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

DJEFERSON PREIS

PROTÓTIPO DE SISTEMA DE GERENCIAMENTO DE BARBEARIAS (NAVALHAPP)

RIO DO SUL
2022

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

DJEFERSON PREIS

PROTÓTIPO DE SISTEMA DE GERENCIAMENTO DE BARBEARIAS (NAVALHAPP)

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

Prof. Orientador: Dr. Marco Aurélio Butzke

RIO DO SUL
2022

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

DJEFERSON PREIS

PROTÓTIPO DE SISTEMA DE GERENCIAMENTO DE BARBEARIAS (NAVALHAPP)

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

_____

Professor Orientador: Dr. Marco Aurélio Butzke

Banca Examinadora:

_____

Professor M.e Fernando Andrade Bastor

_____

Professor M.e Jeancarlo Visentainer

Rio do Sul, 28 de Novembro de 2022.

Dedico este trabalho aos meus pais, Moacir e Dirlene, e ao meu irmão, Djalmir, que sempre me apoiaram para que fazer chegar até aqui.

AGRADECIMENTOS

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

Agradeço meus pais Moacir e Dirlene, por me apoiaram em meus estudos e escolhas e acreditarem em meu potencial, e me auxiliando com tudo que podem para que eu consiga obter minhas conquistas e alcançar meus objetivos.

Em seguida gostaria de agradecer a todos os professores do curso de Sistemas de Informação pelo conhecimento compartilhado, as oportunidades apresentadas e conversas fortemente construtivas para evolução pessoal e profissional. Em especial gostaria de agradecer ao Dr. Marco Aurélio Butzke, pelo suporte na construção deste trabalho desde sua mera idealização.

Por fim gostaria de agradecer todos os meus colegas, por todas as memórias e experiências obtidas ao longo destes 4 anos dentro e fora da vida acadêmica, proporcionando amizades construtivas e passando por diversos tipos de momentos juntos.

RESUMO

O setor de barbearias possui milênios de existência e, por mais que suas técnicas evoluam lenta, mas constantemente não é possível usufruir de processos totalmente tecnológicos até o presente momento, contudo a parte empresa destas barbearias receber automações com a finalidade de proporcionar uma melhor qualidade no atendimento e nos serviços, o que consequentemente auxilia o setor tornando-o mais ativo e competitivo no mercado. Pensando nestes processos empresarias e de gestão do negócio e também em aplicações recentemente em alta como aplicativos de delivery, este trabalho teve como objetivo o desenvolvimento de um protótipo de aplicação web que possibilitará a gestão de barbearias com a facilitação na apresentação de serviços e de controle das agendas e pedidos. A aplicação foi desenvolvida utilizando tecnologias modernas baseadas principalmente na linguagem de programação Javascript, e sendo dividida em duas partes: a API, feita em NodeJS, utilizando o framework NestJS, e o *frontend* utilizando ReactJS com padronização das interfaces por meio da utilização de bibliotecas de responsividade como o Material UI. A aplicação frontend foi separada em outras três partes, sendo elas a sessão administrativa destinada ao controle geral do aplicativo, a sessão empresarial destinada às empresas, onde é possível realizar o cadastro dos serviços, agendas e controle dos pedidos, e a sessão do cliente final, destinada ao usuário, onde é possível visualizar as barbearias disponíveis no aplicativo, fazer seu cadastro e agendar serviços nas barbearias de forma rápida e simples, sem a necessidade de contato direto ou presencial com a empresa. Como resultado final, foi possível analisar que a plataforma desenvolvida pode contribuir para o aumento do canal de vendas das empresas e facilitar a gestão, organização e controle financeiro das barbearias.

Palavras-Chave: barbearias, delivery, tecnologias modernas.

ABSTRACT

The barbershop sector has been in existence for millennia and, even though its techniques evolve slowly but constantly, not possible to take advantage of fully technological processes up to the present moment, however the business part of these barbershops receives automations in order to provide a better quality in the assistance and services, which consequently helps the sector by making it more active and competitive in the market. Thinking about these business and business management processes and also about applications recently on the rise such as deliveries applications, this work aimed to develop a prototype web application that will enable the management of barbershops by facilitating the presentation of services and control of schedules and orders. The application was developed using modern technologies based mainly on the Javascript programming language, and being divided into two parts: the API, made with NodeJS, using the NestJS framework, and the frontend using ReactJS with standardization of the interfaces through the use of libraries of responsiveness like Material UI. The frontend application was separated into three other parts, namely the administrative section for general control of the application, the business session for companies, where it is possible to register services, schedules and order control, and the customer session , aimed at the user, where it is possible to view the available barbershops, register and schedule services in the barbershops quickly and simply, without need direct or face-to-face contact with the company. As a final result, it was possible to analyze that the developed platform can contribute to increase the companies sales channel and facilitate the management, organization and financial control of the barbershops.

Keywords: barbershop, delivery, modern technologies.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
ACID	Atomicidade, Consistência, Isolamento e Durabilidade
COVID-19	Doença infecciosa causada pelo vírus SARS-CoV-2
CRUD	*Create, Read, Update, Delete*
CSS	*Cascading Style Sheets*
HTTP	*Hypertext Transfer Protocol*
IDE	*Integrated Development Environment*
JS	JavaScript
JSON	JavaScript *Object Notation*
JWT	JSON *Web Token*
MVP	*Minimum Viable Product*
PWA	*Progressive Web App*
REST	*Representational State Transfer*
RF	Requisito Funcional
SGBD	Sistema Gerenciador de Banco de Dados
SQL	*Structured Query Language*
UI	*User Interface*
URL	*Uniform Resource Locator*
WEB	*World Wide Web*

SUMÁRIO

1. INTRODUÇÃO

O setor de barbearias possui milênios de história e continua em constante evolução de suas técnicas e das tecnologias envolvidas com este mercado desde sua criação que, conforme citado por CARDOSO (2020), teria sido considerado como profissão por volta do Século II antes de Cristo no Egito.

Este mercado possui uma grande restrição em relação à entrada nos meios digitais que é a obrigatoriedade de que ambos os envolvidos estejam fisicamente juntos, e devido à pandemia do COVID-19, iniciada no final de 2019, este mercado sofreu com diversas dificuldades, tanto pelas perspectivas das empresas, que não poderiam parar, quanto pela perspectiva dos clientes que não poderiam ficar tantos meses sem cuidar minimamente de seus visuais mesmo estando *lockdown.*

Segundo AGÊNCIA BRASIL (2020), “Nove de cada dez micro e pequenas empresas que prestam serviço para beleza, como salões, barbearias, ateliês e estúdios de maquiagem, afirmam ter perdido faturamento por causa das medidas de isolamento social. A perda média do faturamento foi de 57%", e com isto, houve a necessidade destas empresas se reinventarem, desde ações de marketing com a utilização de vouchers à utilização de aplicações digitais para o agendamento e comunicação com os clientes.

Valvente (2021) afirma que a prática na realização de pedidos de refeições pela internet pelo menos uma vez por semana aumentou de 40,5% para 66,1% durante a pandemia do COVID-19 e, mesmo que o indicador apresente que este hábito deva baixar no período de pós-pandemia, a estimativa seria de pelo menos 57,8% das pessoas manterem a prática. Com isso podemos concluir que há uma predisposição para o uso de aplicativos para realizar processos por meios eletrônicos quando disponíveis.

Tendo em vista os dados e situações apresentadas, podemos concluir que o mercado de barbearias, principalmente de médio e pequeno porte, precisa de tecnologias para lhes auxiliar e para poderem realmente competir neste mercado que vem se reinventando nas áreas de marketing e comunicação. Concluímos também que a população em geral possui uma certa predisposição à utilização de procedimentos por meio da internet, possibilitando a elas uma melhor qualidade de vida e mais tempo para realizarem outras atividades.

A proposta deste trabalho é o desenvolvimento de uma plataforma web que possibilitará às barbearias disponibilizarem de forma simples e prática seus serviços na internet, mas não apenas isto, também facilitar esta linha de comunicação entre cliente e empresa de forma a automatizar todo o processo de agendamentos dos clientes conforme a quantidade de barbeiros

das empresas. Concomitantemente permitir uma maior igualdade entre as empresas de pequeno porte com as de médio e grande porte com auxílio de relatórios e informações de seu rendimento e possibilitando a todos os tamanhos de barbearias a visibilidade de si mesmas e de seus serviços ao público em geral.

1.1 PROBLEMA DE PESQUISA

Como gerenciar as agendas, vendas, prestações de serviços e auxiliar na fidelização de clientes de barbearias de pequeno, médio e grande porte?

1.2 OBJETIVOS

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

1.2.1 Geral

- Desenvolver um protótipo para gerenciamento de reservas e serviços em barbearias.

1.2.2 Específicos

- Levantar requisitos de administração dos serviços prestados pelas barbearias aos clientes;
- Definir as ferramentas para implementar as rotinas administrativas internas e a aplicação para a reserva e acompanhamento dos clientes de barbearias;
- Interpretar os processos existentes na barbearia e melhorar o formato de oferta de serviços aos clientes;
- Implementar as rotinas de administração do sistema e o aplicativo de reserva dos clientes;
- Validar as aplicações implantadas e desenvolvidas por meio de entrevista ao usuário.

1.3 JUSTIFICATIVA

Com o surgimento da pandemia do SARS-CoV-2, conhecido popularmente como Covid-19, gerou-se a necessidade de controle de filas de espera e o devido distanciamento social, este processo ocasionou a diminuição dos processos em meio físico, transformando a maior quantidade de processos possíveis dos negócios para o formato online ou a distância.

Tendo este grande aumento de processos com o uso de tecnologias devido ao que passamos na pandemia do Covid-19, pensamos em como poderíamos transformar um processo onde há grande necessidade de interações físicas como os de uma barbearia de forma a diminuir tais interações e transformá-las, em grande parte, em processos feitos à distância? Como auxiliar as pessoas a gerirem se tempo e concomitantemente diminuir os riscos sem que as próprias barbearias tenham que perder ou se desgastar para realizar estes processos?

Com o propósito de resolver os problemas apresentados durante a pandemia Covid-19, facilitar às pessoas gerenciarem seu tempo enquanto cuidam de sua segurança, e auxiliar as barbearias a gerirem seus negócios com benefícios em relação a seus clientes, propõem-se o desenvolvimento de um aplicativo que integrante o cliente à barbearia e gerando os benefícios e processos necessários para facilitar o controle do estabelecimento e das agendas de disponibilidade para prestação dos serviços aos clientes.

Para construção do sistema optou-se por um sistema web devido à grande facilidade para acesso em qualquer ambiente e dispositivo com internet e também, ao utilizar ferramentas web evitamos grandes custos em relação à necessidades de publicação em lojas, poder de processamento dos servidores, manutenibilidade das ferramentas e constante atualização em visão das tecnologias que surgem a todo momento, já que a ferramenta web seria centralizada em duas partições, o *front-end* e o *back-end*, em vez de utilizarmos versões específicas para cada ambiente, sendo ele Android, iOS, Desktop Windows, entre outros.

2. REFERENCIAL TEÓRICO

Neste capítulo serão abordados os assuntos referentes às tecnologias de desenvolvimento utilizadas durante todo o processo de desenvolvimento do projeto.

2.1. WEB SERVICE

Rodrigues e Neumann (2020, p.196) consideram que, “O web service é uma coleção de protocolos e padrões de código aberto utilizada para a troca de dados entre sistemas ou aplicativos na internet.”. E ainda afirmam que “É possível considerar um web service como uma API, contudo, nem toda API é considerada um web service.”.

2.1.1. API

Freitas, Birnfield e Saraiva (2021, p.43) definem API como, “[...] construções de aplicações que permitem que os desenvolvedores criem funcionalidades complexas mais facilmente. Tais construções abstraem o código mais complexo, proporcionando o uso de sintaxes de forma mais simples.”. Ainda sobre a definição de API, Rodrigues e Neumann (2020) complementam definindo também como uma coleção de rotinas e padrões de um sistema, fornecendo acesso externo às funcionalidades sem a necessidade de acesso direto código ou à implementação, mas de forma abstrata e totalmente independente da programação.

MDN (2021) complementa dizendo que uma API é geralmente um conjunto de métodos padronizados, propriedades, eventos, e URLs que podem ser utilizadas no desenvolvimento das aplicações para realizar a interação entre a aplicação e o servidor ou entre a aplicação e serviços de terceiros.

2.2. PROTOCOLO HTTP

Simas et al. (2019) afirma que HTTP (*Hypertext Transfer Protocol*) é a base da tecnologia Web e por meio dele é que as aplicações realizam suas respectivas comunicações com os servidores que, por si, responderão conforme a solicitação da aplicação e a disponibilidade dos dados.

2.2.1. Arquitetura REST

Rodrigues et al. (2020) afirma que o REST (*Representational State Transfer*) surgiu a partir de uma dissertação que tinha como objetivo apresentar uma proposta de padronização da integração entre aplicações baseadas no protocolo HTTP. Red Hat (2020) complementa dizendo que o REST não é um protocolo ou um padrão, e sim um conjunto de restrições de arquitetura que possibilita aos desenvolvedores utilizar a arquitetura de diversas maneiras, possibilitando simples comunicações para adquirir dados até uma aplicação que não armazena dados, mas que adquire todos os dados diretamente do servidor por meio das comunicações da arquitetura REST.

2.2.2 Métodos HTTP

Conforme apresentado por MDN (2021), o Protocolo HTTP define um conjunto de métodos de requisição que indicam o tipo de ação que se deseja processar. Cada método possui características próprias, mas podem compartilhar algumas de suas características com outros. Como podem ser observados os métodos e suas funções no Quadro 1.

Quadro 1 – Métodos HTTP

Método	Função
GET	O método GET solicita a representação de um recurso específico. Requisições utilizando o método GET devem retornar apenas dados.
HEAD	O método HEAD solicita uma resposta de forma idêntica ao método GET, porém sem conter o corpo da resposta.
POST	O método POST é utilizado para submeter uma entidade a um recurso específico, frequentemente causando uma mudança no estado do recurso ou efeitos colaterais no servidor.
PUT	O método PUT substitui todas as atuais representações do recurso de destino pela carga de dados da requisição.
DELETE	O método DELETE remove um recurso específico.
CONNECT	O método CONNECT estabelece um túnel para o servidor identificado pelo recurso de destino.
OPTIONS	O método OPTIONS é usado para descrever as opções de comunicação com o recurso de destino.
TRACE	O método TRACE executa um teste de chamada *loop-back* junto com o caminho para o recurso de destino.
PATCH	O método PATCH é utilizado para aplicar modificações parciais em um recurso.

Fonte: Elaborado a partir de MDN (2021).

2.3. JAVASCRIPT

JavaScript é uma linguagem de programação amplamente utilizada. Segundo Flanagan (2014, p.18) “JavaScript faz parte da tríade de tecnologias que todos os desenvolvedores Web devem conhecer: HTML, para especificar o conteúdo de páginas Web; CSS, para especificar a apresentação dessas páginas; e JavaScript, para especificar o comportamento delas.”. Oliveira e Zanetti (2020) complementam dizendo que é uma linguagem de programação orientada a objetos, interpretada e executada por navegadores web e seu objetivo principal é oferecer maior interatividade às páginas.

> Na verdade, o nome “JavaScript” é um pouco enganoso. A não ser pela semelhança sintática superficial, JavaScript é completamente diferente da linguagem de programação Java. E JavaScript já deixou para trás suas raízes como linguagem de script há muito tempo, tornando-se uma linguagem de uso geral robusta e eficiente. A versão mais recente da linguagem [...] define novos recursos para desenvolvimento de software em grande escala. (FLANAGAN, 2014, p.18).

MDN (2021) adverte sobre JavaScript de que ele não deve ser confundido com a linguagem de programação Java, mesmo tendo os nomes de suas marcas comerciais bastante semelhantes são duas linguagens de programação significativamente diferentes em sintaxes, semânticas e também bem em seus casos de uso.

Sobre a linguagem JavaScript, Oliveira e Zanetti (2020, p.46) certificam que “Uma característica importante da linguagem JavaScript é que ela não apresenta tipos de dados, isto é, qualquer variável definida é do tipo variante.”, e Oliveira e Zanetti (2021, p.8) complementam dizendo que “O tipo de dados de determinada variável pode ser modificado ao longo da execução da aplicação, conforme o seu conteúdo vai sendo alterado.”.

Alves (2015, p.157) ainda conclui dizendo que, “Podem ser desenvolvidas rotinas em JavaScript que rodem tanto na máquina do usuário quanto no servidor web. Com o JavaScript torna-se possível ainda criar páginas dinâmicas, que processam as informações do usuário e devolvem como resultado outras informações.”.

2.3.1. TypeScript

TypeScript (2022) conceitua TypeScript como sendo uma linguagem de programação fortemente tipada baseada no JavaScript, oferecendo sintaxes adicionais ao JavaScript para ofertar um melhor suporte em conjunto ao editor do código para detectar previamente os erros.

Também informam que, ao ser executado, o TypeScript é convertido para JavaScript para poder ser executado em qualquer lugar, seja em um navegador, no Node.js ou em outros aplicativos.

Simas et al. (2019, p.59) afirma que “A linguagem TypeScript é um conjunto do JavaScript que permite uma melhor usabilidade ao programador que já está acostumado a trabalhar com orientação a objetos e faz sua conversão para o JavaScript de forma automática e segura.”. Ivanov e Bespoyasov (2020) ainda concluem afirmando que o TypeScript possibilita especificar os tipos das informações, das variáveis no código, o que permite o desenvolvimento de aplicação com maior confiança.

2.4. HTML

Sobre HTML, segundo Saraiva e Barreto (2018, p.9) afirma que “HTML, do inglês hyper text markup language, ou linguagem de marcação de hipertexto, é uma linguagem utilizada para criar documentos para a web.”.

> A HTML não é uma linguagem de programação propriamente dita. Ela é mais bem classificada como uma linguagem de formatação de textos ou definição da estrutura de um documento. Ela não gera um programa executável autônomo, como ocorre com as linguagens de programação tradicionais C++, Java e Pascal. Em vez disso, tem-se somente um arquivo em formato de texto, geralmente com a extensão .HTM ou .HTML, que é lido e interpretado por um software denominado navegador. Ele é o responsável por apresentar no monitor de vídeo aquilo que foi codificado no documento HTML. (ALVES, 2014, p.26).

Alves (2015, p.15) conclui afirmando que, “Por meio dessa linguagem, o desenvolvedor pode especificar atributos para um texto (como fonte, tamanho, cor etc.) ou criar marcações que definam um vínculo com outro documento, que deve ser apresentado após o usuário ter clicado sobre a marcação.”.

MDN (2021) gera um resumo conceitual de um Documento HTML como um simples texto estruturado com elementos, sendo que os elementos são acompanhados de aberturas e fechamentos de tags, onde cada tag começa e termina com colchetes angulares (<>). E observa que também existem algumas tags que não possuem conteúdo textual, mas que possuem funções específicas como para a apresentação de uma imagem, vídeo ou simplesmente para pular uma linha no texto.

2.5. BANCO DE DADOS

Barboza e Freitas (2018) definem Banco de Dados como um conjunto de valores e arquivos que se relacionam entre si, armazenando estes dados e possibilitando aplicações os consultarem, alterarem, acrescer mais informações ou removê-las.

Conforme apresentado por Pichetti, Vida e Cortes (2021), com a ascensão das tecnologias tornou-se de necessário e de grande interesse o armazenamento de todas as informações, com isso surgiu-se, ainda na década de 1960, os bancos de dados. Desde sua criação os Bancos de Dados possuem o mesmo propósito de armazenar a informação de maneira estruturada e segura.

2.5.1. SGBD (Sistema de Gerenciamento de Banco de Dados)

Segundo Pichetti, Vida e Cortes (2021, p.12), “Os SGBDs são constituídos por um conjunto de software com diferentes funcionalidades, que se complementam para oferecer serviços de criação e manipulação de bancos de dados”.

