UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Igor Mendonça Abe

Sistema de Gerenciamento de Patrimônios Tombados

Uberlândia, Brasil

2024

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Igor Mendonça Abe

Sistema de Gerenciamento de Patrimônios Tombados

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Sistemas de Informação.

Orientador: Bruno Augusto Nassif Travençolo

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2024

Resumo

Neste trabalho é apresentado um sistema para gerenciamento de patrimônios tombados. Essa ferramenta permite o cadastrado dos patrimônios, vincular arquivos como fotos, planilhas e outros documentos, além de permitir realizar a edição nos dados quando necessário. Também foi implementado o gerenciamento e o controle dos usuários que irão utilizar o sistema internamente. Além disso, a aplicação também permite o acesso ao público em geral, para que o conhecimento sobre nossa herança cultural possa ser transmitida a todos. O sistema foi construído utilizando linguagem Java (*back-end*) e foram usadas as tecnologias para a criação de páginas *web*, como o Angular (*front-end*).

Palavras-chave: Patrimônio, tombamento, patrimônio tombado, desenvolvimento de software, bens históricos.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	Application Programming Interface
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
DER	Diagrama Entidade Relacionamento
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
JPA	Java Persistence API
MR	Modelo Relacional
SGBD	Sistema Gerenciador de Banco de Dados
UNESCO	Organização das Nações Unidas para a Educação, a Ciência e a Cultura
URL	Uniform Resource Locator
UML	Linguagem de Modelagem Unificada

Sumário

1 Introdução

Patrimônios tombados referem-se à bens culturais, sendo materiais ou não, reconhecidos pelo Estado por prover uma grande importância histórica para a sociedade e aos fatos memoráveis ao Brasil. De acordo com o Iphan (2022), os patrimônios podem ser monumentos, casas, acervos bibliográficos, entre outros. Eles estão protegidos sob o Decreto-Lei n.º 25, de 30 de novembro de 1937 (DECRETO-LEI..., 1937), que regula a atividade de tombamento e define as suas diretrizes. Artigo 1°, "Constitui o patrimonio histórico e artístico nacional o conjunto dos bens móveis e imóveis existentes no País e cuja conservação seja do interesse público, quer por sua vinculação a fatos memoráveis da história do Brasil, quer por seu excepcional valor arqueológico ou etnográfico, bibliográfico ou artístico."

A preservação dos patrimônios culturais é de suma importância para a sociedade, pois esses representam a história, valores e tradições de uma nação. Eles são a conexão das pessoas com suas raízes, promovendo a identidade de um povo.

Com a implementação de decretos-lei e políticas de proteção, os patrimônios são resguardados pelo Estado, recebendo a devida atenção e cuidado para garantir sua segurança e preservação para as próximas gerações. Essas medidas visam não apenas salvaguardar os bens culturais, mas também promover o respeito à diversidade cultural e incentivar a valorização e a compreensão do legado deixado pelos nossos antepassados.

A proteção do patrimônio cultural vai além da mera conservação física dos monumentos e artefatos. Envolve também a promoção do acesso público, o fomento à educação patrimonial, o estímulo ao turismo cultural responsável e a busca por práticas sustentáveis de gestão.

Portanto, preservar os patrimônios culturais é uma tarefa de todos nós, que demanda recursos e a colaboração de diversas pessoas, incluindo governos, comunidades, instituições culturais e a própria sociedade conforme (IPHAN, 2022) . Esses tesouros tombados nos permite compreender melhor o passado e construir um futuro respeitando à diversidade cultural.

1.1 Objetivos

Este projeto tem como objetivo a contribuir no desenvolvimento de um sistema de gerenciamento de patrimônios tombados, permitindo aos responsáveis legais realizar novos cadastros e gerenciar os bens históricos.

1.2 Justificativa

O desenvolvimento do sistema visa contribuir para com a sociedade e a nação. O gerenciamento de patrimônios tombados é de suma importância para a história do Brasil, preservando seu passado e futuro. Com esta aplicação, os responsáveis por cadastrar e gerenciar possuirão uma ferramenta para gerencia e análise dos patrimônios.

1.3 Metodologia de desenvolvimento

Para começar, foi realizado um levantamento das principais funções na qual um responsável por gerenciar os patrimônios tombados utilize em seu dia a dia. Depois, o processo da modelagem de software foi iniciado, elaborando seus requisitos e casos de uso. Em conjunto, deu-se início à criação do **DER**, e posteriormente à criação do modelo relacional.

O desenvolvimento da aplicação foi divido em duas linhas, sendo elas o *back-end* e o *front-end*, simultaneamente. O primeiro é responsável por conter quase toda a lógica do sistema, realizando operações **CRUD**, checagem e tratamento dos dados, controle de acesso, entre outros. Já o segundo, é a parte na qual o usuário tem o contato com o software, pois é nesse módulo quem ficam as páginas de navegação.

No *back-end* foram usadas tecnologias robustas como a linguagem de programação Java, que é consolidada no mercado, seu uso e conhecimento por outros desenvolvedores é extremamente alto, facilitando assim futuras implementações e correções. Enriquecendo ainda mais esta linha, foi utilizado *frameworks spring*, facilitando no desenvolvimento da **API**.

No *front-end*, foram usadas as tecnologias para a criação de páginas **web**, como o Angular, uma plataforma gratuita para criação de aplicações **web**.

Juntamente, foi desenvolvido o banco de dados relacional para a aplicação. Foi utilizado o **SGBD** *Postgres*, pois o mesmo oferece excelente desempenho para o propósito da aplicação, além de ser totalmente gratuito.

2 Fundamentação Teórica

Neste capítulo são apresentados os trabalhos relacionados e a descrição das tecnologias utilizadas no projeto.

2.1 Trabalhos Correlatos

2.1.1 Instituto do Patrimônio Histórico e Artístico Nacional (Iphan)

O *website* Instituto do Patrimônio Histórico e Artístico Nacional (IPHAN, 2022) está atualmente em processo de migração para a plataforma mantida pelo Governo Federal do Brasil (IPHAN, 2023a).

De acordo com o Iphan (2023b), o instituto possui um repositório de documentos digitalizados, que está sendo alimentado aos poucos com acervos de vários estados. Entretanto, ao tentar acessar o repositório, foi retornado um erro na aplicação.

No então portal a ser desligado, temos logo no início um layout de fácil acesso aos principais tópicos de patrimônios. Na Figura 1 é possível visualizar a clara mensagem da migração do *website* para a plataforma do governo.

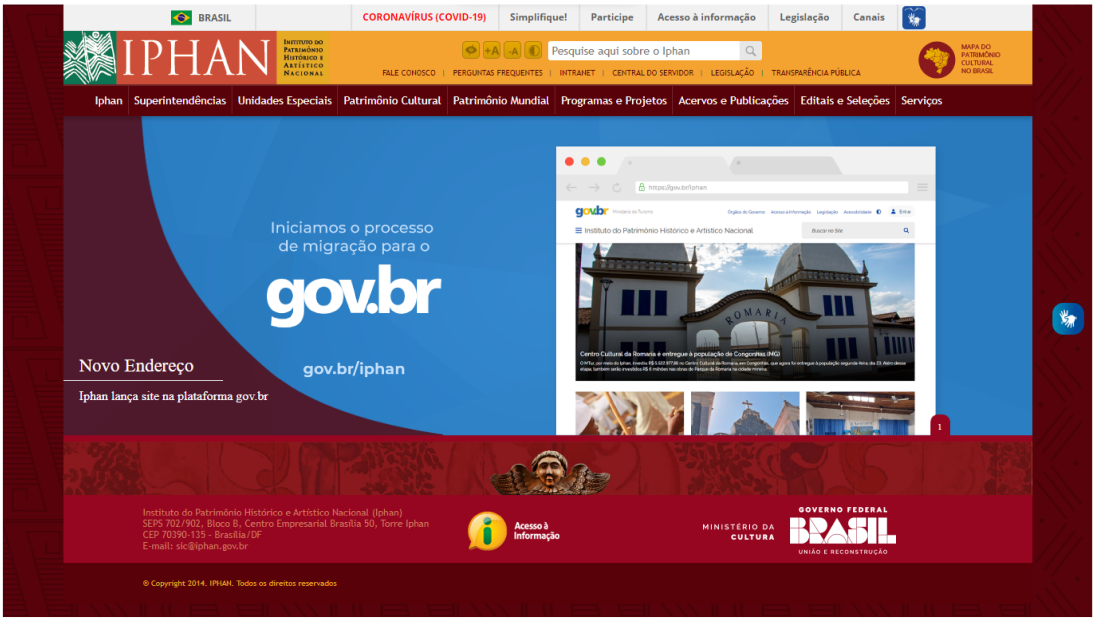

Figura 1 – Página inicial do Instituto do Patrimônio Histórico e Artístico Nacional – Iphan.

A nova página *web* possui um visual mais limpo, avançado e com um *template* padrão do governo, fornecendo os mesmos recursos que o antigo possuía e muito mais,

podendo até mesmo se autenticar.

Figura 2 – Página inicial do novo *website* do Iphan.

A navegação no portal é fácil e intuitiva, bastando alguns cliques para descobrir o imenso acervo de patrimônios tombados, e alguns até mesmo reconhecidos mundialmente pela **UNESCO**. No entanto, algumas páginas ainda não foram migradas e só existem no antigo portal, como no caso da página patrimônio mundial cultural e mundial.

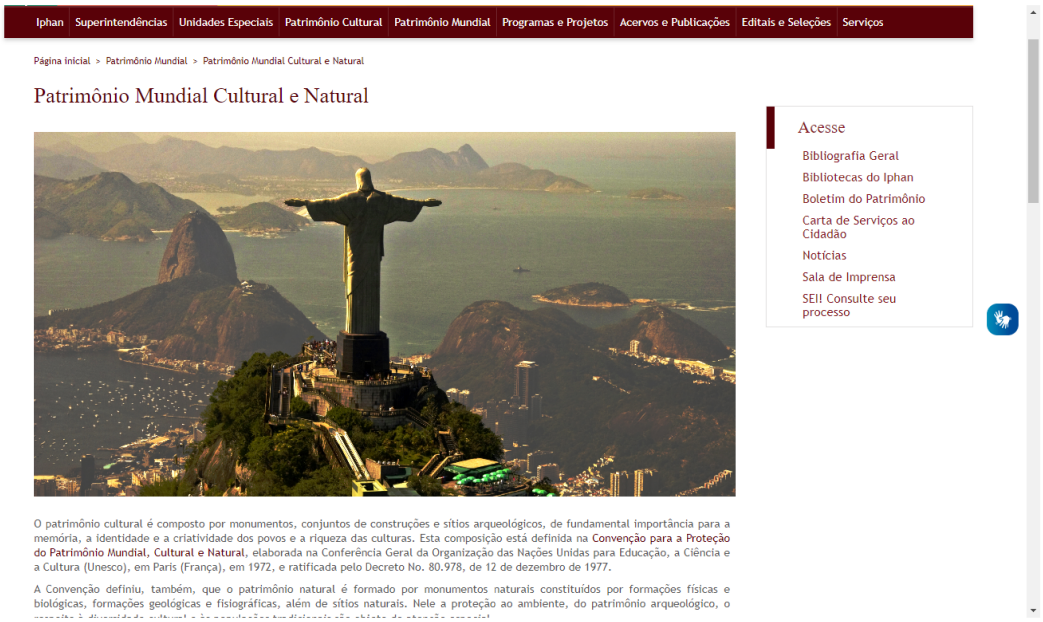

Figura 3 – Página de patrimônio mundial cultural e natural, presente no antigo portal do Iphan.

