PÓS-GRADUAÇÃO EM DESENVOLVIMENTO FULL STACK

DESENVOLVIMENTO DE UM SISTEMA SERVERLESS PARA RESERVA DE SERVIÇOS

FILIPE RAMOS

Orientador: Marlon Leandro Moraes

2024

Dedico este trabalho às pessoas duramente impactadas pela tragédia climática no Rio Grande do Sul. Que encontrem força e esperança.

SUMÁRIO

1. CONTEXTUALIZAÇÃO DA PROPOSTA

O presente trabalho de conclusão de curso representa a culminação de um intenso período de estudos e práticas no campo da tecnologia da informação. O projeto aborda uma gama diversificada de conteúdos adquiridos ao longo do curso de especialização em Desenvolvimento Full Stack, consolidando conhecimentos teóricos e práticos em uma aplicação real.

Trata-se de um projeto *full stack*, conforme delineado pelo título do curso, que compreende não apenas o desenvolvimento de interfaces amigáveis para o usuário final, mas também a implementação de uma infraestrutura robusta nos bastidores. A abordagem *full stack* permite uma compreensão abrangente do ciclo de vida de um sistema de *software*, desde o *front-end* até o *back-end*, capacitando o desenvolvedor a atuar em todas as camadas da aplicação.

A motivação para este projeto surgiu da observação de um problema recorrente em datas comemorativas e momentos de lazer, nos quais as pessoas desejam sair de casa e desfrutar de refeições em restaurantes, mas encontram dificuldades para reservar mesas devido à alta demanda e à burocracia heterogênea envolvida nesse processo. Por vezes, os indivíduos anseiam por experimentar novos estabelecimentos, mas esbarram na falta de soluções práticas para realizar reservas de forma ágil e descomplicada.

Este contexto inspirou a concepção de um sistema inovador, que visa simplificar e otimizar o processo de reserva de serviços, proporcionando aos usuários uma experiência fluida e conveniente. Através da integração de tecnologias modernas e da aplicação de princípios de desenvolvimento ágil, pretendeu-se desenvolver uma plataforma que atenda às necessidades dos usuários, promovendo a interação entre consumidores e prestadores de serviços de forma eficiente e eficaz.

Compreendendo a diversidade de preferências e a busca por experiências inovadoras, o sistema incorpora uma abordagem aleatória, permitindo que os usuários descubram novos locais e vivenciem experiências surpreendentes. Por meio de algoritmos inteligentes, o sistema seleciona automaticamente um serviço com base nas especificações do usuário — como intervalo de preço, data e quantidade de

pessoas. Essa funcionalidade amplia as possibilidades de descoberta e promove a exploração de opções alternativas, de modo a proporcionar aos usuários uma jornada dinâmica e enriquecedora.

Neste contexto, o presente trabalho tem como objetivo principal apresentar o desenvolvimento de um sistema *serverless* — sem provisionamento de servidores (VIEIRA et al., 2020) — para reserva de serviços. Os objetivos específicos são expostos no próximo capítulo.

2. OBJETIVOS DA CONSTRUÇÃO DA SOLUÇÃO

Os objetivos delineados a seguir são uma extensão direta da contextualização apresentada no capítulo anterior e refletem, assim, os propósitos estratégicos traçados para este trabalho de conclusão de curso. Com base na análise do contexto e dos desafios identificados, os objetivos estratégicos estabeleceram um guia claro para o desenvolvimento do projeto, visando ao domínio de tecnologias emergentes em desenvolvimento *full stack* e o aperfeiçoamento de competências em integração de tecnologias e princípios de desenvolvimento ágil.

Os seguintes objetivos estratégicos foram definidos:

1. dominar tecnologias emergentes em desenvolvimento *full stack*;
   1.1. programar o *back-end* da aplicação com NestJS — um *framework* progressivo em Node.js (SABO, 2020) —, garantindo segurança e eficiência;
   1.2. desenvolver interfaces amigáveis para o usuário final (*front-end*) utilizando Next.js, um *framework* que emprega a biblioteca React (THAKKAR, 2020);
   1.3. explorar arquiteturas de nuvem, de modo a compreender suas vantagens e desafios;
   1.4. implementar infraestrutura *serverless* para o sistema de reserva de serviços.
2. desenvolver competências em integração de tecnologias e princípios de desenvolvimento ágil.
   2.1. utilizar Kanban como método ágil para gerenciar o desenvolvimento do projeto de forma iterativa e incremental;
   2.2. integrar componentes da aplicação de forma coesa e eficiente, garantindo interoperabilidade entre os serviços;
   2.3. implementar mecanismos de comunicação entre o *front-end* e o *back-end*, utilizando APIs *RESTful*;

2.4. realizar entregas contínuas e ajustes conforme *feedbacks* dos testes feitos ao longo do processo de desenvolvimento.

Esses objetivos nortearam a execução do trabalho de conclusão de curso, fornecendo um roteiro claro para atingir os resultados desejados e contribuir para o avanço do conhecimento na área de tecnologia da informação.

3. ELABORAÇÃO DA JORNADA DO USUÁRIO

O desenvolvimento da jornada do usuário para a solução proposta envolve uma série de casos de uso que abrangem desde o cadastro inicial até a interação contínua com a plataforma. Este capítulo descreve detalhadamente os passos que um usuário seguirá ao utilizar o sistema, considerando as funcionalidades disponíveis no produto mínimo viável (MVP).

3.1. Requisitos funcionais e não funcionais

A relação entre as jornadas do usuário e os requisitos funcionais e não funcionais é fundamental para garantir que a solução desenvolvida atenda às necessidades e expectativas dos usuários finais de forma eficaz e eficiente. Os requisitos funcionais descrevem as funcionalidades específicas que o sistema deve oferecer para atender aos requisitos do usuário, enquanto os requisitos não funcionais definem as características mais amplas do sistema, como desempenho, segurança e usabilidade (FIGUEIREDO, 2011).

É importante ressaltar que, devido à natureza do produto mínimo viável (MVP), nem todos os requisitos funcionais e não funcionais foram implementados no escopo inicial do trabalho. No entanto, os requisitos listados abaixo foram identificados como essenciais para o funcionamento básico e a qualidade da solução proposta.

3.1.1. Requisitos funcionais

Os requisitos funcionais para a solução final do presente trabalho são detalhados a seguir.

- Cadastro de estabelecimentos: Os responsáveis pelos estabelecimentos devem poder se cadastrar no sistema, fornecendo informações como nome do estabelecimento, localização e informações de contato.

- Cadastro de serviços e ofertas de horários: Os responsáveis pelos estabelecimentos devem poder cadastrar os serviços oferecidos e os horários disponíveis, incluindo informações como tipo de serviço, horários disponíveis e capacidade de atendimento.
- Cadastro de usuários: Os usuários devem poder criar uma conta no sistema, fornecendo informações como nome, localização e informações de contato.
- Cadastro de interesses dos usuários: Os usuários devem poder cadastrar seus interesses em relação aos serviços oferecidos pelos estabelecimentos, especificando o tipo de serviço desejado e o intervalo de datas e preços desejado.
- Alterações nos cadastros: Os usuários e responsáveis pelos estabelecimentos devem poder alterar o cadastro de suas informações e dos serviços que querem ofertar.
- Exclusão de interesses: Os usuários devem poder excluir interesses que registraram anteriormente.
- Exclusão de serviços e ofertas: Os responsáveis pelos estabelecimentos devem poder excluir serviços e ofertas sem interferir nas reservas em andamento.
- Reserva automática: O sistema deve ser capaz de, com base nos interesses e localização do usuário, pré-reservar um serviço disponível, considerando a correspondência entre as especificações do usuário e os serviços oferecidos pelos estabelecimentos. A cidade do estabelecimento selecionado precisa ser a mesma informada pelo usuário em seu cadastro. Depois desse processamento, o usuário deverá poder escolher confirmar a reserva ou descartá-la.
- Notificações: O sistema deve enviar notificações para os usuários e estabelecimentos sobre as reservas.
- Histórico de reservas: Os usuários e responsáveis pelos estabelecimentos devem poder visualizar um histórico de reservas.