Barboza e Freitas (2018) reforçam a necessidade de um Sistema Gerenciador de Banco de Dados indicando que, como o Banco de Dados é uma coleção de dados inter-relacionados e o SGBD por sua vez, é uma coleção de programas que permite a criação e manipulação dos bancos de dados, seria então imprescindível a utilização de um Sistema Gerenciador de Banco de Dados.

2.5.2. Banco de Dados Relacional

Os bancos de dados relacionais são amplamente utilizados e, segundo Carvalho (2017) isso se dá principalmente pela premissa de integridade dada pelo modelo relacional em relação às alterações das estruturas e das informações. Isso ocorre porque seu mecanismo de vinculação de tabelas relacionadas torna o uso do modelo relacional de banco de dados muito seguro.

2.5.3. PostgreSQL

Conforme afirmado por Pichetti, Vida e Cortes (2021), o PostgreSQL, lançado em 1989, é um dos Bancos de Dados mais utilizados na atualidade e é utiliza o modelo de objeto-relacional que auxilia na solução para sistemas Web e diversas demandas de negócios.

PostgreSQL (2022) complementa sobre apresentando que, o PostgreSQL conquistou uma forte reputação por sua arquitetura comprovada, confiabilidade, integridade de dados, conjunto robusto de recursos, extensibilidade e dedicação da comunidade de código aberto por trás do software para fornecer soluções inovadoras e de desempenho consistentes. Também afirmam que o PostgreSQL é executado em todos os principais sistemas operacionais e é compatível com ACID (Atomicidade, Consistência, Isolamento e Durabilidade) desde 2001.

2.6. REACTJS

Roldán (2021) afirma que o React é uma biblioteca JavaScript de código aberto e adaptável com o foco na construção de interfaces de usuário complexas a partir de pequenos códigos separados chamados componentes. O React efetivamente torna as aplicações mais flexíveis, fáceis de manter e com um melhor desempenho, dando ao seu fluxo de trabalho um grande impulso, melhorando a velocidade sem afetar a qualidade.

React (2022) declara que o objetivo do ReactJS é o desenvolvimento de *UIs* interativas de forma fácil e que a estrutura das interfaces se atualize e renderizem na medida que os dados forem sendo alterados de maneira automatizada. Afirmam também que o ReactJS é baseado com componentes encapsulados que podem ser facilmente combinados para interfaces mais complexas e com partes em comum de forma reutilizável.

2.7. NODE.JS

Conforme citado por Simas et al. (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;”.

Zanetti e Oliveira (2021, p.8) afirmam que “O Node.js é um ambiente de servidor gratuito e de código aberto, que possibilita a criação de aplicações no lado servidor [...] com algumas vantagens em relação aos seus principais concorrentes [...], no que se refere a leveza, flexibilidade, suporte e produtividade.”. Pereira (2014, p.2) ressalta a utilidade do Node.JS afirmando que, “[...] é uma plataforma altamente escalável e de baixo nível, pois você vai

programar diretamente com diversos protocolos de rede e internet ou utilizar bibliotecas que acessam recursos do sistema operacional, principalmente recursos de sistemas baseado em Unix".

2.7.1. Npm

Npm (2022) relata que o npm pode ser considerado um gerenciador de pacotes para Node.JS criado em 2009 como um projeto de código aberto para auxiliar desenvolvedores JavaScript no compartilhamento de pacotes de módulos facilmente; O npm *Registry* é uma coleção pública de pacotes para o Node.JS, aplicativos web *frontend* (como o ReactJS) e para inúmeras outras finalidades dentre a comunidade JavaScript; Por fim, o npm é um cliente que, por meio de linhas de comando permite aos desenvolvedores instalar e publicar estes pacotes.

2.8. FRAMEWORK

Conforme afirmado por Freitas, Birnfield e Saraiva (2021, p.47) "Framework é uma abstração que une códigos comuns entre vários projetos de software, provendo uma funcionalidade genérica. Portanto, trata-se de uma forma mais simples de desenvolver aplicações pelo reuso de componentes.".

Oliveira e Zanetti (2020, p.101) complementam a utilidade dos frameworks dizendo que "A adoção de frameworks facilita o desenvolvimento de aplicações, pois implementa as funções mais comumente utilizadas em determinada parte do projeto.".

2.8.1. NestJS

Conforme definido por NestJS (2022), o Nest (ou NestJS) é uma estrutura utilizada para criar aplicações Node.js eficientes e escaláveis do lado do servidor. Ele usa JavaScript progressivo, é construído e possui suporte completo ao TypeScript (Mas ainda permite o uso apenas do JavaScript "puro" para o desenvolvimento) e combina elementos de Programação Orientada a Objetos, Programação Funcional e Programação Reativa Funcional. O Nest fornece um nível de abstração acima das estruturas comuns do Node.js, expondo suas APIs diretamente ao desenvolvedor, dando aos desenvolvedores a liberdade de usar a infinidade de módulos de terceiros que estão disponíveis para a plataforma subjacente.

MDN (2021, n.p.) resume a definição como sendo “Node.js é um ambiente de execução multi-plataforma em JavaScript que permite aos desenvolvedores produzirem aplicações para rede e *server-side* usando o JavaScript.”.

2.8.2. Bootstrap

Segundo Matos e Zabot (2020, p.100), “O Bootstrap é um framework front-end de código aberto para desenvolvimento rápido com tecnologias web. Inclui modelos de design com base em HTML e CSS, formulários, botões, tabelas, barras de navegação e muitos outros elementos.”. Ainda definindo o Bootstrap, Oliveira e Zanetti (2020, p.101) afirmam que, “O Bootstrap é um framework HTML, CSS e JavaScript para desenvolvimento de interfaces responsivas para aplicações na internet. Essas interfaces funcionam adequadamente em computadores e dispositivos móveis.”.

2.8.3. Material UI

Segundo MUI (2022), o Material UI é uma biblioteca de componentes React de Código aberto que implementa o Design Material do Google. Ele abrange diversos componentes visuais e funcionais previamente desenvolvidos e prontos para serem reutilizados e alterados conforme a necessidade do projeto, desde botões e caixas de texto com padrões de layout e funcionalidades bem desenvolvidas à componentes de temas completos para aplicação, apresentando modais e interfaces totalmente estruturadas com o design do Material UI.

2.8.4. Express

Segundo Sharma (2018), o Express é um framework *server-side* minimalista para o Node.js. Desenvolvido sobre o Node.js para torna-lo mais simples e fácil de ser gerenciado. Com o uso do Express temos uma API mais robusta, simples de ser configurada, com formas fáceis de receber dados dos usuários e enviá-los para o banco de dados.

Conforme citado por Express (2022, n.p), “O Express fornece uma camada fina de recursos fundamentais para aplicativos da web, sem obscurecer os recursos do Node.js que você conhece e ama.”, como complemento Lim (2019 afirma que é um framework utilizado por muitas empresas, e que auxilia as aplicações com funcionalidades como a de *middleware* e rotas.

2.8.5. TypeORM

TypeORM (2022) afirma que seu objetivo é oferecer suporte aos recursos do JavaScript e fornecer recursos adicionais que auxiliam no desenvolvimento de qualquer tipo de aplicação que utilize Banco de Dados, desde pequenas aplicações com algumas poucas tabelas à aplicações em grande escalas que podem possuir vários bancos de dados.

2.9. PROGRESSIVE WEB APP (PWA)

MDN (2021) definem PWA como o termo utilizado para descrever o estado moderno do desenvolvimento de aplicações web, envolvendo as melhores partes do desenvolvimento Web como a descoberta pelos mecanismos de busca, e complementá-los com funções de instalação, funcionamento sem conexão e com fácil sincronização e reconexão do usuário com o servidor caso a conexão retorne.

Segundo Pontes (2018), o objetivo de uma aplicação web progressiva é condicionar uma aplicação web a atender o maior número de usuários possíveis em todos os aspectos. A ideia de uma aplicação progressiva é disponibilizar a aplicação para a maior quantidade e variedade de dispositivos incluindo versões atualizadas e obsoletas dos navegadores e a possibilidade de uso mesmo sem a conexão com a internet.

2.10. JSON

Freitas, Birnfield e Saraiva (2021, p.13) geram a definição que, “JSON é o acrônimo de *JavaScript Object Notation* e se trata de um subconjunto da linguagem JavaScript, porém não se limita a ela, apesar de usar a mesma sintaxe. Esse formato foi pensado para implementação de intercâmbio de dados.”. Oliveira e Zanetti (2021, p.15) complementar salientando que o JSON é amplamente utilizado para transmissão de informações entre diferentes aplicações, até mesmo entre alguns tipos de bancos de Dados como o MongoBD.

> JSON e capaz de representar números, booleanos, textos, vazios, e listas (sequência ordenada de valores) e objetos (mapeamento de valores de texto) composto por estes valores (ou por outras listas e objetos). Ele não representa nativamente tipos complexos de dados como funções, expressões regulares, datas, e muito mais. (Objetos de dados por padrão inicializam como texto contendo os dados no formato ISO, enquanto eles não mudam, a informação não é completamente perdida.) Se você precisa preservar muitos valores, você pode transformar valores como eles são inicializados, ou prioritariamente descontinuados, para habilitar JSON para representar tipos de dados adicionais. (MDN, 2022, n.p.)

De acordo com Rodrigues, Silva e Neumann (2020, p.276), “O JSON é portável, não proprietário e independente de tecnologia. Todas as linguagens modernas e plataformas oferecem um ótimo suporte para produzir e consumir dados JSON”. Com isso, Freitas, Birnfield e Saraiva (2021) ainda afirmam que a vantagem do JSON em relação a outras formas de marcação está no fato de ser mais leve e muito mais simples de ser lido e entendido, tanto por pessoas quanto pelas linguagens de programação.

2.10.1. JSON Web Token (JWT)

Em relação ao JWT, Peyrott (2018) retrata o mesmo como um padrão para passar com segurança informações restritas nas requisições. Simplicidade, compacidade e usabilidade são as principais características de sua arquitetura. Embora sistemas muito mais complexos ainda estejam em uso, os JWTs têm uma ampla gama de aplicações.

Conforme ressaltado por Freitas, Birnfeld e Saraiva (2021, p.35), JSON Web Tokens “[...] é uma aplicação JSON utilizada na transferência de dados, que ocorre pelo envio de dados com os comandos POST ou pelo cabeçalho HTTP (header), de forma segura.”.

> Em aplicações que usam a arquitetura RESTful, é necessário que, a cada requisição realizada na API de back-end, se transmita dados sobre a identidade da pessoa. Como não seria uma boa prática (mas uma forte brecha de segurança) transferir o usuário e a senha todas as vezes, essa identificação é feita por um token, geralmente baseado em JavaScript Object Notation (JSON) Web Token (JWT) e emitido pelo authorization server mencionado na especificação. Dentro da especificação OAuth 2.0, ele é chamado de Bearer Token e definido pela RFC6750. (SIMAS; BORGES; COUTO, 2019, p.199)

Payrott (2018) descreve conclusivamente que vários padrões foram implementados no passado com o pretérito de gerar uma solução comum e segura na transferência destas informações encriptadas, mas que o JWT traz um formato simplificado, útil, eficiente e padronizado de realizar estas transferências de dados, e com isso tornando-se uma forma muito conhecida e cada vez mais utilizada de padrão.

2.11. CASCADE STYLE SHEET (CSS)

Conforme declarado por Oliveira e Zanetti (2020) CSS permite a customização da formatação de elementos do HTML, ampliando as possiblidades de exibição das informações aos usuários, tornando as interfaces mais versáteis e gerando uma identidade para o site.

Alves (2021, p.28) afirma que “As folhas de estilo são muito úteis ao desenvolvimento rápido de páginas web, uma vez que possibilitam a reutilização de códigos de formatação em vários arquivos HTML que compõem um site.”.

> Podemos considerar que a linguagem de folhas de estilo CSS diz como o conteúdo de um documento HTML deve ser exibido ao usuário, e não o que deve ser mostrado. Ela não possui comandos de execução de operações, apenas regras de formatação para serem associadas aos diversos elementos que formam uma página HTML. (ALVES, 2021, p.28).

Alves (2014, p.77) ainda complementa dizendo que, “Com elas, é possível definir características que podem ser aplicadas a diversos documentos HTML, sem a necessidade de se repetir as configurações em cada um deles.”.

2.11.1. SASS

Conforme definido pela Sass (2022), é uma linguagem de folha de estilo compilada para CSS. Ele permite o uso de variáveis, *nested rules*, *mixins*, funções e muito mais, tudo com uma sintaxe totalmente compatível com CSS. Sass ajuda a manter grandes folhas de estilo bem organizadas e facilita o compartilhamento de design dentro e entre projetos.

2.12. GIT

Segundo Monteiro et al. (2021, p.26), “A ferramenta Git é um dos sistemas de controle de versão e de controle de código-fonte distribuído mais conhecidos e utilizados pelas equipes de desenvolvimento de software.”.

Chacon e Straub (2022) conceitua Git como um sistema de controle de versão distribuído gratuito e de código aberto, destinado a projetos pequenos até projetos muito grandes com a respectiva velocidade e eficiência. Segundo Chacon e Straub (2014, p.12) “O Git não trata nem armazena seus dados desta forma. Em vez disso, o Git trata seus dados mais como um conjunto de imagens de um sistema de arquivos em miniatura. [...] tira uma foto de todos os seus arquivos e armazena uma referência para esse conjunto de arquivos.”.

2.12.1. Github

Conforme observado por Monteiro et al. (2021, p.44) sobre o Github, “Essa plataforma atua como um servidor remoto de controle de versão e hospedagem de repositórios de projetos, com os quais é possível interagir por meio do Git.”.

2.12.2. Repositório

Conforme resumido por MDN (2022), Em um sistema como o Git, um repositório é um local que hospeda o código-fonte de uma aplicação, juntamente com vários metadados. Git (2022) complementa informando que o Git é um sistema que facilita o controle de versão dos códigos, e o controle de versão é um sistema que registra as alterações deitas em um ou mais arquivos ao longo do tempo sendo possível recuperar versões especificas posteriormente.

2.13. DOCKER

Segundo Silva (2016 p.23), “O Docker lhe permite empacotar uma aplicação com todas as suas dependências em uma unidade padronizada para desenvolvimento de software.”. Monteiro et al. (2021, p.79) complementa afirmando que “o Docker é uma solução baseada em contêineres originalmente baseada em sistemas Unix, cujo principal objetivo é empacotar aplicações em contêineres e enviá-los e executá-los em qualquer lugar, a qualquer momento, quantas vezes for necessário.”.

2.14. CONCEITOS GERAIS

Apresentação, no Quadro 2, de outros termos e conceitos utilizados.

Quadro 2 – Conceitos Gerais

Conceito	Definição
Cache	É uma forma de armazenar dados que foram recebidos por respostas HTTP temporariamente para que possam ser usados em requisições HTTP subsequentes enquanto as condições de seu uso forem satisfeitas.
CORS	É um sistema que consiste na transmissão de HTTP *headers* que determina se navegadores vão bloquear código JavaScript de acessarem respostas provindas de requisições entre origens.
CRUD	CRUD (*Create, Read, Update, Delete*) é um acrônimo para as maneiras de se operar em informação armazenada. É um mnemônico para as quatro operações básicas de armazenamento persistente.

DOM	O DOM é um modelo de documento carregado pelo navegador. Este documento é representado através de uma árvore de nós, onde cada um destes Nós representa uma parte do documento.
Headers	É um campo de uma requisição ou resposta HTTP que passa informações adicionais, alterando ou melhorando a precisão da semântica da mensagem ou do corpo.
Mobile First	É uma abordagem de desenvolvimento e design da Web que se concentra em priorizar o design e o desenvolvimento para tamanhos de tela móvel em vez do design e desenvolvimento para tamanhos de tela de desktop.
Nome de Domínio	Um nome de domínio é um endereço de uma página na Internet. Nomes de domínios são usados em URLs para identificar a qual servidor uma página específica pertence.
Objeto	Objeto refere-se a uma estrutura de dados contendo dados e instruções para se trabalhar com estes dados.
Protocolo	Um protocolo é um sistema de regras que define como o dado é trafegado dentro ou entre computadores.
Rota	Na implementação de uma API em uma camada de serviço, uma rota é um componente de software que analisa uma solicitação e direciona ou roteia a solicitação para vários manipuladores dentro de um programa.
SEO	SEO (*Search Engine Optimization* - Otimização dos Mecanismos de Pesquisa) é o processo de fazer com que um site fique mais visível nos resultados da procura, também denominado melhoramento na classificação da busca.
SQL	SQL (*Structured Query Language*) é uma linguagem descritiva de computador projetada para atualizar, recuperar e calcular dados em bancos de dados baseados em tabelas.
UI	*User Interface* (UI) é qualquer coisa que facilite a interação entre um usuário e uma máquina. No mundo dos computadores, pode ser qualquer coisa, desde um teclado, um joystick, uma tela ou um programa.
W3C	Consiste de mais de 350 membros e organizações que, conjuntamente, desenvolvem a padronização da Internet, coordenam programas de divulgação, e mantém um fórum aberto para falar de Internet.

Fonte: Elaborado a partir de MDN (2021).

2.15. ESTADO DA ARTE

Esta seção apresentará algumas ferramentas já disponíveis no mercado e que se assemelham ao proposto projeto. Junto delas será disponibilizado uma listagem de vantagens e desvantagens utilizadas como base para o processo de levantamento de requisitos da aplicação.

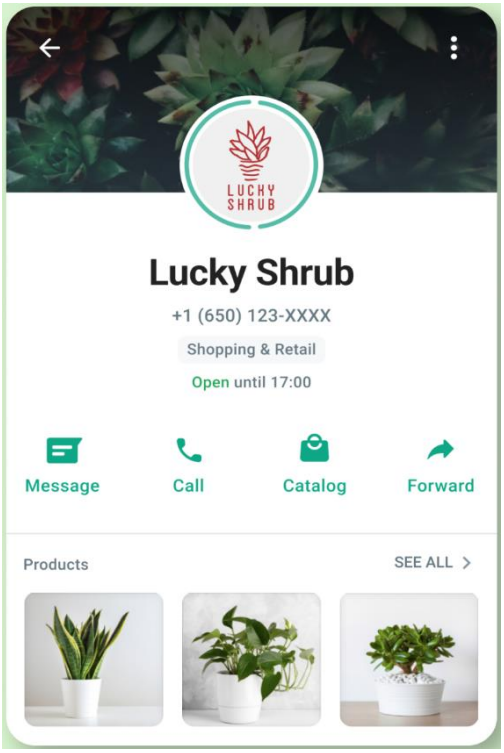
2.15.1 WhatsApp

Já conhecido por quase todos nos dias de hoje, o WhatsApp é uma aplicação destinada à troca de mensagens e chamadas de voz ou vídeo, e ainda é uma das ferramentas mais utilizadas por barbearias de pequeno porte para realizar a comunicação com seus clientes e fazer o agendamento de serviços.

Como evolução deste aplicado, WhatsApp (2022) apresenta o aplicativo WhatsApp Business como um aplicativo gratuito e voltado às necessidades de pequenas empresas, com a possibilidade de criar catálogos de produtos e serviços e permitindo automatizar alguns processos de respostas aos clientes.

A maior vantagem é o conhecimento geral e amplo da ferramenta pelos clientes que já estão familiarizados com o aplicativo convencional ao utilizá-lo em seu dia-a-dia. Já sua desvantagem é que é uma aplicação bastante genérica em relação a negócios, o que, para o caso de uma barbearia que presta serviços presenciais e necessita basicamente de um gerenciador de agendas e apresentação de seus produtos, torna o processo de automação do WhatsApp Business bastante complexo e limitado.

Figura 1 – WhatsApp Business



Fonte: WhatsApp (2022).

