CALOMENO JUNIOR, 2020).

> À medida que novas estruturas e bibliotecas da Web avançam, o React é um grande sucesso. Não só trata com os problemas mais comuns enfrentados pelos desenvolvedores ao criar aplicativos de página única, ele lança em alguns truques adicionais que tornam a criação de visuais para seus aplicativos de página única muito fácil. Desde que foi lançado em 2013, o React encontrou seu caminho em sites populares e aplicativos que você provavelmente usa. Além do Facebook e do Instagram, alguns dos notáveis incluem a BBC, Khan Academy, PayPal, Reddit, The New York Times, Yahoo e muitos mais. (CHINNATHAMBI, 2017, p. 11).

Segundo a documentação do React (2023, n.p.), "O React adota o fato de que a lógica de renderização são inerentemente acopladas a outra lógica de interface do usuário: como os eventos são tratados, como o estado muda com o tempo e como os dados são preparados para exibição.".

> Em vez de separar tecnologias artificialmente colocando marcação e lógica em arquivos separados, o React separa as preocupações com unidades fracamente acopladas chamadas "componentes" que contêm ambos. O React não requer o uso de JSX, mas a maioria das pessoas o considera útil como auxílio visual ao trabalhar com a interface do usuário dentro do código JavaScript. Ele também permite que o React mostre mensagens de erro e aviso mais úteis. (REACT, 2023, n.p.).

Conforme a documentação explica, os componentes nos permitem a divisão da interface em partes independentes e reutilizáveis. Conceitualmente os componentes são como funções do JS, eles aceitam entradas arbitrárias (chamadas de "props") e retorna elementos React descrevendo o que deve aparecer na tela. (REACT, 2023).

2.5.4 Next.js

"O Next JS ou Next.js é uma espécie de *framework* para o React. Ou seja, é correto afirmar que o Next.js é uma 'continuação' do React.js, com algumas funcionalidades a mais." (MARCHIORI, 2023, n.p.).

Silva (2023, n.p.), comenta algumas das características do Next.js: "Renderização estática e pelo lado do servidor, suporte ao TypeScript, serviço de tratamento de rotas, melhora a performance de React, entrega a página pronta para o *browser*, permite a instalação de *plugins* e permite apenas o carregamento necessário, pois separa o código.".

Sobre a utilização do Next.js, "ele faz sucesso nos projetos de *e-commerces*, sites de marketing, websites estáticos e com foco em SEO. Portanto, ao navegar pelas páginas web da

maioria dos sites, você certamente usará algo com a tecnologia Next.js. Afinal, o *framework* garante escalabilidade e agilidade ao desenvolvedor.”. (SILVA, 2023, n.p.)

2.5.5 Fastify

O Fastify é um framework utilizado no *back-end* e possui um baixo custo de infraestrutura e com uma capacidade muito alta de respostas, podendo chegar até a 30 mil requisições por segundo. Possui de forma nativa *logs* de requisições ao sistema, e possui suporte ao TS. Ele é um framework de código aberto produzido e mantido pela OpenJS Foundation. (FASTIFY, 2023, n.p.).

Para Filgueira (2018), o fastify fornece “um núcleo leve e pequeno que é fácil de estender com plugins e amadurecer seu aplicativo baseando-se em serviços, com alto desempenho e baixa sobrecarga. O padrão de arquitetura que é usado para criá-lo permite aplicativos prontos, leves e robustos para microsserviços.”.

2.5.6 Expo

Segundo o Sidelab (2022), o Expo consiste em um *framework* empregado no processo de desenvolvimento de aplicações com React Native. Este *framework* engloba um conjunto de recursos e serviços desenvolvidos para trabalhar em conjunto com as plataformas nativas e o *framework* React Native, facilitando significativamente o ciclo de desenvolvimento, construção, implementação e iteração de aplicativos para iOS, Android e web, tudo a partir de um único código base em JavaScript/TypeScript.

> Quando iniciamos no desenvolvimento *mobile* percebemos que o número de API’s e recursos nativos que podemos controlar através da nossa aplicação é gigante, e muitas vezes não nos recordamos de todas as opções que temos disponíveis. O Expo, por sua vez, oferece grande parte desses recursos de forma nativa e integrada e, por exemplo, você tem acesso à recursos como câmera, microfone, player de música, entre outros, de forma muito simples utilizando essa ferramenta. Apesar de todos esses benefícios, o grande ponto do Expo para quem está iniciando é que para começar a desenvolver suas aplicações mobile com React Native você não precisará instalar a SDK do Android ou o XCode para Mac, isso porque o Expo possui um aplicativo móvel instalável pelas lojas do Android/iOS que contém todo código nativo necessário pelo React Native para iniciar sua aplicação e, dessa forma, a única alteração em código que você faz é em Javascript. (FERNANDES, 2018, n.p.).

Conforme explica Sidelab (2022, n.p.), são necessários duas ferramentas para o desenvolvimento com o Expo “um aplicativo de linha de comando chamado Expo CLI para

inicializar e servir seu projeto, e um aplicativo cliente móvel chamado Expo Go para abri-lo no iOS e Android. Qualquer navegador da web funcionará para abrir o projeto na web.”.

2.5.7 JSON *Web Token* (JWT)

Conforme explica o JWT (2023, n.p.), conhecido como JSON *Web Token*, ele é “um padrão aberto que define uma maneira compacta e independente de transmitir informações com segurança entre as partes como um objeto JSON. Essas informações podem ser verificadas e confiáveis porque são assinadas digitalmente.”.

> Na autenticação, quando o usuário fizer *login* com sucesso usando suas credenciais, um JSON *Web Token* será retornado. Como os *tokens* são credenciais, deve-se tomar muito cuidado para evitar problemas de segurança. Em geral, você não deve manter os tokens por mais tempo do que o necessário. Sempre que o usuário desejar acessar uma rota ou recurso protegido, o agente do usuário deverá enviar o JWT, normalmente no cabeçalho *Authorization* usando o esquema *Bearer*. Isto pode ser, em certos casos, um mecanismo de autorização apátrida. As rotas protegidas do servidor verificarão um JWT válido no cabeçalho e, se estiver presente, o usuário terá permissão para acessar recursos protegidos. (JWT, 2023, n.p.).

Segundo Lima (2021, n.p.), o JWT “é utilizado em dois principais cenários, quando queremos realizar um processo de autorização em nossa aplicação ou quando queremos realizar troca de informações, abaixo detalhamos um pouco mais sobre o uso do JWT em cada um dos cenários citados.”.

2.5.8 Axios

Segundo o próprio Axios (2023, n.p.), ele “é um cliente HTTP baseado em promessas para o node.js e para o navegador. É isomórfico. No lado do servidor usa o código nativo do node.js – o módulo http, enquanto no lado do cliente (navegador) usa XMLHttpRequests”.

Neves (2022, n.p.), explica que “AXIOS é um cliente HTTP baseado em promessas totalmente agnóstico, ou seja, que não depende de *frameworks* e bibliotecas. Ele é super maleável, podendo ser utilizado do lado do cliente e do lado do *backend*.”.

2.5.9 Prisma

Conforme explica Buzzi (2022) antes de se tornar Prisma, o projeto inicialmente conhecido como Graphcool foi concebido por uma equipe de apenas cinco desenvolvedores. O

objetivo desse projeto era criar uma solução inovadora de "*backend-as-a-service*" para o GraphQL. O Graphcool foi bem recebido pela comunidade na época, especialmente por sua reputação de ser “fácil de usar”, mesmo para desenvolvedores *frontend*.

> Prisma é uma ferramenta *open source*, um ORM de próxima geração cuja arquitetura é desenhada em três camadas fundamentais: * Prisma Client: um construtor de *queries* gerado automaticamente e *type-safe* para Node.js e TypeScript; * Prisma Migrate: sistema de migração; * Prisma Studio: o produto principal da tecnologia. Trata-se de uma interface do usuário feita para visualizar e editar os dados na database; Na descrição oficial da ferramenta, Prisma é descrito como um novo tipo de ORM, fundamentalmente diferente dos modelos tradicionais que eram aplicados anteriormente. Uma alternativa para outras ORMs, como TypeORM e Sequelize. Todo o conceito do Prisma está no *schema* proporcionado por eles no desenvolvimento nestas três camadas citadas anteriormente. Segundo eles, o *schema* é descrito como o principal arquivo de configuração para o Prisma. (BUZZI, 2022, n.p.).

Na Figura 4 podemos ver um exemplo do prisma *client*, que utilizado para fazer as criações de tabelas, ou também conhecido como comandos DDL:

Figura 4 – Exemplo de Prisma Client

```
datasource db {
  url      = env("DATABASE_URL")
  provider = "postgresql"
}

generator client {
  provider = "prisma-client-js"
}

model User {
  id        Int      @id @default(autoincrement())
  createdAt DateTime @default(now())
  email     String   @unique
  name      String?
  role      Role     @default(USER)
  posts     Post[]
}

model Post {
  id        Int      @id @default(autoincrement())
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt
  published Boolean  @default(false)
  title     String   @db.VarChar(255)
  author    User?    @relation(fields: [authorId], references: [id])
  authorId  Int?
}

enum Role {
  USER
  ADMIN
}
```

Fonte: Prisma.io (2023, n.p.)

2.6 BANCO DE DADOS

Na visão de Ramakrishnan e Gehrke (2003, p. 4), “Um banco de dados é uma coleção de dados inter-relacionados que são armazenados de forma persistente em uma mídia de armazenamento controlada por um software de gerenciamento de banco de dados.”.

Segundo Heuser (2001), os sistemas para gerenciar os bancos de dados, chamados de SGBD (Sistema de Gerenciamento de Banco de Dados), surgiram na década de 70 com o objetivo de facilitar a programação de aplicações de banco de dados.

> Durante muito tempo, criou-se a ideia de que projetar bancos de dados era uma disciplina com identidade própria, uma atividade específica e, até certo ponto, isolada no ciclo de vida de um sistema, que tinha existência própria e podia ser realizada a partir de primitivas e conceitos exclusivos da técnica de modelagem de dados (MACHADO, 2020, p.14).

Conforme relata Cayres (2015), linguagens de programação e ferramentas de front-end (aplicações gráficas), os SGBDs auxiliam na construção, manutenção e manipulação dos dados armazenados no banco de dados. Internamente eles apresentam linguagens para trabalhar com os dados como DDL (*Data Definition Language*) para definir como os dados serão salvos (comandos *create, update e drop*), DML (*Data Manipulation Language*) para manipular os dados (comandos *insert, update e delete*), DQL (*Data Query Language*) para consulta dos dados (comando *select*) e DCL (*Data Control Language*) para controle de acesso aos dados (comandos *grant* e *revoke*).

Para Heuser (2001, p. 3), “Redundância de dados ocorre quando uma determinada informação está representada no sistema em computador várias vezes. Há duas formas de redundância de dados, a redundância controlada de dados e a redundância não controlada de dados.”.

> A técnica de modelagem de dados mais difundida e utilizada é a abordagem entidade-relacionamento (ER). Nesta técnica, o modelo de dados é representado através de um modelo entidade-relacionamento (modelo ER). Usualmente, um modelo ER é representado graficamente, através de um diagrama entidade-relacionamento (DER). A abordagem ER foi criada em 1976 por Peter Chen. Ela pode ser considerada como um padrão de fato para modelagem conceitual. Mesmo as técnicas de modelagem orientada a objetos que têm surgido nos últimos anos baseiam-se nos conceitos da abordagem ER. (HEUSER, 2001, p. 22)

Heuser (2001, p. 3) sugere que “Para construir um modelo de dados, usa-se uma linguagem de modelagem de dados. Linguagens de modelagem de dados podem ser

classificadas de acordo com a forma de apresentar modelos, em linguagens textuais ou linguagens gráficas.”.

2.6.1 Modelo de Banco de Dados

Para Elmasri e Navathe (2018, p. 29), um modelo de dados se refere à “uma coleção de conceitos que podem ser usados para descrever a estrutura de um banco de dados”.

Ainda segundo Elmasri e Navathe (2018), podemos classificar os modelos de banco de dados em dois conceitos que eles utilizam para descrever a estrutura do banco de dados. Alto nível ou conceitual, que utilizam conceitos como entidades, atributos e relacionamentos, fornecem uma visão mais próxima do modo como os usuários visualizam os dados em toda a sua realidade. E modelo de baixo nível ou físicos que descrevem detalhes como os dados são armazenados, por este motivo é mais voltada para especialistas de computadores e não usuários finais.

> Modelo Conceitual (DER): também conhecido como Diagrama Entidade-Relacionamento, é um modelo de dados abstrato que descreve a estrutura de um banco de dados independentemente de sua implementação; Modelo Lógico: tem como objetivo transformar o modelo conceitual em um modelo que define como o banco de dados será implementado em um SGBD específico. Deve representar relações e restrições do modelo de dados que representa a estrutura de um BD e o esquema de banco de dados; Modelo Físico: nessa fase o modelo do banco de dados é enriquecido com detalhes que influenciam no desempenho do banco de dados, mas não interferem na sua funcionalidade. Script do banco de dados em SQL representa os detalhes dos dados internamente ao BD (campo, tipo/domínio, restrições). (CAYRES, 2015, p. 11).

Elmasri e Navathe (2018) mencionam alguns exemplos de modelos de dados tais como: modelo de dados conceituais, modelo de dados representativos ou de implementação, modelo de dados de objeto, modelo de dados físicos e modelo de dados auto descritivos, conforme apresentado no Quadro 4.

Quadro 4 – Modelos de dados

Modelo de dados conceituais	Utilizam conceitos como entidades, atributos e relacionamentos. Uma entidade representa um objeto ou conceito do mundo real, como um funcionário ou um projeto do minimundo que é descrito no banco de dados. Um atributo representa alguma propriedade de interesse que descreve melhor uma entidade, como o nome ou o salário do funcionário. Um relacionamento entre duas ou mais entidades representa uma associação entre elas — por exemplo, um relacionamento entre um funcionário e um projeto.
Modelo de dados representativos ou de implementação	São os usados com mais frequência nos SGBDs comerciais tradicionais. Estes incluem o amplamente utilizado modelo de dados relacional, bem como os chamados modelos de dados legados — os modelos de rede e hierárquicos — que foram bastante usados no passado. Os modelos de dados representativos mostram os dados usando estruturas de registro e, portanto, às vezes são denominados modelos de dados baseados em registro.
Modelo de dados de objeto	Podemos considerar como um exemplo de uma nova família de modelos de dados de implementação de nível mais alto e que são mais próximos dos modelos de dados conceituais. Um padrão para bancos de dados de objeto, chamado modelo de objeto ODMG, foi proposto pelo grupo de gerenciamento de dados de objeto (ODMG — *Object Data Management Group*). Os modelos de dados de objeto também são frequentemente utilizados como modelos conceituais de alto nível, em particular no domínio da engenharia de software. Um recurso-chave dos bancos de dados de objeto é o poder que eles dão ao projetista para especificar tanto a estrutura dos objetos complexos quanto as operações que podem ser aplicadas a esses objetos.
Modelo de dados físicos	Descrevem como o dado é armazenado como arquivos no computador, com informações como formatos de registro, ordenações de registro e caminhos de acesso. Um caminho de acesso é uma estrutura que torna eficiente a busca por registros de um banco de dados em particular, como indexação ou *hashing*. Um índice é um exemplo de um caminho de acesso que permite o acesso direto aos dados usando um termo de índice ou uma palavra-chave.
Modelo de dados auto descritivos	O armazenamento de dados nos sistemas baseados nesses modelos combina a descrição dos dados com seus próprios valores. Nos SGBDs tradicionais, a descrição fica separada dos dados. Esses modelos incluem XML, além de muitos dos armazenamentos de chave-valor e sistemas NOSQL, que foram criados recentemente para gerenciar big data.

Fonte: Elaborado a partir de Elmasri e Navathe (2018)

2.6.2 PostgreSQL

PostgreSQL é “uma ferramenta que atua como sistema de gerenciamento de bancos de dados relacionados. Seu foco é permitir implementação da linguagem SQL em estruturas, garantindo um trabalho com os padrões desse tipo de ordenação dos dados.” (SOUZA, 2020, n. p.).