3.1.2. Requisitos não funcionais

Assim como os requisitos funcionais, os não funcionais são de suma importância para a solução final desejada. Eles são enumerados abaixo.

- Desempenho: A plataforma precisa garantir uma resposta rápida durante todos os processos iniciados pelos usuários, evitando atrasos significativos.
- Segurança: As informações dos usuários, estabelecimentos e reservas devem ser armazenadas de forma segura.
- Confiabilidade: O sistema deve ser robusto e confiável, minimizando possíveis falhas que possam afetar as reservas dos usuários.
- Escalabilidade: O sistema deve ser escalável para lidar com um aumento no número de usuários e estabelecimentos sem comprometer o desempenho.
- Usabilidade: A interface do usuário deve ser intuitiva e fácil de usar tanto para os responsáveis pelos estabelecimentos quanto para os usuários finais.
- Disponibilidade: O sistema deve estar disponível a maior parte do tempo, com um tempo de inatividade mínimo para manutenção.
- Manutenibilidade: O sistema deve ser desenvolvido de forma a permitir manutenções e atualizações sem interromper significativamente as operações.

3.2. Jornada do usuário

O usuário iniciará sua jornada realizando o cadastro na plataforma. Os campos obrigatórios para o cadastro são nome, CPF, e-mail, senha, data de nascimento, cidade — selecionável a partir de uma lista pré-definida — e endereço. Após preencher todos os campos requeridos, o usuário poderá criar sua conta e ter acesso aos recursos da plataforma.

Após o cadastro, o usuário poderá fazer acesso utilizando seu e-mail e senha. Ao acessar a plataforma, será direcionado para uma página principal, em que encontrará as funcionalidades disponíveis para interação.

Na página principal, o usuário terá a opção de cadastrar um novo interesse. Para isso, ele selecionará o tipo de serviço desejado a partir de opções pré-estabelecidas. Além disso, ele especificará o preço mínimo e máximo que está disposto a pagar, o intervalo de datas e horários desejados e a quantidade de pessoas envolvidas no serviço. Em seguida, o sistema encaminhará o interesse registrado para

uma fila de processamento, da qual ele tentará gerar uma correspondência entre um serviço e o interesse com base nas especificações e na localização do usuário.

O usuário também terá acesso aos interesses previamente registrados nesta página. Ele poderá visualizar os detalhes de cada interesse, como tipo de serviço, preços, data, horário e quantidade de pessoas destinadas ao serviço. Caso o interesse respectivo não tenha sido processado, os intervalos de datas e preços estarão indicados ao invés, justamente com a informação de que o pedido se encontra na fila. Se tiver sido processado e houver correspondência com um serviço ofertado na plataforma, o usuário poderá reservá-lo, e uma indicação de reservado será exposta. Caso não exista um serviço que corresponda ao interesse, o usuário também verá essa informação na listagem.

Na página principal e na descrita a seguir, um menu fixo permite ao usuário navegar entre as páginas e sair — efetuar *logout*. A partir do menu, o usuário poderá navegar para a página de alteração de cadastro, intitulada "Meus dados". Ao acessar essa página, o usuário poderá modificar todas as informações cadastradas, com exceção do CPF.

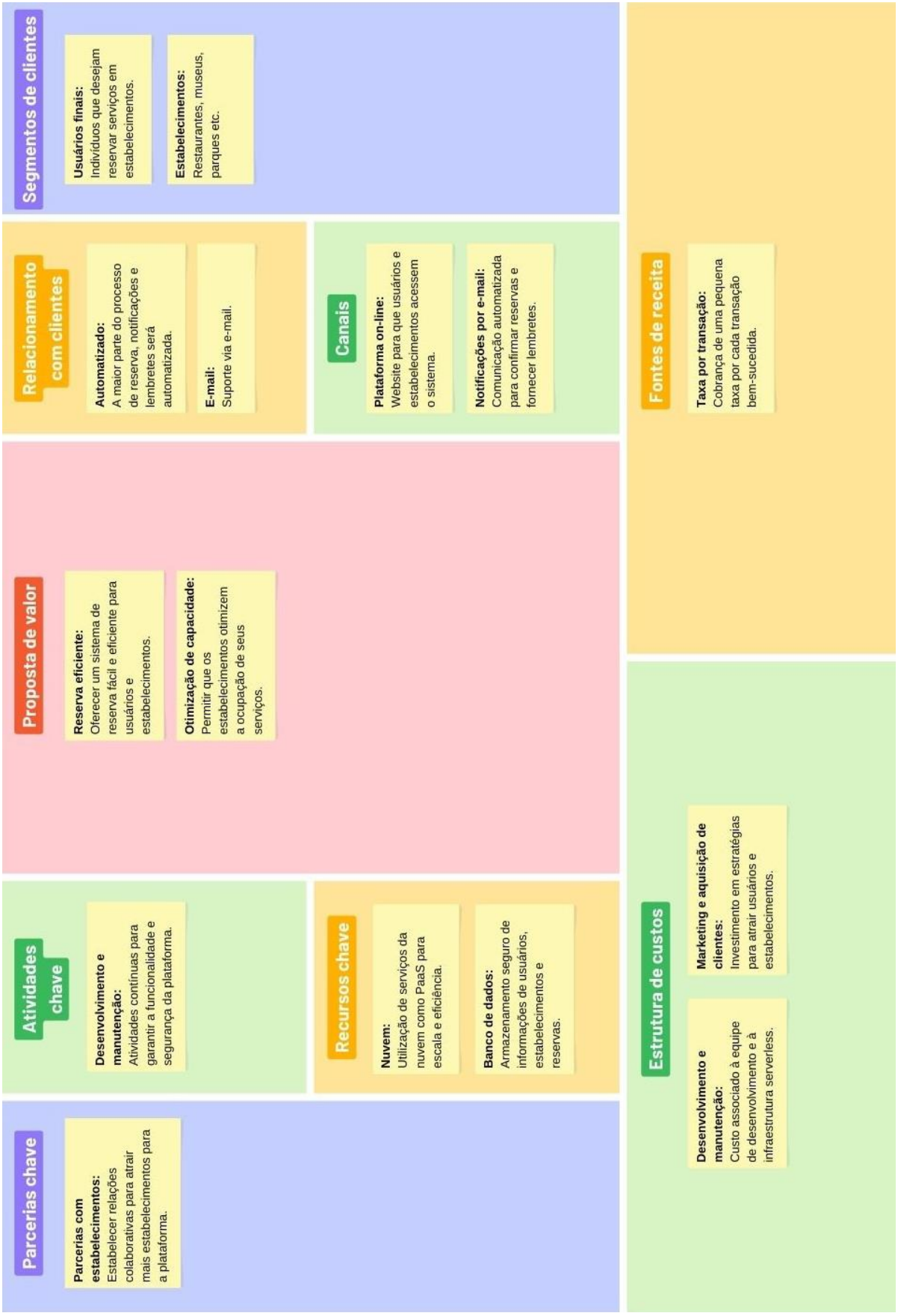
4. APELO MERCADOLÓGICO DA SOLUÇÃO

A proposta apresentada neste trabalho busca atender a uma demanda por soluções eficientes de reserva de serviços em estabelecimentos diversos. Atualmente, o mercado conta com diversas plataformas específicas e genéricas que visam facilitar o processo de reserva para os usuários finais e otimizar a ocupação dos estabelecimentos. Entre as plataformas específicas, pode-se citar o Get In (2024) — que permite encontrar reservas em restaurantes e bares — e o Sympla (2024) — no qual é possível comprar ingressos para eventos, cursos, entre outros. Já as plataformas genéricas, como o Booksy (2024), atendem a uma variedade de setores, como salões de beleza, espaços para eventos etc.

