

## PLANO DE ENSINO

CURSO	MÓDULO	CÓDIGO	
Técnico em Desenvolvimento de Sistemas	Específico	PBE	
UNIDADE CURRICULAR	CARGA HORÁRIA PREVISTA	DOCENTE	TURMA(S)
Programação Back-End	150 + 20	Lindomar	2DS-TB

### OBJETIVO DO CURSO

Desenvolver capacidades técnicas e socioemocionais que permitem desenvolver sistemas promovendo a interação de aplicação entre cliente e servidor e outros sistemas computacionais, realizando persistência de dados.

### CAPACIDADES TÉCNICAS

1. Utilizar o paradigma da programação orientada a objetos
3. Aplicar técnicas de código limpo (clean code)
4. Identificar as características de programação back-end em ambiente web
5. Preparar o ambiente necessário ao desenvolvimento back-end para a plataforma web
6. Definir os elementos de entrada, processamento e saída para a programação da aplicação web
8. Definir os frameworks a serem utilizados no desenvolvimento da aplicação web
9. Utilizar interações com base de dados para desenvolvimento de sistemas web
11. Estabelecer envio de notificações entre cliente e servidor por meio de aplicação web
12. Desenvolver API (web services) para integração de dados entre plataformas
13. Desenvolver sistemas web de acordo com as regras de negócio estabelecidas

### CAPACIDADES SOCIOEMOCIONAIS

1. Demonstrar autogestão
2. Demonstrar pensamento analítico
3. Demonstrar inteligência emocional
4. Demonstrar autonomia

### CONHECIMENTOS

1. Ambiente de desenvolvimento web
  - 1.1. Definição
  - 1.2. Histórico
  - 1.3. Características
  - 1.4. Ambiente de desenvolvimento
    - 1.4.1. Instalação e configuração
    - 1.4.2. Recursos e interfaces
    - 1.4.3. Gerenciamento de dependências

## **2. Web Services**

- 2.1. Definição
- 2.2. REST
  - 2.2.1. Recursos
  - 2.2.2. Semântica da URL REST
- 2.3. Padrão JSON
  - 2.3.1. Sintaxe básica
  - 2.3.2. Tipos de dados
  - 2.3.3. Formatação
  - 2.3.4. Coleção de objetos JSON
- 2.4. XML
  - 2.4.1. Sintaxe básica
  - 2.4.2. Tipos de dados
  - 2.4.3. Formatação

## **3. Protocolo HTTP**

- 3.1. Definição
- 3.2. Métodos HTTP
  - 3.2.1. GET
  - 3.2.2. POST
  - 3.2.3. PUT
  - 3.2.4. DELETE
  - 3.2.5. PATCH
  - 3.2.6. OPTIONS
- 3.3. Tipos de passagem de parâmetros
  - 3.3.1. *Query parameters*
  - 3.3.2. *Body parameters*
- 3.4. Cabeçalhos HTTP
  - 3.4.1. *Host*
  - 3.4.2. *Accept*
  - 3.4.3. *User-Agent*
  - 3.4.4. *Request Method*
  - 3.4.5. *Accept-Language*
  - 3.4.6. *Content-Type*
  - 3.4.7. *application/json*
  - 3.4.8. *Authorization*
- 3.5. Media Types
  - 3.5.1. *application*
  - 3.5.2. *text*
  - 3.5.3. *video*
  - 3.5.4. *image*
  - 3.5.5. *vnd*
- 3.6. Códigos de status
  - 3.6.1. 1XX – Informacionais
  - 3.6.2. 2XX – Códigos de sucesso
  - 3.6.3. 3XX – Redirecionamento
  - 3.6.4. 4XX – Erros originados no cliente
  - 3.6.5. 5XX – Erros originados no servidor

## **4. Programação orientada a objetos**

- 4.1. Definição
- 4.2. Pacotes
- 4.3. Classes
  - 4.3.1. Abstrata
  - 4.3.2. Interna
  - 4.3.3. Anônima

ATIVIDADE	FORMATIVA - SITUAÇÃO PROBLEMA
CONTEXTO e DESCRIÇÃO	
<p>"Catálogo Virtual da Biblioteca da Escola"</p> <p>A direção da escola onde você trabalha como Técnico em Desenvolvimento de Sistemas solicitou a modernização do sistema da biblioteca. Atualmente, os registros dos livros são feitos em planilhas e não há nenhum sistema para consulta pública do acervo. Isso dificulta a organização e impede os alunos de saberem quais livros estão disponíveis.</p> <p>Como parte da equipe de TI da escola, você recebeu a missão de <b>criar um sistema web simples</b> para cadastrar os livros da biblioteca e suas respectivas categorias, permitindo também o acesso às informações via navegador e via API para futuras integrações (como um app mobile).</p>	
<p> <b>Missão do aluno</b></p> <p>Você deverá desenvolver uma aplicação web utilizando <b>Django e Django REST Framework</b>, com as seguintes funcionalidades:</p> <ul style="list-style-type: none"><li>• Cadastrar categorias de livros (Romance, Técnico, Aventura etc.)</li><li>• Cadastrar livros com título, autor, ano e a categoria correspondente</li><li>• Listar todos os livros com seus detalhes e categoria associada</li><li>• Disponibilizar esses dados também via API JSON (GET e POST)</li></ul>	
<p> <b>Requisitos Técnicos</b></p> <ul style="list-style-type: none"><li>• Ambiente virtual configurado</li><li>• Projeto Django com app catalogo</li><li>• Modelos com relacionamento ForeignKey</li><li>• Serializers com campo categoria_id</li><li>• ViewSets com operações de CRUD</li><li>• URLs usando routers do DRF</li><li>• Respostas JSON estruturadas</li></ul>	
<p> <b>Desafio Proposto</b></p> <p>Você deverá entregar:</p> <ul style="list-style-type: none"><li>• O código-fonte funcionando localmente (runserver)</li><li>• O acesso à API via navegador com rotas:<ul style="list-style-type: none"><li>◦ /api/categorias/</li><li>◦ /api/livros/</li></ul></li><li>• A documentação mínima (em texto ou vídeo) de como o sistema foi desenvolvido</li></ul>	