> Como o PostgreSQL é robusto, seguro e extensível, além de ter um rico ecossistema de ferramentas disponíveis, os desenvolvedores usam o PostgreSQL para uma variedade de casos de uso. O software foi projetado para ser compatível com todos os principais sistemas operacionais incluindo Linux, Windows e Macintosh, além de oferecer suporte a texto, imagens, sons e vídeos, sendo assim um banco de dados popular para pessoas e empresas com necessidades diversas. O PostgreSQL é amplamente considerado como a tecnologia de banco de dados favorita dos desenvolvedores, ficando atrás apenas do MySQL. (AZURE, 2023, n. p.).

A Azure (2023) ainda explica que como o PostgreSQL é de código aberto os usuários têm mais liberdade e podem explorar todo o potencial do banco de dados. O autor ainda afirma que por ser de código aberto, a obtenção de suporte se torna simples pela grande quantidade de comunidades que se auxiliam para relatar *bugs* ou resolvê-los.

2.6.3 Supabase

Conforme explica Carnes (2023), o Supabase se destaca como uma ferramenta extraordinária para criar *backends* postgres altamente seguros e eficientes, tudo isso com uma configuração simplificada. Ele disponibiliza aos desenvolvedores uma extensa gama de recursos comparáveis aos oferecidos pelo Firebase, incluindo autenticação, banco de dados com atualizações em tempo real e capacidades de armazenamento. No entanto, como uma alternativa de código aberto, o Supabase proporciona um nível superior de flexibilidade e controle sobre tanto seus dados quanto suas aplicações.

Perera (2021, n.p.) faz uma comparação entre o Firebase e o Supabase: “Supabase pode ser mais útil porque usa tecnologia de código aberto. Oferece flexibilidade para hospedar em sua máquina local, em um provedor de serviços em nuvem ou até mesmo como um contêiner Docker. Isso significa que é livre de restrições.”.

3. METODOLOGIA DA PESQUISA

O presente trabalho de conclusão de curso caracteriza-se como pesquisa aplicada e descritiva, pois tem como objetivo o desenvolvimento de um protótipo de aplicativo para facilitar o gerenciamento de empresas que prestam serviços no ramo de beleza e estética com funcionalidades para o controle de funcionários, serviços prestados, e para funcionalidades do cliente como serviços prestados pela empresa e agendamento dos serviços de forma online e automática.

O trabalho busca responder o seguinte problema: Como melhorar o gerenciamento dos serviços na área de beleza e estética.

Após a finalização do levantamento bibliográfico, foi realizado um estudo mais aprofundado sobre o desenvolvimento de aplicativos, utilizando as ferramentas e padrões de desenvolvimentos mais modernos, visando deixar o aplicativo relevante, mitigando retrabalhos futuros e facilitando implementações futuras. As tecnologias, linguagens de programação e *frameworks* que são utilizados no desenvolvimento do protótipo estão documentadas na revisão de literatura.

Foi realizada uma pesquisa para a seção de estado da arte, analisando sistemas semelhantes ao protótipo proposto neste trabalho, a fim de averiguar possíveis funcionalidades e melhorias e auxiliar no levantamento de requisitos e regras de negócio. Na etapa seguinte, a etapa de análise, foi desenvolvido diagramas (atividade, entidade relacionamento, caso de uso) para facilitar a compreensão do protótipo e no desenvolvimento.

Na etapa de desenvolvimento foi definido que a aplicação seria desenvolvida com auxílio do software Visual Studio Code, utilizando como linguagem base o JS e para o *frontend* será utilizado as bibliotecas do ReactJs com auxílio do *framework* do TailwindCss para a estilização dos componentes.

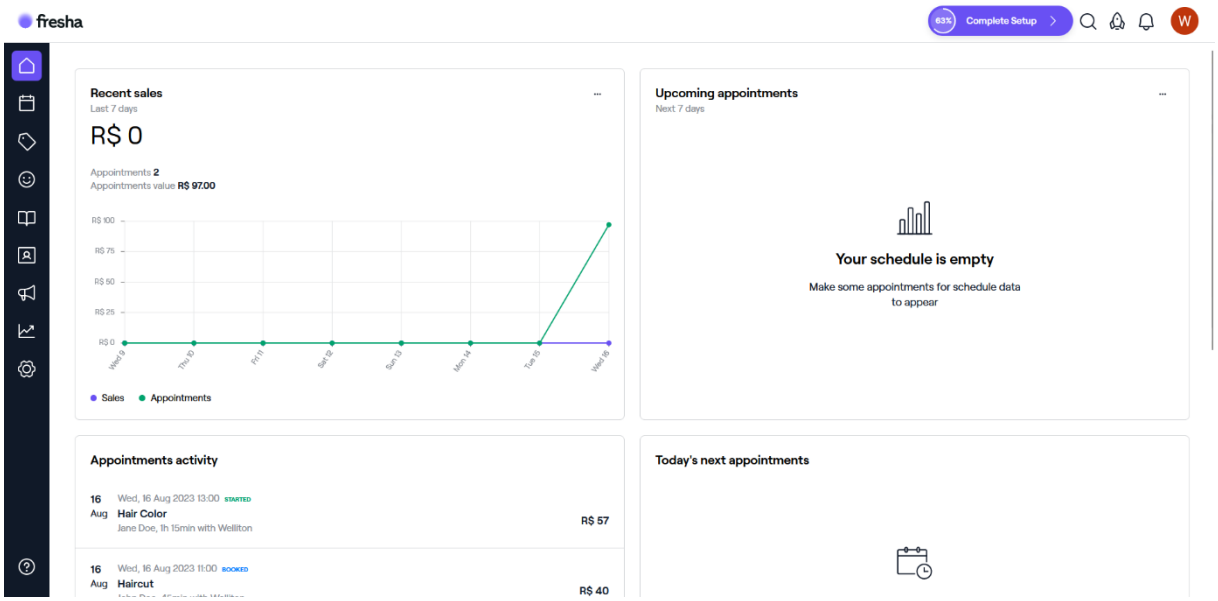
3.1 ESTADO DA ARTE

Neste segmento, são apresentados dois exemplos de aplicações atualmente disponíveis no mercado e um protótipo apresentado em um artigo, as quais compartilham características e objetivos semelhantes em comparação com o protótipo proposto neste trabalho. São exploradas as características gerais e as principais funcionalidades dessas aplicações, visando enriquecer a análise do levantamento de requisitos para o referido protótipo.

3.1.1 Fresha

O aplicativo Fresha (Figura 5) é uma plataforma direcionada principalmente aos profissionais independentes e empresas atuantes na indústria de beleza e bem-estar. Este aplicativo proporciona uma solução prática para que clientes possam agendar serviços, como corte de cabelo, manicure, massagens e outros tratamentos, de maneira eficaz e *on-line*. O sistema permite a criação de diversas categorias e serviços, com informações sobre preço, duração e os profissionais responsáveis pela realização dos serviços, além de viabilizar o agendamento manual por parte dos profissionais. Tudo isso é apresentado por meio de uma interface moderna e otimizada. (FRESHA, 2023).

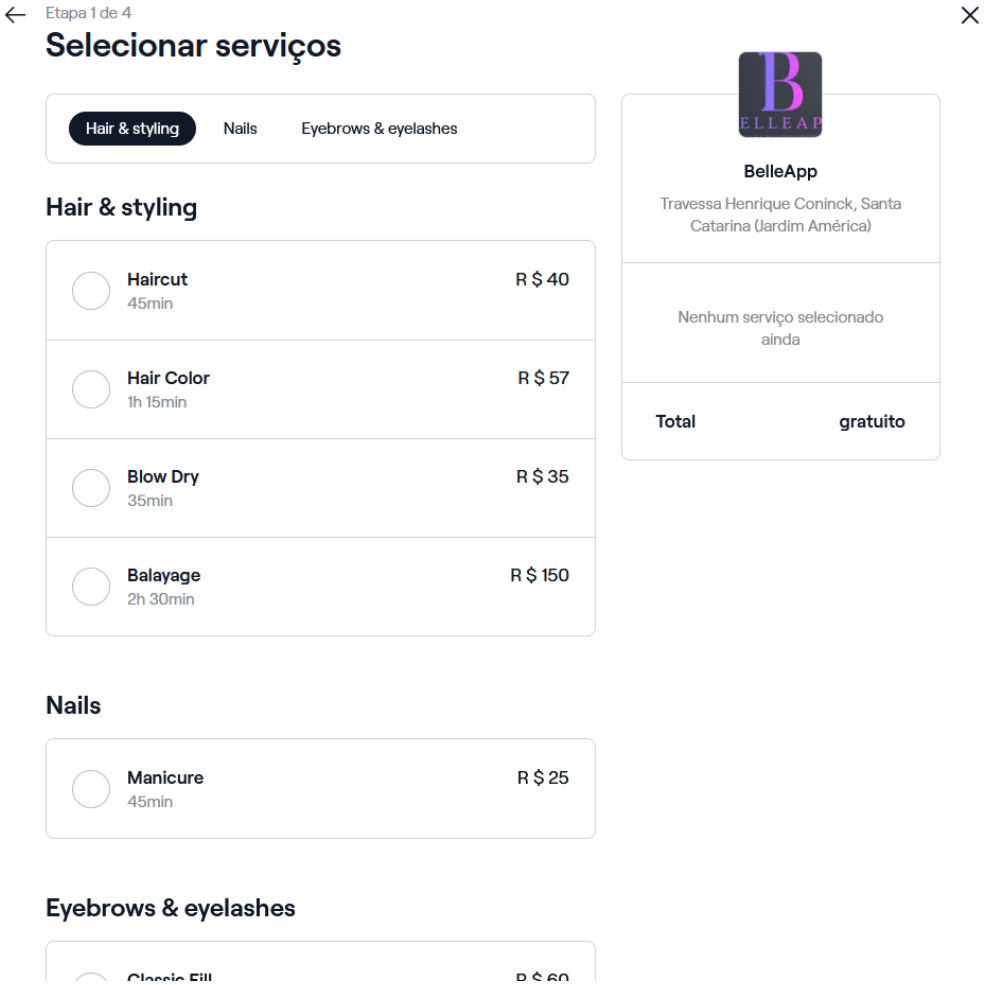
Figura 5 – Software Fresha – Home



Fonte: acervo do autor (2023)

Com o aplicativo é possível gerar um *link* onde o cliente pode fazer o agendamento de forma mais simples e rápida. Também é possível baixar um aplicativo para dispositivos móveis. No aplicativo o cliente consegue consultar todas as empresas que utilizam o Fresha, podendo utilizar de filtros como nome do estabelecimento, localização atual, data e horário. Assim consegue buscar pela empresa que está mais perto ou que possua horário para determinada data ou horário.

Figura 6 – Software Fresha – Agendamento do Cliente



Fonte: acervo do autor (2023)

A Figura 6 mostra a interface do usuário cliente, mostrando a rotina de agendamento de serviços onde é possível selecionar uma empresa, visualizar os serviços prestados e selecionar um além de escolher a data. Também no canto direito temos os detalhes do agendamento de forma resumida.

3.1.2 Trinks

O Trinks é uma aplicação desenvolvida para otimizar os processos de agendamento e gerenciamento em salões de beleza. Essa plataforma oferece aos clientes a capacidade de marcar compromissos com profissionais especializados em diferentes serviços de beleza e estética. Por meio do Trinks, os clientes podem explorar uma ampla variedade de serviços disponíveis, como cortes de cabelo, colorações, tratamentos faciais, manicures e pedicures. (TRINKS, 2023)

Os clientes têm a opção de escolher os profissionais que melhor se adequam às suas preferências, considerando as especialidades, avaliações e disponibilidade de agenda. A aplicação também facilita o processo de avaliação e *feedback*, permitindo que os clientes expressem sua opinião sobre a qualidade dos serviços recebidos.

Na Figura 7 podemos verificar que ao selecionar o serviço e a data é possível escolher o profissional e o horário desejado, além de ser possível colocar uma observação para o agendamento.

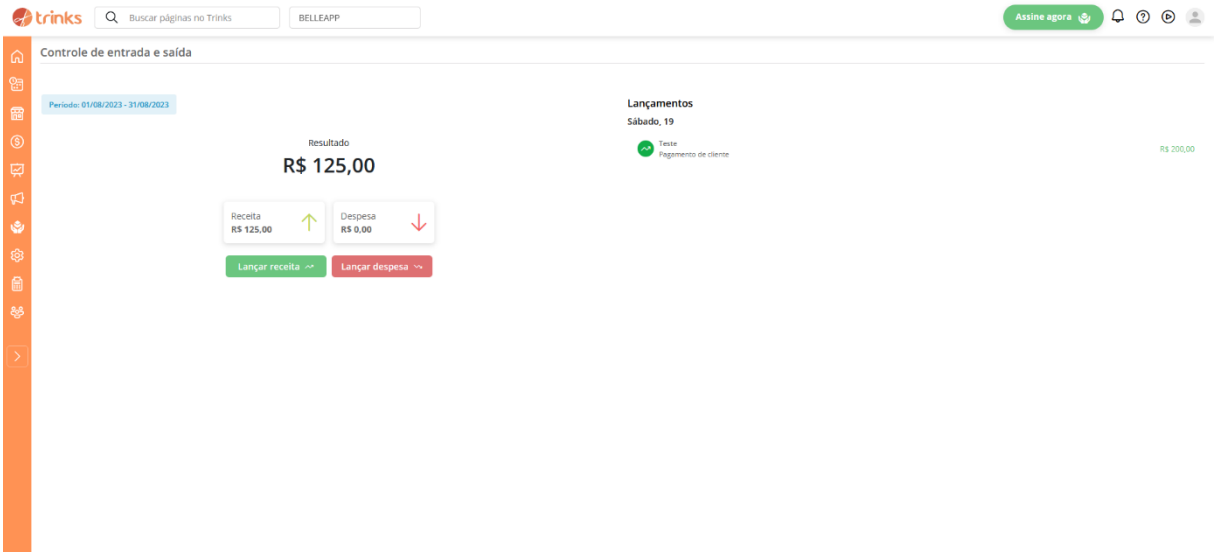
Figura 7 – Trinks – Agendamento do Cliente

Fonte: acervo do autor (2023)

Além disso, o Trinks foi projetado para simplificar a gestão interna dos salões. Ele oferece uma ferramenta de agenda para profissionais, permitindo que eles visualizem e gerenciem seus compromissos de forma eficiente. A plataforma também pode ser equipada com recursos de administração para os proprietários de salões, ajudando-os a gerenciar funcionários,

serviços e estoques. Na Figura 8 podemos ver a rotina de controle financeiro, onde é possível adicionar receitas e despesas, além de visualizar os lançamentos realizados.