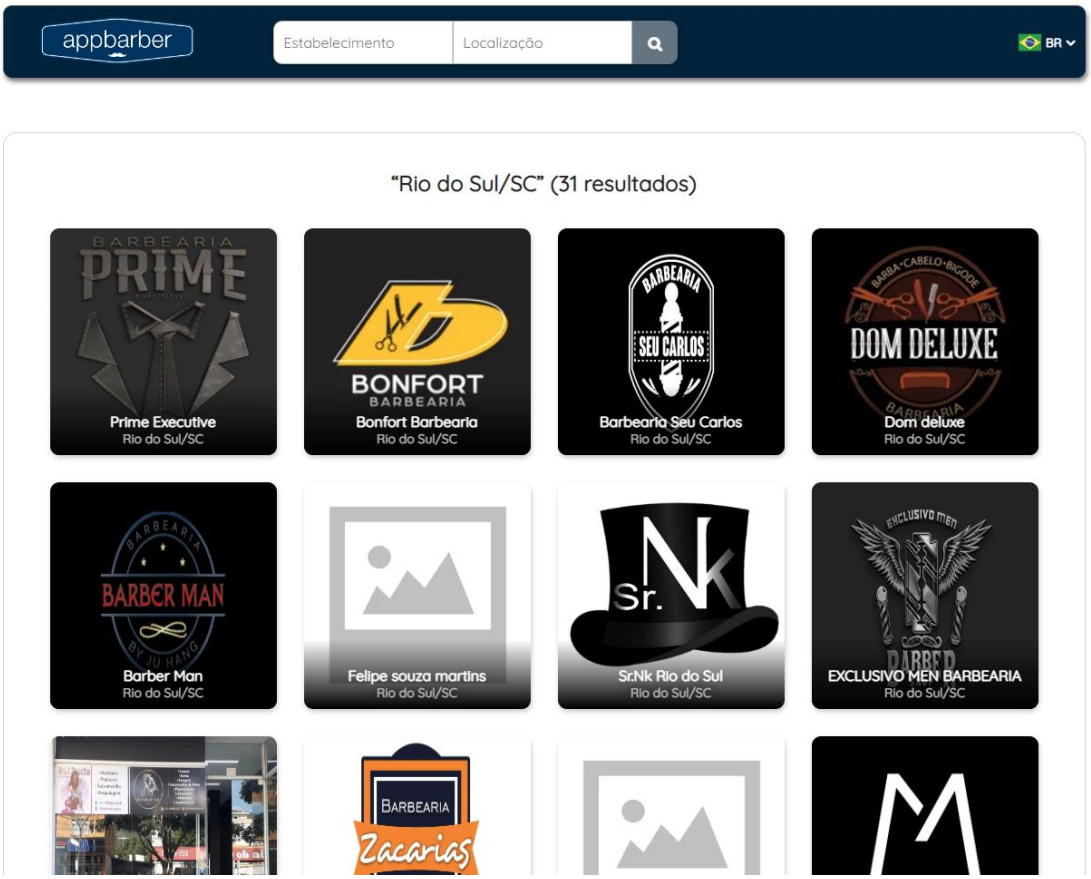
2.15.2 appbarber

O *appbarber* é uma aplicação muito semelhante ao proposto neste trabalho, trazendo uma interface Web com a disponibilização de várias barbearias e, ao selecioná-las traz seu respectivo detalhamento de localização, horários e imagens, além de permitir realizar o agendamento de serviços e avaliação das empresas. A Figura 18 traz um exemplo de uma listagem de barbearias de Rio do Sul – SC cadastradas na aplicação.

Com vantagens da aplicação temos a sua modularização, onde apresenta diversos módulos diferentes para atender a seus clientes, desde painéis administrativos para lembretes de serviços, gestão financeira, relatórios gerenciais e programa de fidelização de clientes.

Como principal desvantagem temos a responsividade de suas interfaces onde, principalmente se tratando de ambiente *mobile* a interface apresenta diversas inconsistências e deixa de apresentar vários botões de menu e opções em relação ao que temos no ambiente Desktop. Isso pode ser resolvido ao baixar a aplicação diretamente pelas Lojas da Google Play e App Store, mas por ser uma aplicação de utilização rápida é preferível a boa parte dos clientes que não seja necessário baixar a aplicação em seus dispositivos.

Figura 2 – Listagem de barbearias appbarber



Fonte: appbarber (2022).

3. METODOLOGIA DA PESQUISA

O presente trabalho se caracteriza como pesquisa aplicada e descritiva, pois seu objetivo é o desenvolvimento de um protótipo de aplicação web responsiva que possua rotinas para barbearias para o controle de suas agendas, serviços, disponibilidade e funcionamento, e rotinas para os respectivos clientes das barbearias, possibilitando a criação de contas, solicitação de agendamentos e visualização dos serviços agendados.

O trabalho buscou responder o seguinte problema de pesquisa: Como gerenciar as agendas, vendas, prestações de serviços e auxiliar na fidelização de clientes de barbearias de pequeno, médio e grande porte?

Após a finalização do levantamento bibliográfico, foi realizado um estudo mais aprofundado sobre o desenvolvimento de aplicações web responsivas, utilizando de ferramentas e tecnologias modernas, visando deixar a aplicação relevante e mitigando retrabalhos futuros e facilitando incrementações ao trabalho no futuro. Para realizar toda a comunicação das informações de forma segura foi optado por utilizar duas aplicações separadas, uma para o *frontend* e outra para o *backend*, ambas utilizando a linguagem JavaScript como base.

Baseado na pesquisa realizada, foi definido a utilização da IDE Visual Studio Code para o desenvolvimento, a linguagem de programação base foi Javascript, utilizando, para a criação das interfaces visuais do *frontend*, as bibliotecas ReactJS com auxílio dos frameworks Material UI e Bootstrap afim de tornar o projeto mais responsivo e visualmente atrativo. Já para o *backend*, foi definido a utilização da biblioteca NodeJS com auxílio das bibliotecas: Express para o desenvolvimento da API; TypeORM para as relações com Banco de Dados; e o framework NestJS afim de facilitar o processo de desenvolvimento e a integração de todas as bibliotecas no *backend.*

Com a finalidade de validar o projeto e a solução disposta neste trabalho, foi optado pela realização de uma entrevista avaliativa do projeto com um operador de barbearia, avaliando as suas respectivas dificuldades e se as soluções apresentadas pelo projeto realmente poderiam auxiliar no negócio, buscou-se assim pelos pontos positivos e pelas sugestões de implementações futuras ou necessárias para a entrada da aplicação do mercado.

Para possibilitar a persistência de dados, foi optado pela utilização do Banco de Dados PostgreSQL relacionado diretamente ao projeto *backend.* E para a comunicação com o *frontend*, foi determinado que seria realizado pela comunicação por meio de requisições REST, com auxílio do JSON Web Token para autenticação e autorização dos usuários à aplicação.

4. PROTÓTIPO DE SISTEMA DE GERENCIAMENTO DE BARBEARIAS (NAVALHAPP)

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

4.1 VISÃO GERAL DO SISTEMA

O objetivo principal do sistema é possibilitar que as barbearias possuam uma maneira de cadastrar e gerenciar os serviços que disponibilizam a seus clientes e também as agendas de seus funcionários, e que os clientes consigam fazer o agendamento dos serviços fácil e rapidamente por meio da aplicação web. O sistema foi dividido em duas partes principais, uma dela sendo o *frontend* e a outra sendo a API representando o *backend*.

A aplicação *frontend* foi subdividida em três outras partes, sendo a primeira referente ao administrador, onde somente operadores administrativos da aplicação NavalhaAPP terão acesso e será utilizada para gerenciar as novas barbearias.

A segunda parte é responsável pelo controle das próprias barbearias, destinado ao processo administrativo do negócio dentro do aplicativo, possibilitando o cadastro dos grupos de serviços, dos serviços, das agendas, horários de funcionamento e outras configurações, além de apresentar alguns relatórios simplificados sobre o andamento de seus negócios.

Por fim a última parte da aplicação *frontend* se refere à sessão destinada ao usuário final, ao cliente das barbearias, onde o mesmo poderá realizar seu cadastro na plataforma, visualizar as barbearias cadastradas e ativas e, ao selecionar uma das barbearias, poderá visualizar os grupos e respectivos serviços além dos horários de disponibilidade para efetivar um agendamento. E, destinado ao usuário final, será permitido também a visualização de seus próprios agendamentos efetivados, dando a oportunidade de contactar as respectivas barbearias.

O fluxo do sistema é um cliente servidor, onde o *frontend* apresentará as telas respectivas ao tipo de usuário que está acessando a plataforma, seja ele um administrador, uma barbearia ou um usuário final, todas estas telas se comunicarão com a API REST através do protocolo HTTP, quando a requisição chegar na API a mesma será processada pelo *backend* e caso não aconteça algum erro, realizará a ação solicitada e responderá como previsto ao *frontend*, sendo que a ação poderá ser de salvar, editar, excluir ou apenas de selecionar dados no banco de dados. Conforme demostrando o fluxo na Figura 3.

Figura 3 – Fluxo da aplicação

Fonte: acervo do autor (2022).

4.2 REQUISITOS

Os requisitos são divididos em requisitos funcionais que são responsáveis pelas atividades / rotinas que serão realizadas nos sistemas, e requisitos não funcionais que demonstram características de qualidades, podendo ser sobre segurança, performance, restrições etc.

No quadro 3, são apresentados os requisitos funcionais referentes à sessão administrativa da aplicação.

Quadro 3 – Requisitos Funcionais da Sessão Administrativa

Número	Nome	Descrição
RF01	Login	O sistema deve dispor de uma interface para o processo de autenticação dos Usuários Administrativos por meio de um formulário contendo campos para e-mail e senha. As demais interfaces desta sessão deverão requerer um usuário autenticado para serem acessadas.
RF02	Criar Barbearias	O sistema deve dispor uma rotina para criar novas barbearias no sistema, tendo como os campos os dados empresariais: Razão Social, CNPJ, Nome Fantasia, E-mail e Telefone; os dados de endereço: Estado, Cidade, Bairro, CEP, Logradouro, Número e Complemento; os dados para acesso: Senha e Confirmação de Senha; e por fim, a imagem com logo da barbearia.
RF03	Consulta de barbearias	O sistema deve dispor uma rotina para listar todas as barbearias cadastradas.
RF04	Alterar Barbearia	O sistema deve dispor de uma rotina que possibilite a alteração das informações cadastradas da barbearia.
RF05	Desabilitar Barbearia	O sistema deve apresentar um botão com a funcionalidade de desabilitar a barbearia, e, a partir deste, não permitir que os usuários finais visualizem suas informações.
RF06	Criar usuário administrativo	O sistema deve dispor uma rotina para criar novos usuários administrativos, tendo como os campos: nome, e-mail, senha e confirmação de senha.
RF07	Consulta de usuários administrativos	O sistema deve dispor uma rotina para listar todos os usuários administrativos

RF08	Alterar usuário administrativo	O sistema deve dispor de uma rotina que possibilite a alteração das informações cadastradas do usuário administrativo.
RF09	Desabilitar usuário administrativo	O sistema deve dispor de uma rotina para desabilitação de um usuário administrativo, com o objetivo de que, ao ser executada, impossibilite o acesso do usuário ao sistema.

Fonte: Acervo do Autor (2022).

No quadro 4, são apresentados os requisitos funcionais referentes à sessão empresarial da aplicação, destinada às barbearias e que são responsáveis pela administração de seus respectivos negócios.

Quadro 4 – Requisitos Funcionais da Sessão Empresarial

Número	**Nome**	**Descrição**
RF10	Login Usuário Barbearia	O sistema deve dispor de uma interface para o processo de autenticação dos Usuários Empresariais por meio de um formulário contendo campos para e-mail e senha. As demais interfaces desta sessão deverão requerer um usuário autenticado para serem acessadas.
RF11	Criar Serviços	O sistema deve dispor uma rotina para criar os serviços que serão disponibilizados pelas barbearias, tendo como campos: Grupo do Serviço onde será possível selecionar Grupos já cadastrados ou cadastrar novos, descrição, preço, tempo estimado para execução do serviço e status, sendo que este último campo poderá ter como valor “Habilitado” ou “Desabilitado”.
RF12	Consultar Serviços	O sistema deve dispor uma rotina para listar todos serviços cadastrados da barbearia.
RF13	Alterar Serviço	O sistema deve dispor de uma rotina para alterar as informações do serviço cadastrado.
RF14	Desabilitar Serviço	O sistema deve dispor de uma rotina para desabilitação rápida de um serviço, ao ser desabilitado o serviço passa a ser apresentado com Status “Desabilitado” e não deve mais ser apresentado aos usuários finais.
RF15	Criar Agenda	O sistema deve dispor uma rotina para criar agendas, tendo como os campos principais os campos: descrição, nome do funcionário responsável e Status; no mesmo formulário deverá ser possível adicionar múltiplos períodos de funcionando com os campos: dia da semana (domingo-sábado), horário inicial e horário final.
RF16	Consulta de Agendas	O sistema deve dispor uma rotina para listar todas as agendas cadastradas.
RF17	Alterar Agenda	O sistema deve dispor de uma rotina para alterar as informações da agenda.
RF18	Desabilitar Agenda	O sistema deve dispor de uma rotina para desabilitação rápida de uma agenda, ao ser desabilitada a mesma passa a ser apresentada com Status “Desabilitado” e não deve mais ser apresentada aos usuários finais.
RF19	Consulta de Pedidos	O sistema deve dispor de uma rotina para listar todos os pedidos, a lista deverá possuir as informações de Data agendada, status do pedido, nome do cliente, descrição abreviada dos serviços (até 2 nomes de serviços, e se

		houver mais inserir "..."), valor e opção para visualizar o pedido detalhadamente. A listagem deverá permitir que o operador empresarial filtre sua busca e ordene por qualquer uma das colunas.
RF20	Detalhamento de Pedido	O sistema deve dispor de uma rotina para apresentação detalhada de um pedido, apresentando as informações gerais do pedido: Data, Horário, Tempo estimado, Valor total, funcionário responsável, status e listagem dos produtos do pedido; informações do cliente emissor: Nome, Email, Telefone, Cidade, Bairro e Logradouro;
RF21	Alterar Status de Pedido	O sistema deve dispor, junto da rotina de Detalhamento de Pedido, uma rotina para alteração do status do pedido, sendo que por padrão, um pedido será emitido com Status "Aguardando" e este poderá ser alterado para "Cancelado" ou "Pago".

Fonte: Acervo do Autor (2022).

No quadro 5, são apresentados os requisitos funcionais referentes à sessão destinada aos usuários finais, os clientes das barbearias.

Quadro 5 – Requisitos Funcionais da Sessão do Cliente

Número	**Nome**	**Descrição**
RF22	Login Usuário Cliente	O sistema deve dispor de uma interface para o processo de autenticação dos Usuários Clientes por meio de um formulário contendo campos para e-mail e senha. As demais interfaces desta sessão deverão requerer um usuário autenticado para serem acessadas.
RF23	Cadastro de usuário cliente	O sistema deve dispor uma rotina para cadastro de novos usuários Clientes que poderá ser acessada por qualquer usuário não autenticado como Cliente. Esta rotina deverá conter como campos: nome completo, e-mail, senha e confirmação de senha, data de nascimento, CPF, número de telefone e endereço composto de Estado, Cidade, Bairro, CEP, Logradouro, Número e Complemento.
RF24	Filtragem de Barbearias	O sistema deve dispor de uma rotina de pesquisa que permita a filtragem das barbearias por nome, Estado e Cidade da barbearia.
RF25	Consulta de barbearias	O sistema deve dispor de uma interface para apresentação da listagem de barbearias cadastradas e ativas no sistema, apresentando suas respectivas imagens, nomes, cidades e estados.
RF26	Detalhamento de Barbearia	O sistema deve dispor de uma interface para o detalhamento da barbearia, apresentando seus agrupamentos de serviços e os serviços disponibilizados pela mesma, juntamente com informações da empresa: Nome, endereço, e-mail e telefone.
RF27	Consulta de Grupos e Serviços	O sistema deve dispor de uma rotina que possibilite a filtragem dos serviços e dos grupos de serviço concomitantemente.

RF28	Solicitar agendamento	O sistema deve dispor de uma interface para a solicitação de agendamento de um serviço, exibindo os barbeiros disponíveis e seus respectivos horários de disponibilidade.

Fonte: Acervo do Autor (2022).

Com base no próximo quadro são apresentados os requisitos opcionais que não fazem parte do principal escopo do projeto, porem serão requisitos para ser desenvolvidos em versões futuras. Podendo ser visto no Quadro 6.

Quadro 6 – Requisitos Opcionais

Número	**Nome**	**Descrição**
RF29	Alteração de usuário cliente	O sistema deve dispor de uma rotina que possibilite a alteração das informações cadastradas do usuário cliente logado no sistema.
RF30	Visualizar histórico de agendamentos	O sistema deve dispor de uma interface que permita a visualização do histórico de agendamentos feitos pelo usuário.
RF31	Esqueci minha senha	O sistema poderá realizar a recuperação de senha de usuários Clientes e Empresariais a partir do e-mail do usuário previamente cadastrado.
RF32	Imagem para Grupos de Serviços	O sistema permitirá vincular imagens a cada Grupo de Serviço das Barbearias.
RF33	Rotina de recomendação de Barbearia	O sistema permitirá analisar o perfil do usuário e recomendar barbearias que combinam com o perfil do usuário, por meio de histórico e/ou distância.
RF34	Rotina de recomendação de Serviços	O sistema permitirá analisar o perfil do usuário e recomendar serviços que combinam com o perfil do usuário, por meio de histórico e/ou distância.
RF35	Dashboard usuário administrador	O sistema permitirá ter um dashboard com informações relevantes de todas as barbearias, relatórios intuitivos e simplificado dos totalizadores das barbearias e da constância dos usuários finais.

Fonte: Acervo do Autor (2022).

A seguir no Quadro 7, são apresentados os requisitos não funcionais do sistema inteiro, levando em consideração a aplicação *frontend* e *backend*.

Quadro 7 – Requisitos Não Funcionais do Sistema

Número	**Nome**	**Descrição**
RNF01	Autenticação Usuário Empresarial (Barbearia)	O sistema deve gerar um *token* para o usuário empresarial com duração de 7 dias.
RNF02	Autenticação Usuário Cliente	O sistema deve gerar um *token* para o usuário cliente com duração de 7 dias.
RNF03	Autenticação Usuário Administrativo	O sistema deve gerar um *token* para o usuário administrativo com duração de 7 dias.
RNF04	Banco de Dados	O banco de dados deve utilizar o SGBD PostgreSQL.
RNF05	Criptografia	Todas as senhas devem utilizar um sistema de criptografia *Hash* antes de ser gravadas no banco de dados.

RNF06	Responsividade	A aplicação *frontend* deve ser responsivo para celulares, *tablets* e *desktop*.
RNF07	*Frontend*	A linguagem utilizada para desenvolver deve ser Javascript, utilizando a biblioteca ReactJS e os frameworks Bootstrap e Material UI.
RNF08	*Backend*	A linguagem utilizada para desenvolver o *backend* deve ser o Javascript utilizando a biblioteca NodeJS e os frameworks Express e NestJS.

Fonte: Acervo do Autor (2022).

Com base nos requisitos funcionais e requisitos não funcionais foram elaborados diagramas de funcionamento das barbearias e, a partir destes, os protótipos de telas que serão apresentados na próxima sessão.

