Evento Fácil: plataforma digital para locação de espaços para eventos com implementação de microsserviços e armazenamento escalável

Pedro Henrique Cavalcante Collin[1], Michele Cristiani Barion[2]

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas
Instituto Federal de Educação, Ciência e Tecnologia de São Paulo – IFSP
CEP 13183-250 – São Paulo – SP – Brazil

pedro.collin@aluno.ifsp.edu.br[1], michele_barion@hotmail.com[2]

Abstract. *The project propose to develop a digital platform to optimize the process of renting venues for events. Currently, this process can be complex and time consuming, involving several steps, such as searching for available spaces, contacting owners, negotiating prices and conditions, and ensuring the security of the transaction. To overcome these challenges, it was proposed to create a preview page that will simplify and improve the entire rental process. This page has an interface that allows users to find available venues for specific events, such as parties, conferences, weddings, and more. The application is built using Spring to assemble the backend Application-Programming Interface (API), Angular for the frontend, Scrum as the development methodology, and Postman to test API calls. Additionally, the API is documented using Swagger. Effective integration between the backend and frontend, combined with the implementation of Amazon S3 for image storage, contributed to optimizing application performance and data security.*

Resumo. *O projeto propõe desenvolver uma plataforma digital para otimizar o processo de aluguel de locais para eventos. Atualmente, esse processo pode ser complexo e demorado, envolvendo diversas etapas, como pesquisar espaços disponíveis, entrar em contato com os proprietários, negociar preços e condições, além de garantir a segurança da transação. Para superar esses desafios, foi proposta a criação de uma página de visualização que irá simplificar e melhorar todo o processo de aluguel. Esta página possui uma interface que permite que o usuário encontre locais disponíveis para eventos específicos, como festas, conferências, casamentos e muito mais. A aplicação é construída usando o Spring para montar a Interface de Programação de Aplicação (API) do backend, Angular para o frontend, Scrum como metodologia de desenvolvimento e Postman para testar as chamadas da API. Além disso, a API está documentada com o Swagger. A integração eficaz entre o backend e o frontend, aliada à implementação do Amazon S3 para o armazenamento de imagens, contribuiu para a otimização do desempenho da aplicação e a segurança dos dados.*

1. Introdução

Em uma sociedade cada vez mais adepta à tecnologia e digitalização, torna-se importante compreender de que forma os sistemas de informação podem ser utilizados visando a administrar fatores internos e externos que permeiam uma empresa, independente do segmento em que ela atua.

Segundo os autores [Laudon e Laundon 2011] esses sistemas são definidos como, “um conjunto de componentes inter-relacionados que coleta (ou recupera), processa, armazena e distribui informações destinadas a apoiar a tomada de decisões, a coordenação e o controle de uma organização". Enfatizam, também, que as principais atividades de um sistema são a entrada (receber dados internos e externos associados à organização que está usando a aplicação), o processamento (converter os dados e/ou fazer comparações) e a saída (transferir informações processadas para possíveis tomadas de decisões), além de se ter um mecanismo de realimentação a fim de atingir um objetivo (uma saída que se transforma em uma nova entrada).

Sabe-se que um sistema pode ser acessado por diferentes plataformas, tais como por dispositivos móveis, como também *desktop* e que seu manuseio se faz por uma instalação ou pela conexão com o uso da internet através de um *site*. Independente de como será a interação com a aplicação, uma das formas que as empresas usam para coletar, organizar e acessar os dados que são gerados pelos sistemas é através de um banco de dados relacional.

Todavia, conforme aborda [Batista 2006], as aplicações de acordo com as informações, podem ser classificadas em dois tipos, sendo:

- No nível operacional: se faz por meio das operações diárias e que são adquiridas pelos componentes de controle interno;

- No nível gerencial: são usadas para tomadas de decisões, visando o processo de planejamento, controle e possíveis interpretações de resultados.

É nesse viés entre a importância de um sistema de informação e os seus processos atrelados entre pessoas e tecnologias que este projeto surge por meio de uma aplicação voltada ao segmento de eventos. Assim, o objetivo geral é o desenvolvimento de um *software* que visa digitalizar e simplificar o comércio de locação de ambientes para a realização de eventos.

Para garantir uma interface *web* disponível vinte e quatro horas por dia, sete dias por semana (expressão conhecida como "24/7"), amigável e de fácil interação com os usuários, serão utilizadas metodologias como extração e análise de requisitos e definição do gerenciador de projetos Scrum, bem como tecnologias adequadas para o desenvolvimento. A aplicação será hospedada em um servidor da nuvem para que todos possam ter acesso *online*.

Para abordagem dos objetivos expostos, além desta introdução, o presente trabalho está organizado em sete seções ordenadas em referencial teórico, trabalhos correlatos, metodologia, desenvolvimento, conclusão, trabalhos futuros e relação das referências utilizadas tanto na pesquisa sobre as metodologias adotadas quanto das ferramentas que foram fundamentais para o desenvolvimento.

2. Referencial Teórico

Esta seção aborda a pesquisa bibliográfica realizada para gerar o embasamento teórico que traz a fundamentação técnica diante dos conceitos e das tecnologias aplicadas.

2.1. Levantamento de requisitos

Conforme mencionado por [Sommerville 2015], os requisitos de um sistema são representações dos serviços e das restrições que regem sua operação. Esses requisitos são

baseados nas necessidades dos clientes para um sistema específico, que pode ter finalidades como controle de dispositivos, realização de pedidos ou busca por informações. Existem várias técnicas disponíveis para coletar esses requisitos junto aos *stakeholders*, sendo as mais comuns questionários e entrevistas.

Enquanto os requisitos funcionais descrevem as funcionalidades que o sistema deve ser capaz de executar, os requisitos não funcionais estabelecem restrições nos serviços ou funções oferecidas pelo sistema. Essas restrições podem envolver aspectos como limitações de tempo no processo de desenvolvimento ou exigências impostas por normas regulatórias [Sommerville 2015].

2.2. Arquitetura de Microsserviços

A arquitetura de microsserviços será adotada no projeto, exigindo uma abordagem diferente da tradicional arquitetura monolítica. Para embasar esse conceito, as seguintes referências são relevantes:

- De acordo com [Newman 2015] "os *micro-services* são serviços pequenos e autônomos que trabalham juntos". Essa afirmação ressalta a abordagem arquitetural que divide uma aplicação em pequenos serviços independentes, cada um com sua própria lógica de negócio e banco de dados. A adoção dessa abordagem traz benefícios como escalabilidade (os serviços são ampliados ou reduzidos de forma elástica quando são necessários, sem exigir aquisição de *hardware* com alto custo), flexibilidade (os serviços são projetados de forma independente uns dos outros, reduzindo problemas e complexidade) e isolamento de falhas (uma falha de um serviço não afeta os demais, além da total capacidade de programação das *APIs* que permite os serviços se comunicarem e lidarem com falhas de forma muito mais eficiente)[1].

- A arquitetura de microsserviços também auxilia a lidar com os desafios enfrentados pela complexidade em uma grande base de código, que é fruto de uma aplicação de gerenciamento como a que será desenvolvida nesse trabalho. Conforme afirmado por [Carnell 2017], "os microsserviços ajudam a combater os problemas tradicionais de complexidade em uma grande base de código, decompondo a grande base de código em pedaços pequenos e bem definidos". Essa abordagem envolve a decomposição da base de código em microsserviços independentes, cada um com sua própria lógica de negócio e responsabilidades bem definidas.

2.3. *Frameworks* de desenvolvimento *web*

Serão utilizados dois *frameworks* de desenvolvimento *web* no projeto, sendo o Angular e o Spring Boot. Para compreender melhor esses *frameworks* e suas aplicações, as seguintes referências são abordadas:

- O AngularJS tem sido amplamente utilizado como uma abordagem de desenvolvimento de aplicações *web*, fornecendo suporte e recursos valiosos para auxiliar no desenvolvimento. Conforme enfatizado por [Schmitz e Lira 2016], "ele funciona como uma extensão ao documento HTML, adicionando novos parâmetros e interagindo de forma dinâmica com vários elementos". Sendo assim, podem ser adicionados novos atributos no

[1] Conceitos extraídos através do estudo realizado pelo site https://www.juniper.net/br/pt/research-topics/what-is-a-cloud-microservice.html.

HTML para conseguir adicionar funcionalidades extras, sem a necessidade da programação em JavaScript. Essa característica única do AngularJS oferece uma maneira eficiente de aprimorar a funcionalidade e a experiência do usuário, permitindo a inclusão de recursos adicionais por meio da definição de atributos personalizados.

- Já a utilização do Spring é bastante eficaz no desenvolvimento de aplicações Java. Conforme é destacado pelos autores [Junior e Afonso 2017], "o Spring não é um *framework* apenas, mas um conjunto de projetos que resolvem várias situações do cotidiano de um programador, ajudando a criar aplicações Java com simplicidade e flexibilidade". Essa afirmação ressalta a natureza abrangente do Spring, que oferece soluções para diversos desafios encontrados no desenvolvimento de *software*.

2.4. UX/UI e *design* de interfaces

A interface do projeto será projetada com foco na experiência do usuário. Para auxiliar nesse processo, será utilizada a estratégia de UX/UI(User experience and User interface) que ajuda na construção e *design* de interface.

Na estratégia de UX pode ser definida como o "panorama geral" do *design* de UX, como mencionado por [Levy 2015]. Ela vai além dos aspectos superficiais do *design* e se concentra em entender profundamente as necessidades e expectativas dos usuários, bem como os objetivos do negócio. A estratégia de UX envolve a definição de metas claras, a identificação dos principais desafios e a criação de um plano para alcançar uma experiência de usuário desejada. É um processo iterativo que guia a tomada de decisões e a alocação de recursos ao longo do ciclo de desenvolvimento do *software*.