Figura 8 – Trinks – Controle Financeiro



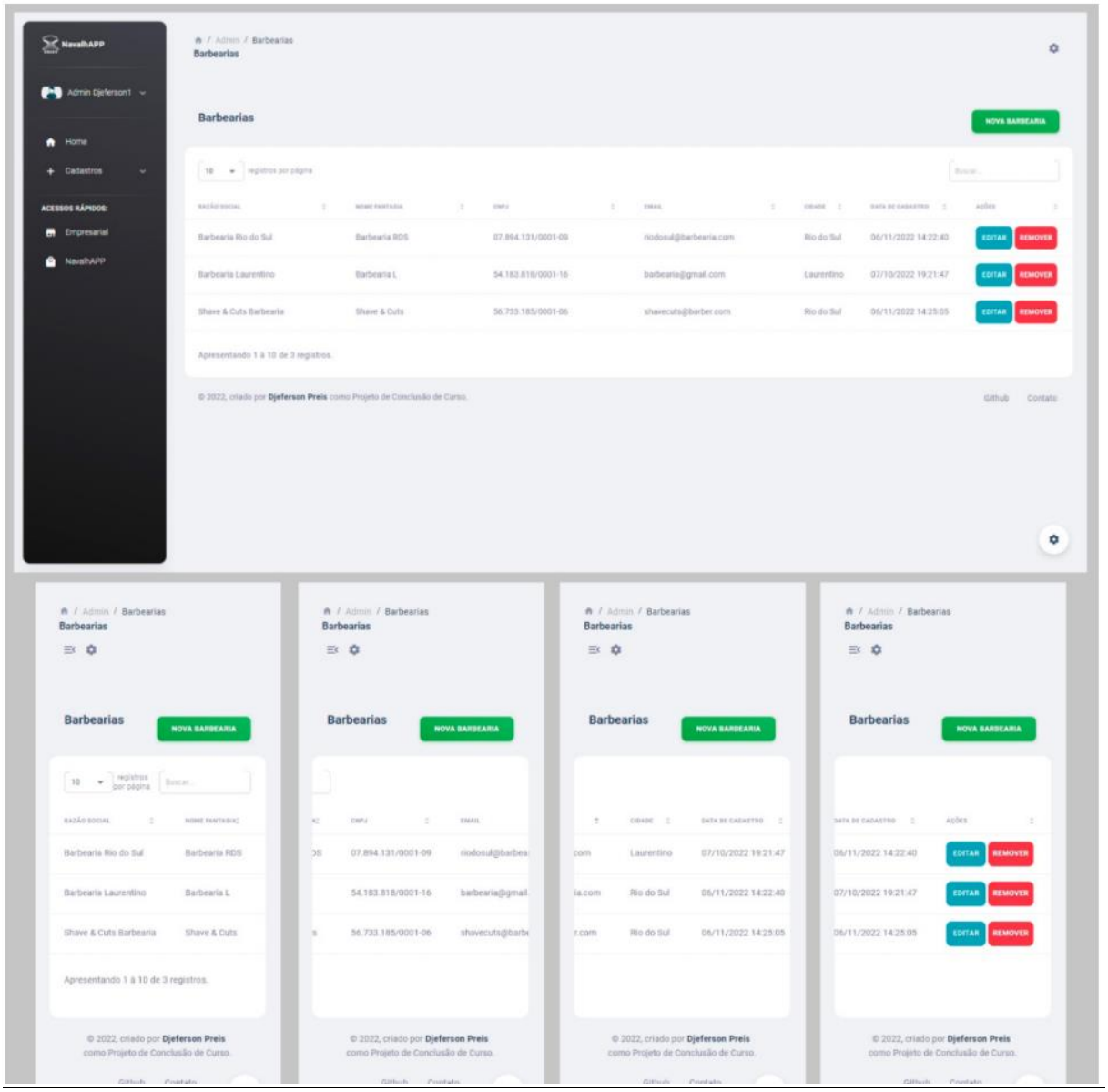
Fonte: acervo do autor (2023)

Uma desvantagem identificada no Trinks é o seu layout, onde o menu apresenta um tamanho ligeiramente menor em comparação com o de seus concorrentes. Além disso, o menu permanece estático, o que pode resultar em limitações durante a navegação, especialmente quando se lida com consultas que envolvem uma quantidade considerável de dados. Quando a barra de rolagem é movida para o final da página, é impossível selecionar qualquer item do menu, uma vez que eles permanecem fixos no cabeçalho do site.

3.1.3 Navalhapp

O protótipo Navalhapp, possui um foco no gerenciamento de agendamentos por parte do funcionário e a solicitação do agendamento do cliente de forma online. Todo o aplicativo é responsivo e pode ser acessado em diferentes plataformas *web*s.

Figura 9 - Navalhapp



Fonte: Preis, Butzke e Bastos (2023, p. g. 154)

Como mencionado, o Navalhapp (Figura 9) possui uma interface responsiva, sendo possível acessar em qualquer equipamento com um navegador web. Ele conta com um menu lateral onde é possível acessar as funcionalidades do mesmo. Em cada submenu possui uma tabela onde é possível filtrar e ordenar os registros, além de pode alterar ou excluir os registros.

Uma desvantagem do mesmo é que só possui rotinas de agendamentos, não tendo um diferencial em relação aos outros sistemas pesquisados.

4. BELLEAPP: PROTÓTIPO DE APLICATIVO PARA GERENCIAMENTO DE SERVIÇOS NA ÁREA DE BELEZA E ESTÉTICA

Neste capítulo serão abordados todos os passos que foram executados durante o processo de análise e desenvolvimento do protótipo.

4.1 ANÁLISE

A análise de um software é um dos primeiros passos para a criação de uma aplicação. Nela temos a engenharia de requisitos que conforme explica Sommerville (2011, p.g. 24), “Especificação de software ou engenharia de requisitos é o processo de compreensão e definição dos serviços requisitados do sistema e identificação de restrições relativas à operação e ao desenvolvimento do sistema.”.

Nos tópicos subsequentes, serão abordados detalhadamente o conceito do protótipo, seus requisitos, suas regras de negócio e os seus diagramas.

4.1.1 Visão Geral

O protótipo tem como principal finalidade oferecer suporte às empresas do setor de beleza e estética, para salões de beleza e clínicas de estética. O protótipo é divido em um aplicativo cliente, para os clientes, e um sistema administrativo para a empresa.

O sistema web trará funcionalidades essenciais para a gestão eficaz desses negócios, incluindo o registro de serviços oferecidos e informações dos funcionários sendo possível também gerenciar mais de uma empresa com o mesmo login. Já o aplicativo mobile, oferece uma ferramenta simplificada de agendamento online, com a inclusão de uma lista de espera. Neste caso, não havendo mais horários disponíveis com os parâmetros desejados, o cliente tem a opção de entrar em uma lista de espera. Quando surgir um horário disponível com as mesmas especificações, o cliente terá prioridade para ocupá-lo.

O aplicativo mobile também tem funcionalidades para realizar o *feedback* dos atendimentos, verificar o seu histórico de agendamentos e cancelamento do agendamento, dando assim mais liberdade para o usuário sem a necessidade de confirmação por parte da empresa.

Este protótipo aborda diversas necessidades-chave, auxiliando as empresas do setor a melhorar sua eficiência operacional e proporcionar uma experiência mais conveniente para seus clientes. Com a capacidade de gerenciar informações internas de forma mais eficaz e oferecer

a opção de agendamento online, ela tem o potencial de otimizar o funcionamento das empresas de beleza e estética, ao mesmo tempo em que melhora a satisfação dos clientes ao proporcionar uma maneira mais conveniente de acessar os serviços oferecidos.

4.1.2 Comparação do Protótipo com o Estado da Arte

Considerando os sistemas apresentados no estado da arte, no Quadro 5 são apresentadas algumas comparações no que se refere a disponibilidade de recursos entre os sistemas.

Quadro 5 – Recursos das aplicações pesquisadas

Recurso	BelleAPP	Fresha	Trinks	Navalhapp
Agendamentos Online	X	X	X	X
Agenda da Empresa	X	X	X	X
Lista de Espera para os Agendamentos	X			
Gerenciamento de Várias Empresas com um login	X			

Fonte: Acervo do autor (2023)

4.1.3 Requisitos Funcionais (RF)

Nesta seção, são apresentados os requisitos funcionais, que para Sommervile (2011, p. 59) são “declarações de serviços que o sistema deve fornecer, de como o sistema deve reagir a entradas específicas e de como o sistema deve se comportar em determinadas situações. Em alguns casos, os requisitos funcionais também podem explicitar o que o sistema não deve fazer.”.

No Quadro 6, temos os requisitos funcionais do sistema administrativo, sendo que na coluna da direita são apresentadas as regras de negócio relacionadas aos respectivos requisitos. As regras de negócio são detalhadas no Quadro 11.

Quadro 6 – Requisitos Funcionais (Sistema Administrativo)

Número	Nome	Descrição	RNs
RF01	Login Sistema Administrativo	O sistema administrativo deverá dispor de uma rotina para o usuário fazer o login para iniciar a aplicação.	RN01, RN02, RN03, RN06
RF02	Logout Sistema Administrativo	O sistema administrativo deverá dispor de uma rotina para permitir que o usuário seja desconectado da aplicação.	
RF03	Cadastro de Usuário	O sistema administrativo deverá dispor em sua tela de login, uma opção para o usuário se registrar.	RN07
RF04	Cadastro de Empresa	O sistema administrativo deverá dispor de uma rotina para o cadastro de empresa.	RN08

RF05	Cadastro de Funcionamento da Empresa	O sistema administrativo deverá dispor de uma rotina para informar os dias da semana e horários de funcionamento da empresa.	
RF06	Cadastro de Serviços da Empresa	O sistema administrativo deverá dispor de uma rotina para cadastrar os serviços que a empresa oferta.	
RF07	Cadastro de Funcionários da Empresa	O sistema administrativo deverá dispor de uma rotina para cadastrar os funcionários da empresa.	RN09
RF08	Troca de Empresa	O sistema administrativo deverá ter uma funcionalidade para que o funcionário possa trocar a empresa, assim poderá trabalhar ou possuir mais de uma empresa e com apenas dois cliques, gerenciar outra empresa.	
RF09	Cadastro de Serviços do Funcionários	O sistema administrativo deverá dispor de uma rotina para relacionar os Funcionários da empresa com os serviços prestados.	
RF10	Horários Disponíveis	O sistema administrativo deverá dispor de uma rotina para consultar os horários disponíveis.	
RF11	Cadastro de Restrições de Agenda	O sistema administrativo deverá dispor de uma rotina para cadastrar restrições da agenda da empresa informando que não terá expediente no dia e horários informados.	RN11
RF12	Cancelamento do Agendamento (Profissional)	O sistema administrativo deve permitir que o profissional cancele um agendamento vinculado a ele.	RN13

Fonte: Acervo do autor (2023)

No Quadro 7 temos os requisitos funcionais do aplicativo cliente, e da mesma forma na última coluna são apresentadas as regras de negócio relacionadas.

Quadro 7 – Requisitos Funcionais (Aplicativo Cliente)

Número	**Nome**	**Descrição**	**RNs**
RF13	Login Aplicativo Cliente	O aplicativo cliente deverá dispor de uma rotina para o usuário fazer o login para iniciar a aplicação.	RN02, RN03, RN06
RF14	Logout Aplicativo Cliente	O aplicativo cliente deverá dispor de uma rotina para permitir que o usuário seja desconectado da aplicação.	
RF15	Cadastro de Usuário	O aplicativo cliente deverá dispor em sua tela de login, uma opção para o usuário se registrar.	RN07
RF16	Cadastro de Agendamentos	O aplicativo cliente deverá dispor de uma rotina para cadastrar agendamentos.	RN04, RN10
RF17	Lista de Espera	O aplicativo cliente deverá dispor de uma lista de espera para os agendamentos.	
RF18	Horários Disponíveis	O aplicativo cliente deverá dispor de uma rotina para o usuário cliente, consultar os horários disponíveis após selecionar a empresa e o funcionário desejado para o serviço.	
RF19	Cancelamento do Agendamento	O aplicativo cliente deverá dispor de uma funcionalidade para cancelar os agendamentos.	
RF20	Histórico de Agendamentos	O aplicativo cliente deve manter um histórico completo dos agendamentos passados de cada cliente, permitindo que eles visualizem serviços anteriores.	
RF21	Feedback Pós-Serviço	O aplicativo cliente deve permitir que os clientes forneçam feedback sobre os serviços recebidos.	

Fonte: Acervo do autor (2023)

O Quadro 8 e o Quadro 9 apresentam os requisitos funcionais opcionais do Sistema Administrativo e do Aplicativo Cliente respectivamente, que, embora planejados, não serão desenvolvidos na versão inicial da aplicação. Tais requisitos constituem melhorias e recursos adicionais que podem ser implementados em futuras versões para enriquecer a experiência do usuário e expandir as funcionalidades da aplicação.

Quadro 8 – Requisitos Funcionais Opcionais (Sistema Administrativo)

Número	**Nome**	**Descrição**
RF22	Cadastro de Receitas da Empresa	O sistema administrativo poderá dispor de uma rotina para as receitas da empresa.
RF23	Cadastro de Produtos	O sistema administrativo poderá dispor de uma rotina para cadastrar produtos para a empresa.
RF24	Cadastro de Despesas da Empresa	O sistema administrativo poderá dispor de uma rotina para cadastrar as despesas da empresa.
RF25	Relatórios de Receita e Faturamento	O sistema administrativo poderá gerar relatórios detalhados sobre a receita e faturamento, incluindo a análise de tendências ao longo do tempo e a categorização por tipo de serviço.
RF26	Relatórios de Retenção de Clientes	O sistema administrativo poderá fornecer relatórios que acompanhem a taxa de retenção de clientes, identificando clientes recorrentes e padrões de agendamentos.
RF27	*Dashboard* de Tendências de Agendamento	O sistema administrativo poderá oferecer um *dashboard* que exiba as tendências de agendamento ao longo do tempo, ajudando a identificar padrões sazonais ou de demanda.
RF28	*Dashboard* de Avaliações e Feedback dos Clientes	O sistema administrativo poderá apresentar um *dashboard* dedicado às avaliações e feedback dos clientes, permitindo uma análise rápida da satisfação.
RF29	Previsão de Demanda e Ocupação	O sistema administrativo poderá empregar técnicas de IA para prever a demanda por serviços e a ocupação dos profissionais, ajudando na otimização da agenda.
RF30	Taxas e Descontos	O sistema administrativo poderá ser capaz de aplicar taxas ou descontos especiais aos serviços, conforme as promoções ou políticas da empresa.
RF31	Registro de Atividades do Usuário	O sistema administrativo poderá registrar as atividades dos usuários, incluindo ações como agendamentos, cancelamentos, alterações de perfil e interações com o sistema.
RF32	Registro de Acessos e Autenticações	O sistema administrativo poderá registrar os acessos de usuários, rastreando tentativas de login bem-sucedidas e malsucedidas.
RF33	Registros de Erros e Exceções	O sistema administrativo poderá registrar erros, exceções e problemas técnicos que ocorrem durante a operação normal do sistema.
RF34	Rastreabilidade de Dados Sensíveis	O sistema administrativo poderá registrar o acesso e uso de dados sensíveis, como informações de pagamento ou dados pessoais dos clientes.
RF35	Cadastro de Promoções Especiais	O sistema administrativo poderá dispor de uma rotina para cadastros de Promoções Especiais.

Fonte: Acervo do autor (2023)

Quadro 9 – Requisitos Funcionais Opcionais (Aplicativo Cliente)

Número	Nome	Descrição
RF36	Notificações do Cliente	O aplicativo cliente poderá fornecer opções no cadastro do usuário para que ele possa escolher as formas de notificação que deseja habilitar.
RF37	Notificações via E-mail para Clientes	O aplicativo cliente poderá enviar notificações via e-mail para os clientes sempre que houver uma atualização ou evento relevante em relação aos serviços agendados ou ao estado da conta do cliente.
RF38	Notificações via WhatsApp para Clientes	O aplicativo cliente poderá dispor de recurso para enviar notificações instantâneas aos clientes por meio do aplicativo WhatsApp, informando sobre eventos relevantes relacionados aos serviços agendados ou ao estado da conta do cliente. Essas notificações devem ser enviadas para o número de telefone associado à conta do cliente.
RF39	Integração WhatsApp	O aplicativo cliente poderá ser integrado ao WhatsApp para permitir consultas sobre horários disponíveis em datas desejadas e habilitar o agendamento automático dos clientes por meio do aplicativo.
RF40	Notificações de Promoções Especiais	O aplicativo cliente poderá dispor de recurso para enviar notificações aos clientes sobre promoções especiais, descontos ou eventos exclusivos.
RF41	Suporte ao Cliente Integrado	O aplicativo cliente poderá oferecer um canal de suporte ao cliente, onde os usuários possam fazer perguntas, reportar problemas ou obter assistência.
RF42	Aviso de Cancelamento do Profissional	O aplicativo cliente poderá dispor de uma funcionalidade para quando um profissional cancelar um agendamento, o sistema deve notificar o cliente afetado para evitar inconvenientes.
RF43	Notificações de Aniversário	O aplicativo cliente poderá enviar notificações de feliz aniversário aos clientes, como um gesto de apreço.
RF44	Notificações de Serviços Recomendados	O aplicativo cliente poderá com base no histórico de agendamentos e preferências do cliente, o sistema deve enviar notificações sugerindo serviços relevantes.
RF45	Recomendações Personalizadas de Serviços	O aplicativo cliente poderá utilizar técnicas de IA para analisar o histórico de agendamentos dos clientes e oferecer recomendações personalizadas de serviços com base em suas preferências.
RF46	Processamento de Pagamentos Online	O aplicativo cliente poderá permitir que os clientes efetuem pagamentos online de forma segura e conveniente ao agendar serviços.
RF47	Múltiplas Opções de Pagamento	O aplicativo cliente poderá aceitar uma variedade de métodos de pagamento, como cartões de crédito, débito, PayPal e outros.
RF48	Histórico de Transações	O aplicativo cliente poderá manter um histórico detalhado de todas as transações de pagamento realizadas pelos clientes.
RF49	Recibo Eletrônico de Pagamento	O aplicativo cliente poderá gerar e enviar automaticamente recibos eletrônicos por e-mail para os clientes após a conclusão do pagamento.
RF50	Registro de Atividades do Usuário	O aplicativo cliente poderá registrar as atividades dos usuários, incluindo ações como agendamentos, cancelamentos, alterações de perfil e interações com o sistema.
RF51	Registro de Acessos e Autenticações	O aplicativo cliente poderá registrar os acessos de usuários, rastreando tentativas de login bem-sucedidas e malsucedidas.
RF52	Registros de Erros e Exceções	O aplicativo cliente poderá registrar erros, exceções e problemas técnicos que ocorrem durante a operação normal do sistema.
RF53	Rastreabilidade de Dados Sensíveis	O aplicativo cliente poderá registrar o acesso e uso de dados sensíveis, como informações de pagamento ou dados pessoais dos clientes.
RF54	Relatório de Dados Pessoais conforme a LGPD	O aplicativo cliente poderá fornecer aos usuários a opção de solicitar um relatório contendo todos os dados pessoais coletados e armazenados de acordo com as disposições da Lei Geral de Proteção de Dados (LGPD).

