UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Yuri Henrique Bernardes Maciel

Desenvolvimento de uma Plataforma Web para Simulações de Incêndios Florestais

Uberlândia, Brasil

2024

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Yuri Henrique Bernardes Maciel

Desenvolvimento de uma Plataforma Web para Simulações de Incêndios Florestais

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

Orientador: Luiz Gustavo Almeida Martins

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2024

Yuri Henrique Bernardes Maciel

Desenvolvimento de uma Plataforma Web para Simulações de Incêndios Florestais

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

Trabalho aprovado. Uberlândia, Brasil, 09 de maio de 2024:

Luiz Gustavo Almeida Martins
Orientador

Maria Adriana Vidigal de Lima

Bruno Augusto Nassif Travençolo

Uberlândia, Brasil
2024

Resumo

Este trabalho aborda o desenvolvimento de uma plataforma acessível e eficiente para o estudo e simulação do comportamento de incêndios florestais, visando facilitar tanto o desenvolvimento quanto o uso de modelos de simulação baseados em autômatos celulares. O sistema desenvolvido, *Forest Fire Simulation Web App* (FFSWA), permite a geração de reticulados a partir de mapas em imagens georreferenciadas, a visualização de simulações de incêndio e o download de dados brutos dessas simulações para análises mais detalhadas. Assim, espera-se que essa ferramenta auxilie, de forma efetiva, nas tomadas de decisão durante o planejamento de ações de prevenção e combate à problemática de incêndios florestais, especialmente no contexto do bioma do Cerrado brasileiro. Ao integrar os modelos de propagação de incêndios baseados em autômatos celulares desenvolvidos pelo Laboratório de Computação Bioinspirada (LCBio) da Universidade Federal de Uberlândia (UFU), o sistema desenvolvido possibilita o estudo e a simulação do comportamento de incêndios, visando fornecer dados úteis que auxiliem na prevenção e gestão desses eventos devastadores. Após o desenvolvimento, foi realizado um estudo de caso sobre o Parque Estadual do Pau furado, utilizando um dos simuladores de incêndio desenvolvidos pelo grupo de pesquisa. A partir dos testes realizados, observou-se que o sistema foi capaz de gerar reticulados de diferentes tamanhos e níveis de precisão, mapeando de forma satisfatória os tipos de vegetação representados nos mapas para as células do reticulado. A integração do sistema web com o modelo de propagação de incêndios permite simulações do espalhamento do fogo sobre o reticulado gerado, gerando dados que podem ser usados para suportar as tomadas de decisão durante o planejamento de políticas públicas de combate e prevenção de incêndios. O módulo de gestão de simuladores e regiões possibilita a adaptação do sistema a diferentes simuladores e mapas, tornando o sistema desenvolvido mais independente e flexível e permitindo sua utilização por diferentes grupos de pesquisa.

Palavras-chave: georreferenciamento, desenvolvimento de sistema web, autômatos celulares, modelos de propagação de incêndios.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	*Application Programming Interface* (Interface de Programação de Aplicação)
SPA	*Single Page Application* (Aplicação de Página Única)
REST	*Representational State Transfe*r (Transferência Representacional de Estado)
CSS	*Cascading Styles Sheets* (Folhas de Estilo em Cascata)
HTML	*Hypertext Markup Language* (Linguagem de Marcação de Hipertexto)
JS	JavaScript
TS	TypeScript
WSL	*Windows Subsystem for Linux* (Subsistema Windows para Linux)
LCBio	Laboratório de Computação Bioinspirada
UFU	Universidade Federal de Uberlândia
VS Code	Visual Studio Code
IDE	*Integrated Development Environment* (Ambiente de Desenvolvimento Integrado)
FFSWA	*Forest Fire Simulation Web App* (Aplicativo Web para Simulação de Incêndios Florestais)

Sumário

1 Introdução

Florestas ao redor de todo o mundo vêm sofrendo com incêndios e as mudanças que estes causam na vida selvagem. Os incêndios florestais ocorrem em sua maioria devido a atividades antropogênicas (MANSOOR et al., 2022), mas também podem ocorrer naturalmente em certos biomas, como o Cerrado brasileiro. Esse bioma é de extrema importância, uma vez que é um dos locais de maior biodiversidade no mundo e proporciona um importante equilíbrio hidrológico para o Brasil (SCHMIDT; ELOY, 2020). Apesar disso, nos últimos anos foi verificado um aumento considerável na quantidade de incêndios nesse bioma, causados de forma natural e pela ação humana.

Com isso em mente, o estudo do comportamento dos incêndios se torna um assunto relevante para que seja possível desenvolver métodos que ajudem no combate e na prevenção dos impactos que esses eventos podem causar na natureza. Ferreira et al. (2023) descrevem o uso de autômatos celulares como uma boa alternativa para a simulação de incêndios florestais e outros fenômenos naturais, devido a sua simplicidade de implementação, correspondência com os comportamentos naturais e computação emergente, baseada em iterações locais. Além disso, essa abordagem multiagente permite o desenvolvimento de aplicações com alto desempenho, pois cada célula do autômato evolui de forma independente e distribuída, possibilitando o uso de estratégias de multiprocessamento, já que simulações de sistemas dinâmicos podem demandar alto poder computacional. Entretanto, os modelos de propagação de incêndios baseados em autômato celular demandam a criação de um espaço celular (reticulado) que represente, de forma adequada, a região a ser monitorada e realizar esse mapeamento manualmente pode ser uma tarefa árdua e demorada, dependendo do tipo de informação que precisa ser extraída e representada em cada célula do reticulado.

Este trabalho tem como objetivo a geração de reticulados de autômatos celulares a partir de imagens de cobertura vegetal da área monitorada, os quais são usados pelos modelos de propagação de incêndio para simular o espalhamento do fogo, de acordo com a vegetação predominante em cada região. No processo de mapeamento, são empregadas técnicas de georreferenciamento a fim de identificar e mapear com alta precisão o tipo de cobertura vegetal presente em cada célula do reticulado, considerando o tamanho total da área selecionada e a granularidade do reticulado (tamanho de cada célula). A plataforma desenvolvida também permite a integração com diferentes modelos baseados em autômato celular. Para isso, foi concebido um módulo de gerenciamento que determina, entre outras coisas, o mapeamento entre os estados previstos no modelo para cada célula e os tipos de coberturas disponíveis no mapa de entrada, bem como a definição dos parâmetros de uma simulação. Assim, é possível simular diferentes cenários e analisar o

comportamento do incêndio em cada um. Por fim, o sistema permite a visualização da simulação, possibilitando acompanhar de forma visual a evolução do fogo pelo reticulado, bem como gera um arquivo texto com a situação do reticulado em cada passo de tempo do autômato celular. Durante o desenvolvimento da plataforma, buscou-se implementar técnicas para aprimorar as telas de interface, visando torná-la mais fácil e amigável a fim de aumentar a satisfação do usuário (PRATAMA; CAHYADI, 2020).

1.1 Justificativa

Autômatos celulares, que se mostram uma boa alternativa para simular incêndios, precisam como entrada de um reticulado que represente a área a ser monitorada. Esse reticulado se trata de um vetor ou matriz de duas ou mais dimensões, onde cada posição representa o estado de uma célula do autômato. Nos modelos de incêndio considerados neste trabalho, cada célula representa uma região quadrada pertencente à área selecionada e seu estado determina o tipo de cobertura do local (tipo de vegetação ou solo exposto), bem como sua situação em relação ao incêndio (preservado, com fogo ou queimado).

A construção desses reticulados pode ser uma tarefa trabalhosa, ainda mais quando se trata de representar um local real, com diversos elementos que representam a sua fisionomia vegetal. Dito isso, uma ferramenta que seja capaz de gerar a partir de uma imagem da área de interesse da simulação e de especificar a fisionomia vegetal de cada região, se torna de extrema utilidade.

Assim como para a entrada, o resultado da simulações dos modelos de autômatos celulares são expressos em reticulados, mas nesse caso um vetor deles. Ou seja, o simulador retorna uma sequência de reticulados, onde cada um deles representa o estado do incêndio em um dado momento do tempo, que é discretizado. Visualizar um vetor de reticulados e entender o que eles representam pode ser nada intuitivo, gerando a necessidade de uma ferramenta que possa ler a saída da simulação e gerar imagens que representam a região processada em cada momento do incêndio.

O Laboratório de Computação Bioinspirada (LCBio) - coordenado pela Professora Gina Maira Barbosa de Oliveira e residido na Universidade Federal de Uberlândia (UFU) - possui um projeto denominado Cerrado Resiliente, que realiza estudos e desenvolvimentos de modelos de simulação de incêndios baseados em autômatos celulares, focando na prevenção de incêndios no Cerrado. O presente trabalho tem como objetivo agregar valor para o projeto Cerrado Resiliente fornecendo uma ferramenta que gere reticulados a partir de imagens georreferenciadas, invoque os simuladores desenvolvidos pelo laboratório e gere imagens animadas que representam o resultado da simulação realizada. Além disso, o sistema desenvolvido, doravante Forest Fire Simulation Web App (FFSWA), deve ser capaz de fornecer o download dos dados brutos de entrada e saída bem como o cadas-

tro de diferentes regiões e simuladores, para que seja possível o estudo, a melhoria e o desenvolvimento de novos modelos de simulação.

1.2 Objetivos

O objetivo principal deste trabalho é fornecer uma plataforma WEB capaz de gerar reticulados a partir de mapas em imagens georreferenciadas, fornecer suporte para uso e estudo de diferentes modelos de simulação de incêndios e permitir que usuários solicitem e visualizem simulações geradas pelos modelos disponíveis.

Para que o objetivo principal seja concluído, os seguintes objetivos específicos devem ser realizados:

- Levantar requisitos funcionais e não-funcionais que o sistema deve possuir.
- Desenvolver método para visualização de simulações baseado em imagens GIF.
- Permitir o download de dados brutos gerados por modelos de simulação de incêndio baseado em autômatos celulares.
- Desenvolver plataforma com interface amigável para o solicitação e visualização de simulações de incêndio.
- Validar funcionamento do sistema desenvolvido através de um estudo de caso com o mapa de fisionomia vegetal do Parque Estadual do Pau Furado e um simulador baseado em autômato celular desenvolvido por Ferreira et al. (2023).

1.3 Organização do texto

O restante deste documento está estruturado da seguinte forma:

- **Referencial Teórico**: introduz as principais técnicas abordadas no desenvolvimento deste trabalho, bem como uma breve descrição de trabalhos similares disponíveis na literatura.
- **Desenvolvimento**: apresenta o procedimento realizado para o desenvolvimento da plataforma, seus requisitos e arquitetura utilizada.
- **Resultados**: descreve a plataforma desenvolvida, mostrando os telas da aplicação e como utilizá-la.
- **Conclusões**: aborda os objetivos alcançados pelo projeto e quais melhorias futuras podem ser feitas na plataforma.

2 Referencial Teórico

Esta capítulo descreve conceitos fundamentais para o entendimento deste trabalho. Além disso, são apresentados trabalhos correlatos, que podem servir de inspiração ou base para a produção deste projeto.

2.1 Autômatos Celulares

Automato celular é um modelo matemático composto por um conjunto de elementos mais simples denominados células que, através do comportamento em conjunto, conseguem simular o comportamento de fenômenos mais complexos (WOLFRAM, 1984). O conjunto de células que compõem o automato é denominado reticulado. Cada célula do reticulado possui um estado e seu valor é determinado por uma função de transição a cada passo de evolução do autômato celular. Todas as células do reticulado evoluem em paralelo, seguindo o comportamento definido pela função de transição, que por sua vez, considera não só o estado atual da célula em análise (central), como das suas vizinhas, determinadas de acordo com o raio de alcance (r) adotado (BERTO; TAGLIABUE, 2023). A vantagem dos autômatos celulares está na sua simplicidade de implementação, onde as células são estruturas simples e de fácil manutenção, mas seu trabalho em conjunto conseguem entregar resultados poderosos para representar diferentes fenômenos, mesmo os mais complexos.

O reticulado utilizados por autômatos celulares possuem tamanho finito e por conta dessa natureza, uma borda é definida. As células de borda precisam ter tratamento especial quanto ao modo de interpretar sua vizinhança, uma vez que a vizinhança tem influência na evolução do autômato (FERREIRA, 2023). Os três principais métodos de tratamento de vizinhança para células de borda são descrita a seguir.

- Bordas cíclicas ou periódicas: as extremidades do reticulado são conectadas. No momento de interpretar a vizinhança, podemos “dar a volta” no reticulado para pegar os valores que faltam.
- Bordas reflexivas: os valores das próprias células de borda são utilizados para completar a vizinhança.
- Bordas fixas: um estado comum é utilizado para completar a vizinhança de todas as células de borda.

Quando se trata de autômatos celulares, o modelo unidimensional é o mais simples. Nesse modelo, o reticulado é representado por um simples vetor e cada célula tem como vizinhança as células laterais, utilizando um raio para determinar a quantidade de células que irão compor a vizinhança (FERREIRA, 2023). Para um automato celular unidimensional com vizinhança de raio 1, a vizinhança será composta por uma célula à esquerda e uma à direita, enquanto que para raio 2, teríamos duas células à esquerda e duas à direita, e assim por diante.

Autômatos celulares bidimensionais possuem um reticulado representado por uma matriz e, comumente, podem utilizar dois tipos de vizinhança (apresentados na Figura 1): von Neumann e Moore (FERREIRA, 2023). A vizinhança de Moore considera todas as células que cercam a célula central como vizinhas. Em contra partida, a vizinhança de von Neumann considera somente as células cardinais em relação à célula central como vizinhas. O mesmo tipo de vizinhança podem ser replicado para autômatos tridimensionais, como demonstrado na Figura 2.

(a) Vizinhança de von Neumann (b) Vizinhança de Moore

Figura 1 – Vizinhanças comumente utilizadas em automatos bidimensionais (Extraído de (FERREIRA, 2023))

(a) Vizinhança de von Neumann (b) Vizinhança de Moore

Figura 2 – Generalização das vizinhanças de um automato bidimensional para um tridimensional (Extraído de (FERREIRA, 2023))

2.1.1 Modelagem de Incêndios com Automatos Celulares

Softwares de simulação de incêndio são importantes, pois permitem cientistas observarem esses fenômenos de forma não destrutiva (Hernández Encinas et al., 2007). Com

isso em mente, o uso de autômatos celulares se mostra eficiente na construção desses programas, pois são capazes de representar comportamentos complexos e dinâmicos com uma complexidade de implementação relativamente baixa. A eficiência e importância de autômatos celulares para este caso é perceptível devido ao seu alto uso em artigos relacionados à simulação de incêndio, como os feitos por Ferreira et al. (2023), Alexandridis et al. (2008), Mutthulakshmi et al. (2020), Alexandridis et al. (2008) e Hernández Encinas et al. (2007).

Para este trabalho, o modelo desenvolvido por Ferreira et al. (2023) no LCBio possui maior importância, pois será utilizado para um estudo de caso a fim de validar a plataforma desenvolvida. O modelo em questão adota a estratégia de representar a área da floresta em uma matriz bidimensional, como exemplificado na Figura 3, de modo que cada célula representa uma área que pode estar queimada ou não, transitando de estado de acordo com o estado das células vizinhas (vizinhança de Moore).

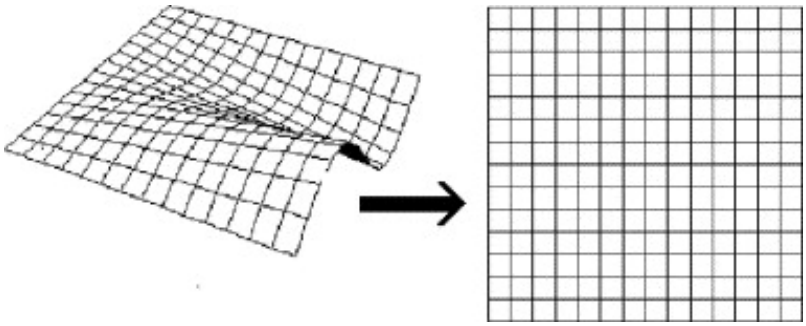


Figura 3 – Área de uma floresta como matriz bidimensional (Extraído de Hernández Encinas et al. (2007))

Ferreira et al. (2023) desenvolveram um modelo estocástico, onde as regras de propagação de incêndio - utilizadas na função de transição das células - descrevem probabilidades de uma célula entrar em combustão baseado no estado das células vizinhas, na fisionomia vegetal presente na área que a célula representa, direção do vento e umidade do local. As fisionomias vegetais suportadas são "floresta", "savânica"e "campestre", todas presentes no Cerrado brasileiro, conforme apresentado na 4.

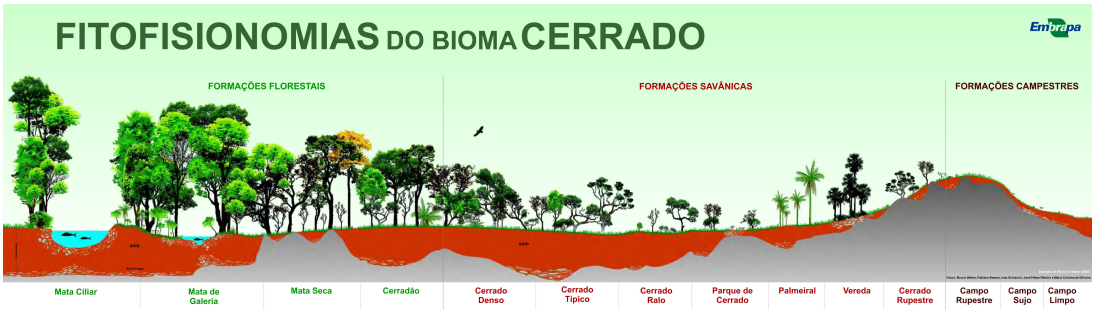


Figura 4 – Fisionomias vegetais do Bioma Cerrado (Extraído de Ribeiro e Walter (2008))

O modelo desenvolvido se mostrou uma melhoria para a simulação de incêndios com automatos celulares, pois consegue representar com mais precisão as características do Cerrado brasileiro. As melhorias responsáveis pela capacidade do modelo de lidar com diferentes cenários de incêndio são: conter estados baseados na fisionomia vegetal do bioma e o uso de coeficiente de umidade para controlar a intensidade das chamas.

2.2 Georreferenciamento

Georreferência se trata de uma técnica onde uma associação é feita entre um objeto e um dado geográfico, como ruas, pontos de interesse (POIs), construções ou qualquer tipo de local com localização geográfica. Neste caso, objeto pode ser definido como qualquer tipo de dado que possa ter associação com informações geográficas. Durante o georreferenciamento é necessário utilizar um sistema de referência em geodésia como o *World Geodetic System 1984* (WGS-84). Esse sistema nos permite mapear a superfície da Terra em um sistema de coordenadas tridimensional, onde cada coordenada recebe um valor numérico que, em conjunto, representam um determinado local do planeta - formando uma coordenada geográfica (HACKELOEER KLAAS KLASING; MENG, 2014).

2.2.1 Georeferenciamento de *Rasters*

Imagens *raster* (ou *bitmap*) são um tipo comum de objeto a ser georreferenciado. Essas imagens são formadas por *pixels*, que são pequenos pontos de visualização que representam uma determinada cor. O número de *pixels* de uma imagem *raster* determina sua resolução. Quanto maior o número de *pixels* em uma imagem, maior sua resolução e quanto menor for esse número, menor sera a resolução, vice-versa. Se o tamanho de apresentação de uma imagem for aumentado sem aumentar sua resolução, ela perderá detalhes e, eventualmente, parecerá embaçada ou *pixelizada* - os *pixels* serão visíveis, dando um aspecto quadriculado para a imagem (ADOBE, 2024).