### Habilidades Desenvolvidas

- Criação de models e serializers com relacionamentos
- Configuração de views e rotas REST
- Uso de ferramentas profissionais (Django, JSON)
- Organização de projeto com separação de responsabilidades
- Compreensão de como sistemas reais se comunicam por API

### Contexto

Após o sucesso da primeira versão do sistema da biblioteca, a equipe gestora da escola percebeu a necessidade de **controlar o acesso ao sistema** e de **melhorar a experiência de navegação dos usuários**. Agora, eles querem que apenas usuários autenticados possam cadastrar e editar livros e categorias. Além disso, os alunos devem conseguir **buscar livros por título, autor, ano ou categoria**, facilitando a localização das obras.

Você, como técnico em desenvolvimento de sistemas da escola, foi encarregado de evoluir a aplicação para atender essas novas necessidades.

### Missão do aluno

Atualizar o sistema anterior, utilizando Django e Django REST Framework, para incluir:

- **Autenticação JWT (JSON Web Token):**
  - Criar rotas de login e geração de token
  - Restringir o acesso às rotas de criação e edição de livros/categorias
- **Sistema de Filtros:**
  - Permitir que os livros possam ser filtrados por título, autor, ano ou categoria usando parâmetros de consulta (?categoria=Drama&autor=Machado)
- **Manter** as funcionalidades anteriores:
  - Cadastro de categorias e livros com ForeignKey
  - Exibição dos livros com detalhes
  - API JSON acessível para listagem

### Tecnologias e recursos a aplicar

- Django e Django REST Framework
- djangorestframework-simplejwt (para autenticação)
- rest\_framework.filters + DjangoFilterBackend
- URLs públicas e privadas
- Modelos, serializers e viewsets com permissões

### Habilidades desenvolvidas

- Segurança e autenticação via JWT
- Filtragem de dados na API (query parameters)
- Criação de APIs REST seguras e escaláveis
- Organização e controle de acesso a recursos

ATIVIDADE	SOMATIVA – Ordem de Serviço
CONTEXTO e DESCRIÇÃO	
 <b>Cenário</b>	<p>A multinacional <b>BOSCH</b>, reconhecida mundialmente por sua atuação nas áreas de tecnologia e serviços industriais, está enfrentando um problema na <b>gestão de ordens de serviço internas</b> nas suas unidades de produção e laboratórios técnicos.</p> <p>Atualmente, os pedidos de manutenção de equipamentos, solicitações de reparo em ambientes e gestão de responsáveis por patrimônios estão sendo feitas por <b>planilhas e e-mails</b>, gerando atrasos, falta de controle e perda de informações importantes.</p> <p>Por isso, a BOSCH decidiu <b>contratar uma equipe jovem e criativa de desenvolvedores juniores</b>, composta por <b>alunos do curso técnico em Desenvolvimento de Sistemas</b>, para projetar e desenvolver um <b>sistema de gestão de ordens de serviço 100% online</b>, moderno, seguro e eficiente.</p>
 <b>Desafio Proposto</b>	<p>Você e sua equipe foram contratados para desenvolver esse sistema. O objetivo é criar uma aplicação web utilizando:</p> <ul style="list-style-type: none"><li>• <b>Back end:</b> Django Rest Framework com <b>autenticação via JWT</b></li><li>• <b>Front end:</b> Angular, com uso de <b>HttpClient</b> para consumir a API</li><li>• Banco de dados: SQLite ou PostgreSQL (a critério da equipe)</li><li>• Integração com planilhas para popular o banco</li></ul>
 <b>Funcionalidades Esperadas</b>	<p><b>1. Autenticação e Controle de Acesso</b></p> <ul style="list-style-type: none"><li>• Login, logout e cadastro de usuários com JWT</li><li>• Níveis de permissão: usuários comuns, manutentores, gestores e administradores</li><li>• Proteção de rotas e páginas no front end com base na permissão</li></ul> <p><b>2. Módulos do Sistema (Tabelas)</b></p> <ul style="list-style-type: none"><li>• <b>Ordem de Serviço</b><ul style="list-style-type: none"><li>✓ <b>descrição:</b> descrição do problema</li><li>✓ <b>abertura:</b> data e hora da abertura da OS</li><li>✓ <b>fechamento:</b> data e hora do fechamento da OS</li><li>✓ <b>status:</b> serão 4 status (<i>iniciada, pendente, finalizada, cancelada</i>)</li><li>✓ <b>prioridade:</b> serão 3 prioridades (<i>alta, média e baixa</i>)</li><li>✓ <b>ambiente:</b> número do ambiente que virá da tabela ambientes</li><li>✓ <b>patrimônio:</b> 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</li></ul></li></ul>

- ✓ **manutentor:** número do manutentor que virá da tabela manutentores
- ✓ **funcionário:** nome do funcionário que abriu a ordem. Capture o nome do usuário que logou
- **Patrimônios**
  - ✓ **ni:** número do equipamento, quando houver
  - ✓ **localização:** código numérico do ambiente ex. 20400001
  - ✓ **descrição:** detalhes sobre o patrimônio
- **Ambientes**
  - ✓ **sig:** código do ambiente, exemplo: 20400024
  - ✓ **descricao:** exemplo: LAB. DE INFORMÁTICA A04
  - ✓ **sn\_resp:** identificação do funcionário, exemplo: SN1085371
  - ✓ **responsavel:** 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:** ex: Operador de Práticas Profissionais, Coordenador Técnico, Coordenador Pedagógico
- **Áreas**
  - ✓ **area:** área de atuação, exemplo: elétrica, informática, alvenaria, pintura, mecânica etc