Fonte: Acervo do autor (2023)

4.1.4 Requisitos Não Funcionais (RNF)

Nesta seção temos os Requisitos Não Funcionais que para Sommerville (2011, p. 59), são descritos como “restrições aos serviços ou funções oferecidos pelo sistema. Incluem restrições de timing, restrições no processo de desenvolvimento e restrições impostas pelas normas. Os requisitos não funcionais, muitas vezes, aplicam-se ao sistema como um todo.”. No Quadro 10 temos os RNF deste protótipo.

Quadro 10 – Requisitos não Funcionais

Número	Nome	Descrição
RNF01	Sistema Administrativo	O sistema administrativo será desenvolvido para navegador utilizando da linguagem JS e do *framework* React.js.
RNF02	Aplicativo Cliente	O aplicativo do cliente será desenvolvido para Android e IOS.
RNF03	Autenticação	Toda a aplicação deverá possuir autenticação, não podendo fazer nada sem estar autenticado, tanto para o painel de administração da empresa quanto para o aplicativo do usuário final.
RNF04	Banco de Dados	O banco de dados deve utilizar o SGBD PostgreSQL.
RNF05	Usabilidade	O sistema deve ser intuitivo e de fácil utilização, mesmo por usuários com pouca experiência técnica. Uma interface amigável e fluxos de trabalho simplificados ajudarão a maximizar a produtividade e reduzir erros.
RNF06	*Frontend*	A linguagem utilizada para o desenvolvimento deve ser JavaScript utilizando das bibliotecas Next.js, TypeScript e TailwindCss.
RNF07	*Backend*	A linguagem utilizada para o desenvolvimento deve ser JavaScript utilizando das bibliotecas Node.js, Fastify e o Prisma como ORM.
RNF08	Suporte a Diferentes Plataformas	O sistema deve ser acessível e funcional em diferentes plataformas, incluindo dispositivos móveis, tablets e navegadores de desktop.

Fonte: Acervo do autor (2023)

4.1.5 Regras de Negócio (RN)

Nesta seção iremos abordar as regras de negócio que para Amoasei (2023, n. p.), “ela traduz uma necessidade do negócio em regras lógicas, permitindo que desenvolvimento, produto e negócio de uma empresa se alinhem com relação a estas necessidades, se guiem por elas e apliquem as regras da forma mais clara possível.”. No Quadro 11, são apresentadas as regras de negócio que serão aplicadas no desenvolvimento do protótipo.

Quadro 11 - Regras de Negócio

Número	Nome	Descrição
RN01	Autenticação – Painel Empresa	Apenas usuários administradores da empresa ou que está vinculado na empresa podem ter acesso as rotinas da empresa como agendamentos, produtos, clientes etc.
RN02	Senha	A senha deverá possuir no mínimo 8 dígitos, sendo eles números, letras ou caracteres especiais.
RN03	Login	O login do usuário deverá ser único, não podendo existir dois usuários com o mesmo login, sendo que será utilizado o e-mail.
RN04	Agendamento – Horário Coincidente	O sistema não poderá permitir dois agendamentos para um funcionário com horários coincidindo.
RN05	Admin. Da Empresa	Quando o usuário criar uma empresa automaticamente ele será o administrador dela.
RN06	Campos Login	Para efetuar o login, deverá informar o e-mail e senha.
RN07	Usuário	Para a realização dos cadastros de usuário deverá possuir os seguintes campos: e-mail (obrigatório), nome (obrigatório) e senha (obrigatório).
RN08	Empresa	Para a realização dos cadastros deverá possuir os seguintes campos: nome (obrigatório), telefone (obrigatório), e-mail (obrigatório), endereço (obrigatório). Deverá ser possível alterar, visualizar, ativar e desativar o registro.
RN09	Funcionários da Empresa	Para a realização dos cadastros deverá possuir os seguintes campos: usuário (obrigatório), situação (obrigatório), salário atual (obrigatório).
RN10	Agendamento	Para a realização dos cadastros deverá possuir os seguintes campos: Usuário (obrigatório), Serviço (obrigatório), Funcionário (obrigatório), data e hora (obrigatório), valor (obrigatório) (pegar do serviço), situação (1-Agendado; 2-Finalizado; 3-Cancelado). Deverá ser possível alterar, visualizar e cancelar um agendamento.
RN11	Restrições da Agenda	Para a realização dos cadastros deverá possuir os seguintes campos: Data (obrigatório), ativo (obrigatório), descrição (obrigatório), horário de início (obrigatório), horário de término (obrigatório),
RN12	Notificações E-mail	As notificações devem ser enviadas para o endereço de e-mail associado à conta do cliente.
RN13	Cancelamento do Agendamento (Profissional)	Quando o profissional cancelar um agendamento, deverá informar um motivo para o mesmo.

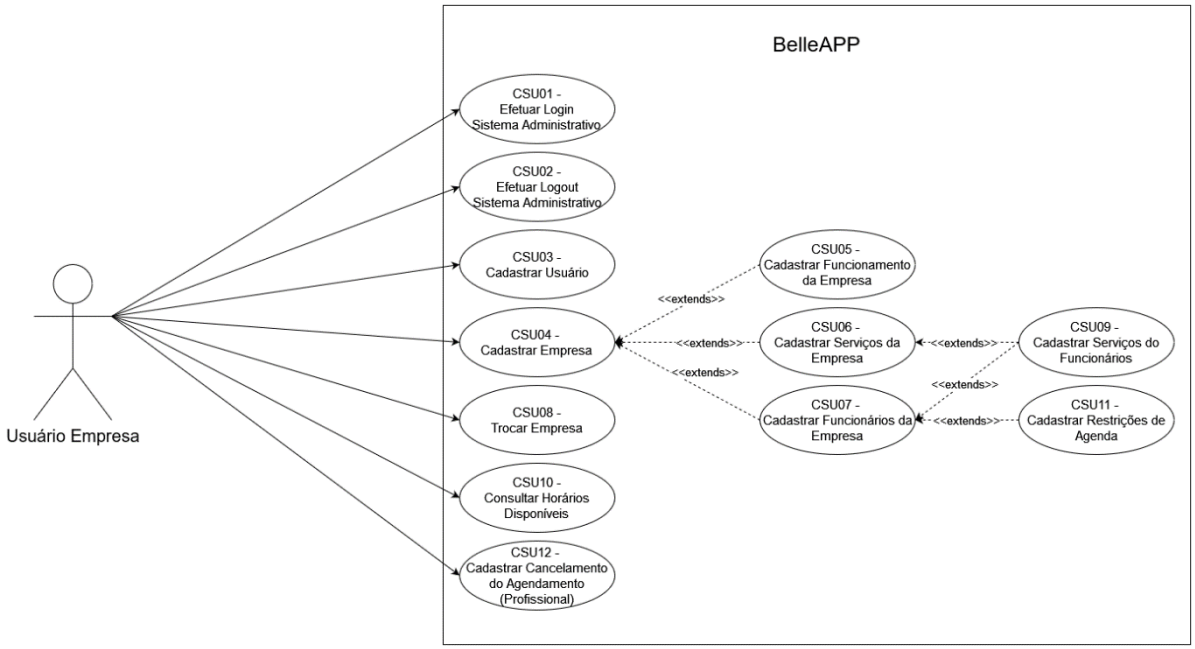
Fonte: Acervo do autor (2023)

4.1.5 Diagramas

Para facilitar a compreensão dos requisitos elencados e das funcionalidades do protótipo foram elaborados alguns diagramas. Os diagramas representam os recursos que estão disponíveis para os usuários, bem como o passo a passo para executar a atividade principal proposta por esse protótipo.

Na Figura 10 podemos ver um diagrama de caso de uso onde são apresentadas as funcionalidades do sistema administrativo e o ator que neste caso é o usuário da empresa.

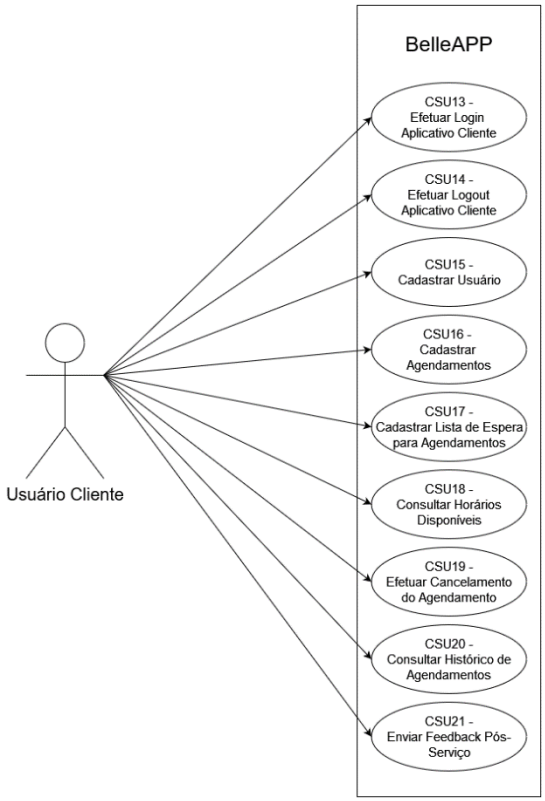
Figura 10 – Diagrama de Caso de Uso (Usuário Empresa)



Fonte: acervo do autor (2023)

Na Figura 11 outro é apresentado o diagrama de caso de uso com as funcionalidades disponíveis para o aplicativo cliente.

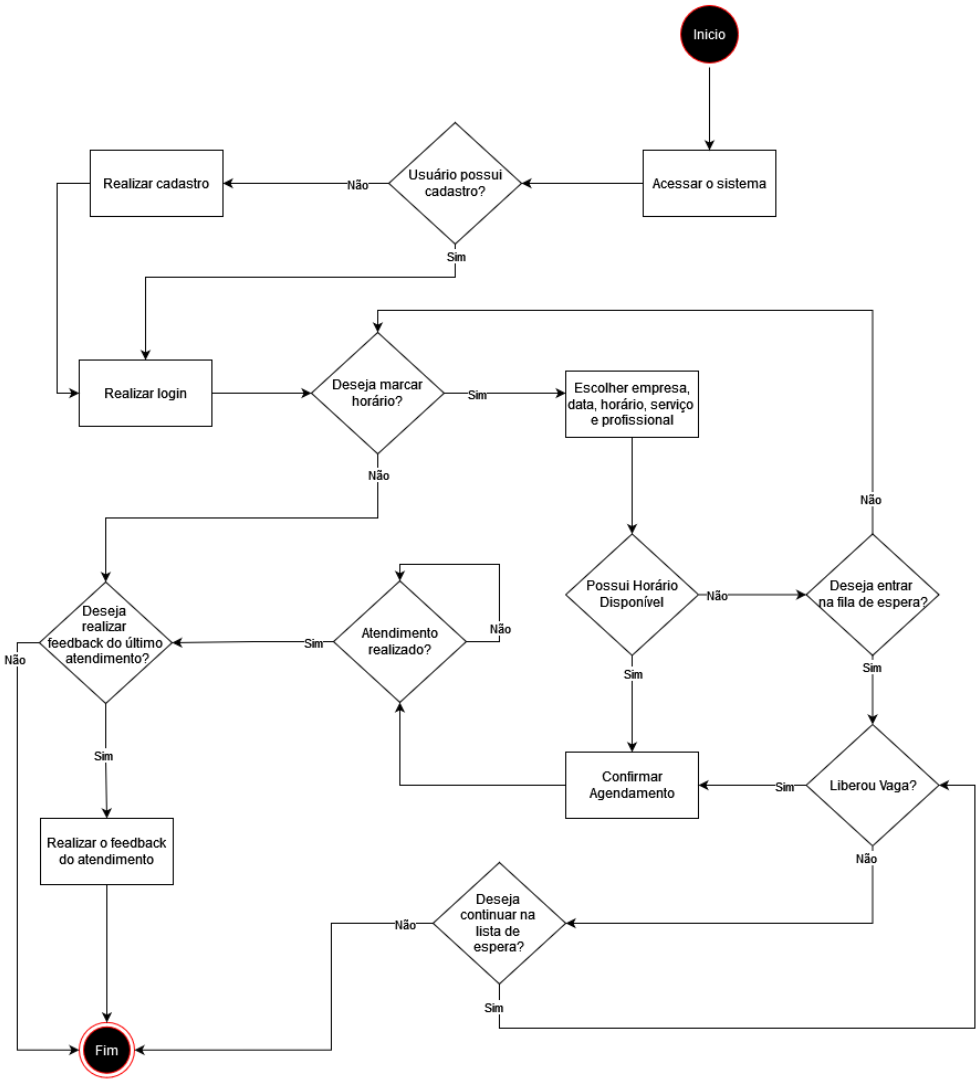
Figura 11 – Diagrama de Caso de Uso (Usuário Cliente)



Fonte: acervo do autor (2023)

Na Figura 12 é apresentado um diagrama de atividades, nele é possível perceber a entender a lógica de um agendamento que será realizado pelo usuário no aplicativo cliente.