Ao georreferenciar *rasters*, faz-se uma associação entre um ponto de visualização na imagem e um local do mapa, utilizando pontos de controle terrestres (GCPs - *Ground Control Points*). Para escolher um GCP, basta demarcar locais que possam ser identificados, tanto na imagem quanto no mapa. Um software de georreferenciamento, como o QGIS (QGIS, 2024), permite que essa associação seja feita e, com base nos pontos demarcados, mapeia cada um dos *pixels* da imagem a uma coordenada geográfica. Dessa forma, cada *pixel* irá representar um local da região mapeada, possuindo um certo nível de precisão a depender dos GCPs escolhidos e da resolução da imagem utilizada. A imagem georreferenciada é denominada GeoTiff, uma imagem no formato de arquivo de imagem marcado (TIFF - *Targged Image File Format*), que contém as informações geográficas embutidas (QGIS, 2024). A Figura 5 mostra um exemplo de georreferenciamento sendo

feito com o QGIS.

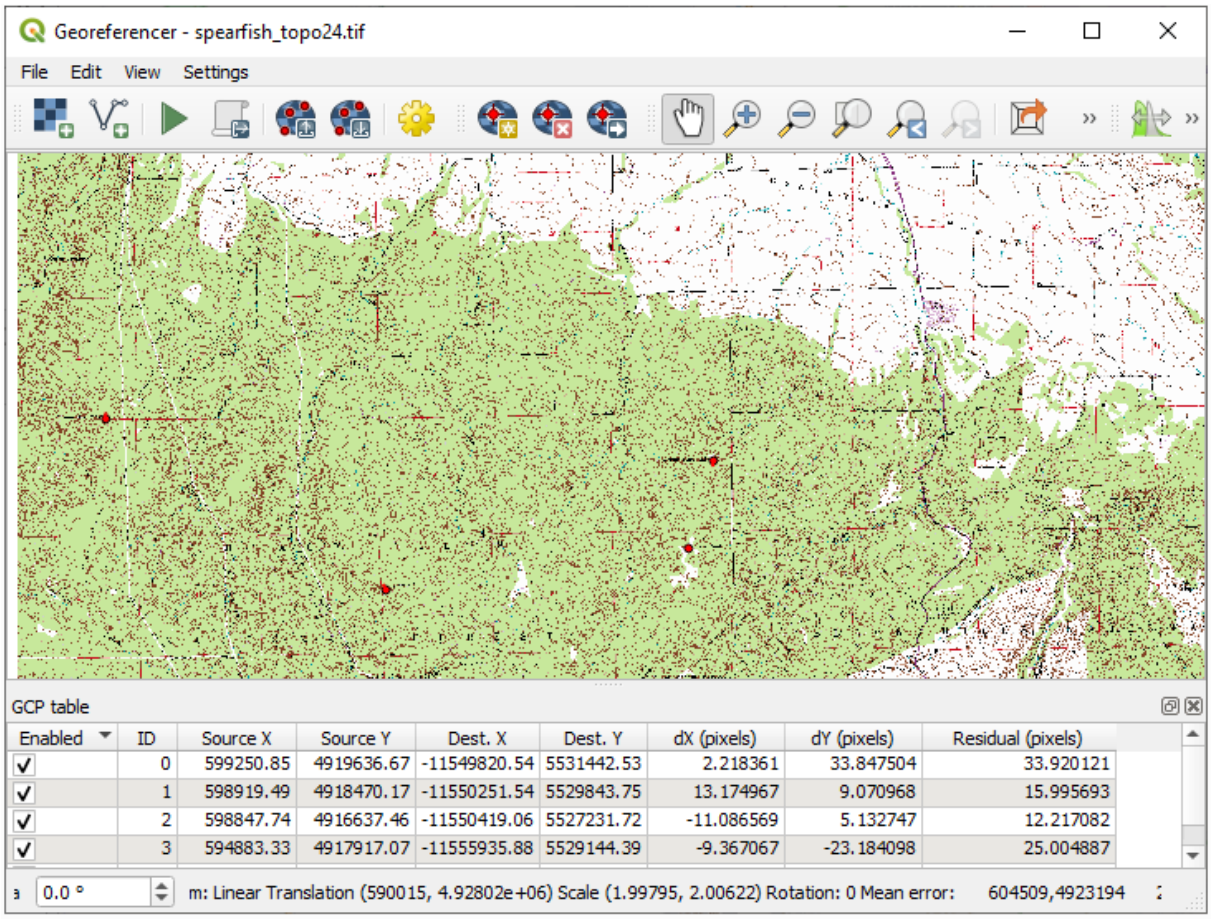


Figura 5 – Janela de georreferenciamento do QGIS (Extraído de QGIS (2024))

Hackeloeer Klaas Klasing e Meng (2014) descrevem diferentes cenários onde o georreferenciamento é utilizado em aplicações (Figura 6). Os cenários apresentados e os usos relacionados estão descritos a seguir.

- **Manipulação de mapas**: Criar novos objetos no mapa ou realizar uma atualização incremental do conteúdo já presente nele.
- **Criação de mapas**: Criação de mapas a partir do zero, seja criando entidades geográficas e referências com base em imagens de satélite ou imagens aéreas, ou combinando dois ou mais mapas de origem em um novo mapa.
- **Validação de mapas**: Avaliação de propriedades em mapas, como correção geométrica ou presença de uma forma normal em relação a certas propriedades topológicas. A validação automática visa validar um mapa de acordo com métricas de qualidade, sem a necessidade de assistência humana.

- **Correspondência de mapas**: Associação de um ponto em um mapa, necessário para navegação, por exemplo, em veículos ou para pedestres, e em outros aplicativos que requerem posicionamento exato, como serviços baseados em localização.
- **Georreferenciamento de mídias**: Atrelar imagens, vídeos ou documentos a uma localização geográfica. Permite mostrar a qual local a mídia está associada.

Utilizar o georreferenciamento de mídias, em *rasters*, nos permite mostrar ao usuário de uma aplicação qual parte do mundo essa imagem representa. Em nosso contexto, georreferenciar regiões de interesse visa determinar a área onde uma simulação de incêndio irá ocorrer, bem como apresentar os resultados sobre um mapa desse local, gerando uma visualização do incêndio sobre a área de interesse.

Figura 6 – Casos de uso de georreferenciamento (Extraído de Hackeloeer Klaas Klasing e Meng (2014))

2.3 Interface e Experiência do Usuário

A Interface do Usuário (UI - *User Interface*), pode ser descrita como o meio pelo qual uma usuário consegue interagir com o sistema. Em outras palavras, a UI é responsável por definir a identidade visual do sistema, englobando a escolha de cores e de *layout* da aplicação, bem como os meios de resposta às interações e solicitações do usuário (IXDF, 2016b). Por outro lado, a Experiência do Usuário (UX - *User Experience*) está relacionada com a usabilidade que um sistema tem, ou seja, se o este é intuitivo, eficiente e se encaixa com as necessidade do usuário (IXDF, 2016a).

Em um artigo sobre importância da UI e da UX, Pratama e Cahyadi (2020) discutem como esses aspectos influenciam um usuário a utilizar/comprar uma aplicação.

Os autores observaram as avaliações e quantidade de *downloads* de duas aplicações de bloco de notas na loja em que foram publicadas. A aplicação A possui uma interface mais simples e intuitiva, enquanto a aplicação B possui uma interface mais complexa e inconsistente, além de apresentar uma funcionalidade de leitura de texto que não é fornecida pela aplicação A. Ao final do experimento, foi constatado que, mesmo possuindo uma funcionalidade extra, a aplicação B teve uma avaliação média de 3,4, de um máximo de 5, e cerca de dez mil *downloads*, enquanto a aplicação A obteve avaliação média de 4,8 e mais de cem milhões de *downloads*. Com base nos resultados, é possível ver o quão impactante pode ser uma boa interface e influencia na vontade dos usuários em utilizar uma aplicação. Desse modo, desenvolver uma boa interface (agradável e amigável) para utilizar e desenvolver os softwares de simulação se mostra uma tarefa importante para despertar o interesse de usuários potenciais e melhorar sua usabilidade.

3 Desenvolvimento

Neste capítulo são descritas as etapas de projeto e desenvolvimento da plataforma proposta, a qual é chamada *Forest Fire Simulation Web App* (FFSWA). Inicialmente, são apresentadas todas as características fundamentais do sistema desenvolvido, bem como o processo percorrido para o desenvolvimento de tal.

3.1 Requisitos do Sistema

A partir de reuniões realizadas com membros do Laboratório de Computação Bio-inspirada (LCBio) da UFU, participantes do Projeto Cerrado Resiliente, foi construído o diagrama de casos de uso apresentado na Figura 7, contendo os principais atores que atuam no sistema e definindo os requisitos funcionais contemplados no escopo deste trabalho. A descrição de cada um desses requisitos é dada a seguir:

- **Cadastrar simuladores**: usuário administrador deve ser capaz de cadastrar modelos de autômatos celulares que serão utilizados para realizar as simulações de incêndio. Esse caso envolve o carregamento (*upload*) do programa que realiza a simulação e é executado do lado do servidor.
- **Cadastrar regiões de simulação**: usuário administrador deve ser capaz de cadastrar as regiões que podem ser utilizadas para especificar a área de simulação. Esse caso envolve o *upload* de um *raster* georreferenciado que descreve a região.
- **Atualizar *raster* das regiões cadastradas**: Usuário administrador pode substituir os *rasters* de regiões previamente cadastradas.
- **Criar simulação**: usuário pode criar uma simulação inserindo um nome, dados do simulador (ex: umidade, direção do vento, etc.) e granularidade do reticulado. Além disso, ele tem que determinar a área do mapa selecionado que será utilizada na simulação bem como indicar os pontos iniciais de foco de incêndio.
- **Realizar simulação de incêndio**: o sistema deve se comunicar com o simulador escolhido, solicitando a simulação do incêndio utilizando os dados de entrada de acordo com as especificações definidas pelo usuário no caso "Criar simulação". Essa troca de dados entre o sistema e o simulador deve ser feita via arquivos.
- **Visualizar resultado das simulações**: A partir de uma lista com todas as simulações cadastradas no sistema, o usuário deve selecionar aquela desejada que, por sua vez, será exibida sobre o mapa correspondente, caso esteja finalizada.

- ***Download* das simulações**: O usuário pode baixar (*download*) os dos dados brutos de uma simulação específica, caso ela esteja finalizada. Os dados brutos referem-se aos arquivos de entrada e de saída trocados entre o simulador e o sistema. O arquivo de entrada deve conter o reticulado do autômato celular gerado a partir de uma área selecionada no mapa, bem como os valores dos parâmetros de configuração do modelo, os quais determinam as características consideradas na simulação. O arquivo de saída deve conter a situação do reticulado em diferentes momentos da simulação, possibilitando que o usuário estude o comportamento do incêndio simulado.

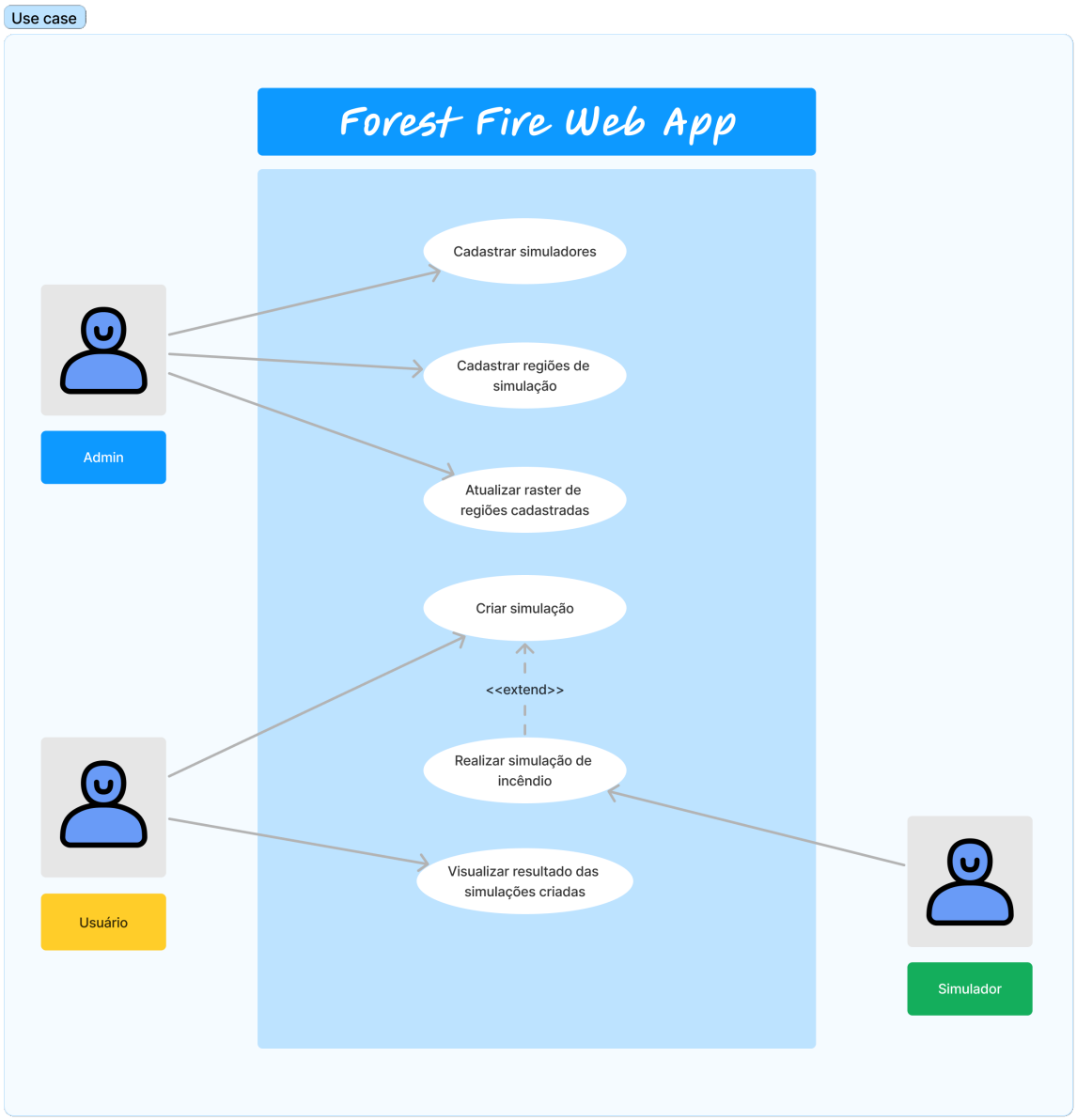


Figura 7 – Diagrama de casos de uso da aplicação

Com o intuito de fornecer uma boa experiência para os usuários da aplicação também foram definidos alguns requisitos não-funcionais, os quais são descritos a seguir:

- **Tempo de resposta**: o sistema deve fornecer um bom tempo de resposta, tanto em relação às ações do usuário, quanto ao tempo de carregamento das páginas. Segundo Nah (2004), a sensação de espera surge após 2 segundos. Portanto, com exceção da simulação em si, que depende de um componente externo (modelo de simulação) e costuma ter um custo computacional alto, espera-se que as demais tarefas do sistema devem performar abaixo desse tempo (2 segundos).
- **A interface deve seguir padrões de projetos modernos**: pode ser alcançado com o uso de alguma biblioteca de componentes de UI. Bibliotecas desse tipo fornecem componentes reutilizáveis que contemplam recursos de acessibilidade e conceitos atuais de UI e UX. No desenvolvimento da nossa plataforma foi empregada a biblioteca NextUI (NEXTUI, 2024).
- **Portabilidade**: é importante que a aplicação possa ser utilizada em uma gama variadas de dispositivos, fornecendo fácil acesso aos usuários. Portanto, a plataforma deve ser concebida para funcionar tanto em computadores, quanto em dispositivos móveis.

3.1.1 Especificação de Requisitos

Após o levantamento inicial dos requisitos do sistema, foi feito a especificação detalhada de cada um dos requisitos identificados. Nessa etapa, foram desenvolvidos quadros que apresentam os fluxos normais e alternativos de cada processo executado na plataforma. Além disso, cada especificação de requisito apresenta a descrição, os atores envolvidos, o evento iniciador do processo, a pré-condição e a pós-condição dos eventos (se houver).

A Tabela 1 apresenta a especificação para o requisito “Cadastrar simuladores”.

<table>
<tr><td colspan="2">Requisito:
Cadastrar simuladores
Descrição:
O usuário administrador consegue utilizar o painel administrativo para cadastrar um novo modelo de autômato celular que poderá ser utilizado para realizar simulações de incêndio.
Atores envolvidos:
Administrador.
Evento iniciador:
Administrador acessa a página de cadastro de simuladores.
Pré-condição:
Usuário administrador deve estar autenticado na plataforma.
Sequência de eventos:
1. Ator acessa página de cadastro de simuladores na plataforma.
2. Ator preenche formulário, fornecendo informações como: nome do modelo de autômato celular, arquivo executável que será carregado para o servidor, valores (estados de células) suportados pelo modelo e suas respectivas cores e valor (estado) padrão.
3. Usuário clica no botão “Cadastrar”.
4. Front-end do sistema faz a validação dos dados e chama API no endpoint responsável pela cadastro do simulador.
5. API cadastra dados no banco de dados e retorna mensagem de sucesso. O formulário é limpo automaticamente.</td></tr>
<tr><td>Fluxos Alternativos:</td><td>Descrição:</td></tr>
<tr><td>4.1. Dados faltando ou inválidos</td><td>4.1.1. Ator preenche formulário de forma incompleta ou incorreta.
4.1.2. Front-end detecta dados faltando ou inválidos no momento da validação.
4.1.3. Front-end apresenta mensagem solicitando correção dos dados no formulário.
4.1.4. Retorna ao passo 2 do fluxo principal.</td></tr>
<tr><td>5.1. API retorna mensagem de erro</td><td>5.1.1. Ocorre erro durante o processamento dos dados enviados pelo front-end para o cadastro do simulador.
5.1.2. API retorna mensagem de erro com mensagem descrevendo o problema ocorrido.
5.1.3 Front-end apresenta mensagem de erro, solicitando que usuário tente realizar o cadastro novamente mais tarde.
5.1.4. Retorna ao passo 2 do fluxo principal.</td></tr>
</table>

Tabela 1 – Especificação para “Cadastrar simuladores”

A Tabela 2 apresenta especificação detalhada do requisito “Cadastrar regiões de simulação”.