2.5. Metodologias ágeis, gerenciamento de projetos e tecnologias

No desenvolvimento do projeto serão utilizadas as metodologias ágeis e um *software* de gerenciamento de projetos. Logo, é possível abordar conceitos de metodologias ágeis, como o Scrum, e suas vantagens em relação a metodologias tradicionais, bem como técnicas de gerenciamento de projetos, como o Kanban, que podem auxiliar no planejamento e execução do projeto. Para embasar esse aspecto, as seguintes referências são relevantes:

- O autor [Sutherland 2014] explora de forma abrangente os princípios e práticas do Scrum. Ele destaca que o Scrum "não é apenas uma metodologia", porém uma abordagem que visa acelerar o esforço humano em qualquer tipo de trabalho. O Scrum incentiva a colaboração, a transparência e o foco em metas claras. Ele também apresenta que o Scrum oferece uma estrutura que permite que as equipes entreguem resultados de alta qualidade em um ritmo mais rápido. “O Scrum acelera o esforço humano não importa qual seja esse esforço", destaca a capacidade do Scrum de impulsionar a produtividade e a eficiência das equipes de desenvolvimento de *software*. Ao adotar a abordagem Scrum, as equipes são capacitadas a auto-gerenciar seus projetos, priorizar tarefas, colaborar de forma mais eficaz e lidar com mudanças de forma ágil. Essa capacidade de acelerar o esforço humano é fundamental para alcançar resultados significativos no desenvolvimento de *software*, permitindo que as equipes entreguem mais valor em menos tempo.

- Combinando Scrum e Kanban para uma abordagem eficaz, uma das frases-chave do livro de [Kniberg e Skarin 2010], é quando eles comentam que "Scrum e Kanban são ferramentas de processo que, em certa medida, ajudam a trabalhar de maneira mais eficaz, dizendo que fazer" e, assim, destacando a complementaridade entre essas duas abordagens.

Neste contexto, enquanto o Scrum define um conjunto de papéis, cerimônias e artefatos para gerenciar o projeto de forma iterativa, o Kanban oferece uma abordagem visual para otimizar o fluxo de trabalho, identificar gargalos e promover a melhoria contínua.

Diante do exposto, essas referências fornecem uma base sólida para a compreensão das metodologias ágeis, como o Scrum, e técnicas específicas de gerenciamento de projetos, como o Kanban, que podem auxiliar no planejamento e execução eficaz do projeto.

Além das metodologias ágeis, foram usadas outras tecnologias para desenvolvimento do projeto, como a linguagem de programação escolhida para o backend, que foi o Java.

A linguagem Java é amplamente reconhecida como uma das linguagens mais importantes para programação de uso geral. Isso é comprovado pela citação do livro do autor [Hortsmann 2004], o qual afirma que o "Java está estabelecido como uma das linguagens mais importantes para programação de uso geral e para o ensino". Algumas justificativas para essa importância incluem a versatilidade, portabilidade e um ecossistema robusto que será crucial para o desenvolvimento do projeto.

Para armazenar os dados da aplicação será utilizado um Sistema Gerenciador de Banco de Dados (SGBD) que servirá para gerenciar o *dataset*, sendo escolhido para o projeto o PostgreSQL, que é um SGBD de código aberto e gratuito que é amplamente reconhecido por sua confiabilidade e robustez. Algumas justificativas para a escolha do PostgreSQL como SGBD incluem a sua alta integridade dos dados, recuperação de falhas e escalabilidade, além de seus recursos avançados [PostgreSQL 2023]. Enfatiza-se que nesta proposta, o modelo relacional se faz diante do armazenamento dos dados, no entanto todo o processo de acesso e manipulação de dados se faz pelo Spring Data que é um *framework* destinado para integrar o *backend* com um *database*.

3. Trabalhos Correlatos

Nesta seção serão apresentadas duas aplicações desenvolvidas na plataforma *web*, sendo abordadas as suas principais características, conforme o estudo feito entre elas.

3.1. Sistema web para gerenciamento de eventos e emissão de documentos - SKED

A autora [Caetano 2019] traz como objetivo o desenvolvimento de um sistema *web* que facilite o gerenciamento de eventos, incluindo o controle de participantes e a emissão de documentos. Embora o foco seja um pouco diferente, ambos os trabalhos compartilham a ideia de oferecer uma solução digital para otimizar o processo relacionado a eventos.

Diferentemente do “Evento Fácil” que traz um foco mais voltado ao aluguel de espaços para eventos na prática, o SKED traz uma visão mais voltada para a documentação do processo.

3.2. e-Vent-Br: proposta de um sistema web de gerenciamento de eventos

Os autores [Carvalho *et al.* 2014] se diferenciam por sua proposta de um sistema *web* voltado para o gerenciamento abrangente de eventos.

Enquanto o "Evento Fácil" tem um foco específico na locação de espaços para eventos, essa aplicação abrange uma perspectiva mais ampla, oferecendo recursos para o gerenciamento completo de eventos, como controle de participantes, emissão de documentos e outras funcionalidades relacionadas.

A primeira aplicação se concentra principalmente na facilidade do processo de

locação de espaços, fornecendo recursos de ranqueamento e filtros personalizados para encontrar espaços adequados às necessidades dos usuários.

Aspecto	Evento Fácil	SKED	e-Vent-Br
Foco	Aluguel de locais para eventos	Gerenciamento de eventos e documentos	Gerenciamento abrangente de eventos
Funcionalidades	Visualização, pesquisa, contato, negociação, segurança	Controle de participantes, emissão de documentos	Controle de participantes, emissão de documentos, funcionalidades abrangentes
Tecnologias	Spring (backend), Angular (frontend), Scrum, Postman, Swagger, Amazon S3	Node, Angular, MySQL	

Figura 1. Tabela de trabalhos correlatos - comparativa

4. Metodologia e materiais

Nesta seção serão apresentadas as etapas para desenvolvimento da proposta, bem como os materiais a serem utilizados.

4.1. Extração e análise de requisitos

Para definir os requisitos do sistema foi adotada uma abordagem pessoal em relação à metodologia de extração. Em vez de conduzir entrevistas com usuários reais foi feita uma análise direta, considerando os trabalhos correlatos que foram estudados. Essa metodologia envolveu uma avaliação reflexiva para identificar características e funcionalidades essenciais na plataforma de aluguel de espaços para eventos corporativos.

Apesar de não incluir a interação com usuários reais, foi possível identificar diferentes perspectivas do contexto do sistema para fundamentar os requisitos essenciais do projeto. Através deste mapeamento foram definidos os requisitos, conforme serão demonstrados na Seção 5.

4.2. Definição do Gerenciador de Projetos Scrum

A metodologia Scrum é escolhida para gerenciar o processo de desenvolvimento do sistema.

O *software* escolhido para suportar o Scrum e o quadro Kanban é o Azure DevOps, que oferece recursos como documentos, lembretes, metas, calendário, agenda- mento, acompanhamento de atividades e integração com o GitHub.

A proposta segue as etapas de desenvolvimento de *software* através da metodologia ágil, sendo definidos nove *sprints*, como são apresentadas no plano de entregas na Figura 1.

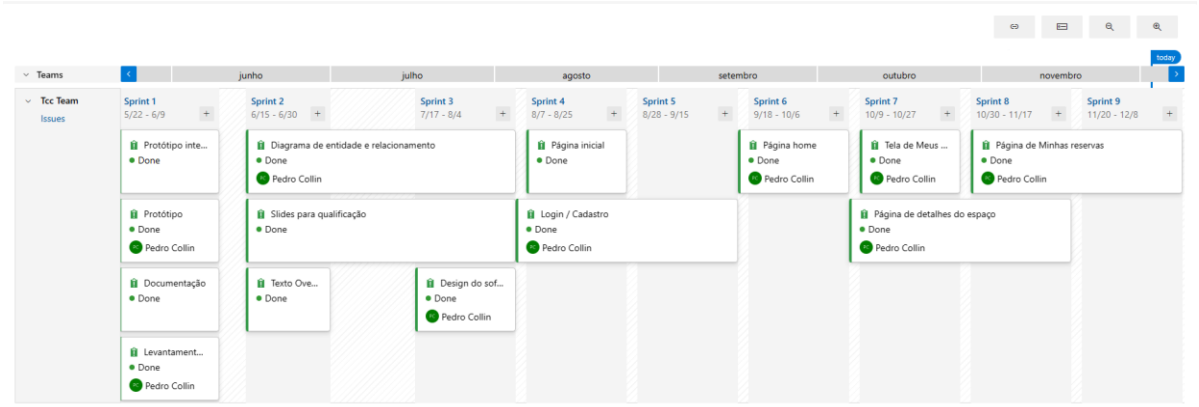

Figura 2. Plano de entrega

4.3. Escolha de tecnologias

Antes de iniciar o desenvolvimento do sistema, realiza-se uma análise das tecnologias disponíveis para a implementação do projeto. Consideram-se aspectos como vantagens, desvantagens e aplicações de cada linguagem, Ambiente de desenvolvimento integrado (IDE), *software*, *framework*, extensão e biblioteca. Com base nessa análise, escolhem-se as tecnologias mais adequadas para o projeto. A Tabela 1 apresenta as tecnologias escolhidas.