2.1.2 Retomba

O Retomba é um aplicativo voltado para *smartphones*, com o intuito de remontar virtualmente construções da cidade e distritos de Uberlândia, que já foram demolidas no passado (RETOMBA, 2024). Com a câmera do aparelho apontando para o local onde haveria a construção, o aplicativo consegue modelar a construção por meio de técnicas em 3D, contribuindo com o registro e preservação arquitetônica da cidade.

O projeto também disponibiliza um *website* conforme a Figura 4, em que é feito uma integração com o *Google Maps*. É possível visualizar o local dos edifícios já catalogados e também verificar informações, como história do local e fotos. Por fim, também é possível sugerir novos edifícios, bastando preencher um formulário com as informações solicitadas.

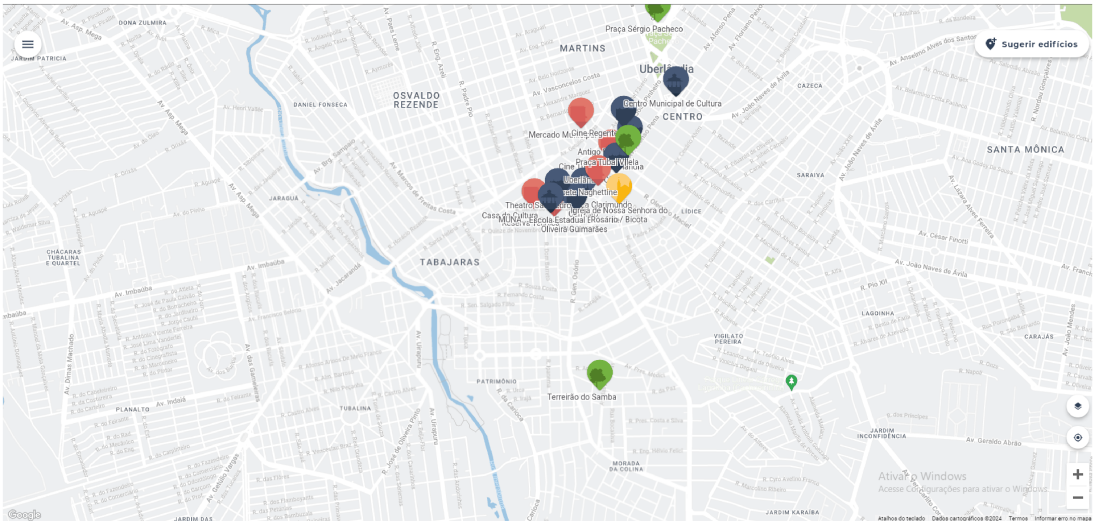

Figura 4 – Mapa do Retomba.

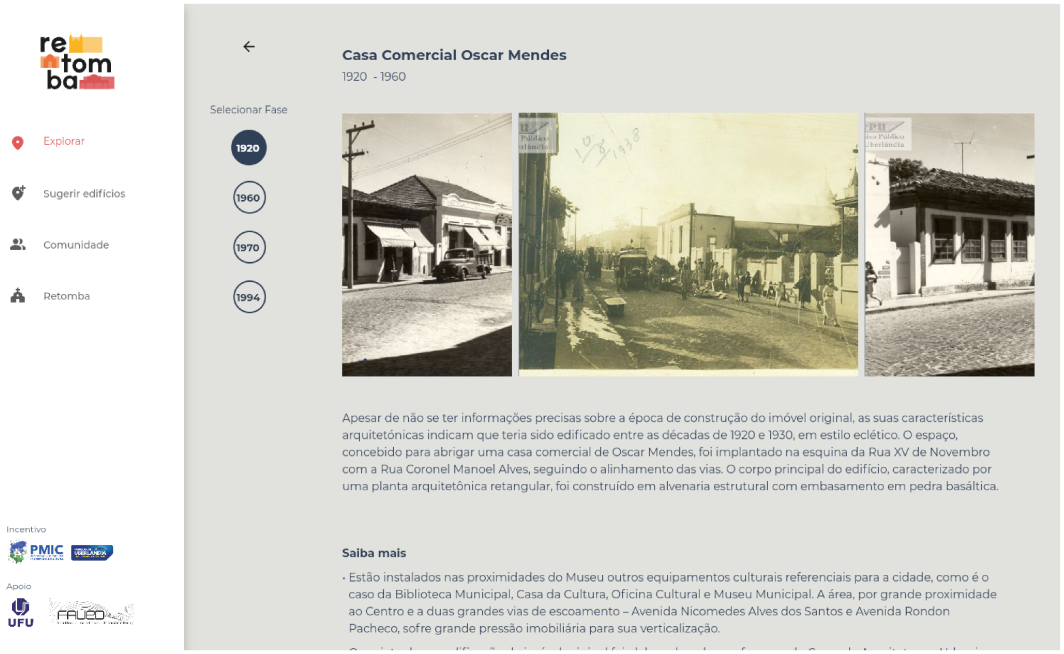

Figura 5 – Detalhes de uma edificação – Retomba.

2.2 Tecnologias utilizadas

Nesta são serão abordadas as tecnologias que foram utilizadas no desenvolvimento da aplicação.

2.2.1 Java

Java é uma linguagem de programação de alto nível e orientada a objetos criada na década de 1990 pela Sun Microsystems (agora propriedade da Oracle). É amplamente utilizada para criar aplicações de software para computadores, servidores e dispositivos móveis. Java é conhecida por ser uma linguagem de programação de propósito geral, o que significa que pode ser usada para criar quase qualquer tipo de software. Além disso, conforme (Sun Microsystems, 1996), "o Java permite que desenvolvedores de aplicativos escrevam um programa uma vez e depois seja capaz de executá-lo em qualquer lugar na Internet".

2.2.2 Spring Boot

"O Spring torna a programação em Java mais rápida, fácil e segura para todos", (Spring Boot, 2024). Spring Boot é um *framework* de aplicação Java que facilita o desenvolvimento de aplicativos de maneira rápida e fácil, fornecendo uma série de configurações e dependências pré-definidas para a criação de aplicativos de micro-serviço. Além disso, o *Spring Boot* oferece recursos como auto-configuração, facilidade de uso e gerenciamento

de dependências, o que permite que os desenvolvedores se concentrem mais na lógica de negócio do aplicativo em vez de se preocupar com configurações complexas.

2.2.3 JavaScript

JavaScript é uma linguagem de programação interpretada em tempo real com funções que são definidas como variáveis (Mozilla Developer Network, 2024). A linguagem de alto nível foi originalmente criada para ser usada principalmente em páginas da *web* para adicionar interatividade aos sites, mas agora é amplamente utilizada em aplicativos *web*, bem como em aplicativos móveis e até mesmo em alguns sistemas operacionais. Ela pode ser usada para criar aplicativos, adicionar efeitos animados às páginas da *web*, validar formulários de entrada de dados e muito mais.

2.2.4 Angular

Angular é uma plataforma de desenvolvimento construída em TypeScript, consistindo em mais de 1,7 milhão de autores e desenvolvedores, utilizando-a para o desenvolvimento de aplicativos *web* (Angular, 2024). O Angular possui uma série de ferramentas e recursos, como uma estrutura de componentes, gerenciamento de rotas, *bind* de dados, validação de formulários e muito mais. O *framework* pode ser utilizado para o desenvolvimento de projetos pessoais até aplicações empresariais.

2.2.5 Bootstrap

O Bootstrap é um *framework* de front-end que simplifica o desenvolvimento de sites responsivos, fornecendo uma biblioteca de componentes e estilos pré-projetados. Ele também inclui um sistema de grid flexível e componentes JavaScript interativos. De acordo com (Bootstrap, 2024), "Bootstrap se tornou uma das estruturas de *front-end* e projetos de código aberto mais populares do mundo.'

2.2.6 PostgreSQL

PostgreSQL é um **SGBD** de código aberto, bastante robusto e confiável. É baseado na estrutura de modelo relacional, em que os dados são armazenados em tabelas e interligados por meio de chaves, sendo elas primárias ou estrangeiras. Esse modelo é utilizado fortemente ainda hoje pela industria, corroborando a solidez dos SGBDs relacionais. Outro ponto, é que por se tratar de um sistema de código aberto, o PostgreSQL não possui nenhum custo por sua utilização e pode ser utilizado livremente de acordo com Postgresql (2023).

2.3 Modelagem de Software

A modelagem de software é o processo de criar representações visuais e/ou lógicas de um sistema, facilitando a compreensão do mesmo. Além disso, ela é usada para construir e validar o sistema que será criado.

Esta etapa é muito importante pois irá auxiliar na criação do projeto, documentação e compreensão, colocando tudo aquilo que foi proposto em prática e validar ao fim todas as funcionalidades.

Para a construção desta modelagem, será utilizado a linguagem UML, que por sua definição, trata-se de uma linguagem unificada voltada para modelagem e de acordo com (LARMAN, 2005), "uma linguagem para especificar, visualizar, construir e documentar os artefatos de sistemas de software".

2.3.1 Diagrama de Casos de Uso

O Diagrama de Casos de Uso é utilizado para descrever um conjunto de ações no qual um usuário irá iterar sobre o sistema (UML, 2022). Os usuários são referenciados como atores e são caracterizados por desenhos de bonecos. Os casos de usos são essenciais para uma visão gráfica e ampla da relação entre um usuário e o sistema para cada requisito funcional. Antes de tudo, deverá ser definido todos os requisitos funcionais que o sistema deverá comportar e cada requisito será um caso de uso. O diagrama é representado pelo usuário seguido de uma flecha apontando para uma elipse, no qual irá conter o nome do caso de uso. Após a representação, deverá ser criado uma tabela na qual irá especificar o caso.

2.4 Modelagem de Dados

A modelagem de dados é o processo no qual será criado formas de representação visual do gerenciamento das informações de um sistema. Essa etapa é de suma importância, pois, ela irá reduzir os erros na criação do sistema e facilitará e agilizará o desenvolvimento do banco de dados (Amazon - AWS, 2024).

2.4.1 Diagrama Entidade Relacionamento

O Diagrama Entidade Relacionamento (**DER**) é a abstração gráfica geral do banco de dados relacional da aplicação. Com ele, tem-se uma ampla visão de todas as tabelas, atributos e seus relacionamentos.

O modelo foi proposto por Peter Chen em 1976 (CHEN, 1976). Originalmente o modelo utiliza-se de formas geométricas como retângulos, elipse e losango. Com es-

sas formas, é possível modelar um banco de dados de forma simples, prática e de fácil compreensão.

Os retângulos representam as tabelas que serão criadas e as elipses representam os atributos que serão vinculados às respectivas tabelas. Já os losangos representam as relações em que as tabelas possuem uma às outras. Essas relações são ligadas por meio de linhas e entre elas são definidos suas cardinalidades. A cardinalidade indicará o grau de relação entre as entidades, fazendo com que as chaves de uma entidade possa migrar para outras entidades e até mesmo gerar novas tabelas.

Com base no conceito da cardinalidade, existem três tipos de relacionamentos, sendo eles:

- Um para um (1:1): em que uma informação só irá poder referenciar à uma outra informação.
- Um para muitos (1:N): em que uma informação poderá referenciar várias outras informações.
- Muitos para muitos (N:M): em que várias informações poderão referenciar à várias outras informações.

