Mestre Sala: Desenvolvimento de um sistema web para gerenciamento de disponibilidade e reserva de salas

Rafael Barbosa da Silva, Fernando Sambinelli

[1]Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP) – Câmpus Hortolândia – São Paulo - SP - Brasil

barbosa.rafael1@aluno.ifsp.edu.br, sambinelli@ifsp.edu.br

Abstract. *Considering the existence of many organizations with different indoor environments that need control to know which ones are free for use or not at certain times, this paper presents the development of a web system to manage reservations and availability of room keys in corporate, residential or any other environments. The objective of the system is to allow users to reserve rooms and check their availability. This project was developed using web development technologies, including React.js and Spring MVC frameworks.*

Resumo. *Considerando a existência de muitas organizações com diferentes ambientes interiores que necessitam de controle para saber quais estão livres para uso ou não em determinados momentos, o presente trabalho apresenta o desenvolvimento de um sistema web para gerenciamento de reservas e disponibilidade de salas em ambientes corporativos, residenciais ou em qualquer outro. O objetivo do sistema é permitir aos usuários reservar salas e consultá-las quanto à sua disponibilidade de maneira remota e online. O projeto foi desenvolvido utilizando tecnologias de desenvolvimento web, incluindo os frameworks React.js e Spring MVC.*

1. Introdução

Nos últimos anos, o avanço da tecnologia tem transformado a maneira como trabalhamos e vivemos, impactando diretamente o setor de gerenciamento de espaços físicos, como salas de reuniões, auditórios e escritórios. De acordo com a pesquisa Global Workplace Survey [GROUP 2019], 80% dos entrevistados afirmaram que "a flexibilidade no local de trabalho aumentaria a produtividade", o que implica em uma crescente demanda por espaços compartilhados e soluções tecnológicas eficientes para o gerenciamento desses ambientes. Nesse contexto, torna-se crucial a adoção de soluções tecnológicas capazes de otimizar o gerenciamento desses espaços, garantindo a disponibilidade e a utilização eficiente dos recursos.

Além disso, uma matéria publicada pela *Infraspeak Team* aponta que a gestão do espaço de trabalho é uma “disciplina” do *Facility Management* (termo utilizado para definição de gestão de infraestruturas), existindo uma sobreposição clara entre ambos, uma vez que ele influencia a produtividade, a satisfação e a motivação dos funcionários. [INFRASPEAK 2022]. Nesse sentido, a utilização de tecnologias como sistemas *web* de gerenciamento de espaços pode ser uma alternativa eficiente para a otimização desse processo, garantindo uma gestão mais eficiente e transparente.

O gerenciamento de espaços envolve uma série de atividades, desde o controle de acesso aos ambientes até a gestão de reservas e disponibilidade dos mesmos. Para facilitar esse processo, o uso de sistemas de gerenciamento de salas tem se tornado cada vez mais comum em empresas e instituições.

Nesse contexto, o presente trabalho apresenta o desenvolvimento de um sistema *web* para gerenciamento de reservas e disponibilidade de espaços em ambientes corporativos, residenciais ou em qualquer outro, com o objetivo de permitir aos usuários reservar salas e consultá-las quanto à sua disponibilidade de maneira remota e *online*.

2. Referencial Teórico

Nesta seção são abordados os principais conceitos e aspectos técnicos quanto ao desenvolvimento deste trabalho:

2.1. *Facility Management*

O termo *facility management* representa uma atividade crucial em organizações, capaz de trazer diversos benefícios dentro de uma instituição. A *Infraspeak Team* aponta, em outro material, acerca do *facility management*, de que a aplicação deste conceito envolve a gestão eficiente do espaço e tarefas de uma organização, sendo capaz de influenciar positivamente o crescimento e a produtividade dela, além de economizar custos e reduzir gastos necessários em diversos âmbitos. Tudo isso, através de tecnologias aplicadas como ferramentas do ambiente, impactando sua atuação sem trazer maiores complexidades aos seus colaboradores [INFRASPEAK 2023].

2.2. *Design* de interfaces e Prototipagem

O *design* de interfaces deve levar em consideração diversos fatores, como o público-alvo, a funcionalidade do sistema e as diretrizes de *design* [TIDWELL 2006]. Já a prototipagem pode ser feita de diversas formas, desde esboços e *wireframes* até protótipos funcionais em alta fidelidade. De acordo com a pesquisa "*Prototyping: A Practitioner's Guide*"[WARFELL 2009], a prototipagem é uma técnica fundamental no processo de *design*, permitindo a exploração rápida de múltiplas ideias e conceitos.

2.3. Arquitetura MVC

A arquitetura *Model-View-Controller* (MVC) é um padrão de desenvolvimento de *software* amplamente utilizado na criação de aplicações *web*. A utilização da arquitetura MVC é considerada uma das melhores práticas de desenvolvimento de *software* para aplicações *web* [OLIVEIRA 2019]. Essa arquitetura separa as responsabilidades em três componentes principais: *Model*, *View* e *Controller*. O *Model* representa os dados e regras de negócio da aplicação, o *View* é responsável pela apresentação visual da aplicação para o usuário e o *Controller* é responsável por intermediar a comunicação entre o *Model* e o *View*, recebendo as requisições do usuário e fazendo as operações necessárias no *Model*.

Dessa forma, a arquitetura MVC foi escolhida como base para o desenvolvimento do sistema *web* proposto neste trabalho, permitindo a separação de responsabilidades e a modularização do código.

2.4. *Spring Framework*

O *Spring Framework* é um dos *frameworks* mais populares para desenvolvimento de aplicações Java, sendo adotado por mais de 50% dos desenvolvedores Java em todo o mundo [REBELLABS 2020]. Ele oferece um conjunto de recursos que facilitam a implementação de soluções robustas e escaláveis, incluindo injeção de dependência, AOP (*Aspect Oriented Programming*), JPA (*Java Persistence API*) e muito mais. O *Spring Framework* tem a capacidade de ser aplicado de acordo com o padrão arquitetural MVC, tornando a aplicação consistente, flexível e minimamente acoplada aos seus módulos e componentes internos [MANE et al. 2013].

2.5. React.js

O React.js é uma biblioteca de código aberto utilizada para desenvolvimento de interfaces de usuário em aplicações *web*. Ele foi criado pelo Facebook em 2011 e atualmente é mantido por uma comunidade de desenvolvedores. Uma das principais características do React.js é o uso de componentes, que são blocos de código reutilizáveis que podem ser compostos para formar interfaces complexas. Isso permite que o desenvolvimento seja mais rápido e eficiente, além de facilitar a manutenção do código. Segundo um relatório da empresa de análise de dados, RedMonk, o React.js é uma das bibliotecas JavaScript mais populares do mundo [O'GRADY 2021].

2.6. JSON

JSON (*JavaScript Object Notation*) é um formato de troca de dados leve e fácil de ler e escrever, tanto para seres humanos quanto para máquinas. Ele é amplamente utilizado em aplicações *web* para enviar e receber dados entre o servidor e o cliente. O JSON é baseado em um subconjunto da linguagem JavaScript e é composto por pares chave-valor, o que torna-o fácil de entender e manipular. Devido à sua popularidade, esse esquema de escrita de dados vem se estabelecendo como um esquema padronizado em diversos sistemas, além de desempenhar papéis primordiais no funcionamento deles [PEZOA et al. 2016].

3. Trabalhos Correlatos

Esta seção apresenta alguns trabalhos correlatos ao sistema proposto por este trabalho. Uma pesquisa foi realizada utilizando o sistema de busca do *Google* a fim de encontrar sistemas gerenciadores de reservas de salas ou com propostas semelhantes a este projeto. Dois trabalhos foram encontrados, sendo eles os sistemas *Smartrooms* e *Invitee*, ambos são apontados a seguir.

