ATIVIDADE	PROJETO INTEGRADOR - SITUAÇÃO PROBLEMA

CONTEXTO e DESCRIÇÃO

Você é um desenvolvedor de sistemas na empresa fictícia TechIndustries, que atua no ramo industrial. A empresa está implementando um sistema de monitoramento de ambiente em suas fábricas para garantir condições ideais de operação e segurança. Este sistema deve capturar dados de sensores de temperatura, luminosidade e umidade instalados nas fábricas.

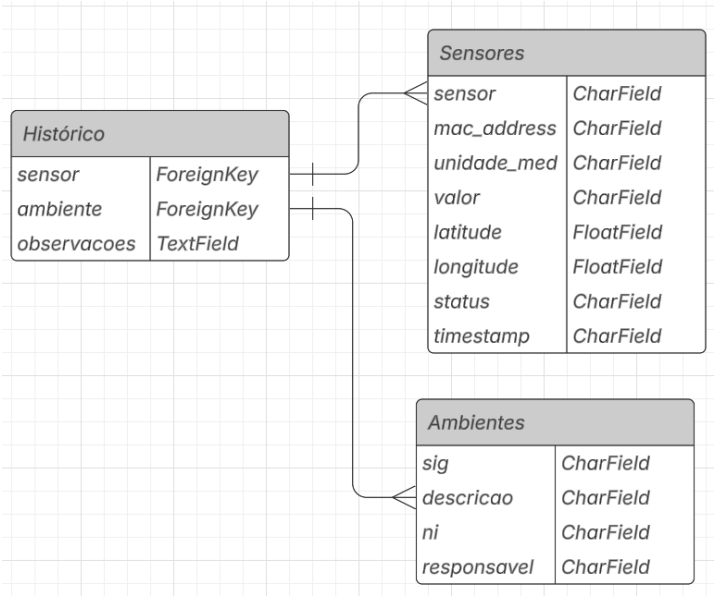
No entanto, como os sensores ainda estão em fase de aquisição e instalação, os dados fornecidos para testes serão simulados. O objetivo do projeto é desenvolver um back-end utilizando Django Rest Framework para gerenciar esses dados e disponibilizá-los para um front-end desenvolvido em React, que será usado pelos supervisores da fábrica para monitorar as condições em tempo real. A autenticação será realizada através de JSON Web Tokens (JWT).

A partir de agora criaremos um projeto chamado "smart_city". Este projeto vai comportar uma aplicação com a finalidade principal de coletar e expor os dados dos sensores da cidade inteligente. Para isso construiremos algumas APIs juntamente com o Banco de Dados da aplicação.

Requisitos do Projeto:

1. **Back-End (Django Rest Framework):**
   - Criação de uma API RESTful para gerenciar dados de sensores.
   - A API deve ter endpoints para criar, ler, atualizar e deletar (CRUD) dados dos sensores e ambientes.
   - Os dados dos sensores devem incluir:
     - Temperatura (°C)
     - Luminosidade (lux)
     - Umidade (%)
     - Contador(num)
   - Os dados devem ser armazenados em um banco de dados dbsqlite.
   - Implementar autenticação utilizando JSON Web Tokens (JWT) para proteger os endpoints.

2. **Front-End (React):**
   - Desenvolvimento de uma aplicação para exibir os dados capturados pelos sensores.
   - A aplicação deve ter uma interface amigável que permita aos usuários visualizarem gráficos e listas com os dados de temperatura, luminosidade e umidade.
   - Implementar funcionalidades de login para acesso seguro à aplicação.
   - Crie a pasta integrador e acesse a.
   - Crie o projeto smart_city.
   - Crie o app da api.
   - Pode-se simplificar os nomes dos campos, mas se fizer coloque por extenso nos comentários.
   - **Não** é aconselhável hospedar em repositório **público.**
   - **No caso de plágio os 2 alunos ficarão com zero.**



2. **Login**

   Criar um super usuário para o nosso api_smart.

   - username = seu primeiro nome (exatamente) sem acentuação.
   - password = seu número de matrícula no senai (está no portal)

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, exemplo: caso selecionarmos 1 sensor na tabela Front, todos os dados desse sensor deverão ser exibidos.
   - Deverá ser criado uma página para cada tabela além de uma home e a de login como já mencionada.

4. **Gerenciamento dos Sensores:**
   - Nas páginas Sensores e Ambientes os elementos deverão ser listados com as opções de CRUD para cada registro.
   - Desenvolva opções de localização de dados, principalmente por sensor, data e status.
   - Atualizar o status do sensor (ativo, inativo).

