CaxiasCultural: Desenvolvimento de um Sistema de Gerenciamento de Bens Materiais Tombados pelo Patrimônio Histórico e Cultural de Caxias do Sul

Pedro Henrique Brogliato[1], Rogério Xavier de Azambuja[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia (IFRS) – Campus Farroupilha
95.174-274 – Farroupilha – RS – Brasil

pedrohbrogliato@gmail.com, rogerio.xavier@farroupilha.ifrs.edu.br

Abstract. *The proposal to develop a management and cataloguing system for the heritage sites of the city of Caxias do Sul aims to address the current difficulty in making information about Historical and Cultural Heritage accessible to citizens in a user-friendly and comprehensive way, as well as seeking better organization of these sites by the public administration. The proposed solution involves creating a Web system that presents heritage sites geographically, alongside images, status data, classifications, and map routes. The backend solution was built using the JavaScript programming language, linked to the Node.js library, the express.js framework, and paired with a MySQL database. For the frontend, the React library and the customizable georeferencing system OpenStreetMap were applied. Digitalization and user experience are concepts that Web systems can naturally contribute to ensuring that the cultural guidelines set out in the 1988 Federal Constitution of Brazil are implemented in daily practice. This initiative will help strengthen the community's cultural identity.*

Keywords**:** *Web development, culture, historical heritage, georeferencing.*

Resumo. *Esta proposta de desenvolvimento de um sistema de gerenciamento para os bens materiais tombados da cidade de Caxias do Sul aborda a atual dificuldade de disponibilização amigável e ampla das informações de Patrimônio Histórico e Cultural aos cidadãos, além de buscar uma melhor organização destes bens por parte da administração pública. Como solução, é realizada a construção de um sistema Web em que os bens tombados são apresentados geograficamente, dispondo de imagens, dados de estado atual, classificações e construção de rotas no mapa. Foi utilizada na construção da aplicação backend a linguagem de programação JavaScript, a biblioteca Node.js e o framework express.js, juntamente a um banco de dados MySQL. Para o frontend, aplicou-se a tecnologia React e um sistema de georreferenciamento personalizável OpenStreetMap. A digitalização e a experiência de usuário são conceitos onde sistemas Web podem contribuir naturalmente, de modo que as diretrizes culturais dispostas na Constituição Federal de 1988 sejam efetivadas no dia a dia e colaborem com o fortalecimento da identidade cultural da cidade.*

Palavras-chave: desenvolvimento *Web,* cultura, patrimônio histórico, georreferenciamento.

1. Introdução

A Constituição Federal de 1988, em seu Art. 216, define como patrimônio cultural brasileiro o conjunto composto por bens de natureza material (tangíveis) e imaterial (intangíveis), que portam referência à identidade, à ação ou à memória dos grupos humanos formadores da sociedade brasileira. Incluem-se nesta classificação, dentre outros, as formas de expressão, as obras, objetos, documentos, edificações e demais espaços destinados às manifestações artístico-culturais, bem como os conjuntos urbanos e sítios de valor histórico, paisagístico, artístico, arqueológico, paleontológico, ecológico e científico (Brasil, 1988).

A Organização das Nações Unidas (ONU), por intermédio de seu programa para o desenvolvimento, redução das desigualdades e promoção do desenvolvimento sustentável (PNUD - Programa das Nações Unidas para o Desenvolvimento), destaca a importância do patrimônio cultural para a memória e à identidade individual e coletiva, de modo a fornecer uma ponte entre o passado, o presente e o futuro. Nesse sentido, o patrimônio cultural amplifica a coesão social, a diversidade, o bem-estar e a qualidade de vida, além de fomentar a atratividade e a criatividade de cidades e garantir o sucesso a longo prazo do turismo (United Nations, 2021).

Em consonância com essa visão, a Constituição Federal de 1988 assegura, em seu Art. 23, a competência da União, dos Estados, do Distrito Federal e dos Municípios quanto à proteção de obras e bens de valor histórico, artístico e cultural, bem como a atuação na oferta de acesso à cultura aos cidadãos (Brasil, 1988).

Caxias do Sul, município no estado do Rio Grande do Sul, possui uma grande quantidade de bens materiais e imateriais tombados, tanto em âmbito municipal quanto em âmbito estadual, um reflexo de sua rica história de 135 anos. A fim de regulamentar e atender à preservação deste patrimônio a nível municipal, especialmente na forma da Lei nº 7.495, de 19 de outubro de 2012 e do decreto nº 16.581, de 24 de julho de 2013, foi criada a Divisão de Proteção ao Patrimônio Histórico e Cultural (DIPPAHC), vinculada à Secretaria Municipal da Cultura (Caxias do Sul, 2024).

Contudo, apesar de amplo arcabouço legal nas esferas federal, estadual e municipal, verifica-se uma deficiência na forma de apresentação das informações de bens tombados aos cidadãos em geral. Atualmente, o único instrumento de consulta fornecido pelo poder municipal caxiense consiste em uma lista estática de bens com seu respectivo endereço e tombamento. Esta modalidade pouco contribui para o envolvimento da comunidade em um tópico tão importante para a formação da identidade coletiva e individual da cidade e de sua comunidade (Caxias do Sul, 2025).

Considerando o cenário retratado, este trabalho tem como objetivo a implementação de um sistema de gerenciamento de bens materiais tombados em Caxias

do Sul, visando fomentar a valorização do Patrimônio Histórico e Cultural, e, por conseguinte, da própria cidade e dos munícipes, de forma interativa e visual.

Nomeado como CaxiasCultural, a fim de enaltecer a conexão entre a cidade e seu Patrimônio Histórico e Cultural, o sistema permite a navegação *Web* entre os bens tombados catalogados a partir da integração a um sistema de mapas, informando os locais, características, materiais relacionados e demais informações relativas ao bem, contribuindo para o acesso democrático e para um gerenciamento mais efetivo por parte da administração municipal.

Este artigo apresenta, em sua Seção 2, o referencial teórico que sustenta a implementação de sistemas similares ao aqui proposto; na Seção 3, constam os procedimentos metodológicos e as tecnologias escolhidas para a efetivação do projeto; na Seção 4, estão contidos aspectos referentes ao desenvolvimento da aplicação *Web* e a descrição das principais funcionalidades do sistema; por fim, as Seções 5 e 6 discutem os resultados obtidos no trabalho realizado, bem como as considerações finais e possíveis trabalhos futuros.

2. Referencial Teórico

Quando tratamos de sistemas informatizados, estamos tratando de fluxos de informação que estão presentes em todas as esferas da vida do século XXI: governo, sociedade, empresas e instituições nacionais e internacionais. O mundo moderno tem fortemente utilizado *softwares*, desde a produção industrial até o sistema financeiro, passando pelo entretenimento e pelos serviços; os sistemas de *software* são abstratos e intangíveis, o que configura um não regimento por leis da física ou processos de produção. Essa natureza dos sistemas torna o seu potencial praticamente ilimitado, mas, ao mesmo tempo, contribui para o surgimento de complexidades lógicas e financeiras aos seus projetos (Sommerville, 2018, p.4).

A Engenharia de Software é a área que atua justamente na organização profissional do desenvolvimento de *softwares*, lançando mão de técnicas de especificação, de projeto e de evolução dos sistemas, enfim, de todo o processo desde a concepção inicial até a operação e manutenção, para mitigar os efeitos dessa complexidade. Contudo, existem diferentes tipos de *softwares*, de forma que não existem métodos universais para a construção e concepção destes sistemas (Sommerville, 2018, p.4-7).

Considerando esta conjuntura, o padrão Modelo-Visão-Controlador (MVC) é um dos mais utilizados para o desenvolvimento de sistemas de *software* (Sommerville, 2018, p. 155). Nele, está disposta a separação entre a apresentação e a interação dos dados, estruturada em três componentes lógicos que interagem entre si. Ainda segundo Sommerville (2018, p.155), o componente ‘Modelo’ é aquele voltado ao gerenciamento dos dados do sistema e das operações associadas a eles, enquanto o componente ‘Visão’

define e gerencia como os dados são apresentados ao usuário; por fim, o componente 'Controlador' diz respeito à interação do usuário (cliques de *mouse*, pressionamento de teclas etc.) e ao transporte dessas interações para os componentes de Visão e de Modelo.

Este padrão MVC é a base do gerenciamento da interação em muitos sistemas *Web*, contando com suporte por parte de diversos *frameworks* atualmente disponíveis para uso. A Figura 1 esquematiza a arquitetura de aplicação *Web* utilizando o padrão MVC.

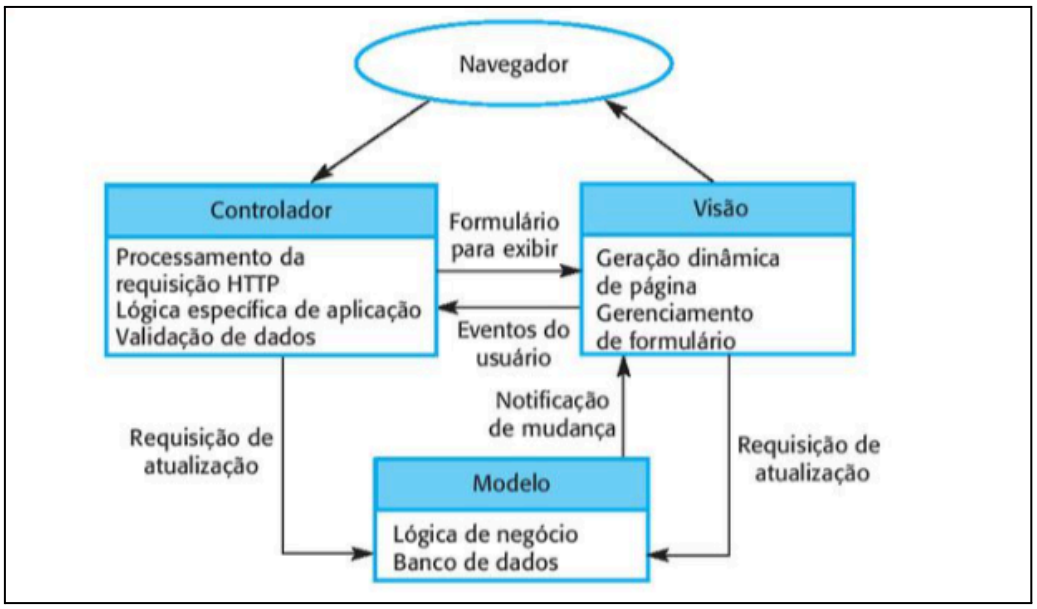

Figura 1. Arquitetura de aplicações *Web* utilizando o padrão MVC.