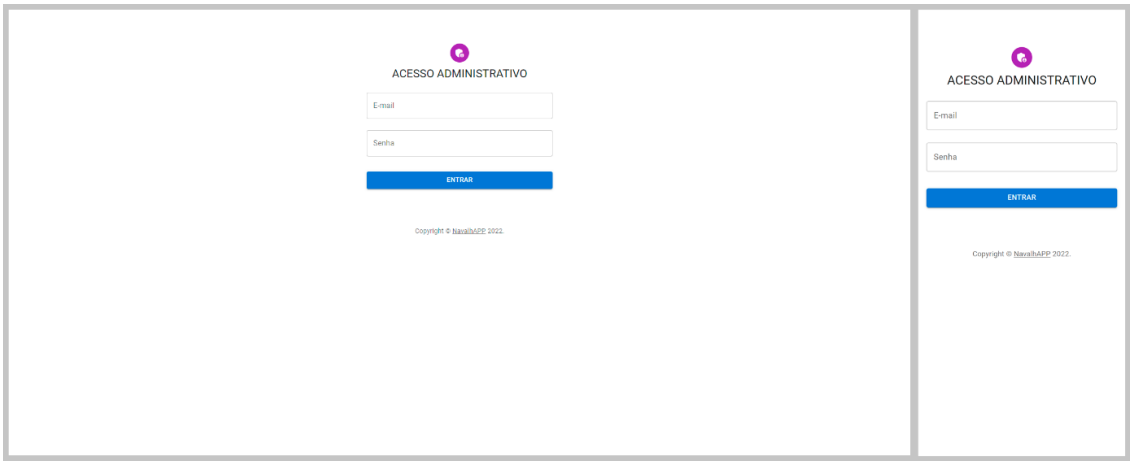
4.3 PROTÓTIPOS

Os protótipos tem por objetivo simplificar e demonstrar como as principais interfaces da aplicação são apresentadas, são desenvolvidos com base nos requisitos funcionais, garantindo o cumprimento dos mesmos. Cada protótipo pode contemplar um ou mais requisitos e se apresentam em visualização *desktop* e *mobile*.

4.3.1 Protótipos da sessão Administrativa

A seguir, na figura 4, é apresentada a interface de autenticação de um Usuário Administrativo através de um formulário contendo os campos de e-mail e senha, representando o RF01.

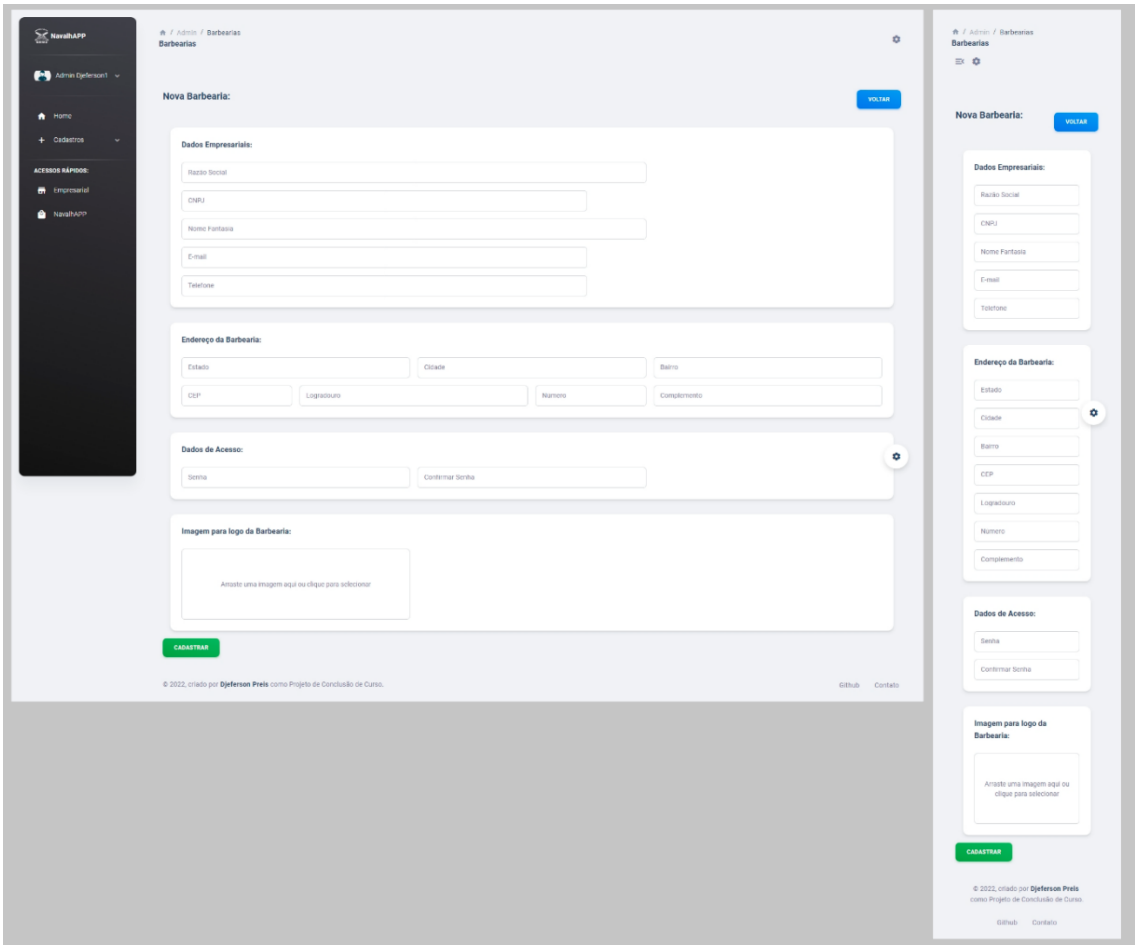
Figura 4 – Protótipo interface de autenticação de usuário Administrativo



Fonte: Acervo do Autor (2022).

A Figura 5 apresenta o RF02, uma interface para o cadastro de novas barbearias no sistema, contendo todos os campos determinados pelo requisito funcional distinguidos em sessões para facilitar o preenchimento e entendimento das informações necessárias. A mesma interface é reutilizada no RF04, recebendo apenas a alteração do título do botão “Cadastrar” para “Alterar” e o preenchimento completo do formulário a partir das informações previamente cadastradas do registro selecionado para edição.

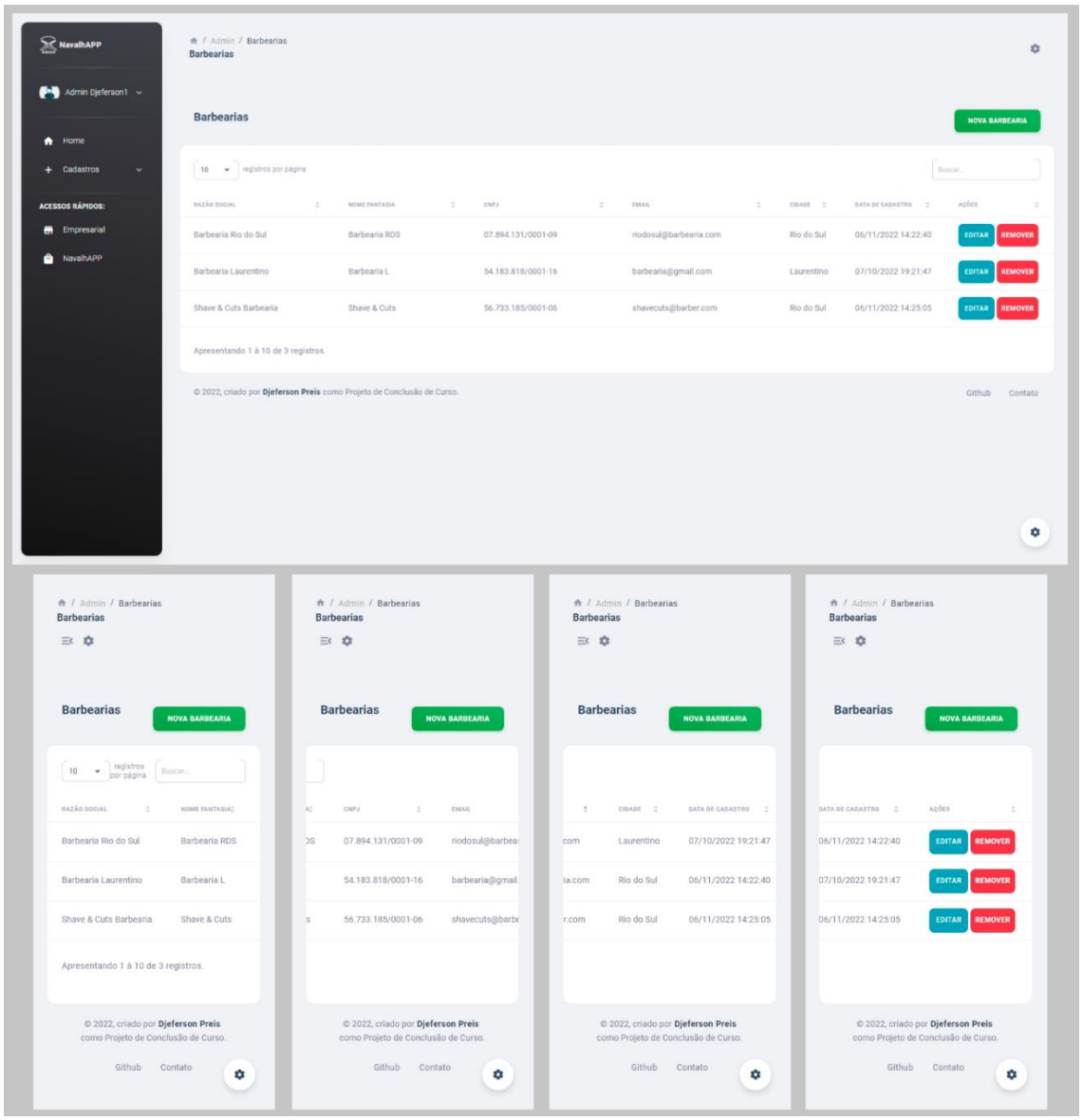
Figura 5 – Protótipo interface de cadastro de novas Barbearias



Fonte: Acervo do Autor (2022).

Na Figura 6 apresenta-se os requisitos funcionais: RF03, com a listagem das barbearias cadastradas no sistema; RF05, com a opção para desabilitar e reabilitar as Barbearias no sistema; e o acesso para o RF04 que ocorre pelo pressionamento do botão “Editar”, sendo apresentado em cada registro com a finalidade de selecionar e exibir o formulário para alteração do mesmo.

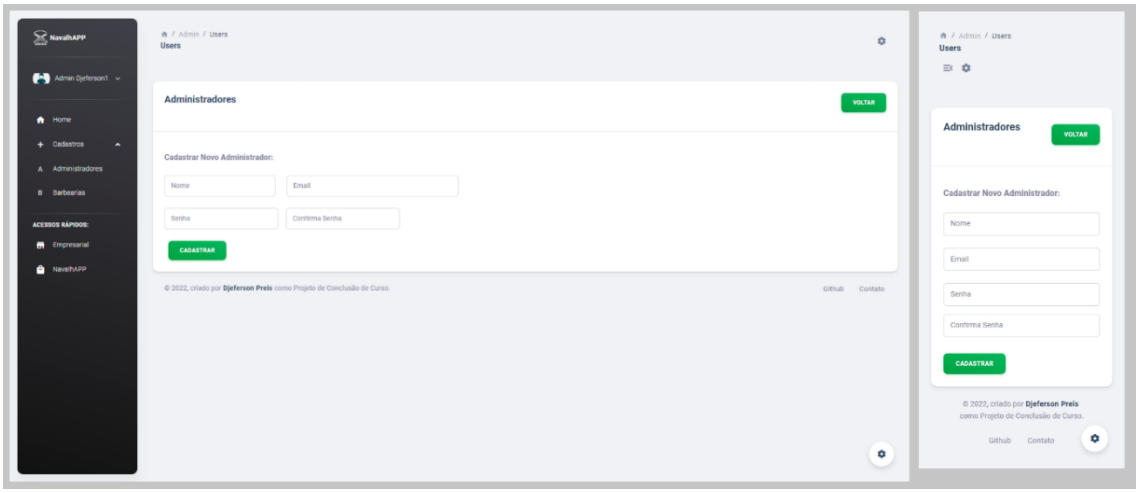
Figura 6 – Protótipo interface de listagem de Barbearias



Fonte: Acervo do Autor (2022).

Para representação do RF06, a interface para o cadastro de novos usuários administrativos possui um formulário bem simplificado e só poderá ser acessada por outros usuários Administrativos. O protótipo da interface em questão é apresentado na Figura 7.

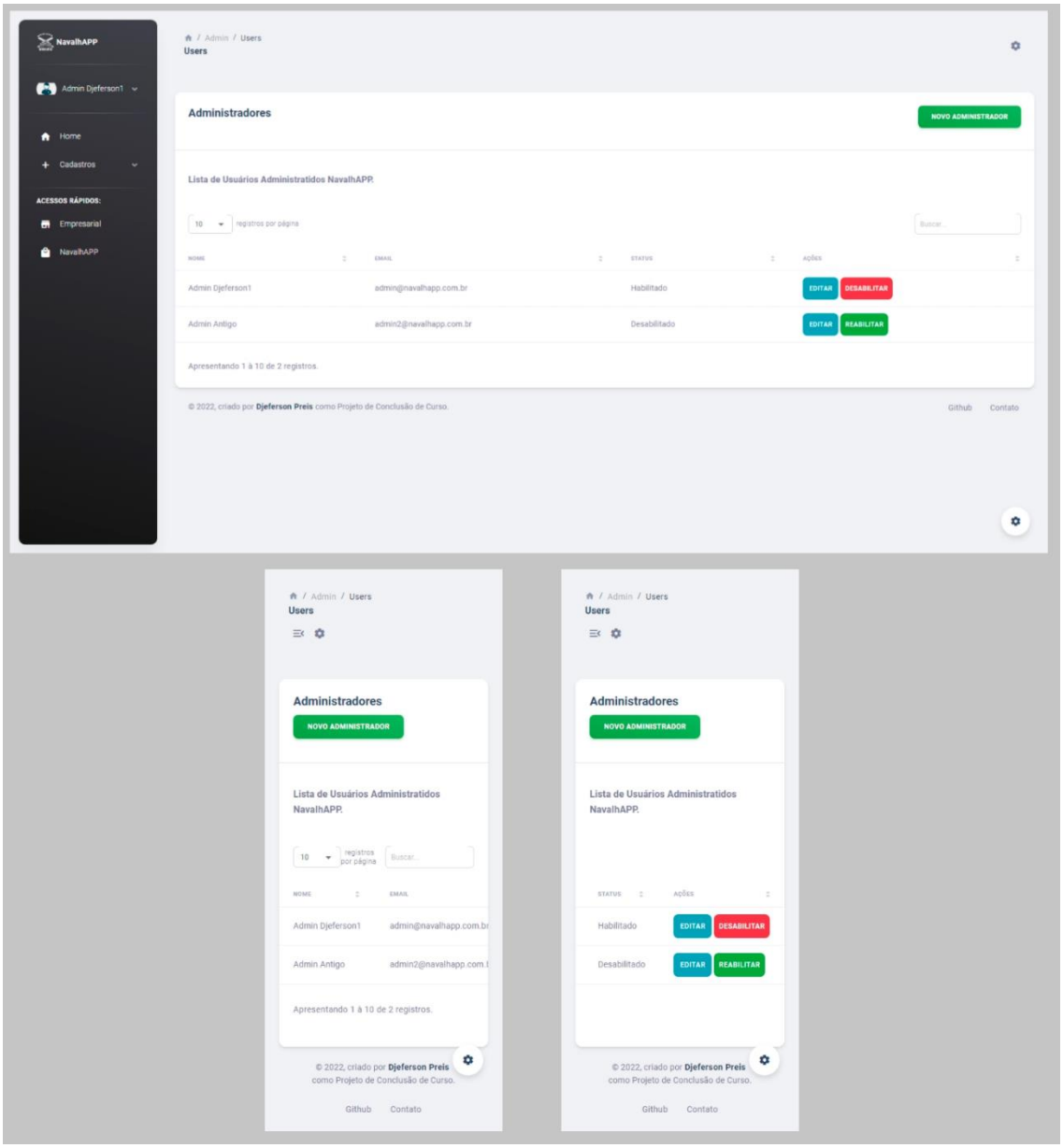
Figura 7 – Protótipo interface de cadastro de Usuário Administrativo



Fonte: Acervo do Autor (2022).

O mesmo padrão de layout é apresentado em outras interfaces, alterando seu conteúdo e deixando os processos mais fáceis de interpretar. Por conta disto, a Figura 8 apresenta os requisitos funcionais: RF07, com a listagem de usuários administrativos cadastrados; RF08, com o botão de “Editar” que, ao ser pressionado exibe a mesma interface de cadastro determinada para o RF06, mas previamente preenchida e com o botão “Cadastrar” alterado para “Alterar”; e o RF09, pelo botão de “Desabilitar” ou “Reabilitar” dependendo do estado prévio do registro.

Figura 8 – Protótipo interface de listagem de Usuários Administrativos

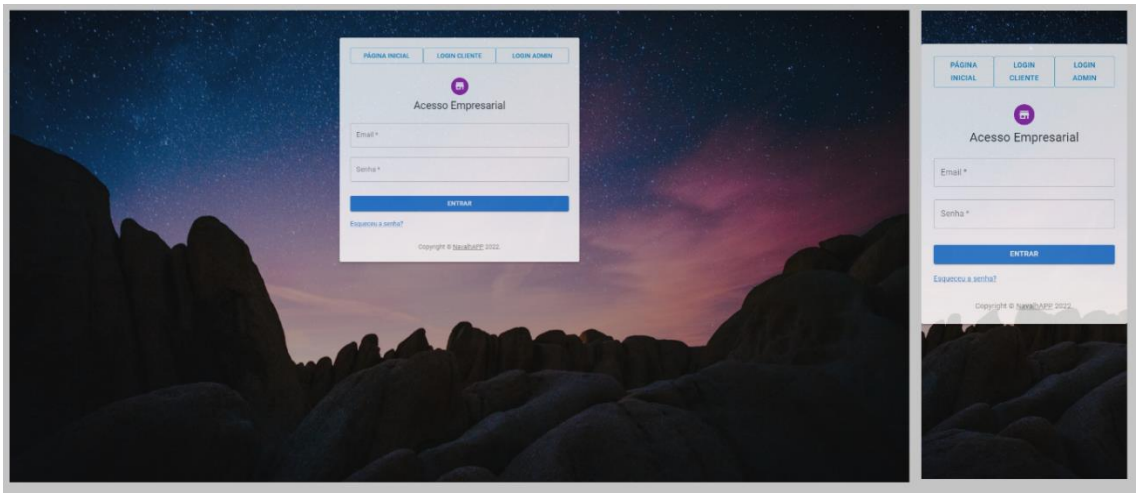


Fonte: Acervo do Autor (2022).

4.3.2 Protótipos da Sessão Empresarial

A seguir, na figura 9, é apresentada a interface de autenticação de um Usuário Empresarial, possuindo acessos para outras interfaces importantes e um formulário contendo os campos de e-mail e senha para autenticação da empresa, representando o RF10.