É importante enfatizar que o estudo realizado por meio dos trabalhos correlatos possibilitou analisar a viabilidade da execução do projeto, além da pesquisa bibliográfica por meio de livros e sites técnicos que fundamentaram os conhecimentos teóricos que estão associados a parte prática quanto a codificação das funcionalidades.

Como já exposto, a proposta é uma aplicação *web*, sendo definidas as seguintes etapas que serão apresentadas na próxima Seção deste trabalho:

- Requisitos funcionais e não funcionais;
- Diagramas de casos de uso e de classe;
- Diagrama Entidade-Relacionamento;
- Protótipo;
- Desenvolvimento da *API*;
- Implementação do Amazon S3 para armazenamento de imagens;
- Implementação da comunicação entre *backend* e *frontend.*

Tabela 1. Tecnologias definidas para desenvolvimento do projeto

Tecnologia	Descrição	Por que essa tecnologia?
Ferramenta de design e prototipagem	Figma	Figma é uma plataforma de design colaborativa baseada em nuvem que permite criar, prototipar e compartilhar interfaces de usuário e designs interativos. É amplamente utilizado por equipes de desenvolvimento frontend para criar e iterar rapidamente em designs de interfaces atraentes e intuitivas.
IDE de desenvolvimento Backend	Intellij IDEA	IDE poderosa e popular para desenvolvimento de aplicativos Java, oferecendo recursos avançados de depuração, refatoração e suporte a frameworks.
IDE de desenvolvimento FrontEnd	Visual Studio Code	Editor de código leve e altamente personalizável, com suporte a uma ampla gama de linguagens de programação e extensões, tornando-o uma escolha popular para o desenvolvimento FrontEnd.
Linguagem de programação (BackEnd)	Java	Uma linguagem de programação amplamente adotada e robusta, conhecida por sua portabilidade, segurança e suporte a paradigmas de programação orientados a objetos.
Framework (BackEnd)	Spring	Um framework de desenvolvimento Java que fornece um conjunto abrangente de recursos para a criação de aplicativos empresariais escaláveis e de alta qualidade. Ele oferece suporte a injeção de dependência, controle transacional e uma ampla gama de outros recursos para o desenvolvimento backend.
Ferramenta para análise de cobertura de testes unitários	Jacoco	Jacoco é uma ferramenta amplamente utilizada para medir a cobertura de testes unitários em aplicativos Java. Ele permite que os desenvolvedores identifiquem quais partes do código estão sendo testadas e até que ponto, auxiliando na melhoria da qualidade do código.
Ferramenta para análise de qualidade de código	SonarLint	O SonarLint é uma ferramenta de análise estática de código que identifica problemas de qualidade e segurança no código-fonte. Ele ajuda os desenvolvedores a manter um alto padrão de qualidade e aderir às melhores práticas de programação.
Linguagem de programação (FrontEnd)	JavaScript e CSS	JavaScript é uma linguagem de programação amplamente usada para criar interatividade e dinamismo em aplicativos web. CSS é uma linguagem de estilo usada para definir a aparência e o layout dos elementos em uma página da web. Ambas as tecnologias são essenciais para o desenvolvimento FrontEnd moderno.
Framework (FrontEnd)	Angular	Um framework JavaScript popular para o desenvolvimento de aplicativos web escaláveis e responsivos. Ele oferece um conjunto abrangente de recursos, como vinculação de dados, roteamento e gerenciamento de estado, simplificando o desenvolvimento de interfaces de usuário complexas e interativas.
Ferramenta de gerenciamento de banco de dados	DBeaver	DBeaver é uma ferramenta gratuita e de código aberto para gerenciamento de bancos de dados. Ele oferece suporte a uma ampla gama de bancos de dados e fornece recursos avançados para consulta, modelagem, visualização e administração de dados. É uma escolha popular entre os desenvolvedores para trabalhar com bancos de dados no contexto do desenvolvimento backend.

5. Desenvolvimento

Esta seção retrata as fases do desenvolvimento do projeto como um todo, sendo elas: levantamento de requisitos e diagramas, protótipos, desenvolvimento da *API*, implementação do Amazon S3, além da implementação do *frontend* e *backend*.

5.1. Levantamento de requisitos

Conforme estabelecido no escopo deste trabalho, foi realizado o levantamento dos requisitos, tanto funcionais quanto não funcionais, para o desenvolvimento do sistema em questão.

Diante do exposto, a Tabela 2 apresenta os requisitos funcionais, enquanto a Tabela 3 exibe os requisitos não funcionais. Após, são detalhados os requisitos identificados, proporcionando uma visão clara e abrangente das expectativas e restrições que orientaram a construção da proposta.

Tabela 2. Requisitos Funcionais

RF	Descrição
RF-1	Cadastrar novos usuários no sistema.
RF-2	Logar no sistema usando credenciais válidas.
RF-3	Pesquisar espaços disponíveis para locação com base em critérios como localização, capacidade, faixa de preço, tipo de evento, etc.
RF-4	Visualizar informações detalhadas sobre cada espaço, incluindo fotos, descrição, disponibilidade e avaliações de outros usuários.
RF-5	Recursos de filtragem e ordenação para ajudar os usuários a refinar suas pesquisas e encontrar os espaços mais adequados às suas necessidades.
RF-6	Permitir que os usuários solicitem uma reserva de um espaço disponível.
RF-7	Opção para os locadores visualizar, aceitar ou rejeitar a solicitação de reserva
RF-8	Criar um anúncio para disponibilizar um espaço para locação.
RF-9	Editar as informações do anúncio.
RF-10	Pausar temporariamente o anúncio, tornando-o indisponível para reservas.
RF-11	Excluir um anúncio quando não desejar mais disponibilizar o espaço.
RF-12	Enviar mensagens de dúvidas ao locador sobre o espaço antes de fazer uma reserva.

Tabela 3. Requisitos Não Funcionais

RF	Descrição
RNF-1	Interface do usuário deve ser intuitiva e de fácil navegação, permitindo que os usuários encontrem as informações e realizem ações de forma rápida e eficiente.
RNF-2	Informações dos usuários, como senhas e dados pessoais, devem ser armazenadas de forma segura, utilizando técnicas adequadas de criptografia.

5.2. Diagramas

Nesta Subseção serão apresentados os diagramas, sendo o Diagrama Entidade-Relacionamento, o de classe e o de caso de uso.

A Figura 3 apresenta o Diagrama Entidade-Relacionamento (DER), no que tange as tabelas criadas e seus relacionamentos e, assim, formando a estrutura fundamental do banco

de dados da aplicação, permitindo o armazenamento organizado e a recuperação eficiente de informações sobre os espaços disponíveis, as reservas feitas pelos usuários, os tipos de espaços, os detalhes dos usuários e as localizações geográficas dos espaços.

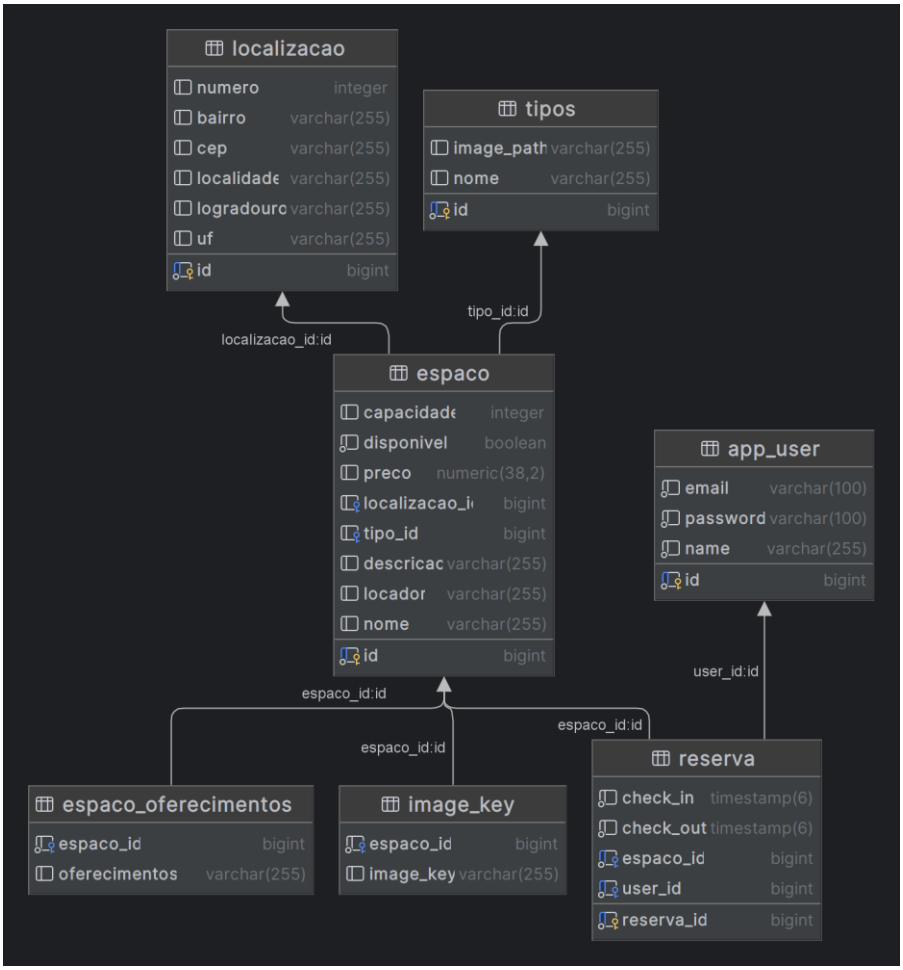

Figura 3. DER

Será feita uma exploração das tabelas criadas, a fim de ter um melhor entendimento de cada uma delas:

- Tabela “espaco”: representa os espaços disponíveis para reserva. Cada registro nesta tabela corresponde a um local físico ou espaço que pode ser reservado por usuários. Os espaços possuem características como nome, descrição, disponibilidade, capacidade, preço e imagens. Além disso, cada espaço está associado a um tipo específico e a uma localização particular, o que permite uma organização eficaz e facilita a busca e reserva de espaços de acordo com a preferência dos usuários.