Figura 12 – Diagrama de Atividade (Agendamento do Cliente)

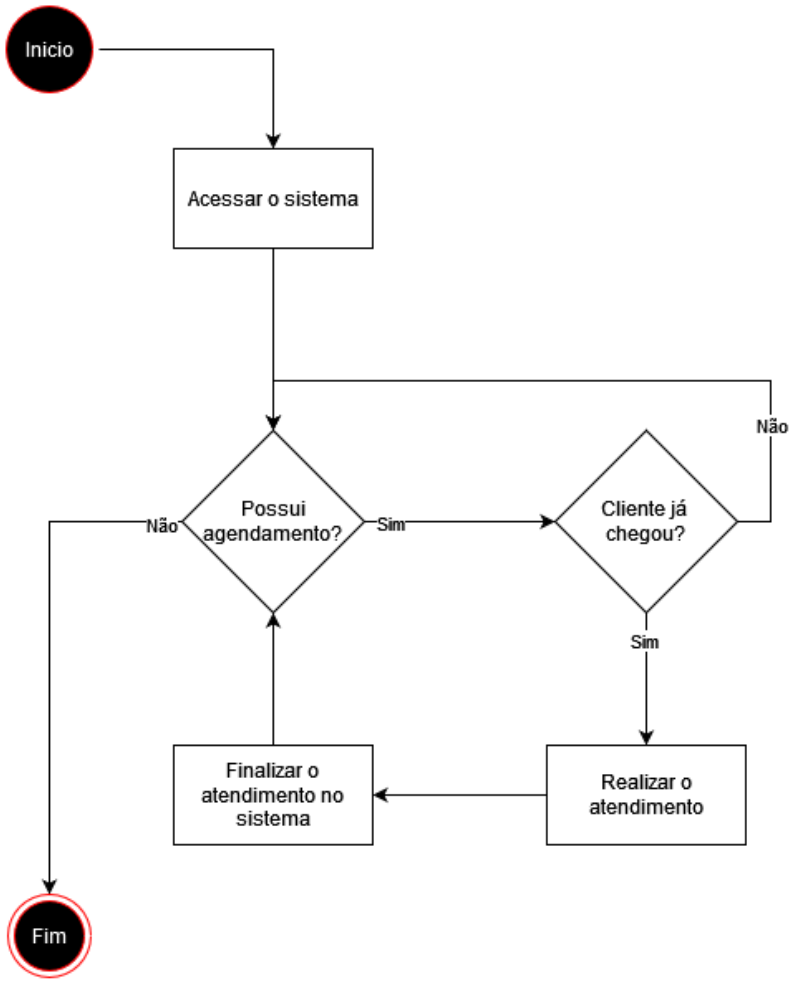


Fonte: acervo do autor (2023)

Conforme pode-se perceber caso o usuário ainda não possui um cadastro ele deverá primeiro fazê-lo. Logo em seguida após realizar o login pode-se fazer um agendamento, caso na data, horário e para o profissional selecionado não houver mais horários disponíveis o usuário poderá entrar para uma lista de espera, onde caso liberar uma vaga condizente com os horários e profissional este poderá confirmar o agendamento. Após o atendimento ou ao entrar no sistema o usuário poderá submeter um *feedback* aos atendimentos recebidos.

Na Figura 13 pode-se observar o diagrama de atividade para o atendimento de um cliente sendo realizado por um funcionário. Aqui ele deverá acessar o sistema, e verificar se possui agendamentos para ele, caso possuir deve aguardar o cliente chegar, realizar o atendimento e ao terminá-lo deve finalizar o atendimento no sistema.

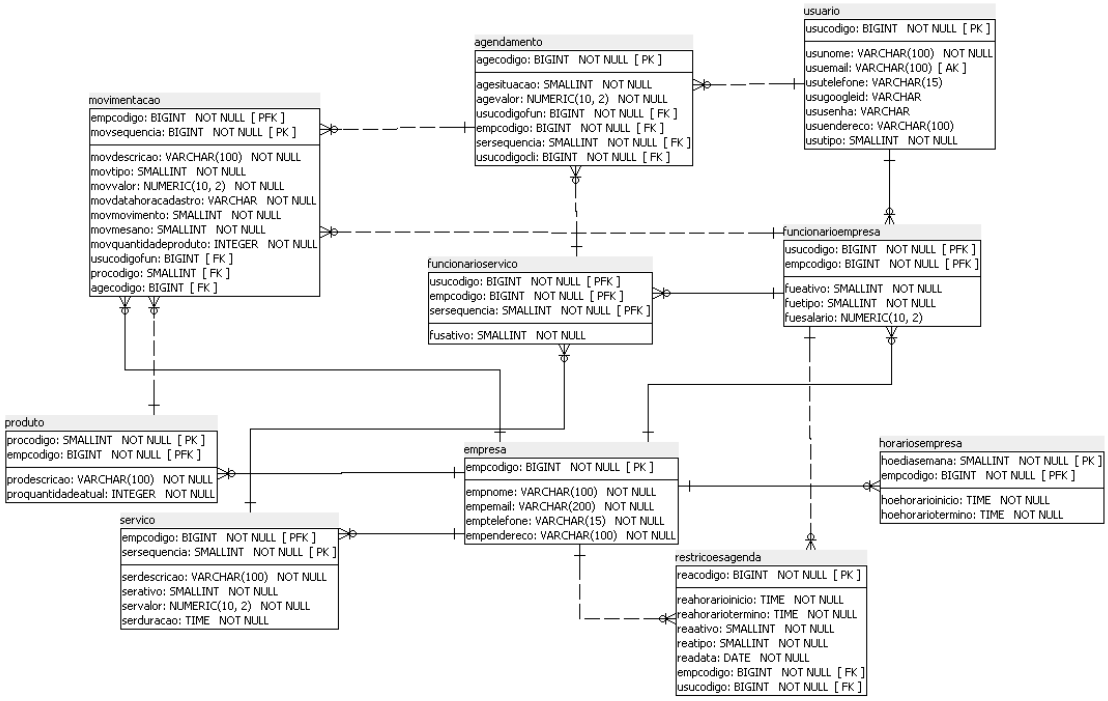
Figura 13 – Diagrama de Atividade (Atendimento)



Fonte: acervo do autor (2023)

Conforme explicado na Revisão da Literatura o diagrama de entidade relacionamento descreve a estrutura do Banco de Dados. Na Figura 14 temos o diagrama de entidade relacionamento elaborado para este protótipo, não contemplando os requisitos funcionais opcionais.

Figura 14 – Diagrama de Entidade Relacionamento



Fonte: acervo do autor (2023)

Foram criado 10 tabelas para salvar os dados de usuário, empresa, serviços, agendamentos, restrições da agenda, funcionários da empresa e os serviços prestados por cada funcionário.

Na próxima seção teremos a explicação da implementação do protótipo proposto.

4.2 IMPLEMENTAÇÃO

Nesta seção, será apresentado um resumo das ferramentas e tecnologias utilizadas no desenvolvimento do protótipo, bem como as rotinas implementadas para atender aos requisitos funcionais do projeto. Essas informações fornecerão uma visão geral do ambiente de desenvolvimento e das funcionalidades incorporadas ao sistema administrativo e ao aplicativo cliente.

4.2.1 Ferramentas e Tecnologias

Para a criação dos diagramas presentes na seção de análise foi utilizado a ferramenta Draw.io. Com ela foram desenvolvidos os diagramas de caso de uso e os diagramas de atividade. Já para a construção do banco de dados e do diagrama de entidade relacionamento

foi utilizado o SQL Power Architect. Para gerenciar os requisitos funcionais, requisitos não funcionais e as regras de negócio foi utilizado o Planilhas do Google (Google Sheets).

No desenvolvimento da plataforma WEB (sistema administrativo), foram utilizadas as tecnologias: HTML, CSS e JavaScript (JS). Para a estruturação semântica do site foi utilizado o HTML 5 que não é de fato uma linguagem de programação, mas sim uma linguagem de marcação de textos comumente utilizado em basicamente tudo que envolve sites. Para estilização fora utilizado o CSS3 com auxílio do TailwindCss que é uma biblioteca de classes, onde ao invés de fornecer um conjunto de estilização para um componente como faz o *Bootstrap*, o TailwindCss fornece classes unitárias onde cada classe é referente a uma estilização. Na Figura 15 podemos verificar a utilização do TailwindCss em um determinado componente.

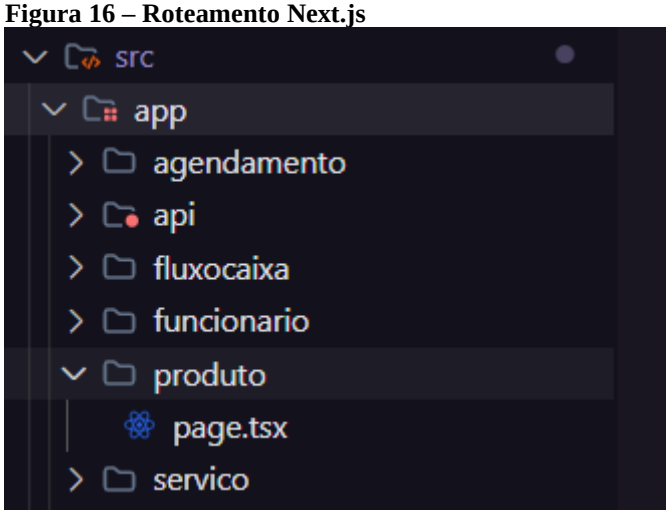
Figura 15 – Exemplo do TailwindCss

```
return (
  <div className="flex h-16 w-full flex-col bg-gray-300 md:space-y-4">
    <header className="z-40 flex h-16 w-full items-center justify-between">
      <div className="ml-6 block lg:hidden">
        <button className="text-md flex items-center rounded-full bg-white p-2 text-gray-500 shadow">
        </button>
      </div>
      <div className="relative z-20 flex h-full  justify-end px-3 md:w-full">
        <div className="relative flex w-full items-center justify-end space-x-4 p-1">
          <Profile avatarUrl={avatarUrl} name={name} />
        </div>
      </div>
    </header>
  </div>
```

Fonte: acervo do autor (2023)

Já como linguagem principal, o JavaScript, faz todo o trabalho de montar os componentes do site e fazer todos os comportamentos necessários. Para auxiliar no *front-end* foi utilizado como framework o Next.js. Com ele é possível criar componentes que são renderizados no lado do servidor, diminuindo assim a carga de trabalho do *browser*, ou renderizados no lado do cliente. O principal aspecto dele é o seu roteamento, pois ele permite criar diferentes URLs tornando mais fácil direcionar solicitações HTTP para funções específicas. Para criar uma rota, dentro da pasta app deve ser criado uma pasta e dentro um arquivo ‘page.tsx’. Na Figura 16 podemos ver a estruturação das pastas. Ao acessar a URL com a terminação ‘/produto’ temos acesso a rota do produto, onde o que estiver no componente ‘page.tsx’ da pasta produto será renderizada para o usuário.

Figura 16 – Roteamento Next.js



Fonte: acervo do autor (2023)

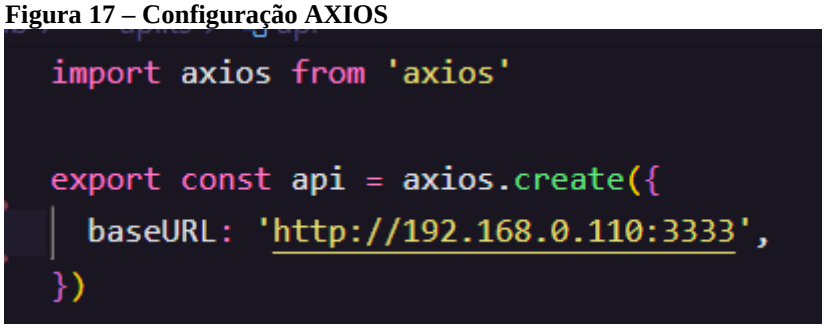
Para montar os componentes é utilizado o React, ele também é uma biblioteca para o JS e de código aberto. Ele foi utilizado para a criação dos componentes de forma interativa e reativa e juntamente com o Next.js ele é fundamental para o desenvolvimento em aplicações web. As principais características do React são: Componentização, Reatividade, Comunidade Ativa, Ecossistema Rico, entre outros. Os arquivos do React possuem as extensões *.js* ou *.jsx*, no entanto ao utilizar o TS é comumente utilizado as extensões *.tsx*, como mencionado anteriormente. O TypeScript (TS), é uma linguagem de programação utilizada como uma extensão do JS. Com ela é possível adicionar tipagem estática e recursos de programação orientada a objetos ao JS.

Para verificar o usuário logado é utilizado o JWT (JSON Web Tokens) que basicamente é utilizado como forma de autenticação de um usuário, ele contém os dados passado por uma estrutura em JSON com uma assinatura digital para garantir a integridade e segurança.

No ambiente mobile (aplicativo cliente) foi utilizado o React Native, que é um *framework* destinado ao desenvolvimento mobile sendo uma extensão do React. Com o React Native é possível criar aplicativos para Android e iOS utilizando uma base de código comum. Da mesma forma que o React, também é possível a reutilização de componentes, possui uma comunidade ativa e possui também integrações com códigos nativos (Java e Kotlin para o Android e Objective-C e Swift para o iOS. Foi utilizado também o Expo para facilitar no desenvolvimento, pois ele fornece muitas ferramentas, não é necessário a configurações de ambientes, possui um emulador, chamado Expo GO, onde ao instalar no dispositivo mobile é possível visualizar em tempo de execução o que se está sendo desenvolvido. Também possui uma comunidade ativa, muita documentação e é código aberto sendo possível a utilização sem a necessidade de uma licença.

Como ferramenta para integração entre a aplicação WEB com a API e do mobile com a API, foi utilizado o AXIOS que é uma biblioteca especializada em fazer requisições HTTP. Ele é amplamente utilizado, pois se torna fácil de implementar, é mais eficiente, possui funcionalidades específicas para cada método HTTP (GET, POST, PUT, DELETE, PATCH), ele também possui uma comunidade ativa e a sua documentação é abundante. Na Figura 17 temos a configuração da biblioteca e na Figura 17 temos a utilização da biblioteca para realizar uma solicitação do tipo GET.

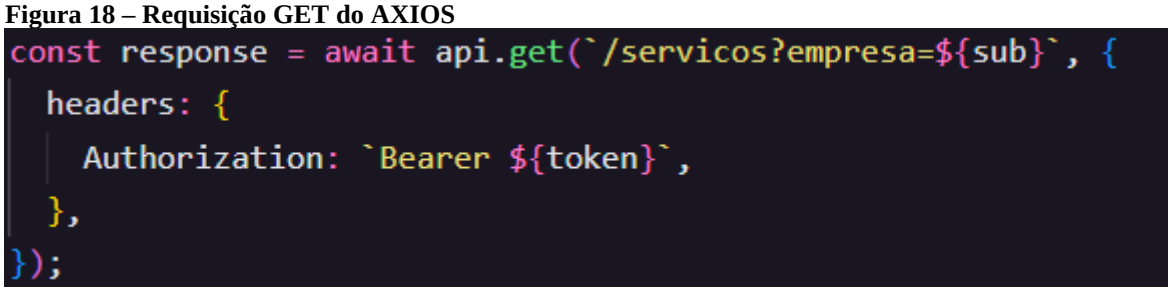
Figura 17 – Configuração AXIOS



Fonte: acervo do autor (2023)

Na configuração apenas se faz necessário exportar uma constante chamada API. Deve ser importado o axios e criado um objeto definindo uma URL base. No axios também existem outras configurações opcionais como *params*, *headers* e *timeout*.

Figura 18 – Requisição GET do AXIOS



Fonte: acervo do autor (2023)

Para fazer uma requisição GET, apenas precisamos importar a API que foi mencionada na Figura 18, e chamar o método '*get*' passando a URL e caso tiver *headers*, *params* ou outras configurações necessárias para a conexão correta entre os serviços.

Já na API foi utilizado o Prisma.io como ORM, o Fastify como *framework* principal para fazer os controles de rotas, o Axios e o JWT já mencionado acima e a plataforma Supabase para gerenciar o banco de dados. O Prisma.io é um ORM e fornece uma camada de abstração entre o código do aplicativo e o banco de dados. Com ele é possível fazer todas as operações

do banco como *insert, update, delete*, também é possível criar as tabelas, campos, definir *PK, FK, unique*, entre outras funcionalidades. O Fastify é muito utilizado em APIs, pois é muito simples de configurar e definir as rotas, além de ser de código aberto e possuir uma comunidade ativa. A plataforma Supabase possui funcionalidades para banco de dados, autenticação, armazenamento de arquivos, integração com serviços terceiros, entre outros recursos disponibilizados. Para este protótipo foi utilizado apenas o recurso de banco de dados.

E como motor para rodar toda a aplicação foi utilizado o Node.js, que permite a execução de códigos JS fora dos navegadores web utilizando o V8, também conhecido como *Chrome's V8 JavaScript engine*, que foi desenvolvida pela Google e é utilizado no Chrome.

4.2.2 Utilização e Funcionamento

O protótipo terá duas formas de acesso, um sistema administrativo (web) acessado pelo administrador da empresa e pelos funcionários da mesma, onde poderão gerenciar os serviços, os funcionários e os agendamentos e um aplicativo cliente (mobile) que será utilizado pelos clientes para realizarem os agendamentos, realizar *feedbacks* e consultar o histórico de agendamentos.