<table>
<tr><td colspan="2">Requisito:
Cadastrar regiões de simulação
Descrição:
Usuário administrador acessa página de cadastro de regiões de simulação e realiza o cadastro de uma nova região.
Atores envolvidos:
Administrador.
Evento iniciador:
Administrador acessa página de cadastro de regiões de simulação.
Pré-condição:
Usuário administrador deve estar autenticado na plataforma.
Sequência de eventos:
1. Ator acessa página de cadastro de regiões na plataforma.
2. Ator preenche formulário, fornecendo informações como: nome da região de simulação, arquivo de imagem georreferenciado do mapa de fisionomia vegetal da região que será carregado para o servidor, cores (presentes no raster da região) e suas respectivas descrições.
3. Usuário clica no botão “Cadastrar”.
4. Front-end do sistema faz a validação dos dados e chama API no endpoint responsável pela cadastro da região.
5. API cadastra dados no banco de dados e adiciona mensagem na fila de processamento de rasters. O formulário é limpo automaticamente.</td></tr>
<tr><td>Fluxos Alternativos:</td><td>Descrição:</td></tr>
<tr><td>4.1. Dados faltando ou inválidos</td><td>4.1.1. Ator preenche formulário de forma incompleta ou incorreta.
4.1.2. Front-end detecta dados faltando ou inválidos no momento da validação.
4.1.3. Front-end apresenta mensagem solicitando correção dos dados no formulário.
4.1.4. Retorna ao passo 2 do fluxo principal.</td></tr>
<tr><td>5.1. API retorna mensagem de erro</td><td>5.1.1. Ocorre erro durante o processamento dos dados enviados pelo front-end.
5.1.2. API retorna mensagem de erro descrevendo o problema ocorrido.
5.1.3 Front-end apresenta mensagem de erro, solicitando que usuário tente realizar o cadastro novamente.
5.1.4. Retorna ao passo 2 do fluxo principal.</td></tr>
</table>

Tabela 2 – Especificação para “Cadastrar regiões de simulação”

A especificação completa do requisito "Atualizar *raster* das regiões cadastradas" é apresentada na Tabela 3.

<table>
<tr><td colspan="2">Requisito:
Atualizar raster das regiões cadastradas
Descrição:
Administrador acessa página de visualização de regiões cadastrados e solicita atualização de raster da região desejada.
Atores envolvidos:
Administrador.
Evento iniciador:
Administrador acessa página de visualização de regiões cadastradas.
Pré-condição:
Usuário administrador autenticado na plataforma.
Sequência de eventos:
1. Ator acessa lista de regiões no painel administrativo e lica no botão laranja com duas setas circulares.
2. Front-end exibe janela suspensa com um formulário contendo um único campo para especificar novo arquivo de imagem a ser carregado para o servidor.
3. Ator preenche o formulário e clica no botão "Enviar".
4. Front-end faz chamada para API enviando o arquivo especificado pelo ator.
5. API armazena arquivo recebido, coloca uma mensagem na fila de processamento de rasters e retorna mensagem de sucesso.
6. Janela suspensa é fechada.</td></tr>
<tr><td>Fluxos Alternativos:</td><td>Descrição:</td></tr>
<tr><td>3.1. Usuário não especifica arquivo para upload</td><td>3.1.1. Usuário clica no botão "Enviar" sem fornecer arquivo para upload.
3.1.2. Front-end não realiza chamada para API e aguarda pelo preenchimento do formulário.
3.1.3. Retorna ao passo 3 do fluxo principal.</td></tr>
<tr><td>5.1. API retorna mensagem de erro</td><td>5.1.1. Ocorre erro durante o processamento dos dados enviados pelo front-end.
5.1.2. API retorna mesagem de erro descrevendo o problema ocorrida.
5.1.3. Front-end apresenta mensagem de erro, solicitando que o usuário tente realizar a ação novamente mais tarde.
5.1.4. Retorna ao passo 3 do fluxo principal.</td></tr>
</table>

Tabela 3 – Especificação para "Atualizar *raster* das regiões cadastradas"

O requisito "Criar simulação" tem sua especificação definida na Tabela 4.

Requisito:
Criar simulação
Descrição:
Usuário autenticado cria uma simulação que é executada por um modelo de autômato celular selecionado sobre uma região de interesse especificada.
Atores envolvidos:
Usuário autenticado.
Evento iniciador:
Usuário acessa página de criação de simulações.
Pré-condição:
Usuário deve estar autenticado na plataforma.
Sequência de eventos:
1. Ator acessa página de criação de simulação, utilizando a barra de navegação.
2. Ator preenche formulário presente na página inserindo informações como: nome da simulação, umidade, direção do vento, região de interesse e modelo de autômato celular a ser utilizado.
2.1. Quando uma região de interesse é selecionada, o front-end requisita a imagem da região e exibe ela em um mapa de imagem de satélite.
2.2. O ator deve desenhar sobre o mapa a área de interesse para a simulação a ser criada.
2.3. Ator demarca os pontos inciais de foco de incêndio.
3. Ator clica no botão "Criar simulação".
4. Front-end valida dados do formulário e os envia para a API.
5. API cadastra informações recebidas no banco de dados, insere mensagem na fila de processamento de simulações e retorna mensagem de sucesso.

Fluxos Alternativos:	**Descrição**:
4.1. Formulário contém dados inválidos ou não inseridos	4.1.1. No momento da validação, o front-end encontra dados inválidos ou não inseridos. 4.1.2. Mensagem de erro é exibida para o usuário, solicitando que o formulário seja revisado e os dados corrigidos. 4.1.3. Retorna ao passo 2 do fluxo original.
5.1. API retorna mensagem de erro	5.1.1. No momento do cadastro a API encontra algum tipo de erro não esperado. 5.1.2. API retorna uma mensagem de erro para o front-end, especificando o problema encontrado. 5.1.3. Mensagem de erro é exibida para o usuário, solicitando que ele preencha o formulário e tente novamente mais tarde. 5.1.4 Retorna ao passo 2 do fluxo original.

Tabela 4 – Especificação para "Criar simulação"

A Tabela 5 contém a especificação do requisito “Realizar simulação de incêndio”.

<table>
<tr><td colspan="2">Requisito:
Realizar simulação de incêndio
Descrição:
Simulador cadastrado realiza simulação de incêndio utilizando arquivo de entrada especificado.
Atores envolvidos:
Simulador (modelo de autômato celular para simulação de incêndios).
Evento iniciador:
Instância responsável (worker) lê mensagem da fila de processamento de incêndios e invoca o simulador para realizar a tarefa.
Pré-condição:
Usuário da plataforma cria uma simulação de incêndio e API cadastra mensagem solicitando seu processamento na fila de mensagens.
Sequência de eventos:
1. Ator é invocado como processo filho do worker, recebendo, via linha de comando, os argumentos especificando os caminhos dos arquivos de entrada e saída.
2. O ator lê o reticulado e os parâmetros necessários do arquivo de entrada realiza a simulação de incêndio.
3. Ao final da simulação, os dados de saída são escritos no arquivo de saída especificado via linha de comando.
4. Processo filho que executa a simulação é finalizado e retorna código de saída 0 (sucesso).</td></tr>
<tr><td>Fluxos Alternativos:</td><td>Descrição:</td></tr>
<tr><td>4.1. Processo filho retorna código de erro</td><td>4.1.1. O simulador encontra algum erro durante sua execução.
4.1.2 Processo que executa a simulação retorna código de erro diferente 0 (falha).
4.1.3 Worker interpreta código e atualiza estados da simulação para falha.</td></tr>
</table>

Tabela 5 – Especificação para “Realizar simulação de incêndio”

A Tabela 6 contém a especificação do requisito “Visualizar resultado das simulações”.

Requisito: Visualizar resultado das simulações **Descrição**: Na página de visualização de simulações, uma simulação é selecionada pelo usuário e exibida sobre o mapa. **Atores envolvidos**: Usuário autenticado. **Evento iniciador**: Usuário acessa a página de visualização de simulações da plataforma. **Pré-condição**: Usuário deve estar autenticado na plataforma. **Sequência de eventos**: 1. Ator acessa página de visualização de simulações da plataforma. 2. Front-end solicita dados das simulações feita pelo usuário à API. 3. Ator seleciona uma simulação na lista exibida na página. 4. Front-end solicita o resultado da simulação selecionada à API. 5. Caso a simulação esteja finalizada, API retorna seus dados de localização (onde ela foi feita), imagem GIF (contendo animação da simulação) e a legenda da imagem. 6. Front-end exibe imagem GIF sobre um mapa de satélite, no local onde a simulação foi utilizada. Em conjunto é exibida a legenda que descreve o significado de cada uma das cores presentes no GIF.	
Fluxos Alternativos:	**Descrição**:
3.1. Usuário não possui simulações na plataforma	3.1.1. API não retorna dados de simulações para o usuário autenticado. 3.1.2. Front-end exibe um botão que permite o usuário ir para a página de criação de simulações.
5.1. Simulação selecionada não está finalizada.	5.1.1. API faz consulta no banco de dados e verifica que a simulação solicidade não está finalizada. 5.1.2. API não retorna dados da simulação. 5.1.3 Front-end exibe o *card* da simulação escolhida com cor diferente (para demonstrar que está selecionado), mas não exibe nada sobre o mapa.

Tabela 6 – Especificação para “Visualizar resultado das simulações”

A Tabela 7 apresenta a especificação completa do requisito “Download das simulações”

Requisito: Download das simulações **Descrição**: Usuário realiza o download dos dados de uma simulação finalizada na página de visualização de simulações. **Atores envolvidos**: Usuário autenticado no sistema. **Evento iniciador**: Usuário acessa a página de visualização de simulações e clica no ícone de *download* em um *card* de uma simulação finalizada. **Pré-condição**: Usuário está autenticado no sistema e possui ao menos uma simulação finalizada. **Sequência de eventos**: 1. Front-end requisita endereço (*link*) para acesso aos dados da simulação requerida pelo ator. 2. API gera *link* de download dos arquivos e retorna ele para o front-end. 3. Front-end abre o *link* recebido em uma nova aba do navegador e o download inicia automaticamente.

Tabela 7 – Especificação para "Realizar Download das simulações"

3.2 Prototipação da Interface

A prototipação de interface é uma parte importante no processo de desenvolvimento de software, pois permite validarmos as telas do sistema antes elas sejam efetivamente implementadas (SOCIETY, 2014). Desse modo, podemos verificar com um usuário, se o visual de cada interface está de acordo com suas necessidades, evitando retrabalho e/ou modificações em código existente. Com isso em mente, foi desenvolvido um protótipo da interface do FFSWA utilizando o Figma - uma ferramenta para prototipação, design e construção de interfaces (GARRET, 2021).

Um protótipo foi apresentado aos membros do LCBio, com o intuito validar as duas principais páginas da aplicação: criação de simulação (Figura 8) e visualização/listagem de simulações (Figura 9). Poucas modificações foram solicitadas ao protótipo e estas foram implementadas na versão final da UI. No momento da prototipação, foi escolhida a biblioteca de componentes de UI que foi utilizada para o desenvolvimento do front-end da aplicação. Optou-se pela biblioteca open-source NextUI (NEXTUI, 2024), pois apresenta um *design* moderno, suporte a renderização do lado do servidor e do cliente, componentes com acessibilidade e outras vantagens. Vale ressaltar que a biblioteca só fornece componentes de interface mais simples como botões, *inputs* de formulário, barras de progresso, *cards* e etc. Componentes mais complexos foram implementados manualmente. Após a validação, as outras páginas do sistema adotaram um projeto semelhante.

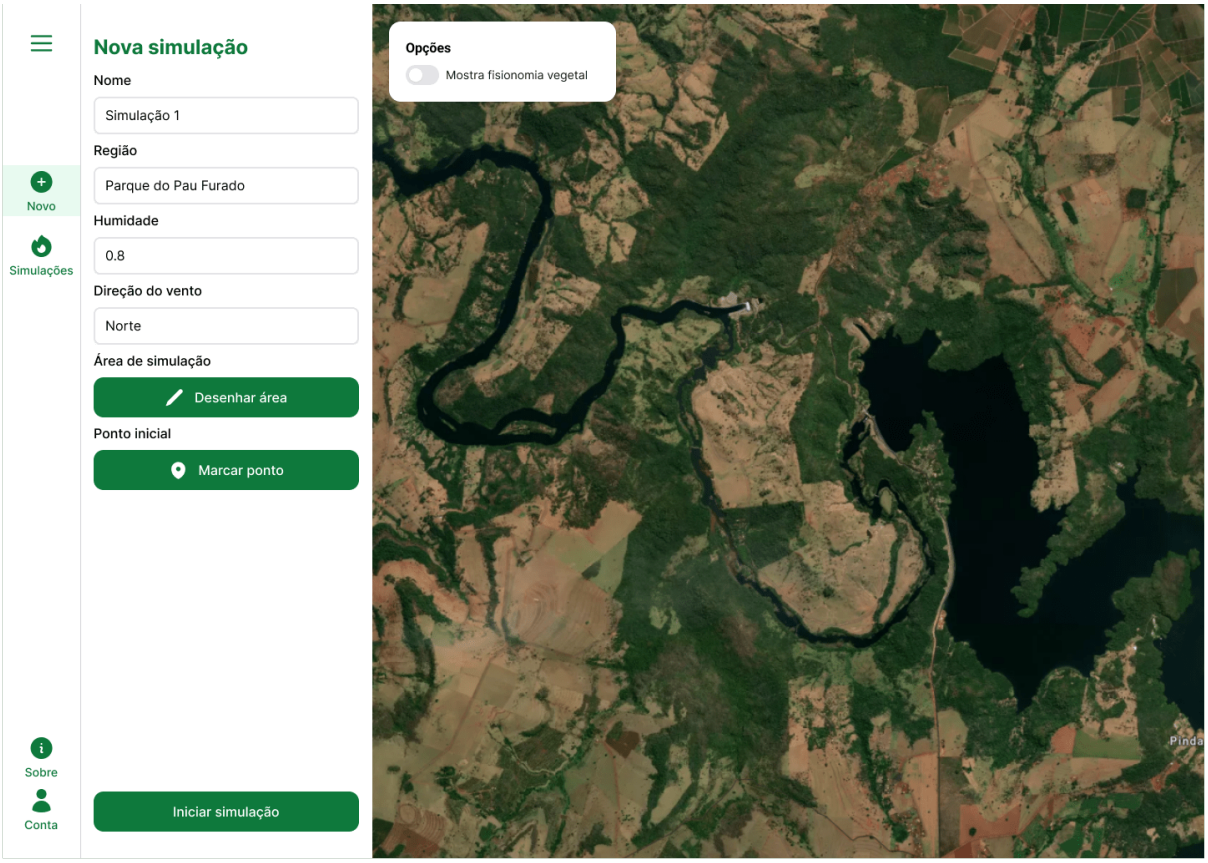


Figura 8 – Protótipo da página de criação de simulações

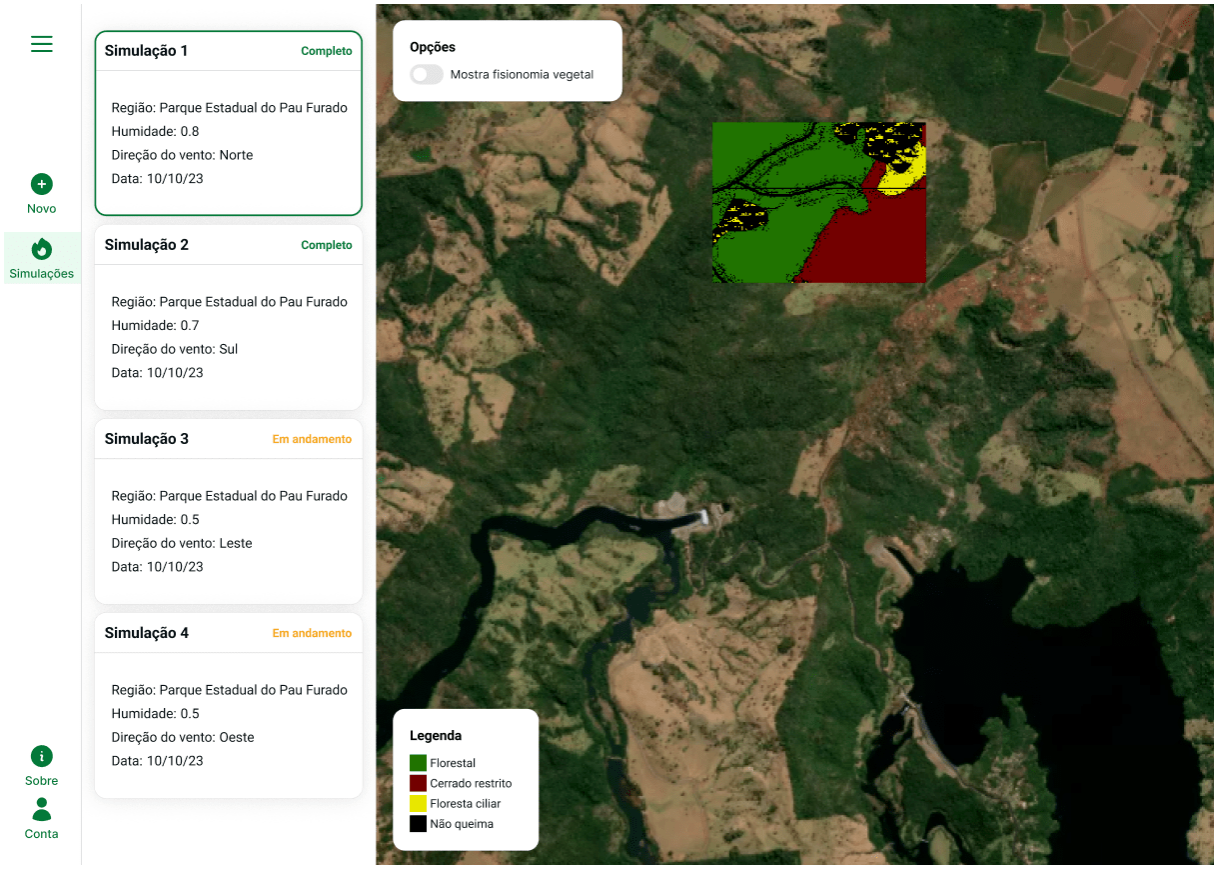


Figura 9 – Protótipo da página de listagem de simulações

A estrutura da interface foi organizada de forma a destacar o mapa, que ocupa a maior parte da tela, sendo o foco principal da aplicação. Este mapa é onde ocorre a visualização das simulações realizadas pelos usuários, refletindo o propósito central do sistema. Para maximizar a usabilidade, uma barra lateral de navegação foi posicionada ao lado esquerdo da tela, fornecendo acesso rápido às principais funcionalidades da aplicação. Essa barra lateral pode ser facilmente ocultada para proporcionar uma visualização mais ampla do mapa, garantindo flexibilidade às preferências dos usuários.

3.3 Arquitetura da Aplicação

O sistema desenvolvido utiliza uma arquitetura cliente-servidor. O cliente solicita serviços do servidor através de uma API RESTful - utilizando requisições HTTP. O desenvolvimento dessa arquitetura foi dividido em duas partes: front-end e back-end. A Figura 10 mostra como os diferentes componentes do sistema interagem entre si.