O diferencial da solução proposta neste trabalho reside na sua abordagem simplificada e inclusiva, especialmente voltada para usuários pouco exigentes que desejam conhecer novos estabelecimentos e experiências. Através da plataforma desenvolvida, os usuários podem ingressar rapidamente em uma fila geral para reservas, eliminando a necessidade de escolha prévia e oferecendo uma experiência mais dinâmica e surpreendente. Além disso, a plataforma permite que os usuários tenham controle sobre suas preferências, possibilitando filtrar as ofertas registradas. Outro diferencial desta proposta é a generalidade dos tipos de serviço a serem oferecidos, não se restringindo a reservas em restaurantes. Ingressos a parques e museus, por exemplo, também poderão ser ofertados.

O Business Model Canvas apresentado na Figura 1 ilustra os principais elementos estratégicos que sustentam a proposta de valor da solução. Destacam-se os segmentos de clientes, que englobam tanto os usuários finais quanto os estabelecimentos, e a proposta de valor centrada na reserva eficiente e na otimização de capacidade dos estabelecimentos. Os canais de distribuição, o relacionamento com clientes e as fontes de receita também são elementos fundamentais para viabilizar o modelo de negócio proposto.

Figura 1 — Business Model Canvas



Fonte: Elaborada pelo autor, 2024.

Em resumo, a solução desenvolvida não apenas busca atender às necessidades dos usuários finais e dos estabelecimentos, mas também oferece uma abordagem inovadora e acessível para a reserva de serviços, alinhada às tendências e demandas do mercado atual.

5. CICLO DE DESENVOLVIMENTO DA SOLUÇÃO

Durante a implementação do sistema, foram seguidas várias etapas para garantir um processo de desenvolvimento eficiente e organizado. O ciclo de desenvolvimento da solução envolveu as seguintes etapas:

1. definição das etapas de implementação: Inicialmente, as etapas de desenvolvimento foram definidas e gerenciadas no Trello, uma plataforma de gestão de projetos on-line que, seguindo o método Kanban, permite organizar tarefas em quadros, listas e cartões (JOHNSON, 2017). Isso possibilitou um acompanhamento claro das tarefas e um fluxo de trabalho otimizado.
2. configuração de recursos locais com Docker: Para economizar tempo e simplificar o processo de configuração do ambiente de desenvolvimento, foram utilizados recursos locais por meio de containers gerenciados pelo Docker (ANDERSON, 2015). Isso evitou a necessidade de instalar os recursos diretamente e facilitou a replicação do ambiente em diferentes máquinas.
3. inicialização dos projetos com NestJS e Next.js: Dois projetos foram inicializados para a implementação da solução — um para a interface de programação de aplicações do *back-end* (API), utilizando o *framework* NestJS, e outro para o *front-end*, mediante o *framework* Next.js. Os comandos de inicialização dos respectivos *frameworks* foram utilizados para configurar os projetos iniciais.
4. implementação do *back-end* com Domain-driven Design (DDD): Por meio do conceito do DDD (EVANS, 2016), cada domínio do projeto foi analisado e implementado de forma isolada. Isso envolveu a definição dos métodos e das estruturas de dados necessárias para cada parte do sistema, garantindo uma arquitetura coesa e modular.
5. desenvolvimento e aplicação de testes: Os testes foram desenvolvidos para garantir a qualidade e a robustez da solução. Além dos testes automatizados,

também foram realizados testes manuais no Postman para verificar o funcionamento correto dos pontos de acesso (*endpoints)* da API.

6. implementação da interface gráfica com Material Design: A interface gráfica foi desenvolvida em conformidade com os padrões do Material Design, a fim de assegurar a usabilidade na interação humano-computador e facilitar a elaboração das telas (MEW, 2015). Concomitantemente ao desenvolvimento da interface, a integração com a API foi realizada para garantir a interação adequada entre o *front-end* e o *back-end* do sistema.
7. implantação em plataformas *serverless* (PaaS): Por fim, a API e o *front-end* foram implantados em plataformas *serverless*, utilizando o conceito de Platform as a Service (PaaS). Isso permitiu uma implantação simplificada e escalável da solução, garantindo alta disponibilidade e desempenho, sem provisionamento de servidor (SRINIVAS; REDDY; QYSER, 2012).

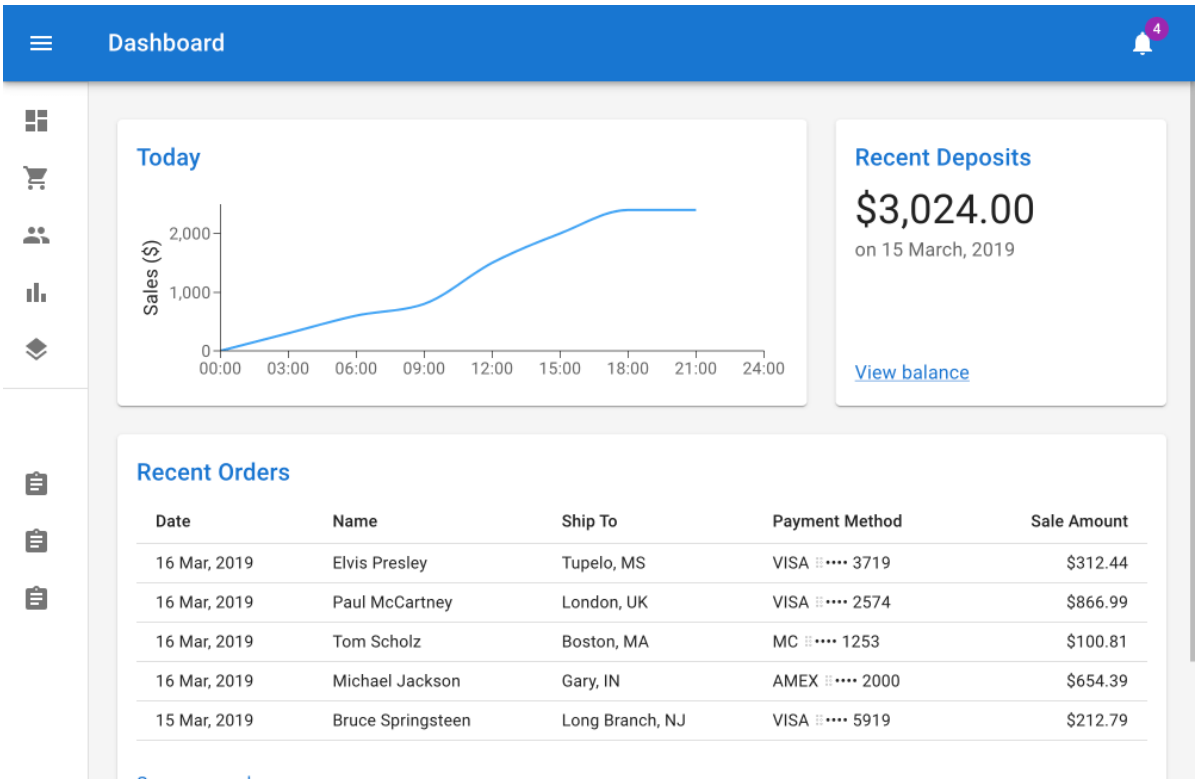
Essas etapas foram seguidas de forma iterativa e incremental, permitindo o desenvolvimento progressivo da solução e a entrega de valor de forma contínua ao longo do ciclo de desenvolvimento.