Fonte: Sommerville (2018, p.156).

Para a implementação de um *software* com o padrão MVC, torna-se necessário que um banco de dados seja incorporado ao projeto. Um banco de dados relacional é voltado ao armazenamento e fornecimento de pontos de dados que possuem uma relação entre si: em vez de se colocar todos os dados em uma única grande entidade, implementa-se uma separação, contribuindo para maior clareza na organização e na performance. O modelo de dados lógico, com objetos como tabelas de dados, *views*, linhas e colunas, oferece um ambiente de programação flexível, a partir de regras que controlam as relações entre diferentes campos de dados, como "um para um", "um para muitos", exclusivo, obrigatório ou opcional. O banco de dados impõe essas regras de integridade referencial para que a aplicação não passe por problemas de dados inconsistentes ou duplicados (Oracle, 2014).

O modelo relacional também implica em uma separação das estruturas de dados lógicas (tabelas de dados, exibições e índices) das estruturas de armazenamento físico. Essa separação determina que os administradores de banco de dados podem gerenciar o armazenamento de dados físicos sem se preocupar com possíveis problemas de acesso a

esses dados como uma estrutura lógica. A Linguagem de Consulta Estruturada *(SQL – Structured Query Language)* é a linguagem mais comumente utilizada para acessar e manipular dados (Oracle, 2023).

Em tempo, um importante recurso para a implementação de softwares no contexto de rápidas e frequentes alterações em escopo, requisitos e conectividade com outros serviços, típicas do desenvolvimento de *software* no século XXI, é representado pelo advento do controle de versão. Trata-se de um sistema que registra as modificações realizadas em arquivos ao longo do tempo e que oferece a possibilidade de reverter os arquivos para uma versão prévia em específico, comparar as mudanças entre versões e trazer informações de quem e quando foram performadas ações, além de prevenir erros de compatibilidade de componentes desenvolvidos por pessoas diferentes (Sommerville, 2018, p.694).

Apesar de sua origem datar da década de 1980, foi com o surgimento do programa Git (Git, 2025a), em 2005, que o controle de versão se tornou popular. Ele é um sistema distribuído de controle de versão, o que significa que seus clientes (usuários) duplicam localmente o repositório de trabalho completo. Logo, se qualquer servidor passar por uma queda, qualquer repositório de cliente pode ser copiado de volta para o servidor, a fim de restaurá-lo. Além disso, trata-se de um sistema que apresenta baixa curva de aprendizagem, ponto que favorece seu uso em diversos tipos de projeto. A Figura 2 ilustra como os clientes mantêm cópias inteiras do repositório que está no servidor, permitindo a criação de novas implementações a partir da cópia original.

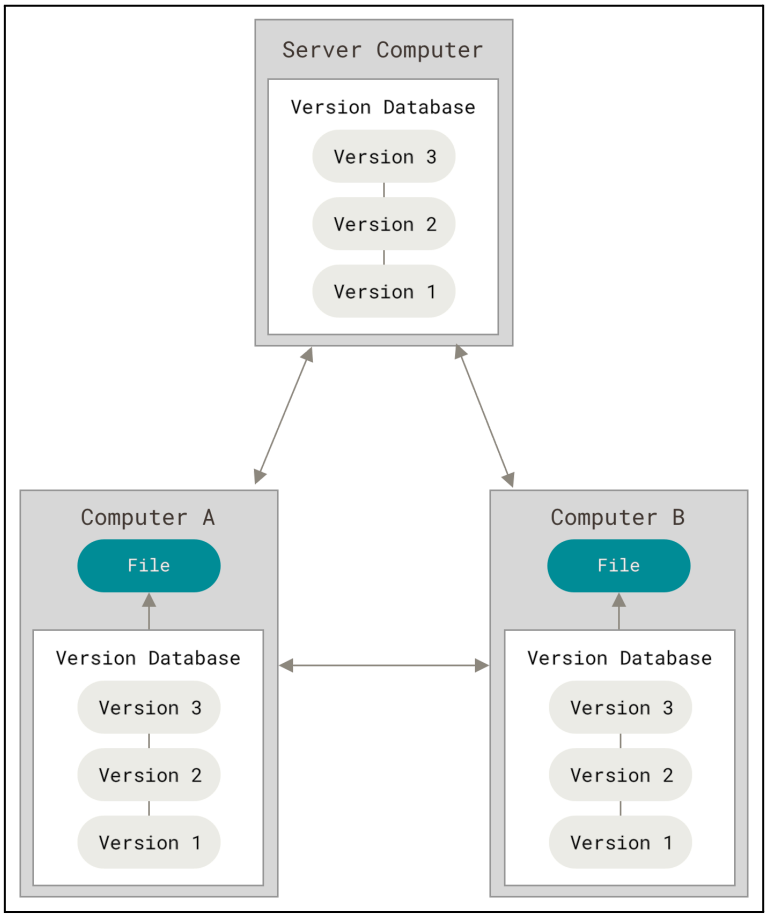

Figura 2. Esquematização de um sistema distribuído de controle de versão.

Fonte: Git, 2025b.

A abordagem referida de controle apresentada pelos sistemas distribuídos é diferente daquela observada em sistemas prévios: os sistemas locais de controle de versão possuem como intuito a cópia de um diretório para outro, que receberá as modificações. Contudo, essa abordagem tem alta propensão a erros, já que pode-se sobrescrever ou copiar arquivos indesejados. Por sua vez, os sistemas centralizados de controle de versão apresentam um repositório compartilhado com todos os desenvolvedores, que realizam modificações e as submetem ao repositório. Isso facilita a visualização por parte da equipe de quais pessoas estão atuando em determinadas seções de código; porém, sua maior desvantagem é oferecer um ponto de falha único, suscetível a problemas que paralisam a produção de código. (Git, 2025b).

3. Procedimentos Metodológicos

Para a construção da solução de gerenciamento de bens materiais tombados de Caxias do Sul, realizou-se os seguintes procedimentos, em ordem cronológica: definição de

escopo, modelagem da proposta, definição de tecnologias a serem empregadas e desenvolvimento da aplicação.

3.1. Tecnologias

O desenvolvimento do sistema foi conduzido a partir da linguagem de programação *JavaScript*, uma linguagem leve, interpretada e multi-paradigma, que suporta desenvolvimentos com orientação a objetos, paradigma imperativo e paradigma declarativo (funcional). A linguagem tem como fundamentação os objetos com função de primeira classe, ou seja, o tratamento de uma função como qualquer outra variável, permitindo a passagem dela como um argumento para outras funções (Mozilla Foundation, 2024a; Mozilla Foundation, 2024b).

JavaScript é amplamente reconhecida como uma linguagem de *script* para páginas *Web*, apesar de sua aplicação ser possível também em diversos ambientes fora de um navegador *Web*, tais como *Node.js* e *Apache CouchDB* (Mozilla Foundation, 2024b). No *JavaScript*, a construção de objetos é baseada em protótipos, logo, os objetos são usados como um modelo do qual se obtém as propriedades iniciais para um novo objeto, o que favorece o reuso de código. Além disso, sua sintaxe foi pensada para ser aplicada de forma similar à de outras linguagens, como *Java* e *C++*, a fim de propiciar uma curva de aprendizado suave (Mozilla Foundation, 2023).

React é uma biblioteca *JavaScript* voltada a criação de interfaces de usuário com o uso de componentes, onde todos os seus componentes são funções *JavaScript*. Foi criada pelo *Facebook Inc.*, atual *Meta Platforms Inc.*, em 2013. O *React* também é uma arquitetura em que os *frameworks* que o implementam permitem buscar dados em componentes assíncronos que rodam no servidor ou até mesmo durante o *build,* que é o processo de transformar código-fonte em artefatos de *software* executáveis (IBM, 2025). Assim, pode-se ler os dados de um arquivo ou banco de dados e os passar para seus componentes interativos (React, 2024).

O React conta ainda com o tipo de arquivo JSX (*JavaScript* XML – *eXtensible Markup Language*), que usa sintaxe especial para escrever em Linguagem de Marcação de Hipertexto (HTML – *HyperText Markup Language*) dentro do *JavaScript*, permitindo uma maior intuitividade e declarabilidade na criação de interfaces. Além disso, é possível destacar sua oferta de *Hooks* (ganchos), que habilitam o uso de estado e ciclo de vida em componentes funcionais, dando maior modularidade ao código-fonte. Os principais são *useState* (permite que valores mutáveis sejam mantidos por componentes funcionais), *useEffect* (conecta um componente a um sistema externo) e *useContext* (importação de contextos em outros arquivos para o atual) (React, 2025).