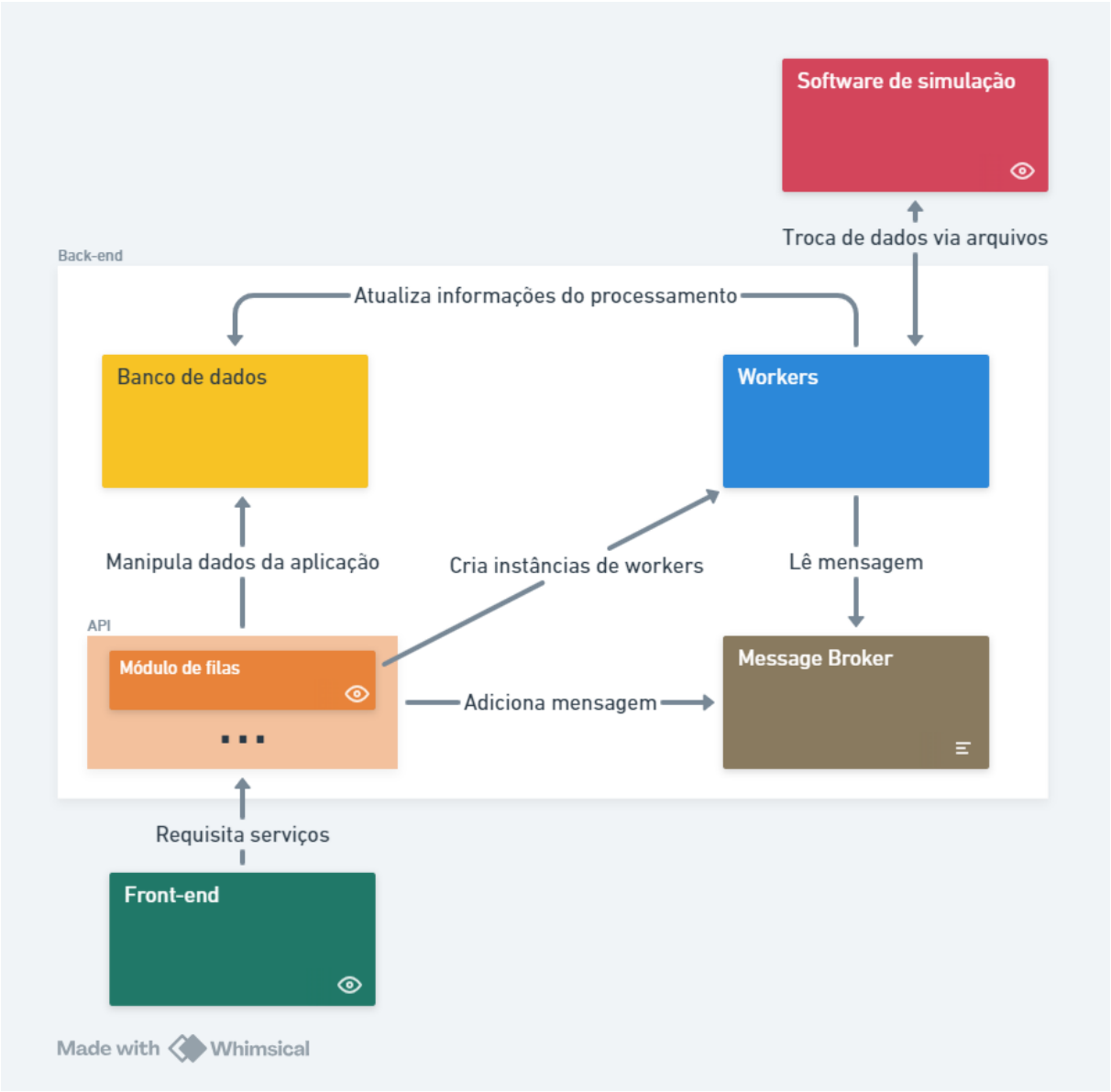


Figura 10 – Interação entre componentes do sistema

O front-end compreende a interface do sistema, pela qual o usuário irá interagir com a plataforma e requisitar simulações aos modelos cadastrados. Em nossa plataforma, o front-end foi implementado como uma aplicação de página única (SPA - *Single Page Application*).

O back-end engloba os componentes que estão do lado do servidor e que são responsáveis por tratar as requisições feitas pelo front-end e invocar o modelo de simulação, quando necessário. O back-end do sistema desenvolvido é constituído por uma interface para troca de informações seguras (API RESTful), um banco de dados, o agente de mensagens (do inglês, *message broker*) e as entidades instanciadas pela API para processar as mensagens do *broker* (*workers*).

Os simuladores são modelos computacionais baseados em autômatos celulares. Eles

são softwares devidamente cadastrados pelo administradores e, embora sejam componentes externos ao sistema, são parte essencial do fluxo de execução da plataforma.

Mais detalhes dos componentes do back-end e front-end são apresentados a seguir.

3.3.1 Back-End

O back-end, aplicação que executa do lado do servidor, é responsável por realizar todos os serviços solicitados pelo front-end. Isso inclui toda a criação, a modificação e o processamento de dados, além da execução das simulações de incêndio através dos simuladores disponíveis. As requisições do front-end são atendidas por uma API RESTful, ou seja, o front-end faz requisições HTTP para certos *endpoints* e, a depender do *endpoint* receptor e dos dados enviados, a API interpreta a requisição e executa os serviços relacionados. Os *endpoints* são Localizadores Uniforme de Recursos (URLs - *Uniform Resource Locators*), endereços utilizados para referenciar algum recurso na Web, que nesse contexto, faz referência para os serviços disponibilizados pela API.

A API foi desenvolvida utilizando TypeScript e o *framework* NestJS. A escolha do NestJS e do TypeScript para o desenvolvimento do back-end foi baseada em suas vantagens específicas, como a tipagem estática e a facilidade de manutenção proporcionada pelo TypeScript, e a robustez e facilidade de desenvolvimento oferecidas pelo NestJS. Essas tecnologias foram fundamentais para garantir a qualidade e a eficiência do back-end da aplicação. Além disso, a escolha pelo TypeScript também foi motivada pelo fato de ser a mesma linguagem utilizada no front-end da aplicação. Essa uniformidade linguística é especialmente vantajosa, pois trabalhar com uma única linguagem de programação facilita a compreensão do código, a reutilização de componentes e a manutenção do projeto como um todo.

A estrutura de módulos utilizada na organização da API é apresentada na Figura 11, sendo cada módulo responsável por realizar serviços relacionados a uma entidade ou funcionalidade do sistema. Por exemplo, o módulo autenticação é responsável por verificar se um usuário está autenticado e gerar *tokens* de acesso, enquanto o módulo de simulações permite o cadastro e obtenção de dados das simulações para o usuário autenticado.

Cada módulo segue um padrão arquitetural chamado *Controller-Service-Repository*. Nesse padrão, um módulo é dividido em camadas, sendo *Controller* a mais alta, *Service* a intermediária e *Repository* a mais baixa. Uma camada só pode depender de classes ou métodos de uma camada de mesmo nível ou inferior, conforme apresentado na Figura 12.

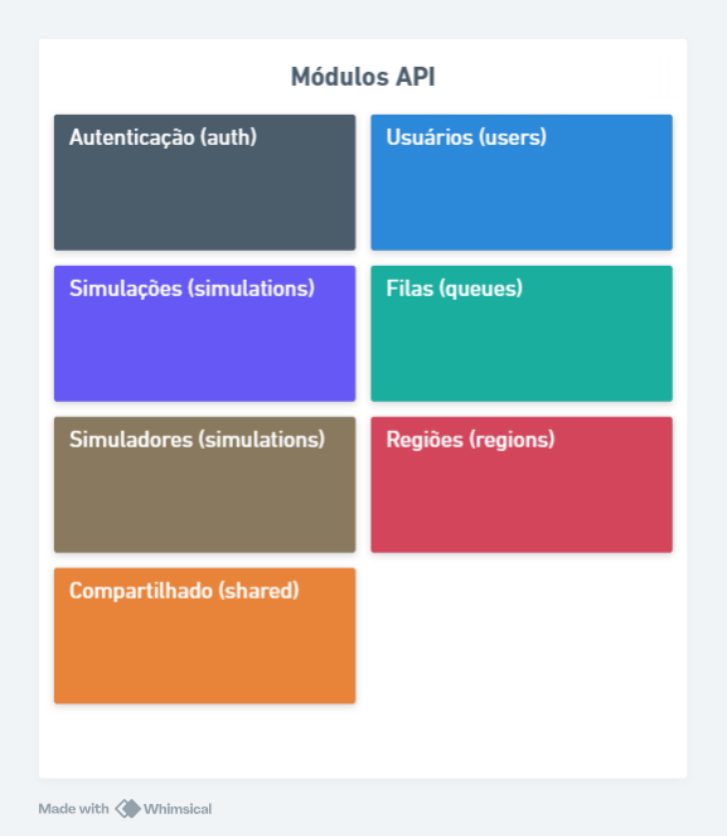


Figura 11 – Módulos da API

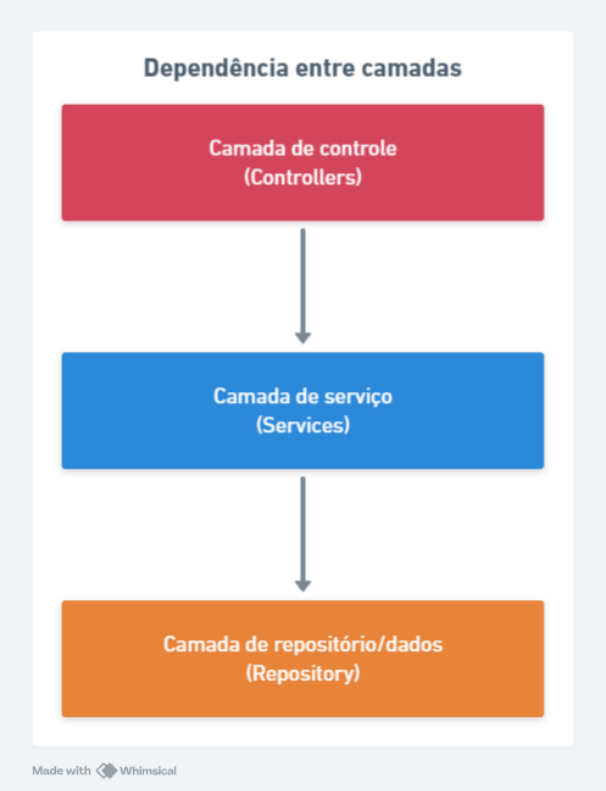


Figura 12 – Dependências entre camadas de um módulo

3.3.1.1 Camada de Controle

A camada de controle possui as classes denominadas *Controllers*, responsáveis por receber as requisições HTTP feita pelo front-end, tratá-las e invocar os serviços necessários. Um *Controller* pode chamar quantos serviços forem necessários e, ao final da execução deles, retorna uma resposta para o cliente (quem fez a requisição). Por exemplo, o *CreateSimulationController* pode receber uma requisição para criar uma nova simulação e, em seguida, chamar o serviço *CreateSimulationService* para realizar essa operação. Após a execução dos serviços necessários, o *CreateSimulationController* retorna uma resposta para o cliente, como uma resposta HTTP com os dados relevantes ou um código de status.

3.3.1.2 Camada de Serviço

A camada de serviços possui as classes denominadas *Services*. Cada *Service* representa um caso de uso do sistema, por exemplo, ter o caso de uso "Criar simulação" implica na existência da classe *CreateSimulationService*. Os objetos desta camada encapsulam a lógica de negócios da aplicação e podem utilizar métodos da camada de dados (*Repositories*) para acessar os dados necessários para realizar suas ações. Por exemplo, o *CreateSimulationService* pode utilizar métodos de um *Repository* para salvar os dados da nova simulação no banco de dados.

3.3.1.3 Camada de Dados

A camada de dados é abstraída pelas classes denominadas *Repositories*. Na aplicação desenvolvida, os *Repositories* utilizados pelos *Services* são fornecidos pelo por um *Object Relational Mapper* (ORM), chamado Prisma (DATA, 2024). Ele gera automaticamente os métodos de consulta aos dados com base em um arquivo de definição de modelos, chamado *schema*. Esses *Repositories* facilitam a interação com o banco de dados e fornecem uma camada de abstração que simplifica o acesso e a manipulação dos dados pela aplicação.

3.3.1.4 Filas de Processamento

Como foi possível observar na Figura 11, a API contém um módulo de filas (*queues*). Esse módulo foi implementado para que fosse possível executar processamentos *CPU Bound* (que utilizam o processador por muito tempo) sem atrapalhar o fluxo de execução da API em atender requisições do front-end. O módulo de filas é responsável por subir/instanciar os *workers* - partes do sistema responsáveis por processar mensagens da fila - em *threads* (fluxos de execução) separadas, possibilitando que eles sejam executados de forma independente. Para gerenciar as filas, inserindo e consumindo mensagens, foi utilizado a biblioteca BullMQ (TASKFORCE.SH, 2024). O BullMQ permite gerenciar

filas através de um sistema robusto e rápido, utilizando Node.js para execução e Redis, banco de dados do tipo chave-valor, como *message broker*. Dois *workers* são instanciados quando a aplicação é executada: *RegionRasterProcessor* e *ForestFireProcessor*, sendo *Processor* um termo utilizado pelo BullMQ para se referir às instâncias do tipo.

O *RegionRasterProcessor* realiza as seguintes operações:

1. Lê *raster* carregado pelo usuário e extrai somente os dados da imagem.
2. Cria um arquivo PNG redimensionado com resolução máxima de 512 *pixels* na altura ou na largura, utilizado somente na visualização da região no front-end.
3. Extrai informações do *raster* georreferenciado e as armazena no banco. Entre as informações armazenadas estão: coordenadas dos cantos da área representada pela imagem (*bounding box*), tamanho da área representada por cada pixel, largura e altuda da imagem.

O *ForestFireProcessor* realiza as seguintes operações:

1. Atualiza a situação da simulação de “PENDING” (“PENDENTE”) para “PROCESSING” (“PROCESSANDO”).
2. Obtém o caminho para o arquivo executável (JAR) do simulador selecionado.
3. Define o nome que será utilizado para os arquivos de entrada e saída da simulação.
4. Utiliza dados geográficos (região selecionada para simulação e pontos de foco inciais de incêndio demarcados), granularidade e *raster* da região de simulação selecionados pelo usuário para gerar o reticulado de entrada do autômato celular usado como modelo de simulação de incêndio..
5. Armazena o reticulado gerado e os dados da simulação, tais como: nível de umidade, direção do vento e índices do reticulado que serão pontos de foco iniciais, em um arquivo JSON, o quel será lido pelo simulador para processar a simulação do incêndio.
6. Executa o simulador em um processo filho, esperando pelo término de sua execução.
7. Obtém as cores cadastradas para os valores de saída (estados das células do reticulado) do simulador.
8. Carrega o arquivo de saída gerado pelo simulador e constrói uma imagem PNG para cada reticulado retornado, o qual representa o estado da simulação em um dado passo de tempo da simulação.

9. Gera uma imagem animada em formato GIF utilizando as imagens PNG geradas anteriormente. A imagem é gravada na memória secundária (disco rígido) do servidor.

10. Comprime os arquivos de entrada, saída e a imagem GIF em um arquivo ZIP.

11. Atualiza a situação da simulação de "PROCESSING" ("PROCESSANDO") para "FINISHED" ("FINALIZADO").

Caso qualquer exceção seja levantada durante a execução de um *Processor*, a tarefa (*job*) é tratada como falha. No caso do processamento de simulações, a situação da simulação é atualizada para "FAILED" ("FALHA").

3.3.1.5 Banco de Dados

Para o armazenamento dos dados da aplicação foi utilizado o PostgresSQL (também chamado de Postgres), um banco de dados relacional e *open-source*. O Postgres foi escolhido para este projeto pois tem suporte ao armazenamento e a consulta de objetos geográficos através da extensão PostGIS, o que será necessário para armazenar os dados das regiões disponíveis e demarcadas para simulação pelo usuário. A Figura 13 apresenta o Diagrama Entidade Relacionamento (DER) utilizado para modelagem das tabelas do banco de dados, assim como sua relações. O DER foi utilizado na escrita do *schema* utilizado pelo Prisma para geração dos métodos de consulta ao banco de dados.

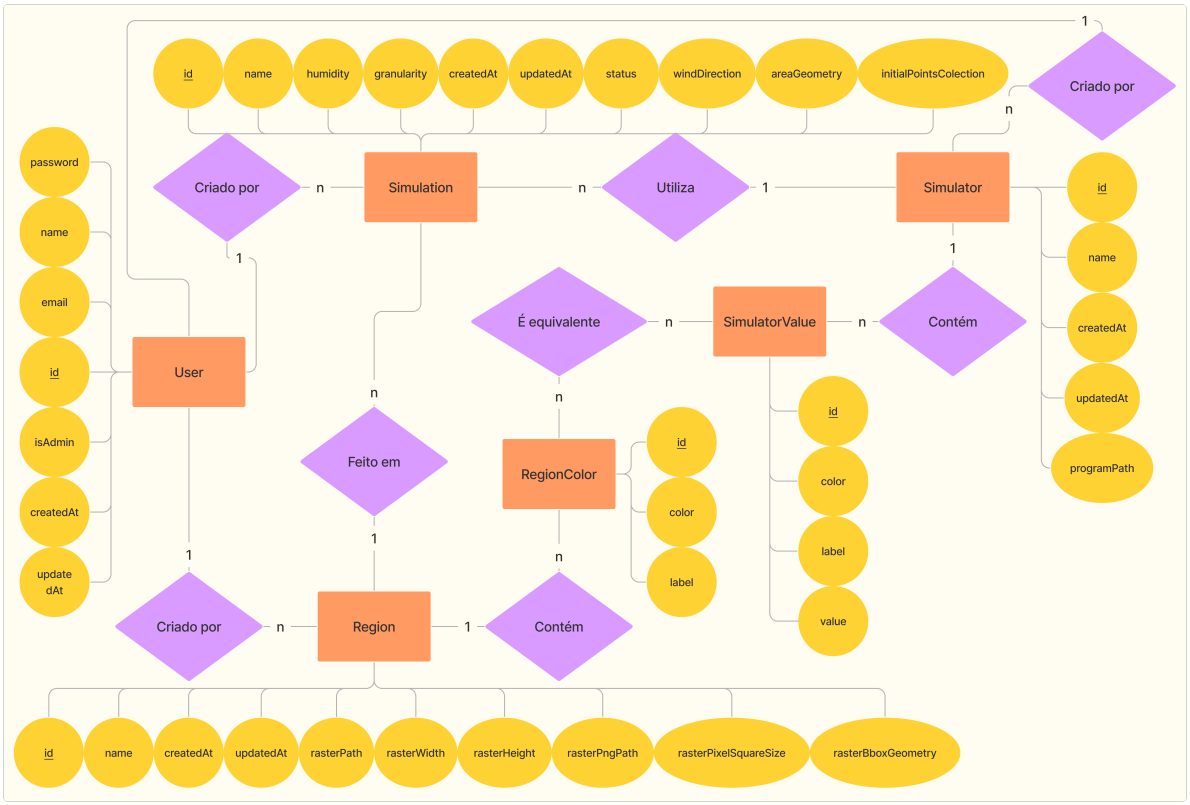


Figura 13 – Diagrama Entidade Relacionamento do sistema

A seguir, é apresentada uma lista das entidades do sistema e o que cada uma representa:

- ***User***: representa um usuário da aplicação, que pode ser do tipo administrador ou não. Todo usuário pode criar e visualizar simulações e, caso seja administrador, pode cadastrar regiões para simulações e simuladores.
- ***Simulation***: representa a simulação criada e visualizada pelo usuário. Contém dados da área selecionada pelo usuário e os pontos iniciais do incêndio, além de outros parâmetros utilizados na configuração da simulação.
- ***Region***: refere-se à região de um mapa que foi cadastrada por um administrador. Ela pode ser utilizada nas simulações de incêndios. Para isso, o usuário escolhe uma área dentro da região cadastrada para ser processada pelo simulador. A criação de uma *Region* envolve o *upload* de um *raster* georreferenciado referente ao mapa de vegetação da região desejada.
- ***RegionColor***: cor presente no *raster* da região cadastrada, podendo representar uma vegetação, tipo de terreno ou qualquer outro estado processado pelo simulador de incêndios. Uma *Region* só fica disponível para seleção na criação de simulações (*Simulation*) se tiver ao menos uma *RegionColor* relacionada com um *SimulatorValue*.