2.4.2 Modelo Relacional

O **MR** (Modelo Relacional), tem como objetivo apresentar uma visão mais técnica do banco de dados. Esse por sua vez é caracterizado por formas de tabelas ligadas uma a outras.

Uma tabela é denominada como uma entidade e irá armazenar diversas informações a respeito de tal, como exemplo, um patrimônio seria a entidade e teria informações como nome, descrição, etc. Essas informações são denominadas de atributos, devem ser unicamente atômicas e serão representadas como as colunas da tabela. Pelo menos um dos atributos deverá ser representado como uma chave primária (PK), não poderá ser vazia e não poderá se repetir. Outros atributos podem ser representados por uma chave estrangeira (FK). As chaves estrangeiras referenciam informações de outras ou da mesma tabela. Essas por sua vez, podem ou não estarem vazias. Para que as chaves estrangeiras possam existir, é preciso definir todos os relacionamentos no qual irá existir entre as tabelas.

3 Desenvolvimento

Este capítulo tem como objetivo descrever o desenvolvimento do sistema como um todo, utilizando-se das tecnologias que foram abordadas e metodologias da modelagem de dados e *softwares*.

3.1 Visão geral das etapas de desenvolvimento

As seguintes etapas foram realizadas para o desenvolvimento do software.

- **Modelagem de Software:** A modelagem do software foi realizada a partir da especificação dos requisitos funcionais e não funcionais, elaboração dos casos de usos e diagramas de classes.
- **Implementação do Banco de Dados**: Após realizada a modelagem do software, foi construído e implementado o banco de dados relacional, utilizando PostgreSQL.
- **Desenvolvimento da Aplicação:** Com base na modelagem do software, foi iniciado o desenvolvimento do software, utilizando linguagem Java e alguns *frameworks Spring* para o desenvolvimento da **API** *back-end*. Para o *front-end*, foi utilizado tecnologias como *javascript*, Angular, **CSS**, etc.
- **Implantação do Sistema:** Após o desenvolvimento concluído e testes realizados, a aplicação ficou pronta para ser repassada e subir para produção.

As seções seguintes detalham as atividades desenvolvidas durante a execução do projeto.

3.2 Requisitos Funcionais

A definição dos requisitos funcionais será a base para o início deste projeto. São os requisitos que irão dizer o que o sistema deverá contemplar para atendar às necessidades do cliente. Conforme a Tabela 1, foi definido os seguintes requisitos:

	Requisitos
1	Permitir que o usuário/administrador realize login.
2	Permitir que usuário/administrador realize logout.
3	Permitir que o usuário/administrador redefina sua senha.
4	Permitir que administrador realize o cadastro de outros usuários.
5	Permitir que o administrador redefinir a senha de outros usuários.
6	Permitir que administrador ative/inative outros usuários.
7	Permitir que o administrador edite informações de outros usuários.
8	Permitir que o usuário/administrador edite suas informações.
9	Permitir que o usuário/administrador cadastre um patrimônio.
10	Permitir que o usuário/administrador edite um patrimônio.
11	Permitir que o usuário/administrador ative/inative um patrimônio.
12	Permitir que o usuário/administrador busque um patrimônio.
13	Permitir que o usuário/administrador visualize os patrimônios cadastrados.

Tabela 1 – Requisitos do software.

3.2.1 Diagrama de Casos de Uso

Esta aplicação terá dois atores, sendo eles o Usuário e o Administrador. Esses são representados por ilustrações de bonecos com setas ligando à elipses, indicando uma ação do ator e a funcionalidade.

Usuário: O usuário é aquele quem irá utilizar o sistema, realizando o cadastramento e manutenção dos patrimônios e podendo realizar o gerenciamento de suas próprias informações.

Administrador: O administrador é um usuário especial, com mais poderes e responsabilidades. Ele será responsável por gerenciar os usuários em geral.

A Figura 6, representa todas as funcionalidades que um usuário e administrador possa ter dentro do sistema. Como observado, a figura representa cada ação que um ator poderá tomar utilizando o sistema. Nas próximas subseções, serão especificadas com mais detalhes tais ações, casos de sucesso e falhas.

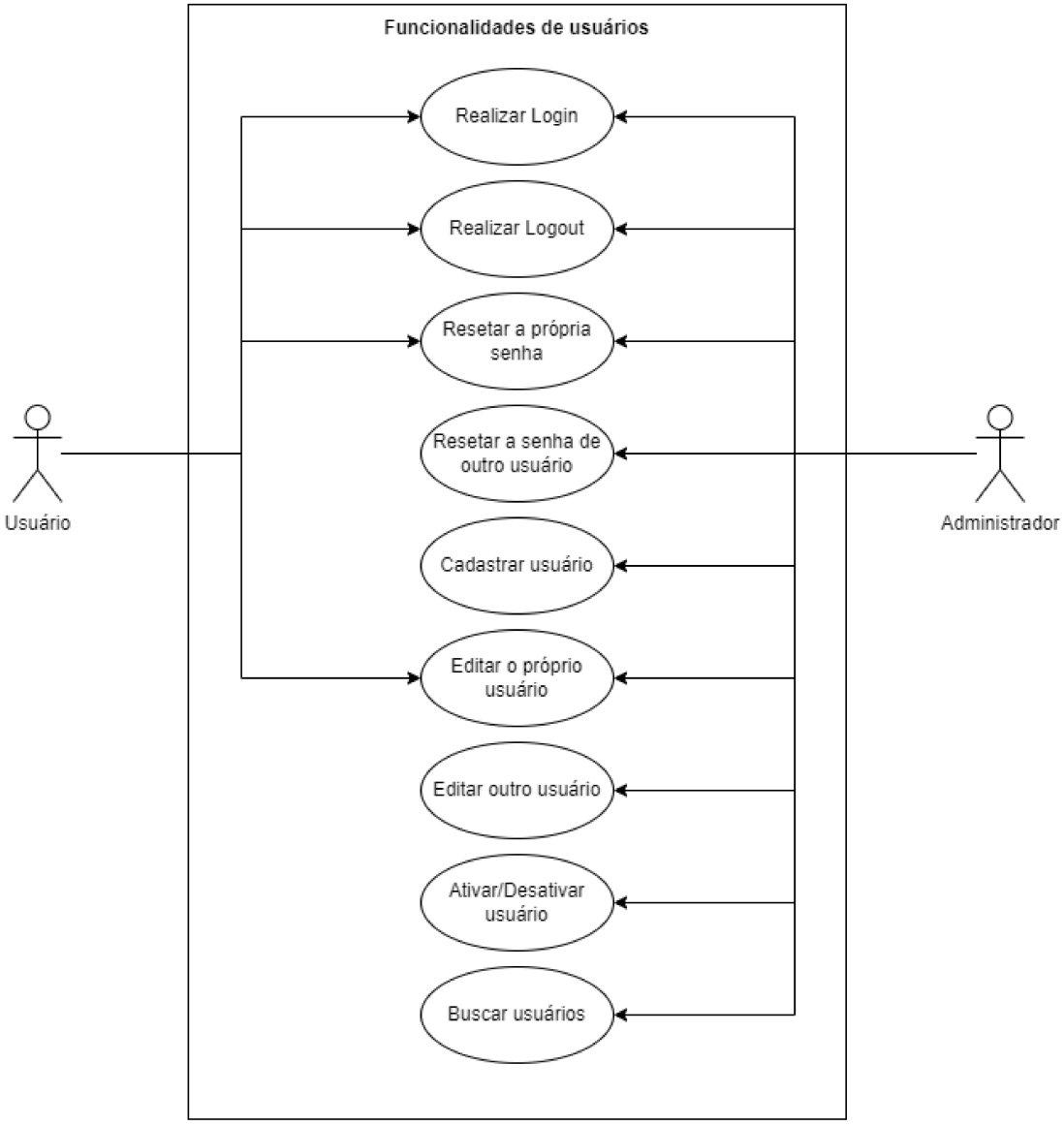

Figura 6 – Diagrama de casos de uso dos requisitos relacionados ao gerenciamento dos usuários.

3.2.1.1 Realizar login

A Tabela 2, representa o caso de uso especificado para tratar a ação de efetuar a entrada na aplicação.

Caso de uso	Realizar Login
Ator	Usuário/Administrador
Descrição	Processo no qual o usuário/administrador realizará a autenticação no sistema.
Pré-condições	O usuário/administrador deverá estar cadastrado previamente no sistema.
Pós-condições	O usuário/administrador estará autenticado e com as suas funcionalidades habilitadas.
Fluxo básico	
1	Usuário irá acessar a página do sistema.
2	Usuário irá inserir suas credenciais nos campos de login.
3	Usuário irá apertar o botão entrar.
4	O sistema irá realizar a validação das credenciais.
5	O sistema irá autenticar o usuário e irá, redireciona-lo para página inicial.
Fluxo Alternativo	
1	Se as credenciais fornecidas estiverem incorretas, o sistema irá exibir uma mensagem de login incorreto e pedirá para tentar novamente.

Tabela 2 – Login.

3.2.1.2 Realizar Logout

O caso de uso da Tabela 3 irá tratar a ação de efetuar a saída na aplicação.

Caso de uso	Realizar Logout
Ator	Usuário/Administrador
Descrição	Processo no qual o usuário/administrador realizará o logout no sistema.
Pré-condições	O usuário/administrador deverá estar autenticado no sistema.
Pós-condições	O usuário/administrador estará desautenticado, sem as suas funcionalidades habilitadas e será direcionado para tela de login.
Fluxo básico	
1	Usuário irá clicar no botão de opções.
2	Usuário irá clicar no botão sair.
3	O sistema irá desautenticar o usuário/administrador e direcioná-lo para tela de login.

Tabela 3 – Logout.

3.2.1.3 Redefinição de senha

Conforme a Tabela 4, o caso de uso irá tratar a ação de efetuar a redefinição da própria senha na aplicação pelo usuário/administrador.

Caso de uso	Realizar a redefinição da própria senha.
Ator	Usuário/Administrador
Descrição	Processo no qual o usuário/administrador realizará a redefinição de sua senha.
Pré-condições	O usuário/administrador deverá estar autenticado.
Pós-condições	A senha do usuário/administrador será alterada para uma nova.
Fluxo básico	
1	Usuário/administrador irá acessar o sistema.
2	Usuário/administrador irá clicar no opções e em seguida, em perfil.
3	Usuário/administrador irá clicar na aba de redefinição de senha.
4	O usuário/administrador irá clicar no botão editar, inserir uma nova senha e clicar em atualizar.
5	O sistema irá validar os dados do usuário/administrador e definir a nova senha fornecida.
Fluxo Alternativo	
	O usuário poderá contatar um administrador do sistema para que o mesmo realize a redefinição de sua senha.

Tabela 4 – Redefinição da própria senha.

3.2.1.4 Redefinição de senha - Administrador

Conforme a Tabela 5, o caso de uso irá tratar a ação de efetuar a redefinição de senha de um usuário na aplicação pelo administrador.

Caso de uso	Realizar resete de senha de outro usuário.
Ator	Administrador
Descrição	Processo no qual o administrador realizará a redefinição de senha de outro usuário.
Pré-condições	O administrador deverá estar autenticado no sistema.
Pós-condições	Senha estará redefinida.
Fluxo básico	
1	Administrador irá acessar o sistema.
2	Administrador irá clicar no botão menu e depois em usuários.
3	Administrador irá procurar o usuário que a senha será redefinida.
4	Administrador irá clicar no botão redefinir senha.
5	O sistema irá gerar uma nova senha aleatória e realizar a alteração.
6	O sistema irá informar a nova senha gerada através de um pop-up.
Fluxo Alternativo	
	O usuário poderá realizar a redefinição de sua senha através da tela de redefinição de senha.