3.1. *Smartrooms*

O sistema *Smartrooms* é um sistema *web* desenvolvido para gerenciamento de salas de reunião em empresas. O sistema permite aos usuários reservarem salas de maneira virtual, além de fornecer informações sobre a disponibilidade das salas em tempo real. O sistema utiliza uma interface simples e intuitiva, permitindo que os usuários realizem a reserva de uma sala em poucos cliques. Ele é desenvolvido utilizando tecnologias *web* modernas, como o *framework* Vue.JS e o *back-end* em Java com Spring Framework. A Figura 1 apresenta uma interface de apresentação do sistema em seu *site* oficial.

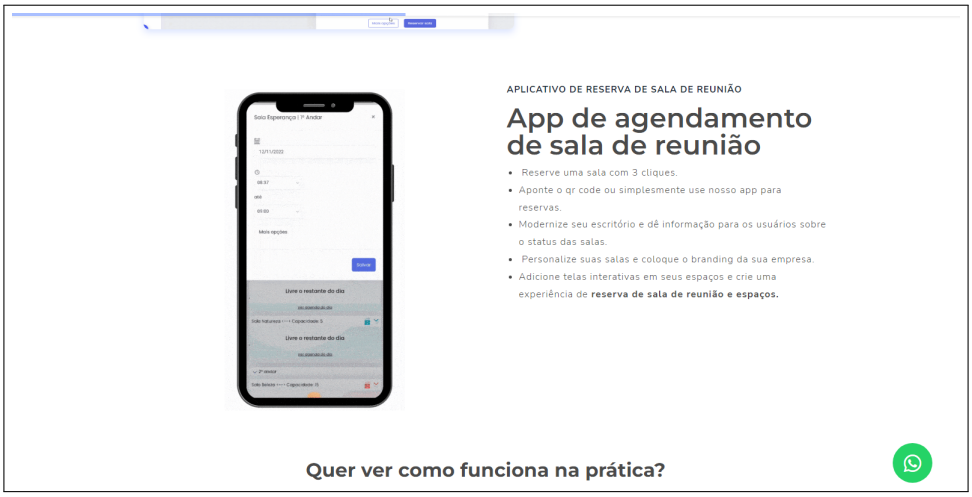

Figura 1. Página inicial do Smartrooms

3.2. *Invitee*

O *Invitee* é um sistema *web* desenvolvido para gestão de salas de reunião em empresas. O sistema permite que os usuários reservem salas de reunião, além de fornecer informações sobre a disponibilidade das salas em tempo real. Ele também oferece recursos avançados, como a possibilidade de agendar serviços adicionais, como *coffee-break* e equipamentos audiovisuais. O sistema foi desenvolvido utilizando tecnologias modernas, como o *framework* Laravel em PHP. A Figura 2 apresenta uma interface de apresentação do sistema em seu *site* oficial.

Figura 2. Página inicial do Invitee

3.3. Comparativo entre trabalhos

Para a melhor compreensão das semelhanças e diferenças entre os trabalhos correlatos e o que foi desenvolvido neste trabalho, a Tabela 1 apresenta um comparativo entre as funcionalidades oferecidas entre tais sistemas.

Tabela 1. Comparativo entre trabalhos correlatos e o sistema deste presente trabalho

Funcionalidades	Smartrooms	Invitee	Mestre Sala
Cadastro de salas e serviços	Sim	Sim	Sim
Cobrar por reservas no sistema	Sim	Não	Não
Web 100% online	Sim	Sim	Sim
Gratuito	Não	Não	Sim
Integração com outros sistemas	Sim	Sim	Não

4. Metodologia

O modelo utilizado para o desenvolvimento do projeto foi o incremental para a realização de atividades, sendo as principais delas a análise, a prototipagem, a codificação e os testes da aplicação.

Durante o desenvolvimento da aplicação, artefatos de engenharia de *software* foram desenvolvidos, tais como: Canvas de Valor[1], Diagrama de Entidade e Relacionamento, histórias de usuário e protótipos de alta fidelidade da aplicação. Após a construção desses materiais, iniciou-se a codificação do sistema, onde as funcionalidades de login e de cadastro, de agendamento e de edição de reservas foram aplicadas. Anterior às tais funcionalidades, a base de dados foi criada e carregada com informações a serem utilizadas para auxiliar o desenvolvimento do sistema. Por fim, alguns testes da *API* do sistema foram aplicados para garantir o funcionamento da solução final.

Para a realização do trabalho, foi utilizado o *framework* Spring com base em Java para a estruturação do *back-end* do projeto. Outras linguagens utilizadas foram o HTML, Sass e JavaScript reunidos no *framework* React.js para a construção do *front-end* da aplicação. Para o desenvolvimento do sistema foram utilizadas duas ferramentas de edição de código, o Visual Studio Code e o IntelliJ IDEA.

5. Desenvolvimento

Esta seção apresenta o desenvolvimento realizado neste trabalho, com descrições de cada tarefa realizada nas diferentes fases de sua elaboração.

5.1. Canvas de Valor

A primeira atividade a ser realizada para o refinamento das propostas e finalidade do projeto foi o desenvolvimento de um Canvas de valor, que torna explícito como um projeto cria valor para seus clientes [POKORNÁ et al. 2015]. No ano de 2020, entre empreendedores e microempreendedores que fecharam as portas de seus negócios, 17% diziam não ter feito nenhum planejamento de seu negócio [SEBRAE 2023]. Com isso, o Canvas de valor surge como uma declaração que transmite o que uma marca faz e como ela se difere dos concorrentes [GUSHIKEN 2023]. Com ele é possível apontar as dores, os ganhos e as tarefas atuais do público-alvo da solução e mapear os alívios de dor, os criadores de ganho e os produtos e serviços a serem alcançados e oferecidos pelo projeto. A Figura 3 exibe o canvas desenvolvido para este trabalho.

[1]Também conhecido como *Value Proposition Canvas*; Material derivado do *Business Model Canvas*.