- ***Simulator***: representa um simulador de incêndio cadastrado por um usuário administrador. O cadastro de um *Simulator* envolve o *upload* de um arquivo JAR que é executado no back-end para realizar a simulação de incêndios.

- ***SimulatorValue***: valor de um estado válido para as células do reticulado de um simulador cadastrado. Um *Simulator* só aparece para seleção na criação de simulações (*Simulation*) se possuir ao menos um *SimulatorValue* associado com alguma *RegionColor* da região (*Region*) selecionada pelo usuário.

3.3.2 Front-End

Para desenvolvimento do front-end foi utilizado a linguagem TypeScript e o *framework* Next.js, que é construído em cima da biblioteca React - ambos sendo tecnologias *open-source* (VERCEL, 2024). O Next.js traz todas as vantagens do React - como pedaços reutilizáveis de código (chamados de componentes), alta reatividade em relação às ações do usuário e escrita de interfaces utilizando JSX (JavaScript XML, uma extensão do JavaScript que integra uma linguagem de marcação semelhante ao HTML). Além disso, permite realizar otimizações automáticas no carregamento das páginas, renderização de componentes do lado do servidor, *streaming* de UI (partes da interface são entregues ao cliente conforme a necessidade), entre outras vantagens.

3.3.2.1 Componentes

A construção da interface foi toda "componentizada", ou seja, dividida em componentes independentes e reutilizáveis. No contexto do React com Next.js, os componentes desempenham um papel crucial na criação de uma aplicação consistente e modular. Eles podem ser de dois tipos: funcionais e de classe. Os componentes funcionais são simples funções JavaScript que recebem um argumento denominado *props* como parâmetro e retornam elementos React. Já os componentes de classe são classes JavaScript que estendem a classe React.Component e possuem um método render() que retorna um elemento React.

O Next.js facilita a organização e a utilização desses componentes através do seu sistema de roteamento dinâmico, que permite a criação de páginas e componentes de forma hierárquica. Na implementação do front-end deste sistema, os componentes foram utilizados para representar desde elementos de interface simples, como botões e campos de formulário, importados da biblioteca NextUI, até componentes mais complexos, como listas de itens e formulários completos. Essa abordagem modular e reutilizável não apenas simplifica o desenvolvimento e a manutenção do código, mas também promove uma melhor organização e escalabilidade da aplicação.

3.3.2.2 Comunicação com API

Para a comunicação com a API do sistema, foi adotada a biblioteca React Query (TANSTACK, 2024). O React Query é uma ferramenta poderosa que simplifica o gerenciamento de dados na aplicação, fornecendo uma maneira intuitiva de realizar requisições assíncronas e manter o estado dos dados de forma eficiente. Uma das principais vantagens do React Query é o seu sistema de *cache* inteligente, que armazena automaticamente os resultados das requisições e os atualiza de acordo com as mudanças nos dados ou nas configurações de consulta. Isso reduz a quantidade de requisições redundantes e melhora significativamente o desempenho da aplicação.

Cada componente React da aplicação cuida de requisitar os dados que precisa para renderizar as informações necessários utilizando o React Query. Uma *key* (chave) identifica cada uma das consultas feitas à API pela biblioteca e quando mais de um componente utilizam a mesma *key*, somente uma consulta é feita e o dado é compartilhado entre todos os componentes que utilizam o mesmo identificador. Isso permite uma experiência de usuário mais fluida e responsiva, mesmo em casos de conexões de rede lentas ou instáveis. Além disso, elimina uma possível sobrecarga de requisições para a API e reduz a complexidade adicional que seria necessária ao implementar um sistema de compartilhamento de estado entre componentes para os dados buscados na API.

4 Resultados

Ao final do desenvolvimento, foi construída uma plataforma web que cumpre com o que atende aos requisitos funcionais e não funcionais definidos. Nas seções a seguir, serão apresentadas as diferente páginas da aplicação e o estudo de caso utilizando um *raster* que apresenta a fisionomia vegetal do Parque Estadual do Pau Furado (IEF, 2024) e o modelo de propagação de incêndios desenvolvido por Ferreira et al. (2023)

4.1 Páginas da Aplicação

Nesta seção, são apresentadas todas as páginas desenvolvidas para o FFSWA, bem como o seu comportamento perante as ações do usuário.

4.1.1 Página Inicial

A página inicial, apresentada na Figura 14, contém um texto simples explicando a motivação central do projeto: “apresentar uma interface amigável para simulação de incêndios utilizando os simuladores desenvolvidos pelo LCBio”. Há um botão abaixo do texto que pode ser clicado para acessar a plataforma. Caso o usuário não esteja autenticado, ele é redirecionado para a página de login (4.1.2). Caso contrário, é aberta a página de visualização de simulações (4.1.4).

Figura 14 – Página inicial

4.1.2 Cadastro e Login de Usuário

As páginas de login e cadastro estão diretamente ligadas com a autenticação no sistema. Ambas as páginas apresentam um formulário simples e *links* que permite ao usuário alternar entre elas. A tela de login (Figura 15), permite que um usuário previamente cadastrado se autentique na plataforma. Durante o processo de autenticação, o front-end requisita serviços fornecidos pelo módulo de autenticação da API que retorna um *token* de acesso que deve ser incluído no cabeçalho de cada requisição HTTP para o uso de serviços da API que são restritos a usuários autenticados. Quando um usuário termina o processo de autenticação ele é redirecionado para a página de cadastro de simulações (4.1.3).

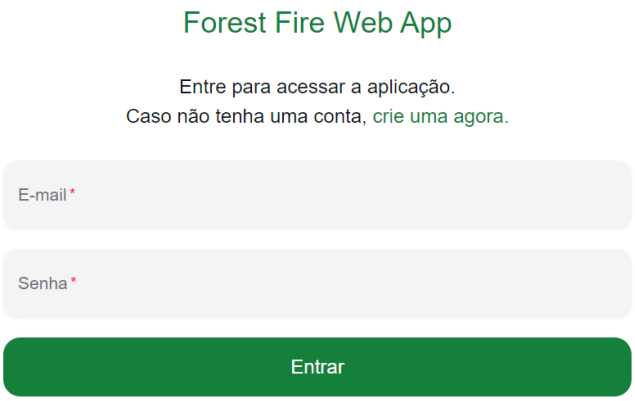


Figura 15 – Página de login de usuário

A página de cadastro (Figura 16) permite que um novo usuário possa se cadastrar no sistema, mediante o fornecimento de informações simples que serão utilizadas para sua identificação: nome, e-mail e senha. Quando um usuário termina o processo de cadastro, ele é automaticamente autenticado no sistema e redirecionado para a página de cadastro de simulações.

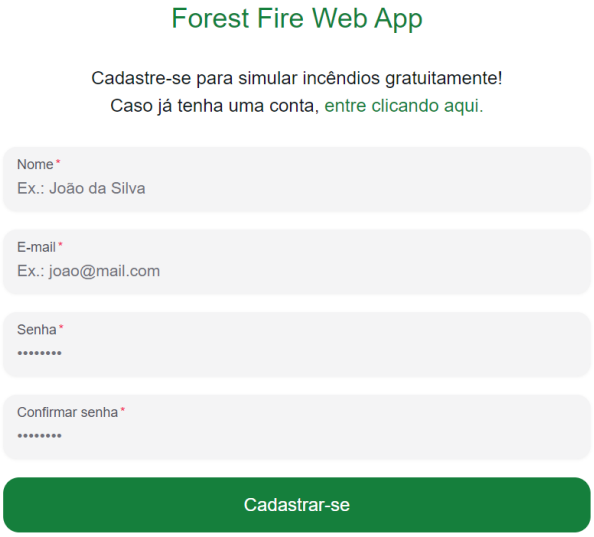


Figura 16 – Página de cadastro de usuário

4.1.3 Cadastro de Simulações

Na página de cadastro de simulações, acessada pelo botão "Novo" na barra de navegação (*navbar*) à esquerda, o usuário é apresentado a um mapa de imagens de satélite, onde acontece a visualização da região de interesse para simulação no formulário. Na esquerda, ao lado direito da *navbar*, é exibido uma barra lateral (*sidebar*), que contém o formulário onde o nome e os parâmetros da simulação (umidade, direção do vento, região de interesse, simulador a ser utilizado, granularidade do reticulado e focos iniciais de incêndio) podem ser fornecidos. Quando uma região é selecionada, através da lista de seleção com identificador "Região", sua área é apresentada no mapa como uma geometria de cor azul (Figura 17) e uma legenda é apresentada no canto da tela em conjunto com um botão que ativa/desativa a exibição da imagem da região. A legenda exibe as cores cadastradas para o *raster* e caso o botão de exibição seja clicado, o *raster* da área previamente selecionada é exibido, conforme ilustrado na Figura 18. Essa imagem é redimensionada pelo *RegionRasterProcessor*, pois é mais leve para ser enviado do servidor para o cliente.

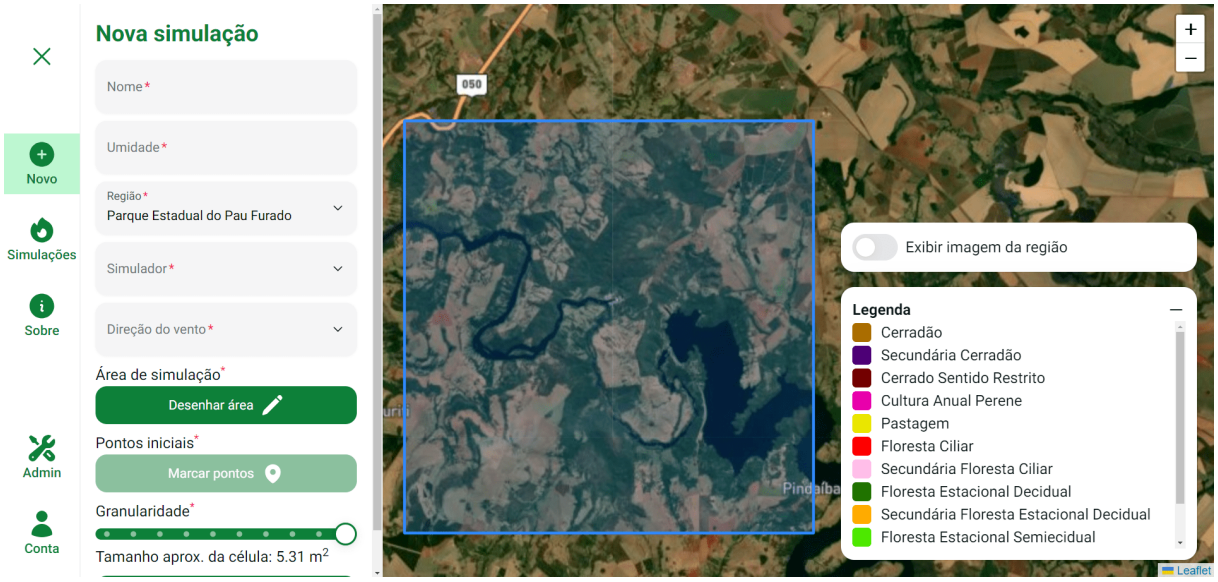


Figura 17 – Página de cadastro de simulações

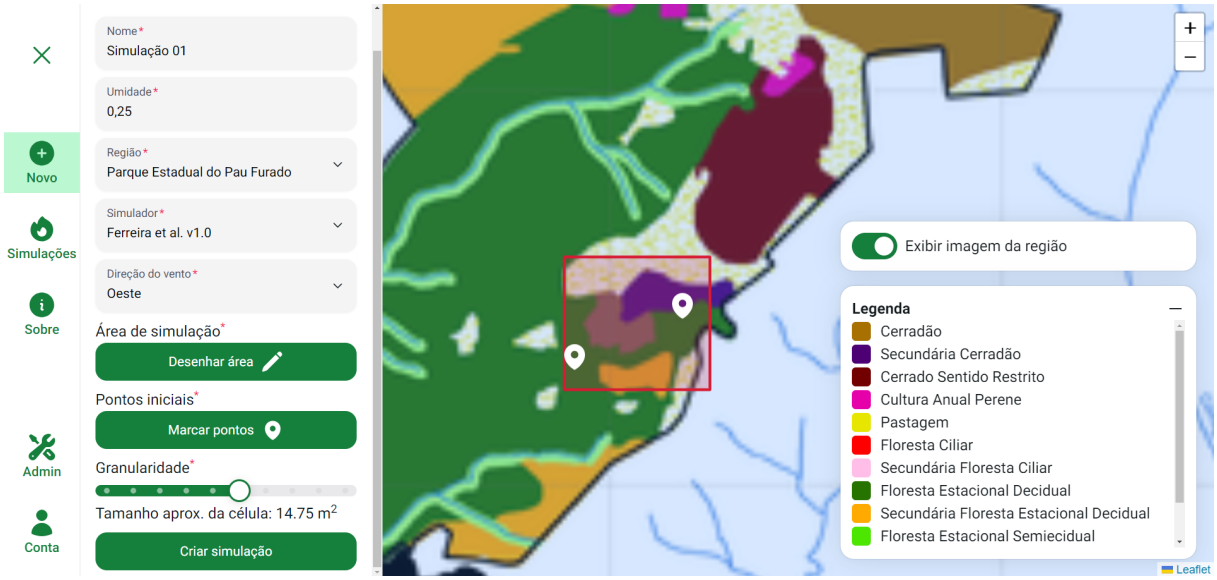


Figura 18 – Página de cadastro de simulações - Formulário preenchido

No formulário é apresentada uma barra que pode ser deslizada para alterar a granularidade e o usuário pode ver em tempo real a área em metros quadrados que cada célula irá representar. Uma granularidade menor resulta em um reticulado menor, onde cada uma das células representa uma área maior, diminuindo a precisão da simulação, assim como seu tempo. A barra completa representa granularidade 1 e, nesse caso, uma célula é criada por *pixel* da imagem (região) selecionada. Se o valor de granularidade é diferente de 1 (a barra foi arrastada para a esquerda), o *ForestFireProcessor* (apresentado na seção 3.3.1.4) redimensiona o *raster* que será utilizado para o processamento multiplicando a altura e largura pelo valor de granularidade que deve ser de 0,1 a 1, com variação de 0,1, resultando em 10 escalas.

Após preencher os dados no formulário, o usuário deve clicar no botão “Criar simulação”. Nesse momento, uma requisição é feita para a API, enviando os dados e solicitando a criação de uma nova simulação. A API, ao receber os dados e validá-los, cria uma nova tupla na tabela de simulações (entidade *Simulation*) e adiciona uma nova mensagem na fila de processamento correspondente (lida pelo *ForestFireProcessor*).

4.1.4 Visualização de Simulações

Possui estrutura semelhante à tela de cadastro de simulações (4.1.3), com uma *sidebar* na esquerda e o mapa para visualização de dados na direita. Essa página é acessada ao clicar na opção “Simulações” da *navbar* e apresenta uma lista de *cards* com os dados das simulações cadastradas pelo usuário (Figura 19). O usuário pode filtrar as simulações por nome ou *status* ao utilizar os *inputs* correspondentes. Ao clicar em uma simulação com o *status* "FINALIZADO", o front-end solicita o GIF da simulação e o apresenta sobre o mapa, sobrepondo região que ele representa (Figura 20). Quando a visualização da

simulação está em andamento, uma caixa de legenda aparece no canto inferior direito da tela, apresentando o significado de cada uma das cores apresentadas nas imagens.

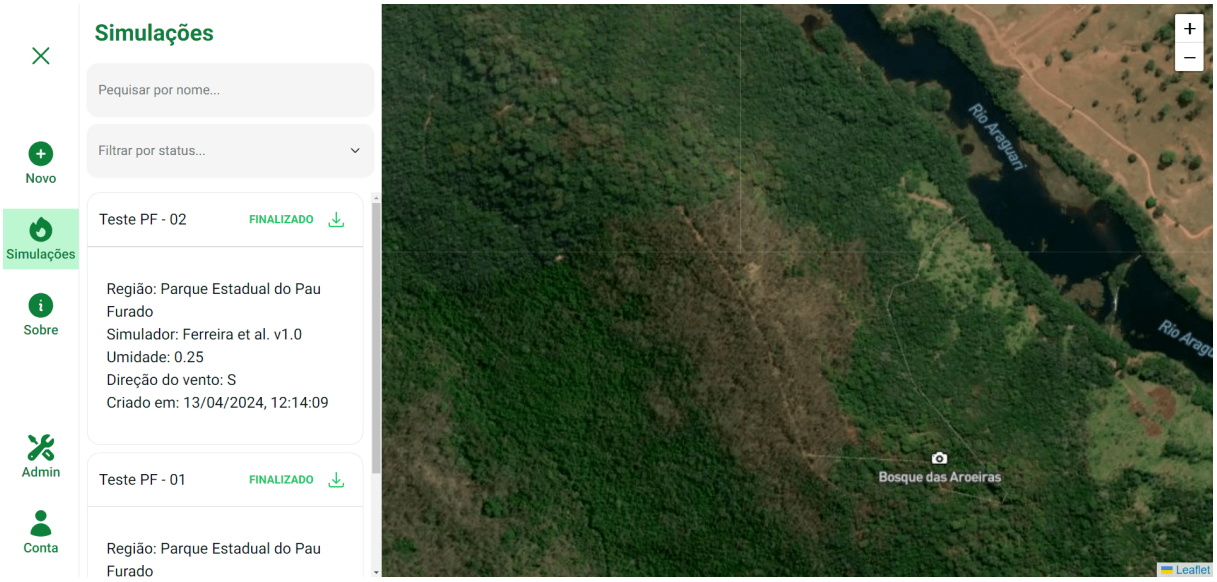


Figura 19 – Página de visualização de simulações

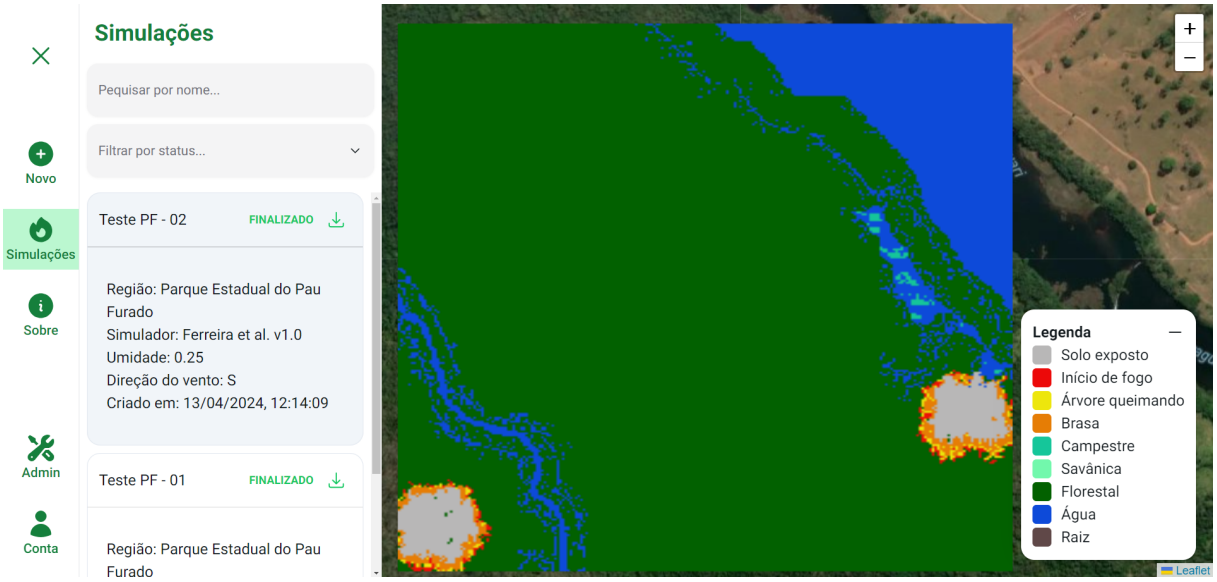


Figura 20 – Página de visualização de simulações - Simulação selecionada

4.1.5 Painel Administrativo

O painel administrativo só pode ser acessado por usuários administradores. Quando um usuário é autenticado na aplicação, a API retorna suas informações para o front-end, incluindo o tipo do usuário. Com essa informação, é possível determinar quais páginas podem ou não ser acessadas pelo usuário autenticado. Na página inicial do painel administrativo (Figura 21), é exibido um ícone de forma central na tela e um pequeno texto indicando que as funções administrativas podem ser acessadas através da *sidebar*. As funções administrativas da plataforma (representadas por *links* que são exibidos na *sidebar*

nessa página): “Cria simulador”, “Visualizar simuladores”, “Criar região” e “Visualizar regiões”.