- Tabela “reserva”: registra as reservas feitas pelos usuários para os diferentes espaços disponíveis. Cada entrada nesta tabela inclui informações sobre o espaço reservado, o usuário que fez a reserva, bem como as datas de *check-in* e *check-out*. Essa estrutura permite o rastreamento eficaz das reservas e a gestão adequada dos espaços disponíveis.

- Tabela “tipo”: descreve os diferentes tipos de espaços disponíveis para reserva. Cada tipo de espaço possui um nome e um caminho de imagem associado, fornecendo informações adicionais sobre os atributos específicos de cada categoria de espaço.

- Tabela "app_user”: armazena informações sobre os usuários que acessam o sistema. Cada usuário possui um nome, um endereço de e-mail e uma senha associada. Esses registros permitem a autenticação e identificação dos usuários que desejam fazer reservas ou acessar outras funcionalidades do sistema.

- Tabela “localizacao”: armazena informações detalhadas sobre as localizações físicas dos espaços disponíveis. Cada entrada nesta tabela inclui dados como CEP, número, logradouro, bairro, localidade e UF, fornecendo informações cruciais para localizar os espaços geograficamente e facilitar a busca com base na localização dos usuários.

- Tabela “image_key”: está associada à tabela “espaco” e armazena as chaves de imagens relacionadas a cada espaço. Como um espaço pode ter várias imagens, a utilização desta tabela permite o armazenamento eficiente das chaves de imagens associadas a cada espaço específico. Isso garante a disponibilidade das imagens relacionadas a um espaço específico para visualização e referência futura.

- Tabela “espaco_oferecimentos”: está relacionada à tabela “espaco” e armazena os diferentes tipos de serviços ou comodidades oferecidas. Isso pode incluir serviços como *wi-fi*, estacionamento, café da manhã, entre outros. A utilização desta tabela permite a categorização eficiente dos diferentes serviços oferecidos em cada espaço, proporcionando aos usuários informações detalhadas sobre as comodidades disponíveis ao fazer uma reserva.

Após a abordagem das tabelas associadas à base de dados, a Figura 4 ilustra o diagrama de classe da aplicação, revelando as entidades fundamentais que compõem a estrutura principal do sistema. As principais entidades incluem Espaco, Reserva, Tipo, User e Localizacao. Cada uma dessas entidades representa elementos cruciais para a funcionalidade da aplicação e mantém relações específicas entre si, otimizando tanto o armazenamento quanto o acesso às informações.

Essas entidades formam a base sólida do sistema, permitindo uma gestão eficiente dos espaços disponíveis, das reservas dos usuários, dos diferentes tipos de espaços, dos detalhes dos usuários e das informações de localização. A interconexão inteligente entre essas entidades contribui para uma estrutura robusta que suporta integralmente as operações e a lógica do sistema.

Em seguida, a Figura 5 aborda o diagrama de caso de uso, levando em consideração a tabela de requisitos funcionais apresentada no início desta Seção (Tabela 2).

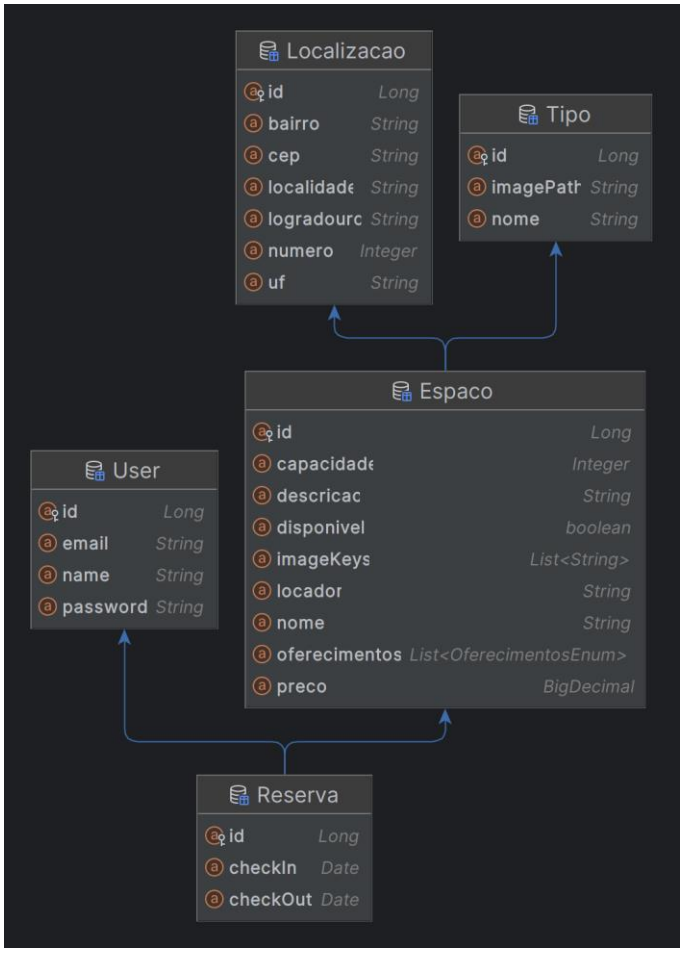

Figura 4. Diagrama de Classe

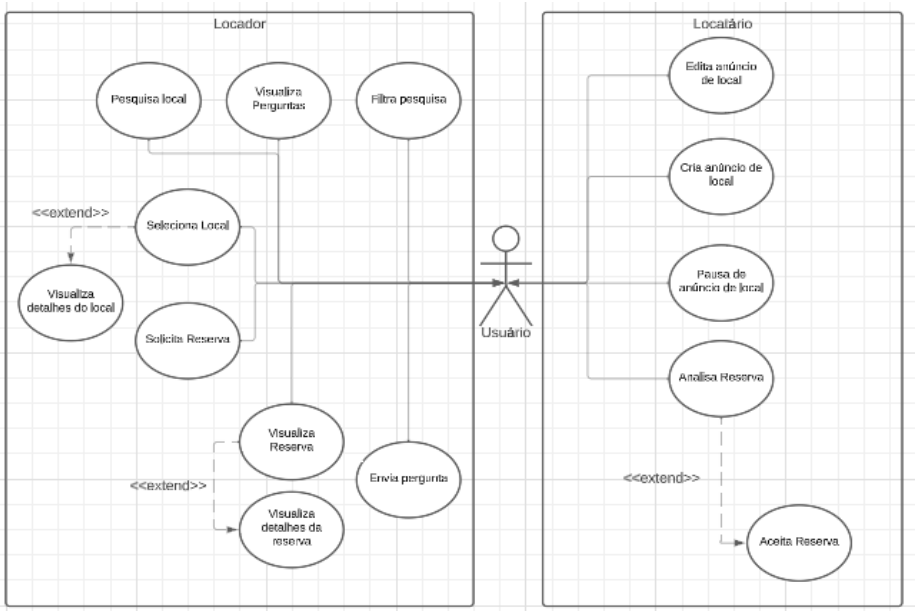

Figura 5. Diagrama de Caso de Uso

5.3. Protótipos X Implementação

Serão apresentados nesta Subseção os protótipos das telas de cada funcionalidade. Sendo a

proposta desenvolvida para plataforma *web*, as telas foram implementadas com o uso da ferramenta *opensource* Figma que é usada para criar protótipos. A fim de ilustrar os protótipos e as telas reais, cada Figura será apresentada nas suas duas versões.

A Figura 6 se refere a tela de apresentação da aplicação. Através dela duas navegações são possíveis: efetuar o login no sistema (Figura 7) ou procurar reserva (Figura 9).

Figura 6. Tela principal

A Figura 6 retrata a tela da funcionalidade de login no sistema. O usuário deve preencher os campos de Email e Senha e, ao clicar no botão “Entrar”, a fim com que a aplicação verifique se as informações estão correspondentes ao banco de dados.

Se estiverem de acordo, uma chave de sessão será gerada por meio do Token JWT para autenticar o usuário, conforme estipulado no requisito RF-2. Caso incorretas, um aviso será apresentado na tela para correção. Se o usuário clicar no botão “Cadastrar-se”, será redirecionado à tela de cadastro.

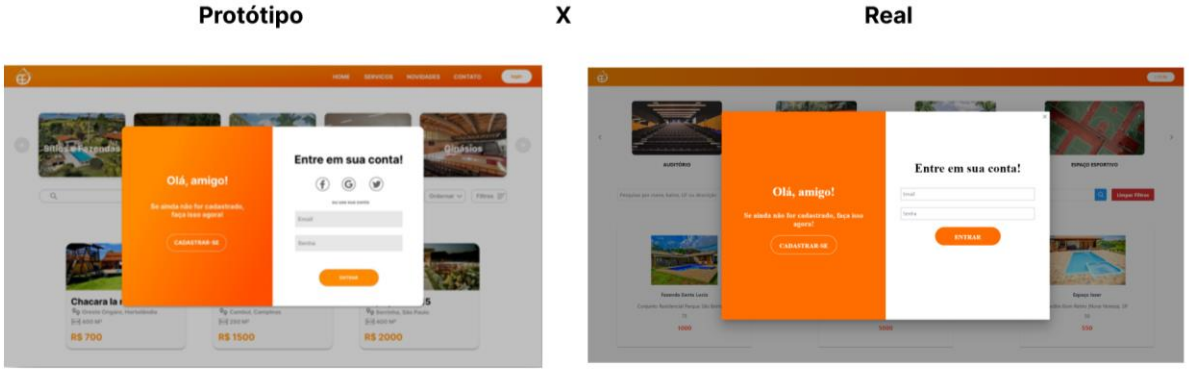

Figura 7. Tela de Login

A Figura 8 representa a tela da funcionalidade de cadastro no sistema. O usuário deve preencher os campos de Nome, Email e Senha e, ao clicar no botão 'Cadastrar-se', a aplicação verificará se as informações seguem as regras para serem armazenadas no banco de dados.