Já o *Node.js* é um ambiente de execução de *JavaScript* gratuito e de código aberto, voltado à criação de servidores, aplicações *Web*, ferramentas de linha de comando e a automação de tarefas. Ele roda a *engine* V8 do *JavaScript*, que é o coração

do navegador *Google Chrome*. Um aplicativo *Node.js* é executado em um único processo, o que evita a criação de novas *threads*, ou seja, caminhos de execução de uma aplicação, para cada solicitação a ser processada. Assim, as operações de entrada e saída do *Node.js*, como acesso a banco de dados e leitura de rede, são retomadas apenas quando a resposta for retornada, sem desperdício de ciclos da Unidade de Processamento Central (CPU – *Central Processing Unit*) em espera e bloqueios de *thread*. Além disso, sua grande vantagem é justamente o uso de uma única linguagem para desenvolvimento *frontend* e *backend*, eliminando a necessidade de aprendizado de outra linguagem por parte dos desenvolvedores (Node.js, 2024).

Em consonância, o *Express.js* é um *framework Node.js* de aplicativos para a *Web* flexível e minimalista, que fornece uma camada fina de recursos sem obscurecer o poder do já estabelecido *Node.js*. Sua filosofia inclui a provisão de pequenas e robustas ferramentas para servidores do tipo Protocolo de Transferência de Hipertexto (HTTP – *Hypertext Transfer Protocol*), com atributos como roteamento robusto, ajudantes de HTTP e foco em alta performance (Express.js, 2024).

Com relação à persistência de dados dos bens tombados, o *MySQL* foi a ferramenta escolhida, em acordo com a ideia de uso de *software* de código aberto para a implementação do projeto. Algumas das principais características do *MySQL* para o gerenciamento de banco de dados relacionais são sua confiabilidade, escalabilidade e facilidade de uso. Ele foi originalmente desenvolvido para processar grandes volumes de dados rapidamente e é usado em ambientes de produção altamente exigentes (Oracle, 2023).

Para o controle de versão do CaxiasCultural, foi empregada a ferramenta Git, vinculada ao uso de um perfil no GitHub. Com isso, foi possível armazenar o código em um repositório, local que reúne também arquivos e seus históricos de revisão, facilitando o gerenciamento de trabalho (GitHub, 2025). O Git foi utilizado por ser um sistema distribuído de controle de versão, gratuito e *open-source*, desenhado para atender projetos de qualquer escala, mantendo velocidade e eficiência como guias (Git, 2025a).

A ferramenta escolhida para a implementação do mapa de navegação dos patrimônios foi o *OpenStreetMap*, devido a sua natureza *open-source* e gratuita, sem limitações de chamadas via Interface de Programação de Aplicações (API – *Application Programming Interface*), com uma boa integração para o propósito da ferramenta CaxiasCultural (OpenStreetMap, 2024). A Figura 3 exemplifica o uso do *OpenStreetMap* num contexto de mapeamento de escolas públicas no estado de Queensland, Austrália.

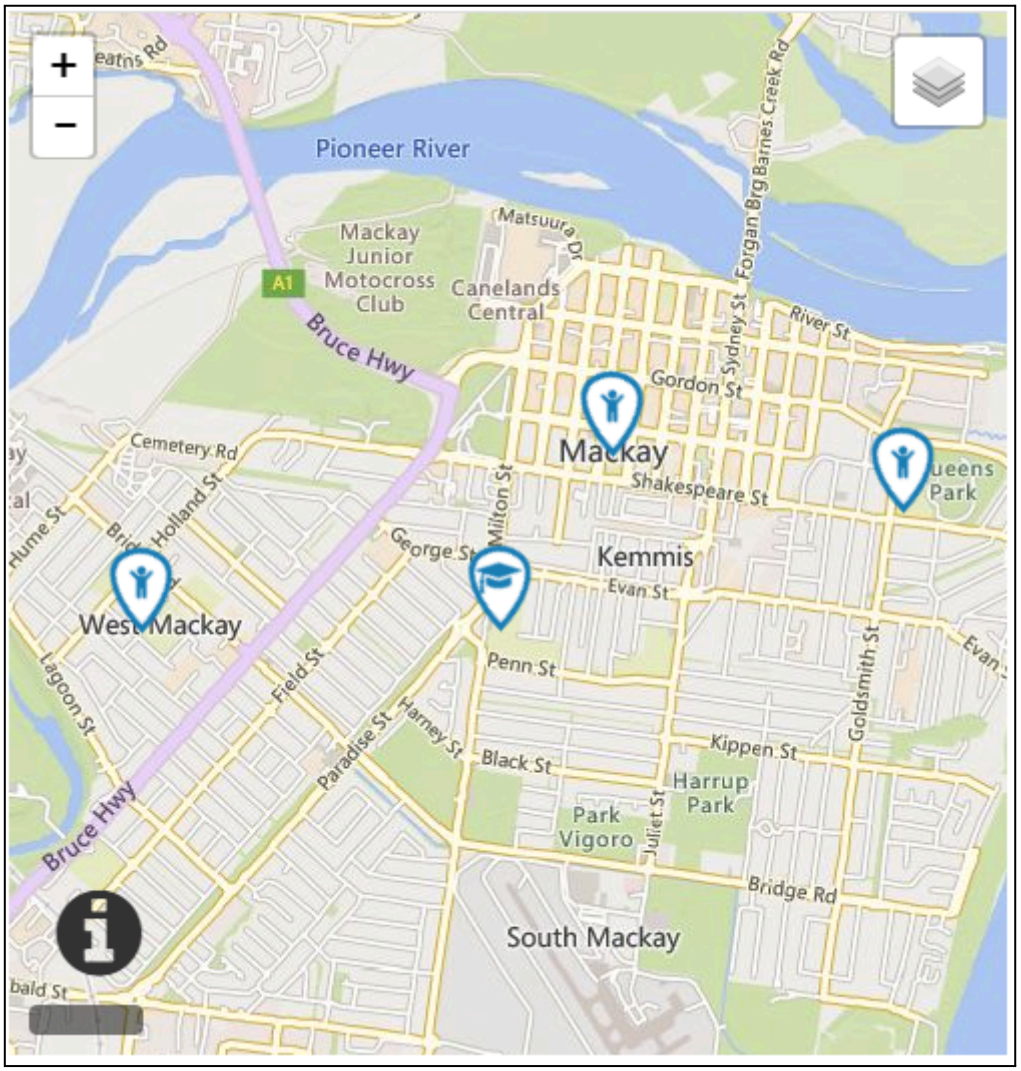

Figura 3. Exemplo de mapa criado via *OpenStreetMap*.

Fonte: Queensland, 2025.

Em tempo, o uso do *OpenStreetMap* foi facilitado pela biblioteca *JavaScript Leaflet*, para mapas interativos. Também *open-source*, o *Leaflet* conta com diversos recursos de mapas, como zoom, navegação por teclado, comutador de camadas, entre outras; além disso, é extremamente leve: é constituído por apenas 431 KB na sua versão-fonte (Leaflet, 2024).

4. Desenvolvimento

Considerando as tecnologias apresentadas na seção anterior, o desenvolvimento da aplicação foi estruturado para contemplar um sistema *Web* completo. Esta escolha se deu pelo amplo alcance e compatibilidade de dispositivos a essa plataforma, de modo

que ambos os públicos-alvo da ferramenta (administração municipal e comunidade externa) possam usufruir em plenitude as possibilidades da ferramenta construída.

O projeto tem como paradigma de programação predominante a programação funcional, aliado a uma arquitetura baseada em componentes no *frontend* devido ao alinhamento da linguagem de programação escolhida para o projeto (*JavaScript*) e da biblioteca *React* com esse modo de operação.

4.1. Modelagem

Para o melhor entendimento das funcionalidades do projeto, foram confeccionados quatro artefatos de modelagem de dados. São eles: análise e classificação de requisitos, diagrama de casos de uso, diagrama de entidade e relacionamento (ER) e interfaces de protótipo.

4.1.1 Análise e Classificação de Requisitos

Os requisitos obtidos para a aplicação foram analisados considerando os dois diferentes papéis para os quais o sistema foi modelado: gerenciador de bens (administrador) e comunidade externa (usuário final). A Tabela 1 esquematiza os requisitos confeccionados para compor o CaxiasCultural, incluindo requisitos funcionais (RF), não-funcionais (RNF) e requisitos de sistema (RS).

Tabela 1. Lista de requisitos do sistema

Requisito	Descrição	Classificação
RF1	Gerenciador cadastra bem tombado	Funcional
RF2	Gerenciador atualiza dados do bem tombado	Funcional
RF3	Gerenciador arquiva um bem tombado	Funcional
RF4	Usuário final busca bem tombado	Funcional
RF5	Usuário final traça rotas até bem tombado	Funcional
RF6	Usuário final consulta informações do bem tombado	Funcional
RF7	Gerenciador faz cadastramento de imagens	Funcional
RF8	Gerenciador faz cadastramento de tags	Funcional
RNF1	O sistema será desenvolvido para o ambiente *web*	Não-Funcional
RNF2	O cadastro de bens tombados será via inserção das coordenadas geográficas ou apontando o local no mapa	Não-Funcional
RS1	O sistema utilizará software livre para o banco de dados	Sistema
RS2	O sistema criará uma identificação única para cada bem tombado por um ID autogerado	Sistema

Fonte: Autoria Própria, 2024.

Adicionalmente, foram definidas duas regras de negócio (RN) pertinentes a este *software*: A RN1 define que, para arquivar (destombar) um bem, a legislação precisa ser alterada indicando que aquele não é mais um bem tombado. Ainda, no destombamento do bem neste sistema, é necessário inserir a legislação que alterou o entendimento. Já a RN2 dispõe que o cadastro de bens também necessita de modificação na legislação municipal. O administrador do sistema precisará indicar qual o número do processo administrativo e as folhas do livro do Tombo correspondentes ao tombamento do bem para que a inserção no CaxiasCultural ocorra.

