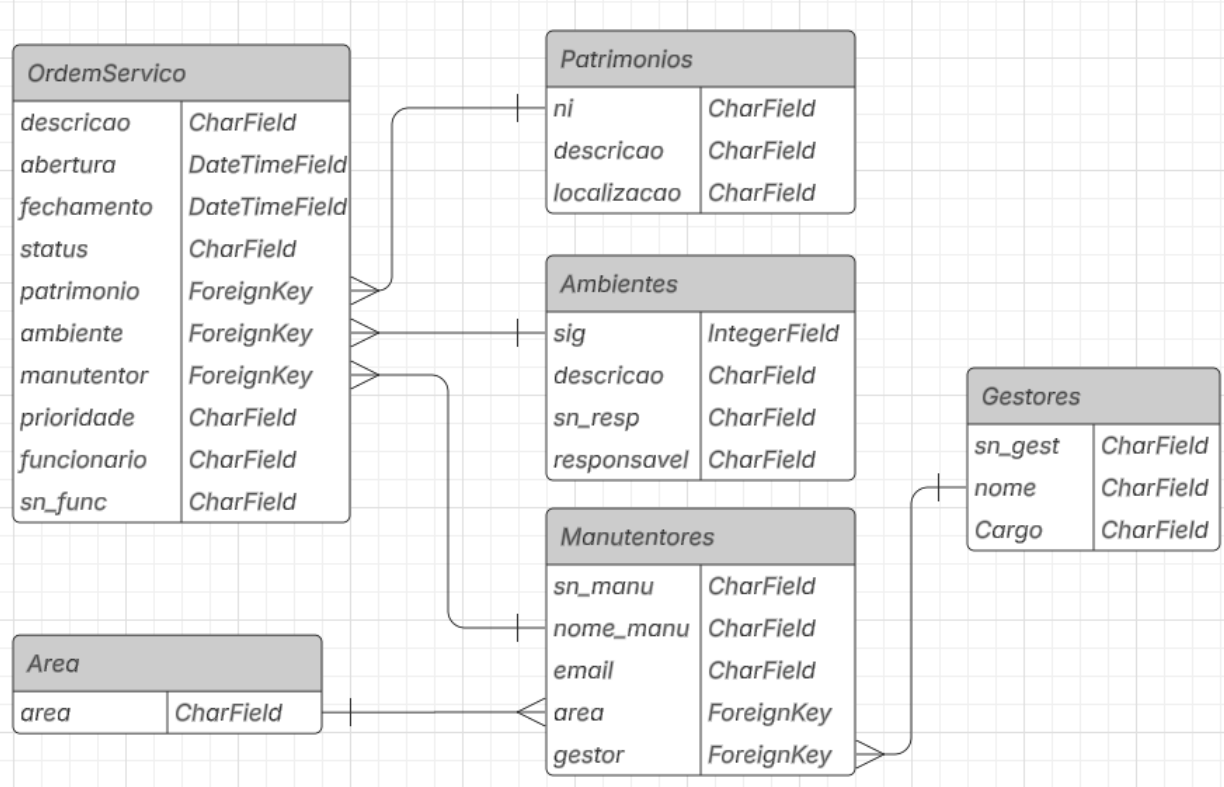


ATIVIDADE	Situação Problema
Contexto	
<p>Você foi contratado para desenvolver o Sistema de Ordem de Serviço da TechEdu, utilizando as tecnologias Django no back end e React no front end. O sistema deve incluir as seguintes funcionalidades:</p> <ol style="list-style-type: none"> Cadastro e Autenticação de Usuários: <ul style="list-style-type: none"> 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. Tabelas necessárias para o sistema: <div data-bbox="199 772 1428 1556">  <pre> graph LR OS[OrdemServico] --> P[Patrimonios] OS --> A[Ambientes] OS --> M[Manutentores] OS --> Ar[Area] M --> G[Gestores] Ar --> M </pre> </div> <ul style="list-style-type: none"> Ordem de Serviço (OrdemServico) <ul style="list-style-type: none"> descrição: descrição do problema. abertura: data e hora da abertura da OS. fechamento: data e hora do fechamento da OS. status: serão 4 status (iniciada, pendente, finalizada, cancelada). patrimônio: número do patrimônio que virá da tabela <i>patrimônios</i>, 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 <i>ambientes</i>. manutentor: número do manutentor que virá da tabela <i>manutentores</i>. prioridade: serão 3 prioridades (alta, media e baixa). funcionário: nome do funcionário que abriu a ordem. Capture o nome do usuário que logou. sn: código alfanumérico do funcionário, exemplo: sn1021328. 	

- **Patrimônios**
 - **ni:** número do equipamento, quando houver.
 - **descrição:** detalhes sobre o patrimônio.
 - **localizacao:** local físico.
- **Ambientes**
 - **sig:** código do ambiente, exemplo: 20400024.
 - **descricao:** exemplo: LAB. DE INFORMÁTICA A04.
 - **sn:** identificação do funcionário, exemplo: SN1085371.
 - **responsável:** nome do responsável pelo laboratório, exemplo: VICTOR SERRA BRAGA LEMOS.
- **Manutentores**
 - **sn:** código alfanumérico do funcionário, exemplo: sn1021328.
 - **nome:** nome do manutentor, exemplo: Lindomar José Batistão.
 - **email:** e-mail do manutentor.
 - **area:** exemplo: Informática.
 - **gestor:** nome do gestor desse manutentor que virá da tabela *Gestores*.
- **Gestores**
 - **sn:** código alfanumérico do funcionário, exemplo: sn1021328.
 - **nome:** nome do manutentor, exemplo: José da Silva.
 - **cargo:** exemplo: Operador de Práticas Profissionais, Coordenador Técnico, Coordenador Pedagógico.
- **Area**
 - **area:** área de atuação, exemplo: elétrica, informática, alvenaria, pintura, mecânica etc.