Tabela 5 – Redefinição de senha – Administrador.

3.2.1.5 Cadastrar usuário

O caso de uso especificado na Tabela 6, irá tratar a ação de um administrador do sistema, cadastrando um novo usuário.

Caso de uso	Cadastrar usuário.
Ator	Administrador
Descrição	Processo no qual o administrador do sistema realizará o cadastro de um novo usuário.
Pré-condições	O administrador deverá estar autenticado no sistema.
Pós-condições	Estará criado um novo usuário.
Fluxo básico	
1	Administrador irá logar no sistema.
2	Administrador irá acessar a tela de usuários através do botão menu e usuários.
3	Administrador irá acessar a aba cadastrar usuário e preencher algumas informações do usuário a ser criado.
4	Administrador irá informar o login e senha do novo usuário.

Tabela 6 – Cadastrar usuário.

3.2.1.6 Inativar usuário

O caso de uso especificado na Tabela 7, irá tratar a ação de efetuar a inativação de um usuário pelo administrador do sistema.

Caso de uso	Inativar usuário.
Ator	Administrador
Descrição	Processo no qual o administrador inativará um usuário.
Pré-condições	O administrador deverá estar autenticado no sistema.
Pós-condições	Determinado usuário estará inativo.
Fluxo básico	
1	Administrador irá logar no sistema.
2	Administrador irá acessar a tela de usuários através do botão menu e usuários.
3	Administrador irá procurar o usuário que será inativado.
4	Administrador irá clicar no botão ativo.

Tabela 7 – Inativar Usuário.

3.2.1.7 Editar outros usuários

O caso de uso especificado na Tabela 8, irá tratar a ação de um administrador realizar a edição de um usuário no sistema.

Caso de uso	Editar outros usuários.
Ator	Administrador
Descrição	Processo no qual o administrador irá editar suas informações.
Pré-condições	O adminstrador deverá estar autenticado no sistema.
Pós-condições	Usuário possuirá novas informações.
Fluxo básico	
1	Administrador irá logar no sistema.
2	Administrador irá acessar a tela de usuários através do botão menu e usuários.
3	Administrador irá clicar no botão editar.
4	Administrador irá editar as informações e clicar em enviar.

Tabela 8 – Editar outros usuários.

3.2.1.8 Editar o próprio usuário

O caso de uso especificado na Tabela 9, irá tratar a ação de um usuário/administrador realizar a edição do seu cadastro no sistema.

Caso de uso	Editar usuário.
Ator	Usuário/Administrador
Descrição	Processo no qual o usuário/administrador irá editar suas informações.
Pré-condições	O usuário/administrador deverá estar autenticado no sistema.
Pós-condições	Usuário/administrador possuirá novas informações.
Fluxo básico	
1	Usuário/administrador irá autenticar no sistema.
2	Usuário/administrador irá clicar no botão opções e em seguida, em perfil.
3	Usuário/administrador irá clicar no botão editar.
4	Usuário/administrador irá editar suas informações.
5	Usuário/administrador irá clicar em enviar.

Tabela 9 – Editar o próprio usuário.

A Figura 7 representa todas as funcionalidades que um usuário e administrador possuem referente aos patrimônios.

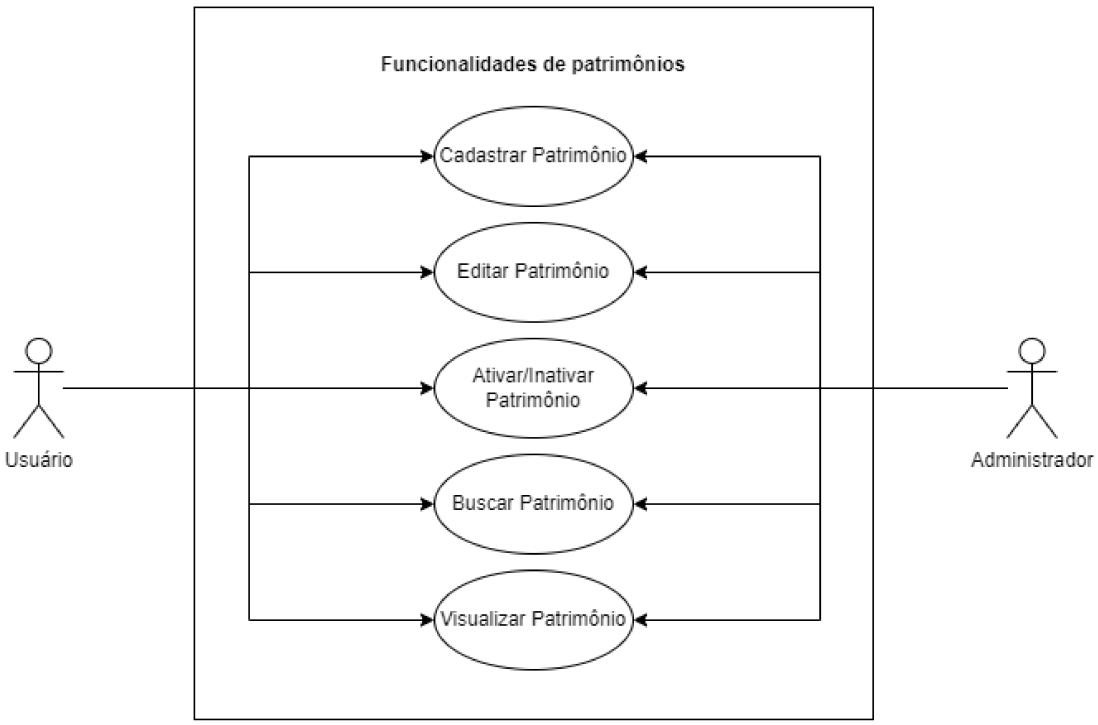

Figura 7 – Diagrama de casos de uso dos requisitos relacionados à gerenciamento dos patrimônios.

3.2.1.9 Cadastrar Patrimônio

O caso de uso especificado na Tabela 10, irá tratar a ação de um usuário efetuar o cadastro de um novo patrimônio.

Caso de uso	Cadastrar patrimônio
Ator	Usuário
Descrição	Processo no qual o usuário irá cadastrar um novo patrimônio.
Pré-condições	O usuário deverá estar logado no sistema.
Pós-condições	Novo patrimônio estará cadastrado.
Fluxo básico	
1	Usuário irá logar no sistema.
2	Usuário irá clicar no botão novo patrimônio.
3	Usuário irá inserir as informações do patrimônio.
4	Usuário irá clicar em salvar.

Tabela 10 – Cadastrar Patrimônio.

3.2.1.10 Editar Patrimônio

O caso de uso especificado na Tabela 11, irá tratar a ação de um usuário efetuar a edição de um patrimônio já existente.

Caso de uso	Editar patrimônio
Ator	Usuário
Descrição	Processo no qual o usuário irá editar um patrimônio existente.
Pré-condições	O usuário deverá estar logado no sistema.
Pós-condições	Patrimônio irá conter novas informações.
Fluxo básico	
1	Usuário irá logar no sistema.
2	Usuário irá buscar pelo patrimônio.
3	Usuário irá clicar no botão de editar patrimônio.
4	Usuário irá inserir as informações a serem alteradas.
5	Usuário irá clicar em salvar.

Tabela 11 – Editar Patrimônio.

3.2.1.11 Ativar/inativar Patrimônio

O caso de uso especificado na 12, irá tratar a ação de um usuário remover um patrimônio cadastrado.

Caso de uso	Ativar/inativar patrimônio.
Ator	Usuário
Descrição	Processo no qual o usuário irá remover um patrimônio existente.
Pré-condições	O usuário deverá estar logado no sistema.
Pós-condições	Patrimônio estará removido.
Fluxo básico	
1	Usuário irá logar no sistema.
2	Usuário irá buscar pelo patrimônio.
3	Usuário irá clicar no botão de remover patrimônio.
4	Usuário irá clicar no botão de confirmar.

Tabela 12 – Remover Patrimônio.

3.2.1.12 Visualizar Patrimônios

O caso de uso especificado na Tabela 13, irá tratar a ação de um usuário visualizar os patrimônios cadastrados.

Caso de uso	Visualizar patrimônios
Ator	Usuário
Descrição	Processo no qual o usuário irá visualizar os patrimônios cadastrados.
Pré-condições	O usuário deverá estar logado no sistema.
Pós-condições	Visualizar patrimônios
Fluxo básico	
1	Usuário irá logar no sistema.
2	Usuário irá clicar no botão de patrimônios.

Tabela 13 – Visualizar Patrimônio.

3.2.1.13 Buscar Patrimônio

O caso de uso especificado na Tabela 14, irá tratar a ação de um usuário consultar patrimônios cadastrados.

Caso de uso	Buscar patrimônio
Ator	Usuário
Descrição	Processo no qual o usuário irá buscar os patrimônios cadastrados.
Pré-condições	O usuário deverá estar logado no sistema.
Pós-condições	Visualizar patrimônios buscados.
Fluxo básico	
1	Usuário irá logar no sistema.
2	Usuário irá clicar no botão de patrimônios.
3	Usuário irá selecionar o filtro desejado ou buscar por texto.
4	Sistema retornará a busca.

Tabela 14 – Buscar Patrimônio.

3.2.2 Modelagem do Banco de Dados

Para a modelagem dos dados, foi adotada uma abordagem inicial que consistiu no desenvolvimento do Diagrama de Entidade-Relacionamento (**DER**). Esse processo permitiu uma compreensão holística dos dados a serem armazenados, oferecendo uma representação visual das entidades, seus atributos e os relacionamentos entre elas. Na Figura 8, é possível observar detalhadamente essa estrutura, fornecendo percepções valiosas para a concepção e implementação do banco de dados. Essa etapa é fundamental para garantir a integridade e eficácia do sistema, pois orienta a definição precisa das tabelas, campos e associações necessárias para suportar os requisitos da aplicação de forma eficiente.

