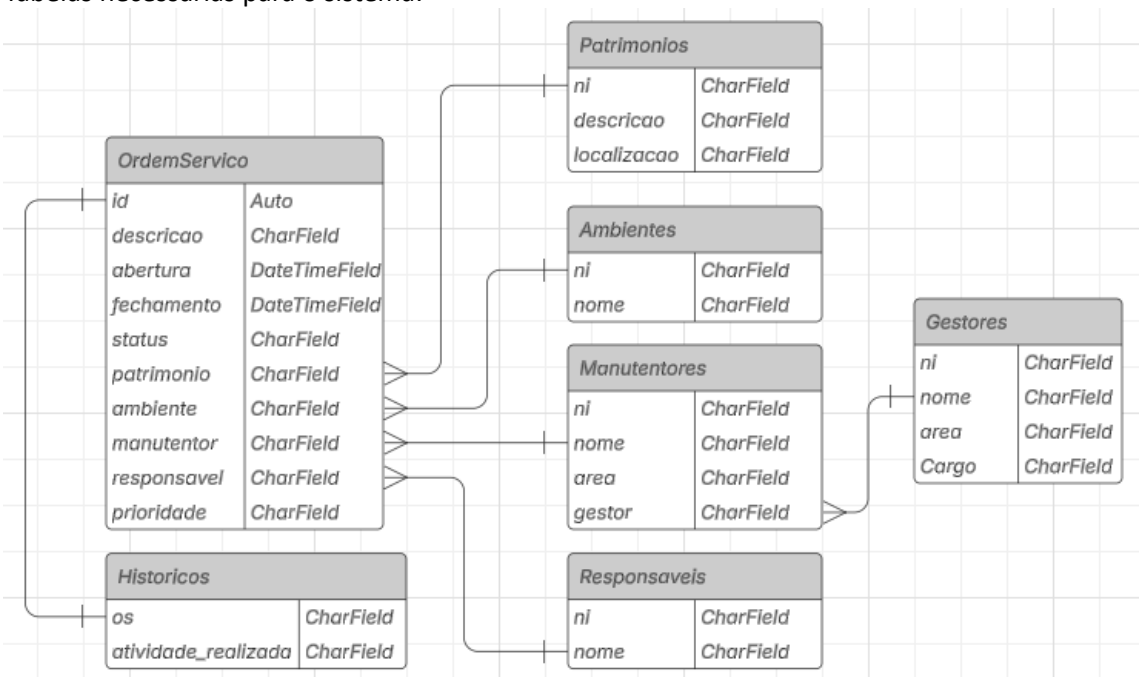
Situação Problema

Você foi contratado para desenvolver o **Sistema de Ordem de Serviço da TechEdu**, utilizando as tecnologias **Django no back end** e **React no front end**. O sistema deve incluir as seguintes funcionalidades:

1. **Cadastro e Autenticação de Usuários:**
    - Implementar um sistema de login e logout com autenticação via **JWT**.
    - Somente usuários autenticados podem criar e visualizar ordens de serviço.
    - Implementar **sign_in, sign_up e logoff** utilizando Django Rest Framework com **JWT**.
    - Criar um sistema de permissões para diferenciar **técnicos**, **chefes de manutenção** e **administradores**.
    -
2. Tabelas necessárias para o sistema:



- **Ordem de Serviço** (*OrdemServico*)
    - **id**: gerado automaticamente pelo banco de dados e será o número da Ordem de Serviço.
    - **descrição**: local onde serão descritos os problemas.
    - **abertura**: data e hora da abertura da OS.
    - **fechamento**: data e hora do fechamento da OS.
    - **status**: serão 3 status (iniciada, em andamento, finalizada, cancelada).
    - **patrimônio**: número do patrimônio que virá da tabela *patrimônios*, esse campo não deve ser obrigatório já que nem toda OS é para equipamento ou algo que tenha número, exemplo alvenaria.
    - **ambiente**: número do ambiente que virá da tabela *ambientes.*
    - **manutentor**: número do manutentor que virá da tabela *manutentores.*
    - **responsável**: número do responsável que virá da tabela *responsáveis*, esse campo não deve ser obrigatório.
    - **prioridade**: serão 3 prioridades (alta, media e baixa).

- **Patrimônios**
    - **ni:** número do equipamento, quando houver.

- **descrição:** detalhes sobre o patrimônio.
- **localizacao:** local físico.

- **Ambientes**
  - **ni:** código do ambiente, exemplo: LabA105.
  - **nome:** exemplo: Laboratório de Informática.

- **Manutentores**
  - **ni:** código alfanumérico do funcionário, exemplo: sn1021328.
  - **nome:** nome do manutentor, exemplo: Lindomar José Batistão.
  - **area:** exemplo: Informática
  - **gestor:** nome do gestor desse manutentor que virá da tabela *Gestores*.