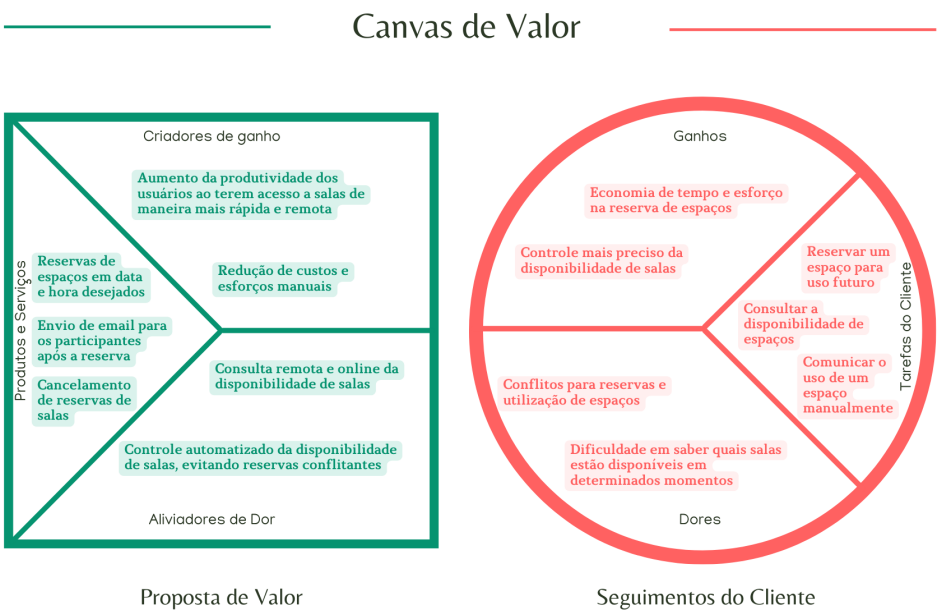

Figura 3. Canvas de Valor elaborado

5.2. Requisitos Funcionais e Não Funcionais

Segundo Sommerville (2018), os requisitos do sistema representam os serviços que o mesmo oferece, bem como sua restrições de operação, ou seja, refletem as necessidades de seus clientes além de seus propósitos de operação. Ele aponta também que este processo de descoberta e análise de requisitos do sistema denomina-se como Engenharia de Requisitos. Os requisitos de um sistema podem ser classificados como funcionais e não funcionais. Ainda segundo Sommerville, os requisitos funcionais representam os serviços que o sistema deve fornecer, o modo como ele deve reagir às entradas e em determinadas situações. Portanto, a Tabela 2 exibe os requisitos funcionais do sistema Mestre Sala.

Tabela 2. Requisitos funcionais do sistema Mestre Sala

Identificação	Nome	Descrição
RF01	Agendar reserva	Permitir que usuários agendem uma reserva no sistema.
RF02	Editar reserva	Permitir que usuários editem informações de sua reserva previamente agendada.
RF03	Excluir reserva	Permitir que usuários excluam uma de suas reservas previamente agendada.
RF04	Detalhar reserva	Exibir detalhes de uma reserva.
RF05	Detalhar sala	Exibir detalhes de uma sala.
RF06	Registrar-se	Permitir que usuários se cadastrem no sistema.
RF07	Efetuar login	Permitir que usuários tenham um perfil de acesso ao sistema.

De acordo com um material publicado no *site* oficial da IBM, em 2021, os casos de uso e os agentes nos diagramas de caso de uso descrevem o que o sistema faz e como

os agentes o utilizam, mas não como o sistema opera internamente [IBM 2021]. Ainda segundo o material, os diagramas de caso de uso ilustram e definem o contexto e os requisitos de um sistema inteiro, descrevendo as funções de alto nível e escopo de uma aplicação. Desta maneira, a Figura 4 representa os casos de uso deste trabalho.

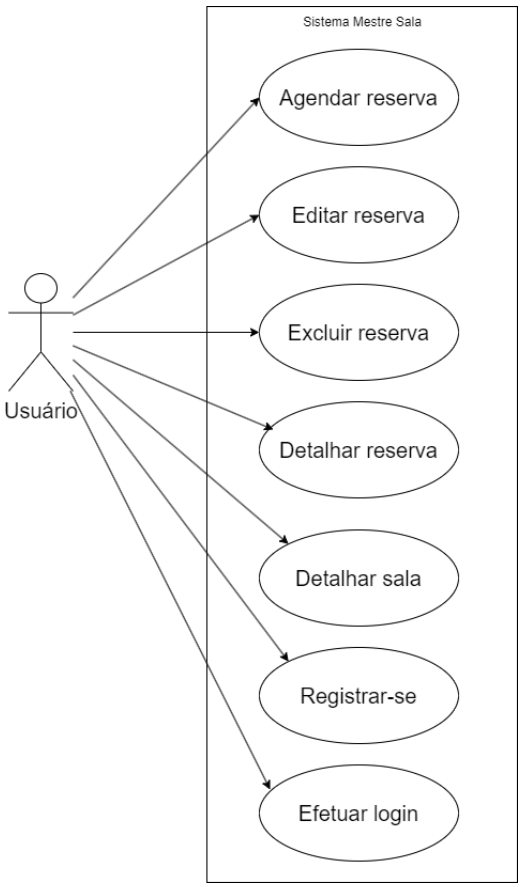

Figura 4. Diagrama de caso de uso elaborado

Após o processo de compreensão dos requisitos do sistema, foi definido e criado um fluxo da visão geral de negócio da aplicação para representar como ele seria utilizado pelos usuários. A Figura 5 demonstra o fluxo de negócio criado.

O fluxo se inicia quando um usuário acessa ao sistema a fim de consultar as datas disponíveis (passo 1) para agendar uma reserva ou visualizar reservas já agendadas por ele ou outros usuários dentro de um dia selecionado. Após agendar sua reserva, o usuário poderá consultar suas reservas agendadas (passo 2) e realizar operações de mantimento, como editar dados de suas reservas ou até mesmo excluí-las. A fim de facilitar sua localização dentro do ambiente em contexto, o usuário poderá consultar as salas disponíveis (passo 3) e visualizar mais detalhes da mesma.

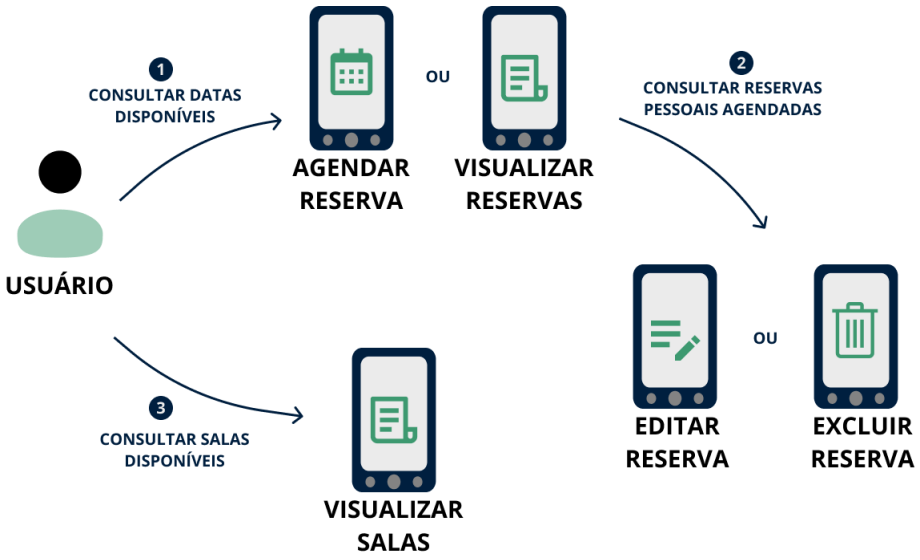

Figura 5. Esquema de Visão de Negócio elaborado

Os requisitos não funcionais são restrições sobre os serviços ou funções oferecidas pelo sistema. Eles incluem restrições de tempo, restrições sobre o processo de desenvolvimento e restrições impostas por padrões, se aplicando ao sistema como um todo [SOMMERVILLE 2018]. Em consideração a isso, a Tabela 3 exibe os requisitos não funcionais do sistema Mestre Sala.

Tabela 3. Requisitos não funcionais do sistema Mestre Sala

Identificação	Nome	Descrição
RNF01	Acesso a Internet	O usuário deve possuir acesso à Internet para utilizar o sistema.
RNF02	Responsividade	Garantir que o sistema se adapte aos diferentes tamanhos e resoluções de tela.
RNF03	Segurança	Garantir que somente um usuário previamente logado no sistema possa realizar o agendamento, edição ou exclusão de uma reserva.