Se estiverem em conformidade, as informações serão registradas e o usuário será autenticado na aplicação, atendendo ao requisito RF-1.

Se o usuário clicar no botão 'Entrar', será redirecionado para a Tela de Login (Figura 6).

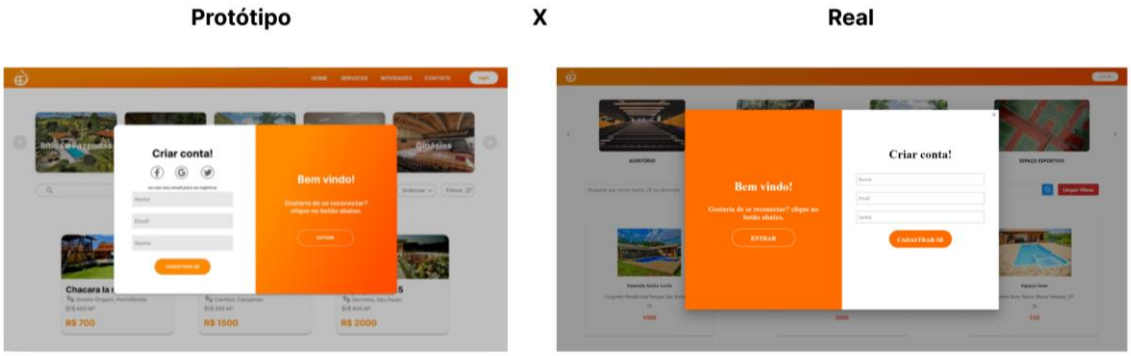

Figura 8. Cadastro de Usuário

A Figura 9 representa a tela de apresentação dos espaços disponíveis para locação, conforme especificado no requisito RF-3.

Nesta tela, os usuários têm a opção de aplicar filtros, ordenar os resultados e realizar pesquisas por nome para facilitar a busca, conforme especificado no RF-5. Os dados dos espaços são carregados diretamente do banco de dados, a partir da tabela “espacos”. Ao clicar em um *card* (modelo de exibição de imagem e detalhes), o usuário é redirecionado à tela de detalhes do espaço, como ilustrado na Figura 10 (RF-4). Caso o usuário clique em “login” o mesmo será redirecionado para essa funcionalidade, como já foi demonstrado por meio da Figura 7.

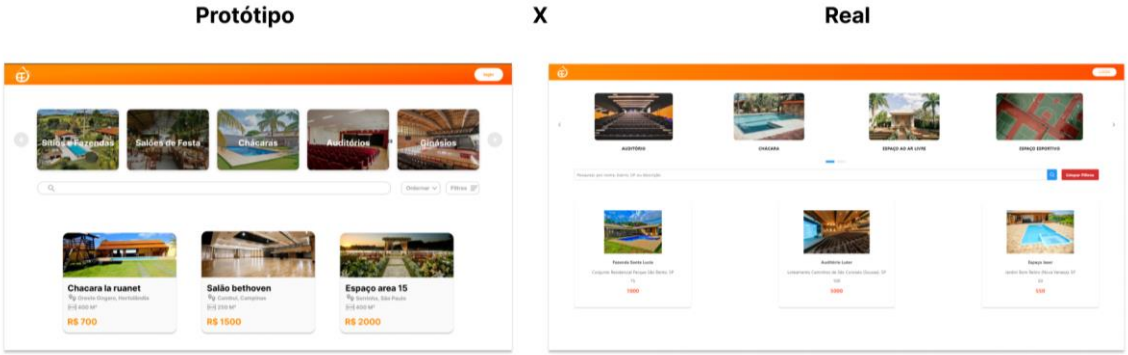

Figura 9. Apresentação dos espaços disponíveis para reserva

A Figura 10 representa a tela que exibe informações sobre o espaço disponível que foi selecionado. Ao clicar no campo “Dias”, um calendário é disponibilizado para facilitar a visualização das datas. Quando o usuário seleciona os dias desejados e clica no botão “Reservar”, uma notificação é exibida no canto inferior da tela, indicando que o usuário deve estar logado para efetuar uma reserva. Se o usuário optar por "Login", ele será redirecionado para a Tela de Login (Figura 7). Após fazer o login, será necessário preencher novamente as informações sobre os “Dias” desejados.

Já a Figura 10 representa a tela de apresentação dos espaços disponíveis para locação, porém com o usuário logado.

Nesta tela, o diferencial são as permissões para o usuário confirmar reservas e, assim, contemplar o RF-6.

Caso o usuário clique em “Minhas reservas” ele será redirecionado para a Tela de Reservas (Figura 12) e caso clique em “Meus Anúncios” será redirecionado para a Tela de

Anúncios (Figura 13).

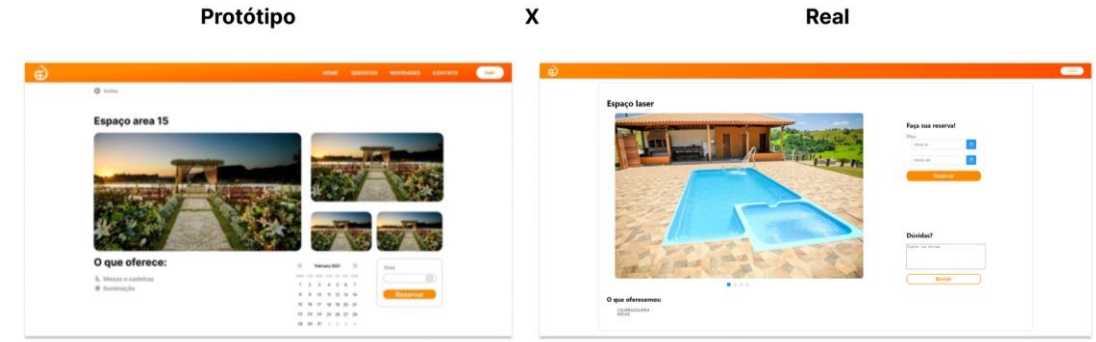

Figura 10. Detalhes do espaço sem login

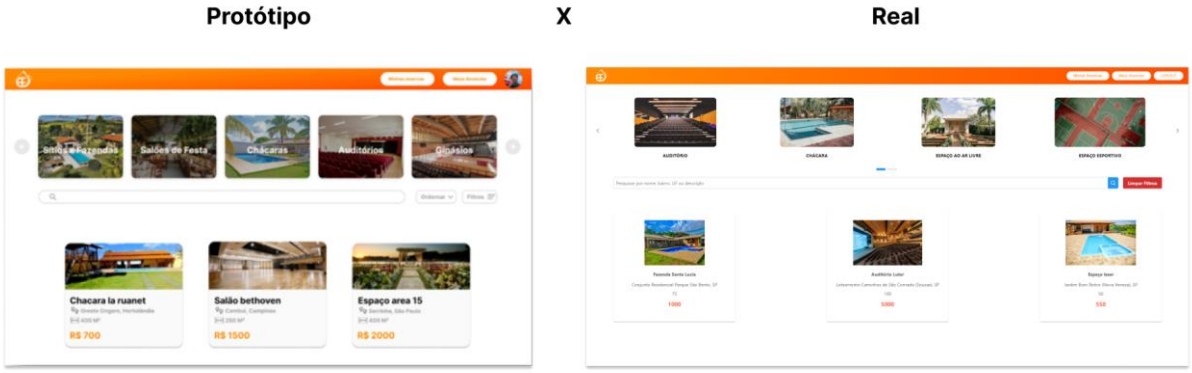

Figura 11. Detalhes do espaço com login

Ainda contemplando o RF-6, a Figura 12 representa a tela que exibe informações sobre o espaço selecionado na Tela de Espaços, conforme RF-4. Ao clicar no campo de 'Dias', um calendário é disponibilizado para facilitar a visualização das datas.

Quando o usuário seleciona os dias desejados e clica no botão 'Reservar', as informações do espaço, os detalhes do usuário logado e as datas de início e término são registradas na tabela 'reserva' do banco de dados, conforme RF-6. Por meio dessa tela o usuário também poderá encaminhar mensagens ao responsável pelo anúncio, a fim de tirar dúvidas e, assim, contemplar o RF-12.

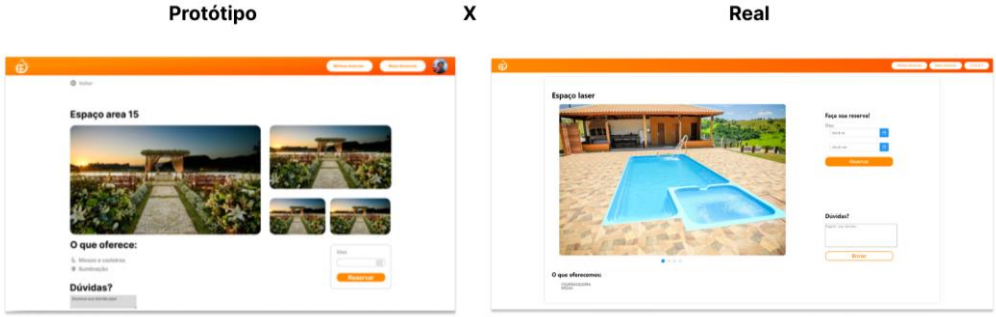

Figura 12. Detalhes do espaço para reserva com login

O usuário cadastrado também poderá consultar suas reservas, como é apresentada na Figura 13.