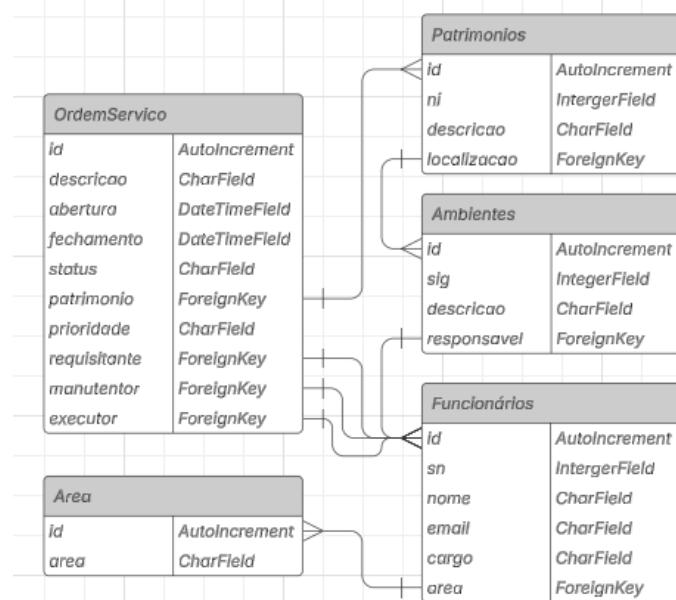


Figura 1 - Diagrama de Entidade e Relacionamento do Projeto

**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 Angular 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.

**7. Requisitos Técnicos**

- CRUD completo (exceto Ordem de Serviço, que só poderá ser criada)
- Interface web com formulários, listagens e filtros
- Página inicial com menu de navegação
- Upload de planilhas para popular tabelas (ex: patrimônios)
- Filtros por status e prioridade nas OS
- Atualização de status da OS
- Histórico de OS com paginação

**8. Critérios de Qualidade**

- Código limpo e modular
- Componentes reutilizáveis no Angular (ex: header, footer)
- Feedback ao usuário em caso de erro
- Responsividade da interface
- Uso adequado de hooks e services

## **9. Pesquisas**

- Localizar por ID de sensor.
- Localizar por tipo de sensor, exemplo “temperatura”.
- Localizar por data de sensor.
- Localizar por código “sig” do ambiente.
- Localizar por ID do Histórico.
- Filtro duplo por data e por sensor
- Filtro triplo por data e por sensor e hora

## **10. População de Banco de Dados**

- Implementação de **exportação** de relatórios em Excel (XLSX ou CSV)

## **11. Exportação de Banco de dados**

- Implementação de **exportação** de relatórios em Excel (XLSX ou CSV)

## **12. Metodologias Ágeis**

- De acordo com o prazo proposto, divida-o em sprints e faça um documento explicando o que seria entregue em cada uma.
- Crie um documento e acrescente dados artificiais de: Product Owner, Scrum Master, Sprints(datas), Sprint Planning e Sprint Retrospective.

## **13. 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.

**14. Entrega**

O aluno deverá criar repositório no Github privado e dar acesso ao professor (lindomarbatistao@gmail.com), enviando um convite como colaborador.

Crie um arquivo README.md com todos os endpoints do problema de forma que seja somente copiar e colar no INSOMNIA para testar além do usuário e senha do banco de dados. Esse arquivo deverá estar dentro do repositório.

O nome do repositório deverá ser o nome e sobrenome do aluno separados por underline. Dentro deverá ter 2 pastas: back e front, sendo que na back deveram ser inseridos todos os arquivos e pastas do backend e na front todos do frontend.

 **Objetivo Final**

Criar um sistema funcional de **gestão de ordens de serviço** para a BOSCH, que resolva os seguintes problemas:

- **Agilidade** no atendimento das solicitações
- **Rastreamento** de histórico de serviços realizados
- **Gestão centralizada** de ambientes, patrimônios e responsáveis