5.3. Prototipação de interfaces

Nesta etapa realizada no desenvolvimento do projeto, foram elaborados protótipos de alta fidelidade das interfaces do sistema, de modo a ilustrar a essência do trabalho desenvolvido, bem como algumas de suas funcionalidades. Esta etapa do desenvolvimento deve ser inequívoca e precisa, a fim de se obter claro conhecimento de como o sistema será desenvolvido [SOMMERVILLE 2018]. Ainda de acordo com ele, esta é uma etapa crucial do desenvolvimento, uma vez que avalia a viabilidade de algumas decisões de projeto e permite que as mudanças necessárias sejam descobertas e tratadas previamente. A Figura 6 exibe os protótipos de interface desenvolvidos para o *front-end* do sistema.

Figura 6. Protótipos das interfaces de login (a) e cadastro (b)

Essas interfaces exibem formulários onde os usuários poderão inserir suas credenciais de acesso e, caso não possuam, a interface de cadastro fornecerá a oportunidade de criar uma nova conta ao fornecerem as informações necessárias.

A Figura 7 exibe a interface da tela principal do sistema, que permitirá aos usuários navegarem através de um calendário para verificar a disponibilidade de reservas para um dia em específico.

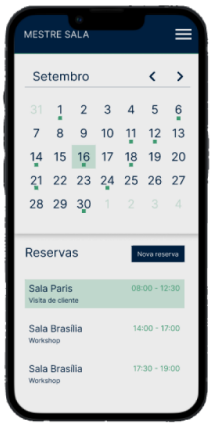

Figura 7. Protótipo da interface principal do sistema

5.4. Histórias de Usuários

Nesta seção, será apresentada uma série de histórias de usuários que foram elaboradas com base nas interfaces projetadas e implementadas no sistema juntamente com os requisitos funcionais levantados. Essas histórias de usuários são cenários fictícios que descrevem as interações e necessidades dos usuários em relação ao sistema, sendo uma declaração informal de requisitos de usuário em vez de um documento de requisitos [LONGO 2014], ajudando a compreender melhor como as interfaces foram projetadas para atender às suas demandas e proporcionar uma experiência satisfatória.

Cada história de usuário destaca um caso de uso específico e demonstra como o sistema pode ser utilizado para atender às necessidades dos usuários em diferentes contextos. Elas descrevem as principais tarefas, recursos e benefícios que os usuários podem obter ao interagirem com as interfaces desenvolvidas.

Ao explorar essas histórias de usuário, teremos uma visão mais aprofundada de como o sistema atende aos requisitos e às demandas dos usuários, além de entender como as interfaces contribuem para uma experiência intuitiva, eficiente e satisfatória.

5.4.1. Cadastro do Usuário

Nesta história são abordados os critérios de aceite que devem ser seguidos pelo usuário para efetuar seu cadastro no sistema. Para melhor construir a história deste cenário, levou-se em consideração que um usuário não cadastrado deseja realizar seu cadastro para que ele possa acessar ao sistema.

A partir destes pontos, os critérios de aceite foram estipulados a seguir, conforme apontados na Figura 8.

1. Um usuário não pode se cadastrar sem preencher todos os campos obrigatórios (nome, e-mail, senha e confirme sua senha).
2. Um usuário não pode se cadastrar caso digite um e-mail inválido.
3. Ao cadastrar-se, o usuário deve ser redirecionado a página de login para realizar seu acesso.

Figura 8. Imagem representando a história de cadastro de usuário

5.4.2. Login do Usuário

Nesta história são abordados os critérios de aceite que devem ser cumpridos pelo usuário para efetuar seu login no sistema e usufruir das funcionalidades oferecidas por ele. Para a construção deste cenário, levou-se em consideração que um usuário já cadastrado gostaria de realizar o login para que possa acessar as funcionalidades do sistema.

A partir destes pontos, os critérios de aceite foram estipulados abaixo, conforme apontados na Figura 9.

1. Um usuário não poderá realizar login sem preencher todos os campos (e-mail e senha).
2. Ao clicar no botão "Entrar", o sistema deve autenticar as credenciais fornecidas e, caso haja sucesso, o usuário deve ser direcionado a página principal do sistema.

Figura 9. Imagem representando a história de login de usuário

5.4.3. Mantimento de reservas

Nesta história são abordados os critérios de aceite para o mantimento de reservas no sistema por parte de seus usuários, bem como as funcionalidades fornecidas e as validações necessárias para sua utilização. Considerou-se, neste cenário, que um usuário, já possuindo acesso ao sistema, deseja realizar a reserva de uma sala, com a possibilidade de editá-la ou excluí-la, caso necessário.

A partir destas considerações, os critérios de aceite foram definidos abaixo para o devido funcionamento desta história:

- Somente um usuário logado poderá acessar esta funcionalidade.
- Uma reserva não poderá ser realizada sem que todos os campos sejam preenchidos.
- O sistema deve impedir que o usuário agende duas reservas conflitantes para a mesma sala.
- O usuário deve ter a opção de editar ou excluir uma reserva existente.

5.5. Visão da arquitetura

O processo de decisão do padrão de arquitetura a ser utilizado por um sistema é no qual se projeta a organização de um sistema que vai satisfazer seus requisitos funcionais e não funcionais [SOMMERVILLE 2018]. Com isso, o padrão de arquitetura deste projeto é exibido pela Figura 10. Nela é possível observar a comunicação entre programas dentro

do sistema Mestre Sala, onde o programa no lado do cliente se comunica com a *API* desenvolvida através de uma requisição *HTTP* (*Hypertext Transfer Protocol*), que a envia ao servidor para o processamento dela. Após isso, o servidor realiza comunicações com o banco de dados do sistema para manipular informações, caso necessário, e retorna-as para o cliente através de uma resposta *HTTP* no formato *JSON*, esquema de escrita de dados citado anteriormente.

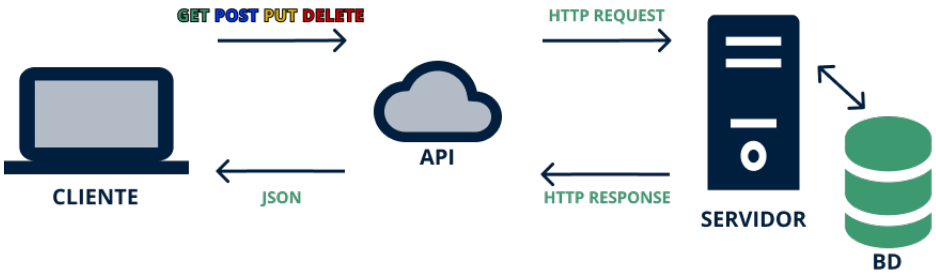

Figura 10. Representação visual da comunicação entre programas do Mestre Sala

5.6. Modelagem da base de dados

Nesta seção é apresentada a modelagem do banco de dados utilizada pela aplicação para a manipulação de dados pelo *back-end* do sistema. Com essas informações é possível compreender melhor o armazenamento de informações do sistema e como as mesmas estão sendo guardadas pela aplicação para utilizações quando necessário. Um diagrama entidade relacionamento (DER) é um tipo de fluxograma que ilustra como "entidades", pessoas, objetos ou conceitos se relacionam entre si dentro de um sistema [Franck et al. 2021]. A fim de facilitar a compreensão, o DER elaborado para o sistema é exibido pela Figura 11.