Já a Figura 14 ilustra possíveis anúncios que foram cadastrados pelo usuário logado,

contemplando, assim, os requisitos funcionais do 7 ao 11. A seção oferece uma visão geral rápida de todas as suas ofertas.

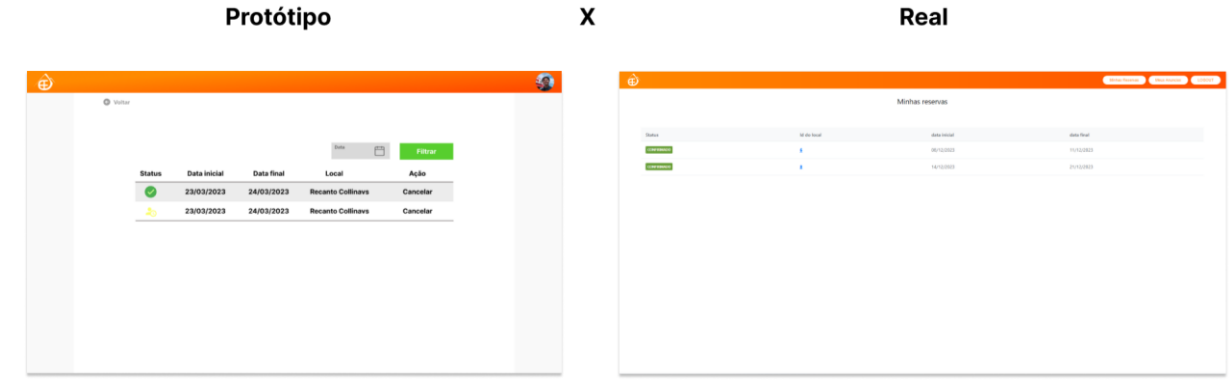

Figura 13. Consultar reservas

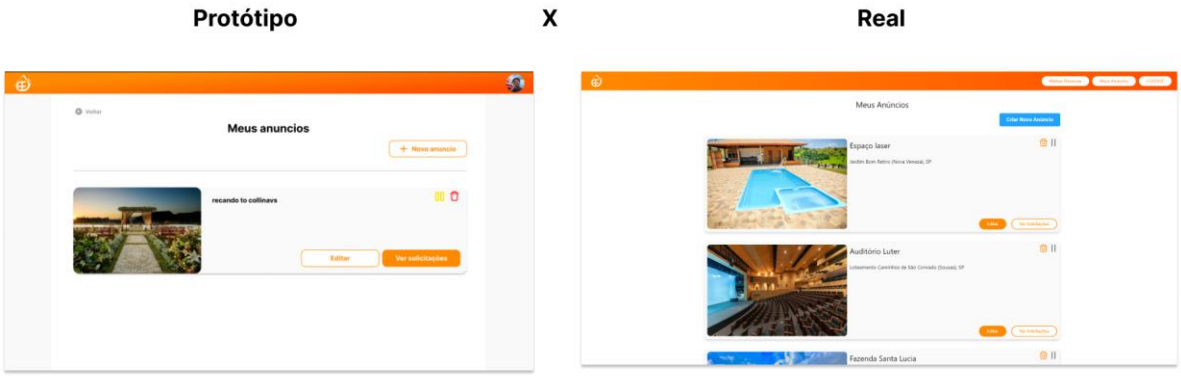

Figura 14. Gerenciar anúncios

5.4. Desenvolvimento da *API*

Na *API* do sistema foi optado por adotar o *framework* Java Spring devido à sua ênfase na robustez, escalabilidade e facilidade de manutenção. O Java Spring oferece uma arquitetura eficiente para o desenvolvimento de *APIs* RESTful, simplificando a criação de pontos de acesso que permitem uma comunicação eficaz entre o sistema e seus usuários.

Para garantir a autenticação do usuário e proteger rotas específicas, foi implementado o *framework* Spring Security 6. Esse mecanismo foi essencial para controlar o acesso dos usuários a páginas e rotas do sistema. Além disso, o Spring Security foi utilizado para gerenciar a autenticação de login e o registro de novos usuários, gerando tokens do tipo Bearer JWT[2] para autorizar o acesso a rotas destinadas à obtenção de informações pelo lado

[2] O JSON Web Token (JWT) é composto por três partes: um cabeçalho (*header*) com informações sobre o tipo de token e algoritmo de criptografia, um *payload* (carga útil) com as informações do usuário e um tempo de expiração, e uma assinatura digital para garantir a integridade do token. A estrutura JWT é geralmente armazenada do lado do cliente, em um cookie ou armazenamento local (como o localStorage do navegador) para ser enviado de volta ao servidor em cada requisição. O servidor pode decodificar o token para verificar a

do cliente do sistema.

A fim de facilitar a compreensão das diferentes funcionalidades das rotas disponíveis no sistema, foi escolhido o *framework* Swagger para essa documentação. O Swagger especifica os recursos acessíveis na *API* REST e as operações que podem ser executadas nesses recursos. Além disso, detalha os parâmetros necessários para cada operação, incluindo nome, tipo e informações sobre os valores aceitáveis a serem inseridos. O Swagger é acompanhado pelo Swagger UI, uma interface de usuário que simplifica a exploração e interação com a documentação da *API*, tornando mais acessível a visualização das rotas, a execução de chamadas e a compreensão dos resultados (Figura 15).

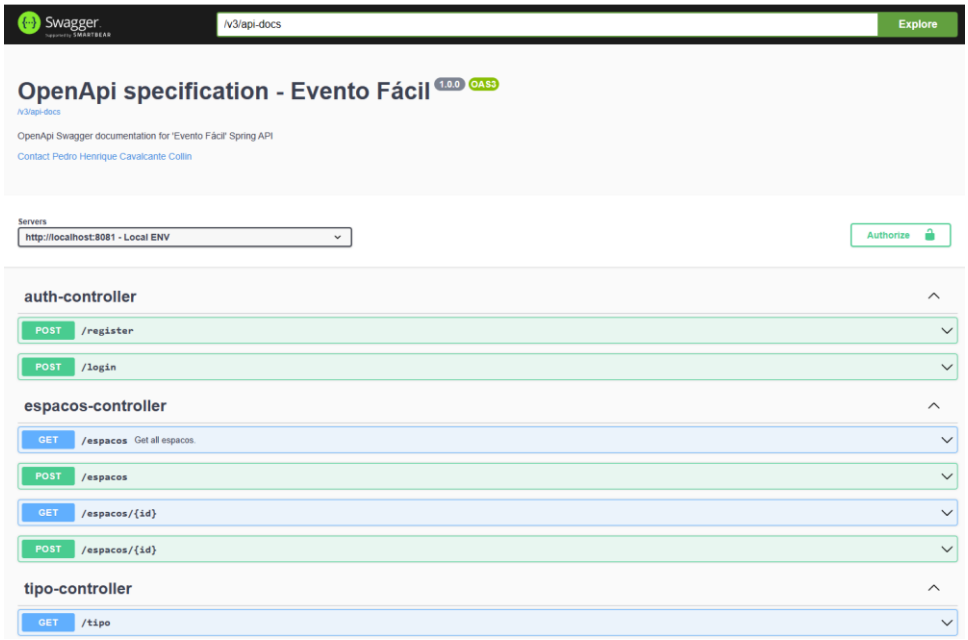

Figura 15. Ferramenta Swagger UI

5.5. Implementação do Amazon S3 para armazenamento de imagens

Para otimizar o armazenamento e o gerenciamento das imagens que foram utilizadas na aplicação, foi realizada uma implementação eficiente utilizando o Amazon S3 (Simple Storage Service) da AWS (Amazon Web Services). O Amazon S3 oferece um serviço de armazenamento de objetos altamente escalável, confiável e de baixo custo, adequado para o caso da aplicação do Evento Fácil.

Na estrutura do banco de dados, a tabela image_key armazena as chaves de acesso às imagens que são armazenadas no Amazon S3. Em vez de armazenar as imagens diretamente no banco de dados, essa abordagem oferece diversos benefícios, incluindo a redução da carga de armazenamento no banco de dados e o acesso mais rápido e eficiente às imagens, resultando em uma melhor experiência do usuário.

A implementação consistiu em configurar um *bucket*[3] no Amazon S3 para o armazenamento das imagens. A partir disso, cada vez que uma nova imagem é carregada no

autenticidade e a autorização do usuário.

[3] *Buckets* são os *containers* para objetos, podendo ter um ou mais. Para cada *bucket* é possível controlar o acesso a ele (quem pode criar, excluir e listar objetos nele), exibir *logs* de acesso para ele e seus objetos escolher a região geográfica onde o Amazon S3 armazenará o *bucket* e seu conteúdo.

sistema, a aplicação gera uma chave exclusiva e armazena essa chave na tabela image_key, que atua como um identificador único para a imagem no *bucket* do Amazon S3.

Além disso, durante o desenvolvimento, medidas de segurança e controle de acesso foram implementadas para garantir a proteção e confidencialidade das imagens armazenadas. Isso inclui a configuração de políticas de acesso e permissões apropriadas para o *bucket* do Amazon S3, assegurando que apenas os usuários autorizados tenham permissão para visualizar e acessar as imagens armazenadas.