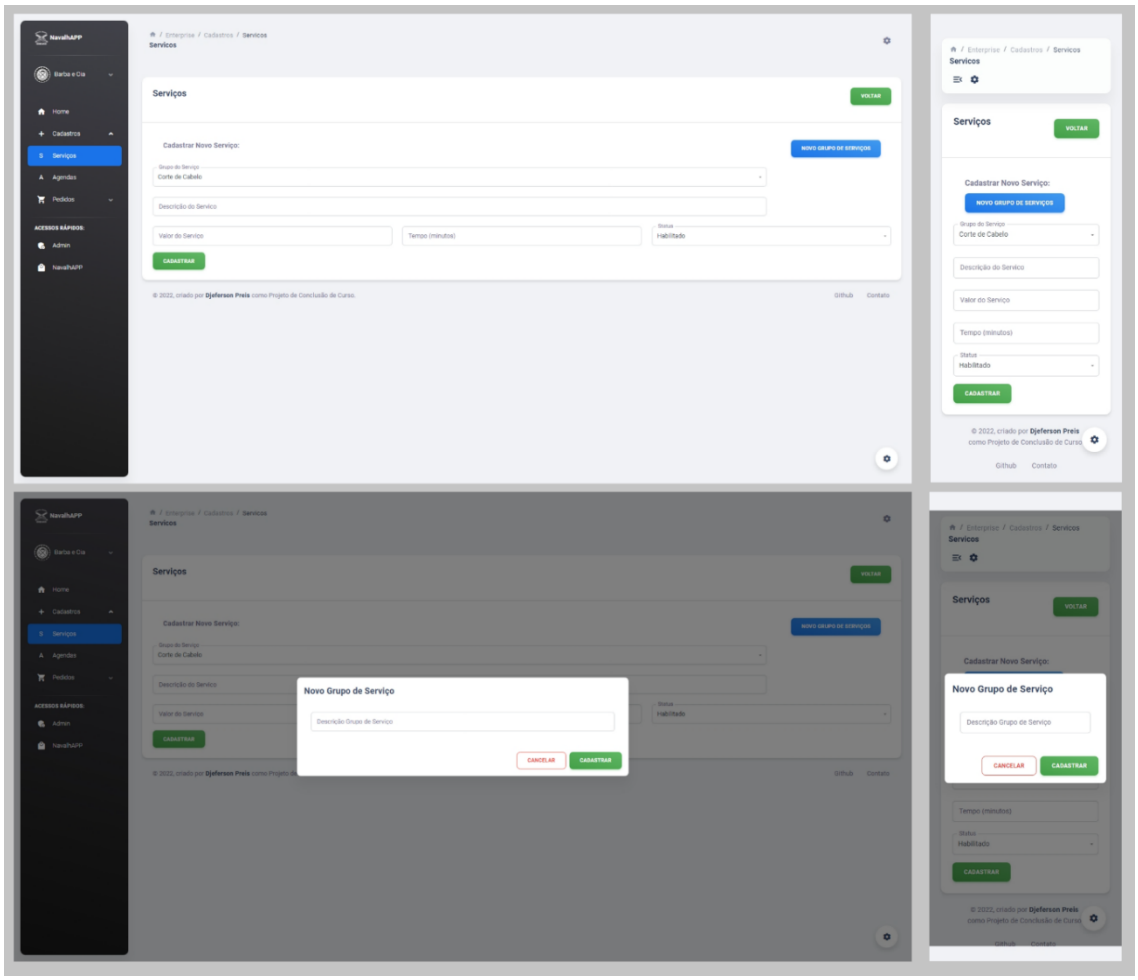
Figura 9 – Protótipo interface de autenticação de usuário Empresarial



Fonte: Acervo do Autor (2022).

Após estar autenticado como empresa, o operador poderá realizar o cadastro de seus respectivos serviços que serão disponibilizados aos clientes, para isto a Figura 10 apresentada abaixo representa o formulário de cadastro de novos Serviços, contendo campos simples para seleção do Agrupamento no qual o serviço estará, a descrição, valor, tempo estimado para execução do serviço e status, representando “habilitado” para os produtos que serão apresentados ao cliente final. Também havendo um botão “Novo grupo de serviços”, onde ao ser pressionado uma sub-interface é apresentada com a possibilidade de cadastrar novos grupos de serviços e ao salvar o mesmo grupo já se torna opção no campo de seleção “Grupo do Serviço” do formulário principal. Esta interface representa o RF11, e é reaproveitada para o RF13 onde o formulário é exibido com os dados previamente carregados e com o botão “Cadastrar” alterado para “Alterar”.

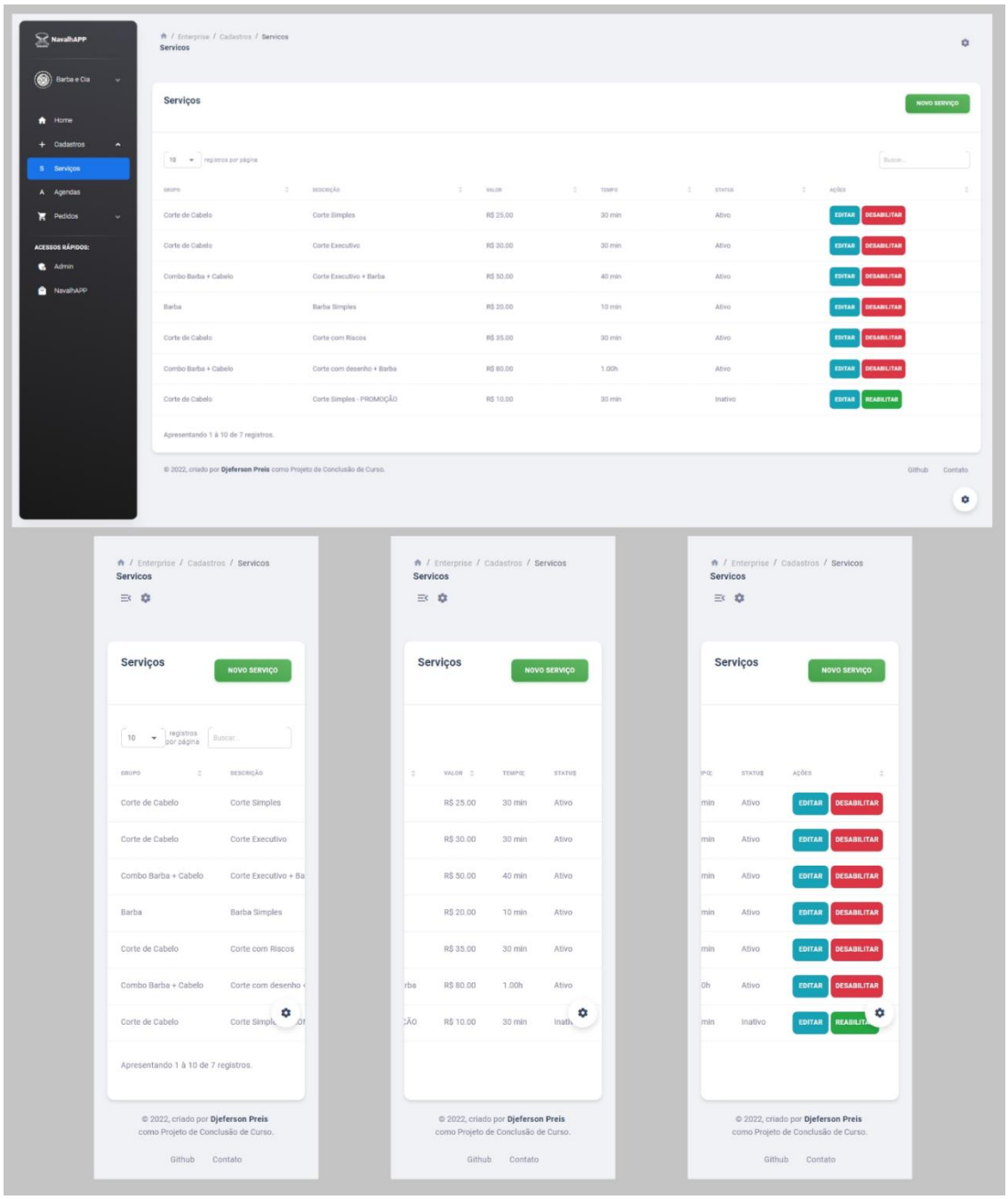
Figura 10 – Protótipo interface de Cadastro de Serviços



Fonte: Acervo do Autor (2022).

No mesmo acesso de menu do cadastro de serviços tem-se a respectiva listagem, representando o RF12 e as formas de acesso para RF11, RF13 e RF14, conforme apresentado na Figura 11, em uma interface contendo uma tabela com a listagem de serviços cadastrados, botões para o cadastro de novos serviços, alteração de serviços pré-existentes e para a desabilitação ou reabilitação de um serviço. Ao pressionar “Desabilitar” o status do registro passa a ser desabilitado, não sendo mais apresentado aos clientes finais e o botão torna-se “Reabilitar”, que ao ser pressionado realiza o processo inverso.

Figura 11 – Protótipo interface de Listagem de Serviços



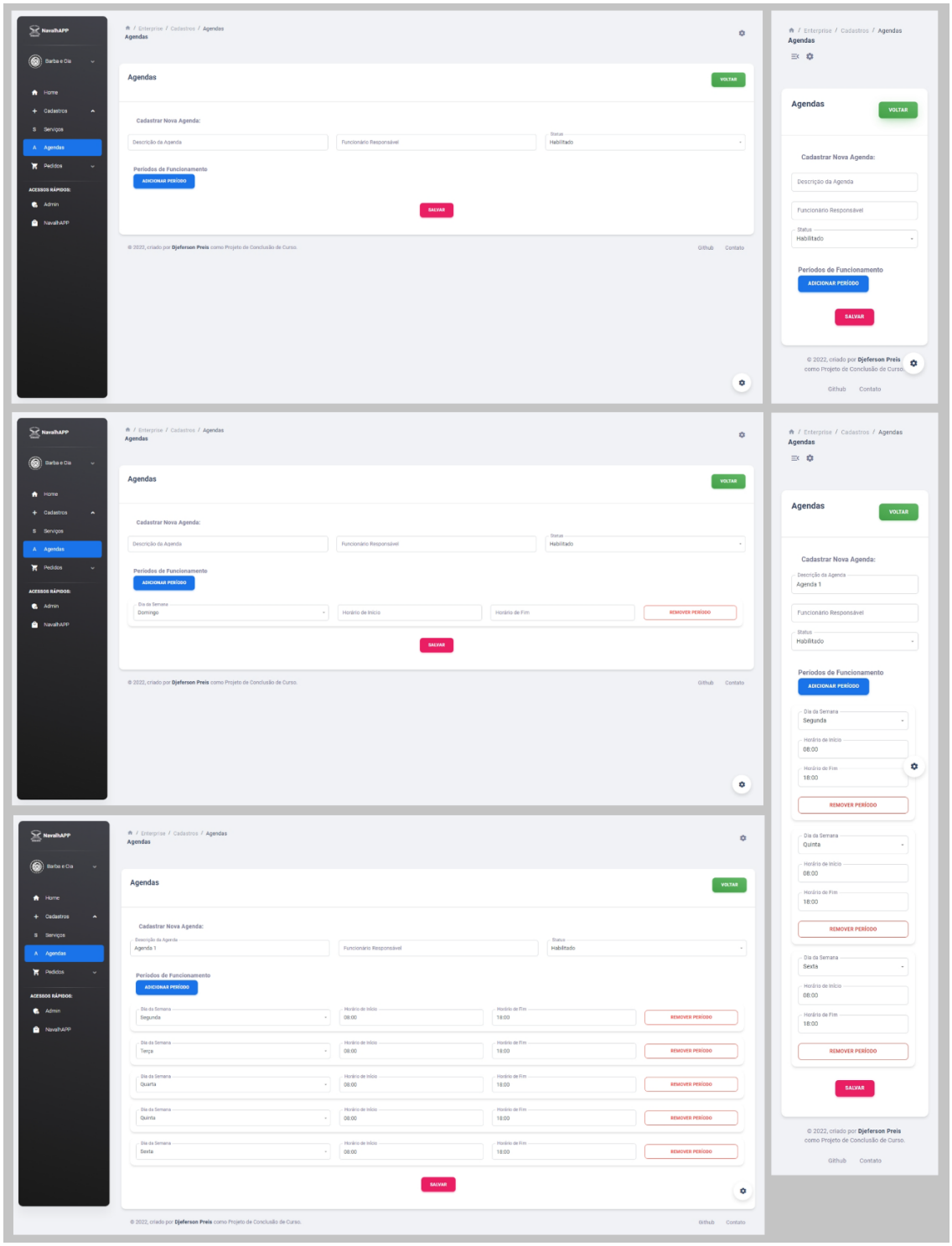
Fonte: Acervo do Autor (2022).

As agendas serão utilizadas como parâmetro para identificar os momentos em que as barbearias estarão ou não trabalhando, portanto, em seu cadastro temos, além de uma identificação por meio de uma descrição e do nome do funcionário responsável pela agenda, um botão onde poderá ser inserido inúmeros registros diferentes de horários de atendimento.

Desta forma, a Figura 12 apresenta o formulário para o cadastro de novas agendas e, para cada pressionamento do botão “Adicionar período” novos campos são apresentados contendo campos para informar o dia da semana, horário de início e de término do expediente, além de

um botão “Remover período” que, conforme indicado, remove o registro de período de funcionamento em questão. Representando assim o RF15 com a criação de novas agendas e também o mesmo formulário é reutilizado para o RF17, para alteração dos registros.

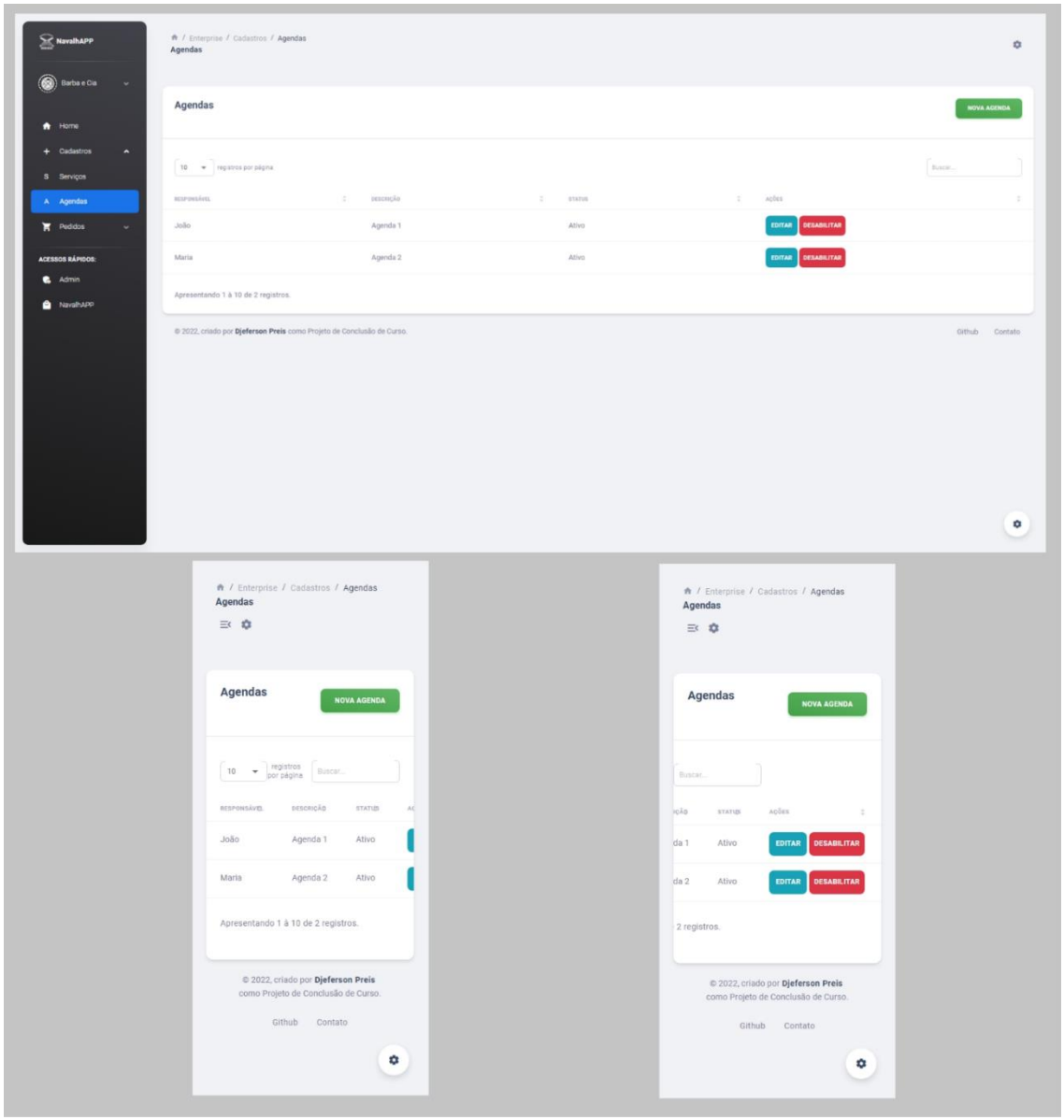
Figura 12 – Protótipo interface de Cadastro de Agendas



Fonte: Acervo do Autor (2022).

Para a representação do RF16 com a consulta de agendas, e as formas de acesso para RF15, RF16 e RF17, respectivamente para o cadastro, alteração e desabilitação de agendas da barbearia, a Figura 13, apresenta uma interface contendo a tabela com a listagem de agendas com informações simplificadas e botões para o acesso aos demais. Como já padronizado em outras interfaces, ao pressionar “Desabilitar” o status do registro passa a ser desabilitado, não sendo mais apresentado aos clientes finais e o botão torna-se “Reabilitar”.

Figura 13 – Protótipo interface de Listagem de Agendas

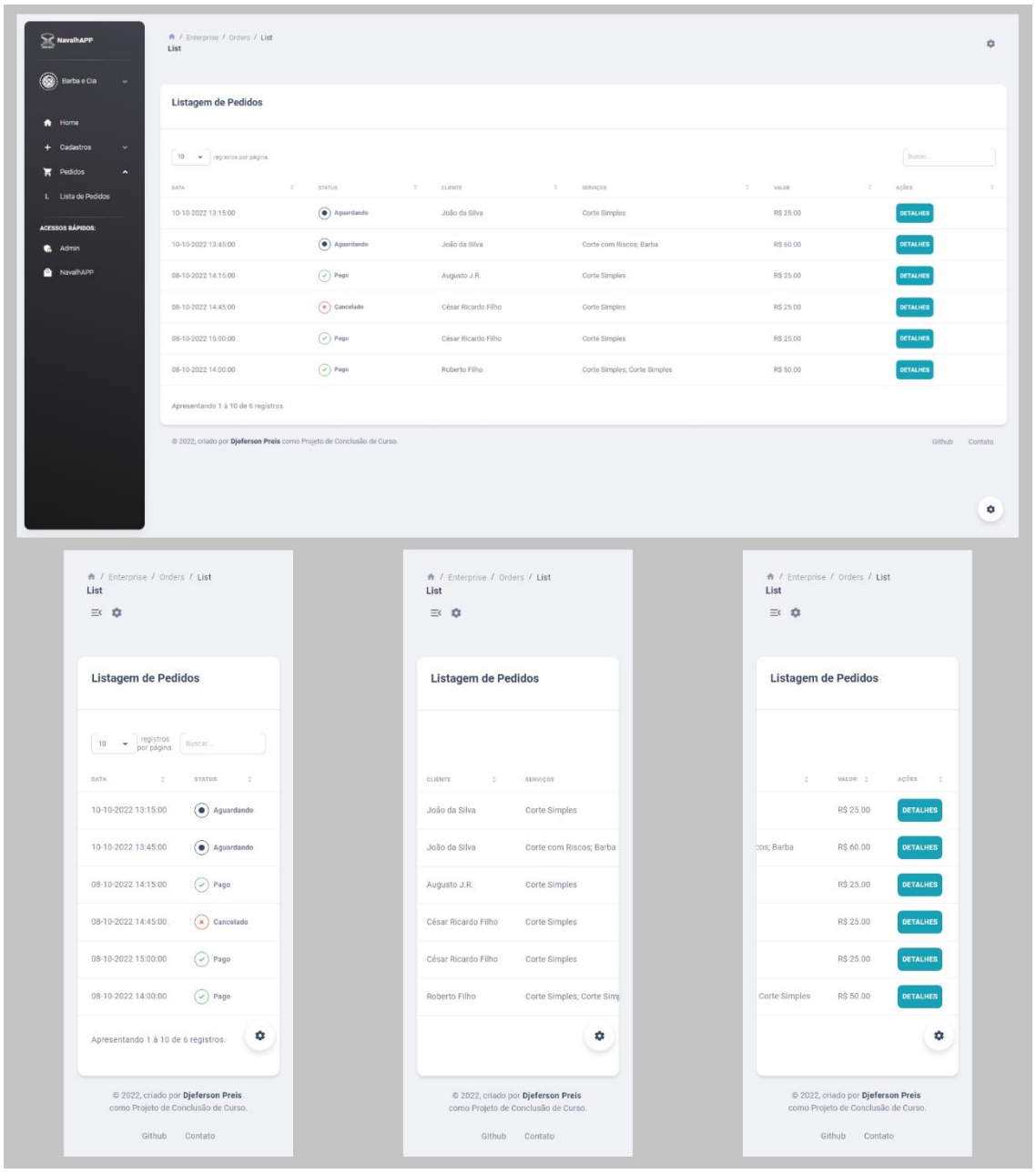


Fonte: Acervo do Autor (2022).