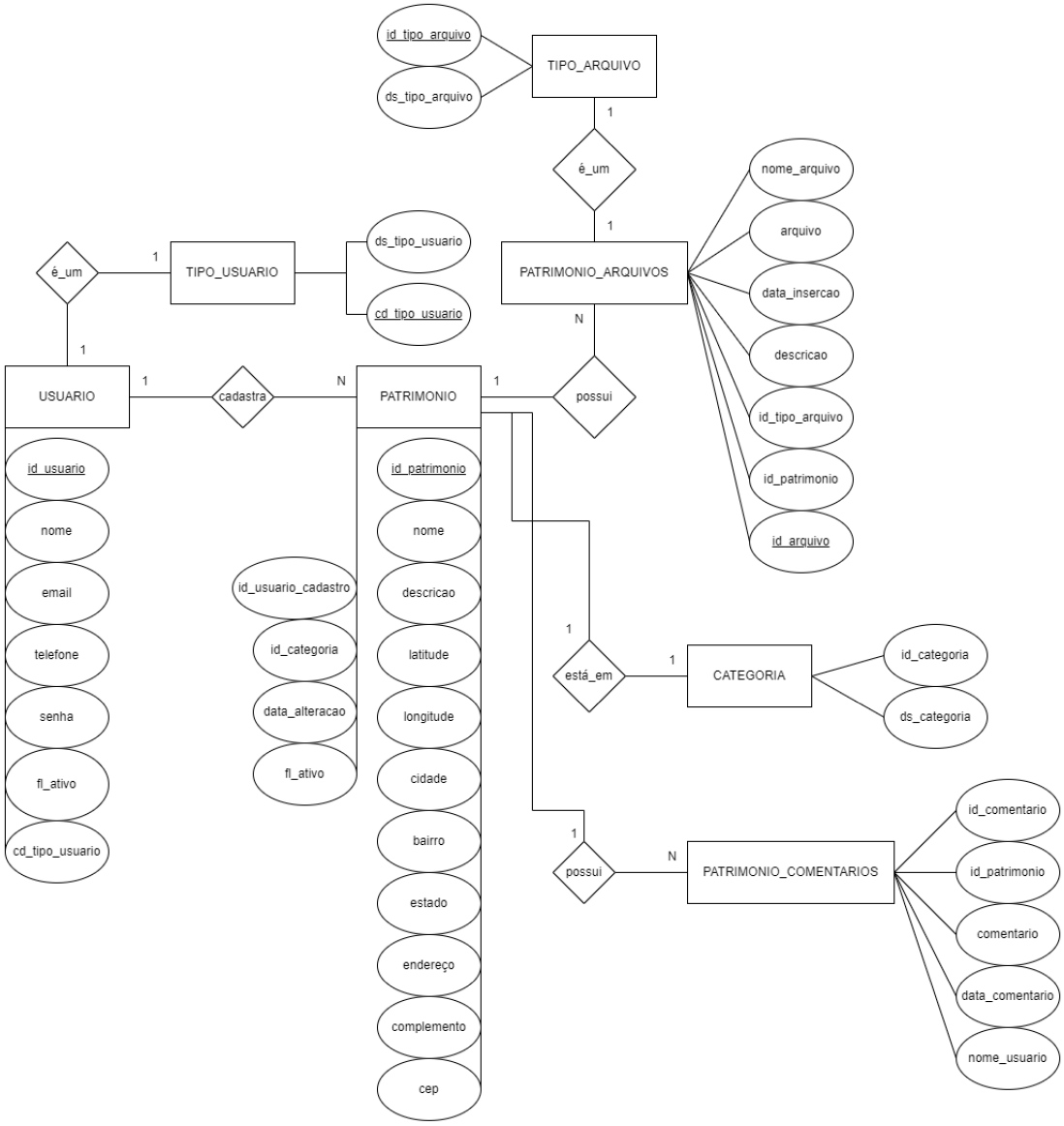

Figura 8 – Diagrama Entidade Relacionamento.

Após a conclusão do Diagrama de Entidade-Relacionamento (**DER**), foi imprescindível avançar para a concepção do Modelo Relacional (**MR**), que proporciona uma representação mais concreta da estruturação modelada anteriormente. Nessa etapa, é possível obter uma visualização detalhada de como as entidades, atributos e relacionamentos mapeados no **DER** serão efetivamente implementados no banco de dados. A Figura 9 oferece uma visão panorâmica e simplificada dessa transição, apresentando as informações anteriormente definidas na Figura 8 agora organizadas em formato tabular. Essa abordagem facilita a compreensão e validação da estrutura do banco de dados, além de servir como uma documentação para o desenvolvimento e manutenção do sistema.

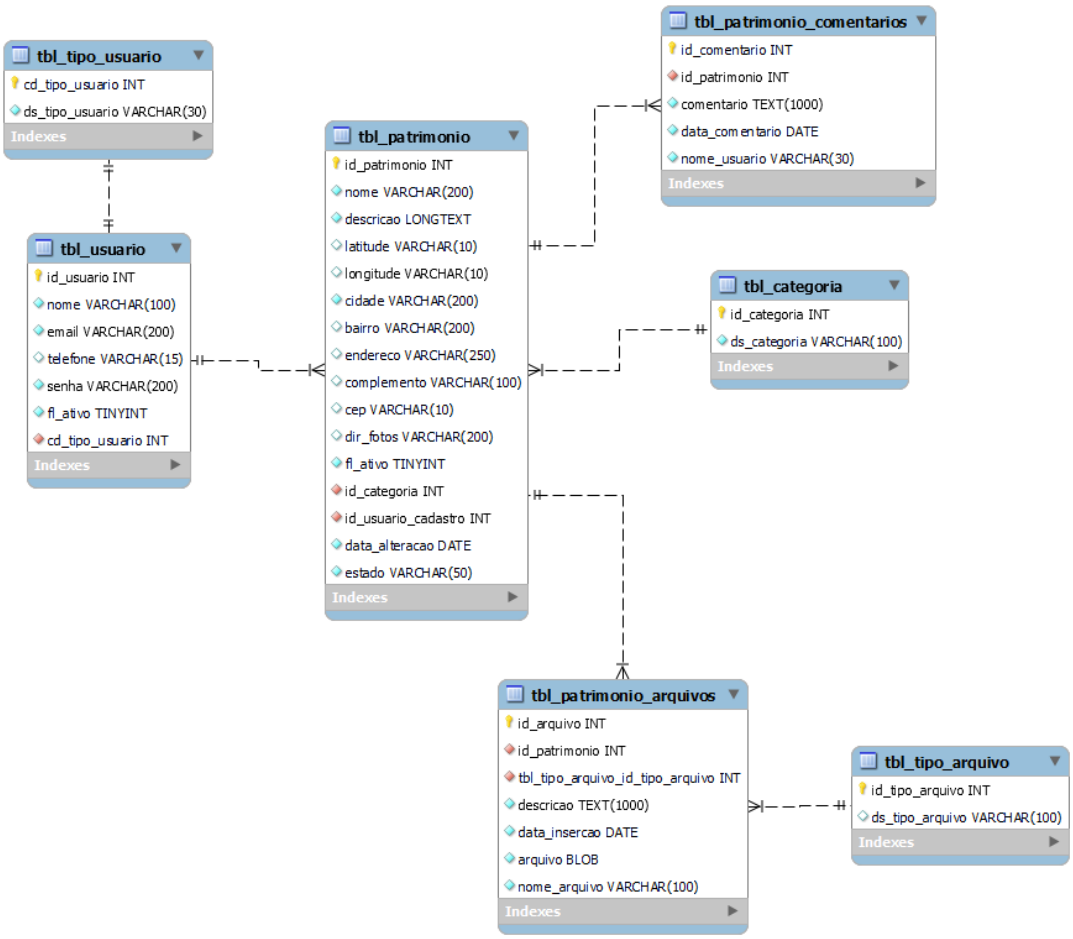

Figura 9 – Modelo Relacional.

3.3 *Back-end*

Para a criação do *back-end* foi utilizado a linguagem de programação Java, juntamente com o *framework Spring Boot*. Definimos o *back-end* como uma **API**, pois ela irá expor todos as interfaces na qual o *front-end* irá utilizar. A **API** foi criada de forma de limpa e direta, utilizando-se três camadas, sendo elas a *Controller*, *Service* e *Repository*.

No contexto de segurança, foi implementado o módulo *Spring Security Crypto* em conjunto com o padrão *JSON Web Token* (**JWT**). O *Spring Security Crypto* é responsável pela criptografia e descriptografia de senhas, enquanto o **JWT** tem como objetivo garantir a segurança na troca de informações entre a **API** e o **front-end**.

Para garantir essa segurança, de forma simplificada, o **JWT** gera um *token* utilizando informações como o identificador do usuário, o email e as chaves secretas da **API**. Além disso, são incluídos o momento em que o *token* é gerado e o tempo máximo de validade do mesmo. Como resultado, é gerada uma *string* dividida em três partes por pontos.

Essas medidas de segurança permitiram a implementação da autenticação dos usuários, verificação da autenticidade do acesso e garantia das permissões necessárias para realizar operações específicas, entre outras funcionalidades.

3.3.1 *Controllers*

As controladoras tem como objetivo expor os *end-points* de forma estruturada e organizada do sistema. A comunicação é realizada via protocolo **HTTP** e serve como a ponte de comunicação com o *front-end*. Os *end-points* possuem o mínimo de regra possível, pois, todas as regras de negócio são definidas dentro dos *services*.

Para distinguir uma *controller* de outra foi utilizado a notação do *spring* @RequestMapping. Essa notação irá definir a assinatura única da classe controladora. Juntamente, é utilizado também a @RestController, o qual define que determinada classe é uma controladora. Já os métodos da classe controladora são identificados como *endpoints*. Esses são assinados utilizando as notações @GetMapping, @PostMapping, @PutMapping e outros. Cadas uma dessas notações tem um propósito diferente, seguindo os protocolos **HTTP**. O conjunto entre o @RequestMapping e algum mapeamento de um método, resulta na geração de uma **URL** que será exposta para ser acessada e consumida. Como exemplo, a controladora de usuários do sistema foi definida com o barramento "/usuario", e o método responsável por efetuar a busca de usuários foi definido com o mapeamento "busca-usuarios". Logo, para consumir essa busca, a **URL** que deverá ser utilizada é "/usuario/busca-usuarios".

3.3.2 *Services*

Na camada de serviço encontra-se toda as regras de negócio para processamento das requisições. As classes *services* são definidas pela notação @Service do Spring. Nelas, são definidos vários métodos nos quais terão um objetivo especifico. Em geral, esses objetivos são para processar alguma informação, buscar ou gerenciar algum dado no banco de dados. Para isso, é necessário acionar a camada *repository*.

Na camada de serviço, encontram-se todas as regras de negócio responsáveis pelo processamento das requisições. As classes de serviço são definidas pela anotação @Service do *spring*. Nessas classes, são implementados diversos métodos, cada um com um propósito específico. Esses métodos geralmente têm como objetivo processar informações, realizar validações, coordenar transações e acessar a camada de persistência para buscar ou manipular dados no banco de dados.

Ao definir os métodos nas classes de serviço, é importante garantir que cada método tenha uma responsabilidade clara e bem definida, seguindo os princípios de coesão e baixo acoplamento. Isso facilita a manutenção do código e a compreensão das funcionalidades

implementadas em cada serviço.

A interação entre a camada de serviço e a camada de persistência geralmente ocorre por meio da injeção de dependência de classes de repositório. Os serviços utilizam os métodos dos repositórios para realizar operações de leitura, escrita e manipulação dos dados no banco de dados. Essa separação de responsabilidades entre a camada de serviço e a camada de persistência promove uma arquitetura mais organizada e modular, facilitando a implementação de novas funcionalidades e a manutenção do sistema como um todo.

3.3.3 *Repository*

Por fim, a camada de repositório desempenha um papel crucial ao integrar a lógica da aplicação com o Sistema Gerenciador de Banco de Dados (**SGBD**). Os *repositories*, marcados com a anotação @Repository, funcionam como uma ponte entre a **API** e o armazenamento de dados.

Para essa integração, foram adotadas duas estratégias distintas de implementação.

Na primeira abordagem, foram definidas interfaces abstraídas pelo *framework* **JPA**. Esse modelo oferece uma camada de abstração que simplifica o desenvolvimento, pois encapsula todo o fluxo de conexão, a criação de *queries* e a obtenção dos resultados. Isso permite uma implementação mais rápida e menos propensa a erros.

Já na segunda abordagem, optou-se por um método mais convencional, utilizando classes. Nesse caso, foi necessário desenvolver manualmente todo o processo de conexão com o banco de dados, elaboração de *queries* e manipulação dos resultados. Embora mais complexa, essa abordagem oferece maior flexibilidade, permitindo a execução de consultas mais elaboradas e a realização de atualizações específicas na base de dados.