A utilização do Amazon S3 proporcionou uma solução robusta e escalável para o armazenamento e gerenciamento das imagens, contribuindo para a otimização do desempenho da aplicação, a economia de custos de armazenamento e a segurança dos dados. Essa implementação oferece uma base sólida para o crescimento e a escalabilidade da aplicação, atendendo às demandas cada vez maiores de armazenamento e acesso a imagens em um ambiente de aplicação dinâmico. A Figura 16 aborda uma classe nomeada "ImageServiceImpl" que representa a implementação do serviço responsável pelo gerenciamento de imagens utilizando o Amazon S3.

```java
public class ImageServiceImpl implements ImageService {

    @Value("${aws.secretKey}")
    private String secretKey;

    @Value("${aws.s3.bucketName}")
    private String s3BucketName;

    @PostConstruct
    private void initializeS3Client() {
        AWSCredentials credentials = new BasicAWSCredentials(accessKey, secretKey);
        this.s3Client = AmazonS3ClientBuilder.standard()
                .withCredentials(new AWSStaticCredentialsProvider(credentials))
                .withRegion(Regions.SA_EAST_1)
                .build();
        this.bucketName = s3BucketName;
    }

    @Override
    public void uploadImage(MultipartFile file, String key) throws IOException {
        ObjectMetadata metadata = new ObjectMetadata();
        metadata.setContentLength(file.getSize());

        s3Client.putObject(new PutObjectRequest(bucketName, key, file.getInputStream(), metadata));
    }

    @Override
    public void deleteImage(String key) {
        s3Client.deleteObject(bucketName, key);
    }

    @Override
    public String getImageUrl(String key) {
        return s3Client.generatePresignedUrl(bucketName, key, Date.from(Instant.now().plus(5, ChronoUnit.MINUTES))).toString();
    }
}
```

Figura 16. Implementação do Serviço de Imagem

A seguir são listadas as principais características da implementação:

- **Configuração do Cliente S3:** a classe utiliza a biblioteca AWS SDK for Java para configurar um cliente Amazon S3. As credenciais de acesso e a região são obtidas a partir de propriedades definidas no arquivo de configuração.

- **Método uploadImage:** este método recebe um arquivo (MultipartFile) e uma chave como parâmetros e realiza o *upload* da imagem para o *bucket* do Amazon S3. As informações sobre o arquivo, como tamanho são configuradas antes do envio.

- **Método deleteImage:** remove uma imagem do *bucket* do Amazon S3 com base na chave fornecida.

- **Método getImageUrl:** gera uma *URL* assinada temporária para acesso à imagem. A *URL* é válida por 5 minutos a partir do momento da geração, proporcionando uma abordagem

segura para compartilhamento temporário de imagens.

- Inicialização do Cliente S3 no Método initializeS3Client: utiliza as credenciais configuradas para criar uma instância do cliente Amazon S3 durante a inicialização do serviço.

Diante da abordagem, enfatiza-se que essa implementação permite uma integração eficiente entre a aplicação e o Amazon S3, proporcionando armazenamento escalável, rápido acesso às imagens e a implementação de medidas de segurança, como *URLs* assinadas temporariamente, a fim de garantir a confidencialidade e a integridade dos recursos armazenados.

5.6. Implementação da comunicação eficiente entre *backend* e *frontend*

Para integração do *backend* com o desenvolvimento em Spring Boot, com o *frontend* em Angular, deve ser adotada uma abordagem de comunicação eficiente e contínua. Esta abordagem visa estabelecer uma interação sólida entre as duas partes do sistema.

A estrutura do *backend* é configurada com *endpoints* RESTful, os quais possibilitam a troca de dados de maneira segura e consistente com o *frontend*. É fundamental a definição de serviços no *backend* visando a exposição das funcionalidades necessárias para o *frontend*. Por meio de rotas e controladores específicos, a comunicação controlada e confiável é assegurada, permitindo o acesso restrito aos recursos e operações disponíveis.

No *frontend* esses serviços são integrados por meio de requisições HTTP bem estruturadas. A implementação de chamadas HTTP, como GET, POST, PUT e DELETE, garante a integridade e a consistência das informações trocadas entre as camadas do sistema.

O uso de recursos específicos do Angular, como módulos e serviços, facilita o consumo desses *endpoints* e a apresentação adequada dos dados na interface do usuário. Essa prática resulta em uma experiência fluida e responsiva, permitindo interações eficazes dos usuários sem comprometer a segurança ou a integridade dos dados.

Ao adotar essa abordagem de integração, uma clara separação entre as responsabilidades do *backend* e do *frontend* é mantida, assegurando a escalabilidade e a manutenibilidade do sistema como um todo. Esta integração contínua é fundamental para o desenvolvimento de uma aplicação robusta e coesa, capaz de atender às necessidades e expectativas dos usuários finais. O código apresentado na Figura 17 faz parte do serviço Angular denominado “EspacosService”, responsável por gerenciar as interações entre o *frontend* e o *backend* relacionadas aos espaços.

A seguir são exploradas as principais características do código:

- Injeção de dependência: o serviço utiliza o decorador (espécie de anotação) @Injectable para indicar que pode ser injetado em outras partes da aplicação. A dependência do serviço HttpClient também é injetada no construtor do serviço.

- Definição da *URL* do *backend*: a *URL* do *backend* é definida como "http://localhost:8080" e armazenada na variável *URL*, tornando-se, assim, a base para as requisições HTTP que serão feitas para obter dados do *backend*.

- **Métodos para obter tipos e espaços:**
    - getTipos(): este método utiliza o serviço HttpClient para fazer uma requisição GET à *URL* ${this.url}/tipo, buscando informações sobre os tipos disponíveis.
    - getEspacos(): similar ao método anterior, porém esse realiza uma requisição

GET à *URL* ${this.url}/espacos para obter informações sobre os espaços disponíveis.

- Utilização de Tipos Genéricos: a resposta das requisições HTTP é do tipo “any”, indicando que o serviço está tratando dados de forma genérica e não específica.

- DTO (Data Transfer Object): o serviço utiliza o tipo ReservaDTO importado de 'src/app/shared/dtos/espaco-dto'. Isso sugere a presença de um objeto de transferência de dados para representar as informações relacionadas às reservas.

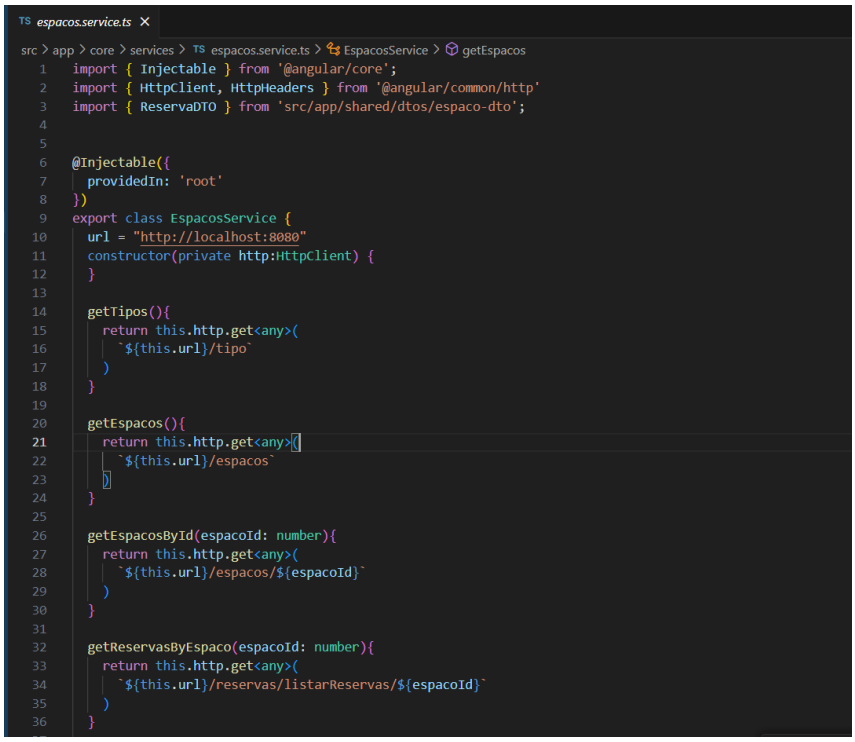

Figura 17. Camada de serviços *frontend*

Essa estrutura permite a comunicação eficiente entre o *frontend* em Angular e o *backend* em Spring Boot, seguindo as melhores práticas para integração de serviços em uma aplicação *web*.

Já o trecho de código apresentado na Figura 18 corresponde a um controlador Spring Boot para a entidade "Espacos". Abaixo estão as principais características e funcionalidades deste controlador:

- A classe é anotada com @RestController, indicando que seus métodos tratam requisições HTTP e retornam diretamente os dados, geralmente no formato JSON.

- A annotation @RequestMapping("/espacos") define o mapeamento de *URL* para todos os métodos deste controlador. Todas as requisições feitas a /espacos serão tratadas por ele.

- Injeção de dependências: o controlador faz uso da injeção de dependências no construtor para obter instâncias dos serviços necessários: EspacosService, EspacosMapper, e ImageService. Esta prática promove a modularidade e facilita a manutenção do código.

- Método Construtor: o construtor da classe inicializa as instâncias dos serviços necessários para o funcionamento do controlador.