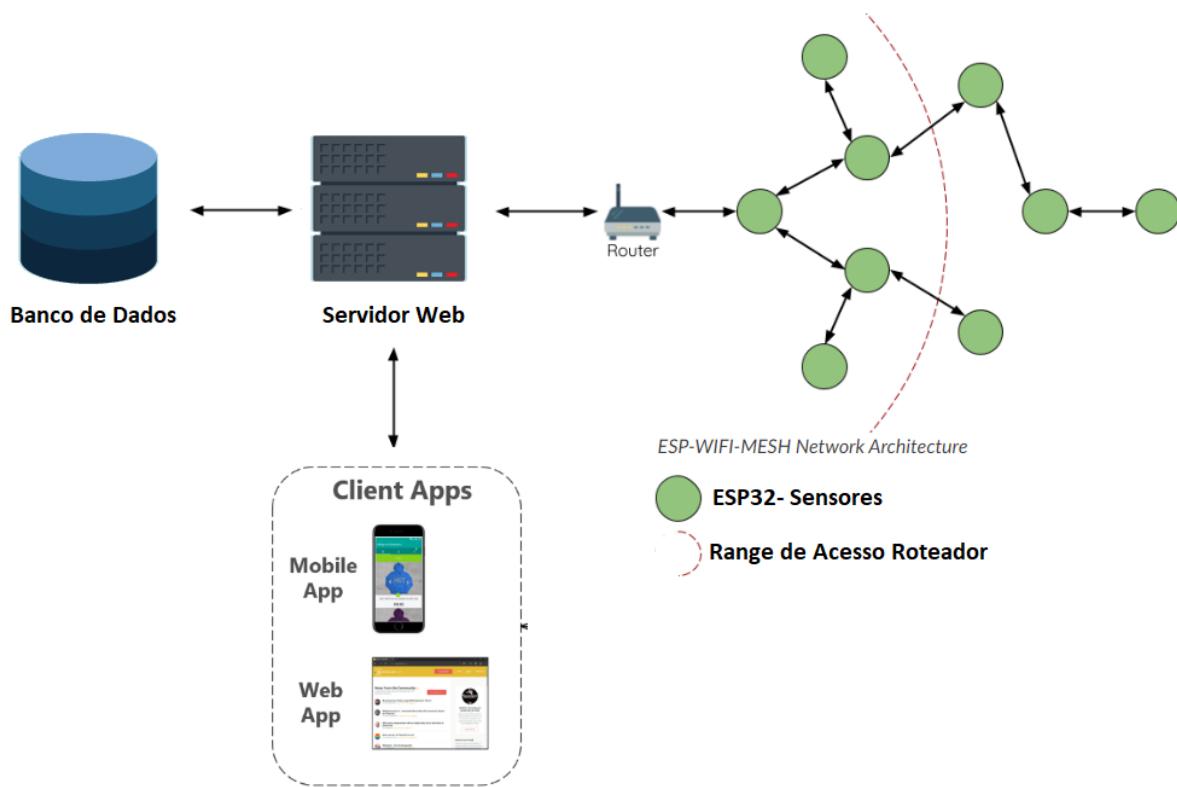
ATIVIDADE	PROJETO INTEGRADOR - SITUAÇÃO PROBLEMA CONTEXTO e DESCRIÇÃO
<p>Imagine-se como exploradores de um mundo digital inexplorado! Este projeto não é apenas uma simulação de uma Cidade Inteligente; é uma jornada fascinante em que transformaremos a nossa "Escola e Faculdade de Tecnologia Senai Roberto Mange" em um ecossistema inovador, onde cada sala, corredor e laboratório desempenha um papel crucial nessa Cidade do Futuro.</p>  <p>imagem ilustrativa: <a href="https://blog.intelbras.com.br/smart-building/">https://blog.intelbras.com.br/smart-building/</a></p> <p><b>Exploração Tecnológica:</b></p> <p>Para desbravar esse território inexplorado, iremos utilizar tecnologias de ponta. Sensores, redes WiFi, bancos de dados e aplicativos web e mobile serão as ferramentas em nossas mãos enquanto construímos uma experiência com vocês, futuros desenvolvedores de sistemas.</p> <p><b>Desafios e Metodologias Ágeis:</b></p> <p>Enfrentaremos desafios reais ao coletar dados de sensores ESP32 e integrá-los em uma plataforma coesa. Utilizaremos metodologias ágeis, sprints dinâmicas e colaboração contínua para adaptar-nos à medida que avançamos, garantindo que cada passo seja um avanço audacioso na nossa jornada.</p>  <p>imagem ilustrativa: <a href="https://troposlab.com">https://troposlab.com</a></p>	

**Conectando a Cidade: Sensores e Dispositivos ESP32:**

Nosso ponto de partida será instalar e configurar sensores de temperatura e umidade, transformando cada sala em um ponto vital na nossa cidade. Os ESP32, pequenos heróis dessa história, coletarão dados em tempo real, enviando-os para um servidor central, trazendo vida à nossa cidade digital.

**Teia de Conectividade: Rede WiFi e Servidor Web:**

Conectar o ESP32 à uma rede WiFi própria da Cidade Inteligente será como tecer uma teia que conecta cada ambiente. Os dados coletados serão transmitidos para um servidor web central, onde nossa base de dados ganhará vida, registrando as nuances do ambiente escolar em tempo real.

**Comando Central: APIs e Aplicação Web para Administrador:**

Nossa jornada ganha um comando central com a criação de uma aplicação web para o administrador. Em sprints ágeis, implementaremos funcionalidades essenciais, desde o cadastro de usuários até a exibição do mapa da cidade com a localização dos sensores, proporcionando uma visão panorâmica da temperatura e umidade em cada ambiente.

**Explorando Além do Conhecido: Funcionalidades Adicionais:**

Nossos exploradores não se limitarão apenas a nossa sala de aula. Utilizaremos dados não apenas do nosso sensor, mas também de outros 3 sensores distribuídos em pontos estratégicos da cidade (escola). A aplicação responsiva exibirá o mapa da cidade, indicando os sensores e as últimas leituras, enquanto gráficos comparativos oferecerão insights visuais sobre os ambientes medidos.

**Exploradores Móveis: Aplicativo Mobile:**

Os exploradores também carregarão consigo um dispositivo móvel - nosso aplicativo. Em sprints empolgantes, adicionaremos camadas de autenticação, localização e visualização de dados, permitindo que cada explorador identifique seu próprio local na cidade, visualize os sensores ao redor e receba alertas vibrantes ou sonoros quando algo extraordinário acontecer.

**Registro de Nossa Jornada: Observações:**

Cada passo será documentado, não apenas para cumprir uma formalidade, mas para criar um legado. Boas práticas de programação, colaboração intensa, testes contínuos e revisões regulares garantirão que nossa jornada seja sólida, adaptável e deixe um impacto duradouro.

Este não é apenas um projeto; é uma aventura rumo ao desconhecido, uma busca por conhecimento, uma jornada que moldará o futuro tecnológico da nossa escola e, quem sabe, do mundo lá fora!