A seguir serão apresentadas as telas do protótipo começando pelo sistema administrativo e posteriormente pelo aplicativo cliente.

4.2.2.1 Sistema Administrativo

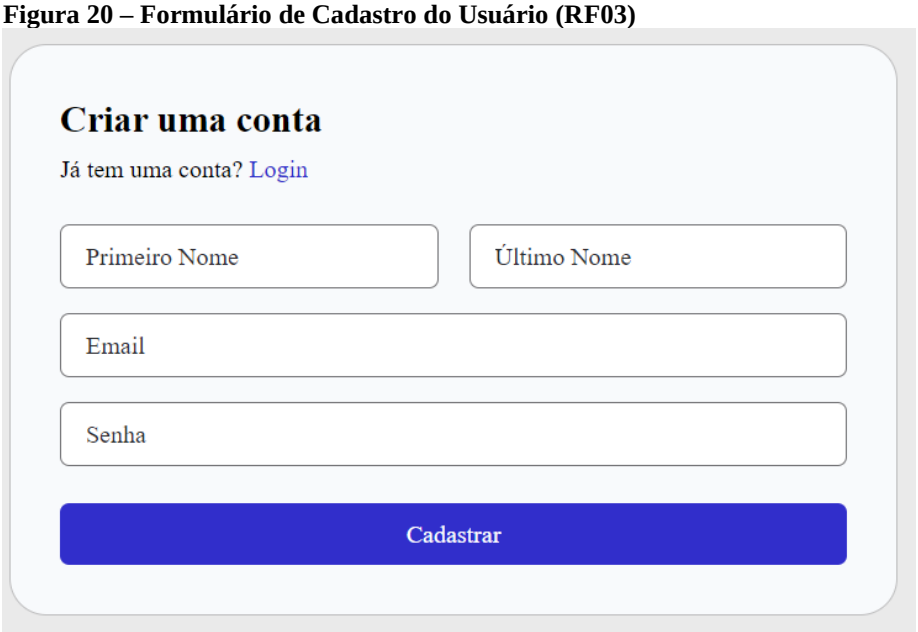
Para acessar o sistema o usuário deverá efetuar o *login* no sistema (RF01) informando um e-mail e uma senha. Na Figura 19 é apresentada a tela de *login* do sistema administrativo.

Figura 19 – *Login* Sistema Administrativo (RF01)

Fonte: Acervo do Autor (2023)

Caso o usuário não possua um cadastro para efetuar o *login*, o mesmo poderá se registrar no sistema (RF03), para isso ao clicar no link ‘Registre-se’ conforme visualizado na Figura 18, o sistema irá alterar os campos para que o usuário possa se registrar. Na Figura 20 temos o formulário para o registro, onde o usuário deverá informar o primeiro e último nome, e-mail e sua senha.

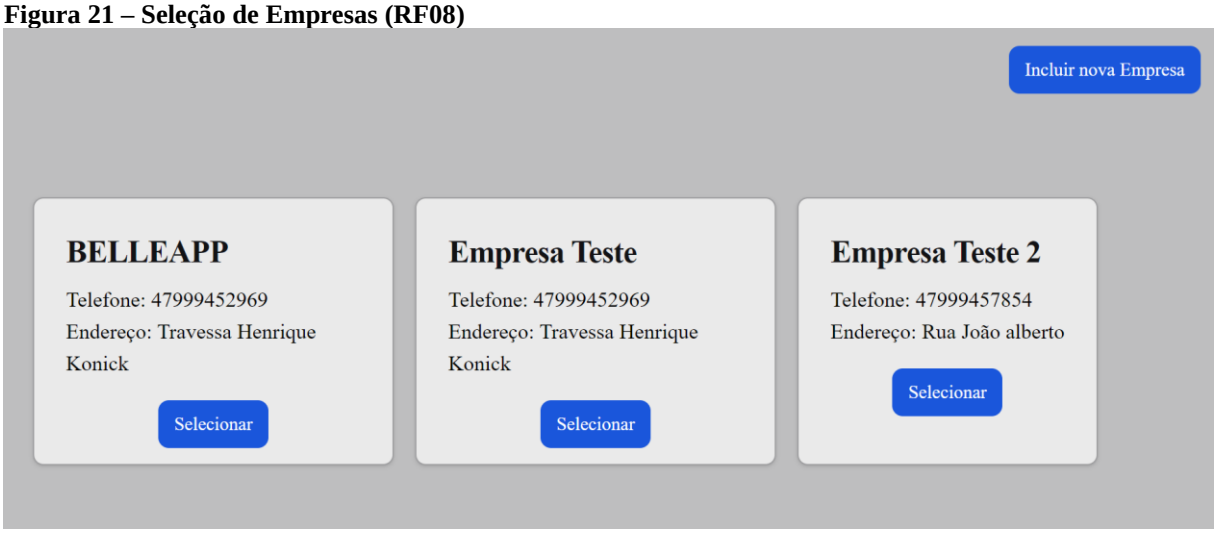
Figura 20 – Formulário de Cadastro do Usuário (RF03)



Fonte: Acervo do Autor (2023)

Após fazer o *login* no sistema administrativo, o usuário poderá cadastrar uma empresa (RF04) ou selecionar uma empresa (RF08) onde o mesmo está como funcionário. Na Figura 21 temos a tela para o usuário selecionar a empresa desejada.

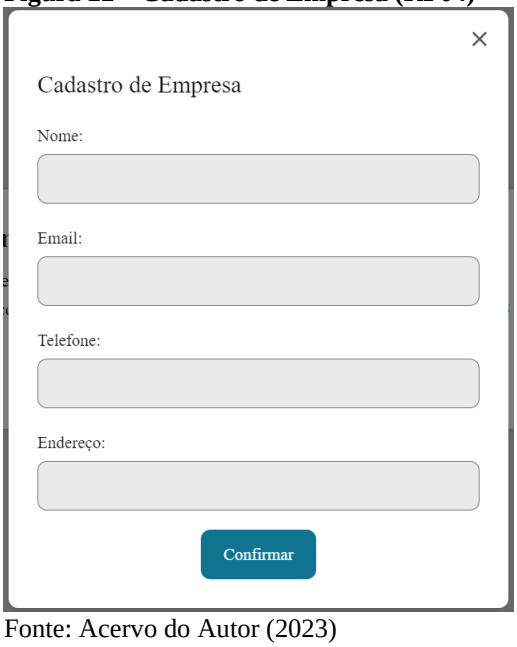
Figura 21 – Seleção de Empresas (RF08)



Fonte: Acervo do Autor (2023)

Caso não possuir uma empresa ainda o usuário poderá clicar no botão de ‘Incluir nova Empresa’ (Figura 21) que irá abrir o formulário mostrado na Figura 22. Neste formulário deverá ser inserido o nome, e-mail, telefone e endereço (RF04). Caso desejar cancelar a inclusão, a tela possui um botão para fechá-la, esta funcionalidade é aplicada para todas as telas de inclusão e alteração de registros.

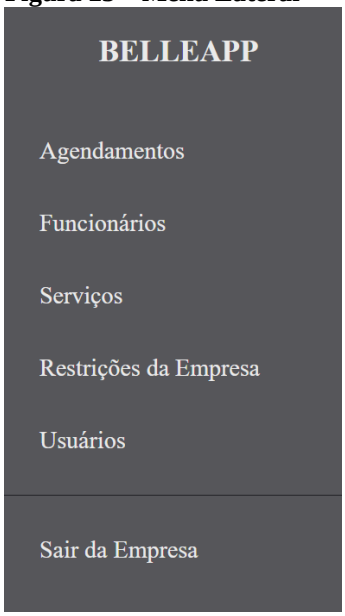
Figura 22 – Cadastro de Empresa (RF04)



Fonte: Acervo do Autor (2023)

Após selecionar a empresa o usuário será redirecionado para o *home* do sistema administrativo, onde em todo o seu contexto estará presente um menu lateral com as seguintes opções: Agendamentos, Funcionários, Serviços e Usuários (Figura 23). Estes levam para as consultas e possibilitam o cadastro de novos registros, alteração ou exclusão. No menu também possui a opção de sair da empresa, que ao clicar o usuário fica logado mas poderá escolher outra empresa para acessar o sistema.

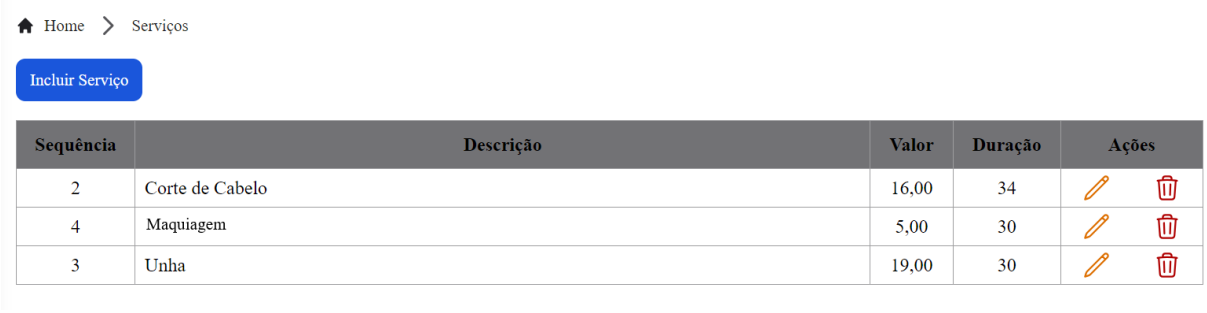
Figura 23 – Menu Lateral



Fonte: Acervo do Autor (2023)

Ao acessar o item ‘Serviços’, é apresentado todos os serviços ofertados pela empresa (Figura 24) (RF06). São apresentados os campos de sequência, descrição, valor (em R$) e a duração (em minutos), também possui botões para alterar e excluir o registro. Nesta tela também possui um botão para a inclusão de um novo serviço, onde é necessário inserir uma descrição, valor (em R$) e a duração (em minutos). Acima do botão de inclusão possui uma navegação estrutural, facilitando a localização do usuário no sistema e na otimização de páginas. Todas as telas de consulta seguem este padrão, contendo os campos principais, inclusão, ações e localização do usuário.

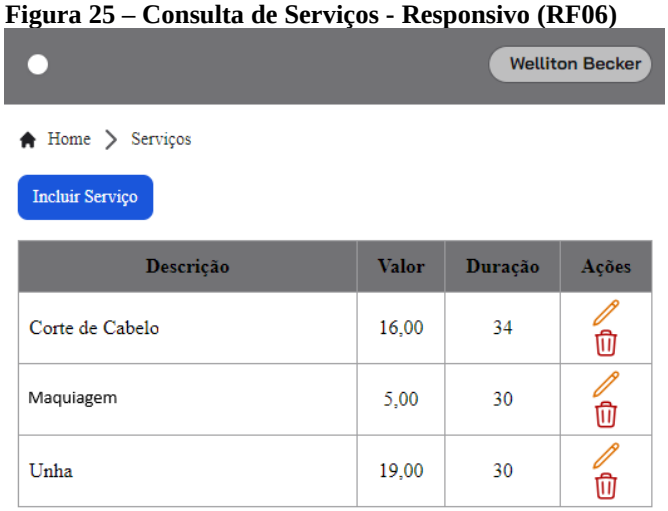
Figura 24 – Consulta de Serviços (RF06)



Fonte: Acervo do Autor (2023)

Conforme citado, o protótipo do sistema administrativo também é responsivo, dessa forma se adapta a diferentes tamanhos de telas conforme pode ser visto na Figura 25 um exemplo com a Consulta de Serviços.

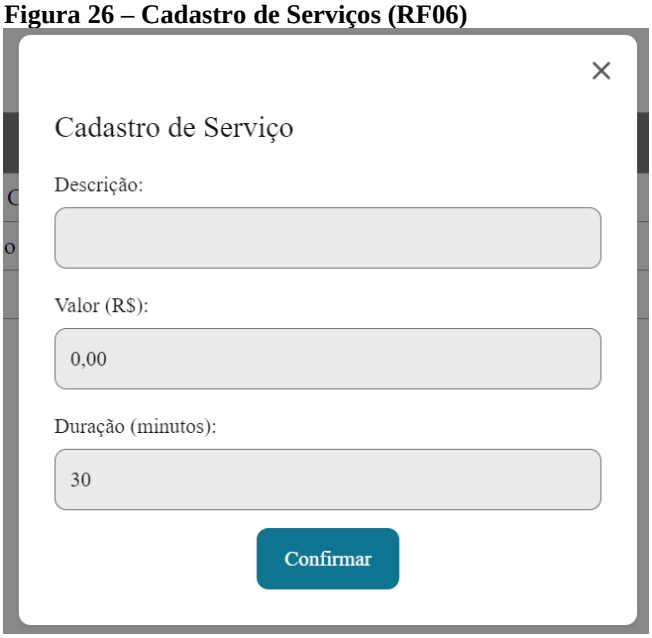
Figura 25 – Consulta de Serviços - Responsivo (RF06)



Fonte: Acervo do Autor (2023)

Ao clicar no botão de incluir serviço (RF06), um modal é aberto com um formulário para fazer a inclusão (Figura 26). O cadastro de serviço conta com os campos de descrição, valor em reais (R$) e a duração em minutos.

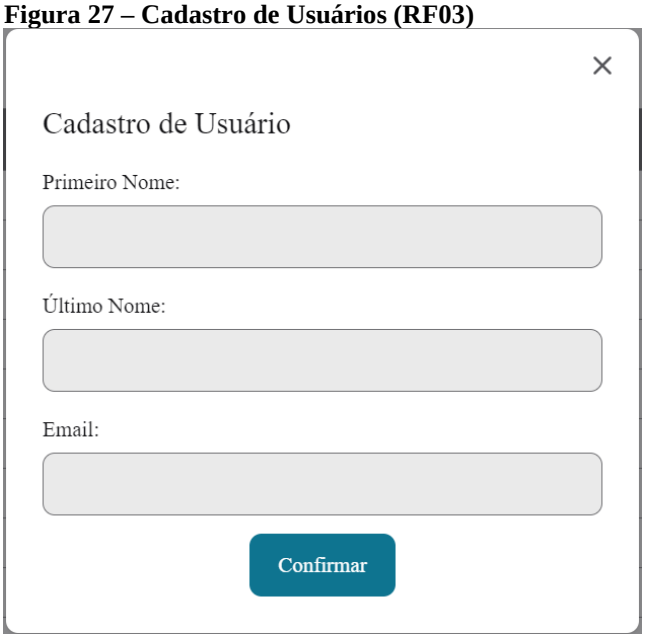
Figura 26 – Cadastro de Serviços (RF06)



Fonte: Acervo do Autor (2023)

Já no item de ‘Usuários’ é apresentado os usuários do sistema, onde apenas são apresentado o nome e o telefone. Nesta consulta apenas é disponibilizado o botão para a inclusão de um usuário (RF03), que abre um modal com um formulário (Figura 27), que possui os campos de primeiro e último nome, o e-mail e o botão para confirmar.

Figura 27 – Cadastro de Usuários (RF03)

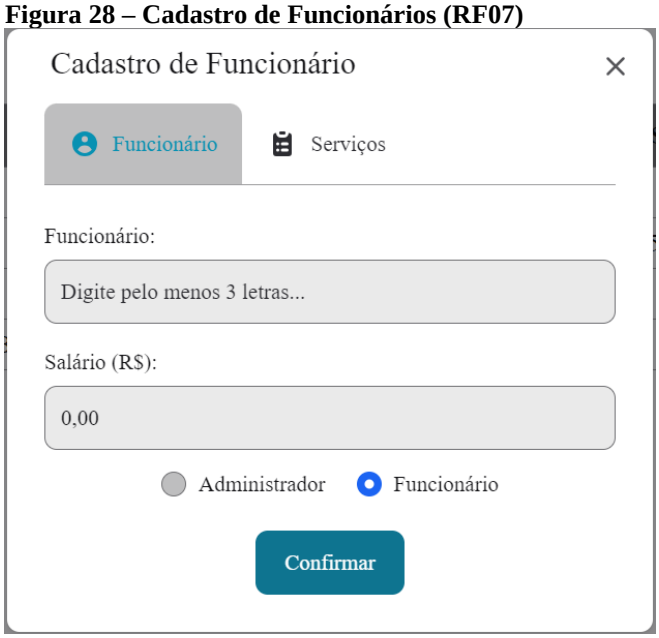


Fonte: Acervo do Autor (2023)

Ao acessar o item ‘Funcionários’, é apresentado todos os funcionários da empresa. Na inclusão de um funcionário (Figura 28), é apresentado duas abas: ‘Funcionário’ (RF07) e

‘Serviços’ (RF09). A primeira aba possui os campos de funcionário (ao digitar pelo menos 3 letras o sistema vai criar uma lista com usuários que possuem o nome digitado), o campo de salário (em R$) e deverá ser selecionado o tipo de funcionário (Administrador ou Funcionário).