6. MOCKUP DA PROPOSTA DE SOLUÇÃO

Neste capítulo, são apresentadas as telas desenvolvidas para a proposta prática deste trabalho, ilustrando a interface e funcionalidades que foram implementadas. Embora não tenha sido criado um *mockup* detalhado com uso de ferramentas como Figma ou Adobe XD, a utilização da biblioteca de componentes MUI (ou Material-UI) possibilitou a aplicação de usabilidade no desenvolvimento do *front-end* deste projeto.

A escolha pelo MUI deve-se ao fato de ser uma biblioteca robusta que segue os princípios do Material Design, um sistema de design desenvolvido pela Google. O Material Design oferece diretrizes compreensivas para a criação de interfaces intuitivas e consistentes, de modo a focar na experiência do usuário e na estética moderna (MEW, 2015). Utilizar componentes prontos do MUI proporciona diversas vantagens, incluindo consistência visual, acessibilidade e responsividade, além de acelerar o desenvolvimento das interfaces (ELROM, 2021).

Figura 2 — Material UI



Fonte: MUI, 2024.

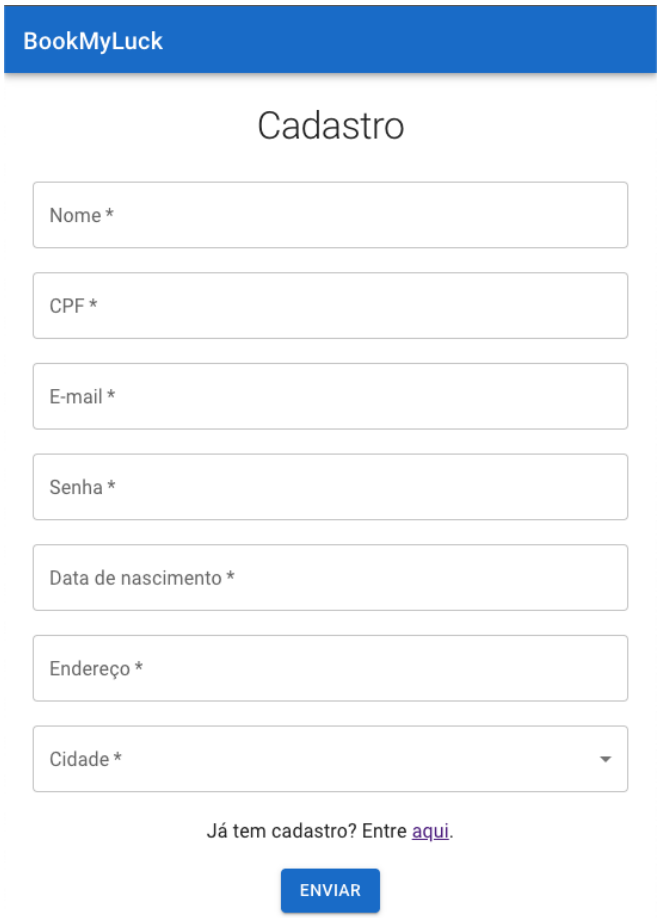
A Figura 2 exemplifica o Material Design, destacando como os componentes do MUI são utilizados para criar interfaces que são ao mesmo tempo funcionais e visualmente atraentes. A interface do usuário é composta por telas fundamentais como a de cadastro, *login*, página principal e alteração de dados pessoais, todas construídas com componentes do MUI. Esses componentes são pré-estilizados, seguindo práticas de design consagradas, e podem ser facilmente customizados para atender às necessidades específicas do projeto.

Para o cadastro de usuários, foi desenvolvida uma tela que inclui campos como nome, CPF, e-mail, senha, data de nascimento, cidade e endereço, todos utilizando campos estilizados do MUI. A tela de *login* é simples, com campos para e-mail e senha estilizados e programados da mesma forma. Na página principal, os usuários poderão visualizar e registrar interesses, além de navegar para uma página em que poderão ver e editar seus cadastros. A tela de adição de interesses permite que os usuários selecionem o tipo de serviço, preço, intervalo de datas e horários e a quantidade de pessoas. Essas interfaces são ilustradas pelas Figuras 3 a 6.

A integração dos componentes do MUI (2024) não só garante a aderência aos padrões do Material Design, mas também facilita a manutenção e escalabilidade do

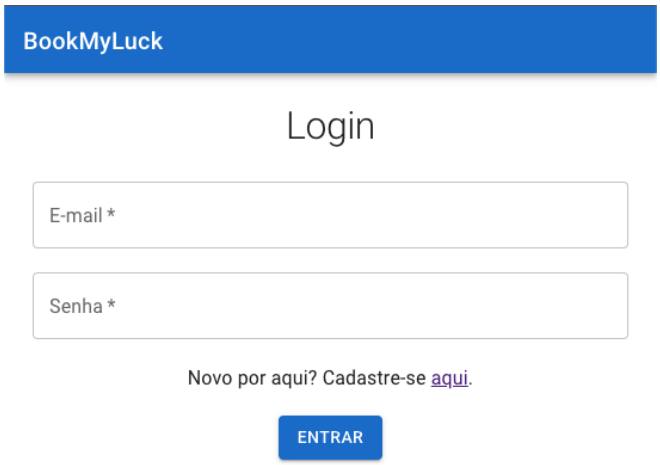
código. Tal abordagem também assegura que a interface seja responsiva e acessível, proporcionando uma experiência de usuário de alta qualidade (ELROM, 2021). O uso do Material Design e dos componentes do MUI destaca-se como uma escolha estratégica para o desenvolvimento das interfaces deste projeto, o que garante um resultado que é tanto eficiente quanto esteticamente agradável.