Uma das interfaces que serão mais utilizadas pelos usuários administrativos será a interface de pedidos, e para isso, representando o RF19 para consulta de pedidos temos uma interface contendo a tabela de pedidos com informações simplificadas como: Data inicial do

agendamento; o Status do pedido, sendo possível os status: “Aguardando” para pedidos ainda não atendidos, “Pago” para pedidos concluídos e “Cancelado”; em sequência, temos ainda os campos de: nome do cliente emissor do agendamento; uma breve lista de serviços, até o máximo de 2 itens listados; valor total do pedido e um botão para acessar o detalhamento completo do pedido. A Figura 14 representa esta listagem.

Figura 14 – Protótipo interface de Consulta de Pedidos

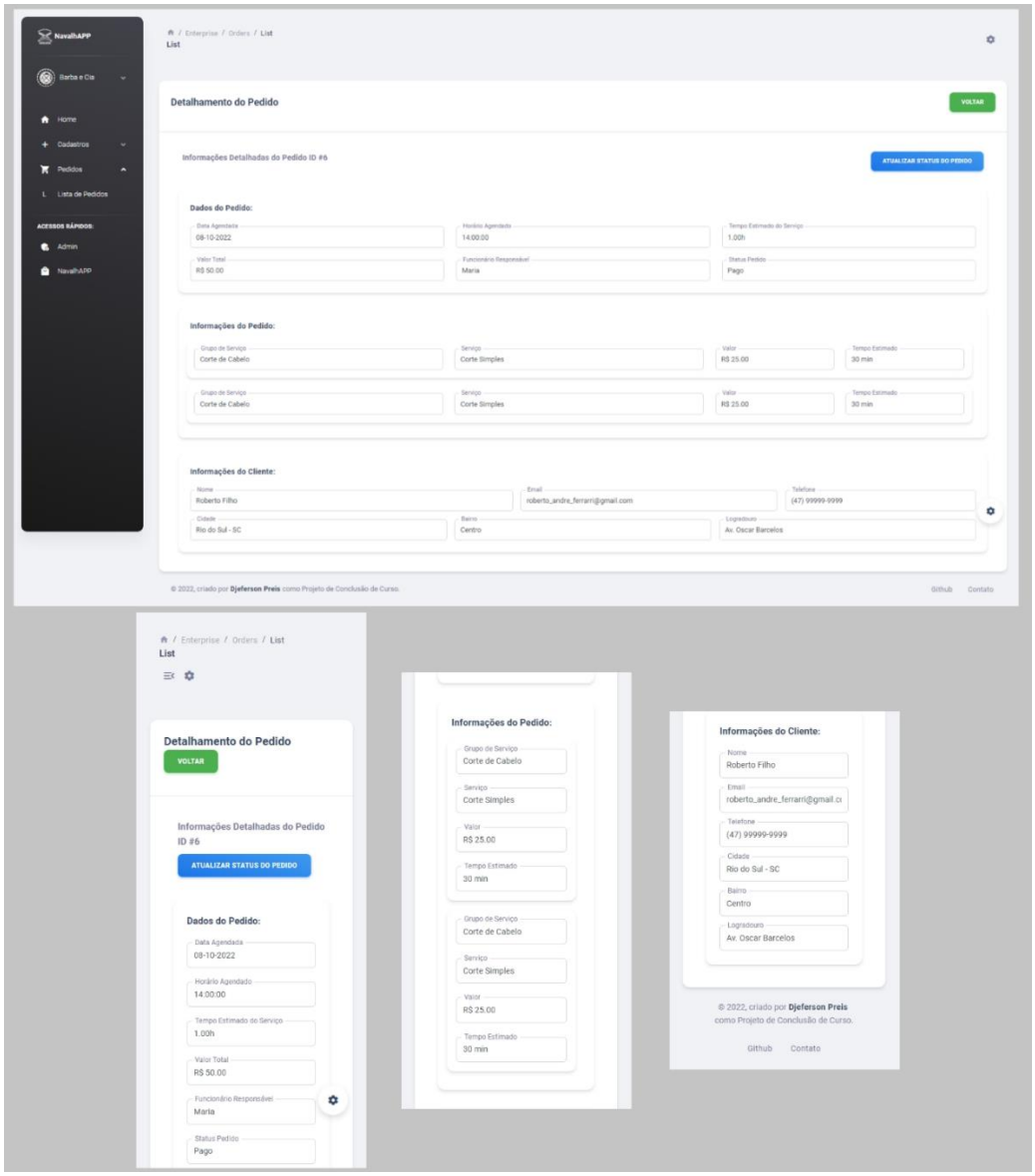


Fonte: Acervo do Autor (2022).

Como a consulta de pedidos apenas apresenta uma série de informações principais e não totais das solicitações de agendamentos, tem-se a interface representada pela Figura 15,

contendo uma série de sessões diferentes envolvendo todas as informações referentes ao pedido, desde as informações do próprio pedido como a data e hora de agendamento, estimativa de tempo para conclusão do serviço (baseado na soma de tempo necessário para cada serviço solicitado), valor total, nome do funcionário responsável (baseado na agenda) e status do pedido; também há uma sessão com a listagem de serviços solicitados, que apresenta o respectivo grupo, serviço, valor e tempo estimado de cada serviço selecionados pelo usuário em seu agendamento; e por fim algumas breves informações do cliente como nome, contato e seletivas informações de endereço. Esta interface representa o RF20 com o detalhamento do pedido além de apresentar o botão “Atualizar status do pedido” que serve de acesso para o RF21.

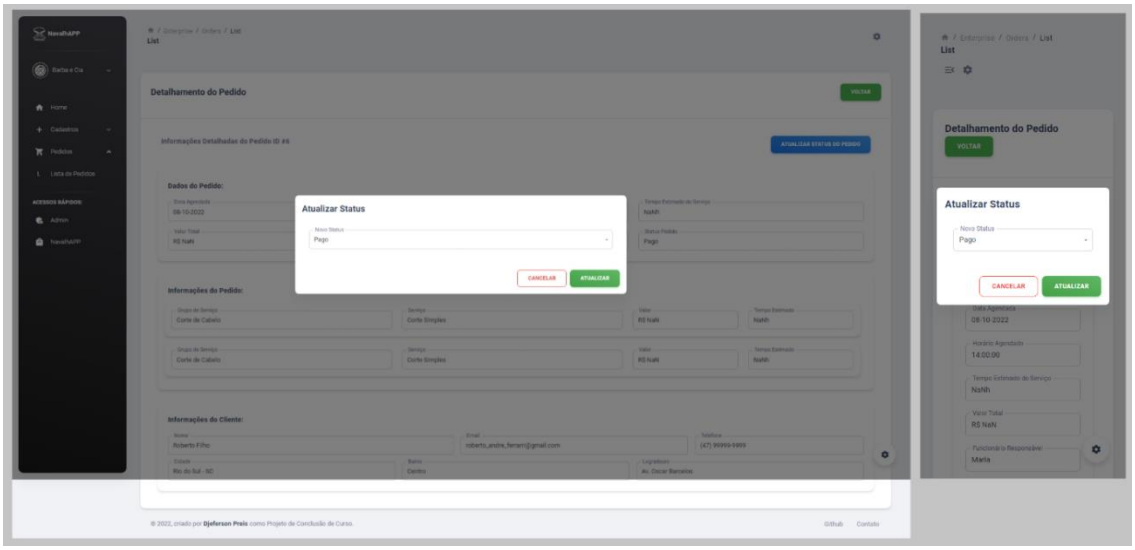
Figura 15 – Protótipo interface para detalhamento de Pedido



Fonte: Acervo do Autor (2022).

Pela necessidade de controle dos pedidos, o botão “Atualizar status do pedido” representado na interface de detalhamento leva à apresentação de uma sub-interface contendo um pequeno formulário, apenas com o campo “Novo Status” onde é possível alterar o status do pedido selecionado para “Aguardando”, “Pago” ou “Cancelado”, a sub-interface apresenta também os botões de: “Atualizar”, para aplicar a respectiva alteração; e “Cancelar” para interromper e cancelar a ação de alteração. Representando assim o RF21, Alterar Status de Pedido, conforme apresentado pela Figura 16.

Figura 16 – Protótipo interface de ação de alteração de status de Pedido



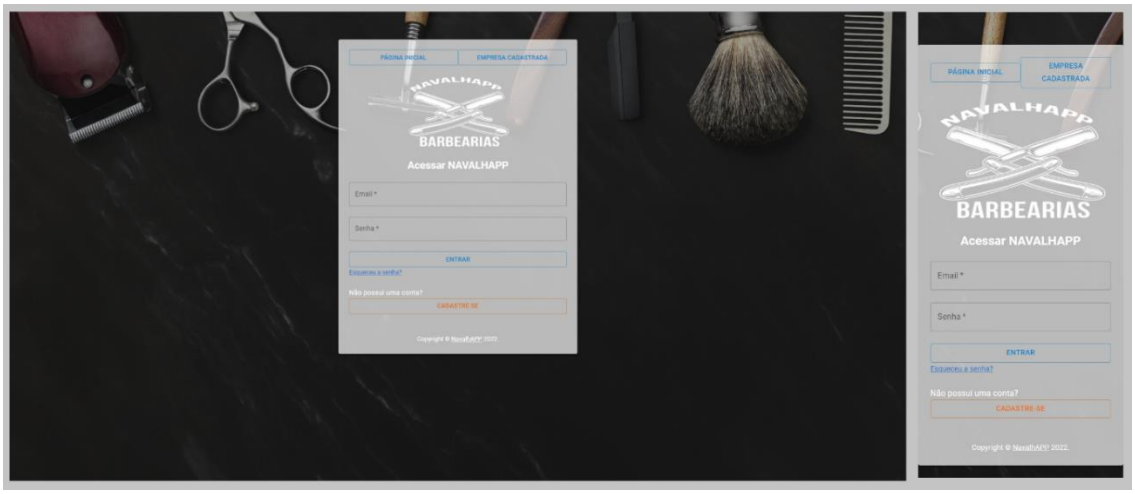
Fonte: Acervo do Autor (2022).

4.3.3 Protótipos da Sessão do Usuário Final

A sessão destinada ao usuário final será a sessão com maior número de acessos, principalmente de novos usuários e que é destinada aos clientes e futuros clientes das lojas ativas na aplicação. Sabendo disso foram realizadas algumas pequenas alterações na estrutura de layouts, com a finalidade de tornar a interação dos usuários mais simples além de deixar a aplicação mais intuitiva. Grande parte das interações são realizadas a partir de ações de pressionamento em cards e com a apresentação de formulários em sub-interfaces.

Para dar início ao acesso na aplicação será necessário efetivar o processo de Login e, como representado pela Figura 17, temos a rotina representando o RF22, Login Usuário Cliente, contendo um formulário simples com os campos de “Email” e “Senha”, e botões para acessar, cadastrar-se e até mesmo de ação no caso do esquecimento de senha.

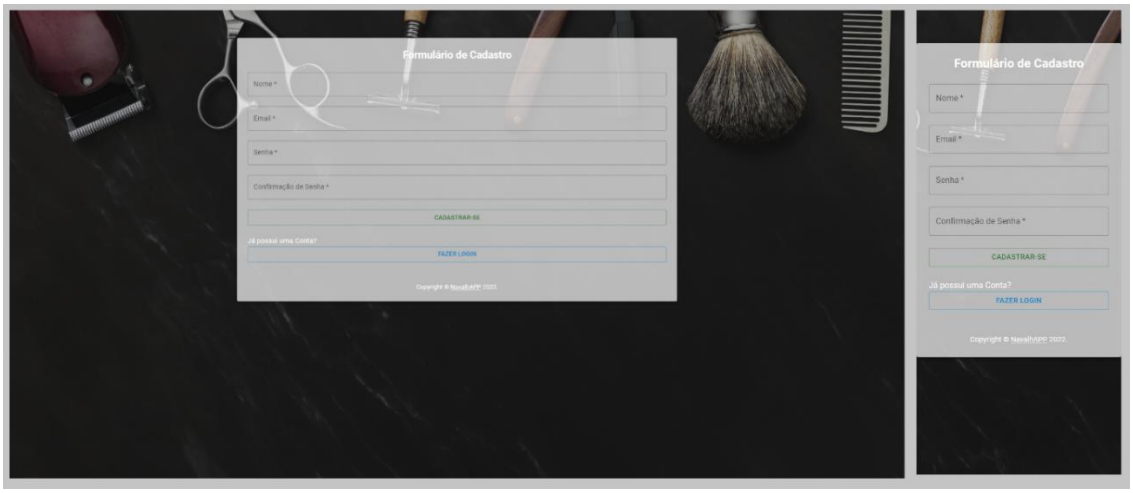
Figura 17 – Protótipo interface de Login de Usuário Final



Fonte: Acervo do Autor (2022).

Conforme representado na interface de login, o usuário final tem a opção de se cadastrar diretamente pela aplicação, sendo assim, ao pressionar o botão para se cadastrar, a interface exibe um card contendo um formulário simplificado com “Nome”, “Email”, “Senha” e “Confirmação de Senha” além de um botão de confirmar sua solicitação de cadastro, também havendo opção para retornar à tela de login. A Figura 18, representando o protótipo da interface em questão, que é relacionada ao RF23, Cadastro de usuário cliente.

Figura 18 – Protótipo interface de Cadastro de Usuário Final

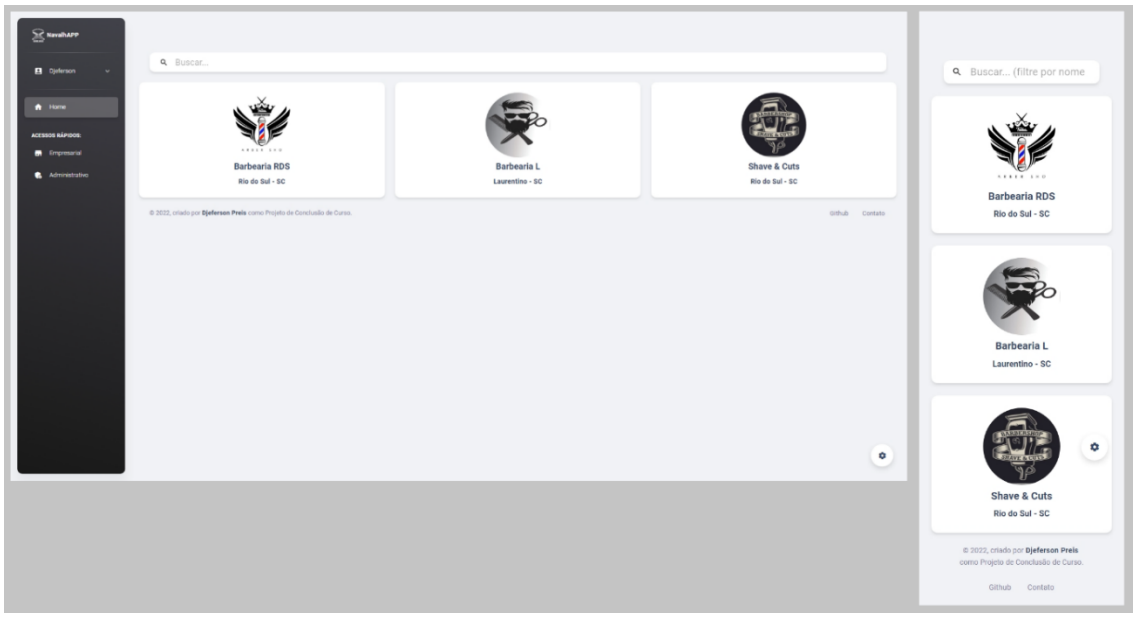


Fonte: Acervo do Autor (2022).

Após estar autenticado na sessão de usuário final uma interface bastante simples é exibida, apresentando uma barra de pesquisa que pode ser utilizada de filtro, como especificado no RF24, Filtragem de Barbearias, e pode filtrar pelo nome da barbearia, ou então pela cidade

ou Estado de localização da mesma. E também apresenta uma listagem de todas as barbearias cadastradas em card separados e intuitivos contendo apenas as informações de Logo, Nome, Cidade e Estado de cada empresa cadastrada, sendo a representação do RF25, Consulta de Barbearias, conforme mostra a Figura 19.

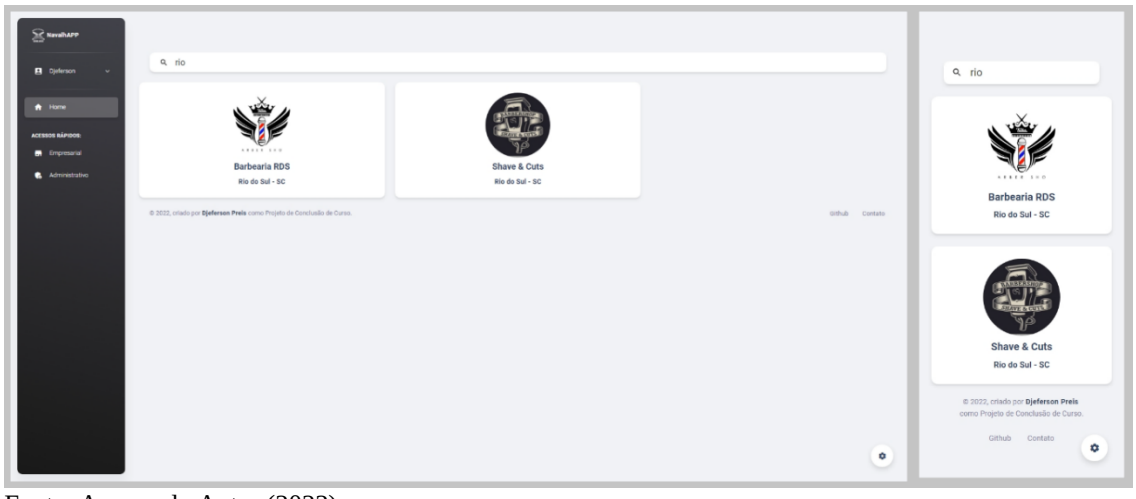
Figura 19 – Protótipo interface de Consulta de Barbearias



Fonte: Acervo do Autor (2022).

As opções de filtro são bastante intuitivas e realizam a filtragem em tempo real conforme o conteúdo informado no campo de busca, sem a necessidade de nenhuma ação adicional além de escrever. Representando este RF24 conforme indicado pela Figura 20.

Figura 20 – Protótipo interface de listagem de barbearias filtradas

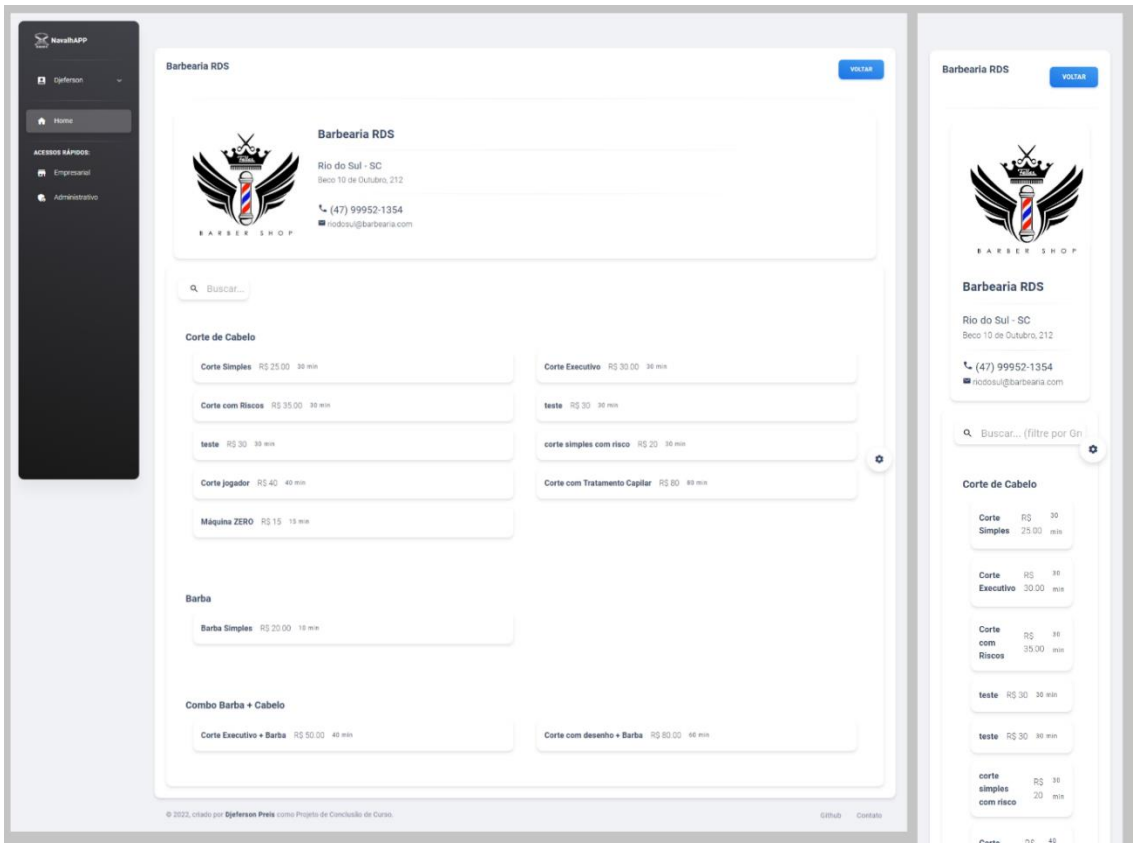


Fonte: Acervo do Autor (2022).