4.1.2 Diagrama de Casos de Uso

A fim de proporcionar uma melhor visualização das funcionalidades básicas identificadas para o CaxiasCultural, elaborou-se um Diagrama de Casos de Uso, conforme descrito em Sommerville (2018, p. 109) e apresentado na Figura 4.

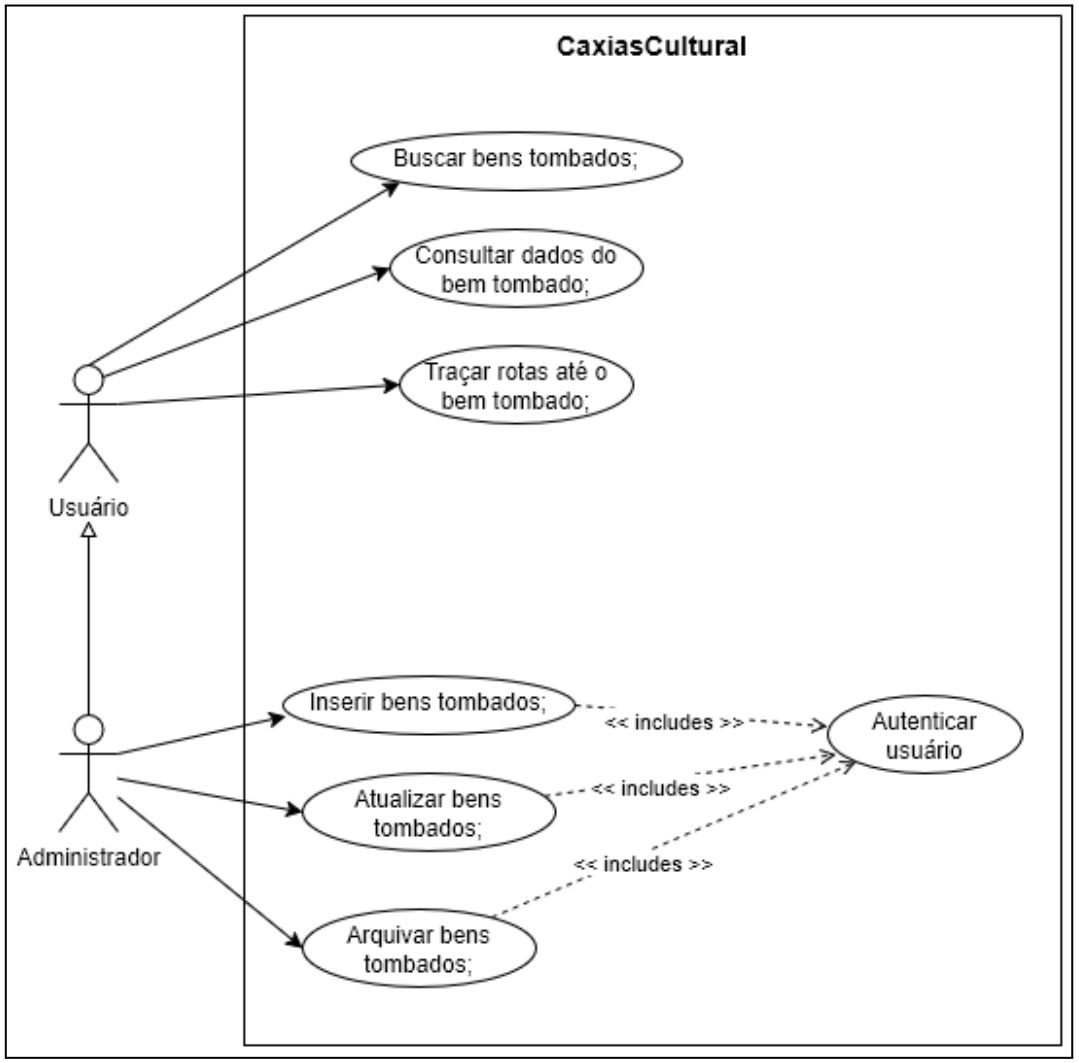

Figura 4. Diagrama de Casos de Uso do sistema de gerenciamento de bens tombados

Fonte: Autoria Própria, 2024.

O diagrama retrata os cenários de funcionalidades dos dois perfis de usuário previstos: final e de administrador para o sistema de gerenciamento de bens materiais tombados pelo Patrimônio Histórico e Cultural de Caxias do Sul. É possível notar que apenas as operações de administrador incluem uma autenticação, e também que o papel de 'Administrador' herda as operações do papel 'Usuário', visto que elas não envolvem mudanças no banco de dados, em contraste com as operações exclusivas de administrador.

4.1.3 Modelo Entidade e Relacionamento

O diagrama de entidade e relacionamento busca explicitar a modelagem do banco de dados relacional de forma clara e concisa. A Figura 5 demonstra o esquema para o sistema de gerenciamento de bens materiais tombados, apresentando as tabelas de patrimônio, usuário, *tags* (que servirão para classificação dos bens) e imagens.

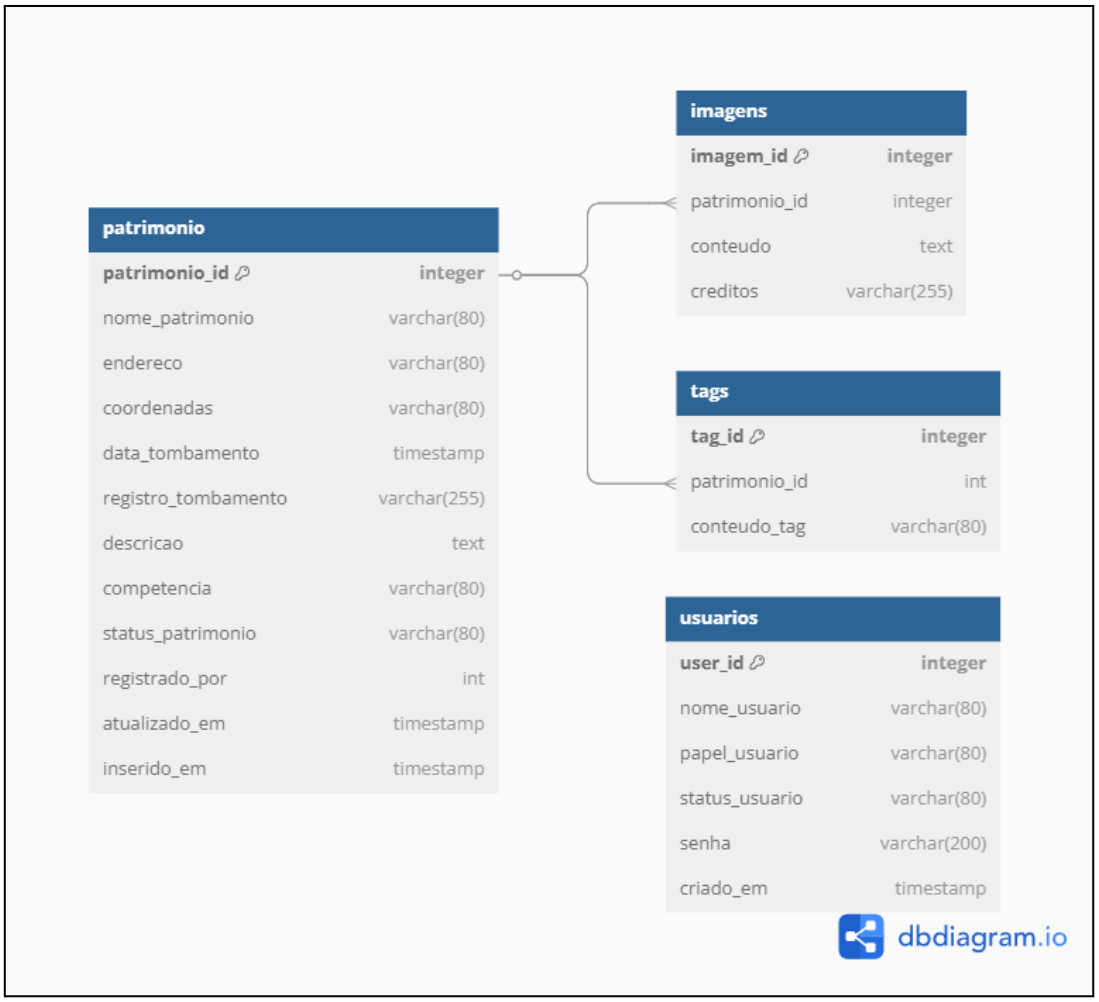

Figura 5. Diagrama de Entidade e Relacionamento do sistema de gerenciamento de bens tombados.

Fonte: Autoria Própria, 2025.

A tabela de patrimônio contém as informações que caracterizam o bem material tombado, enquanto a tabela de '*tags*' busca classificar o bem em diversos possíveis estados, como 'fechado', 'em reforma', 'restaurante', entre outros. A tabela de imagens atrela os conteúdos de mídia ao patrimônio, e a tabela de usuários propõe o gerenciamento dos usuários administradores. A cardinalidade dos relacionamentos da tabela de patrimônio com as tabelas de imagens e de *tags* é de um para muitos (1:N). Ou seja, um patrimônio pode ter muitas (zero ou mais) *tags*/imagens, mas cada *tag*/imagem

corresponde a exatamente um patrimônio. Ainda, pode se destacar que o atributo ‘registrado_por’ na tabela de patrimônio serve como uma referência do tipo *‘soft’* ao atributo de ‘user_id’ da tabela de usuários, a fim de possibilitar a identificação do autor da última modificação sem engessar as alterações na tabela principal.

4.1.4 Interfaces de Protótipo

Nesta subseção, constam interfaces de protótipo classificadas como diretrizes de implementação de interfaces no sistema de gerenciamento de bens tombados. As Figuras 6 e 7 trazem o conceito de visualização em mapa proposto, onde estão localizados os patrimônios, em uma disposição que permite a navegação entre pontos (*pins*) no mapa.