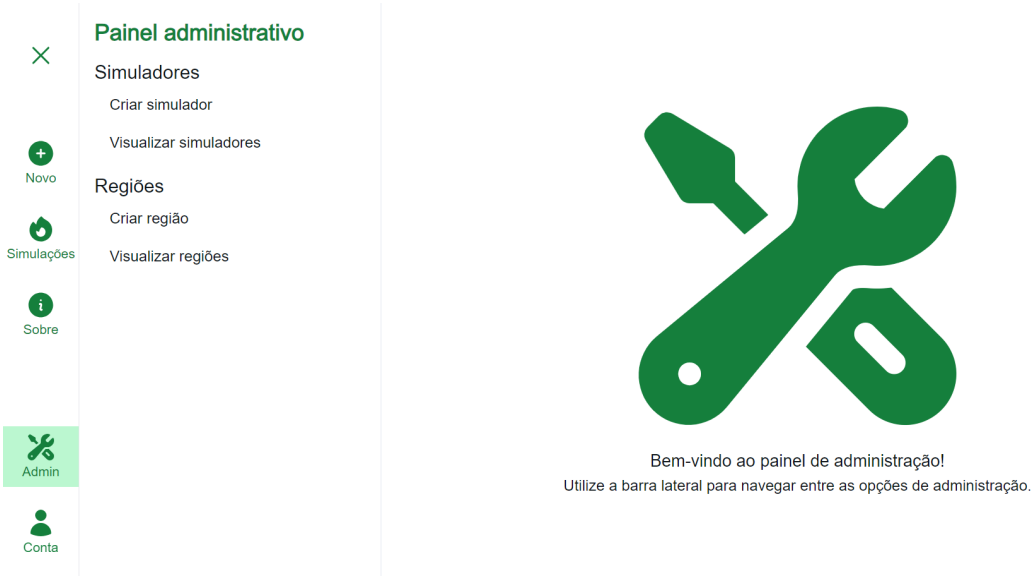


Figura 21 – Página do painel administrativo

4.1.5.1 Cadastro de Regiões

O cadastro de regiões, cujo a tela é exibida na Figura 22, permite usuários administradores adicionarem novas regiões que podem ser utilizadas nas simulações de incêndios pelos usuários da plataforma na página de cadastro de simulações (descrita na seção 4.1.3). A página em questão possui um formulário na parte central, contendo campos para inserir o nome da região, selecionar o arquivo que contém o *raster* georreferenciado (somente formato TIFF) da área desejada, o qual será carregado e armazenado na aplicação, bem como definir quais as cores presentes no *raster*.

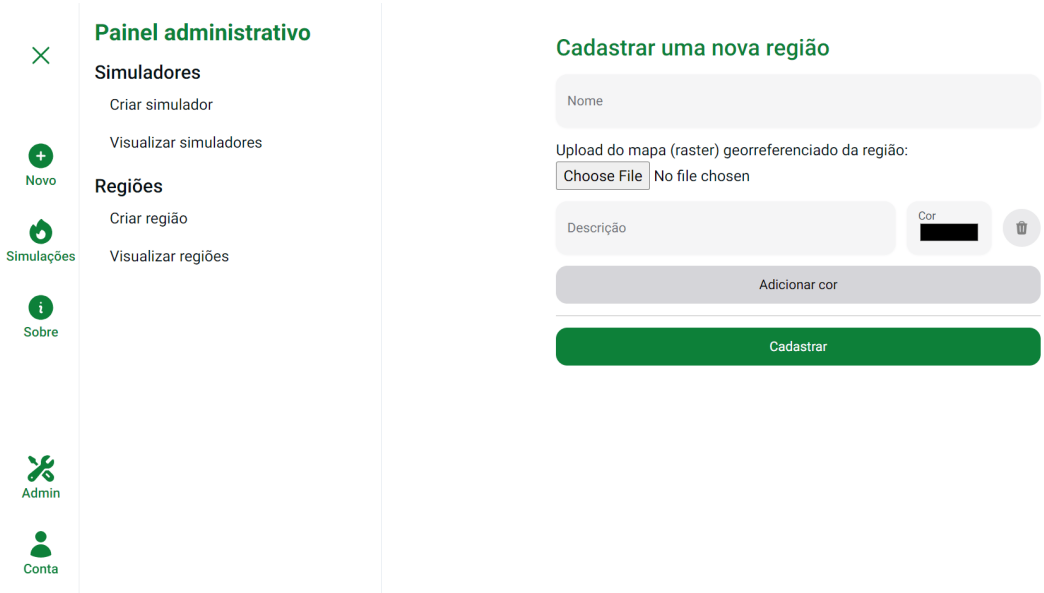


Figura 22 – Página de cadastro de regiões

Caso haja mais de uma cor presente na região a ser cadastrada, o administrador pode clicar no botão cinza com o texto "Adicionar cor", o que resultará em um *input* adicional sendo apresentado para o cadastro de mais uma cor. Qualquer quantidade de cores pode ser adicionada, mas vale ressaltar que cada uma delas deve ser associada a um valor interpretado pelo simulador na página de vincular regiões e simuladores (descrita na seção 4.1.5.5). Para excluir uma cor, o botão com ícone de lixeira pode ser utilizado. Uma descrição para a cor deve ser fornecida no campo com o identificador "Descrição" e a cor pode ser definida no campo a direita ("Cor"). Ao clicar no campo "Cor", uma caixa para a seleção da cor é exibida, permitindo definir a cor em um seletor ou por inserção de valores HEX, RGB ou HSL (Figura 23). Quando uma região é cadastrada, o serviço responsável por essa ação adiciona uma mensagem na fila para que o *RegionRasterProcessor* possa processar e extrair as informações presentes no *raster* carregado, conforme descrito na seção 3.3.1.4.

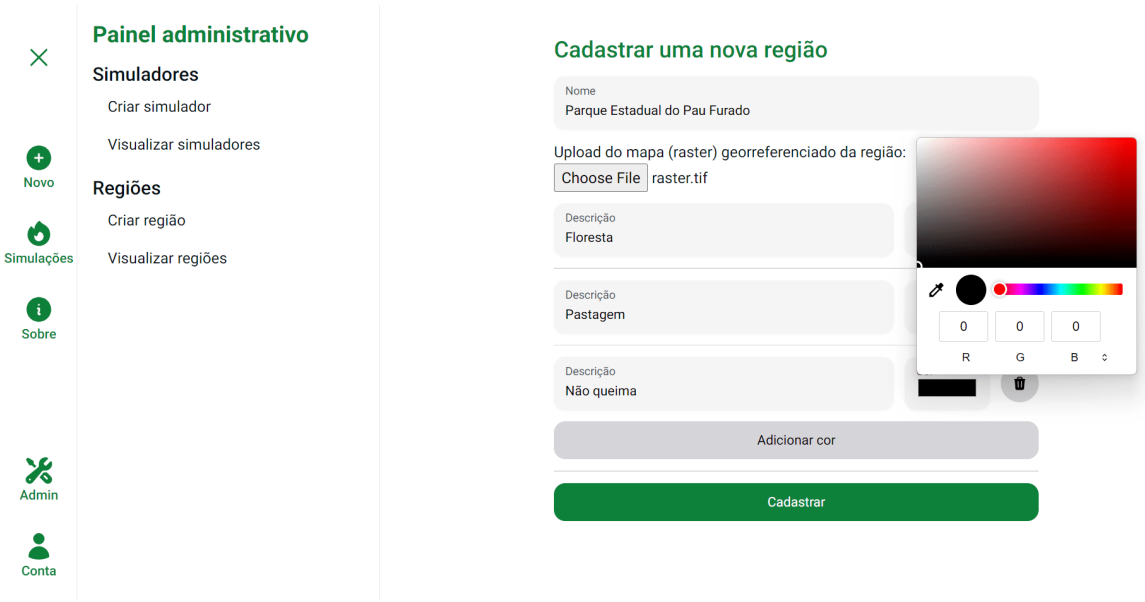


Figura 23 – Página de cadastro de regiões - Adição de cores

Após o preenchimento do formulário, o botão "Cadastrar" deve ser utilizado para que a criação da região seja feita no back-end. A validação dos dados é feita antes do envio e, caso algum dado esteja faltando ou incorreto, uma mensagem é exibida alertando sobre a necessidade de correções nos dados do formulário. Caso contrário, os dados são enviados e uma mensagem de confirmação é exibida.

4.1.5.2 Cadastro de Simuladores

O cadastro de simuladores funciona de maneira muito semelhante ao cadastro de regiões. Na parte central da página de cadastro (Figura 24), é apresentado um texto informando como o simulador deve se comportar para que funcione corretamente na plataforma. O simulador deve interpretar dois argumentos para linha de comando: "–input"

que especifica o caminho para o arquivo JSON contendo os dados de entrada da simulação e "–output" que especifica o caminho para o arquivo JSON que deverá ser gerado pelo simulador, contendo os dados da simulação realizada. Ambos o argumentos são seguidos por um texto (*string*) contendo o valor ao caminho referido.

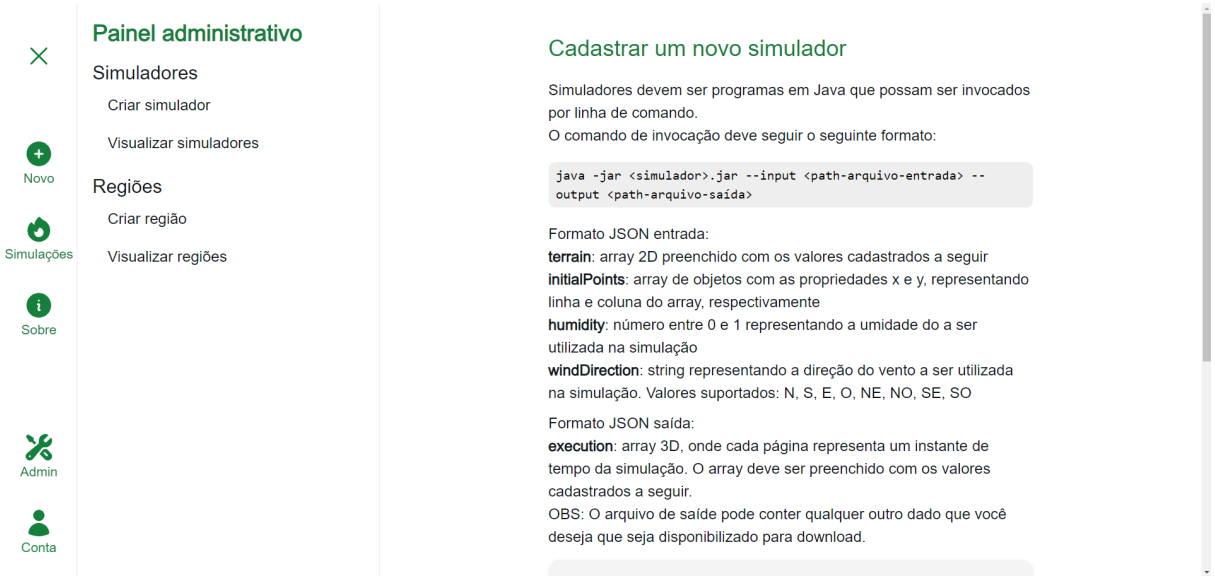


Figura 24 – Página de cadastro de simuladores - Texto explicativo

O arquivo JSON de entrada, criado pelo *ForestFireProcessor*, deve ser interpretado pelo simulador e tem a estrutura descrita a seguir:

- **terrain**: matriz 2D (reticulado) preenchido com os valores (estados de células) cadastrados para o simulador.
- **initialPoints**: vetor de objetos com as propriedades "x" e "y", representando linha e coluna do matriz, respectivamente. Cada par "x" e "y" especifica uma posição do reticulado que deve ser tratado como um foco inicial de incêndio.
- **humidity**: número entre 0 e 1 representando a umidade do ar ser utilizada na simulação.
- **windDirection**: *string* representando a direção do vento a ser utilizada na simulação. Os valores suportados são N, S, E, O, NE, NO, SE, SO.

O arquivo JSON de saída, gerado pelo simulador, deve conter obrigatoriamente o campo "execution", contendo uma matriz 3D, preenchido com os valores cadastrados para o simulador, onde cada página representa um instante de tempo da simulação. O arquivo também pode conter quaisquer outros valores que o desenvolvedor do simulador em questão queira que fique disponível para *download* na plataforma, entretanto, eles não serão interpretados internamente pelo sistema.

Após o texto explicativo (Figura 25), é apresentado um formulário com campos para inserir um nome para o modelo de propagação de incêndios que está sendo cadastrado, selecionar o arquivo executável do simulador a ser carregado e informar os valores (estados) reconhecidos pelo autômato celular do modelo, incluindo suas descrições e cores que serão utilizadas para apresentar a imagem de visualização do resultado das simulações realizadas. O botão "Adicionar valor" pode ser usado para adicionar mais estados do AC, informando seus respectivos valores, descrições e cores. O valor da cor é adicionado do mesmo modo que na página "Cadastro de Regiões" (4.1.5.1). Lembrando que, posteriormente na página "Vincular Regiões e Simuladores" (4.1.5.5), cada estado/valor deve ser associado a uma cor presente na imagem georreferenciada da região usada na simulação. Ao final do formulário há um campo para que um valor padrão seja selecionado. Durante a geração do reticulado inicial (entrada) da simulação, esse valor padrão é associado a todas as cores da imagem georreferenciada (*raster*) que não foram mapeadas para algum estado do modelo de simulação. Dessa forma, evita-se o usuário tenha que mapear todas as cores em uma imagem e que haja problemas no mapeamento de cores entre o mapa e o reticulado gerado, dado que podem haver pequenas variações em certos tons.

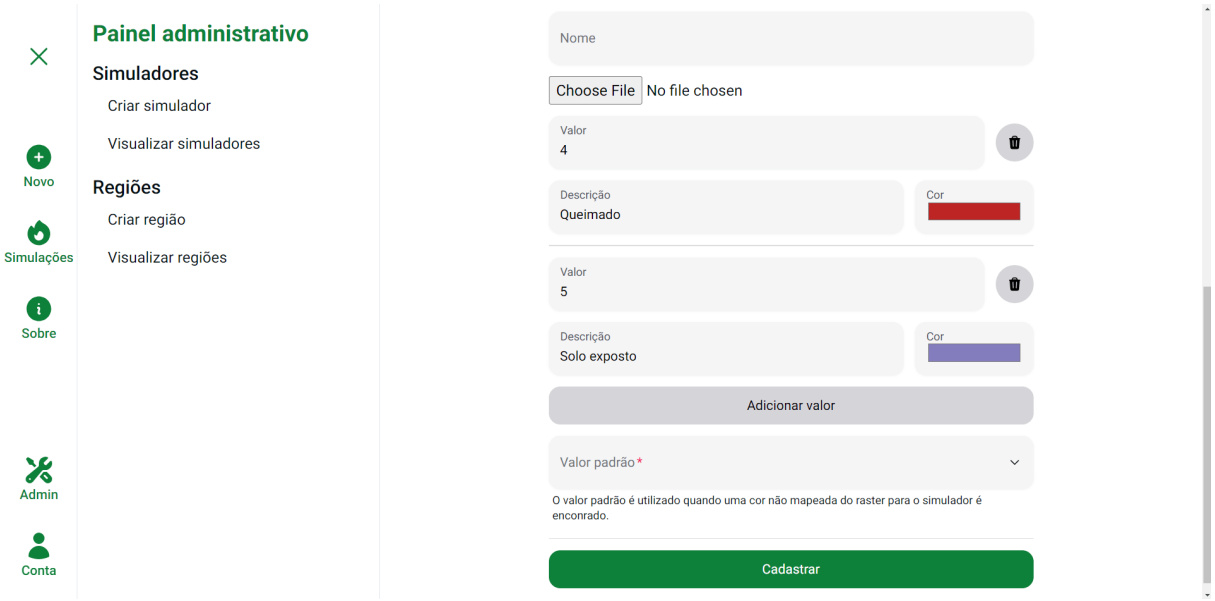


Figura 25 – Página de cadastro de simuladores - Formulário

Após o preenchimento do formulário, o botão "Cadastrar" deve ser utilizado para que os dados sejam validados e enviados para o back-end, caso estejam corretos. Se houver alguma inconsistência nos dados fornecidos, o sistema exibe uma mensagem indicando que os valores do formulário devem ser checados.

4.1.5.3 Visualização de Regiões

A página mostrada na Figura 26, exibe uma tabela com todas as regiões cadastradas na plataforma, incluindo aquelas que ainda não tiveram o *raster* devidamente

processado pelo *RegionRasterProcessor* e, portanto, ainda não podem ser utilizadas na criação de simulações.Nessa tabela, são apresentadas informações de cada região cadastrada, tais como: nome, data de criação, data de atualização e status de processamento do *raster* pelo *RegionRasterProcessor*. Caso o raster da região já tenha sido processado pelo sistema, a coluna "Raster" exibe um ícone verde. Caso contrário, uma ícone vermelha é exibida. Nessa janela também é possível pesquisar as regiões pelo nome na barra de pesquisa acima da tabela.