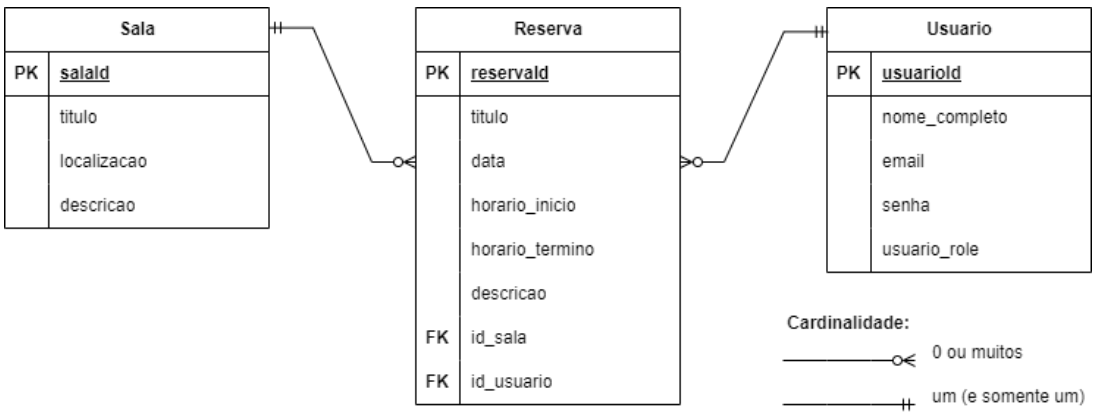

Figura 11. Diagrama de Entidade Relacionamento

Na imagem é possível observar as estruturas das tabelas a serem utilizadas bem como as relações entre elas.

5.7. Desenvolvimento da *API* do sistema

Nesta seção será apresentada a construção da *API* (*Application Programming Interface*) do sistema, que é uma etapa essencial do desenvolvimento. A escolha do *framework Java*

Spring revela a ênfase na robustez, escalabilidade e facilidade de manutenção. O Java Spring oferece uma arquitetura eficiente para o desenvolvimento de *APIs REST* (*Representational State Transfer*), que representa um conjunto interligado de recursos de dados para auxiliar na comunicação de requisição e resposta entre programas dentro de um sistema no processo de captura de informações [Masse 2011].

Para a autenticação do usuário e proteção de rotas específicas, foi utilizando, em conjunto, o *framework Spring Security 6*, a fim de controlar o acesso de usuários às páginas e rotas do sistema. Além disso, esse mesmo *framework* foi utilizado para o controle de autenticação de login e cadastro de novos usuários, gerando *tokens* do tipo *Bearer JWT* (*token* do tipo opaco, ou seja, não possui informações autocontidas) para serem utilizados como chaves de autorização de acesso às rotas para a captura de informações pelo lado do cliente do sistema.

Com a finalidade de facilitar a compreensão e entendimento das rotas disponíveis no sistema quanto ao seus funcionamentos, o *framework Swagger* foi utilizado para tal documentação. O *Swagger* especifica a lista de recursos disponíveis na *API REST* e as operações que podem ser chamadas nesses recursos. O documento *Swagger* também especifica a lista de parâmetros para uma operação, incluindo o nome e o tipo dos parâmetros, se os parâmetros são obrigatórios ou opcionais e informações sobre valores aceitáveis para serem inseridos como parâmetros da *API REST* em contexto [Surwase 2016]. O *Swagger* oferece consigo o *Swagger UI* uma interface de usuário que permite explorar e interagir com a documentação da *API* de uma maneira amigável, facilitando a visualização das rotas, a execução de chamadas e a compreensão dos resultados.

Abaixo, a Figura 12 ilustra a interface do *Swagger UI* com seu mapeamento de endereços da *API REST* utilizada pelo sistema.

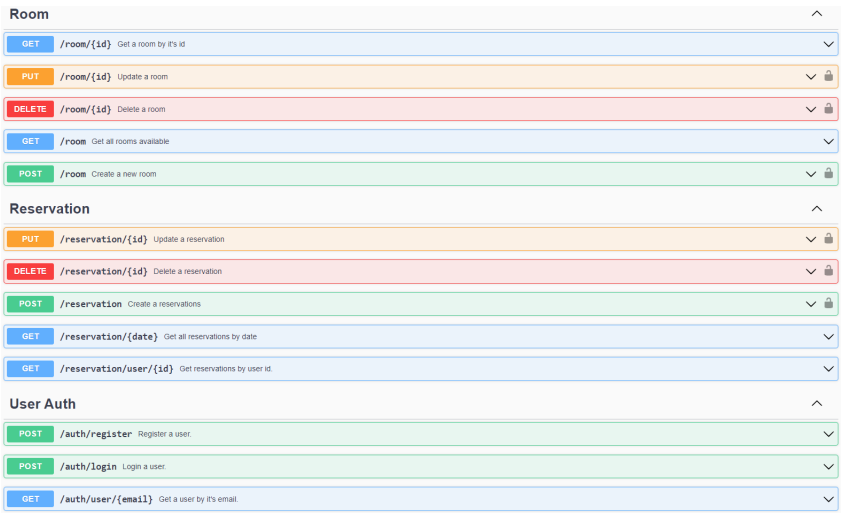

Figura 12. Ferramenta *Swagger UI* sendo usada para documentação de rotas da *API* do sistema

Portanto, com sua capacidade de mapear *endpoints*[2], métodos, parâmetros e modelos de dados, o *Swagger* se torna uma peça fundamental para a construção, manutenção

[2]Um *endpoint*, em português "Ponto de extremidade", representa uma extremidade de conexão de uma *API* com o sistema do lado do cliente, permitindo a interconexão e o compartilhamento de informações

e uso eficiente da *API*, contribuindo para uma experiência de desenvolvimento e integração facilitada e bem documentada.

5.8. Desenvolvimento da aplicação

Após a etapa de desenvolvimento da *API* do sistema, foi iniciado o processo de desenvolvimento e integração da mesma com as interfaces reais da aplicação construídas em *React.js*, *framework* citado anteriormente.

5.8.1. Páginas de Login e Cadastro

Ao inicializar a aplicação, é possível visualizar a tela inicial para autenticação no sistema, na qual o usuário fornece seu e-mail e senha para realizar o acesso. Caso os dados fornecidos reflitam em um usuário já cadastrado no sistema, o usuário é direcionado à página principal da aplicação. Caso o usuário não esteja previamente cadastrado no sistema, ele tem a possibilidade de clicar no texto para realizar um novo cadastro, fornecendo as informações necessárias para a criação de sua conta. O processo de cadastro permite a inclusão de novos usuários no sistema, ampliando a base de usuários e possibilitando a utilização das funcionalidades disponíveis através de sua autenticação de login, que garante a segurança e a autenticação dos usuários, fornecendo-lhes acesso a plataforma. Ambas as interfaces construídas e implementadas podem ser observadas na Figura 13.

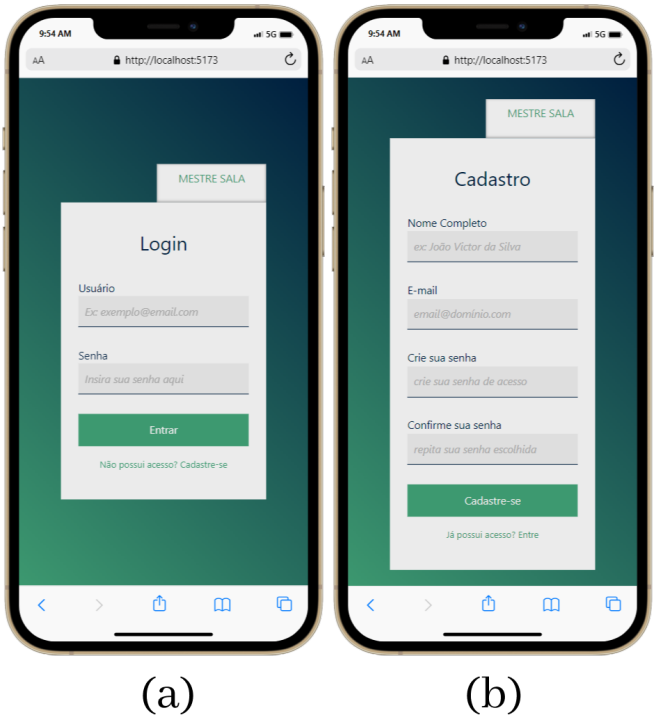

Figura 13. Interfaces de login (a) e cadastro (b) implementadas

entre eles.