Cada abordagem possui suas vantagens e foi escolhida de acordo com as necessidades específicas do projeto.

3.3.4 *Front-end*

Com a adoção do Angular, a estruturação desta camada foi refinada, adotando a abordagem de contêineres para uma organização mais eficiente. Cada contêiner representa uma página específica da aplicação e abrange não apenas a estruturação **HTML** e a estilização **CSS**, mas também a lógica de apresentação fornecida pelo componente correspondente.

Os componentes desempenham um papel crucial ao controlar dinamicamente a interface, adaptando-a conforme as interações do usuário. Eles encapsulam a lógica de apresentação e interação com o usuário, promovendo uma arquitetura modular e reutilizável.

Além disso, em determinadas circunstâncias, é possível implementar sub-contêineres para abstrair funcionalidades específicas, como é o caso da camada de serviço. Neste contexto, a camada de serviço assume a responsabilidade de gerenciar a comunicação com a **API**, garantindo uma separação clara de preocupações e facilitando a manutenção e escalabilidade do sistema.

Foi adotado o *framework Bootstrap* na criação das interfaces devido à sua robustez e facilidade de uso. Com a utilização do *Bootstrap*, conseguimos implementar de forma rápida e eficiente recursos como *carousels* para exibição de imagens, bem como garantir a responsividade dos contêineres em diferentes dispositivos.

Essa abordagem proporcionou um ambiente de desenvolvimento altamente organizado e flexível, permitindo a construção de interfaces de usuário sofisticadas e altamente interativas, enquanto manteve um código limpo e de fácil manutenção.

4 Resultados

Com o propósito de oferecer uma visão abrangente do sistema desenvolvido, este capítulo se destina a apresentar os resultados alcançados ao longo do processo de desenvolvimento. Além de descrever as funcionalidades implementadas, serão destacadas algumas páginas que exemplificam a interface e a experiência do usuário ao interagir com o sistema. Essa abordagem permitirá uma compreensão mais completa das características e do funcionamento do sistema.

4.1 Login

Iniciando pela tela de Login, representada na Figura 10, o usuário é recebido por uma interface minimalista projetada para facilitar o acesso ao sistema. Com um design simplificado, o formulário exibe apenas duas entradas: email e senha. Além disso, são apresentados dois botões - um para permitir a visualização da senha digitada e outro para iniciar o processo de login. Antes de qualquer interação, o sistema realiza uma verificação preliminar para detectar a presença de um *token* de autenticação armazenado no navegador. No caso de um *token* expirado ou ausente, ocorre uma limpeza no armazenamento local do navegador e o usuário é redirecionado imediatamente para realizar o login novamente.

Quanto à ação de login, ela envolve uma chamada **HTTP** para a **API** por meio da camada de serviço definida dentro de seu respectivo contêiner. As credenciais fornecidas são submetidas a processos de criptografia e validação para garantir sua autenticidade. Em caso de sucesso, um *token* válido é gerado e retornado ao navegador. Este *token* representa um passe de acesso que concede ao usuário a permissão para realizar consultas e operações futuras no sistema. Essa abordagem de autenticação e autorização é essencial para garantir a segurança e integridade dos dados durante a interação com o sistema.

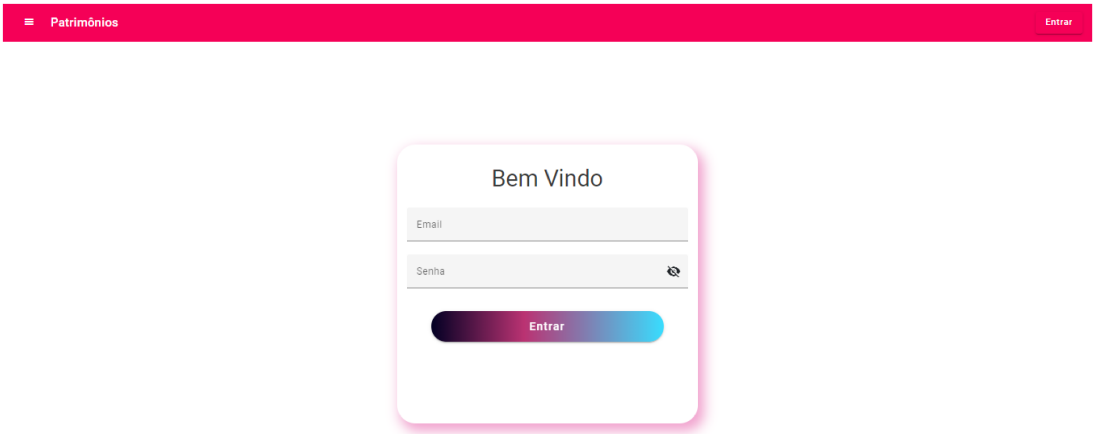

Figura 10 – Tela de Login

4.2 Gerenciamento de Patrimônios

A tela de gerenciamento de patrimônios foi segmentada em três partes por meio da funcionalidade de abas proporcionada pelo Angular. Essa abordagem possibilitou centralizar todas as funcionalidades em uma única página.

Conforme ilustrado na Figura 11, todas as opções de controle são facilmente acessíveis. Por exemplo, para editar um patrimônio, basta clicar no ícone de lápis. Isso abrirá um modal, conforme Figuras 12 e 13, contendo as informações atuais do patrimônio e os arquivos associados. No modal, é viável editar informações, remover ou adicionar arquivos. Para isso, basta realizar as edições no formulário e enviar, clicar no botão de remoção (X) do arquivo para excluí-lo, ou selecionar arquivos para adicionar.

Figura 11 – Tela de Gerenciamento de Patrimônios

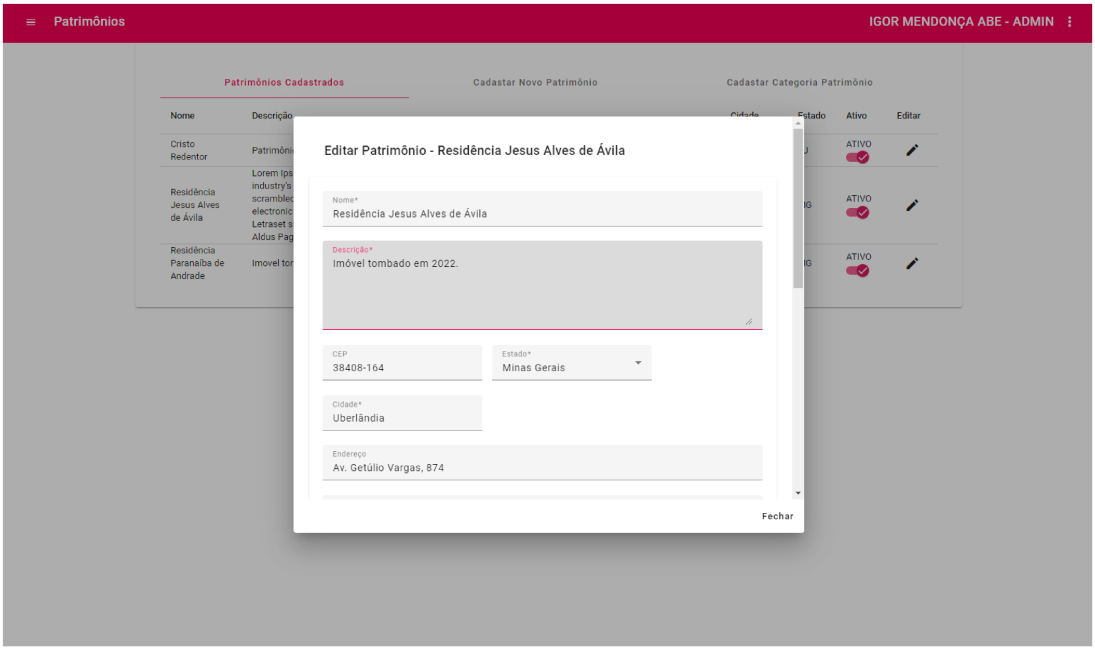

Figura 12 – Tela de edição de patrimônio.

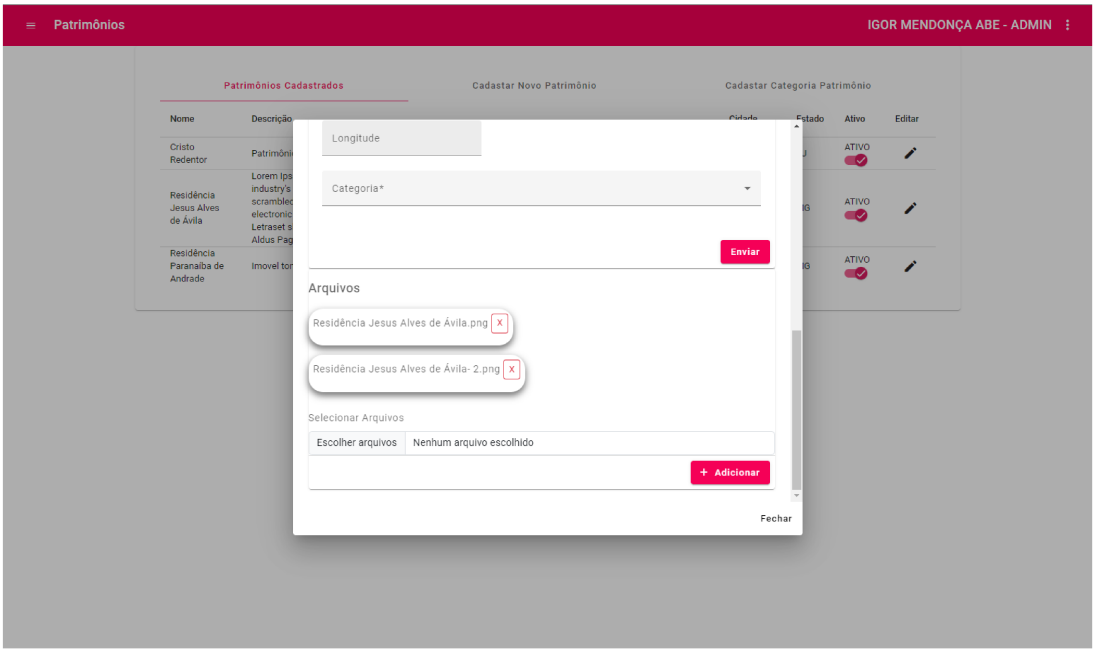

Figura 13 – Continuação da tela de edição de patrimônio.

Para realizar o cadastro, o processo é bastante simples. Conforme ilustrado na Figura 14, é apresentado um formulário completo com todas as informações necessárias, além da opção de fazer o *upload* de arquivos logo abaixo.

Com a utilização do *Bootstrap*, foi utilizado a funcionalidade de formulário para múltiplos arquivos. Isso permite selecionar e enviar vários arquivos simultaneamente, tornando o processo ainda mais eficiente.