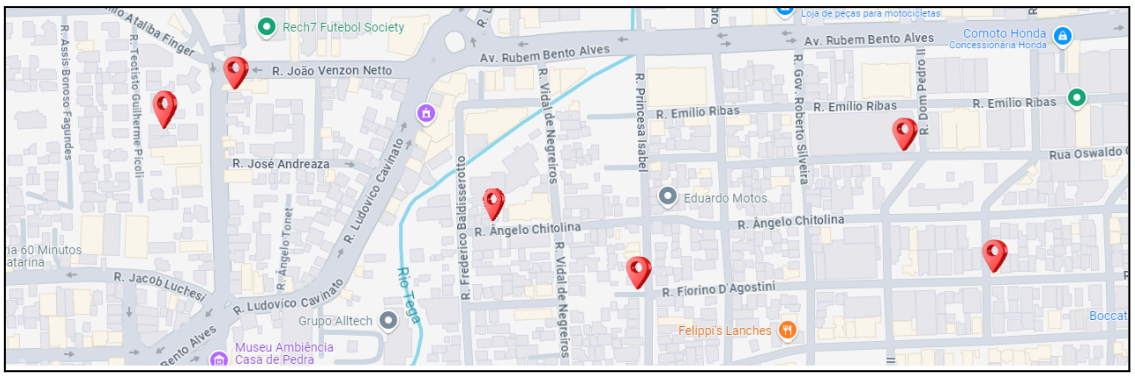

Figura 6. Conceito de visualização em mapa do sistema de gerenciamento de bens tombados.

Fonte: Autoria Própria, 2024.

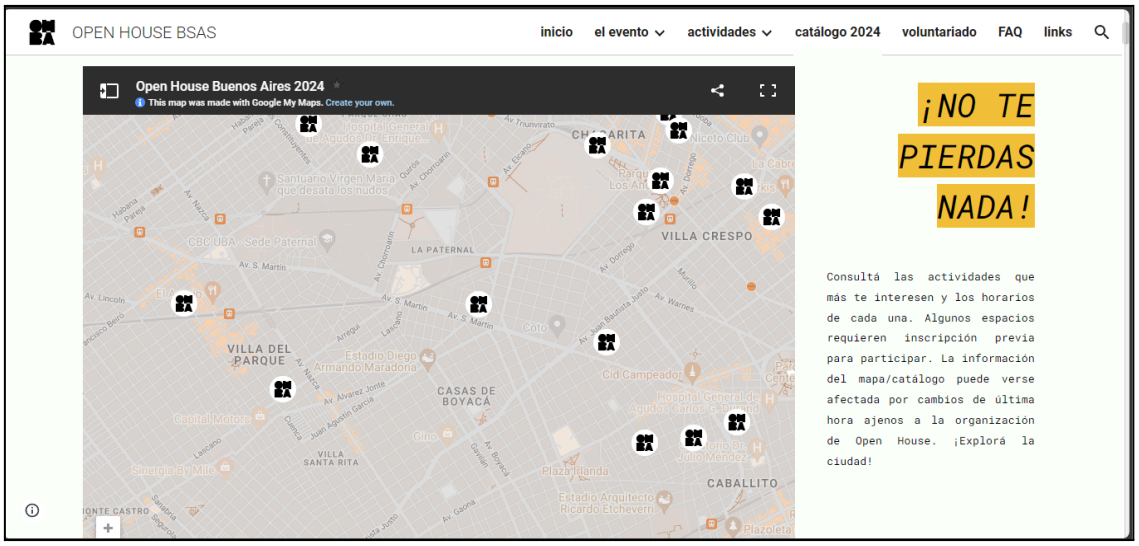

Figura 7. Conceito de personalização dos ícones no mapa do sistema de gerenciamento de bens tombados.

Fonte: Open House BsAs, 2024.

A Figura 8 traz o conceito de catálogo de tombamentos, que é complementar ao conceito de visualização em mapa das Figuras 6 e 7 no sistema de gerenciamento de bens tombados. Assim, os bens ficam dispostos de forma clara ao usuário do sistema, além de estarem geograficamente localizados.

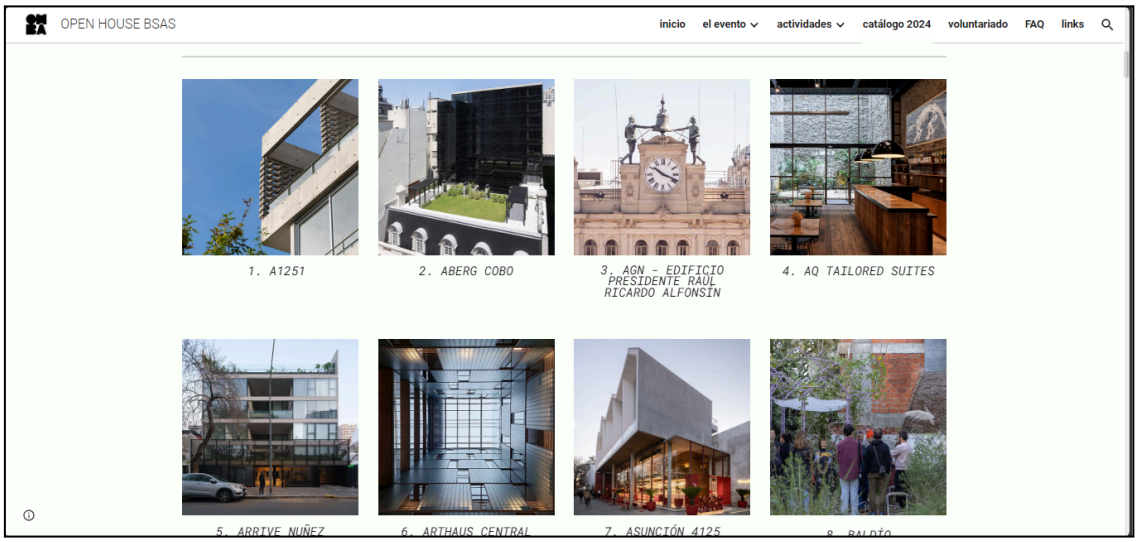

Figura 8. Conceito de catalogação dos bens do sistema de gerenciamento de bens tombados.

Fonte: Open House BsAs, 2024.

Cada um dos itens, aqui representados por quadrados com a imagem de cada local, foi pensado para guardar as informações específicas daquele patrimônio, que será expandido em outra tela ao ser acessado.

O sistema conta ainda com as telas de *login* e de informações para usuários gerenciadores, além de opções visuais de alteração e arquivamento de bens.

4.2. Implementação

A partir da conceitualização desenvolvida por meio dos artefatos de modelagem de dados e de sistema, deu-se início à implementação do projeto em março de 2025, com o término se dando no mês de maio de 2025. A codificação do CaxiasCultural foi realizada por meio do *Visual Studio Code*, ferramenta da *Microsoft* (Visual Studio Code, 2025). A Figura 9 ilustra a estrutura de diretórios e arquivos resultante do processo.

Figura 9. Estrutura do projeto CaxiasCultural.

Fonte: Autoria Própria, 2025.

A subdivisão básica do projeto é representada pelo diretório de *frontend*, que reúne os elementos gráficos e de interação com o usuário, e pelo diretório de *backend*, que compõe as implementações referentes à conexão com o servidor e as lógicas de banco de dados e APIs.

Dentro do diretório de *backend*, os arquivos '*connection.js*' e '*server.js*' gerenciam a conexão com o banco de dados MySQL e configuram e iniciam o servidor Express, além de trazer as definições de Compartilhamento de Recursos entre Origens (CORS – *Cross-Origin Resource Sharing*) do projeto CaxiasCultural, que são responsáveis pela integração entre o domínio de cliente e outro domínio que contém maiores recursos (no caso do projeto, duas portas diferentes do *localhost*). A pasta de '*middleware*' traz as definições de autenticação e verificação de *Tokens Web JSON* (JWT – *JSON Web Tokens*) para a transmissão de informações de forma segura e compacta, relacionadas ao processo de *login* e de autorização das operações de edição e inserção no sistema (JWT.IO, 2025).

O diretório '*routes*' dispõe das rotas relacionadas a patrimônios e usuários, com as operações de criação, leitura e atualização, *upload* de imagens, gerenciamento de *tags* e administração dos usuários. A pasta '*public/uploads*' reúne os arquivos de imagem que são submetidos à plataforma, viabilizada por meio do Multer, um *middleware Node.js* responsável por definir o local de armazenamento e a geração de um nome único usando *timestamp* (Multer, 2025). Com isso, os arquivos se tornam acessíveis porque o servidor está configurado para servir arquivos estáticos no arquivo '*server.js*'.

Ainda, o diretório de '*services*' contém a lógica de negócio relacionada aos usuários, incluindo a criptografia de senhas por meio do '*bcrypt*', ferramenta que utiliza *hash*, definindo sua complexidade e salvando a senha hasheada no banco de dados a partir do texto original (bcrypt, 2025). A Figura 10 contém o trecho do código-fonte construído que implementa a criação de novos usuários administradores.

```
async createUser(con, userData, adminRole) {
    try {
      // Verifica se o administrador tem permissão
      if (adminRole !== "admin") {
        throw new Error("Apenas administradores podem criar usuários");
      }

      // Verifica dados obrigatórios
      if (
        !userData.nome_usuario ||
        !userData.senha ||
        !userData.papel_usuario
      ) {
        throw new Error("Nome de usuário, senha e papel são obrigatórios");
      }

      // Verifica se usuário já existe
      const [existingUser] = await con.query(
        "SELECT user_id FROM usuarios WHERE nome_usuario = ?",
        [userData.nome_usuario]
      );

      if (existingUser.length > 0) {
        throw new Error("Nome de usuário já existe");
      }

      // Hash da senha
      const hashedPassword = await bcrypt.hash(userData.senha, SALT_ROUNDS);

      // Insere novo usuário
      const [result] = await con.query(
        `INSERT INTO usuarios (
          nome_usuario,
          senha,
          papel_usuario,
          status_usuario,
          criado_em
        ) VALUES (?, ?, ?, 'pre', NOW())`,
        [userData.nome_usuario, hashedPassword, userData.papel_usuario]
      );

      return result.insertId;
    } catch (error) {
      throw new Error(error.message);
    }
  },
```

Figura 10. Trecho de código-fonte do arquivo '*userService*.js' responsável pela criação de novos usuários.