Figura 3 — Tela de cadastro



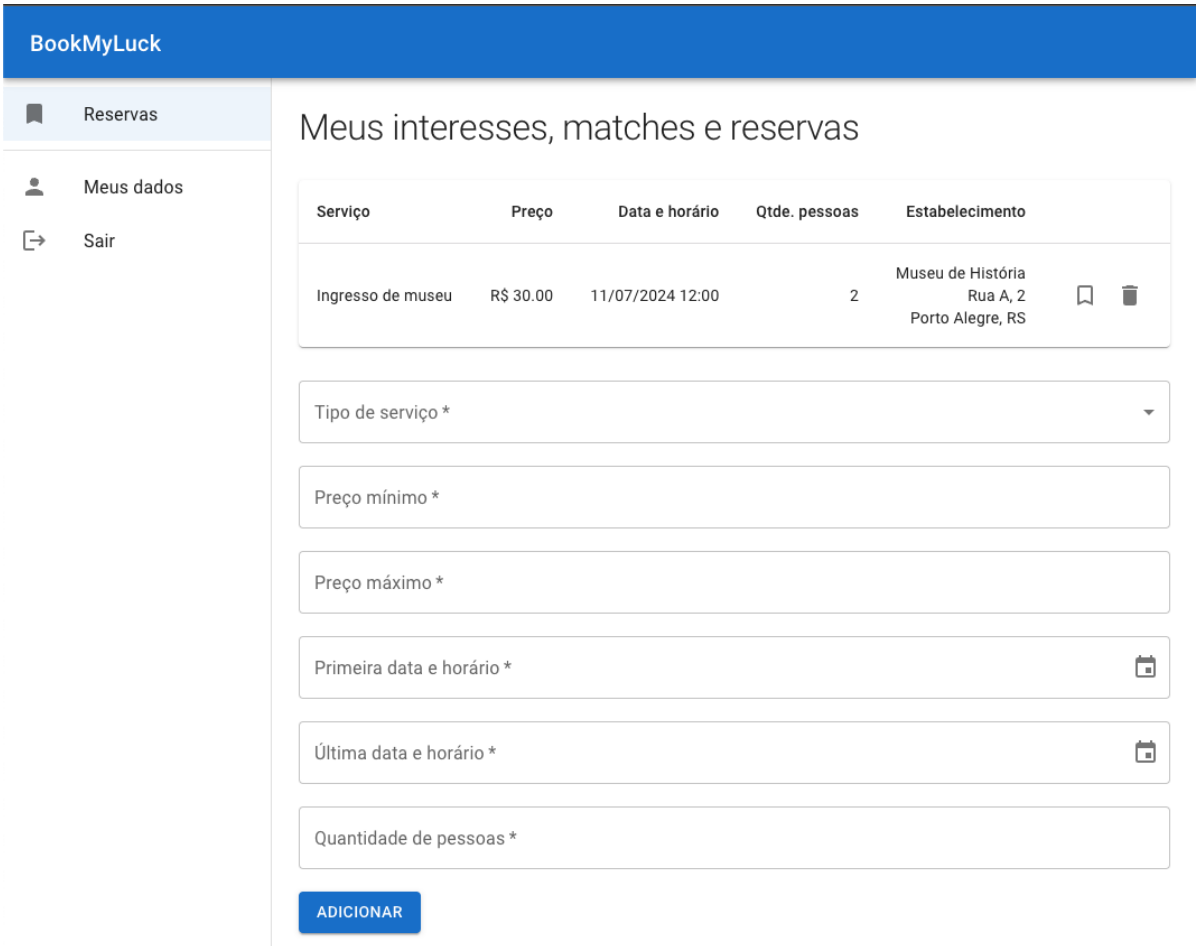
Fonte: Elaborada pelo autor, 2024.

Figura 4 — Tela de login



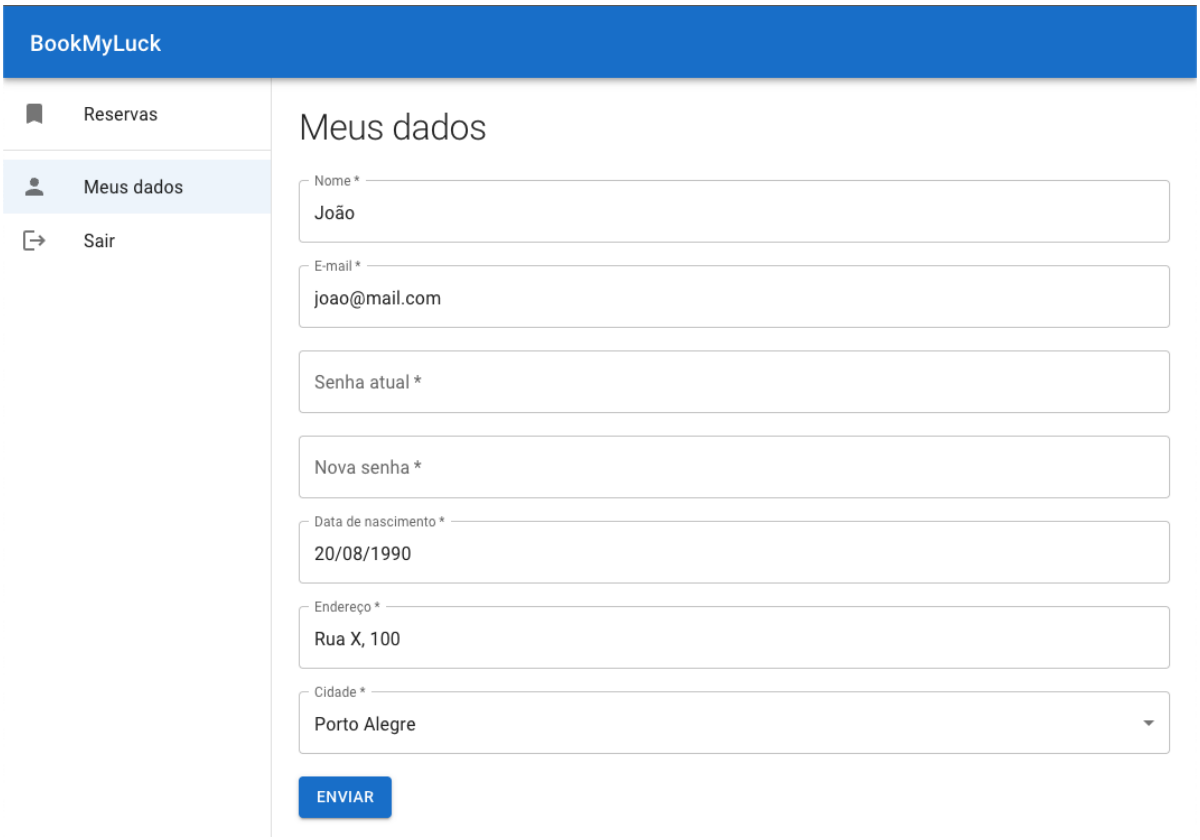
Fonte: Elaborada pelo autor, 2024.

Figura 5 — Tela principal



Fonte: Elaborada pelo autor, 2024.

Figura 6 — Tela de alteração de dados pessoais



Fonte: Elaborada pelo autor, 2024.