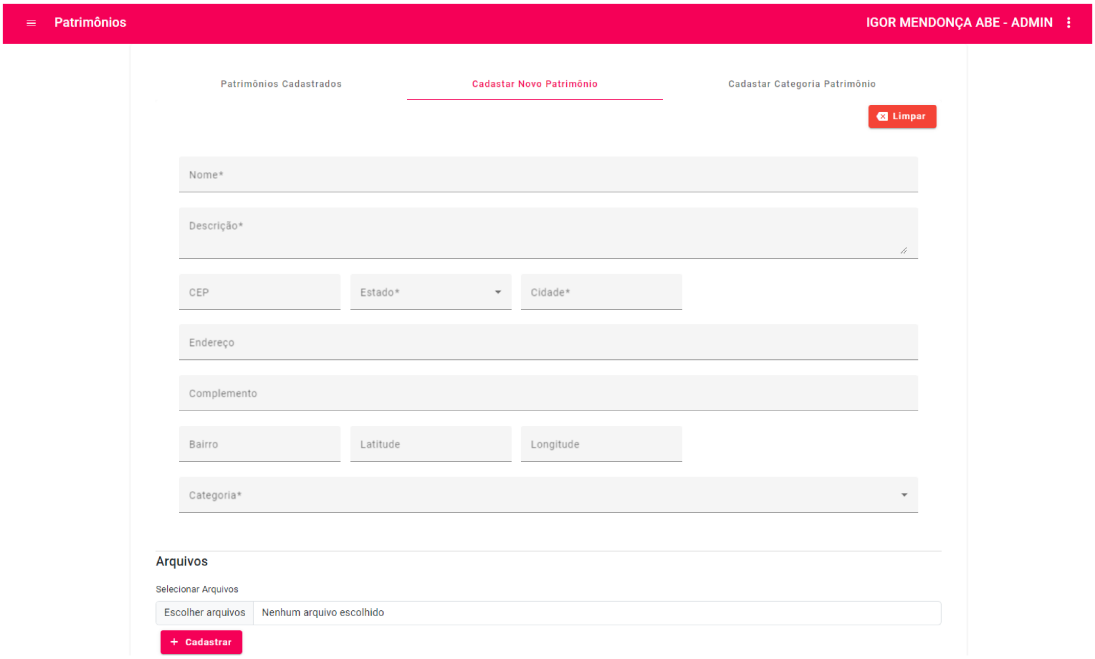

Figura 14 – Tela de Gerenciamento de Patrimônios - Cadastrar

Por fim, na Figura 15, é apresentada a aba destinada ao cadastro das categorias que serão relacionadas aos patrimônios.

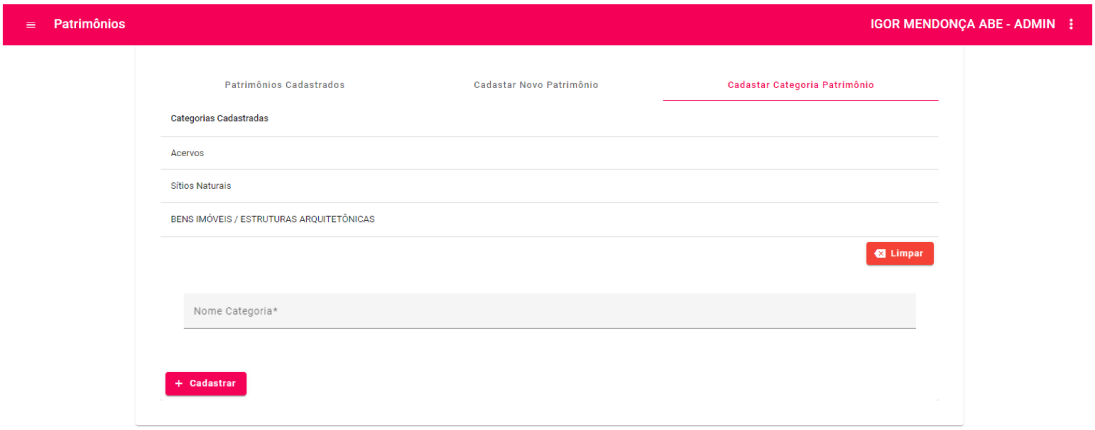

Figura 15 – Tela de Gerenciamento de Patrimônios - Categorias

4.3 Gerenciamento de Usuários

Assim como no desenvolvimento do Gerenciamento de Patrimônios, implementamos a mesma funcionalidade de abas e segmentamos também em três partes. Na Figura 16,

é possível observar todas as abas, sendo que esta em específico permite o cadastro de um novo usuário. É importante ressaltar que apenas usuários com perfil de Administrador têm acesso a esse gerenciamento.

Quando o cadastro é enviado e a requisição chega à **API**, primeiro é verificado se o *token* do usuário que enviou ainda é válido. Caso seja válido, é então validado se o usuário possui o perfil de Administrador e somente após isso o cadastro é efetuado ou não. Essas mesmas regras se aplicam a outras funcionalidades, como a ativação/desativação do usuário e consultas.

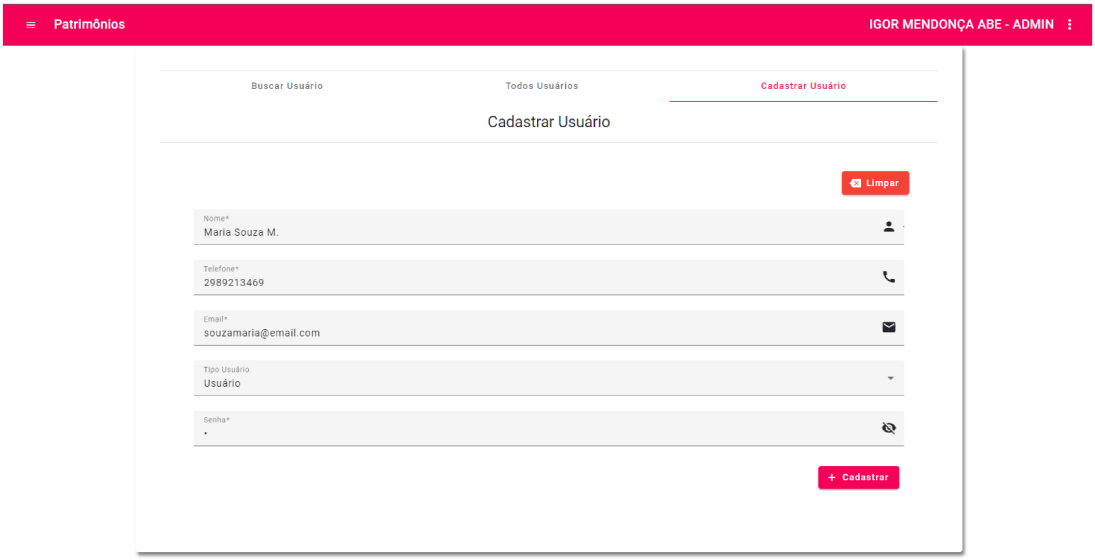

Figura 16 – Tela de Gerenciamento de Usuários - Cadastro de Usuário

No gerenciamento de usuários, também é possível realizar a edição, ativação e desativação, e a redefinição de senha. Conforme mostrado na Figura 17, foram implementados botões para essas funcionalidades.

Para ativar ou desativar um usuário de forma simples, basta clicar no botão de alternância (*slider*) na coluna Ativo. A redefinição de senha também é bastante simples e prática: basta clicar no botão indicado pela chave. Internamente, será gerada uma cadeia de caracteres aleatórios, criptografada e armazenada. Na tela, a senha será exibida para que o administrador possa enviá-la ao usuário. Quanto à edição do usuário, é disponibilizado um botão em forma de lápis. Ao clicar neste botão, será aberto um modal onde será possível realizar a edição dos dados do usuário.

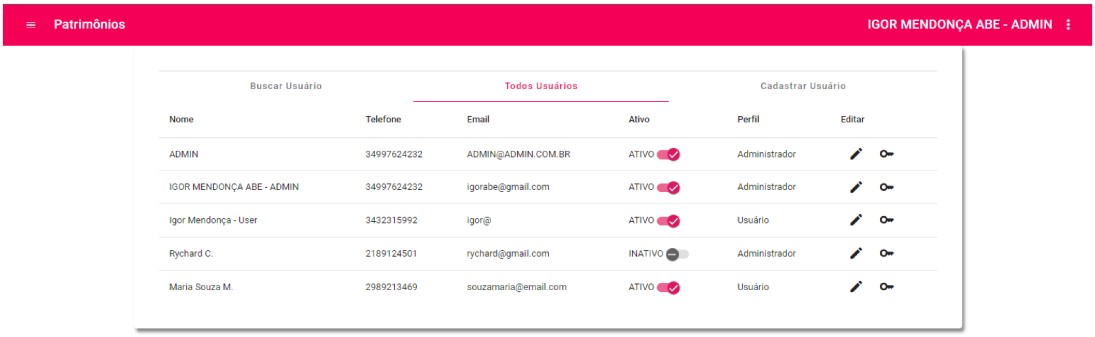

Figura 17 – Tela de Gerenciamento de Usuários – Todos Usuários.

Foi implementada uma aba para realizar a busca específica de usuários, conforme mostrado na Figura 18. Essa interface também inclui as mesmas funcionalidades implementadas anteriormente.

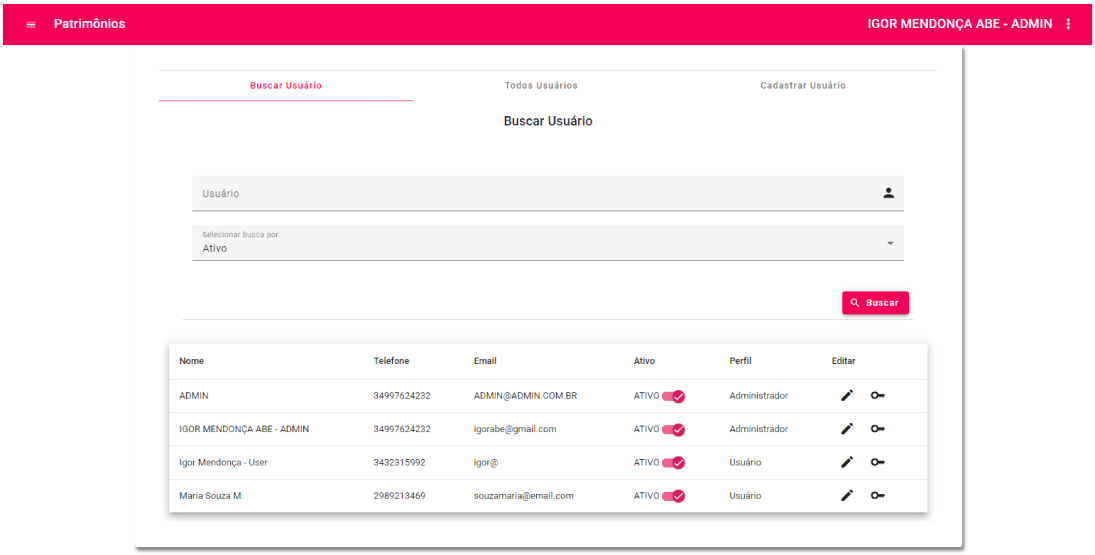

Figura 18 – Tela de Gerenciamento de Usuários – Buscar Usuários.

4.4 Visualização de um Patrimônio

A visualização de um patrimônio é acessível a qualquer pessoa e não requer login para ser acessada. Esta tela exibirá todas as informações relevantes de um patrimônio, como seus detalhes, arquivos anexados e comentários associados.

Conforme a Figura 19, logo de início foi implementado uma funcionalidade disponibilizada também pelo *Bootstrap*, o *Caroulsel*. Essa implementação permite que todas as imagens referentes ao patrimônio seja exibidas em um único quadro.

