Documentação do Projeto: Sistema de Gerenciamento de Lavanderias ## 1. Introdução Este documento descreve a implementação do Sistema de Gerenciamento de Lavanderias, um projeto full-stack desenvolvido para permitir o gerenciamento de lavanderias, suas máquinas e pagamentos, além de oferecer uma interface para clientes consultarem a disponibilidade das máquinas. O backend foi construído utilizando Django e Django Rest Framework (DRF), enquanto o frontend foi desenvolvido com React e Tailwind CSS. ## 2. Arquitetura da Aplicação O sistema é dividido em duas partes principais: * **Backend (Django + DRF):** Responsável pela lógica de negócios, gerenciamento de dados (Lavanderias, Máquinas, Pagamentos) e exposição de APIs RESTful para consumo pelo frontend. * **Frontend (React + Tailwind CSS):** Responsável pela interface do usuário, permitindo a interação do administrador para gerenciar o sistema e dos clientes para consultar informações. ### 2.1. Backend * **Frameworks:** Django, Django Rest Framework * **Banco de Dados:** SQLite (padrão do Django para desenvolvimento, pode ser alterado para produção) * **Autenticação:** Autenticação básica do DRF (sem JWT, conforme solicitado). * **Principais Modelos:** * `Lavanderia`: Armazena informações sobre cada lavanderia (nome, endereço, status, localização). * `Maquina`: Associada a uma `Lavanderia`, armazena o status da máquina (livre/ocupada) e previsão de liberação. * `Pagamento`: Registra os pagamentos e seu status de aprovação. A aprovação de um pagamento atualiza o status da máquina correspondente. * **Principais APIs:** * `/api/lavanderias/`: CRUD para Lavanderias. * `/api/maquinas/`: CRUD para Máquinas. * `/api/pagamentos/`: CRUD para Pagamentos. A criação/atualização de um pagamento (marcando como aprovado) dispara a lógica para ocupar a máquina e definir a previsão de liberação. * `/api/consulta-maquinas/`: Endpoint de leitura para clientes consultarem o status e previsão de liberação das máquinas. ### 2.2. Frontend * **Frameworks/Bibliotecas:** React, Tailwind CSS, Axios (para chamadas HTTP). * **Estrutura de Componentes (Exemplos):** * `LavanderiaList`: Exibe a lista de lavanderias. * `MaquinaList`: Exibe a lista de máquinas, podendo ser filtrada por lavanderia. * `MaquinaStatus`: Exibe o status de uma máquina individual. * `MapaLavanderias` (Placeholder): Representa onde o mapa interativo seria renderizado. * `PagamentoForm`: Formulário para simular o processo de pagamento de uma máquina. * **Interação com API:** O frontend consome as APIs do backend para buscar e enviar dados. ## 3. Como Rodar o Projeto Localmente Siga os passos abaixo para configurar e executar o projeto em seu ambiente local. ### 3.1. Pré-requisitos * Python 3.8 ou superior * Node.js e npm (ou Yarn) * Git (opcional, para clonar o repositório) ### 3.2. Configuração do Backend (Django) 1. **Clone o repositório (se aplicável) ou extraia os arquivos do backend.** 2. **Navegue até o diretório do backend:** ```bash cd caminho/para/o/projeto/backend ``` 3. **Crie e ative um ambiente virtual (recomendado):** ```bash python -m venv backend_venv source backend_venv/bin/activate # No Linux/macOS # backend_venv\Scripts\activate # No Windows ``` 4. **Instale as dependências Python:** ```bash pip install -r requirements.txt ``` *(Nota: O arquivo `requirements.txt` será gerado ao final do desenvolvimento do backend. Por enquanto, as dependências principais são `django` e `djangorestframework`)* 5. **Aplique as migrações do banco de dados:** ```bash python manage.py makemigrations api python manage.py migrate