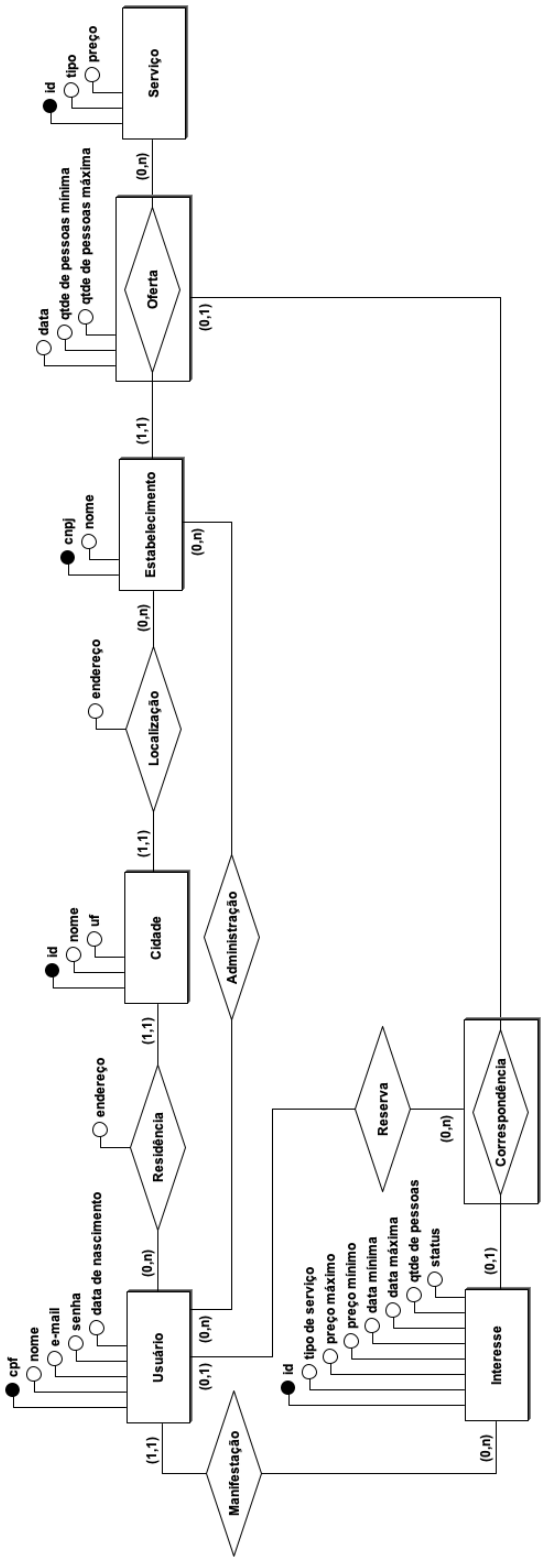
7. ARQUITETURA DE SOFTWARE

A arquitetura de software do projeto foi meticulosamente planejada e implementada para garantir a eficiência, escalabilidade e robustez da solução. O sistema foi desenvolvido em um container Docker, proporcionando um ambiente de desenvolvimento isolado e portátil. Utilizando o *framework* NestJS em Node.js, a API foi construída seguindo padrões modernos de desenvolvimento, oferecendo uma estrutura modular e escalável para lidar com as operações de *back-end*.

Para armazenar os dados, optou-se por um banco de dados MySQL, gerenciado com auxílio do phpMyAdmin, cuja interface gráfica facilitou a administração e visualização dos dados. Antes da implementação, foi criado um modelo conceitual e um modelo lógico mediante o programa brModelo. O modelo conceitual, representado na Figura 7, foi refinado para o modelo lógico, representado na Figura 8. Esses modelos serviram de esboço para as entidades posteriormente codificadas nas tabelas de usuários, cidades, estabelecimentos, serviços, ofertas, interesses e correspondências. A entidade Usuário contém atributos de informação pessoal e um identificador, bem como relaciona-se com a entidade Cidade para formar a identificação do endereço do usuário. Esse relacionamento compõe uma chave estrangeira para o identificador da cidade. Da mesma forma é modelada a entidade Estabelecimento, mas com atributos como nome da empresa e CNPJ. Tal entidade relaciona-se com a entidade Serviço para compor a entidade associativa Oferta. Esse relacionamento é logicamente feito por uma chave estrangeira do estabelecimento no serviço, que faz chave estrangeira na tabela de oferta mediante os respectivos identificadores numéricos. De modo semelhante, modelaram-se os interesses do usuário em uma entidade Interesse, logicamente ligada à entidade Usuário por meio do identificador do usuário. Os atributos dessa entidade incluem as especificações de interesse por serviço do usuário, como intervalos de preço e data, tipo de serviço e quantidade de pessoas, tal qual um número inteiro para representar status — o status 1 (um) significa que o sistema não encontrou correspondência para o dado interesse. A correspondência entre determinado interesse e oferta é modelada por uma entidade

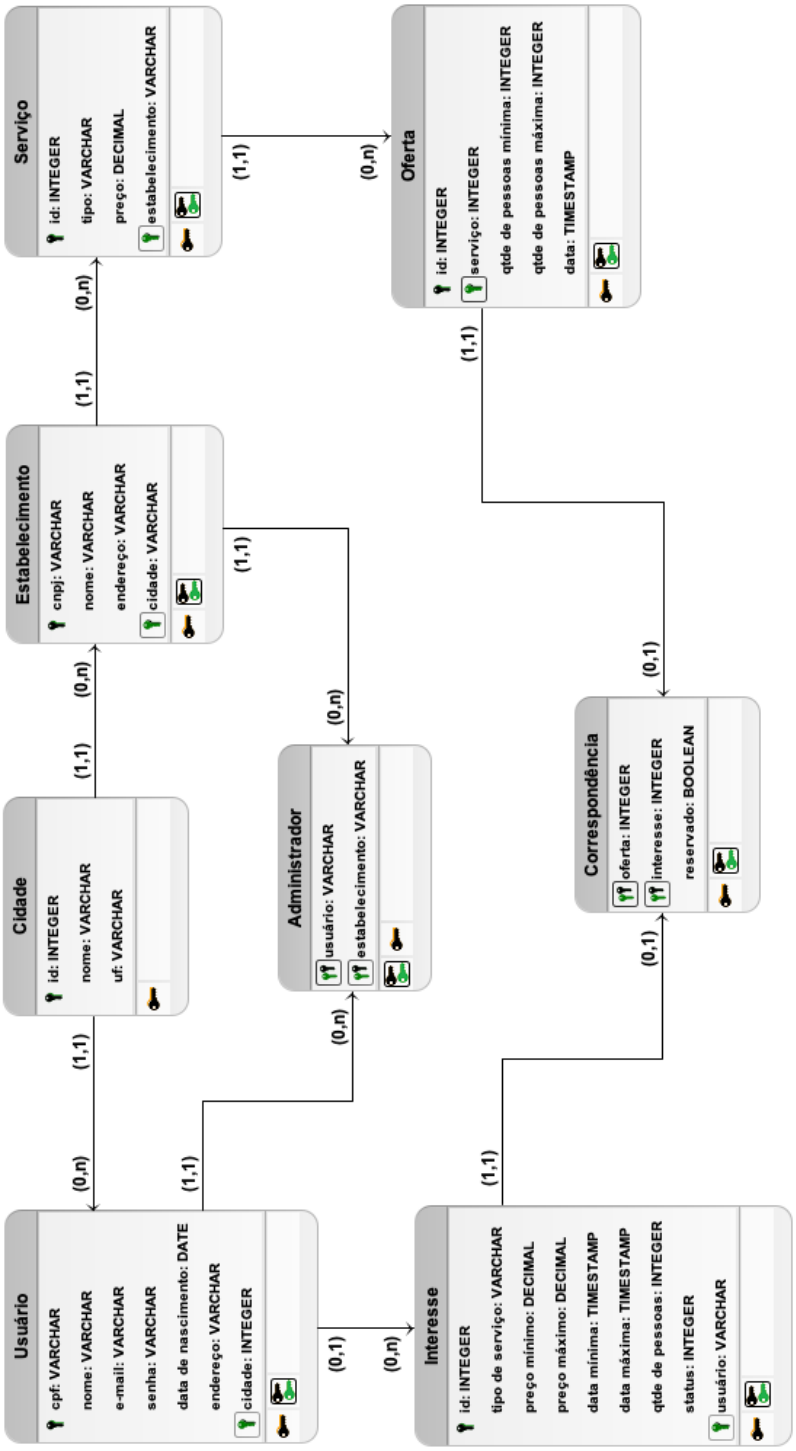
associativa, que resultou em uma tabela contendo identificador de oferta e interesse únicos.