5. **Gerenciamento de Acesso:**
   - Como pode ser observado, todas as tabelas acima são de uso administrativo, ou seja, somente o gestor que deverá ter todas essas opções. Você deverá pesquisar como limitar os acessos aos manutentores e funcionários nas tarefas. Vide https://chatgpt.com/share/67f7fb61-42a0-8006-badd-d5cc69ccf7bd.

6. **Dados:**
   - Criar método para capturar dados de sensores e ambientes que estão nas planilhas disponibilizadas.
   - Os dados poderão ser exportados no formato de planilhas.

7. **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.

Metodologia Scrum:

A equipe utilizará a metodologia Scrum para organizar e gerenciar o desenvolvimento do projeto. O Scrum é uma estrutura ágil que promove o desenvolvimento iterativo e incremental, permitindo a adaptação rápida às mudanças e foco na entrega de valor.

Papéis no Scrum:

- **Product Owner**: Responsável por definir os requisitos e prioridades do produto. Para este projeto, o papel será desempenhado pelo instrutor.
- **Scrum Master**: Responsável por garantir que a equipe siga as práticas do Scrum. Pode ser um aluno designado ou o próprio instrutor.
- **Equipe de Desenvolvimento**: Composta pelos alunos, que são responsáveis pela implementação dos requisitos.

Artefatos do Scrum:

- **Product Backlog**: Lista priorizada de todas as funcionalidades desejadas no produto. Inclui histórias de usuário detalhando os requisitos.
- **Sprint Backlog**: Conjunto de histórias de usuário selecionadas do Product Backlog para serem trabalhadas durante a Sprint.
- **Incremento**: Soma de todos os itens do Product Backlog completados durante uma Sprint e todas as Sprints anteriores.

Eventos do Scrum:

- **Sprint Planning**: Reunião no início de cada Sprint para definir quais histórias de usuário do Product Backlog serão trabalhadas.
- **Daily Scrum**: Reuniões diárias de 15 minutos para sincronizar as atividades e resolver impedimentos.
- **Sprint Review**: Reunião no final da Sprint para revisar o trabalho realizado e adaptá-lo conforme necessário.
- **Sprint Retrospective**: Reunião para refletir sobre a Sprint e identificar melhorias para o próximo ciclo.

Tarefas a Serem Realizadas:

1. **Desenvolvimento do Back-End:**
    - **Histórias de Usuário:**
        1. Como administrador, eu quero criar um endpoint para registrar dados de sensores, para que eu possa armazenar os dados de temperatura, luminosidade e umidade.
        2. Como administrador, eu quero criar um endpoint para visualizar os dados dos sensores, para que eu possa monitorar as condições ambientais.
        3. Como administrador, eu quero implementar autenticação JWT, para garantir que apenas usuários autorizados acessem os dados.

- **Tarefas:**
  - Configurar projeto Django e instalar o Django Rest Framework e djangorestframework-jwt.
  - Criar modelos para dados de sensores.
  - Implementar serializers e views.
  - Configurar URLs e autenticação JWT.

2. **Desenvolvimento do Front-End:**
   - **Histórias de Usuário:**
     1. Como supervisor, eu quero visualizar os dados dos sensores em uma lista, para que eu possa monitorar as condições em tempo real.
     2. Como supervisor, eu quero visualizar gráficos dos dados dos sensores, para analisar as variações ao longo do tempo.
     3. Como supervisor, eu quero realizar login na aplicação, para acessar os dados de forma segura.
   - **Tarefas:**
     - Configurar projeto React.
     - Criar telas de login, registro e dashboard.
     - Conectar a aplicação à API utilizando fetch ou axios.
     - Implementar visualização de gráficos utilizando bibliotecas adequadas.
     - Implementar autenticação JWT no front-end.
3. **Testes e Simulação:**
   - Implementar scripts para gerar dados simulados de sensores.
   - Testar a API com ferramentas como Postman ou Insomnia.
   - Garantir que a aplicação móvel exiba corretamente os dados simulados.