Figura 28 – Cadastro de Funcionários (RF07)



Fonte: Acervo do Autor (2023)

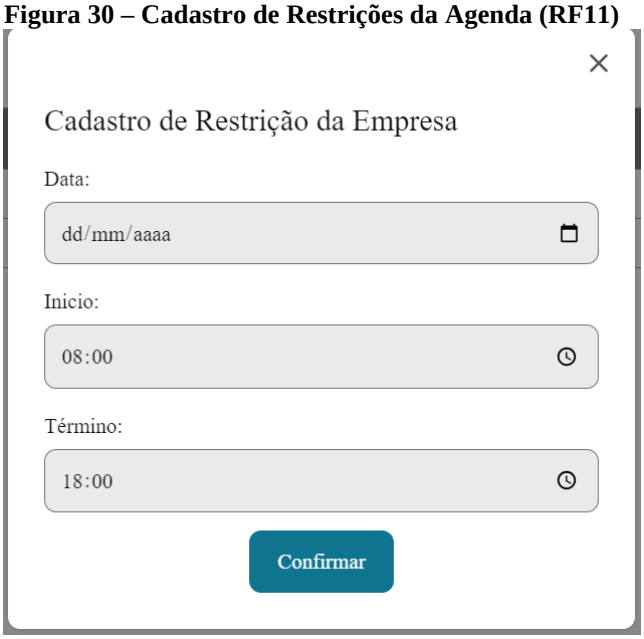
Já na segunda aba do cadastro de funcionário temos uma lista com os serviços prestados pela empresa (Figura 29). Os serviços selecionados aqui serão atribuídos ao funcionário (RF09), podendo assim serem selecionados pelo cliente no agendamento.

Figura 29 – Cadastro de Serviços do Funcionário (RF09)

Fonte: Acervo do Autor (2023)

No item ‘Restrições’ é possível consultar as restrições de agenda para a empresa ou fazer a inclusão de novas restrições (RF11). As restrições são vinculadas a datas e horários, e assim o usuário não poderá fazer agendamentos para datas e os horários que possuem restrições. Na Figura 30 é apresentado o formulário de cadastro de uma restrição, onde deve ser definido uma data e os horários de início e de término da restrição, desta forma é possível cadastrar uma restrição para um dia inteiro ou só por um período dele.

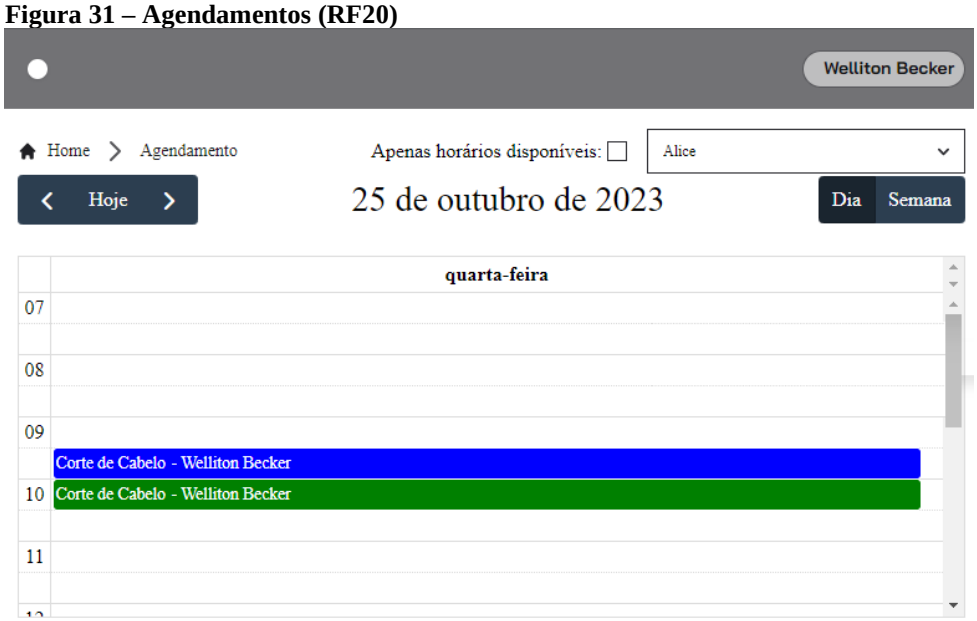
Figura 30 – Cadastro de Restrições da Agenda (RF11)



Fonte: Acervo do Autor (2023)

No item de ‘Agendamentos’ (RF20) temos a rotina principal do sistema administrativo. Nele é possível visualizar os agendamentos ao selecionar um funcionário no canto direito da consulta (Figura 31). Nesta consulta temos um componente em formato de agenda, onde facilmente podemos trocar o dia, colocar na data atual e formas de visualização em dia ou semana. No item do agendamento temos a descrição do serviço e o nome do cliente. Os itens na cor azul são os que estão concluídos, onde o atendimento já foi realizado e os itens nas cores verdes são os que estão agendados, esperando o atendimento.

Figura 31 – Agendamentos (RF20)



Fonte: Acervo do Autor (2023)

Ao clicar em um item do agendamento será aberto um formulário (Figura 32), onde é possível ver mais detalhes do agendamento, além de poder cancelar (RF12) ou finalizar o mesmo quando este estiver com a situação ‘agendado’.

Figura 32 – Cancelamento do Agendamento (RF12)

Fonte: Acervo do Autor (2023)

Na consulta de Agendamentos também é possível selecionar a flag ao lado da lista de usuários para listar os horários disponíveis do funcionário (RF10) (Figura 33). Ao clicar em um

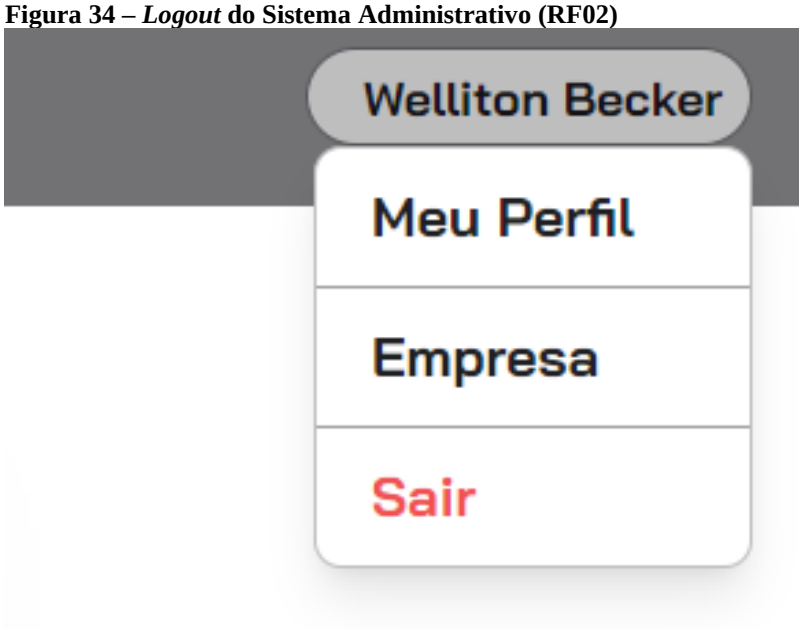
horário que está disponível, é possível selecionar um usuário e serviço e cadastrar um agendamento pelo próprio sistema administrativo.

Figura 33 – Horários Disponíveis (RF10)

Fonte: Acervo do Autor (2023)

No menu ao clicar no nome do usuário possui as opções para editar o usuário, editar a empresa e fazer o *logout*. Na Figura 34 mostra este menu, ao clicar no item de ‘Sair’ o sistema irá fazer o logout do usuário (RF02).

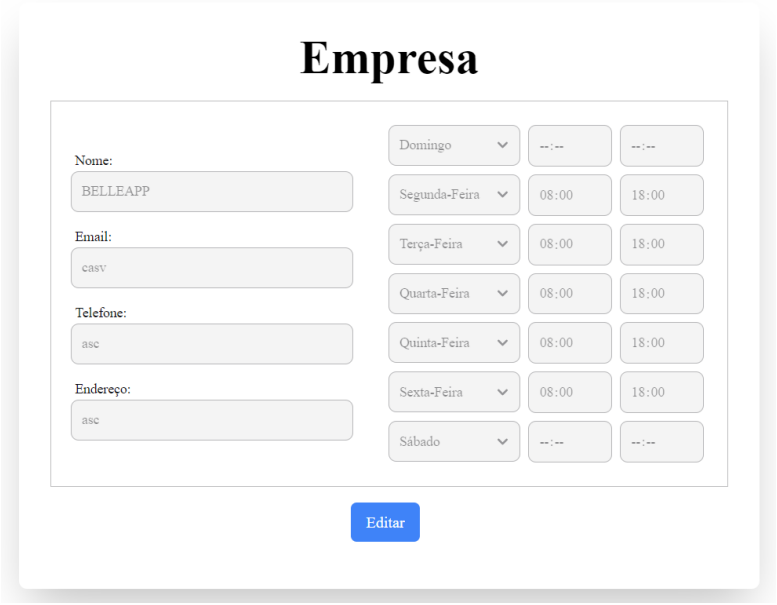
Figura 34 – *Logout* do Sistema Administrativo (RF02)



Fonte: Acervo do Autor (2023)

Ao clicar no item ‘Empresa’ é apresentado os detalhes da empresa (Figura 35). Ao clicar no botão de editar é possível alterar os dados da empresa (RF04) e os seus horários de funcionamento (RF05).

Figura 35 – Funcionamento da Empresa (RF05)

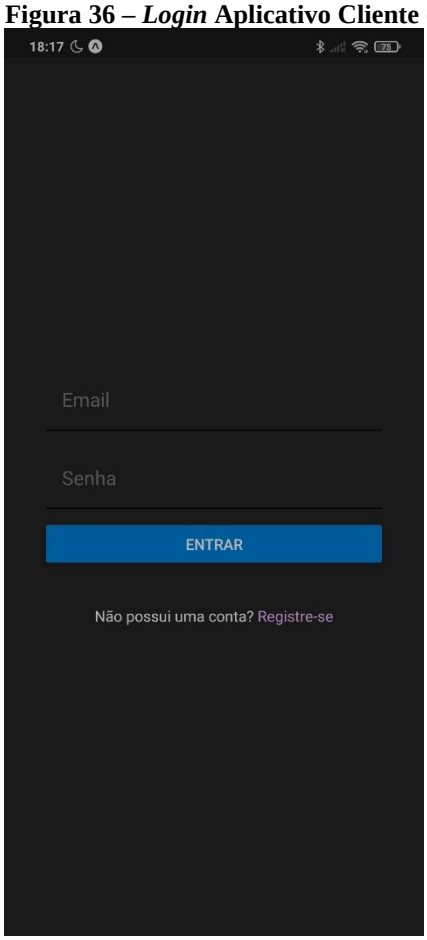


Fonte: Acervo do Autor (2023)

4.2.2.2 Aplicativo Cliente

Para acessar o aplicativo cliente o usuário deverá efetuar o *login* no aplicativo (RF13) informando um e-mail e uma senha (Figura 36).

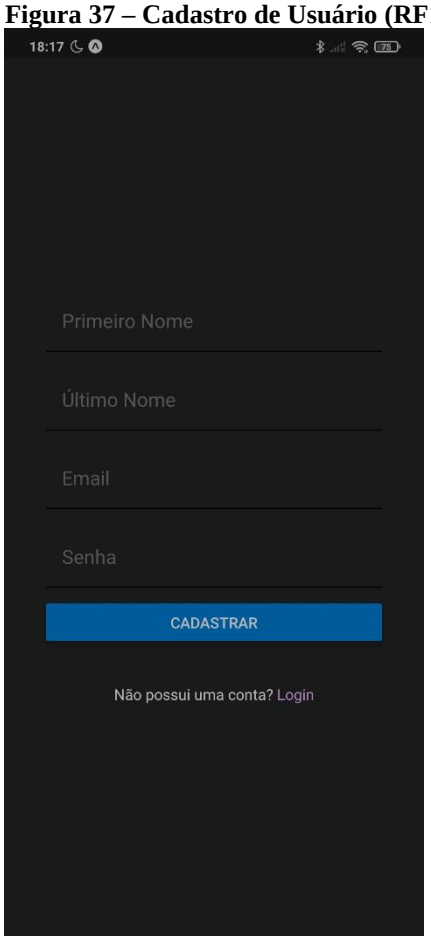
Figura 36 – *Login* Aplicativo Cliente (RF13)



Fonte: Acervo do Autor (2023)

Caso o usuário ainda não possua um *login* cadastrado, ele poderá se registrar ao clicar no texto escrito com ‘Registre-se’. Ao clicar ele será redirecionado para a tela mostrada na Figura 37, onde poderá cadastrar o seu usuário (RF15) informando um nome, sobrenome, e-mail e senha. Após confirmar ele já será redirecionado para a tela inicial do aplicativo, já logado.

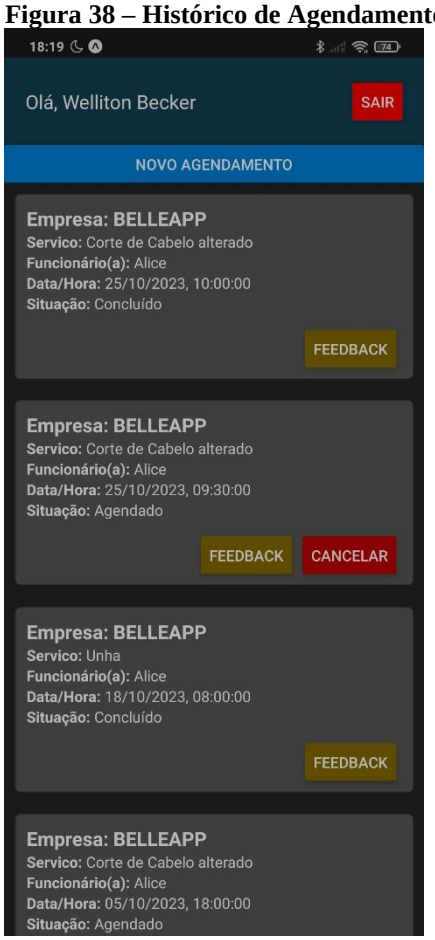
Figura 37 – Cadastro de Usuário (RF15)



Fonte: Acervo do Autor (2023)

Após o login, na tela inicial do aplicativo (Figura 38) temos o histórico dos agendamentos do usuário (RF20). No topo da tela temos o nome do usuário e um botão para sair (RF14), logo abaixo temos um botão para realizar o cadastro de um novo agendamento. Abaixo do botão de inclusão temos os agendamentos cadastradas do usuário, com a opção de realizar o *feedback* (RF21) ou cancelar o mesmo (RF19). Em cada agendamento é mostrado informações sobre a empresa, o serviço agendado, o profissional, a data e horário e a situação atual do agendamento (agendado, cancelado, concluído, lista de espera).