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 métodos ou classes nas **views** para popular o banco.
- **Não** é aconselhável hospedar em repositório **público**.
- **No caso de plágio os 2 alunos ficarão com zero.**

3. Relacionamento entre tabelas

- Os relacionamentos deverão ser aplicados nas tabelas conforme diagrama já mencionado acima.
- No front-end, dados de tabelas relacionadas deverão ser listados nos campos relacionados.

4. Gerenciamento de Ordens de Serviço:

- Em todas as páginas que possuam elementos listados deverão possuir as opções de CRUD para cada registro.
- [Desenvolva barra de pesquisa de dados como ID e Nome.](#)
- Atualizar o status da OS (iniciada, pendente, finalizada, cancelada).

5. 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.
- Inicie com uma página de login com a opção de cadastro de usuário.
- Ao logar direcione para a página home em que teremos todas as opções, ou seja, como para cada tabela será criada uma página então deve-se colocar todos os links para todas as páginas.
- A página de Ordem de Serviço deverá possuir apenas o Create, já as outras deverá possuir o CRUD completo.

6. Popular o banco de dados:

- Crie uma página para popular o banco de dados, deve-se ter a opção de escolher qual tabela será populada e escolher a planilha.
- Pode-se popular apenas a patrimônio.

Critérios de Avaliação – Back End (Django)

Critério	Descrição	Peso (%)
Autenticação e Permissões	Implementação de sign in com JWT no Django Rest	5
	Desenvolver página de cadastro de usuários sign up	5
Modelagem de Dados (Django) Criação correta dos modelos	Modelagem de todas as tabelas: OrdemServico, Patrimonios, Ambientes, Manutentores, Gestores e Area.	15
	Relações apropriadas (ForeignKeys) e validações.	5
API Rest (Django Rest Framework)	Implementação dos endpoints CRUD para todas as páginas que possua dados.	20
	Incluir filtros pelo menos em Ordem de Serviço e Patrimônio .	5
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	Implementação de exportação de relatórios em Excel(XLSX ou CSV).	10
Funcionalidades Extras 2	Desenvolvimento de código para popular o banco a partir de planilhas disponibilizadas.	5
Organização do Código e Boas Práticas	Estrutura do código, modularidade e organização do código Django. Código limpo.	5

Crítérios de Avaliação – Front End (React)

Nº	Item Avaliado	Descrição	Peso (%)
1	Página de Login Funcional	Interface de login limpa, com validação e envio correto do JWT para autenticação	3
2	Cadastro de Usuário (Sign Up)	Página de cadastro com campos obrigatórios, envio correto para a API e feedback ao usuário	5
3	Redirecionamento após Login	Após login bem-sucedido, redireciona para a Home do sistema	2
4	Logout com Limpeza de Token	Implementação de logout que remove o JWT e redireciona para o login	3
5	Proteção de Rotas	Páginas protegidas por verificação do JWT; usuários não logados são redirecionados a uma página que lhe informará que não está logado . Crie essa página.	5
6	Página Inicial com Navegação Completa	Página Home com links para todas as funcionalidades (OS, Patrimônio, Ambientes, etc)	5
7	Cabeçalho Reutilizável	Componente de cabeçalho presente em todas as páginas (exceto login/cadastro) com título e botão de logout. Adicione como título desse cabeçalho o nome da página, exemplo: Patrimônio, quando estiver com a lista de patrimônios. Observação: apenas 1 cabeçalho para todas as páginas previstas.	5
8	Rodapé Reutilizável	Rodapé padrão com informações da aplicação, presente em todas as páginas (exceto login/cadastro) Observação: apenas 1 rodapé para todas as páginas previstas.	2
9	Tela de Criação de Ordem de Serviço	Interface funcional que permite criação de OS com campos obrigatórios, dropdowns de tabelas relacionadas	5
10	Listagem de Ordens de Serviço com Filtros	Tabela com todas as OS criadas, permite filtragem por status, prioridade, ambiente, etc	5
11	Atualização de Status da OS	Crie botão de opção para atualizar status	5
12	CRUD Completo de Patrimônios	Páginas para listar, adicionar, editar e excluir patrimônios	5
13	CRUD de Ambientes, Manutentores, Gestores e Áreas	Interfaces completas para todas as demais tabelas com rotas separadas	5
14	Histórico de Ordens de Serviço	Página com histórico (visualização apenas) das OS, com paginação se necessário	5
15	Paginação de Listagens	Implementação de paginação em listas com muitos dados (por exemplo, OS e Patrimônios)	5
16	Responsividade e UI	Layout adaptável a diferentes resoluções, com experiência fluida e intuitiva	5
17	Tratamento de Erros da API	Mensagens amigáveis de erro ao usuário quando a API retorna erro (ex: 400, 401, 500)	3
18	Indicadores de Carregamento	Uso de spinners/loaders ao carregar dados ou enviar formulários	2
19	Organização do Código	Separação adequada entre components, pages, services, utils, etc.	5
20	Reutilização e Boas Práticas	Uso adequado de props, hooks (useState, useEffect), e componentes reaproveitáveis. Código limpo.	5