Cronograma do Projeto:

- **Sprint 1 (1 semana):**
  - Planejamento da Sprint.
  - Configuração do ambiente de desenvolvimento (Django e React).
  - Implementação inicial do back-end (endpoints básicos e autenticação JWT).
  - Implementação inicial do front-end (telas de login e dashboard).
  - Daily Scrums e reunião de revisão e retrospectiva ao final da Sprint.
- **Sprint 2 (1 semana):**
  - Planejamento da Sprint.
  - Finalização dos endpoints e integração com banco de dados.
  - Desenvolvimento das funcionalidades de visualização de dados no front-end.
  - Testes e validação dos dados simulados.
  - Daily Scrums e reunião de revisão e retrospectiva ao final da Sprint.

Critérios de Avaliação – Back End (Django)

Nº	Critério	Descrição	Peso (%)
1	**Autenticação e Permissões**	Signin com JWT	5
2		signup	5
4	**Modelagem de Dados (Django)** Criação correta dos modelos	Modelagem de todas as tabelas: Históricos, Sensores e Ambientes.	10
5		Relações apropriadas (ForeignKeys) e validações.	5
6	**API Rest (Django Rest Framework)**	Implementação dos endpoints **CRUD** para todas as páginas que possua dados.	15
7		Incluir filtro para localizar por ID de sensor.	3
8		Incluir filtro para localizar por tipo de sensor, exemplo "temperatura".	3
9		Incluir filtro para localizar por data de sensor.	3
10		Incluir filtro para localizar por código "sig" do ambiente.	3
11		Incluir filtro para localizar por ID do Histórico.	3
12	**Consumo da API (Axios e React)**	Comunicação correta entre o front e o back usando Axios para listar.	10
13	**Funcionalidades**	Implementação de **exportação** de relatórios em Excel (XLSX ou CSV).	12
14	**Popular banco de dados** *(Observação: o ideal é quebrar o campo **timestamp** para facilitar as pesquisas)*	Desenvolvimento de código para popular o banco a partir da planilha temperatura.	2
15		Desenvolvimento de código para popular o banco a partir da planilha umidade.	2
16		Desenvolvimento de código para popular o banco a partir da planilha luminosidade.	2
17		Desenvolvimento de código para popular o banco a partir da planilha contador.	2
18	**Organização do Código e Boas Práticas**	Estrutura do código, modularidade e organização do código Django. Código limpo.	10
19	**Apresentação**	Apresentar trabalho para à classe em Canvas.	5

Critérios 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. Criar campo de confirmação de senha e testar.	**5**
3	Redirecionamento após Login	Após login bem-sucedido, redireciona para a Home do sistema	**2**
4	Relacionamento entre tabelas	No front-end, dados de tabelas relacionadas deverão ser listados nos campos relacionados, exemplo: caso selecionarmos 1 sensor na tabela Front, todos os dados desse sensor deverão ser exibidos.	**10**
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 com opção para voltar ao login.	**10**
6	Página Inicial com Navegação Completa	Página Home com links para todas as outras páginas.	**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: Histórico, quando estiver com os históricos. Observação: apenas 1 cabeçalho para todas as páginas previstas.	**10**
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.	**5**
9	CRUD	As páginas **Sensores** e **Ambientes** deverão possuir opções para listar, adicionar, editar e excluir dados.	**15**
10	Histórico de Sensores	Página com histórico (visualização apenas), com paginação.	**5**
11	Paginação de Listagens	Implementação de paginação em todas as páginas que possuem listagem de dados. É opcional deixa-la oculta caso os dados não preencham 1 página.	**5**
12	Responsividade e UI	Layout adaptável a diferentes resoluções, com experiência fluida e intuitiva.	**5**
13	Tratamento de Erros da API	Mensagens amigáveis de erro ao usuário quando a API retorna erro (ex: 400, 401, 500)	**5**
14	Indicadores de Carregamento	Uso de spinners/loaders ao carregar dados ou enviar formulários	**5**
15	Organização do Código	Separação adequada entre components, pages, services, utils, etc.	**5**
16	Reutilização e Boas Práticas	Uso adequado de props, hooks (useState, useEffect), e componentes reaproveitáveis. Apagar todos os consoles. Código limpo.	**5**
17	Gráficos	***Caso consiga mostrar graficamente os valores de cada sensor por ano, será acrescido 10 pontos na nota do front, lembrando que o limite será 100.***	**10**

Observações:

1. O desenvolvimento será **individual**.
2. Se o aluno precisar do professor caso não conseguir prosseguir por ser uma sequência, será auxiliado, mas perderá o requisito.
3. A nota do back-end valerá 80% da nota final e o front-end 20%.
4. O aluno deverá criar repositório no Github **privado** e dar acesso ao professor (lindomarbatistao@gmail.com), enviando um e-mail com endereço do repositório.