5.8.2. Registro e visualização de reservas

Na interface principal do sistema, após o usuário realizar login na aplicação, é apresentado a ele um calendário que exibe os dias do mês atual (Figura 14a), permitindo também a navegação entre os meses. Através dessa visualização, o usuário pode identificar de forma rápida e clara os dias disponíveis para reservas e os ocupados com reservas já realizadas.

Abaixo do calendário, encontra-se uma listagem das reservas realizadas para o dia selecionado. Essa listagem fornece informações relevantes sobre as reservas, como horário, sala reservada e o título da reserva. Dessa forma, o usuário pode ter uma visão geral das atividades agendadas para o dia em questão.

Além disso, a Figura 14c apresenta a interface de visualização de detalhes de uma reserva na listagem. Nessa visualização, o usuário pode obter informações mais detalhadas sobre uma reserva específica, como quem a realizou, horário, sala, título e sua descrição, contendo sua finalidade. Essa funcionalidade permite que o usuário tenha acesso rápido e fácil aos detalhes importantes de uma reserva específica.

Por fim, um botão de 'Nova Reserva' está localizado abaixo do calendário, permitindo ao usuário acessar a interface de criação de uma nova reserva para o dia selecionado (Figura 14b). Nessa interface, são disponibilizados campos intuitivos para inserção dos detalhes da reserva, como sala a ser reservada, horário, título da reserva e finalidade.

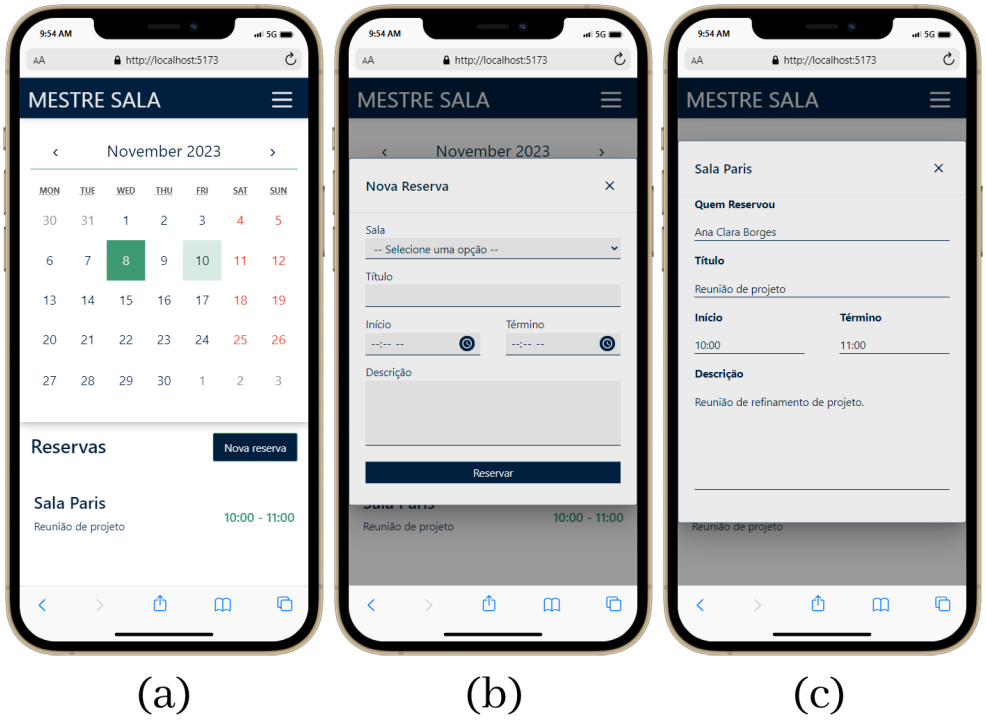

Figura 14. Interfaces de calendário (a), de realização de nova reserva (b) e de visualização de uma reserva já agendada (c) mencionadas

5.8.3. Visualização, edição e exclusão de reservas do usuário

Outra interface implementada é a de reservas já realizadas pelo usuário no sistema. Nessa tela o usuário tem acesso a uma lista completa de todas as suas reservas realizadas no sistema (15). Essa lista fornece informações relevantes sobre cada reserva, como data, horário, sala e título (Figura 15a).

O sistema também permite a edição e a exclusão dessas reservas anteriormente realizadas. Na tela exibida na Figura 15b, o usuário tem a opção de editar os detalhes de uma reserva existente, como alterar a data, horário ou título. Também é possível excluir uma reserva caso não seja mais necessária. Essa funcionalidade dá ao usuário o controle total sobre suas reservas, permitindo que ele faça ajustes ou remova reservas conforme necessário.

Essas interfaces fornecem ao usuário a capacidade de gerenciar suas próprias reservas de forma conveniente, com a possibilidade de editar e excluir reservas realizadas. A inclusão dessas funcionalidades contribui para uma experiência de usuário mais completa e satisfatória, garantindo que os usuários possam manter suas reservas atualizadas e de acordo com suas necessidades.

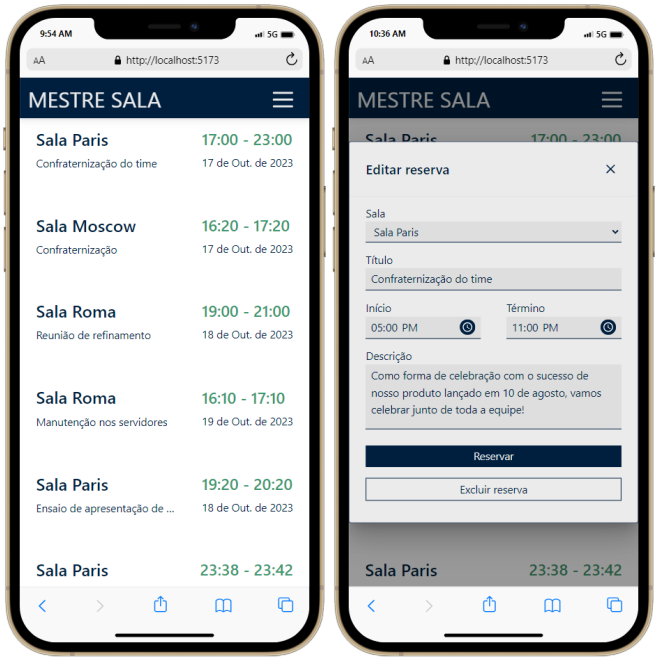

Figura 15. Interfaces de visualização (a) e edição ou exclusão (b) de reservas por parte de um único usuário

5.8.4. Visualização de salas disponíveis

Por fim, outra interface implementada é a de visualização de salas que podem ser reservadas. O intuito dessa página é fornecer uma lista com mais informações das salas físicas presentes no ambiente em contexto, como suas respectivas localizações e descritivos so-

bre sua estrutura (Figura 16a). Ao selecionar uma sala desejada na listagem exibida pela Figura 16a o usuário pode obter seus detalhes conforme exibido na Figura 16b.