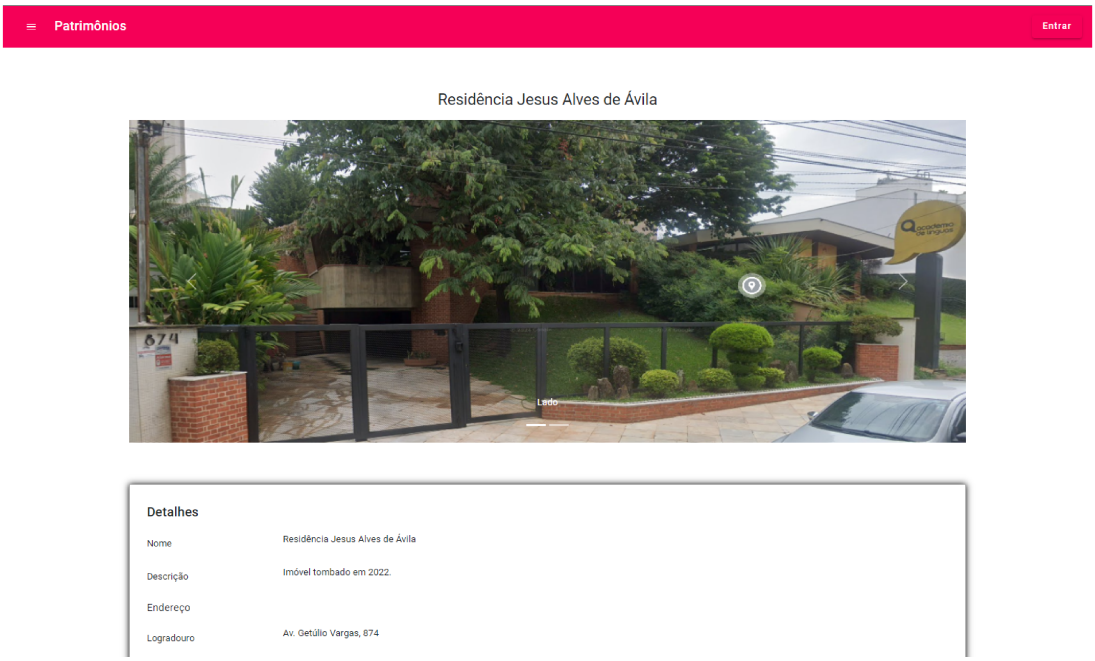

Figura 19 – Tela de patrimônio – dados do bem tombado.

Logo abaixo na interface, conforme as Figuras 20 e 21, são exibidos os detalhes, comentários e anexos. Comentários poderão ser enviados por qualquer pessoa, bastando apenas realizar a identificação com o nome. Caso seja algum usuário logado, o sistema irá identificar de forma automática e não deixará alterar o nome. Para evitar ocupar muito espaço na tela, foi implementado uma paginação para exibir no máximo 5 comentários por vez. Por fim, na parte de anexos são exibidos todos os arquivos associados ao patrimônio. Implementamos uma funcionalidade que permite o download dos anexos, bastando clicar sobre o anexo desejado.

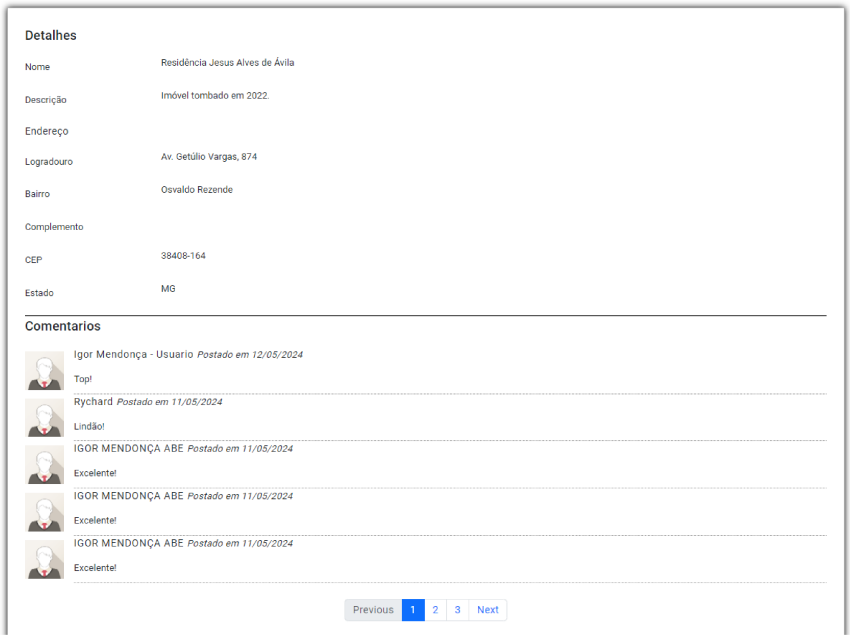

Figura 20 – Tela de patrimônio – comentários.

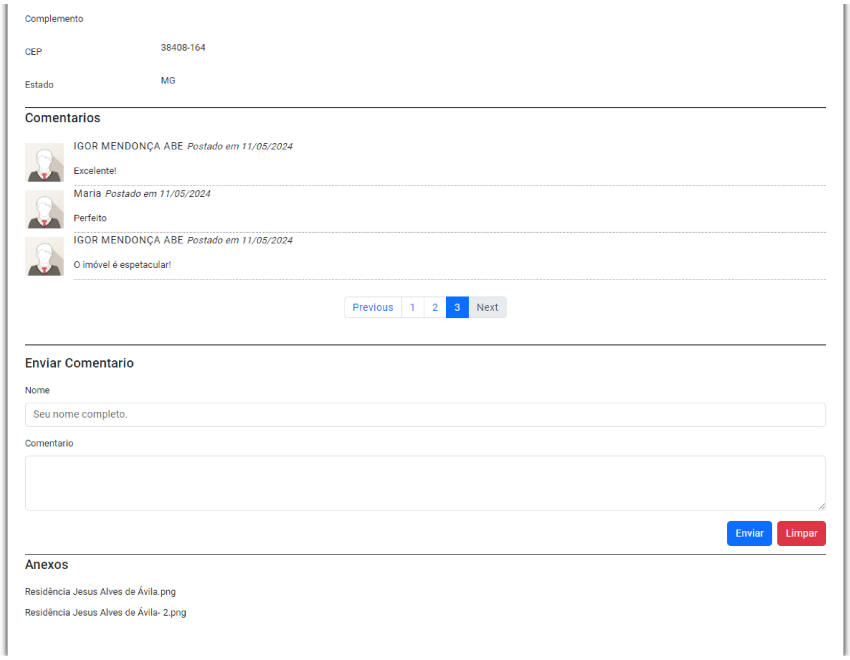

Figura 21 – Tela de patrimônio – envio de comentários.

5 Conclusão

Neste projeto foi explorado o tema dos patrimônios tombados, que por lei são bens culturais protegidos pelo Estado, afim da preservação da nossa história cultural. Para tanto, foi desenvolvido um sistema capaz de realizar o gerenciamento dos tombos e também disseminar a informação para o público geral.

Antes de realizar o desenvolvimento, foi realizado um levantamento dos requisitos funcionais que o sistema deveria comportar, e realizado o mapeamento dos casos de uso. Foi realizado também a criação da modelagem dos dados, diagramas no qual torna a compreensão do sistema muito mais fácil e contribui para o desenvolvimento tanto da aplicação quanto do banco de dados, agilizando assim o processo.

Durante o desenvolvimento, foi realizado a criação de todas as funcionalidades nas quais foram definidas nos casos de uso. Essas implementações estão presentes tanto na parte do *back-end* quanto no *front-end*, como, por exemplo, o simples caso de realizar a autenticação no sistema, primeiramente teve-se de criar uma página no *front-end*, contendo um formulário onde o usuário pudesse inserir suas credenciais. No *back-end*, foi desenvolvido a funcionalidade de receber essas informações inseridas pelo usuário, realizar toda a regra de negócio como criptografia e validação. Dentro dessa regra, teve de se implementar também a geração de um *token*, para que nas próximas requisições que o usuário realizar, seja validado a autenticidade do usuário e checagem de permissão para efetuar tal comando.

A implementação do sistema de patrimônios tombados não apenas simplificará a catalogação e manutenção desses bens, mas também contribuirá significativamente para conscientizar sobre a importância da preservação da história cultural. Ao disponibilizar o acesso à visualização para o público, o sistema torna-se uma ferramenta poderosa para divulgar e valorizar nossa herança cultural.

O desenvolvimento deste sistema destaca a relevância da preservação cultural em nosso país, que possui uma riqueza cultural incomparável que merece ser protegida, documentada e divulgada. Os possíveis próximos passos serão levar a proposta desse projeto para as autoridades públicas, mostrando todo o potencial no qual o sistema oferece e os benefícios para a utilização do mesmo. A aplicação também poderá receber novas funcionalidades e melhorias sem muito custo, pois, as tecnologias que foram utilizadas, provem uma manutenção simples e fácil.

Referências

Amazon - AWS. **Modelagem de Dados**. 2024. Disponível em: <https://aws.amazon.com/pt/what-is/data-modeling/>. Citado na página 15.

Angular. **Angular**. 2024. Disponível em: <https://angular.io/guide/what-is-angular>. Citado na página 14.

Bootstrap. **Bootstrap**. [S.l.], 2024. Disponível em: <https://getbootstrap.com/docs/5.3/about/overview/>. Citado na página 14.

CHEN, P. P. The entity-relationship model—toward a unified view of data. **ACM Transactions on Database Systems**, ACM, v. 1, n. 1, p. 9–36, 1976. Citado na página 15.

DECRETO-LEI n. 25. 1937. Brasil. Disponível em: <http://portal.iphan.gov.br/uploads/ckfinder/arquivos/Decreto-Lei%20n%C2%B0%2025%20de%2030%20de%20novembro%20de%201937.pdf>. Citado na página 8.

IPHAN. **IPHAN - Instituto do Patrimônio Histórico e Artístico Nacional**. 2022. Disponível em: <http://portal.iphan.gov.br/pagina/detalhes/126>. Acesso em: 24 abril 2023. Citado 2 vezes nas páginas 8 e 10.

IPHAN, G. F. do B. **Gov - Iphan**. 2023. Disponível em: <https://www.gov.br/iphan/pt-br>. Acesso em: 12 maio 2023. Citado na página 10.

_____. **Rede de arquivos - Iphan**. 2023. Disponível em: <https://www.gov.br/iphan/pt-br/centrais-de-conteudo/publicacoes/rede-de-arquivos-iphan>. Acesso em: 12 maio 2023. Citado na página 10.

LARMAN, C. **Utilizando UML e Padrões**. 3. ed. Porto Alegre: Pearson Education, 2005. Acesso em: 22 maio 2023. Citado na página 15.

Mozilla Developer Network. **JavaScript**. 2024. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>. Citado na página 14.

POSTGRESQL. **PostgreSQL: License**. 2023. Disponível em: <https://www.postgresql.org/about/licence/>. Acesso em: 22 maio 2023. Citado na página 14.

RETOMBA. **Retomba**. 2024. Disponível em: <https://comunica.ufu.br/noticias/2024/01/alunos-e-professores-do-curso-de-arquitetura-da-ufu-criam-aplicativo-de-0>. Acesso em: 13 maio 2024. Citado na página 12.

Spring Boot. **Spring Boot**. 2024. Disponível em: <https://spring.io/why-spring>. Citado na página 13.

Sun Microsystems. **Java Language Specification (JLS)**. 1st. ed. [S.l.]: Addison-Wesley, 1996. Citado na página 13.

UML. **Use Case Diagram**. 2022. Disponível em: <https://www.uml-diagrams.org/use-case-diagrams.html>. Citado na página 15.