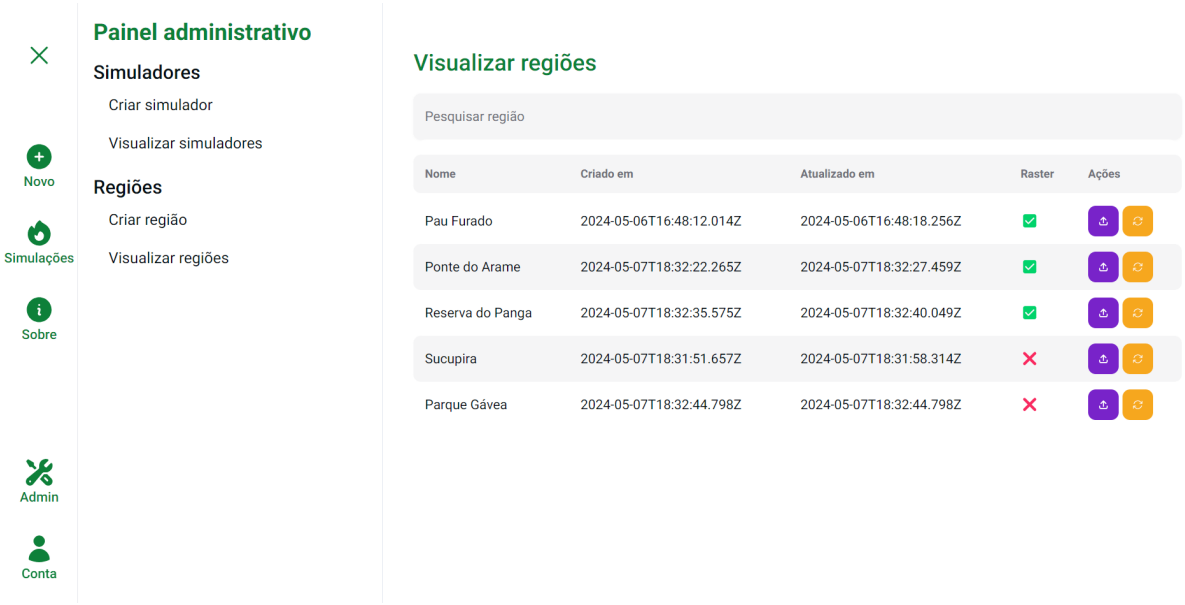


Figura 26 – Página de visualização de regiões

Para cada região apresentada na tabela, ainda é possível substituir sua imagem georreferenciada e associá-la à simuladores. A primeira ação é realizada ao clicar no botão roxo, que contém um ícone de uma seta apontada para cima. Esse botão abre uma janela suspensa (*modal*), como ilustrado na Figura 27, que permite o *upload* de um novo *raster* para a região cadastrada, o que irá desencadear um novo processamento pelo *RegionRasterProcessor*. A segunda ação é acionada pelo botão laranja, que contém um ícone de duas setas circulares, redirecionando o usuário para a página de vinculação de regiões e simuladores (descrita na seção 4.1.5.5).

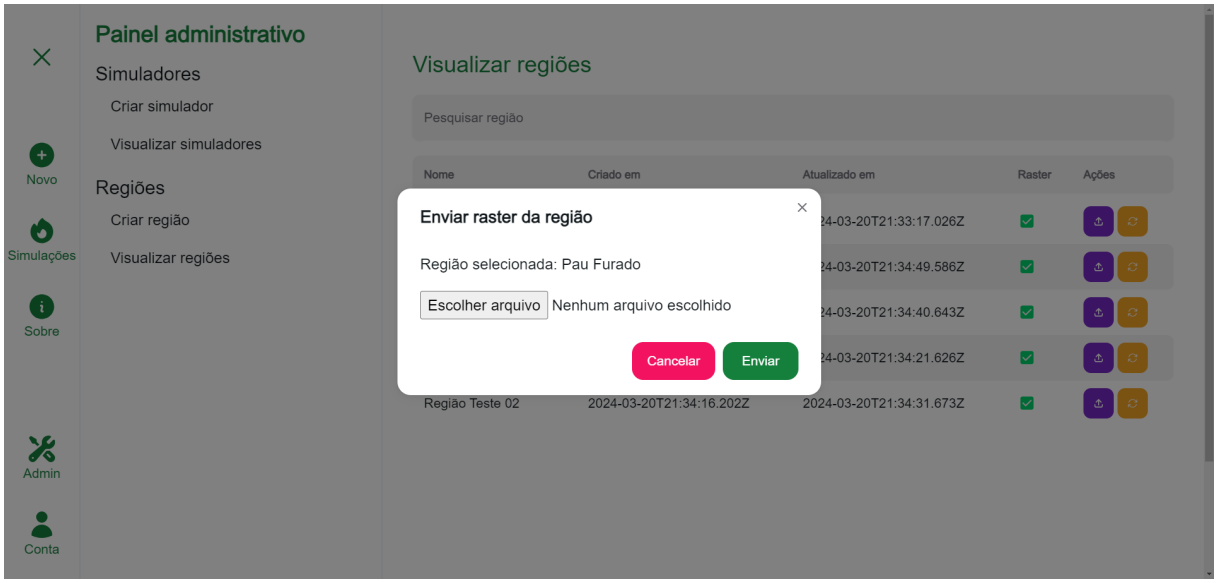


Figura 27 – Janela de *upload* da imagem para a região selecionada

4.1.5.4 Visualização de Simuladores

A página de visualização dos simuladores disponíveis na plataforma é mostrada na Figura 28 e apresenta um comportamento muito semelhante ao da página de visualização de regiões cadastradas. Aqui uma tabela exibe as informações de nome, datas de criação e de atualização de cada simulador cadastrado no sistema. Também é possível pesquisar os simuladores pelo nome na barra de pesquisa acima da tabela.

Figura 28 – Página de visualização de simuladores

4.1.5.5 Vincular Regiões e Simuladores

Quando uma região ou um modelo de simulação são cadastrados no sistema, eles ainda não podem ser utilizados para simular incêndios. Isso acontece porque é necessário

mapear as cores do mapa de vegetação de um região, que podem ter significados diversos e são interpretados a partir de sua leitura das cores no *raster* da região, para os estados (valores) definidos para um simulador. Desse modo, essa página, responsável por realizar esse mapeamento, a qual é ilustrada na Figura 28, apresenta um formulário dinâmico que exibe todos as cores de uma região, com suas respectivas descrições (dados à esquerda do formulário), e com um campo de formulário para sua associação à algum estado do modelo de simulação. Essa página é acessada a partir da página de visualização de regiões cadastradas, conforme explicado na seção 4.1.5.3.

Figura 29 – Página para vincular regiões e simuladores

O processo de vinculação inicia-se com a seleção do simulador desejado no primeiro campo do formulário. Em seguida, o mapeamento entre as cores do mapa e seus respectivos valores no modelo é feito através do campo com o texto “Selecione um valor”. Ao clicar nesse campo, uma lista suspensa é exibida, contendo todos os valores suportados pelo simulador selecionado (Figura 30).

Figura 30 – Menu de seleção dos estados das células no modelo de simulação

Ao final da página (Figura 31), há o botão “Enviar”. Ele deve ser usado assim que o formulário for preenchido, para que os dados sejam enviados ao back-end e a relação entre as respectivas cores da região e os valores do simulador seja criada no banco de dados. O sistema valida os dados antes do envio e, caso haja algum erro, uma mensagem é apresentada ao usuário para que as devidas correções sejam feitas.

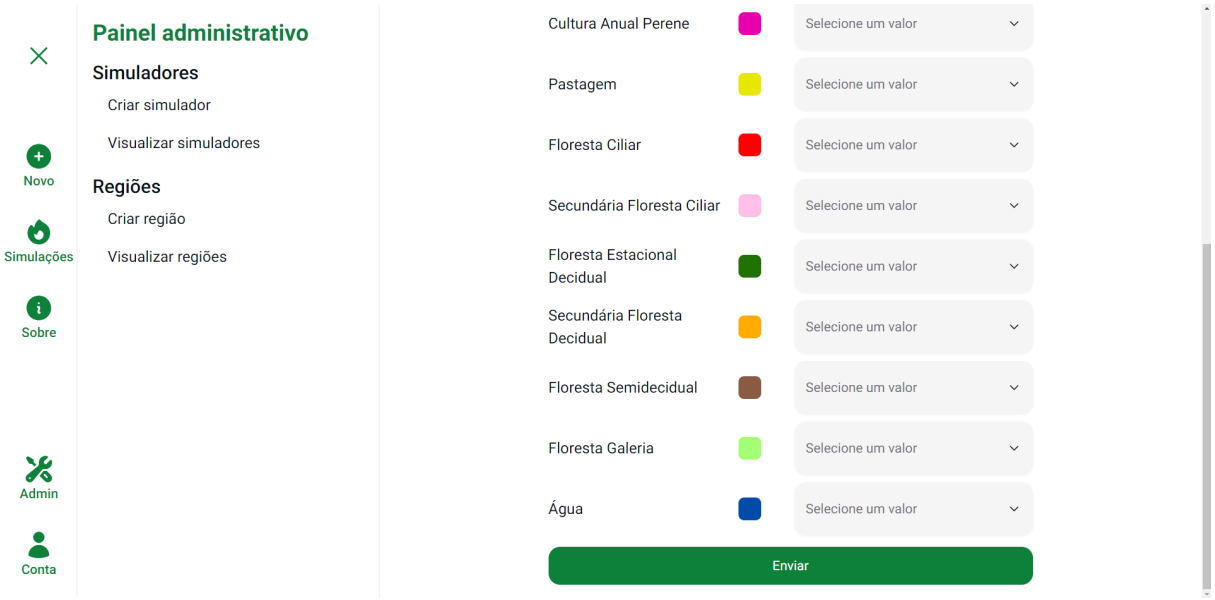


Figura 31 – Página para vincular regiões e simuladores - Botão de envio

4.2 Estudo de Caso

Após o desenvolvimento completo do sistema, foi realizado um estudo de caso para validar seu funcionamento e verificar se ele atende aos requisitos definidos no inicio

do desenvolvimento e aos objetivos propostos inicialmente. A seguir, são apresentados os resultados observados neste estudo de caso. Os testes realizados nesta etapa do trabalho visam:

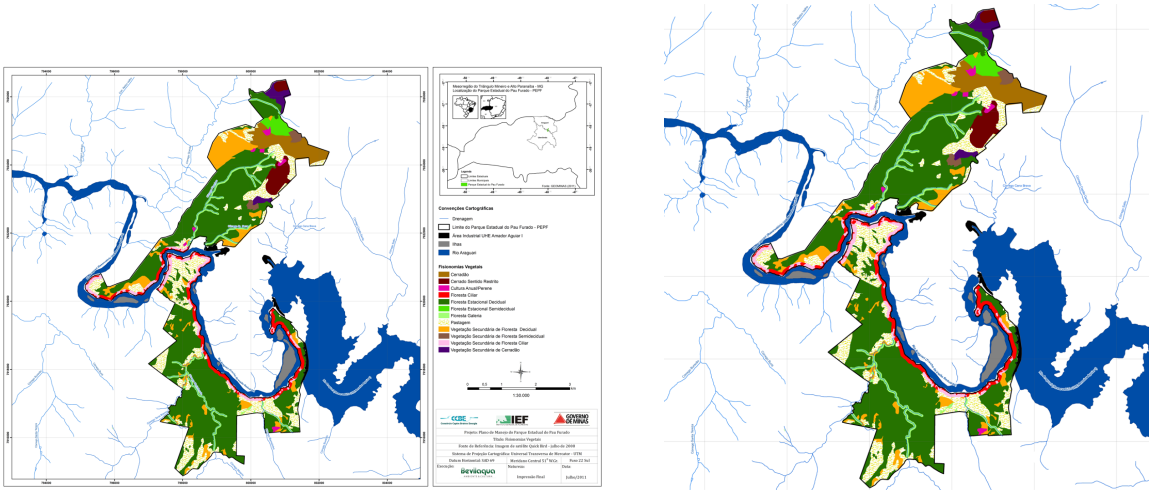
- Validar a criação de reticulados para um modelo de propagação de incêndio baseado em autômatos celulares, a partir do mapa de plano de manejo do Parque Estadual do Pau Furado.
- Validar a integração da plataforma com modelos de simulação externos, realizando e visualizando simulações de incêndio sobre os reticulados criados, utilizando o simulador desenvolvido por Ferreira et al. (2023).
- Avaliar o tempo de carregamento das páginas para garantir que o sistema responda rapidamente às ações do usuário.

Os testes foram realizados com o front-end e back-end da plataforma hospedados em uma mesma máquina local com o bancos de dados Postgres e o Redis, ambos utilizando Docker. O equipamento utilizado neste estudo de caso tem a seguinte configuração:

- **Processador**: 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz
- **RAM**: 16GB DDR4 2667MHz
- **Armazenamento**: SSD de 240GB
- **Sistema Operacional**: Ubuntu 22.04 no WSL (*Windows Subsystem for Linux*) executado no Windows 11

4.2.1 Criação e Visualização de Simulações

O mapa do Parque Estadual do Pau Furado utilizado a geração dos reticulados de autômato celular utilizados nas simulações de incêndio é apresentado na Figura 32. Ele contém regiões coloridas que representam as diferentes fisionomias vegetais do local. Como o simulador possui um modelo que analisa a propagação de incêndio utilizando a cobertura vegetal da área analisada, esse mapa se torna um ótimo exemplo para estudo. Além disso, o Parque Estadual do Pau Furado é uma área protegida do Cerrado brasileiro e verificar a efetividade do sistema em regiões desse bioma, se alinha com o projeto Cerrado Resiliente, no qual este trabalho está inserido. O mapa foi editado para que não apresentasse os nomes e descrições que aparecem sobre as regiões de vegetação e na lateral, pois são informações irrelevantes para a simulação e poderiam causar a presença de valores incorretos no reticulado gerado.



(a) Mapa original (b) Mapa editado

Figura 32 – *Raster* do mapa de fisionomia vegetal do Parque Estadual do Pau Furado

Ao acessar o sistema como usuário administrador, é possível acessar a página de cadastro de regiões, cadastrar uma nova região chamada "Parque Estadual do Pau Furado", utilizando o mapa editado, apresentado na Figura 32b, para identificar e cadastrar, no formulário da região, as cores presentes na imagem georreferenciada a ser carregada no sistema. A Figura 33 apresenta a página de cadastro da região usada neste estudo de caso, devidamente preenchida.

Figura 33 – Cadastro da região "Parque Estadual do Pau Furado"

O modelo de propagação de incêndio desenvolvido por Ferreira et al. (2023) também foi cadastrado na plataforma com o nome de "Ferreira et al. v1.0", conforme apresenta na Figura 34. Para realizar o cadastro, os estados suportados pelo modelo foram inseridos no formulário, sendo eles: "solo exposto", "inicio fogo", "arvore queimando", "brasa",

"campestre", "savanica", "florestal", "agua" e "raiz". Uma cor arbitrária foi escolhida para cada um dos valores cadastrados.

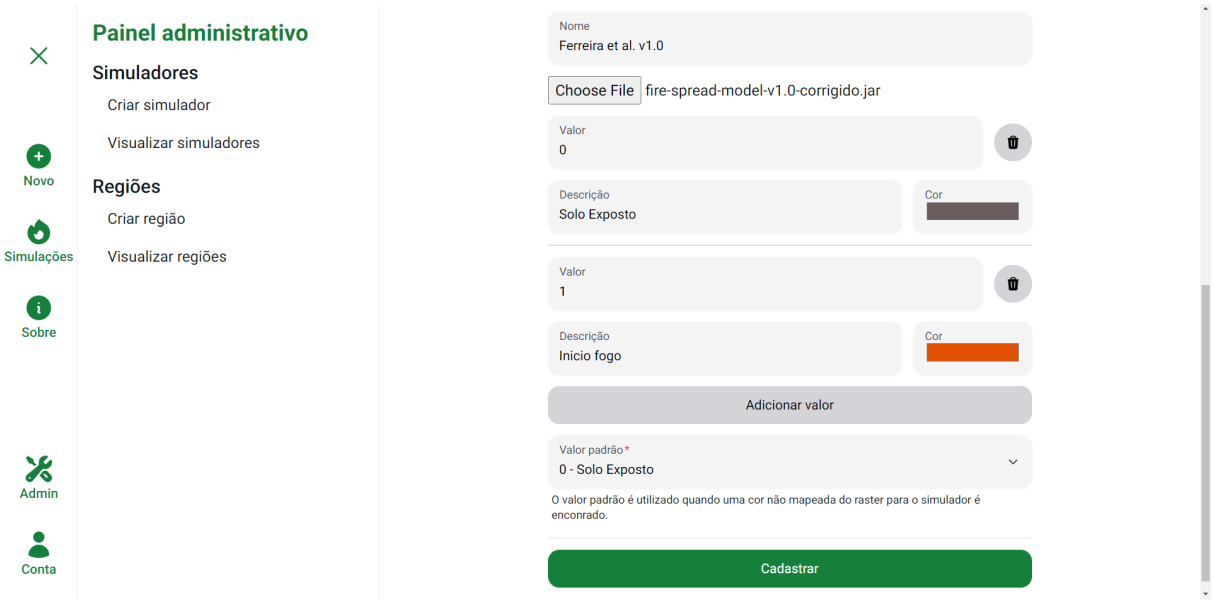


Figura 34 – Cadastro do simulador "Ferreira et al. v1.0"

Após o cadastro do simulador e do mapa especificados, a página de vincular regiões e simuladores foi acessada para associar a região "Parque Estadual do Pau Furado" ao simulador "Ferreira et al. v1.0". O modelo de simulação utilizado prevê apenas três estados relacionados a vegetação ("campestre", "savanica" e "florestal"), enquanto o mapa de cobertura vegetal da região em questão especifica diversas vegetações, um mesmo valor do modelo foi atribuído a diferentes tipos de vegetação do mapa. A Figura 35 apresenta o processo de associação realizado, com bordas vermelhas explicitando agrupamentos de vegetação realizados.

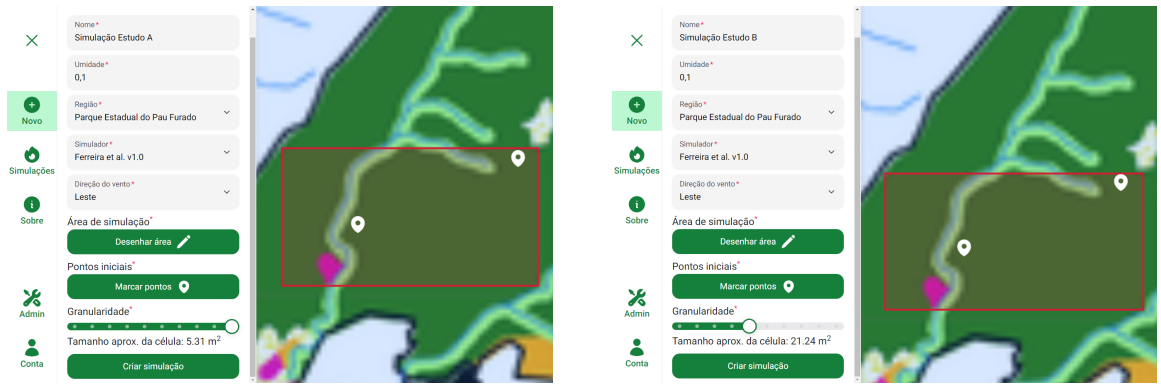
Figura 35 – Associação da região e simulador para o estudo de caso

Com a região e o simulador de interesse cadastrados e relacionados, ambos se tornaram disponíveis para o cadastro de simulações. Diversos testes foram realizados, mas aqui serão apresentadas duas simulações (A e B) para uma mesma região, mas com granularidades diferentes, resultando em reticulados com células de 5,31 metros quadrados (granularidade 1) e 21,24 metros quadrados (granularidade 0,5), respectivamente. Essas simulações foram escolhidas com o intuito de mensurar a redução no tempo de processamento ao utilizar uma granularidade menor e quão correto está a exibição do resultado gerado pela simulação, assim como verificar as diferenças nas propagações dos incêndios entre as simulações. As imagens a seguir apresentam os formulários preenchidos para a criação das duas simulações.