Fonte: Autoria Própria, 2025.

Por fim, o arquivo '.env' traz os dados de conexão com o banco de dados e o arquivo '*package.json*' apresenta as dependências utilizadas no *backend*.

Com relação à estrutura contida no diretório de '*frontend*', a pasta '*src/assets*' comporta as imagens que compõem a identidade visual do CaxiasCultural, como a logo, a *favicon* e outras. A pasta de '*src/components*' oferece os arquivos de cabeçalho, rodapé, *cards* dos patrimônios tombados, o mapa interativo de catalogação dos patrimônios, bem como os modais de criação e alteração de patrimônios e de alteração

de senha de usuário administrador. Em '*src/contexts*' estão o gerenciamento de estado de autenticação e o gerenciamento de estado de busca de patrimônios. Já na subpasta '*pages*' de '*src*', estão as páginas de administração de usuários, a *homepage* com a lista de patrimônios, a página de criação de usuários, a página de *login* e a página de rota não encontrada no servidor (padrão para erros do tipo 404). A Figura 11 contém trecho de código-fonte da implementação da parte visual do mapa na homepage.

```
{/* Mapa dos Patrimônios */}
<div className="py-8 bg-gray-50">
  <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
    <h2 className="text-2xl font-medium tracking-tight text-gray-900 mb-8">
      Patrimônios Históricos e Culturais de Caxias do Sul
    </h2>
    <div className="bg-white p-4 rounded-lg shadow-lg">
      {filteredPatrimonios.length > 0 ? (
        <PatrimoniosMap patrimonios={filteredPatrimonios} />
      ) : (
        <div className="h-[500px] flex items-center justify-center">
          <p className="text-gray-500">
            {loading
              ? "Carregando mapa..."
              : "Nenhum patrimônio encontrado"}
          </p>
        </div>
      )}
    </div>
  </div>
</div>
```

Figura 11. Trecho de código do arquivo '*HomePage.js*' responsável pela disponibilização do mapa de patrimônios na *homepage* do CaxiasCultural.

Fonte: Autoria Própria, 2025.

Já os arquivos '*index.html*', '*App.jsx*' e '*main.jsx*' servem como ponto de entrada HTML da aplicação CaxiasCultural, como componente raiz do *React* e como inicialização da aplicação *React*, respectivamente. Por fim, alguns arquivos diversos tratam das configurações das ferramentas *Vite*, *Tailwind CSS* e *PostCSS*, que apoiaram a configuração do sistema e estão intimamente ligadas às Folhas de Estilo em Cascata (CSS – *Cascading Style Sheets*).

O *Vite* é um recurso voltado ao *build* e ao desenvolvimento *frontend* que tem como objetivo fornecer um ambiente de desenvolvimento extremamente rápido e eficiente, a partir da utilização de *ES Modules* nativos (ESM) para o carregamento instantâneo de arquivos, atualização imediata de módulos e amplo suporte para linguagens de programação como *JavaScript* e *TypeScript* (Vite, 2025). A Figura 12

mostra a tela de instalação do *Vite* e as configurações de *framework* (*React*) e linguagem de programação (*JavaScript*) adotadas no projeto CaxiasCultural:

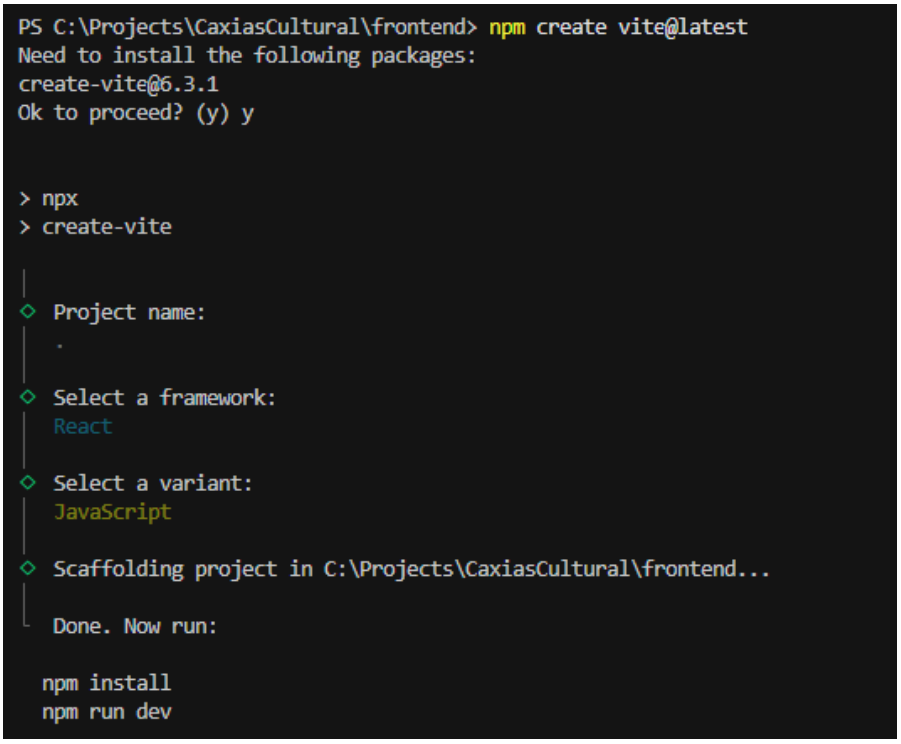

Figura 12. Instalação do *Vite* via npm (terminal).

Fonte: Autoria Própria, 2025.

O *Tailwind CSS*, por sua vez, é um *framework* utilitarista que dispõe de classes que podem ser aplicadas diretamente no código HTML, ao invés de alocar em arquivos CSS separados, facilitando assim a execução do projeto (TailwindCSS, 2025).

5. Resultados

O software CaxiasCultural, resultado do desenvolvimento desta pesquisa, pode ser apresentado sob dois pontos de vista principais, que espelham seus atores de casos de uso: o de usuário final e o de administrador do sistema.

5.1 Usuário Final

Para o usuário final, ao acessar a *homepage* do CaxiasCultural, estarão dispostas de pronto a contextualização do projeto, o mapa interativo de Patrimônios Históricos e a catalogação dos Patrimônios inseridos no portal, sem necessidade de *login*. A Figura 13 mostra o mapa interativo com um dos patrimônios em destaque, enquanto a Figura 14 ilustra a catalogação de quatro bens tombados pelo Município de Caxias do Sul.

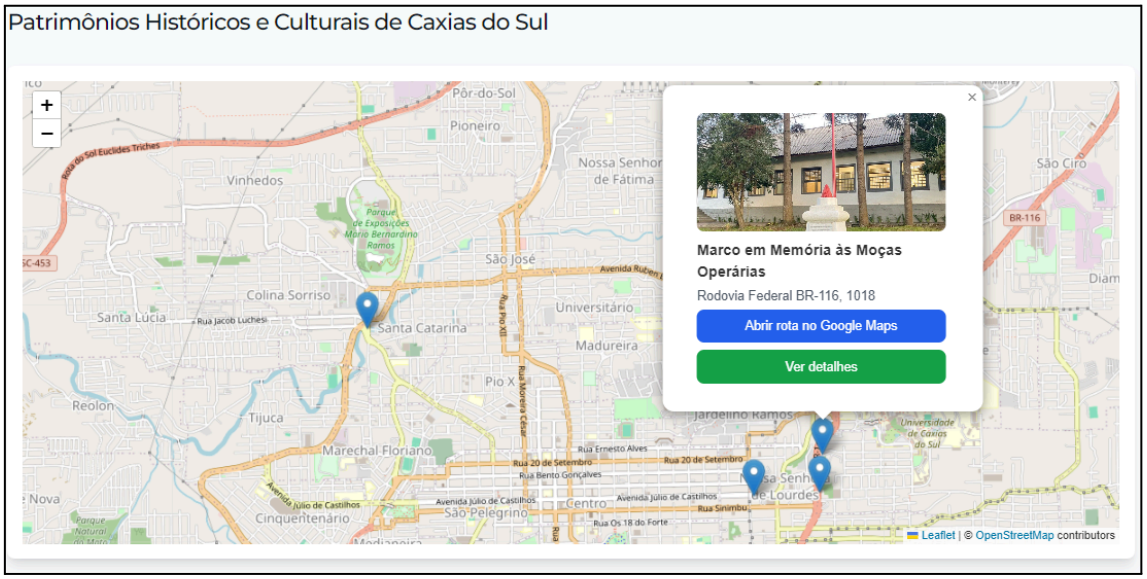

Figura 13. Mapa interativo do CaxiasCultural.

Fonte: Autoria Própria, 2025.

Figura 14. Catálogo de bens tombados do CaxiasCultural.

Fonte: Autoria Própria, 2025.

É importante ressaltar que Caxias do Sul possui atualmente 56 patrimônios históricos sob competência municipal, entre tombados, destombados e em processo de tombamento. Para o projeto CaxiasCultural, quatro deles foram selecionados para compor o catálogo e revelar o potencial da ferramenta, bem como a identificação de novas funcionalidades a serem implementadas futuramente, numa próxima versão.