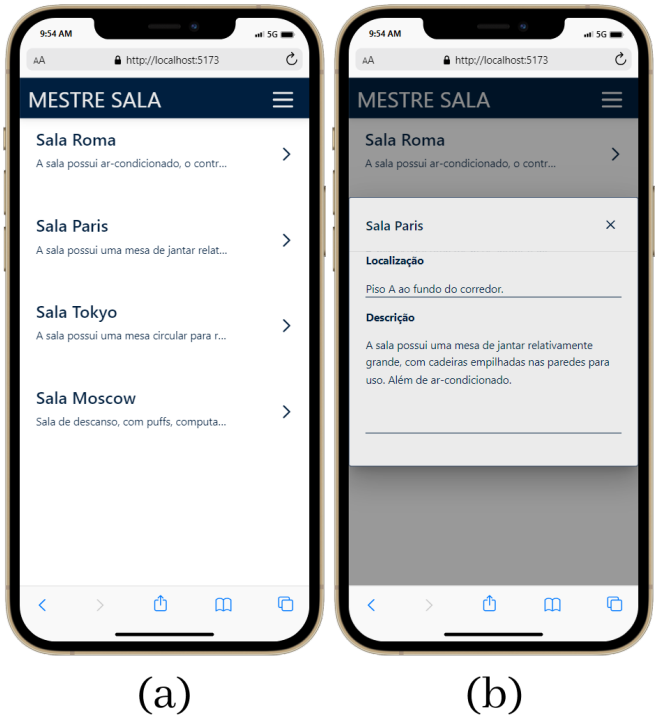

Figura 16. Interfaces de listagem (a) e visualização (b) de salas disponíveis

5.9. Testes Unitários

De acordo com Sommerville (2018), os testes pretendem mostrar que um programa faz o que foi destinado a fazer e descobrir os defeitos antes que ele seja colocado em uso. Com isso, testes unitários foram implementados dentro da *API* do sistema Mestre Sala, a fim de validar entradas e saídas e garantir o funcionamento adequado da solução. Atualmente, os testes de unidade implementados garantem uma cobertura de 30% sob as funcionalidades da *API* do sistema. Para a construção das mesmas, o *framework Mockito*, do qual seu principal objetivo é instanciar classes e controlar o comportamento dos métodos [CANDIOLLI 2020], foi utilizado para simular retornos de funções implementadas, juntamente com o *JUnit5*, do qual é responsável por fornecer um ambiente de tempo de execução para os testes [JUNIOR 2023].

Um trecho de código de um dos testes de unidade implementados pode ser visualizado na Figura 17. Nela, observa-se um método de teste para validar se uma lista de salas pode ser retornada pelo serviço da *API* do sistema. Na linha 46 observa-se o uso dos métodos *when* e *thenReturn* do *framework Mockito*, que simulam o comportamento quando o método *findAll* do repositório de salas do sistema for chamado, onde ele deve retornar uma lista contendo ao menos uma sala disponível. As linhas 48 e 49 são responsáveis por garantir que, ao capturar uma lista de salas disponíveis no sistema, a mesma possua um tamanho maior que zero, ou seja, que não seja vazia.

```java
@Test
void shouldBeAbleToGetListOfRooms() {
    when(roomRepository.findAll()).thenReturn(List.of(testRoom));

    List<Room> roomsList = roomService.getAllRooms();
    assertThat(roomsList.size(), greaterThan( value: 0));
}
```

Figura 17. Teste de unidade implementado para a *API* do Mestre Sala

5.10. Testes de funcionalidade

Além dos testes unitários, testes de funcionalidades foram realizados durante o processo de desenvolvimento do sistema. O teste de funcionalidade tem como função avaliar as funções do sistema observando se estão funcionando corretamente, além de checar se o mesmo está adequado aos objetivos do negócio [CARVALHO 2019].

5.11. Testes de *API*

Teste de *API* é um tipo de teste de *software* que se concentra em determinar se a *API* desenvolvida vai de encontro às expectativas relativas à funcionalidade, confiabilidade, performance e segurança da aplicação[BENJAMIN 2018].

Para a realização desses testes, a ferramenta *Insomnia* foi utilizada, sendo um *framework Open Source* destinado à execução deste tipo de teste. Seu diferencial é que, além de um *layout* limpo e fácil de usar, é possível utilizar variáveis e funções para facilitar a manutenção de endereços [RIBEIRO 2020]. A Figura 18 exibe a interface do *Insomnia* contendo o ambiente utilizado para a execução dos testes. Através dela observa-se à esquerda uma coluna contendo as rotas do sistema Mestre Sala. Ao selecionar uma rota desta coluna, a coluna central pode ser utilizada para a personalização do corpo de uma chamada à essa rota, e após a sua execução, seus resultados e retornos podem ser visualizados na coluna da direita da interface.

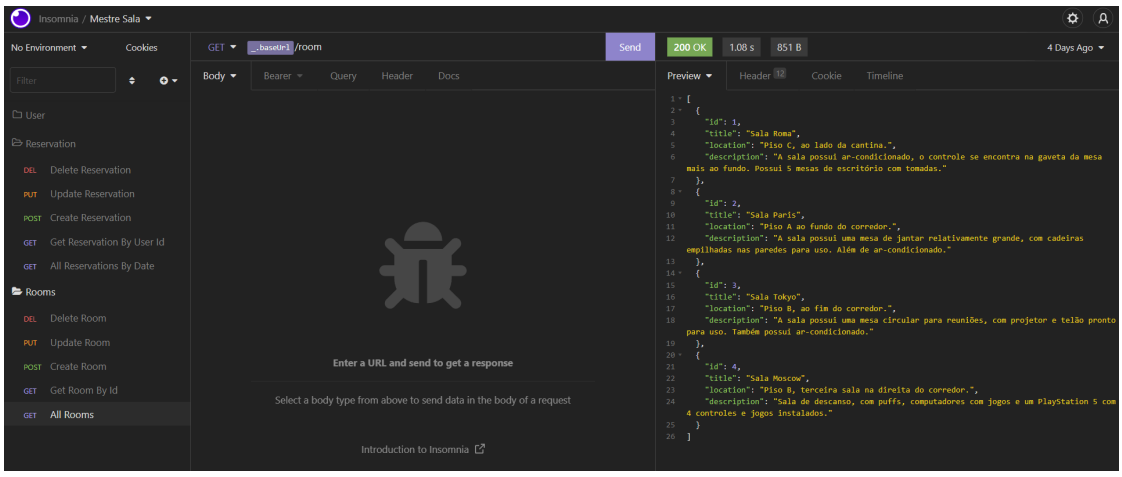

Figura 18. Ambiente de testes funcionais na ferramenta *Insomnia*

5.12. Artefatos e código fonte

Com a finalidade de tornar acessível o projeto desenvolvido, toda a aplicação implementada e seu conteúdo, incluindo o código fonte, foram disponibilizados no serviço em nuvem GitHub, onde estão armazenados o *front-end* do projeto na url https://github.com/RafaSilvaDev/Mestre-Sala-FE.git e o *back-end* na url https://github.com/RafaSilvaDev/Mestre-Sala-BE.git respectivamente.

6. Conclusão

Através deste trabalho, foi apresentado o desenvolvimento do Mestre Sala, um sistema *web* projetado para otimizar o gerenciamento de reservas e disponibilidade de salas em uma variedade de ambientes, incluindo corporativos e residenciais. Este projeto reúne o uso de tecnologias modernas, como o *framework* React.js no *front-end* e o *framework* Spring no *back-end*.

Espera-se que este projeto tenha uma contribuição significativa para a otimização do gerenciamento de espaços físicos em ambientes diversos. O sistema foi capaz de demonstrar como a combinação de tecnologias *web* modernas e boas práticas de desenvolvimento podem resultar em uma possível solução para os desafios enfrentados por organizações que precisam gerenciar salas e espaços.