ATIVIDADE	PROJETO INTEGRADOR (PWBE)
<b>CONTEXTO e DESCRIÇÃO</b>	
<p>A escola <b>TecnoVille</b> está desenvolvendo um projeto de transformação urbana baseado no conceito de <b>Smart City</b>. A ideia é implementar sensores em pontos estratégicos da cidade para coletar dados em tempo real sobre:</p> <ul style="list-style-type: none"> <li>•  Temperatura</li> <li>•  Umidade</li> <li>•  Luminosidade</li> <li>•  Contador</li> </ul> <p>Esses sensores serão instalados em locais como praças, corredores, pátios etc.</p> <p>A gestão escolar contratou sua equipe técnica para desenvolver <b>somente o back end do sistema de monitoramento</b> usando <b>Django e Django Rest Framework</b>. Os dados dos sensores serão enviados para a API e posteriormente visualizados por um painel web (a ser feito por outra equipe).</p>	
<p> <b>Desafio proposto</b></p> <p>Vocês devem <b>modelar e desenvolver o back end completo da aplicação</b>, com base nos seguintes requisitos:</p>	<p><input checked="" type="checkbox"/> <b>Requisitos Funcionais</b></p> <ul style="list-style-type: none"> <li>• O sistema deve permitir <b>registrar sensores</b>, com campos como: <ul style="list-style-type: none"> <li>◦ ID do sensor</li> <li>◦ Sensor (temperatura, umidade, luminosidade, contador)</li> <li>◦ Identificação (mac-address)</li> <li>◦ Localização (latitude e longitude)</li> <li>◦ Status operacional (ativo/inativo)</li> </ul> </li> <li>• Deve ser possível <b>registrar medições</b> vindas dos sensores: <ul style="list-style-type: none"> <li>◦ ID do sensor relacionado</li> <li>◦ Ambiente (relacionado com a tabela de ambientes)</li> <li>◦ Valor da medição</li> <li>◦ Data e hora da leitura (timestamp)</li> </ul> </li> <li>• Deve se registrar os <b>ambientes</b> da escola: <ul style="list-style-type: none"> <li>◦ Sig (código do ambiente)</li> <li>◦ Descrição</li> <li>◦ Ni (número de identificação do responsável pelo ambiente)</li> <li>◦ Responsável</li> </ul> </li> <li>• A API deve fornecer endpoints para: <ul style="list-style-type: none"> <li>◦ Cadastrar, editar e listar sensores</li> <li>◦ Cadastrar e listar medições por sensor</li> <li>◦ Listar as medições mais recentes (últimas 24h, por exemplo)</li> </ul> </li> </ul>

**Requisitos do Projeto:**

**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.

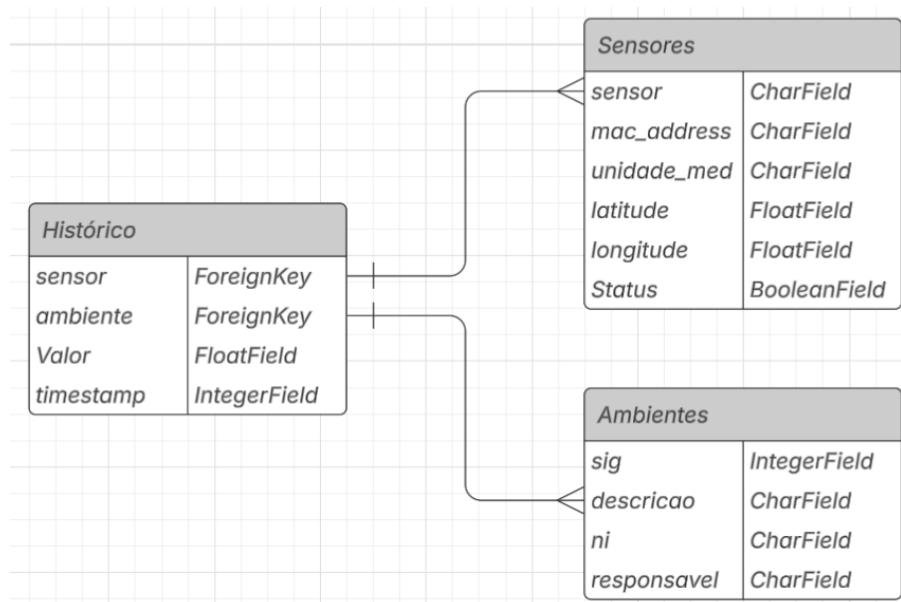


Figura 2 - Diagrama de Entidade e Relacionamento

- **Login**

Criar um super usuário para o nosso api\_smart.

- username = lin
- password = 123

- **Relacionamento entre tabelas**

- Os relacionamentos deverão ser aplicados nas tabelas conforme diagrama já mencionado acima.

- **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).

- **Dados:**
  - Criar método para capturar dados de sensores e ambientes que estão nas planilhas disponibilizadas.
  - Os dados poderão ser exportados no formato de planilhas.
- **Integração entre Front End e Back End:**
  - Utilizar **Axios** no Angular 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:**

- **Desenvolvimento do Back-End:**
  - **Histórias de Usuário:**
    - Como administrador, eu quero criar um endpoint para registrar dados de sensores, para que eu possa armazenar os dados de temperatura, luminosidade e umidade.
    - Como administrador, eu quero criar um endpoint para visualizar os dados dos sensores, para que eu possa monitorar as condições ambientais.
    - Como administrador, eu quero implementar autenticação JWT, para garantir que apenas usuários autorizados accessem 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.
- **Desenvolvimento do Front-End:**
  - **Histórias de Usuário:**
    - Como supervisor, eu quero visualizar os dados dos sensores em uma lista, para que eu possa monitorar as condições em tempo real.
    - Como supervisor, eu quero visualizar gráficos dos dados dos sensores, para analisar as variações ao longo do tempo.
    - Como supervisor, eu quero realizar login na aplicação, para acessar os dados de forma segura.
  - **Tarefas:**
    - Configurar projeto Angular.
    - 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.
- **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 Angular).
- 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.