Figura 7 — Modelo conceitual



Fonte: Elaborada pelo autor, 2024.

Figura 8 — Modelo lógico



Fonte: Elaborada pelo autor, 2024.

O acesso e manipulação dos dados do banco foram simplificados com o uso do TypeOrm, um *object-relational mapping* (ORM) que mapeia objetos JavaScript para o banco de dados relacional (TYPEORM, 2024). Esse mapeamento facilitou a integração entre a lógica da aplicação e o banco de dados, seguindo os princípios do Domain-driven Design.

Na arquitetura de software deste projeto, a segurança da informação foi prioridade. Para proteger as comunicações e os dados dos usuários, implementou-se a autenticação via JSON Web Token — JWT. Este mecanismo garante que apenas usuários autenticados possam acessar e interagir com determinados *endpoints* da API, fornecendo um método seguro e eficiente de verificação de identidade (SHINGALA, 2019). Além disso, as senhas são criptografadas antes de serem armazenadas no banco de dados. Essas medidas de segurança não apenas protegem os dados contra acessos não autorizados, porém também garantem a integridade e a confidencialidade das informações.

Além da estruturação das entidades, foram desenvolvidos testes unitários usando a biblioteca Jest, seguindo os modelos gerados pela inicialização do NestJS. Esses testes garantiram a qualidade e a robustez do código, enquanto o Postman foi utilizado para testar manualmente os *endpoints* da API, de modo a assegurar o correto funcionamento das operações.

O *front-end*, desenvolvido em Next.js, um *framework* JavaScript baseado em React (THAKKAR, 2020), proporcionou uma experiência de usuário fluida e responsiva. Utilizando a biblioteca de componentes MUI, a interface gráfica adotou os padrões do Material Design da Google, garantindo uma apresentação visualmente atraente e consistente, além de simplificar a implementação da interface gráfica (ELROM, 2021). A estilização adicional dos componentes foi feita por intermédio de pré e pós-processadores de Cascading Style Sheet (CSS) — a linguagem de estilização padrão dos navegadores —, que simplificam a sintaxe dessa linguagem e convertem os códigos para folhas de estilo com melhorias, para o suporte de navegadores legados por exemplo (QUEIRÓS, 2018). Para comunicação entre o *front-end* e *back-end*, as requisições à API são realizadas por meio da Fetch API do navegador e do Node.js.

Para garantir alta disponibilidade e escalabilidade, tanto o *front-end* quanto o *back-end* foram implantados de forma *serverless* em plataformas como serviços (PaaS) do Vercel e Render respectivamente. O banco de dados foi criado na nuvem utilizando a plataforma como serviço Aiven, enquanto a fila do RabbitMQ foi hospedada no CloudAMQP, oferecendo um serviço robusto e confiável para o processamento de mensagens assíncronas (SELVARAJ, 2023). A configuração de Cross-origin Resource Sharing (CORS) no *back-end* restringiu o acesso apenas ao domínio da aplicação *front-end*, para garantir segurança e controle sobre as

requisições externas. Essa arquitetura, cuidadosamente planejada e implementada, proporcionou uma base sólida para o desenvolvimento e operação da solução. Todo o código-fonte foi versionado no Git e publicado no GitHub, facilitando o controle de versão e eventual colaboração entre futuros desenvolvedores.

Neste projeto, vale destacar que, por se tratar de um MVP, as interfaces para os estabelecimentos ainda não foram implementadas. Em vez disso, os dados dos estabelecimentos podem ser adicionados manualmente no banco de dados pelo proprietário e desenvolvedor do projeto. Esta abordagem é semelhante ao método utilizado no início da Easy Taxi, quando era um MVP do tipo Mágico de Oz, em que as solicitações de viagem de táxi eram feitas por um formulário e os proprietários da plataforma realizavam manualmente a chamada dos veículos. Esse método permite validar a viabilidade da solução e coletar *feedback* valioso antes de investir no desenvolvimento completo das funcionalidades automatizadas (CAROLI, 2015).

8. VALIDAÇÃO DA SOLUÇÃO

A validação da solução desenvolvida foi conduzida por meio de uma abordagem que incluiu tanto testes automatizados quanto manuais. Este capítulo detalha os métodos empregados para validar a funcionalidade e a eficácia da plataforma.

Tal procedimento de validação assegurou que a solução desenvolvida atendesse aos requisitos funcionais e de qualidade, de forma a proporcionar uma experiência confiável e satisfatória para os usuários.

8.1. Testes automatizados

Foram escritos testes unitários utilizando a biblioteca Jest para JavaScript, abrangendo diversas funcionalidades chave da API, entre elas a autorização do usuário para realizar ações na plataforma. Os testes produzidos seguem enumerados:

- alteração de dados de estabelecimento;
- autoridade sobre estabelecimento;
- cadastro de interesses inválidos e válidos;
- cadastro de ofertas inválidas;
- remoção de oferta;
- listagem de serviço;
- autoridade sobre serviço.

Os testes unitários foram executados utilizando a base de dados local para garantir a consistência e a funcionalidade adequada das operações realizadas pelo sistema. A utilização de testes automatizados permitiu identificar e corrigir eventuais falhas ou inconsistências no código de forma eficiente.

8.2. Testes manuais

Além dos testes automatizados, todas as funcionalidades da plataforma foram testadas manualmente por meio da interface gráfica desenvolvida. Os testes manuais

abrangeram diversas situações de uso, a garantir que a plataforma se comportasse conforme o esperado em diferentes cenários.

A validação da solução também foi realizada por meio do Postman (2024), uma ferramenta que permite fazer consultas a APIs de maneira intuitiva. Os *endpoints* da API foram testados para verificar a integridade e a precisão das respostas fornecidas pelo sistema.

Para testar a sistema desenvolvido, é necessário seguir alguns passos. Primeiramente, um usuário novo deve ser cadastrado, podendo utilizar CPFs gerados automaticamente por ferramentas on-line. A cidade informada no cadastro precisa ser Porto Alegre para fins de validação. Após o cadastro, o acesso é realizado por meio do e-mail e senha previamente cadastrados. Na página principal, o usuário tem a opção de clicar no botão “Adicionar interesse” e cadastrar um interesse do tipo “Ingresso de museu”, especificando um intervalo de datas para o mês de julho. Este teste é direcionado à funcionalidade de reservas para o serviço hipotético de ingresso para o “Museu de História”, que foi cadastrado especificamente para validar a implementação deste trabalho de conclusão de curso. Caso o intervalo inserido no cadastro não coincida com as datas registradas para o serviço ou o preço de R$30,00 não esteja limitado pelos valores mínimo e máximo escolhidos, uma indicação de nenhuma oferta de serviço encontrada será mostrada. O mesmo ocorrerá caso o tipo de serviço seja diferente do supracitado, se for informado um número de pessoas maior que quatro ou caso a cidade do usuário não seja Porto Alegre. O processamento do pedido pode demorar alguns minutos.

9. REGISTROS DAS EVIDÊNCIAS DO PROJETO

Este capítulo destina-se a fornecer acesso às evidências do projeto desenvolvido, incluindo o código-fonte, documentação e outros artefatos relevantes.