- **Responsaveis**
  - **ni:** código alfanumérico do funcionário, exemplo: sn1021328.
  - **nome:** nome do manutentor, exemplo: José da Silva.

- **Gestores**
  - **ni:** código alfanumérico do funcionário, exemplo: sn1021328.
  - **nome:** nome do manutentor, exemplo: José da Silva.
  - **area:** área de atuação do gestor, exemplo: elétrica, informática etc.
  - **cargo:** exemplo: Operador de Práticas Profissionais, Coordenador Técnico, Coordenador Pedagógico.

- **Histórico**
  - **os:** número da ordem de serviço realizada.
  - **atividade_realizada:** descrição da tarefa que foi feita.

Observações:

- pode-se simplificar os nomes dos campos, mas se fizer coloque por extenso nos comentários.
- Algumas planilhas serão disponibilizadas para popular o banco de dados, crie um código em Python para isso, se preferir pode ser com endpoints.

3. **Relacionamento entre tabelas**
   - Os relacionamentos deverão ser aplicados nas tabelas conforme diagrama já mencionado acima, lembrando que não foram explicados em aula, portanto você deverá fazer pesquisas para implementação.
   - No front-end, dados de tabelas relacionadas deverão ser listados nos campos relacionados.

4. **Gerenciamento de Ordens de Serviço:**
   - Em todas as páginas os elementos deverão ser listados com as opções de CRUD para cada registro.
   - Desenvolva opções de localização de dados.

   - Atualizar o status da OS (pendente, em andamento, concluído).

5. **Gerenciamento de Acesso:**

- Como pode ser observado, todas as tabelas acima são de uso administrativo, ou seja, somente o gestor de manutenção que deverá ter todas essas opções. Você deverá pesquisar como limitar os acessos aos manutentores e funcionários nas tarefas.
- Funcionários podem **criar** ordens de serviço.
- Técnicos de manutenção podem **visualizar e atualizar** o status das OS.
- O administrador tem acesso total.

6. **Integração entre Front End e Back End:**
   - Utilizar **Axios** no React para consumir a API Django.
   - Criar uma interface intuitiva para cadastro e acompanhamento das OS.

Critérios de Avaliação – Back End (Django)

Critério	**Descrição**	**Peso (%)**
Autenticação e Permissões	Implementação de **sign in, sign up e logoff** com JWT no Django Rest Framework, além da criação de permissões para técnicos, OPPs e administradores.	**20%**
Modelagem de Dados (Django)	Criação correta dos modelos (OrdemServico, Patrimonios, Ambientes, Manutentores, Responsaveis, Gestores e Históricos) com relações apropriadas (ForeignKeys) e validações.	**10%**
API Rest (Django Rest Framework)	Implementação dos endpoints **CRUD** para Ordens de Serviço e usuários, incluindo filtros por ambiente, status e período.	**25%**
Interface do Usuário (React)	Construção da interface usando React, incluindo todas as páginas, ou seja, uma para cada tabela no mínimo.	**15%**
Consumo da API (Axios e React)	Comunicação correta entre o front end e o back end usando Axios para listar, criar e atualizar OS.	**10%**
Funcionalidades Extras 1	Implementação de **histórico** e **exportação** de relatórios em Excel(XLSX ou CSV), alertas e notificações para mudanças no status das OS.	**5%**
Funcionalidades Extras 2	Desenvolvimento de código para popular o banco.	**5%**
Organização do Código e Boas Práticas	Estrutura do código, modularidade, uso adequado de componentes no React e organização do código Django.	**10%**

Critérios de Avaliação – Front End (React)

Critério	Peso (%)	Descrição
Interface e Experiência do Usuário (UX/UI)	20%	O design da aplicação é intuitivo, responsivo e de fácil navegação. Botões, formulários e listagens são bem organizados.
Funcionalidade e Dinâmica	20%	As funcionalidades implementadas atendem aos requisitos do sistema. O usuário consegue cadastrar, visualizar e atualizar ordens de serviço sem erros.
Página Inicial	10%	Desenvolva a página Home com todas as opções de navegação.
Integração com Back End (API Django)	10%	Consumo da API via Axios, envio e recebimento de dados corretamente. Os estados da aplicação são bem gerenciados.
Cabeçalho e Rodapé	15%	Desenvolva apenas 1 cabeçalho e 1 rodapé para todas as páginas, exceto a de login. O título de cada cabeçalho deverá ser de acordo com a página, exemplo: página de Ordem de Serviço deverá ter o mesmo tema. Adicione o logout no cabeçalho de forma que limpe o token e retorne para a página de login.
Código e Organização	10%	Código limpo, bem estruturado, uso adequado de componentes reutilizáveis e boas práticas de desenvolvimento.
Autenticação e Controle de Acesso	15%	Implementação correta do login/logout, proteção de rotas e armazenamento seguro do token JWT.