O botão 'Abrir rota no *Google Maps*', presente na Figura 13, foi desenvolvido com o propósito de facilitar a busca por direções referentes ao patrimônio histórico selecionado pelo usuário. A grande expertise da *Google* em relação aos seus mapas, bem como seu amplo uso dentro da comunidade digital, foram pontos cruciais para essa escolha. Além disso, todo o gerenciamento de permissão de coleta de dados de localização fica a cargo do usuário final e da *Google*, pois o CaxiasCultural não coleta

nem armazena esses dados, mas sim direciona, conforme coordenadas geográficas do bem tombado e o parâmetro '*current-location*' (localização atual) já existente no *Google Maps*, a um *hyperlink* válido; caso o usuário não tenha consentido em fornecer sua localização, ele ainda terá a opção de escolher livremente outro ponto de saída na ferramenta da *Google* a partir da nova aba aberta no navegador.

Tanto os botões de 'Saiba Mais' (Figura 14) e 'Ver Detalhes' (Figura 13), levam para a tela de informações específicas de um patrimônio. Ainda, a página inicial conta com uma ferramenta de busca implementada no cabeçalho, que realiza cruzamentos com o nome, endereço, *tags* e status do patrimônio, bem como a identificação visual da plataforma e o botão de acesso administrativo do CaxiasCultural, conforme Figura 15.

Figura 15. Detalhes do cabeçalho e do *banner* do CaxiasCultural.

Fonte: Autoria Própria, 2025.

Ao acessar a tela de detalhes de um patrimônio, o usuário se depara com um *banner* e informações como endereço do bem, status do patrimônio (ativo, em restauração, destombado, inativo), data de tombamento, registro de tombamento, competência do tombamento, *tags* vinculadas ao patrimônio, seção de descrição textual, carrossel de fotos do patrimônio, e botão de retorno à página inicial. As Figuras 16 e 17 ilustram essa disposição das informações.

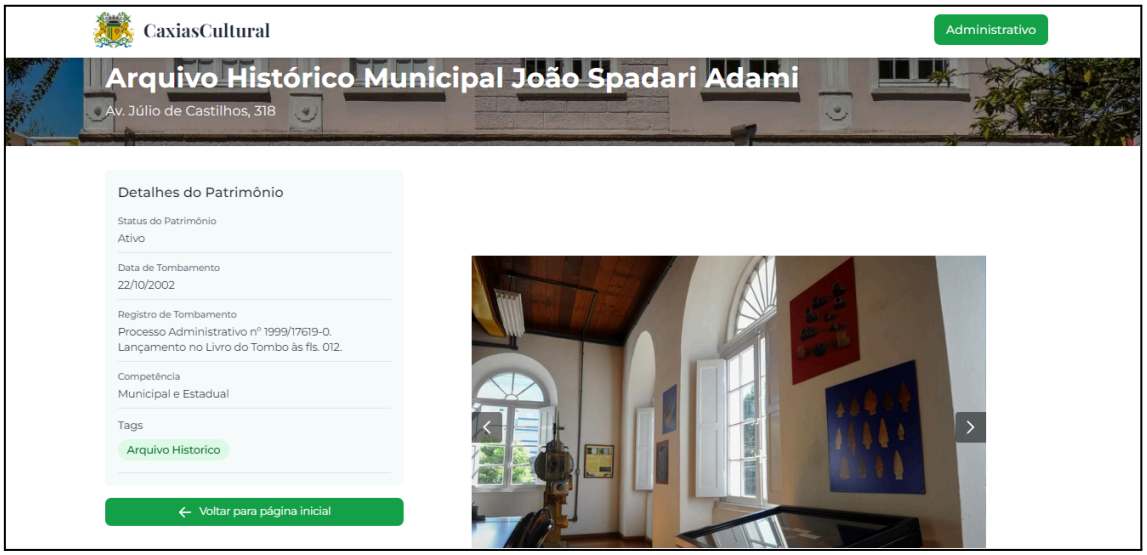

Figura 16. Detalhes do patrimônio e do carrossel de fotos de um bem cadastrado no CaxiasCultural.

Fonte: Autoria Própria, 2025.

Figura 17. Detalhes da seção de descrição de um bem tombado cadastrado no CaxiasCultural.

Fonte: Autoria Própria, 2025.

Ainda, o CaxiasCultural conta com um rodapé com função que permite o retorno do usuário para a página inicial.

5.1 Administrador do sistema

Com relação à experiência de uso do administrador do sistema, existe um grande reúso de informações já aplicadas para o usuário final. Após a realização do *login* (via inserção de usuário e senha), por meio do botão 'Administrativo' presente no cabeçalho, ocorre o redirecionamento para a página inicial; as únicas modificações visuais são o surgimento de um botão para cadastro de novos patrimônios, que faz abrir um formulário de cadastro, onde todas as informações (menos as *tags*) são obrigatórias, e uma mensagem de boas-vindas no cabeçalho, que considera o usuário logado. A Figura 18 esquematiza o formulário de cadastro de novos bens.

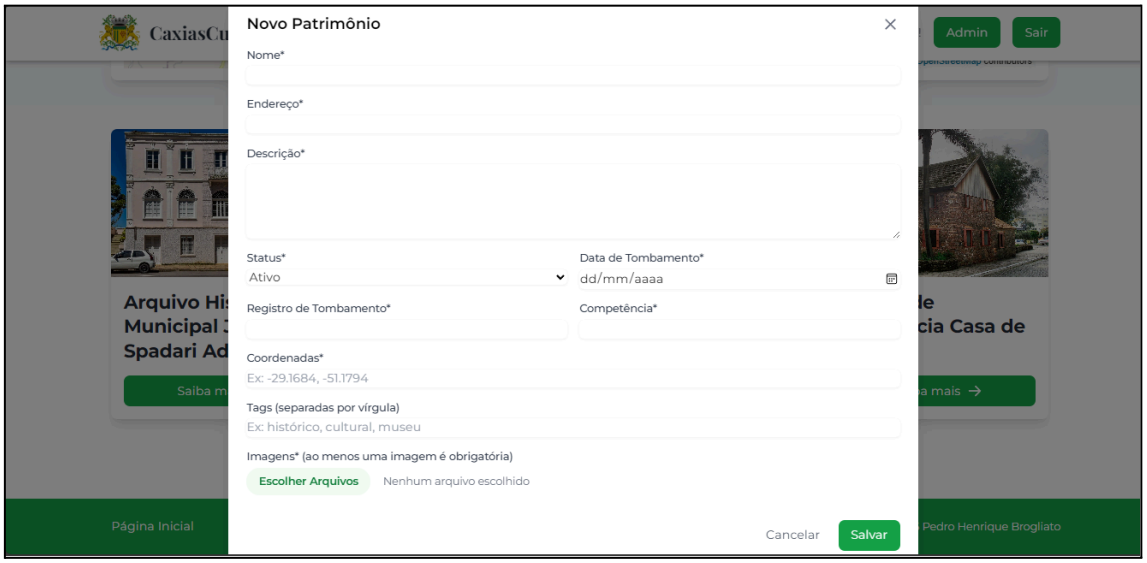

Figura 18. Formulário de cadastro de um novo bem tombado.

Fonte: Autoria Própria, 2025.

Já na seção de um bem tombado em específico, a mudança existente para usuários administradores é a capacidade de editar as informações já estabelecidas, por meio de um formulário idêntico ao utilizado para cadastramento de bens, bem como as fotos que compõem a apresentação do patrimônio em questão. A data de inserção/modificação é guardada no banco de dados, bem como o identificador (ID) do usuário que performou a ação. No CaxiasCultural, não está prevista a deleção de um bem da base de dados, apenas a troca de estado dele para 'destombado', em uma configuração que substitui o arquivamento previsto no escopo inicial; isso porque um bem destombado pode ainda possuir um valor histórico, tal qual seu processo de destombamento.

Por fim, através do botão 'admin' que surge no cabeçalho ao logar-se, o administrador consegue alterar sua senha, bem como cadastrar novos usuários administradores e consultar aqueles que já fazem parte do sistema. A Figura 19 dispõe graficamente esse cenário.

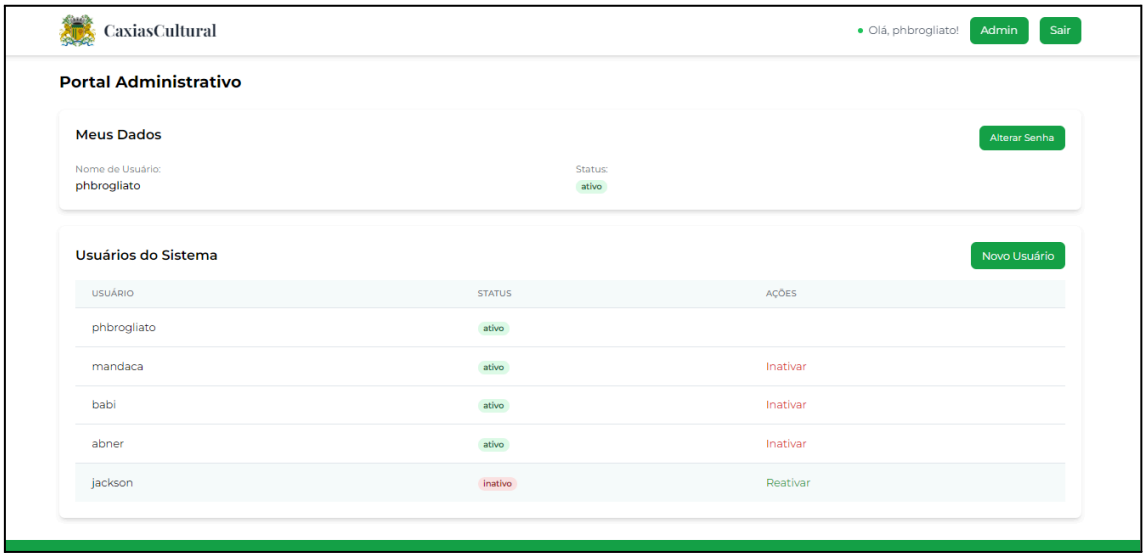

Figura 19. Tela de gerenciamento de usuários administradores.