## ESTRATÉGIAS DE ENSINO, APRENDIZAGEM E AVALIAÇÃO

Nº horas / aulas	Capacidades a serem trabalhadas	Conhecimentos relacionados	Estratégias de ensino e aprendizagem e instrumentos de avaliação	Recursos e ambientes pedagógicos
10	<p><b>Técnicas:</b></p> <ul style="list-style-type: none"> <li>1. Utilizar o paradigma da programação orientada a objetos</li> <li>4. Identificar as características de programação back-end em ambiente web</li> <li>5. Preparar o ambiente necessário ao desenvolvimento back-end para a plataforma web</li> <li>6. Definir os elementos de entrada, processamento e saída para a programação da aplicação web</li> </ul> <p><b>Socioemocionais:</b></p> <ul style="list-style-type: none"> <li>1. Demonstrar autogestão</li> <li>2. Demonstrar pensamento analítico</li> <li>4. Demonstrar autonomia</li> </ul>	<ul style="list-style-type: none"> <li>1. Ambiente de desenvolvimento web</li> <li>1.1. Definição</li> <li>1.2. Histórico</li> <li>1.3. Características</li> <li>1.4. Ambiente de desenvolvimento</li> <li>1.4.1. Instalação e configuração</li> <li>1.4.2. Recursos e interfaces</li> <li>1.4.3. Gerenciamento de dependências</li> </ul>	<p><b>Aulas Expositivas:</b> Fornecer uma visão geral teórica sobre os tópicos, incluindo definições, histórico, e características do ambiente de desenvolvimento web.</p> <p><b>Demonstrações Práticas:</b> Mostrar passo a passo a instalação e configuração do ambiente de desenvolvimento, bem como o uso de recursos e interfaces.</p> <p><b>Aulas Práticas em Laboratório:</b> Permitir que os alunos configurem e gerenciem seus próprios ambientes de desenvolvimento sob supervisão.</p> <p><b>Tutoriais e Guias:</b> Disponibilizar materiais didáticos, como tutoriais e guias passo a passo, para que os alunos possam seguir no seu próprio ritmo.</p> <p><b>Projetos em Grupo:</b> Incentivar a colaboração em projetos práticos onde os alunos possam aplicar o que aprenderam em um cenário real.</p> <p><b>Sessões de Perguntas e Respostas:</b> Facilitar sessões onde os alunos possam fazer perguntas e esclarecer dúvidas.</p> <p><b>Feedback Contínuo:</b> Oferecer feedback contínuo sobre o progresso dos alunos em suas tarefas e projetos.</p>	<ul style="list-style-type: none"> <li>• Laboratório</li> <li>• Projetor multimídia</li> <li>• Notebook</li> <li>• Computador</li> </ul>

## ESTRATÉGIAS DE ENSINO, APRENDIZAGEM E AVALIAÇÃO

Nº horas / aulas	Capacidades a serem trabalhadas	Conhecimentos relacionados	Estratégias de ensino e aprendizagem e instrumentos de avaliação	Recursos e ambientes pedagógicos
10	<p><b>Técnicas:</b></p> <p>4. Identificar as características de programação back-end em ambiente web</p> <p>6. Definir os elementos de entrada, processamento e saída para a programação da aplicação web</p> <p>8. Definir os frameworks a serem utilizados no desenvolvimento da aplicação web</p> <p>9. Utilizar interações com base de dados para desenvolvimento de sistemas web</p> <p>11. Estabelecer envio de notificações entre cliente e servidor por meio de aplicação web</p> <p>12. Desenvolver API (web services) para integração de dados entre plataformas</p> <p><b>Socioemocionais:</b></p> <p>2. Demonstrar pensamento analítico</p> <p>3. Demonstrar inteligência emocional</p> <p>4. Demonstrar autonomia</p>	<p>2. Web Services</p> <p>2.1. Definição</p> <p>2.2. REST</p> <p>2.2.1. Recursos</p> <p>2.2.2. Semântica da URL REST</p> <p>2.3. Padrão JSON</p> <p>2.3.1. Sintaxe básica</p> <p>2.3.2. Tipos de dados</p> <p>2.3.3. Formatação</p> <p>2.3.4. Coleção de objetos JSON</p> <p>2.4.1. Sintaxe básica</p> <p>2.4.2. Tipos de dados</p> <p>2.4.3. Formatação</p>	<p><b>Aulas Expositivas:</b> Apresentar conceitos teóricos fundamentais sobre Web Services, REST, JSON e XML.</p> <p><b>Demonstrações Práticas:</b> Mostrar exemplos práticos e funcionando de Web Services em ação, incluindo chamadas de APIs e análise de respostas.</p> <p><b>Laboratórios Práticos:</b> Realizar atividades onde os alunos criam e testam seus próprios Web Services, consomem APIs, e manipulam dados em JSON e XML.</p> <p><b>Tutoriais e Guias:</b> Fornecer tutoriais e guias detalhados para os alunos praticarem em casa.</p> <p><b>Estudo de Casos:</b> Analisar exemplos reais e casos de uso de Web Services em diferentes contextos.</p> <p><b>Discussões em Grupo:</b> Facilitar discussões e debates sobre melhores práticas e padrões de design para Web Services.</p> <p><b>Feedback Contínuo:</b> Oferecer feedback regular sobre as atividades práticas e projetos dos alunos.</p>	<ul style="list-style-type: none"> <li>• Laboratório</li> <li>• Projetor multimídia</li> <li>• Notebook</li> <li>• Computador</li> </ul>