``` 6. **Crie um superusuário para acessar o Admin Django (opcional, mas recomendado):** ```bash python manage.py createsuperuser ``` Siga as instruções para definir nome de usuário, email e senha. 7. **Inicie o servidor de desenvolvimento do Django:** ```bash python manage.py runserver ``` Por padrão, o backend estará rodando em `http://localhost:8000`. Você poderá acessar o Admin Django em `http://localhost:8000/admin/`. ### 3.3. Configuração do Frontend (React) 1. **Navegue até o diretório do frontend:** ```bash cd caminho/para/o/projeto/frontend ``` 2. **Instale as dependências do Node.js:** ```bash npm install ``` 3. **Inicie o servidor de desenvolvimento do React:** ```bash npm start ``` Por padrão, o frontend estará rodando em `http://localhost:3000` e tentará se conectar ao backend em `http://localhost:8000/api` (configurado em `frontend/src/config.js`). ## 4. Dependências ### 4.1. Backend * Django: Framework web de alto nível. * Django Rest Framework: Toolkit poderoso e flexível para construir APIs Web. * (Outras dependências podem ser adicionadas e serão listadas no `requirements.txt` final) ### 4.2. Frontend * React: Biblioteca JavaScript para construir interfaces de usuário. * Tailwind CSS: Framework CSS utility-first para design rápido. * Axios: Cliente HTTP baseado em Promises para fazer requisições às APIs. * PostCSS: Ferramenta para transformar CSS com plugins JavaScript. * Autoprefixer: Plugin PostCSS para adicionar prefixos de fornecedores ao CSS. * (Outras dependências podem ser adicionadas e estarão listadas no `package.json`) ## 5. Como Usar a Aplicação ### 5.1. Visão Geral A aplicação principal (acessada via `http://localhost:3000` após iniciar o frontend) exibe uma visão geral do sistema, incluindo: * **Lista de Lavanderias:** Mostra as lavanderias cadastradas com nome, endereço, status e localização. * **Status das Máquinas:** Exibe todas as máquinas do sistema, seu status (livre/ocupada) e, se ocupada, a previsão de liberação. * **Mapa das Lavanderias (Placeholder):** Indica onde um mapa interativo com a localização das lavanderias seria exibido. * **Pagamento:** Permite selecionar uma máquina (que esteja livre) e simular um processo de pagamento. Após o pagamento simulado ser "aprovado", o status da máquina é atualizado para "ocupada" e uma previsão de liberação é definida (atualmente, 1 hora a partir do momento do pagamento). ### 5.2. Admin Django (`http://localhost:8000/admin/`) O Admin Django é a interface de administração do backend. Após criar um superusuário, você pode acessá-la para: * **Gerenciar Lavanderias:** Criar, visualizar, editar e excluir lavanderias. * **Gerenciar Máquinas:** Criar, visualizar, editar e excluir máquinas, associando-as às lavanderias. * **Gerenciar Pagamentos:** Visualizar os registros de pagamento. *(A funcionalidade de visualização de localização no mapa diretamente no Admin Django não foi implementada como parte desta interface padrão, mas os dados de latitude e longitude são armazenados e podem ser usados por ferramentas externas ou no frontend.)* ### 5.3. Tela de Cliente (Frontend) A tela principal do frontend (`http://localhost:3000`) serve como a tela do cliente, onde é possível: * Consultar as lavanderias disponíveis. * Verificar o status das máquinas em cada lavanderia (se estão ocupadas ou livres). * Visualizar a previsão de liberação de cada máquina ocupada. * Simular um pagamento para ocupar uma máquina. ### 5.4. Tela de Admin (Frontend - Placeholder) As funcionalidades específicas da tela de admin no frontend (como criar lavanderias e máquinas diretamente pela interface React) não foram completamente implementadas nesta fase, focando-se na visualização e na simulação de pagamento. O gerenciamento de dados primariamente ocorre
```

via Admin Django. *Para criar novas lavanderias e máquinas, utilize o Admin Django.* ## 6. Considerações Adicionais * **Autenticação no Frontend:** A autenticação básica está configurada no `frontend/src/services/api.js` com credenciais fixas (`admin`/`adminpassword`). Em um ambiente de produção, isso deve ser substituído por um sistema de login seguro. * **Mapa Interativo:** O componente `MapaLavanderias.js` é um placeholder. Para uma funcionalidade de mapa real, seria necessário integrar uma biblioteca como `react-leaflet` (para OpenStreetMap) ou `@react-google-maps/api` (para Google Maps), o que pode envolver a obtenção de chaves de API. * **Processamento de Pagamento Real:** A funcionalidade de pagamento é uma simulação. Um sistema de pagamento real exigiria integração com um gateway de pagamento seguro. * **Atualização em Tempo Real:** As atualizações de status da máquina (após pagamento) podem exigir que o usuário atualize a página ou que uma lógica de polling/WebSockets seja implementada para atualizações em tempo real.