Fonte: Autoria Própria, 2025.

É importante destacar que o CaxiasCultural permite que um administrador inative outros administradores, desde que se mantenham dois administradores ativos no total. Além disso, quando um novo usuário for cadastrado, o mesmo recebe o status 'pre', inferindo uma espécie de pré-cadastro, e uma senha inicial, que deverá ser alterada no primeiro acesso.

6. Considerações Finais

Este artigo teve como objetivo primário a construção de uma solução para a apresentação e o gerenciamento do Patrimônio Histórico e Cultural tombado no município de Caxias do Sul. A Constituição Federal de 1988 e o corpo de leis municipais existentes em Caxias do Sul asseguram o acesso à cultura aos cidadãos, um escopo que pode ser facilitado e escalonado pela Tecnologia da Informação. Considerando este cenário, confeccionou-se o CaxiasCultural, um sistema *Web* que atua na apresentação de patrimônios históricos tombados em um mapa interativo e em *cards*, divulgando informações como status do patrimônio, registro de tombamento, competência (municipal, estadual, federal) do tombamento, fotos e interface de rotas para os patrimônios.

O desenvolvimento do *software* proposto foi concluído com êxito, com algumas pequenas modificações que não prejudicaram a concepção original de uso da plataforma de gerenciamento de bens tombados pelo Patrimônio Histórico e Cultural de Caxias do Sul. A potencialidade do CaxiasCultural quanto à organização e centralização dos processos relacionados ao Patrimônio Histórico está em consonância com o propósito cultural explicitado na Constituição Federal de 1988 a ser estabelecido pelo poder público, consolidando-se como uma ferramenta viável para o fortalecimento da herança cultural e da vivência da cidade de Caxias do Sul.

Os resultados aqui apresentados foram fruto da confecção do *software* CaxiasCultural em ambiente de desenvolvimento, estabelecendo um produto mínimo viável; portanto, suas funcionalidades atuais não foram disponibilizadas abertamente na *Web* e são acessíveis apenas localmente. Para trabalhos futuros, além da condução de testes de validação com usuários reais e a implementação da ferramenta em ambiente produtivo, sugere-se a implementação de padrões de acessibilidade, como a Iniciativa de Acessibilidade *Web* (WAI – *Web Accessibility Initiative*), liderada pelo Consórcio da *World Wide Web*, bem como a complementação de catalogação de todos os bens tombados a nível municipal e a inclusão e manutenção dos patrimônios tombados na esfera estadual em Caxias do Sul. Com relação a este último movimento, a verificação da responsabilidade de gestão destes bens dentro da plataforma é necessária, visto que o seu tombamento não é de competência da administração municipal.

Ainda, verifica-se um grande potencial de universalização da ferramenta para abarcar não somente bens históricos localizados na cidade de Caxias do Sul, mas também nas demais esferas municipais, estaduais e federal do Brasil; também se faz possível a inclusão de bens imateriais, ou até mesmo a composição de rotas turísticas na interface desenhada para o CaxiasCultural; nestes cenários, é importante avaliar o funcionamento do gerenciamento de usuários e a plataforma das rotas de patrimônio-usuário, para atestar seu comportamento em configurações de maior complexidade operacional, bem como fomentar a colaboração entre os diversos níveis do poder público e comunidade em geral para que a expansão se torne exequível.

7. Referências

BCRYPT. **Documentação**. Disponível em: https://www.npmjs.com/package/bcrypt. Acesso em 19 mai. 2025.

BRASIL. Constituição (1988). **Constituição da República Federativa do Brasil**. Disponível em: https://www.planalto.gov.br/ccivil_03/constituicao/constituicaocompilado.htm. Acesso em: 5 dez. 2024.

CAXIAS DO SUL. **Divisão de Proteção ao Patrimônio Histórico e Cultural - DIPPAHC**. Disponível em: https://caxias.rs.gov.br/servicos/cultura/dipphac. Acesso em: 5 dez. 2024.

CAXIAS DO SUL. Secretaria Municipal da Cultura. Divisão de Proteção ao Patrimônio Histórico e Cultural - DIPPAHC. **Relação dos Bens Tombados pelo Patrimônio Histórico e Cultural de Caxias do Sul**. Município de Caxias do Sul, 2025. Disponível em: https://gcpstorage.caxias.rs.gov.br/documents/2025/02/5beab27c-2b30-4ec7-8b36-ad6e6382c8f9.pdf. Acesso em: 17 mai. 2025.

EXPRESS.JS. **Repositório oficial do Express.js no GitHub.** OpenJS Foundation, 2024. Disponível em: https://github.com/expressjs/express. Acesso em: 5 dez. 2024.

GIT. **Free and open source distributed version control system**. Git, 2025a. Disponível em: https://git-scm.com/. Acesso em 01 jun. 2025.

GIT. **Sobre Controle de Versão**. Git, 2025b. Disponível em: https://git-scm.com/book/pt-br/v2/Come%C3%A7ando-Sobre-Controle-de-Vers%C3%A3o. Acesso em 19 mai. 2025.

GITHUB. **Sobre repositórios**. GitHub Docs, 2025. Disponível em: https://docs.github.com/pt/repositories/creating-and-managing-repositories/about-repositories. Acesso em: 02 jun. 2025.

IBM. **Designing the build strategy**. IBM Documentation, 2025. Disponível em: https://www.ibm.com/docs/en/z-devops-guide?topic=pipeline-designing-build-strategy. Acesso em: 02 jun. 2025.

JWT.IO. **JSON Web Token Introduction**. Auth0, 2025. Disponível em: https://jwt.io/introduction. Acesso em 19 mai. 2025.

LEAFLET**. Biblioteca JavaScript de código aberto para mapas interativos, leve e fácil de usar**. Leaflet, 2024. Disponível em: https://leafletjs.com/. Acesso em: 5 dez. 2024.

MOZILLA FOUNDATION. MDN Web Docs. **First-class Function**. Mozilla Foundation, 2024a. Disponível em: https://developer.mozilla.org/en-US/docs/Glossary/First-class_Function. Acesso em: 5 dez. 2024.

MOZILLA FOUNDATION. MDN Web Docs. **JavaScript**. Mozilla Foundation, 2024b. Disponível em: https://developer.mozilla.org/en-US/docs/Web/JavaScript. Acesso em: 5 dez. 2024.

MOZILLA FOUNDATION. MDN Web Docs. **Sobre JavaScript**. Mozilla Foundation, 2023. Disponível em: https://developer.mozilla.org/pt-BR/docs/conflicting/Web/JavaScript. Acesso em: 5 dez. 2024.

MULTER. **Documentação**. OpenJS Foundation, 2025. Disponível em: https://github.com/expressjs/multer/blob/master/doc/README-pt-br.md. Acesso em 19 mai. 2025.

NODE.JS. **Introduction to Node.js**. OpenJS Foundation, 2024. Disponível em: https://nodejs.org/pt/learn/getting-started/introduction-to-nodejs. Acesso em: 5 dez. 2024.

OPEN HOUSE BSAS. **Catálogo 2024: catálogo digital do evento de arquitetura Open House Buenos Aires**. Open House BsAs, 2024. Disponível em: https://www.openhousebsas.org/catalogo-2024. Acesso em: 5 dez. 2024.

OPENSTREETMAP. **Sobre o OpenStreetMap: plataforma colaborativa de mapeamento global**. OpenStreetMap, 2024. Disponível em: https://www.openstreetmap.org/about. Acesso em: 5 dez. 2024.

ORACLE. **O que é MySQL?** Oracle, 2023. Disponível em: https://www.oracle.com/br/mysql/what-is-mysql/. Acesso em: 5 dez. 2024.

ORACLE. **O que é RDBMS (Banco de Dados Relacional)?** Oracle, 2014. Disponível em: https://www.oracle.com/br/database/what-is-a-relational-database/. Acesso em: 5 dez. 2024.

QUEENSLAND. **Schools Directory**. Department of Education of Queensland, 2025. Disponível em: https://schoolsdirectory.eq.edu.au/. Acesso em 19 mai. 2025.

REACT. **Built-in React Hooks**. Meta Platforms Inc., 2025. Disponível em: https://react.dev/reference/react/hooks. Acesso em 24 mai. 2025.

REACT. **Documentação oficial do React**. Meta Platforms Inc., 2024. Disponível em: https://pt-br.react.dev/. Acesso em: 5 dez. 2024.

SOMMERVILLE, Ian. **Engenharia de software**. 10. ed. São Paulo, SP: Pearson, 2018. *E-book*. Disponível em: https://plataforma.bvirtual.com.br. Acesso em: 5 dez. 2024.

TAILWINDCSS. **Documentação**. Tailwind Labs Inc., 2025. Disponível em: https://tailwindcss.com/docs/installation/using-vite. Acesso em: 19 mai. 2025.

UNITED NATIONS. **UNDP Social and Environmental Standards Toolkit**. Standard 4: Cultural Heritage. United Nations Development Programme, 2021. Disponível em: https://ses-toolkit.info.undp.org/standard-4. Acesso em: 5 dez. 2024.

VISUAL STUDIO CODE. **Code Editor**. Microsoft Corporation, 2025. Disponível em: https://code.visualstudio.com/. Acesso em 03 jun. 2025.

VITE. **Guia Geral de Uso**. VoidZero Inc., 2025. Disponível em: https://pt.vite.dev/. Acesso em: 19 mai. 2025.