## ESTRATÉGIAS DE ENSINO, APRENDIZAGEM E AVALIAÇÃO

Nº horas / aulas	Capacidades a serem trabalhadas	Conhecimentos relacionados	Estratégias de ensino e aprendizagem e instrumentos de avaliação	Recursos e ambientes pedagógicos
10	<p><b>Técnicas:</b></p> <p>4. Identificar as características de programação back-end em ambiente web</p> <p>6. Definir os elementos de entrada, processamento e saída para a programação da aplicação web</p> <p>11. Estabelecer envio de notificações entre cliente e servidor por meio de aplicação web</p> <p>12. Desenvolver API (web services) para integração de dados entre plataformas</p> <p><b>Socioemocionais:</b></p> <p>2. Demonstrar pensamento analítico</p> <p>3. Demonstrar inteligência emocional</p> <p>4. Demonstrar autonomia</p>	<p>3. Protocolo HTTP</p> <p>3.1. Definição</p> <p>3.2. Métodos HTTP</p> <p>3.2.1. GET</p> <p>3.2.2. POST</p> <p>3.2.3. PUT</p> <p>3.2.4. DELETE</p>	<p><b>Aulas Expositivas:</b> Introduzir conceitos teóricos sobre o Protocolo HTTP e seus componentes.</p> <p><b>Demonstrações Práticas:</b> Mostrar exemplos de como os métodos HTTP são usados em APIs e como os cabeçalhos e parâmetros são manipulados.</p> <p><b>Laboratórios Práticos:</b> Proporcionar atividades onde os alunos implementem e testem requisições HTTP utilizando ferramentas como Postman e cURL.</p> <p><b>Tutoriais e Guias:</b> Fornecer materiais de leitura e tutoriais para prática individual sobre o uso dos métodos HTTP, parâmetros, cabeçalhos e códigos de status.</p> <p><b>Estudos de Caso:</b> Analisar exemplos reais de APIs para entender como o protocolo HTTP é utilizado em diferentes cenários.</p> <p><b>Discussões em Grupo:</b> Facilitar discussões sobre melhores práticas e padrões para o uso dos diferentes componentes do HTTP.</p> <p><b>Feedback Contínuo:</b> Oferecer feedback sobre as atividades práticas e projetos dos alunos.</p>	<ul style="list-style-type: none"> <li>• Laboratório</li> <li>• Projetor multimídia</li> <li>• Notebook</li> <li>• Computador</li> </ul>

## ESTRATÉGIAS DE ENSINO, APRENDIZAGEM E AVALIAÇÃO

Nº horas / aulas	Capacidades a serem trabalhadas	Conhecimentos relacionados	Estratégias de ensino e aprendizagem e instrumentos de avaliação	Recursos e ambientes pedagógicos
10	<p><b>Técnicas:</b></p> <ul style="list-style-type: none"> <li>1. Utilizar o paradigma da programação orientada a objetos</li> <li>3. Aplicar técnicas de código limpo (clean code)</li> <li>6. Definir os elementos de entrada, processamento e saída para a programação da aplicação web</li> <li>13. Desenvolver sistemas web de acordo com as regras de negócio estabelecidas</li> </ul> <p><b>Socioemocionais:</b></p> <ul style="list-style-type: none"> <li>2. Demonstrar pensamento analítico</li> <li>3. Demonstrar inteligência emocional</li> <li>4. Demonstrar autonomia</li> </ul>	4. Programação orientada a objetos <ul style="list-style-type: none"> <li>4.1. Definição</li> <li>4.2. Pacotes</li> <li>4.3. Classes           <ul style="list-style-type: none"> <li>4.3.1. Abstrata</li> <li>4.3.2. Interna</li> <li>4.3.3. Anônima</li> <li>4.3.4. Atributos</li> <li>4.3.5. Métodos</li> </ul> </li> <li>4.3.6. Modificadores de acesso (encapsulamento)</li> <li>4.4. Objetos</li> <li>4.5. Interface</li> <li>4.6. Polimorfismo</li> <li>4.7. Enumerações</li> <li>4.8. Relacionamentos de objetos           <ul style="list-style-type: none"> <li>4.8.1. Herança</li> <li>4.8.2. Agregação</li> <li>4.8.3. Composição</li> </ul> </li> </ul>	<p><b>Aulas Expositivas:</b> Introduzir os conceitos teóricos de POO com definições, exemplos e explicações detalhadas.</p> <p><b>Demonstrações Práticas:</b> Mostrar como implementar conceitos de POO em código, com exemplos práticos e demonstrações ao vivo.</p> <p><b>Laboratórios Práticos:</b> Proporcionar atividades práticas onde os alunos possam implementar classes, objetos, e outros conceitos de POO em projetos reais.</p> <p><b>Tutoriais e Guias:</b> Fornecer materiais de leitura e tutoriais passo a passo para que os alunos pratiquem os conceitos de POO em projetos individuais.</p> <p><b>Estudos de Caso:</b> Analisar exemplos de código real para entender como a POO é aplicada em projetos de software.</p> <p><b>Discussões em Grupo:</b> Facilitar discussões sobre design orientado a objetos e melhores práticas de implementação.</p> <p><b>Feedback Contínuo:</b> Oferecer feedback sobre a implementação dos conceitos de POO.</p>	<p>• Laboratório</p> <p>• Projetor multimídia</p> <p>• Notebook</p> <p>• Computador</p>