Após selecionar uma das opções de empresas da interface de listagem o respectivo detalhamento é apresentado e separado em sessões diferentes, sendo a primeira sessão as informações gerais da barbearia, como a logo, nome, endereço completo e contato da empresa, representando o RF26, Detalhamento de Barbearia.

Na mesma interface de detalhamento, mas nas demais sessões, são representadas as informações determinadas pelo RF27, Consulta de Grupos e Serviços. Onde é apresentado, logo na segunda sessão da interface, a listagem de todos os grupos e respectivos serviços vinculados aos grupos em cards simplificados contendo apenas a descrição, valor e tempo estimado de cada serviço, conforme indicado pelo protótipo apresentado na Figura 21.

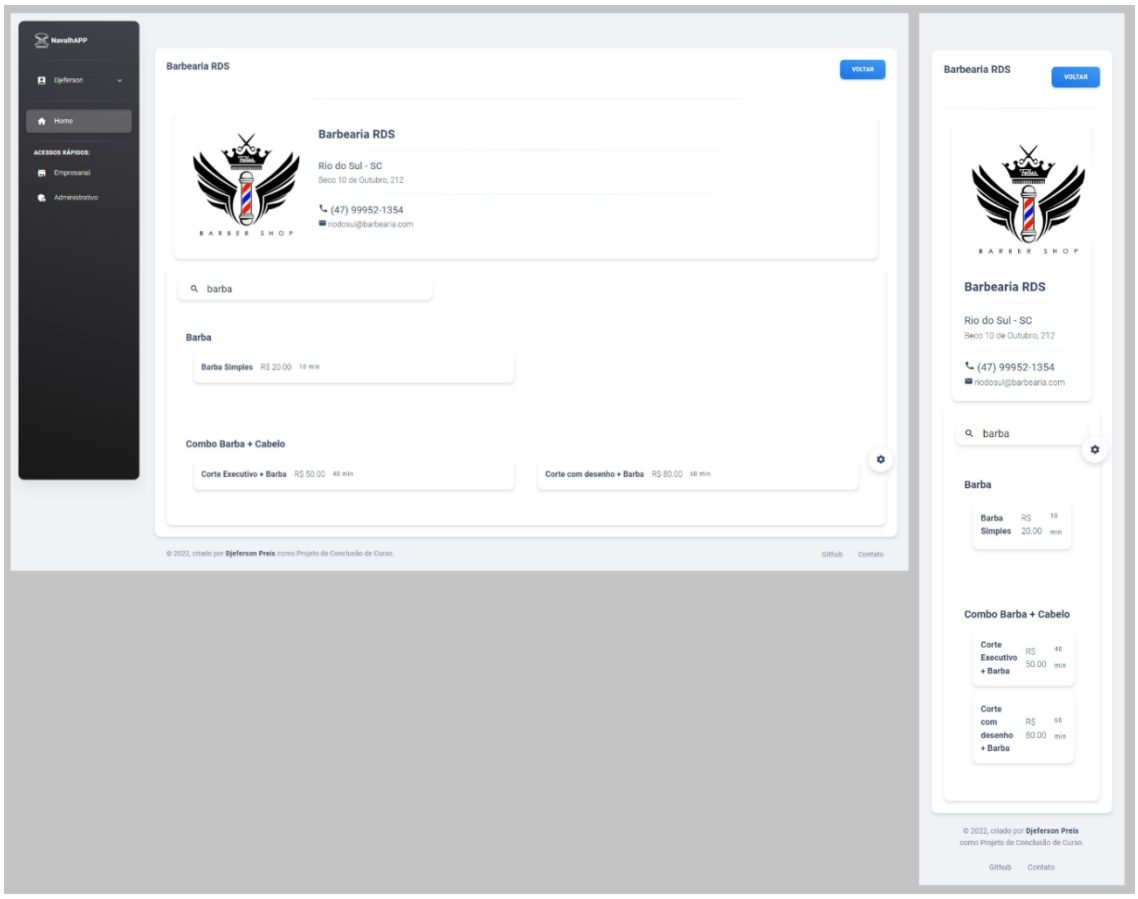
Figura 21 – Protótipo interface de Detalhamento de Barbearia



Fonte: Acervo do Autor (2022).

O mesmo requisito RF27 também determina a necessidade de uma opção de filtragem dos grupos e dos serviços, que ocorre pela apresentação do campo de busca no topo da sessão onde, tal como na interface de listagem de empresas, temos um processo de filtragem em tempo real das opções conforme o conteúdo do campo é informado, sem necessidade de nenhuma ação adicional. Para representa a ação temos a pesquisa pela descrição “barba” conforme indicado pela Figura 22.

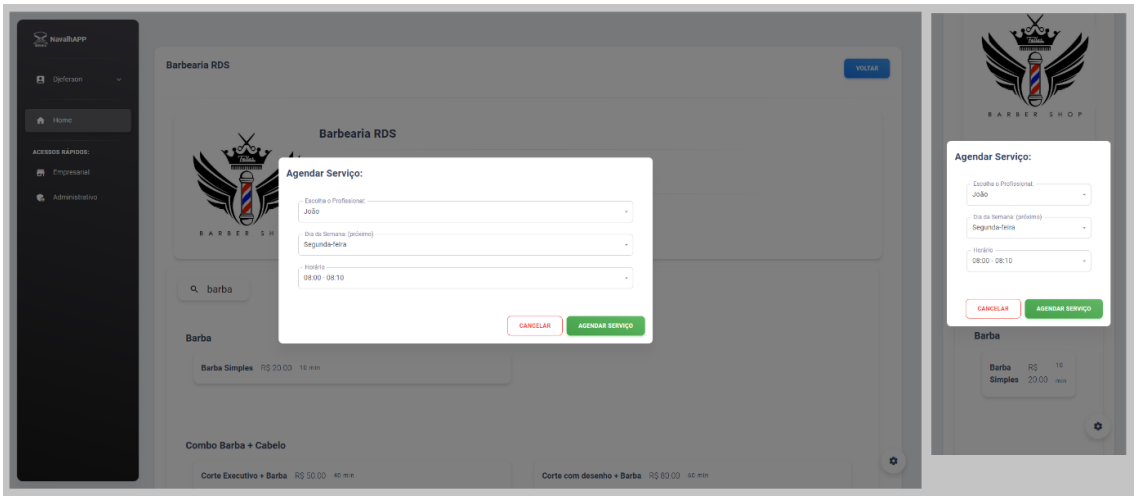
Figura 22 – Protótipo interface de Detalhamento de Barbearia com filtragem de serviço



Fonte: Acervo do Autor (2022).

Para que um usuário final possa realizar um agendamento basta selecionar o serviço desejado na listagem de serviços da interface de detalhamento da empresa, ao realizar esta ação uma sub-interface é apresentada contendo campos para seleção do Profissional (baseado nas agendas), dia da semana e período. As opções do período são determinadas a partir de alguns critérios principais, sendo eles a disponibilidade do Profissional no dia da semana em questão, então os períodos exibidos apresentam uma relação entre o horário inicial e final do profissional naquele dia, e também uma relação com o tempo estimado para execução do serviço, sendo assim, ao selecionar um pedido onde é estimado a necessidade 30 minutos para execução, os pedidos apresentados possuem uma diferença de 30 minutos, concomitantemente, ao selecionar serviços com estimativas diferentes, há alteração das opções. A representação desta sub-interface apresentada na Figura 23 representa o RF28, solicitar agendamento.

Figura 23 – Protótipo sub-interface para agendamento de Serviço.



Fonte: Acervo do Autor (2022).

Ao selecionar o serviço, configurar o profissional, dia da semana e período desejado, basta pressionar “Agendar serviço” que o pedido será direcionado à empresa em questão e passará a ser exibido na listagem de pedidos.

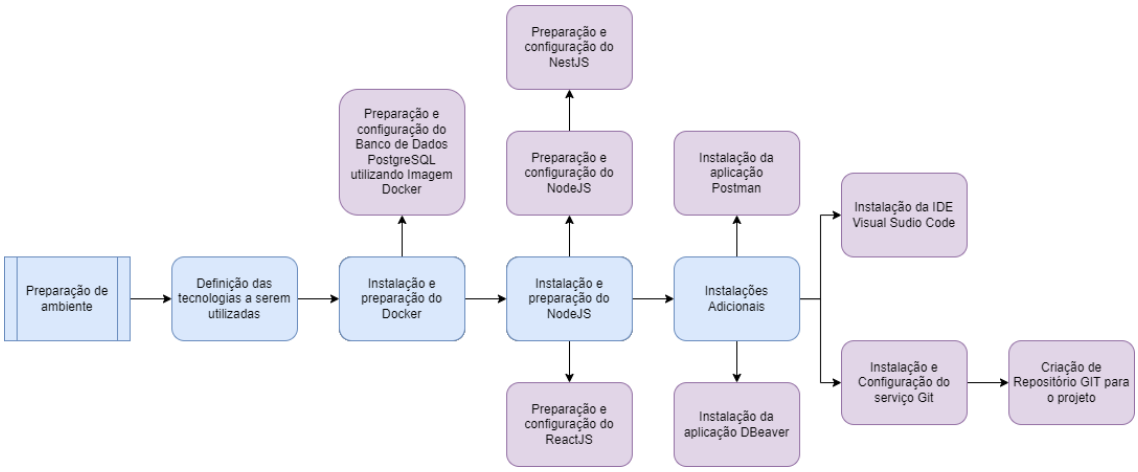
4.4 IMPLEMENTAÇÃO

Nesta sessão são apresentadas as tecnologias e ferramentas utilizadas durante o desenvolvimento do projeto, desde a aplicação *frontend* quanto *backend* e também será abordado a utilização e funcionamento do sistema.

4.4.1 Preparação de ambiente

Para a preparação do ambiente foi definido inicialmente quais seriam as tecnologias a serem utilizadas para o desenvolvimento e em seguida, foi realizado o processo de preparação conforme diagrama apresentado na Figura 24 apresentada abaixo.

Figura 24 – Diagrama de preparação do ambiente



Fonte: Acervo do Autor (2022).

Ambas as aplicações *frontend* e *backend* foram desenvolvidas utilizando o mesmo ambiente de desenvolvimento, composto por uma máquina com Sistema Operacional Windows 10 convencional e as aplicações instaladas.

Para comportar o Banco de Dados da aplicação, utilizou-se um container Docker com uma imagem virtualizada de um SGBD PostgreSQL afim de mitigar dificuldades e transtornos com outras versões do banco ou com outras conexões locais destinadas a bancos diferentes do projeto. O Docker vem sendo muito utilizado pois permite que uma mesma aplicação rode em qualquer Sistema Operacional sem haver necessidades de configurações adicionais como a instalação de extensões ou desabilitação de serviços.

Destinado do desenvolvimento, partiu-se para a instalação do NodeJS, onde optou-se pela versão recomendada mais atual no momento inicial do projeto, a versão v14.17.5. Em seguida já se deu início à instalação do Framework NestJS em sua versão recomendada no mesmo momento, a versão 9.0.0.

Após a preparação do ambiente iniciou-se as preparações adicionais, como a instalação da IDE Visual Studio Code, o editor de código fonte mais popular da atualidade e que permite a instalação de diversas extensões com a finalidade de facilitar o desenvolvimento independentemente do tipo de projeto ou linguagem utilizada.

Dado a necessidade de avaliação do projeto *backend* durante o desenvolvimento foi instalado a aplicação Postman, que atualmente é uma das melhores ferramentas para auxiliar nos testes de APIs e que atualmente disponibiliza funcionalidades que facilitam a documentação textual das rotas da API testada e até mesmo baterias de testes configuráveis.

Para avaliação e suporte também no desenvolvimento do projeto *backend* foi instalado a ferramenta DBeaver para conexão manual no Banco de Dados com a finalidade de avaliar a situação das tabelas e relações e facilitar a idealização de relações e retornos de dados da API.

Como gestor de toda alteração e servindo de guia para o desenvolvimento do início ao fim do projeto, utilizou-se a ferramenta Git, um sistema de controle de versão de código que permite visualizar todas as alterações feitas no código ao longo do tempo, avaliar códigos escritos anteriormente e recoloca-los no projeto se necessário e principalmente para manter uma boa segurança do projeto. Sobre o Git, utilizando a plataforma Github foi criado um repositório para hospedar todos os códigos da aplicação

4.4.2 Tecnologias e ferramentas utilizadas no desenvolvimento do *backend*

Para o desenvolvimento da aplicação *backend* definiu-se a utilização de uma API REST, utilizando como base a biblioteca NodeJS, e, afim de facilitar o processo de desenvolvimento o Framework NestJS que integra várias bibliotecas como o Express que resulta em um código JavaScript executado nativamente como uma aplicação do tipo Web Service, sem uma interface visual, mas para comunicação entre o *frontend* e o Banco de Dados. A Figura 25 demonstra a criação de uma API básica, e partindo dela nota-se a facilidade obtida pela utilização do framework no desenvolvimento.

Figura 25 – Criação de API com NestJS

```
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';

async function bootstrap() {
  const app = await NestFactory.create(AppModule);
  await app.listen(3000);
}
bootstrap();
```

Fonte: Acervo do Autor (2022).

No desenvolvimento da aplicação *backend* foi utilizado a biblioteca do *TypeScript* que força a tipagem de dados e consequentemente facilita a validação de parâmetros e a manutenção futura, além de gerar uma maior integridade dos dados que são enviados ao banco de dados. E,

se tratando de banco de dados, optou-se pela utilização da biblioteca TypeORM, que gera objetos espelhados no código em relação às tabelas existentes no Banco de Dados, o que permite com que toda a definição de dados seja gerada pela aplicação e automaticamente repassada para o banco de dados. A Figura 26 apresenta uma simplificação da utilização da biblioteca na implementação de uma tabela para os usuários administradores do sistema.

Figura 26 – Objeto TypeORM para criação de Usuário Administrador

```
import {
    BaseEntity,
    Entity,
    Unique,
    PrimaryGeneratedColumn,
    Column,
    CreateDateColumn,
    UpdateDateColumn,
} from 'typeorm';
import * as bcrypt from 'bcrypt';

...
@Entity()
@Unique(['email'])
export class UserAdmin extends BaseEntity {
    @PrimaryGeneratedColumn('uuid')
    id: string;

    @Column({ nullable: false, type: 'varchar', length: 200 })
    email: string;

    @Column({ nullable: false, type: 'varchar', length: 200 })
    name: string;

    @Column({ nullable: false, default: true })
    status: boolean;

    @Column({ nullable: false })
    password: string;

    @Column({ nullable: false })
    salt: string;

    @Column({ nullable: true, type: 'varchar', length: 64 })
    confirmationToken: string;

    @CreateDateColumn()
    createdAt: Date;

    @UpdateDateColumn()
    updatedAt: Date;

    async checkPassword(password: string): Promise<boolean> {
        const hash = await bcrypt.hash(password, this.salt);
        return hash === this.password;
    }
}
```

Fonte: Acervo do Autor (2022)

Com o auxílio destas bibliotecas o processo de desenvolvimento foi facilitado, e sempre que uma nova requisição da API era desenvolvida, utilizou-se da aplicação Postman para avaliar o funcionamento da requisição, realizando o envio manual das requisições e avaliando se o conteúdo retornado era realmente o conteúdo desejado, ou então, para outros tipos de comandos, como o de criação, alteração ou remoção de objetos, avaliando se estes realmente recebiam as ações esperadas após comandado.

4.4.3 Tecnologias e ferramentas utilizadas no desenvolvimento do *frontend*

Na criação do *frontend* foi utilizado a linguagem de programação Javascript em conjunto com a biblioteca ReactJS. A biblioteca em questão é um *framework* que permite a componentização de elementos HTML, CSS e JS e reutilização destes de forma extremamente simplificada e variada. Podemos ter componentes simples que apenas escrevem um conteúdo em tela e que podem receber parâmetros, tal como apresentado na Figura 27.

Figura 27 – Exemplo de Componente ReactJS

```
class ComponenteExemplo extends React.Component {
    render() {
        return <h1>Olá {this.props.nome}</h1>
    }
}
```

Fonte: Acervo do Autor (2022).

A partir da criação de componentes podemos realizar a importação e reutilizá-los quantas vezes forem necessárias e de diferentes formas, como apresentado na Figura 28, com um exemplo de chamada do Componente “CompoenenteExemplo” repassando diferentes parâmetros em cada chamada e resultando na apresentação do mesmo componente três vezes em tela, mas com os nomes diferentes em cada apresentação.

Utilizando o ReactJS é possível criar interfaces bem padronizadas em toda a aplicação, gerando um melhor resultado visual. Mas ainda houve a inclusão do *framework* Material UI, uma biblioteca que disponibiliza diversos componentes previamente preparados e prontos para serem utilizados dentro dos projetos, já possuindo uma excelente responsividade em relação às diferentes telas que podem vir a acessar a aplicação, como a diferença entre telas de dispositivos *Desktop*, *Mobile* e Tablets.

Figura 28 – Exemplo de uso de componentes no ReactJS

```
function App() {
    return (
        <div>
            <ComponenteExemplo nome="João" />
            <ComponenteExemplo nome="Pedro" />
            <ComponenteExemplo nome="Augusto" />
        </div>
    )
}
```

Fonte: Acervo do Autor (2022).

O Material UI, ou MUI em sua versão mais recente, possui quase todos os componentes fundamentais para uma aplicação, como botões, campos de formulários, caixas de divisão, caixa de imagem entre diversos outros, e todos os componentes são passíveis de alteração dentro de cada projeto, o que torna sua utilização ainda melhor. Como exemplo, a figura 29 apresenta um componente de Alerta que utiliza como base componentes previamente programados do MUI e os altera um pouco para a realidade do projeto.