- O código inclui documentação para a *API* usando as anotações do Swagger/OpenAPI (@Operation, @ApiResponses, @ApiResponse, @Schema). Essas anotações são valiosas

para documentá-la, especificando o resumo da operação, possíveis respostas e esquema da resposta.

Esse controlador, juntamente com a documentação fornecida, sugere uma abordagem organizada e bem documentada para a exposição de recursos relacionados aos espaços na sua aplicação Spring Boot.

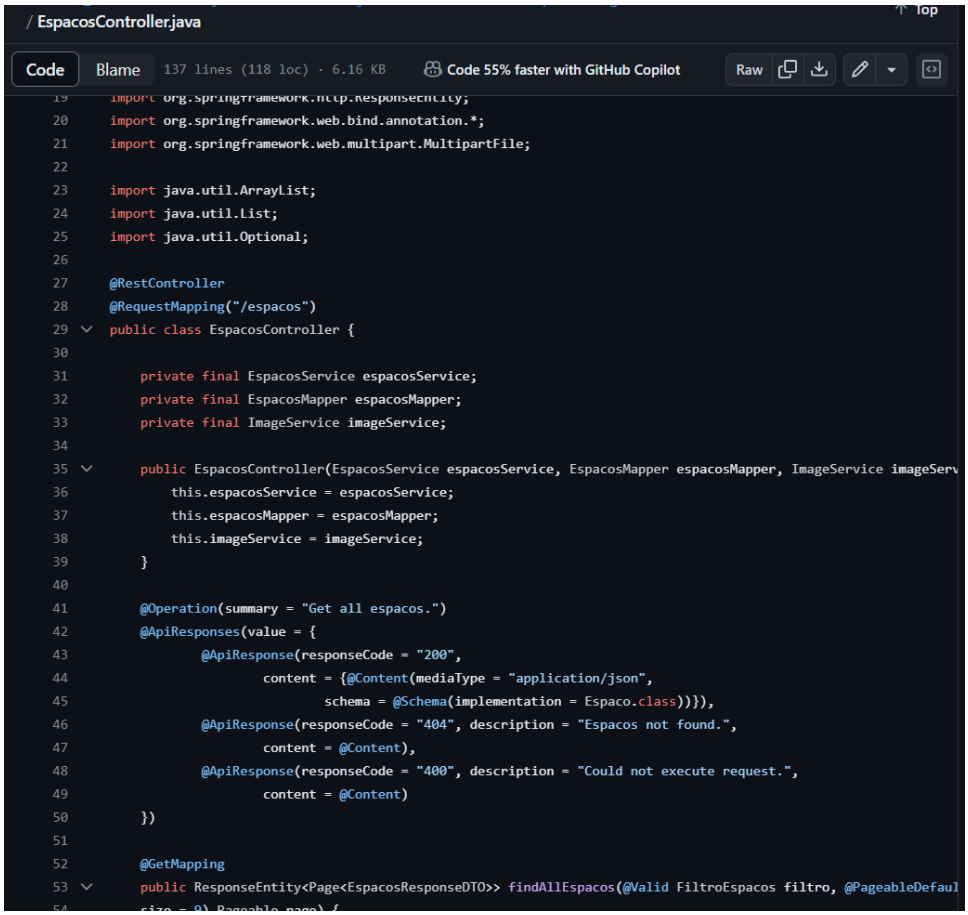

Figura 18. Controller Espaços

6. Conclusão

Ao longo do desenvolvimento deste projeto, o objetivo foi criar uma plataforma digital que otimizasse o processo de aluguel de locais para eventos. Para isso, foi optado por uma abordagem centrada no usuário, utilizando tecnologias modernas e metodologias ágeis para garantir uma experiência eficiente e intuitiva.

6.1 Checklist de Funcionalidades Atendidas:

- Cadastro de novos usuários no sistema;
- Login no sistema;
- Pesquisa de espaços disponíveis;
- Filtragem de espaços de diferentes maneiras;
- Solicitação de reserva por parte do usuário;
- Aceitação ou recusa de reserva pelo locador;
- Criação, pausa ou exclusão de anúncio pelo locador;
- Edição de anúncio.

A escolha do framework Spring para o *backend* e o Angular para o *frontend* proporcionou uma base sólida para a construção da aplicação. A decisão pela arquitetura de microsserviços visou escalabilidade e flexibilidade, enquanto a implementação do Swagger facilitou a documentação da *API*, tornando-a acessível na hora do desenvolvimento da mesma.

A integração eficaz entre o *backend* e o *frontend*, aliada à implementação do Amazon S3 para o armazenamento de imagens, contribuiu para a otimização do desempenho da aplicação e a segurança dos dados.

Durante a reflexão sobre o trabalho realizado, foi identificado oportunidades de aprimoramento, como uma análise mais aprofundada dos requisitos o qual poderia ter proporcionado uma compreensão ainda mais refinada das necessidades dos usuários.

6.2 Trabalhos Futuros

Para os próximos passos, consideramos importante implementar funcionalidades adicionais que enriqueçam a experiência do usuário, como a possibilidade de avaliações e comentários sobre os locais de eventos. Essa funcionalidade não só proporcionaria aos usuários uma maneira de compartilhar suas experiências, mas também forneceria informações valiosas para aqueles que estão procurando um local para seus eventos. Além disso, a inclusão de um sistema de notificações poderia manter os usuários informados sobre eventos próximos ou comentários relevantes, aumentando ainda mais o engajamento na plataforma.

Outra possibilidade a ser explorada é a expansão da plataforma para incluir uma aplicação móvel. Isso ampliaria significativamente o alcance da plataforma, permitindo que os usuários acessem facilmente os serviços oferecidos em seus dispositivos móveis. Uma aplicação móvel bem projetada poderia aproveitar recursos específicos dos dispositivos, como geolocalização, para oferecer uma experiência mais personalizada e conveniente aos usuários.

Este projeto representa um avanço significativo na digitalização do processo de aluguel de espaços para eventos. A busca contínua por aprimoramento e adaptação às demandas do mercado são essenciais para garantir o sucesso contínuo desta plataforma no cenário digital. Para isso, é fundamental manter-se atualizado sobre as tendências tecnológicas e as necessidades dos usuários, buscando sempre inovar e oferecer soluções que agreguem valor.

É importante ressaltar que ao longo do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, as disciplinas de Arquitetura de Software, Desenvolvimento Web II, Engenharia de Software, Inglês Técnico, Linguagem de Programação II, Metodologias Ágeis, Programação Orientada a Objetos e Qualidade de Software forneceram o embasamento técnico e prático necessário para o desenvolvimento de todas as etapas deste projeto. A aplicação dos princípios e práticas aprendidos nessas disciplinas foi fundamental para garantir a qualidade e eficiência do sistema desenvolvido.

A parte de teste também é crucial para o sucesso do projeto e deve ser tratada como um trabalho futuro. A implementação de testes automatizados, como testes unitários e de integração, pode ajudar a identificar e corrigir problemas de forma rápida e eficiente, garantindo a estabilidade e confiabilidade da plataforma. Além disso, a realização de testes de usabilidade com usuários reais pode fornecer insights valiosos sobre a experiência do usuário e ajudar a otimizar o design e a funcionalidade da aplicação.

7. Referências

Batista, E. O. (2006). “Sistemas de informação: o uso consciente da tecnologia para o gerenciamento”. Editora Saraiva.

Caetano, A. S. (2019). “Sistema Web Para Gerenciamento de Eventos e Emissão de Documentos – SKED”. https://repositorio.ufu.br/bitstream/123456789/28923/1/SistemaWebGerenciamento.pdf. [Online: acessado em 23 de junho de 2023].

Carnell, J. (2017). “Spring Microservices in Action”. Manning Publications.

Carvalho, L. P.; Brito, M. S.; Matos, P. F.; Barbosa, L. A.; Gomes, C. L.; Ferreira, I. R. L. “e-Vent-Br: Proposta de um Sistema *Web* de Gerenciamento de Eventos Acadêmicos”. https://www.academia.edu/7918302/e_Vent_Br_Proposta_de_um_Sistema_Web_de_Gerenciamento_de_Eventos_Acad%C3%AAmicos. [Online: acessado em 23 de junho de 2023].

Horstmann, C. (2004). “Big Java”. Editora Bookman.

Junior, N., & Afonso, A. (2017). “Produtividade no Desenvolvimento de Aplicações Web com Spring Boot”. Editora AlgaWorks Softwares, Treinamentos e Serviços LTDA.

Kniberg, H., & Skarin, M. (2010). Kanban e Scrum - Obtendo o Melhor de Ambos. https://www.infoq.com/br/minibooks/kanban-scrum-minibook/. [Online: acessado em 07 de maio de 2023].

Laudon, K. C.; Laudon, J. P. (2011). "Sistemas de Informação Gerenciais". Editora Pearson Education do Brasil, 9ª Edição.

Levy, J. (2015). “UX Strategy”. O'Reilly Media, Inc.

Newman, S. (2015). “Building Microservices”. O'Reilly Media, Inc.

PostgreSQL. (2023). “PostgreSQL Tutorial”. https://www.postgresqltutorial.com/. [Online: acessado em 08 de agosto de 2023].

Schmitz, D., & Lira, D. (2016). “AngularJS Na Prática”. https://www.academia.edu/31451036/AngularJS_na_pr%C3%A1tica_PT_BR. [Online: acessado em 08 de agosto de 2023].

Sommerville, I. (2015). “Engenharia de Software”. Editora Pearson Education do Brasil, 10ª Edição.

Sutherland, J., & Sutherland, J.J. (2014). “Scrum: The Art of Doing Twice the Work in Half the Time”. Crown Business.