Figura 38 – Histórico de Agendamentos (RF20) e *Logout* (RF14)



Fonte: Acervo do Autor (2023)

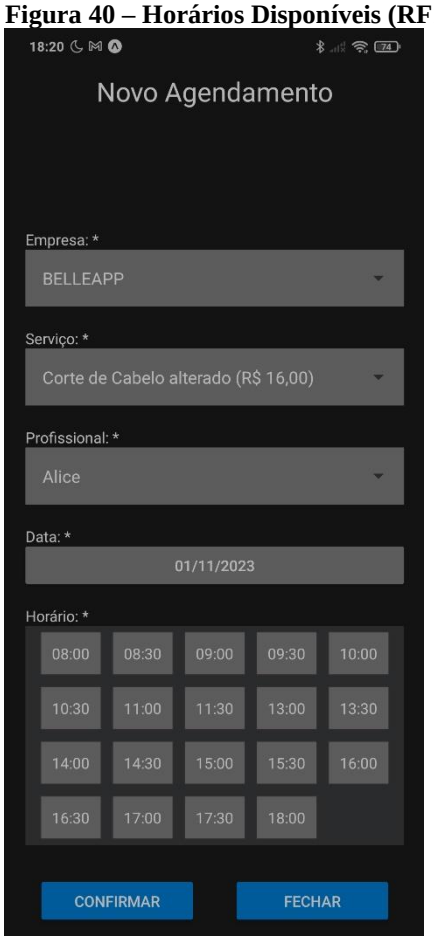
Para realizar um novo agendamento (RF16), ao clicar no botão na tela inicial o usuário é redirecionado para a tela apresentada na Figura 39, onde deverá selecionar uma empresa, após isso um serviço prestado pela empresa, selecionar um funcionário que preste o serviço selecionado e selecionar uma data para o atendimento.

Figura 39 – Cadastro de Agendamento (RF16)

Fonte: Acervo do Autor (2023)

Caso no dia selecionado possui algum horário disponível, estes serão apresentados conforme a Figura 40, na parte de horários (RF18). Após selecionar um horário o usuário poderá confirmar o formulário, incluído assim o seu agendamento.

Figura 40 – Horários Disponíveis (RF18)



Fonte: Acervo do Autor (2023)

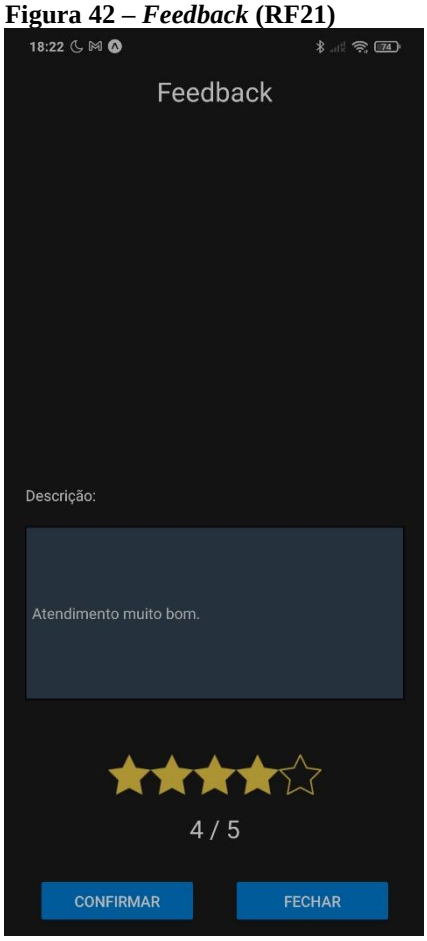
Caso na data selecionada não possua mais horários disponíveis, então o usuário será informado (Figura 41), que na data selecionado não há mais horários disponíveis, mas caso ele quiser confirmar o agendamento entrará em uma lista de espera (RF17).

Figura 41 – Lista de Espera (RF17)

Fonte: Acervo do Autor (2023)

Na tela inicial, o usuário poderá registrar um *feedback* (RF21) para cada agendamento (Figura 42), pode ser informado uma pequena descrição sobre o atendimento e adicionar uma nota de 0 a 5. Essa nota está representada em estrelas, assim a quantidade de estrelas ‘cheias’ representa a nota do atendimento agendado.

Figura 42 – *Feedback* (RF21)



Fonte: Acervo do Autor (2023)

No próximo capítulo são apresentadas as considerações finais e recomendações de trabalhos futuros.

5. CONSIDERAÇÕES FINAIS

Este trabalho apresentou o desenvolvimento de um protótipo de um sistema administrativo e de um aplicativo cliente, voltado para o gerenciamento de empresas que prestam serviços na área de beleza e estética, permitindo o cadastro de serviços, funcionários e o controle sobre os agendamentos além de permitir ao cliente final, utilizando o aplicativo, de realizar os agendamentos para uma determinada data e horário ou a possibilidade de entrar em uma lista de espera para a data desejada.

As ferramentas e linguagens de programação utilizadas foram adequadas para a conclusão do protótipo. Em todo o projeto foi utilizado o JavaScript com o auxílio do TypeScript. No *backend* foi utilizado o Node.js como *framework*, e para controlar o banco de dados foi utilizado a *framework* para ORM chamado Prisma.io. Já no *frontend* foi utilizado a *framework* Next.js que trabalha em cima do React.js e tem um controle mais dinâmico sobre as rotas. Para o aplicativo cliente (*mobile)* foi utilizado o próprio React Native como *framework* para montar as interfaces e foi utilizado o Expo para facilitar o desenvolvimento, gerenciamento de pacotes e para fazer os testes em tempo de execução em um celular conectado na mesma rede.

Em relação aos objetivos específicos do trabalho, todos foram alcançados. Em relação ao primeiro objetivo, que era de descrever as tecnologias utilizadas, foi construído a revisão da literatura, onde foi conceituado e explicado sobre: HTML, CSS, JS, TS, Desenvolvimento Mobile, alguns *frameworks* para trabalhar com o JS, Banco de Dados, PostegreSQL entre outros.

Com relação ao objetivo de identificar aplicações semelhantes, foram apresentadas duas aplicações por meio do estado da arte: Fresha e Trinks.

Após o estado da arte foi possível dar andamento a outro objetivo, o de levantar os requisitos e regras de negócios, auxiliado com os diagramas de caso de uso, atividade e entidade relacionamento. Este objetivo foi cumprido conforme apresentado no capítulo de análise.

E por fim sobre o último objetivo, de desenvolver o protótipo, este é apresentado no capítulo de implementação onde é mostrado o protótipo do sistema administrativo e o protótipo do aplicativo cliente.

O protótipo desenvolvido, poderá auxiliar ao empreendedor do setor de beleza e estética a manter a sua equipe, gerir melhor o tempo além de ter acesso de forma simples aos horários que estão disponíveis ou aos horários que já estão agendados pelos clientes, e com a utilização

do aplicativo cliente o funcionário não terá mais a necessidade marcar em uma agenda, sendo que o próprio cliente poderá verificar os horários disponíveis e realizar o agendamento com o profissional desejado na palma da mão.

Para os usuários clientes, utilizar o aplicativo irá agilizar o processo de agendamentos, não ficando o mesmo com a preocupação se o funcionário realmente agendou um horário para ele, ou que agendou para outro cliente no mesmo horário. Com o aplicativo também ele poderá verificar a melhor data e horário, conforme as suas necessidades e disponibilidade da empresa.

Embora todos os objetivos específicos tenham sido cumpridos e os requisitos funcionais propostos inicialmente foram desenvolvidos, novas ideias foram surgindo para aprimorar o protótipo. Sendo que estes são tratados como requisitos funcionais opcionais (Quadros 8 e 9) e propostos como sugestões para um futuro desenvolvimento.

5.1 RECOMENDAÇÕES DE TRABALHOS FUTUROS

Para continuidade deste trabalho, a primeira recomendação é a implementação de *log*'s no sistema administrativo e no aplicativo cliente. Em segundo lugar, seria a utilização de algoritmos inteligentes, para uma melhor gestão e previsão de agendamentos.

Como terceira sugestão seria o desenvolvimento do controle financeiro (entradas e saídas), além da possibilidade de utilização de pagamentos *online* e relatórios gerenciais para o sistema.

Fica como recomendação também a integração com aplicações externas como o WhatsApp, onde poderia ser possível enviar mensagens para a empresa e por meio de um *chatbot* fazer consultas de horários disponíveis e até mesmo fazer agendamentos.

Para o sistema administrativo, seria recomenda-se a criação de *dashboards* com informações úteis mostrando o faturamento, horários, disponibilidades, custos etc. Recomendado também rotinas de notificações ao usuário cliente para alertá-lo sobre promoções, horários disponíveis e sobre a liberação de vagas para datas em que o mesmo está na lista de espera, verificando se ele realmente deseja agendar o horário.

Recomenda-se também, a criação de um aplicativo para os funcionários com as principais rotinas do sistema administrativo.

Por fim, também se sugere a homologação do sistema administrativo e aplicativo cliente com testes pilotos em uma ou duas empresas e alguns usuários, validando a regra e a usabilidade dos sistemas.

REFERÊNCIAS

AMOASEI, Juliana. **O que são regras de negócio?.** 2023. Disponível em: https://www.alura.com.br/artigos/o-que-sao-regras-de-negocio. Acesso em: 26 set. 2023.

AZURE. **O que é PostgreSQL?.** 2023. Disponível em: https://azure.microsoft.com/pt-br/resources/cloud-computing-dictionary/what-is-postgresql/. Acesso em: 27 set. 2023.

AXIOS. **Introdução.** 2023. Disponível em: https://axios-http.com/ptbr/docs/intro. Acessado em: 28 agosto 2023.

BUZZI, Felipe, **Prisma: uma das melhores coisas que já aconteceu no ecossistema?.** 2022. Disponível em: https://blog.rocketseat.com.br/prisma-uma-das-melhores-coisa-que-ja-aconteceu-no-ecossistema/. Acessado em: 09 maio 2023.

CARNES, Beau. **Aprenda Supabase, uma alternativa de código aberto ao Firebase.** 2023**.** Disponível em https://www.freecodecamp.org/news/learn-supabase-open-source-firebase-alternative/. Acessado em 28 agosto 2023.

CALOMENO JUNIOR, **Adil. React Native: o que é, quais as funcionalidades e as vantagens desse framework.** 2020. Disponível em: https://ateliware.com/blog/react-native. Acessado em: 30 Junho 2023.

CAYRES, Paulo Henrique. **Modelagem de banco de dados**. Rio de Janeiro 2015.

CHINNATHAMBI, D. **React em Ação**. Rio de Janeiro: Novatec, 2017.

DOS SANTOS, Robson. **Tailwind CSS: O que é? Como Usar?.** 2023. Disponível em: https://www.brasilcode.com.br/tailwind-css-o-que-e-como-usar/. Acessado em: 14 setembro 2023.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de Banco de Dados**. São Paulo: Pearson, 2018.

ESTADÃO, **Em dois anos país abriu 343 mil salões de beleza.** 2022. Disponível em: https://www.estadao.com.br/pme/em-dois-anos-pais-abriu-343-mil-saloes-de-beleza Acessado em: 08 fevereiro 2023.

FASTIFY. **Fastify**. 2023. Disponível em: https://fastify.dev/. Acesso em: 08 nov. 2023.

FERNANDES, Diego. **Expo: o que é, para que serve e quando utilizar?.** 2018. Disponível em: https://blog.rocketseat.com.br/expo-react-native/. Acessado em: 29 agosto 2023.

FILGUEIRA, Israel Eriston. **Fastify — Não é mais um framework para node.js**. 2018. Disponível em: https://medium.com/@IsraelEriston/fastify-não-é-mais-um-framework-para-node-js-783c3990cd55. Acesso em: 08 nov. 2023.

FLANAGAN, D. (2002). **JavaScript: O guia definitivo**.

FRESHA, **Para Empresas.** 2023. Disponível em: https://www.fresha.com/pt/for-business. Acessado em: 20 agosto 2023.

GAEA. **Desenvolvimento mobile: entenda os seus princípios!.** 2021. Disponível em: https://gaea.com.br/desenvolvimento-mobile. Acesso em: 19 maio 2023.

GONÇALVES, Ariane. **O que é CSS? Guia Básico para Iniciantes.** 2022**.** Disponível em: https://www.hostinger.com.br/tutoriais/o-que-e-css-guia-basico-de-css#:~:text=CSS%20%C3%A9%20a%20sigla%20para,de%20marca%C3%A7%C3%A3o%20HTML%20ou%20XHTML. Acessado em: 19 maio 2023.

HEUSER, Carlos Alberto. **Projeto de Banco de Dados**. Editora Sagra Luzzatto, 2001.

JWT. **Introdução aos tokens JSON Web.** 2023. Disponível em: https://jwt.io/introduction. Acessado em 28 agosto 2023.

LIMA, Cleyson. **O que é JWT**. 2021. Disponível em: https://www.treinaweb.com.br/blog/o-que-e-jwt. Acessado em 28 agosto 2023.

MACHADO, Felipe Nery Rodrigues. **Banco de dados: projeto e implementação**. São Paulo Erica 2020.

MARCHIORI, Lucas. **Next JS: o que é, para que serve e por que usar?**. 2023. Disponível em: https://blog.betrybe.com/tecnologia/next-js. Acessado em: 29 agosto 2023.

MOZILLA, **Elemento HTML**. 2023a. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn/Getting_started_with_the_web/HTML_basics. Acessado em: 19 maio 2023.

MOZILLA, **HTML**. 2023b, Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/HTML. Acessado em: 09 maio 2023.

NEVES, Vinicios. **Requisições HTTP utilizando o AXIOS.** 2022. Disponível em: https://www.alura.com.br/artigos/requisicoes-http-utilizando-axios. Acessado em: 28 agosto 2023.

PEREIRA, Caio Ribeiro, (2014), **Aplicações web real-time com Node.js.**

PERERA, Madushika. **Explorando Supabase, a alternativa de código aberto do Firebase**. 2021. Disponível em: https://blog.logrocket.com/exploring-supabase-the-open-source-firebase-alternative/. Acessado em: 28 agosto 2023.

PREIS, Djeferson; BUTZKE, Marco Aurélio; BASTOS, Fernando Andrade. **PROTÓTIPO DE SISTEMA DE GERENCIAMENTO DE BARBEARIAS (NAVALHAPP). Caminhos**, Rio do Sul, p. 142-163, nov. 2023.

PRISMA.IO. **Quickstart**. 2023. Disponível em: https://www.prisma.io/docs/getting-started/quickstart. Acesso em: 10 ago. 2023.

RAMAKRISHNAN, R., & Gehrke, J. (2003). **Database management systems**. McGraw-Hill Education.

REACT **Documentação.** 2023. Disponível em: https://legacy.reactjs.org/docs/components-and-props.html. Acessado em: 19 maio 2023.

ROCHA, Helder Lima Santos da (1999) **Desenvolvendo Web Sites Interativos com JavaScript.**

SIDELAB. **Conhecendo o Expo e iniciando um projeto.** 2022. Disponível em: https://www.sidelab.com.br/post/conhecendo-o-expo-e-iniciando-um-projeto#:~:text=Expo%20Go%20%C3%A9%20um%20aplicativo,no%20browser%20ou%20no%20terminal. Acessado em: 29 agosto 2023.

SIMAS, Victor Luiz. **Desenvolvimento para dispositivos móveis**. Porto Alegre, 2019.

SILVA, Gizele. **O que é Next.js?**. 2023. Disponível em: https://coodesh.com/blog/dicionario/o-que-e-next-js/. Acesso em: 29 ago. 2023.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

SOUZA, Ivan de. **O que é o PostgreSQL?.** 2020. Disponível em: https://rockcontent.com/br/blog/postgresql/. Acesso em: 27 set. 2023.

TRINKS, **Gestão Online Completa.** 2022. Disponível em: https://www.trinks.com/produto/gestao-online-beleza. Acessado em: 20 agosto 2023.

TYPESCRIPT **Documentação**. 2023. Disponível em: https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html. Acessado em: 19 maio 2023.

W3SCHOOLS, **CSS**. 2023a. Disponível em: https://www.w3schools.com/css/css_intro.asp. Acessado em: 09 maio 2023.

W3SCHOOLS, **HTML.** 2023b. Disponível em: https://www.w3schools.com/html/html_intro.asp. Acessado em: 19 maio 2023.

XPEDUCACAO. **Desenvolvimento mobile: principais desafios e motivos para aprender**. 2022. Disponível em: https://blog.xpeducacao.com.br/desenvolvimento-mobile/#O_que_e_o_desenvolvimento_mobile. Acesso em: 09 maio 2023.