Figura 29 – Exemplo componente Alerta

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

// @mui material components
import MuiAlert from '@mui/material/Alert';
import Typography from '@mui/material/Typography';

/**
 * Alert component
 * @param props.type => "success" / "error" / "warning" / "info"
 * @param props.title => Alert title
 * @param props.message => Body message
 */
export default function Alerta(props) {

    return (
        <Fragment>
            <MuiAlert className="mb-12" severity={props.type}>
                <div className="align-items-center align-content-center">
                    <Typography variant="h6" color="text.danger" >
                        {props.title}
                    </Typography>
                    <br/>
                    <Typography variant="body2" color="text.danger">
                        {props.message}
                    </Typography>
                </div>
            </MuiAlert>
            <br />
        </Fragment >
    );
}
```

Fonte: Acervo do Autor (2022).

Em conjunto com o MUI utilizou-se a biblioteca Bootstrap, em sua versão 4.5.2, que também disponibiliza diversos layouts já programados e com grande responsividade com a finalidade de manter uma boa padronização de layout entre as interfaces independentemente do tipo de dispositivo que está utilizando a aplicação. O Bootstrap disponibiliza diversas classes que podem ser utilizadas nos componentes do HTML ou até mesmo em conjunto com os componentes do ReactJS e do Material UI e que alteram a forma de sua apresentação em tela, alterando o CSS e até adiciona eventos e iterações com o JS. Como apresentado no exemplo da Figura 30, podemos adicionar classes como a "*align-items-center*" para posicionar blocos de conteúdos no centro do respectivo objeto e utilizar a classe "*align-content-center*" para alinhar o conteúdo textual no centro do respectivo objeto ou de seu "filho" direto.

Figura 30 – Exemplo de uso do Bootstrap

```
<div className="align-items-center align-content-center">
    <Typography variant="h6" color="text.danger" > {props.title} </Typography>
    <br/>
    <Typography variant="body2" color="text.danger"> {props.message} </Typography>
</div>
```

Fonte: Acervo do Autor (2022).

Além da apresentação das interfaces visuais o *frontend* precisa realizar sua comunicação com a API REST que desenvolvemos por meio de requisições, e para realizar esta comunicação utilizou-se a biblioteca Axios, que permite realizar qualquer requisição REST como cliente a um servidor.

A utilização do Axios foi dividida em duas partes, a primeira foi a configuração da comunicação, que se resume a configurar os dados de comunicação da API REST em um arquivo do projeto *frontend* que possui a importação do Axios, a criação de um elemento com a URL da API e sua respectiva exportação. Como complemento desta configuração foi acrescentado um "*interceptor*" que executa uma ação adicional a cada requisição enviada do cliente ao servidor e que é utilizado para adicionar as informações de autenticação do usuário, se existirem. A Figura 31 apresenta um exemplo de arquivo de configuração da sessão de administrador, validando e enviando o token de autorização de um Administrador do sistema para a API.

Figura 31 – Arquivo de Configuração de API com Axios

```
import axios from "axios";
import { TOKEN_KEY } from 'utils/admin/useToken';

const api = axios.create({
    baseURL: process.env.REACT_APP_API_URL, // http://localhost:3333
});

api.interceptors.request.use(async config => {
    let token = localStorage.getItem(TOKEN_KEY);
    if (token) {
        config.headers.Authorization = `Bearer ${token}`;
    }

    config.headers['Content-Type'] = 'application/json';
    config.headers['Access-Control-Allow-Origin'] = '*';

    return config;
});

export default api;
```

Fonte: Acervo do Autor (2022).

Com a integração entre o ReactJS, MUI, Bootstrap e Axios foi possível o desenvolvimento de interfaces bastante responsivas, altamente padronizadas em toda a aplicação e, mesmo sem muito conhecimento em estilização CSS, foi possível o desenvolvimento de interfaces visualmente atrativas para o projeto.

4.5 VALIDAÇÃO

Para validação do trabalho foi realizado entrevista com um gestor e dono de uma barbearia regional onde os protótipos foram apresentados juntamente com a ideia do projeto, a partir desta avaliação foi possível identificar algumas necessidades primordiais e boas sugestões para evolução do trabalho que constam como trabalhos futuros, sobre a possibilidade de dar sequência no desenvolvimento e publicação deste projeto no mercado.

Solicitando a análise do projeto em consideração às necessidades do gestor em seu dia-a-dia foi salientado a necessidade de uma opção para emitir as ordens de agendamento diretamente pelo sistema de gerenciamento da empresa, com a possibilidade de funcionários configurarem pedidos feitos presencialmente ou por outros meios por seus clientes que normalmente aparecem “de surpresa” ou conversam por telefone ou mensagem.

Em aspecto geral o produto foi considerável agradável, com um excelente layout e intuitivo, para uma primeira versão (protótipo) a aplicação já, se aplicada em sua empresa, resolveria alguns gargalos no processo de agendamento e de controle das filas dos clientes que, segundo o entrevistado, em momentos de pico (comumente sextas-feiras, sábados e domingos) acaba tendo a necessidade de chamar um auxiliar que trabalha como barbeiro apenas para realizar o gerenciamento das agendas e o atendimento aos clientes até que chegue sua vez ou que um horário fique disponível.

Mesmo sendo uma aplicação em estado inicial de prototipagem, já possui uma pequena variedade de soluções aos problemas contantes apresentados pela empresa, sobre a qual atualmente possui dois barbeiros e o contato se dá apenas pelo telefone principal, o que gera uma pequena perda de constância no resultado diário quando há muitas solicitações e agendamentos.

5. CONCLUSÃO

O desenvolvimento do presente protótipo permitiu demonstrar a possibilidade de um sistema para o gerenciamento de reservas e serviços no mercado de barbearias desenvolvido de forma online e que pode ser acessado facilmente por qualquer cliente por meio de um dispositivo eletrônico sendo smartphone ou computador conectado à internet.

O mesmo projeto apresenta sessões distintas com todas as principais necessidades apontadas para clientes e para os próprios gestores das barbearias, tornando possível e simples a busca e agendamento de um serviço por parte dos clientes enquanto aos gestores e funcionários das barbearias temos um serviço centralizado de agenda onde podem usufruir tanto para avaliar melhorias para seus negócios quanto para os próprios benefícios de organização de horários.

Durante o processo de levantamento de requisitos gerou-se um maior entendimento das necessidades reais de uma empresa em um sistema online, quais os processos exercidos pelas barbearias e que muitas vezes gera dificuldades aos respectivos gestores por terem que ficar dando atenção às suas agendas, “filas” de clientes e seu próprio horário de funcionamento enquanto executa seu serviço.

O levantamento de requisitos apontou principalmente que, mesmo sendo um serviço bastante físico, pode muito bem receber os benefícios deste mundo eletrônico e que está cada vez mais dependente da internet para facilitar e agilizar os negócios no ramo empresarial. Apontando que mesmo empresas de pequeno porte, com até mesmo um único funcionário podem receber benefícios, como um simples agendamento por meio de uma aplicação web por valores que podem ser bastante acessíveis tendo em vista que o mesmo processo executado por elas é executado (em diferentes dimensões) por empresas de maior porte.

Para a definição das ferramentas e tecnologias a serem utilizadas no desenvolvimento do protótipo do projeto levou-se em consideração principalmente as tecnologias atualmente em alta no mercado, justamente para aumentar a competitividade e ao mesmo tempo, utilizando tecnologias e padrões bem estabelecidos, para possibilitar a continuidade do desenvolvimento e a continuação com desenvolvimento de novas interfaces e processos no futuro.

Além das tecnologias foram utilizados frameworks para as interfaces visuais com a finalidade de gerar um padrão geral estabelecido para todas as interfaces do projeto, o que, além de gerar telas mais harmônicas, facilitou o processo de adequação destas mesmas interfaces de maneira responsiva entre os dispositivos de diferentes telas tais como as diferenças entre as versões para *desktop* e *mobile*.

Durante o levantamento de requisitos, o estudo de ferramentas semelhantes já disponíveis no mercado e a definição das tecnologias, analisamos os processos já existentes nas barbearias afim de melhorá-los com o uso da aplicação, gerando um melhor formato de oferta de serviço aos clientes.

Durante esta análise identificou-se que a maior parte das comunicações para agendamento de serviços é feita por meio de *smartphones*, sabendo disso definiu-se então a utilização de interfaces altamente responsivas em dispositivos *mobile*, sem perder a responsividade quando em ambientes *desktop*, principalmente para as sessões empresariais e administrativas do sistema.

Os protótipos se baseiam principalmente nos requisitos e nas necessidades apontadas inicialmente pelas barbearias, o desenvolvimento iniciou-se pela parte administrativa onde foi possível o cadastro dos registros de barbearias e dos demais administradores do sistema NavalhAPP, em seguida pela parte empresarial focada em um MVP a ser utilizado pelas empresas para configuração e gerenciamento de seus negócios, tendo nesta sessão a possibilidade do cadastro dos serviços prestados, das agendas e horários de disponibilidade.

Em seguida, focou-se no desenvolvimento da sessão destinada aos clientes, os usuários finais do protótipo, onde ocorre o cadastro dos usuários feitos por eles mesmos, a apresentação dos negócios disponibilizados na aplicação, e dos serviços disponibilizados por estes negócios. Havendo também a possibilidade de gerar um agendamento em um horário disponível para que o cliente possa chegar no horário marcado e o serviço ser executado, sem a necessidade de aguardar em “filas” com ordem de chegada.

A criação destes protótipos teve como objetivo a sua intuitividade e responsividade, o que também leva ao objetivo de permitir que usuários leigos consigam entender o funcionamento e usufruir da aplicação conforme houver demanda, possibilitando assim que qualquer cliente convencional do estabelecimento que antes utilizava de outros meios para solicitar os serviços da barbearia possa, facilmente, ser transferido para a nova aplicação, tendo uma maior qualidade de seu próprio dia-a-dia.

Com a finalidade de validar o resultado deste trabalho foi realizado uma entrevista com um usuário, dono de barbearia e que atualmente utiliza apenas formas verbais físicas e por meio de ligações ou mensagens via WhatsApp para emitir o agendamento dos seus clientes. Este mesmo usuário já se depara com muitas dificuldades em sua agenda o que acaba requerendo de ajuda de familiares para o atendimento dos clientes durante horários de pico como sextas-feiras, sábados e domingos.

Esta validação permitiu uma perspectiva direcionada totalmente ao negócio e a partir de seu resultado foi identificado tanto pontos positivos quanto negativos do trabalho, sendo estes apresentados também como sugestão para trabalhos futuros e que permitirão a evolução momentânea e constante da ferramenta no caso de seguir para a produção, para uma publicação da mesma no mercado e não mais como um protótipo.

5.1 SUGESTÃO DE TRABALHO FUTURO

A ideia geral do presente trabalho é tornar a aplicação NavalhAPP uma grande ferramenta para auxiliar os gestores e empregados de barbearias em todos os processos que podem ser feitos sem a necessidade de interações físicas. Partindo disso há diversas funcionalidades e evoluções que este projeto pode receber para estar cada vez mais adequado ao objetivo de como gerenciar as agendas, vendas, prestações de serviços e auxiliar na fidelização de clientes de barbearias de pequeno, médio e grande porte.

5.1.1 Sugestões mais impactantes

As sugestões de curto prazo fazem referência às necessidades maiores em relação ao funcionamento do protótipo, com a finalidade de trazer resultados rápidos às necessidades das empresas relacionadas.

Uma das principais sugestões é a implementação de interfaces para alteração das informações do usuário final, juntamente com formulários para preencher as informações adicionais do cliente, como endereço, telefone, preferências e outras informações que poderão ser utilizadas tanto pelas lojas que receberem os agendamentos quanto para evolução da sessão destinada ao usuário final, de forma a determinar previamente parâmetros de filtragem ou destaques na seleção de empresas e serviços.

Foi sugerido também o desenvolvimento do RF30 destacado como opcional, e que permitirá a visualização do histórico de agendamentos do cliente, o que facilitará a parte do cliente possibilitando um acesso fácil ao contato com as empresas para eventuais cancelamentos ou reagendamentos.

5.1.2 Sugestões relevantes

Em relação aos reagendamentos, também foi sugerido, para evolução da sessão empresarial, opções para reagendamento e para alteração da listagem de serviços, possibilitando adicionar ou remover serviços antes de efetivar a conclusão do pedido.

Para facilitar a interação das gestões e profissionais das barbearias com o sistema, é relevante a implementação de uma nova interface de listagem de pedidos no formato de agenda, onde será possibilitado a visualização das informações por filtragem de dia, semana e profissional responsável no formato de visualização por períodos diários (de hora-em-hora).

Afim de facilitar a intercomunicação entre loja e cliente é cogitado o desenvolvimento de um formato de comunicação por meio de mensageria entre ambos, onde, para uma rápida solução seria possível a implementação de uma opção para contato via WhatsApp.

5.1.3 Sugestões opcionais de evolução

Entre as principais sugestões de evolução do projeto foi analisado a possibilidade e necessidade de transformação da aplicação em um *Progressive Web App* (PWA), que se resume a transformar uma aplicação web em uma instância de aplicativo instalado nos dispositivos permitindo a exibição de notificações e outras ações mais específicas sem perder a respectiva responsividade e clareza da aplicação Web.

Nos dias atuais, uma das funcionalidades mais comuns em sistemas de empresas de médio e grande porte são as rotinas de inteligência, e isso não pode ficar de fora de uma das evoluções deste projeto. Sugere-se que futuramente seja implementado rotinas de análise do histórico da barbearia em relação a suas vendas, serviços e horários e, com base nisso, além de permitir ao gestor uma melhor tomada de decisão em suas ações, será possível sugerir melhorias e automatização de processos da empresa.

Após conhecer várias barbearias é perceptível que cada uma possua características distintas, e levando isso em consideração têm-se a sugestão de implementação de rotinas diferenciadas e layouts alternativos para as empresas, podendo trazer a venda de produtos além da prestação de serviços, a apresentação de informações adicionais de vários contatos e até de múltiplos endereços e por fim, até mesmo o gerenciamento de filiais de uma mesma barbearia em diferentes localidades, com a possibilidade de gerar relatórios exclusivos de cada empresa ou de todas a partir da matriz.

REFERÊNCIAS

AGÊNCIA BRASIL. **Empresas de beleza buscam se reinventar para sobreviverem à pandemia**. 2020. Disponível em: <https://economia.uol.com.br/noticias/redacao/2020/07/04/covid-19-cansa-a-beleza-crise-no-setor-forca-saloes-a-se-reinventarem.htm>. Acesso em: 15 out. 2022.

ALVES, William P. **Desenvolvimento e design de sites**. São Paulo: Editora Saraiva, 2014. *Ebook.*

ALVES, William P. **HTML & CSS:** aprenda como construir páginas web**.** São Paulo: Editora Saraiva, 2021. *Ebook.*

ALVES, William P. **Java para web:** desenvolvimento de aplicações. São Paulo: Editora Saraiva, 2015. *Ebook.*

ALVES, William P. **Projetos de sistemas web conceitos, estruturas, criação de banco de dados e ferramentas de desenvolvimento**. São Paulo: Editora Saraiva, 2015. *Ebook.*

BARBOZA, Fabrício Felipe M.; FREITAS, Pedro Henrique C. **Modelagem e desenvolvimento de banco de dados**. Porto Alegre: Grupo A, 2018. *Ebook.*

CARDOSO, Maycon Moreira. **Barbearias, o início e sua história**. 2020. Disponível em: <https://seducintec.com.br/noticias/barbearias-o-inicio-e-sua-historia/>. Acesso em: 15 out. 2022.

CARVALHO, Vinícius. **PostgreSQL**: banco de dados para aplicações web modernas. São Paulo: Casa do Código, 2017. *Ebook.*

CHACON, Scott; STRAUB, Bem. **Pro Git**. 2. ed. Mountain View: Apress, 2022. *Ebook.*

EXPRESS. **Express:** framework web rápido, flexível e minimalista para Node.js. Disponível em: <https://expressjs.com/pt-br/>. Acesso em: 25 jul. 2022.

FLANAGAN; DAVID. **Javascript**. Porto Alegre: Grupo A, 2014. *Ebook.*

FREITAS, Pedro Henrique C. et al. **Programação *back end* III**. Porto Alegre: Grupo A, 2021. *Ebook.*

IVANOV, Maksim; BESPOYASOV, Alex. ***Fullstack React with TypeScript****: learn pro patterns for hooks, testing, redux, SSR, and GraphQL*. [s.l.], Newline, 2020. *Ebook.*

LIM, Greg. ***Beginning Node.js, express & MongoDB development***. 2019. *Ebook.*

MATOS, Ecivaldo; ZABOT, Diego. **Aplicativos com bootstrap e angular:** como desenvolver apps responsivos. São Paulo: Editora Saraiva, 2020. *Ebook.*

MDN. **Glossário**. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Glossary>. Acesso em: 27 mai. 2022.

MDN. **Métodos de requisição HTTP**. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods>. Acesso em: 25 mai. 2022.

MONTEIRO. et al. **DevOps**. Porto Alegre: Sagah, 2021. *Ebook.*

MUI. **Material UI** - Overview. Disponível em: <https://mui.com/pt/material-ui/getting-started/overview/>. Acesso em 20 mai. 2022.

NPM. ***About npm***. Disponível em: <https://www.npmjs.com/about>. Acesso em 25 jun. 2022.

OLIVEIRA, Cláudio Luís V.; ZANETTI, Humberto Augusto P. **Javascript descomplicado:** programação para web, IOT e dispositivos móveis. São Paulo: Editora Saraiva, 2020. *Ebook.*

OLIVEIRA, Cláudio Luís V.; ZANETTI, Humberto Augusto P. **Node.js:** programe de forma rápida e prática. São Paulo: Editora Saraiva, 2021. *Ebook.*

PEREIRA, Caio Ribeiro. **Node.js**: aplicações web real-time com node.js. São Paulo: Casa do Código, 2014. *Ebook.*

PEYROTT, Sebastián. ***The JWT handbook***. Seattle: Auth0, 2018. Disponível em: <https://assets.ctfassets.net/2ntc334xpx65/o5J4X472PQUI4ai6cAcqg/13a2611de03b2c8edbd09c3ca14ae86b/jwt-handbook-v0_14_1.pdf>. Acesso em: 29 abr. 2022.

PICHETTI, Roni F.; VIDA, Edinilson da S.; CORTES, Vanessa Stangherlin Machado P. **Banco de dados**. Porto Alegre: Grupo A, 2021. *Ebook.*

PONTES, Guilherme. ***Progressive web apps***: construa aplicações progressivas com react. São Paulo: Casa do Código, 2018. *Ebook.*

POSTGRESQL. ***What is PostgreSQL?***. Disponível em: <https://www.postgresql.org/docs/current/intro-whatis.html>. Acesso em: 13 mai. 2022.

REACT. **Uma biblioteca javascript para criar interfaces de usuário**. Disponível em: <https://pt-br.reactjs.org/>. Acesso em 27 mai. 2022.

RED HAT. **API REST**. 2020. Disponível em: https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api. Acesso em: 27 maio 2022.

RODRIGUES, Thiago N. et al. **Integração de aplicações**. Porto Alegre: Grupo A, 2020. *Ebook.*

ROLDÁN, Carlos Santana. ***React 17****: design patterns and best practices*. 3. ed. Birmingham: Packt, 2021. *Ebook.*

SARAIVA, Maurício de O.; BARRETO, Jeanine dos S. **Desenvolvimento de sistemas com PHP**. Porto Alegre: Grupo A, 2018. *Ebook.*

SASS. ***Documentation***. Disponível em: <https://sass-lang.com/documentation>. Acesso em: 13 mai. 2022.

SHARMA, Aneeta. ***Full-stack web development with Vue.js and Node***. Birmingham: Packt, 2018. *Ebook.*

SILVA, Wellington Figueira da. **Aprendendo Docker**. São Paulo: Novatec, 2016. *Ebook.*

SIMAS, Victor L. et al. **Desenvolvimento para dispositivos móveis**. Volume 2. Porto Alegre: Grupo A, 2019. *Ebook.*

TYPEORM. ***Getting Started.*** Disponível em: <https://typeorm.io/>. Acesso em: 15 out. 2022.

TYPESCRIPT. ***TypeScript is JavaScript with syntax for types****: what is TypeScript*. Disponível em: <https://www.typescriptlang.org/>. Acesso em: 25 jun. 2022.

VALENTE, Jonas. **Pesquisa revela aumento de pedidos de comida por app durante pandemia**: alternativa permite economizar tempo, afirmam entrevistados. 2021. Disponível em: <https://agenciabrasil.ebc.com.br/geral/noticia/2021-12/pesquisa-revela-aumento-de-pedidos-de-comida-por-app-durante-pandemia>. Acesso em: 15 out. 2022.

WHATSAPP. **WhatsApp**. 2022. Disponível em: <https://www.whatsapp.com/>. Acesso em: 15 out. 2022.

ANEXOS