Assim, este trabalho foi parcialmente desenvolvido, com foco em atender seu objetivo através do desenvolvimento de suas principais funcionalidades, como agendamento e gerenciamento de reservas por parte do usuário, além de conter sistemas de login e cadastro funcionais.

Como trabalhos futuros, funcionalidades relacionadas ao gerenciamento de informações ainda podem ser implementadas, como a de edição do perfil por parte do usuário e de gerenciamento de ambientes através de uma interface administrativa, além da implementação de testes de ponta a ponta dentro da interface desenvolvida, a fim de garantir a qualidade do projeto.

Com o desenvolvimento desse trabalho, foi possível aplicar diversos conhecimentos adquiridos em várias disciplinas ao decorrer do Curso Superior de Análise e Desenvolvimento de Sistemas. Dentre elas, estão: Arquitetura de *Software*; Análise Orientada a Objetos; Engenharia de *Software*; Banco de Dados I e II e Desenvolvimento *Web*. Ademais, foram adquiridos conhecimentos relacionados a construções de interfaces *Web* com a utilização do *framework* React.js.

Referências

[BENJAMIN 2018] BENJAMIN (2018). Top 10 de ferramentas de teste de apis para 2018. Disponível em: https://tinyurl.com/jdt26dp2. Acesso em: 29 nov. 2023.

[CANDIOLLI 2020] CANDIOLLI, S. (2020). Testando seu código java com o mockito framework. Disponível em: https://tinyurl.com/mwscn3ma. Acesso em: 20 nov. 2023.

[CARVALHO 2019] CARVALHO, I. (2019). Teste de funcionalidade, teste de desempenho e teste de usabilidade (part 1). Disponível em: https://tinyurl.com/2v5hn9nb. Acesso em: 24 nov. 2023.

[Franck et al. 2021] Franck, K. M., Pereira, R. F., and Dantas Filho, J. V. (2021). Ratio-entity diagram: a tool for conceptual data modeling in software engineering. *Research, Society and Development*, 10(8):e49510817776. Acesso em: 06 nov. 2023.

[GROUP 2019] GROUP, I. W. (2019). Global workspace survey. Disponível em: https://www.iwgplc.com/global-workspace-survey-2019. Acesso em: 05 abr. 2023.

[GUSHIKEN 2023] GUSHIKEN, A. (2023). Value proposition canvas: o que é e como funciona essa metodologia? Disponível em: https://g4educacao.com/portal/value-proposition-canvas. Acesso em: 06 nov. 2023.

[IBM 2021] IBM (2021). Diagramas de caso de uso. Disponível em: https://www.ibm.com/docs/pt-br/rsm/7.5.0?topic=diagrams-use-case. Acesso em: 19 nov. 2023.

[INFRASPEAK 2022] INFRASPEAK, T. (2022). O que é a gestão do espaço de trabalho (para principiantes). Disponível em: https://blog.infraspeak.com/pt-br/gestao-do-espaco-de-trabalho/. Acesso em: 09 nov. 2023.

[INFRASPEAK 2023] INFRASPEAK, T. (2023). What is facility management? Disponível em: https://blog.infraspeak.com/what-is-facility-management/. Acesso em: 16 nov. 2023.

[JUNIOR 2023] JUNIOR, G. B. (2023). Garanta a qualidade do seu código java com junit: Uma visão geral da ferramenta essencial para desenvolvedores. Disponível em: https://tinyurl.com/y5skjbcy. Acesso em: 20 nov. 2023.

[LONGO 2014] LONGO, H. E. R. (2014). A utilização de histórias de usuários no levantamento de requisitos ágeis para o desenvolvimento de software. *International Journal of Knowledge Engineering and Management*, 3(6):1–30. Acesso em: 17 nov. 2023.

[MANE et al. 2013] MANE, D., Ojha, N., and Chitnis, K. (2013). The spring framework: An open source java platform for developing robust java applications. *International Journal of Innovative Technology and Exploring Engineering*, 3(2). Acesso em: 16 nov. 2023.

[Masse 2011] Masse, M. (2011). *REST API design rulebook: designing consistent RESTful web service interfaces*. "O'Reilly Media, Inc.". Acesso em: 16 nov. 2023.

[O'GRADY 2021] O'GRADY, S. (2021). The redmonk programming language rankings: January 2021. Disponível em: https://redmonk.com/sogrady/2021/08/05/language-rankings-6-21/. Acesso em: 26 abr. 2023.

[OLIVEIRA 2019] OLIVEIRA, R. R. (2019). *Desenvolvimento de sistemas web com PHP e MySQL*. Novatec. Acesso em: 26 abr. 2023.

[PEZOA et al. 2016] PEZOA, F., Reutter, J. L., Suarez, F., Ugarte, M., and Vrgoč, D. (2016). Foundations of json schema. Disponível em: https://dl.acm.org/doi/abs/10.1145/2872427.2883029. Acesso em: 16 nov. 2023.

[POKORNÁ et al. 2015] POKORNÁ, J., Pilař, L., Balcarová, T., and Sergeeva, I. (2015). Value proposition canvas: identification of pains, gains and customer jobs at farmers' markets. *AGRIS on-line Papers in Economics and Informatics*, 7(665-2016-45080):123–130. Acesso em: 17 nov. 2023.

[REBELLABS 2020] REBELLABS (2020). Developer productivity report 2020: The java ecosystem in 2020. Disponível em: https://zeroturnaround.com/rebellabs/java-ecosystem-2020-report/. Acesso em: 26 abr. 2023.

[RIBEIRO 2020] RIBEIRO, L. (2020). Insomnia, um poderoso testador de rotas. Disponível em: https://lucassr.medium.com/insomnia-um-poderoso-testador-de-rotas-3d77d2cd8e89. Acesso em: 24 nov. 2023.

[SEBRAE 2023] SEBRAE (2023). A taxa de sobrevivência das empresas no brasil. Disponível em: https://sebrae.com.br/sites/PortalSebrae/artigos/a-taxa-de-sobrevivencia-das-empresas-no-brasil,d5147a3a415f5810VgnVCM1000001b00320aRCRD. Acesso em: 11 mar. 2023.

[SOMMERVILLE 2018] SOMMERVILLE, I. (2018). *Emgenharia de Software*. Pearson, 10th edition. Acesso em: 16 nov. 2023.

[Surwase 2016] Surwase, V. (2016). Rest api modeling languages-a developer's perspective. *Int. J. Sci. Technol. Eng*, 2(10):634–637. Acesso em: 06 nov. 2023.

[TIDWELL 2006] TIDWELL, J. (2006). *Design de Interfaces: Padrões para Web*. O'Reilly Media, 1st edition. Acesso em: 26 abr. 2023.

[WARFELL 2009] WARFELL, T. Z. (2009). Prototyping: A practitioner's guide. Disponível em: https://rosenfeldmedia.com/books/prototyping/. Acesso em: 26 abr. 2023.

Documento Digitalizado Público

Artigo de TCC - Versão Final

Assunto: Artigo de TCC - Versão Final
Assinado por: Fernando Sambinelli
Tipo do Documento: Formulário
Situação: Finalizado
Nível de Acesso: Público
Tipo do Conferência: Documento Digital

Documento assinado eletronicamente por:

- **Fernando Sambinelli**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 19/12/2023 20:41:05.

Este documento foi armazenado no SUAP em 19/12/2023. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1530183
Código de Autenticação: 6f2655bea1