O repositório GitHub disponível em *https://github.com/fil1pe/tcc-pos-graduacao* contém todos os registros e recursos relacionados ao presente trabalho. O acesso ao repositório fornecerá uma visão detalhada do trabalho realizado, de modo a possibilitar uma análise mais aprofundada das etapas práticas de desenvolvimento e das tecnologias utilizadas.

O *website* principal, em que é possível interagir com a plataforma, está disponível em *https://book-my-luck.vercel.app*. Já a API, que serve como interface para o *back-end* do sistema, pode ser acessada em *https://book-my-luck-api.onrender.com/api*.

10. CONSIDERAÇÕES FINAIS E EXPECTATIVAS

Este trabalho de conclusão de curso representou uma oportunidade valiosa para colocar em prática muitos dos conceitos aprendidos ao longo do curso de especialização em Desenvolvimento Full Stack. A implementação deste projeto, por exemplo, permitiu ao autor melhorar significativamente seu entendimento sobre bancos de dados relacionais e aplicar, de forma concreta, o conceito de ORMs. A utilização do TypeOrm facilitou a interação com o banco de dados MySQL, solidificando o conhecimento e prática nessa área. Além disso, a prática de implementação de APIs com o *framework* NestJS foi especialmente enriquecedora. O estudo deste *framework* durante as aulas do curso e na elaboração deste trabalho estendeu-se a outras tecnologias, como Spring para Java, com o qual o NestJS compartilha muitas semelhanças. O aprofundamento no aprendizado destes artifícios favorece a participação no mercado de trabalho e na engenharia de *software*.

Outro aspecto crucial foi a aplicação dos conceitos de computação em nuvem aprendidos ao longo do curso. A publicação de todo o sistema em plataformas *serverless* não apenas demonstrou a viabilidade dessas tecnologias, mas também sintetizou o esforço do autor para obter sua certificação em Amazon Web Services (AWS), que sucedeu em virtude dos incentivos dos professores.

Embora a especialidade atual do autor seja o desenvolvimento *front-end*, seguir os princípios do Material Design foi uma experiência reveladora. Reconheceu-se a importância desses princípios para criar interfaces consistentes e intuitivas, algo que é fundamental na carreira.

A segurança da informação foi outro campo em que conhecimentos teóricos e práticos foram aplicados. Implementar criptografia de dados e *tokens* JWT garantiu que o sistema fosse seguro e confiável, protegendo as informações dos usuários de maneira eficaz.

Todo o aprendizado adquirido ao longo do curso culminou neste TCC, que é funcional e multdisciplinar. No entanto, reconhece-se que há melhorias a serem

implementadas em trabalhos futuros. Por exemplo, desenvolver interfaces para os estabelecimentos, visto que este trabalho focou na perspectiva do usuário final, considerando que os estabelecimentos poderiam ser inseridos manualmente via phpMyAdmin em uma visão de MVP. Além disso, é necessário desenvolver mais testes, inclusive de integração, e melhorar o ciclo de DevOps com ferramentas como GitHub Actions. Não existe *feedback* de erro 404 — que ocorre ao tentar acessar página inexistente —, o que outrossim deve ser prioridade futura.

A trajetória no curso, que encerrou neste trabalho, traz expectativas de crescimento profissional, incentivo para mais especializações e uma maior participação na comunidade de desenvolvimento. As oportunidades decorrentes desta formação desempenharam incentivo à aplicação dos conhecimentos adquiridos e à contínua evolução da carreira, de forma a sempre buscar novas oportunidades de aprendizado e desenvolvimento.

REFERÊNCIAS

ANDERSON, Charles. Docker. **IEEE Software**, v. 32, n. 3, p. 102-c3, 2015. Disponível em: https://ieeexplore.ieee.org/abstract/document/7093032. Acesso em: 15 maio 2024.

BOOKSY. **Agende Serviços de Beleza em Poucos Cliques**. Disponível em: https://booksy.com/pt-br/. Acesso em: 15 maio 2024.

CAROLI, Paulo. **EasyTaxi uma história de MVP**. 2015. Disponível em: https://caroli.org/easytaxi-uma-historia-de-mvp/. Acesso em: 15 maio 2024.

ELROM, Elad. React Router and Material-UI. In: **React and Libraries:** Your Complete Guide to the React Ecosystem. Berkeley: Apress, 2021. p. 79-113.

EVANS, Eric. **Domain-Driven Design:** Atacando as Complexidades no Coração do Software. 3. ed. Rio de Janeiro: Alta Books, 2015.

FIGUEIREDO, Eduardo. Requisitos funcionais e requisitos não funcionais. **ICEX, DCC/UFMG**, v. 14, 2011. Disponível em: https://homepages.dcc.ufmg.br/~figueiredo/disciplinas/aulas/req-funcional-rnf_v01.pdf. Acesso em: 15 maio 2024.

GET IN. **Descubra novos restaurantes, entre na fila e faça reservas online**. Disponível em: https://www.getinapp.com.br/. Acesso em: 15 maio 2024.

JOHNSON, Heather A. Trello. **Journal of the Medical Library Association**, v. 105, n. 2, p. 209, 2017. Disponível em: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5370621/. Acesso em: 15 maio 2024.

MEW, Kyle. **Learning Material Design**. Birmingham: Packt Publishing, 2015.

MUI. **9+ Free React Templates**. Disponível em: https://mui.com/material-ui/getting-started/templates/. Acesso em: 15 maio 2024.

POSTMAN. **Postman API Platform**. Disponível em: https://www.postman.com/. Acesso em: 15 maio 2024.

QUEIRÓS, Ricardo. CSS preprocessing: Tools and automation techniques. **Information**, v. 9, n. 1, p. 17, 2018. Disponível em: https://www.mdpi.com/2078-2489/9/1/17. Acesso em: 15 maio 2024.

SABO, Mario. **NestJS**. 2020. Tese (Graduação em Matemática e Ciência da Computação) - Department of Mathematics, Josip Juraj Strossmayer University of Osijek, Osijek, 2020. Disponível em: https://zir.nsk.hr/en/islandora/object/mathos%3A441. Acesso em: 15 maio 2024.

SELVARAJ, Sivaraj. Queues and Job Scheduling. In: **Building Real-Time Marvels with Laravel:** Create Dynamic and Interactive Web Applications. Berkeley: Apress, 2023. p. 219-235.

SHINGALA, Krishna. JSON Web Token (JWT) based client authentication in Message Queuing Telemetry Transport (MQTT). **arXiv:1903.02895**, 2019. Disponível em: https://arxiv.org/abs/1903.02895. Acesso em: 15 maio 2024.

SRINIVAS, J.; REDDY, K. Venkata Subba; QYSER, A. Moiz. Cloud computing basics. **International journal of advanced research in computer and communication engineering**, v. 1, n. 5, p. 343-347, 2012.

SYMPLA. **Ingressos para Eventos, Teatros, Shows, Cursos e mais**. Disponível em: https://www.sympla.com.br/. Acesso em: 15 maio 2024.

THAKKAR, Mohit. Next.js. In: **Building React Apps with Server-Side Rendering:** Use React, Redux, and Next to Build Full Server-Side Rendering Applications. Berkeley: Apress, 2020. p. 93-137.

TYPEORM. **Amazing ORM for TypeScript and JavaScript**. Disponível em: https://typeorm.io/. Acesso em: 15 maio 2024.

VIEIRA, André G. et al. Computação Serverless: Conceitos, Aplicações e Desafios. In: **Minicursos do XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos**. Porto Alegre: Sociedade Brasileira de Computação, 2020. p. 190-236.