Os parâmetros utilizados na configuração dessas simulações são apresentados na Tabela 8.

Parâmetros\Simulação	**A**	**B**
Umidade	0,1	0,1
Direção do vento	Leste	Leste
Granularidade	1	0,5
Tamanho das células	5,21m^2	21,24m^2
Quantidade de focos de incêndio	2	2

Tabela 8 – Parâmetros de configuração das simulações avaliadas



(a) Simulação A com granularidade 1 (b) Simulação B com granularidade 0,5

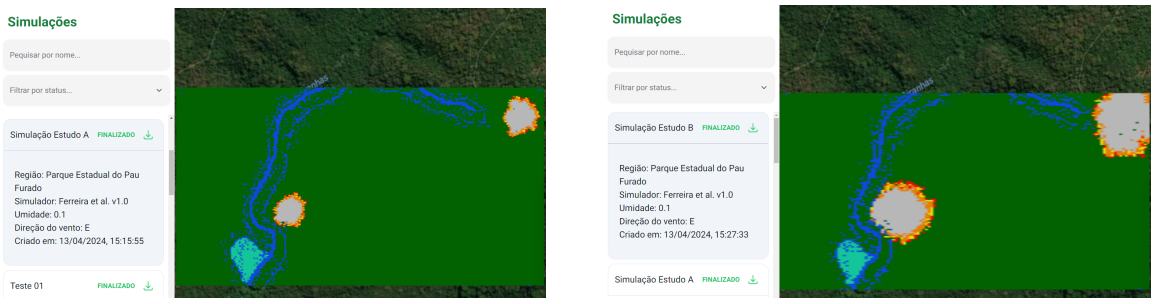
Figura 36 – Simulações cadastradas para o estudo de caso

Utilizando a ferramenta do back- end para visualização dos *jobs* nas filas da aplicação, foi possível visualizar o tempo gasto pelo *ForestFireProcessor* com cada uma das simulações (Figura 37). A simulação A levou 1 minuto e 2 segundos, enquanto a simulação B precisou de apenas 16 segundos para ser finalizada.

(a) Tempo para simulação A (b) Tempo para simulação B

Figura 37 – Tempo de execução do *ForestFireProcessor* para simulações A e B

Também foi possível notar diferença entre as simulações por meio das respectivas páginas de visualização, apresentadas na Figura 38. Como pode ser observado, a simulação B apresenta uma imagem de resolução menor (notável pelo aspecto embaçado) e um incêndio que se espalha por uma região maior, causado pelo fato das células do reticulado representarem uma área maior em relação à simulação A.



(a) Simulação A (b) Simulação B

Figura 38 – Visualização do resultado das simulações A e B

A possibilidade do *download* dos dados brutos de ambas as simulações também foi verificada. Para os dois casos, quando a simulação teve seu *status* atualizado para "FINALIZADO", o botão de *download* apareceu na parte superior direita do *card*. Ao clicar nesse botão, uma nova aba foi aberta no navegador, abrindo o a URL dos arquivos e iniciando o *download*. Todos os dados necessários (dados de entrada, saída e imagem de visualização) vieram dentro de um arquivo comprimido em formato ZIP, conforme esperado.

4.2.2 Tempo de Carregamento das Páginas

Para validar a velocidade do carregamento das páginas, foi utilizado a ferramenta de depuração do navegador Google Chrome em uma janela anônima, para que extensões ou *caches* não atrapalhassem a medição. Utilizando a ferramenta do navegador na aba "Rede", é possível visualizar o tempo que a página leva para carregar todo o seu conteúdo - tempo para obter os arquivos HTML, CSS e JS somado ao tempo para requisitar a API e solicitar os dados de apresentação. A Figura 39 exibe um exemplo de medição realizada

na página de visualização de simulações. O tempo de carregamento é exibido na parte inferior da imagem, após o texto “Concluir”.

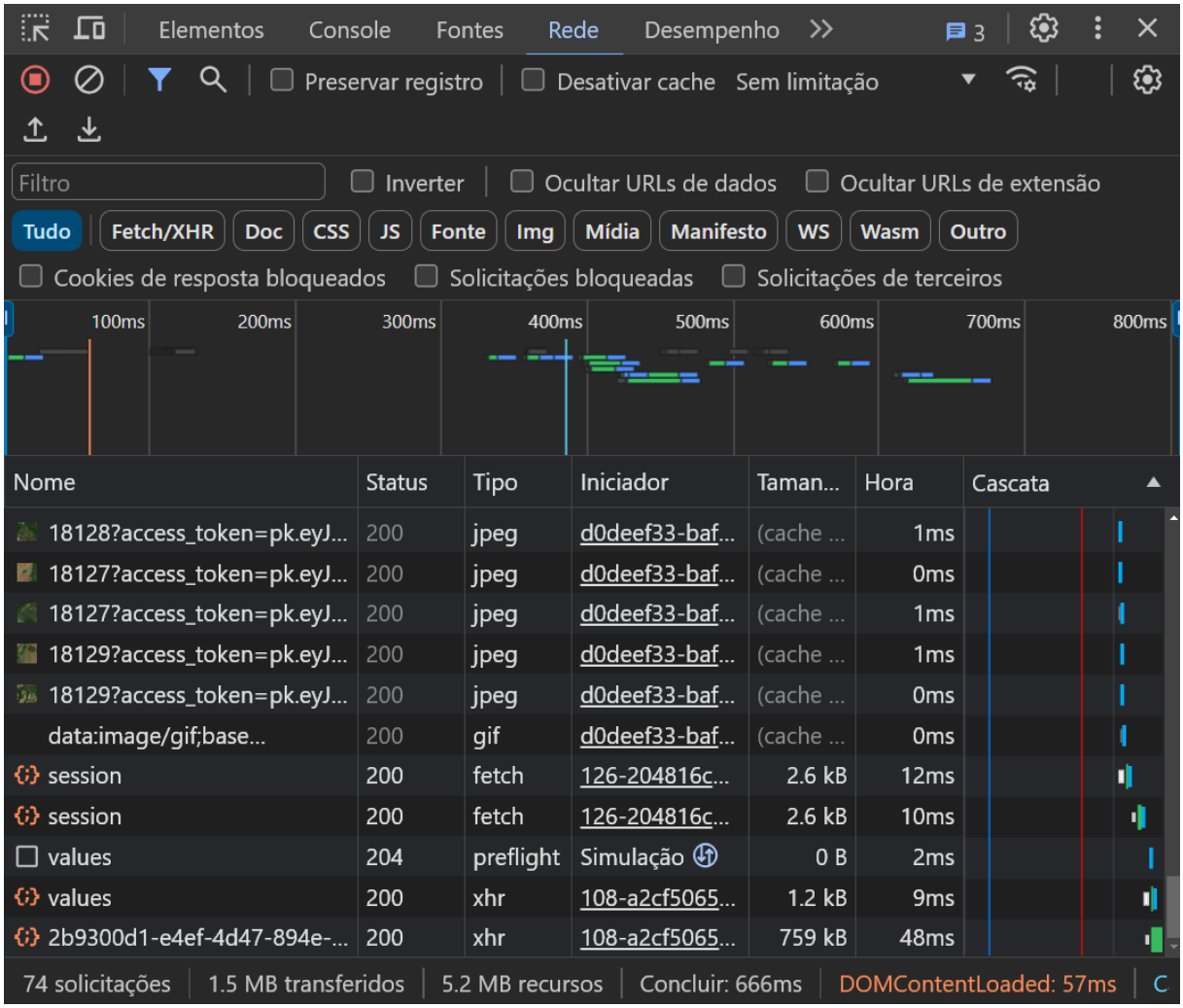


Figura 39 – Medição de tempo de carregamento de página

Visualização de simulações (com uma simulação selecionada) e cadastro de simulações foram as duas páginas testadas, já que apresentam o mapa da aplicação, resultando em diversas requisições de imagens de satélite que são custosas. Nossa escolha se baseia na premissa que, se ambas as páginas apresentarem resultados satisfatórios no tempo de carregamento, as demais também apresentarão, pois precisam solicitar menos conteúdo de fontes externas para sua completa exibição.

Os tempos para ambas as páginas estão descritos a seguir e representam a média do tempo de carregamento após 10 testes:

- **Visualização de simulações**: 739,1 milissegundos
- **Cadastro de simulações**: 613,1 milissegundos

Os tempos médios registrados para o carregamento das duas páginas testadas foram satisfatórios, já que o requisito não-funcional associado exige que as páginas não levem mais do que 2 segundos para apresentaram uma resposta à ação do usuário. Desse modo, podemos concluir que a plataforma apresenta um desempenho rápido quando se trata da navegação e usabilidade de suas funcionalidades.

5 Conclusões

Este trabalho consiste no desenvolvimento e implementação do *Forest Fire Simulation Web App* (FFSWA) representa um passo em direção à disponibilização de uma plataforma acessível e eficiente para o estudo e simulação do comportamento de incêndios florestais. O sistema foi projetado com o objetivo de facilitar tanto o desenvolvimento quanto o uso de modelos de simulação baseados em autômatos celulares, visando contribuir no entendimento da dinâmica dos incêndios em áreas com cobertura vegetal e na prevenção dos impactos devastadores desses eventos na natureza. Através da plataforma FFSWA, os usuários têm acesso a uma interface amigável que permite a solicitação e visualização de simulações de incêndio, bem como o *download* de dados brutos para análises mais detalhadas.

Os objetivos inicialmente propostos foram alcançados com sucesso. A plataforma foi desenvolvida de acordo com os requisitos funcionais e não-funcionais levantados e métodos eficazes foram implementados para a visualização das simulações, permitindo aos usuários compreenderem de forma intuitiva os resultados obtidos. Além disso, a validação do sistema através de um estudo de caso com o mapa de cobertura vegetal do Parque Estadual do Pau Furado demonstrou a eficácia e a aplicabilidade do FFSWA em cenários reais.

No entanto, é importante ressaltar que ainda existem pontos a serem aprimorados no sistema. Entre eles, destaca-se a implementação de um CRUD (*Create*, *Read*, *Update* and *Delete*) completo para a gestão de simuladores e regiões pelos administradores, visando proporcionar uma maior flexibilidade e controle sobre os modelos disponíveis na plataforma. Em outras palavras, o sistema de gestão dos simuladores e regiões pode ser melhorado através de novos módulos que permitam editar as características desses elementos. Adicionar o suporte a simuladores que não sejam implementados em JAVA pode ampliar significativamente o alcance e a utilidade do FFSWA, possibilitando a integração de novos modelos e abordagens de simulação. Espera-se que a incorporação dessas melhorias e a disponibilização de novas funcionalidades, aumente o valor agregado da plataforma, tornando-a mais atrativa a novos usuários. Assim, o contínuo desenvolvimento e aperfeiçoamento do FFSWA têm o potencial de fortalecer a capacidade de pesquisa e prevenção de incêndios florestais, contribuindo para a proteção e preservação dos ecossistemas em todo o mundo.

Em suma, o Forest Fire Simulation Web App representa uma ferramenta promissora para a comunidade acadêmica e profissional envolvida no estudo e combate de incêndios florestais, oferecendo recursos essenciais para avançar no conhecimento e na

mitigação desses eventos.

Com o estudo de caso, foi possível notar que a plataforma desenvolvida cumpre com os objetivos propostos, mas também foram identificadas oportunidades de melhoria, incluindo maior completude nas funcionalidades de gestão de simuladores e regiões, bem como o suporte a simuladores implementados em outras linguagens além de JAVA. O contínuo desenvolvimento e aperfeiçoamento do FFSWA têm o potencial de fortalecer a capacidade de pesquisa e prevenção de incêndios florestais, contribuindo para a proteção e preservação dos ecossistemas em todo o mundo.

Referências

ADOBE, I. **What are raster image files?** 2024. Disponível em: <https://www.adobe.com/creativecloud/file-types/image/raster.html>. Acesso em: 10 abr. 2024. Citado na página 16.

ALEXANDRIDIS, A.; VAKALIS, D.; SIETTOS, C.; BAFAS, G. A cellular automata model for forest fire spread prediction: The case of the wildfire that swept through spetses island in 1990. **Applied Mathematics and Computation**, v. 204, n. 1, p. 191–201, 2008. ISSN 0096-3003. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0096300308004943>. Citado na página 15.

BERTO, F.; TAGLIABUE, J. Cellular Automata. In: ZALTA, E. N.; NODELMAN, U. (Ed.). **The Stanford Encyclopedia of Philosophy**. Winter 2023. Metaphysics Research Lab, Stanford University, 2023. Disponível em: <https://plato.stanford.edu/archives/win2023/entries/cellular-automata>. Acesso em: 04 mar. 2024. Citado na página 13.

DATA, I. P. **Prisma | Simplify working and interacting with databases**. 2024. Disponível em: <https://www.prisma.io/>. Acesso em: 07 mai. 2024. Citado na página 35.

FERREIRA, H. F.; TINOCO, C. R.; MARTINS, L. G. A.; OLIVEIRA, G. M. B. Stochastic model for wildfire simulation based on the characteristics of the brazilian cerrado. In: RUTKOWSKI, L.; SCHERER, R.; KORYTKOWSKI, M.; PEDRYCZ, W.; TADEUSIEWICZ, R.; ZURADA, J. M. (Ed.). **Artificial Intelligence and Soft Computing**. Cham: Springer Nature Switzerland, 2023. p. 487–496. ISBN 978-3-031-42505-9. Disponível em: <https://dx.doi.org/10.1007/978-3-031-42505-9_41>. Acesso em: 03 mar. 2024. Citado 6 vezes nas páginas 10, 12, 15, 41, 55 e 56.

FERREIRA, M. E. de Ávila. Ajuste evolutivo de parâmetros de autômatos celulares probabilísticos em modelos de propagação de incêndios. Universidade Federal de Uberlândia, 2023. Disponível em: <http://doi.org/10.14393/ufu.di.2023.192>. Citado 3 vezes nas páginas 4, 13 e 14.

GARRET, F. **O que é Figma? Quatro perguntas sobre como usar o site**. 2021. Disponível em: <https://www.techtudo.com.br/listas/2021/06/o-que-e-figma-quatro-perguntas-sobre-como-usar-o-site.ghtml>. Acesso em: 26 dez. 2022. Citado na página 29.

HACKELOEER KLAAS KLASING, J. M. K. A.; MENG, L. Georeferencing: a review of methods and applications. **Annals of GIS**, Taylor & Francis, v. 20, n. 1, p. 61–69, 2014. Disponível em: <https://doi.org/10.1080/19475683.2013.868826>. Citado 3 vezes nas páginas 16, 17 e 18.

Hernández Encinas, A.; Hernández Encinas, L.; Hoya White, S.; Martín del Rey, A.; Rodríguez Sánchez, G. Simulation of forest fire fronts using cellular automata. **Advances in Engineering Software**, v. 38, n. 6, p. 372–378, 2007. ISSN 0965-9978. Advances in Numerical Methods for Environmental Engineering. Disponível em:

<https://www.sciencedirect.com/science/article/pii/S0965997806001293>. Citado 2 vezes nas páginas 14 e 15.

IEF, I. E. d. F. **Parque Estadual do Pau Furado**. 2024. Disponível em: <http://www.ief.mg.gov.br/component/content/article/205>. Acesso em: 07 mai. 2024. Citado na página 41.

IXDF, I. D. F. **What is User Experience (UX) Design?** 2016. Disponível em: <https://www.interaction-design.org/literature/topics/ux-design>. Acesso em: 10 abr. 2024. Citado na página 18.

_____. **What is User Interface (UI) Design?** 2016. Disponível em: <https://www.interaction-design.org/literature/topics/ui-design>. Acesso em: 10 abr. 2024. Citado na página 18.

MANSOOR, S.; FAROOQ, I.; KACHROO, M. M.; MAHMOUD, A. E. D.; FAWZY, M.; POPESCU, S. M.; ALYEMENI, M.; SONNE, C.; RINKLEBE, J.; AHMAD, P. Elevation in wildfire frequencies with respect to the climate change. **Journal of Environmental Management**, v. 301, p. 113769, 2022. ISSN 0301-4797. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0301479721018314>. Citado na página 10.

MUTTHULAKSHMI, K.; WEE, M. R. E.; WONG, Y. C. K.; LAI, J. W.; KOH, J. M.; ACHARYA, U. R.; CHEONG, K. H. Simulating forest fire spread and fire-fighting using cellular automata. **Chinese Journal of Physics**, v. 65, p. 642–650, 2020. ISSN 0577-9073. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0577907320300873>. Citado na página 15.

NAH, F. F.-H. A study on tolerable waiting time: how long are web users willing to wait? **Behaviour & Information Technology**, Taylor & Francis, v. 23, n. 3, p. 153–163, 2004. Disponível em: <https://doi.org/10.1080/01449290410001669914>. Citado na página 22.

NEXTUI, I. **NextUI - Beautiful, fast and modern React UI Library**. 2024. Disponível em: <https://nextui.org/>. Acesso em: 07 mai. 2024. Citado 2 vezes nas páginas 22 e 29.

PRATAMA, M. A. T.; CAHYADI, A. T. Effect of user interface and user experience on application sales. **IOP Conference Series: Materials Science and Engineering**, IOP Publishing, v. 879, n. 1, p. 012133, jul 2020. Disponível em: <https://dx.doi.org/10.1088/1757-899X/879/1/012133>. Acesso em: 03 mar. 2024. Citado 2 vezes nas páginas 11 e 18.

QGIS, p. **Georeferencer**. 2024. Disponível em: <https://docs.qgis.org/3.34/en/docs/user_manual/working_with_raster/georeferencer.html>. Acesso em: 10 abr. 2024. Citado 2 vezes nas páginas 16 e 17.

RIBEIRO, J. F.; WALTER, B. M. T. As principais fitofisionomias do bioma cerrado. **Cerrado: ecologia e flora**, Embrapa Cerrados Planaltina, v. 1, p. 151–212, 2008. Citado na página 15.

SCHMIDT, I. B.; ELOY, L. Fire regime in the brazilian savanna: Recent changes, policy and management. **Flora**, v. 268, p. 151613, 2020. ISSN 0367-2530. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0367253020300773>. Citado na página 10.

SOCIETY, I. C. **Software Engineering Body of Knowledge**. London: IEEE Computer Society Staff, 2014. Citado na página 29.

TANSTACK, L. **Overview | TanStack Query React Docs**. 2024. Disponível em: <https://tanstack.com/query/latest/docs/framework/react/overview>. Acesso em: 07 mai. 2024. Citado na página 40.

TASKFORCE.SH, I. **What is BullMQ**. 2024. Disponível em: <https://docs.bullmq.io/>. Acesso em: 07 mai. 2024. Citado na página 35.

VERCEL, I. **Next.js by Vercel - The React Framework**. 2024. Disponível em: <https://nextjs.org/>. Acesso em: 07 mai. 2024. Citado na página 39.

WOLFRAM, S. Cellular automata as models of complexity. **Nature**, v. 311, n. 5985, p. 419–424, Oct 1984. ISSN 1476-4687. Disponível em: <https://doi.org/10.1038/311419a0>. Citado na página 13.