## ESTRATÉGIAS DE ENSINO, APRENDIZAGEM E AVALIAÇÃO

Nº horas / aulas	Capacidades a serem trabalhadas	Conhecimentos relacionados	Estratégias de ensino e aprendizagem e instrumentos de avaliação	Recursos e ambientes pedagógicos
10	<p><b>Técnicas:</b></p> <p>3. Aplicar técnicas de código limpo (clean code)</p> <p>12. Desenvolver API (web services) para integração de dados entre plataformas</p> <p>13. Desenvolver sistemas web de acordo com as regras de negócio estabelecidas</p> <p><b>Socioemocionais:</b></p> <p>1. Demonstrar autogestão</p> <p>2. Demonstrar pensamento analítico</p>	<p>5. Documentação</p> <p>5.1. Diagrama de classes</p> <p>5.2. Documentação de API</p> <p>5.2.1. Swagger</p> <p>5.2.2. Postman</p> <p>5.2.3. Insomnia</p>	<p><b>Aulas Expositivas:</b> Introduzir conceitos teóricos sobre a importância da documentação e os diferentes tipos de documentação para APIs e diagramas de classes.</p> <p><b>Demonstrações Práticas:</b> Mostrar como criar e manter documentação usando ferramentas específicas como Swagger, Postman e Insomnia.</p> <p><b>Laboratórios Práticos:</b> Proporcionar atividades práticas onde os alunos documentam APIs e criam diagramas de classes.</p> <p><b>Tutoriais e Guias:</b> Fornecer tutoriais e guias para que os alunos pratiquem a criação de documentação e uso de ferramentas de documentação.</p> <p><b>Estudos de Caso:</b> Analisar exemplos reais de documentação de APIs e diagramas de classes para entender melhores práticas e padrões.</p> <p><b>Discussões em Grupo:</b> Facilitar discussões sobre melhores práticas de documentação e como diferentes ferramentas podem ser utilizadas.</p>	<ul style="list-style-type: none"> <li>• Laboratório</li> <li>• Projetor multimídia</li> <li>• Notebook</li> <li>• Computador</li> </ul>

## ESTRATÉGIAS DE ENSINO, APRENDIZAGEM E AVALIAÇÃO

Nº horas / aulas	Capacidades a serem trabalhadas	Conhecimentos relacionados	Estratégias de ensino e aprendizagem e instrumentos de avaliação	Recursos e ambientes pedagógicos
0	<p><b>Técnicas:</b></p> <ul style="list-style-type: none"> <li>4. Identificar as características de programação back-end em ambiente web</li> <li>6. Definir os elementos de entrada, processamento e saída para a programação da aplicação web</li> <li>8. Definir os frameworks a serem utilizados no desenvolvimento da aplicação web</li> <li>12. Desenvolver API (web services) para integração de dados entre plataformas</li> <li>13. Desenvolver sistemas web de acordo com as regras de negócio estabelecidas</li> </ul> <p><b>Socioemocionais:</b></p> <ul style="list-style-type: none"> <li>2. Demonstrar pensamento analítico</li> <li>3. Demonstrar inteligência emocional</li> <li>4. Demonstrar autonomia</li> </ul>	<ul style="list-style-type: none"> <li>6. Padrão de desenvolvimento MVC</li> <li>6.1. Definição</li> <li>6.2. Aplicabilidade</li> <li>6.3. Design patterns</li> </ul>	<p><b>Aulas Expositivas:</b> Introduzir conceitos teóricos sobre o padrão MVC, sua definição, aplicabilidade e como ele se integra com outros design patterns.</p> <p><b>Demonstrações Práticas:</b> Mostrar exemplos práticos de implementação do padrão MVC utilizando Django, destacando a separação de responsabilidades entre o modelo, a visão e o controlador.</p> <p><b>Laboratórios Práticos:</b> Proporcionar atividades práticas onde os alunos possam implementar o padrão MVC em projetos de API, aplicando os conceitos aprendidos.</p> <p><b>Tutoriais e Guias:</b> Fornecer tutoriais e guias detalhados sobre a implementação do padrão MVC, incluindo exemplos e melhores práticas.</p> <p><b>Estudos de Caso:</b> Analisar exemplos reais de uso do padrão MVC para entender como ele é aplicado em projetos reais e os benefícios que oferece.</p> <p><b>Discussões em Grupo:</b> Facilitar discussões sobre a aplicabilidade do padrão MVC em diferentes contextos e como ele se relaciona com outros design patterns.</p>	<ul style="list-style-type: none"> <li>• Laboratório</li> <li>• Projetor multimídia</li> <li>• Notebook</li> <li>• Computador</li> </ul>

## ESTRATÉGIAS DE ENSINO, APRENDIZAGEM E AVALIAÇÃO

Nº horas / aulas	Capacidades a serem trabalhadas	Conhecimentos relacionados	Estratégias de ensino e aprendizagem e instrumentos de avaliação	Recursos e ambientes pedagógicos
10	<p><b>Técnicas:</b></p> <p>4. Identificar as características de programação back-end em ambiente web</p> <p>5. Preparar o ambiente necessário ao desenvolvimento back-end para a plataforma web</p> <p>6. Definir os elementos de entrada, processamento e saída para a programação da aplicação web</p> <p>8. Definir os frameworks a serem utilizados no desenvolvimento da aplicação web</p> <p>13. Desenvolver sistemas web de acordo com as regras de negócio estabelecidas</p> <p>14. Publicar a aplicação web</p> <p><b>Socioemocionais:</b></p> <p>1. Demonstrar autogestão</p> <p>2. Demonstrar pensamento analítico</p> <p>4. Demonstrar autonomia</p>	<p>7. Frameworks</p> <p>7.1. Definição</p> <p>7.2. Modelos e tipos</p> <p>7.3. Instalação e configuração</p> <p>7.4. Criação de projetos utilizando framework</p>	<p><b>Aulas Expositivas:</b> Introduzir conceitos teóricos sobre o que são frameworks, os diferentes modelos e tipos, e a importância da instalação e configuração adequadas.</p> <p><b>Demonstrações Práticas:</b> Mostrar como instalar e configurar frameworks e criar projetos utilizando esses frameworks.</p> <p><b>Laboratórios Práticos:</b> Oferecer atividades práticas onde os alunos possam instalar frameworks, configurar ambientes e criar projetos.</p> <p><b>Tutoriais e Guias:</b> Fornecer materiais de leitura e tutoriais passo a passo sobre frameworks, incluindo exemplos de instalação e configuração.</p> <p><b>Estudos de Caso:</b> Analisar exemplos reais de uso de frameworks em projetos para entender melhores práticas e padrões.</p> <p><b>Discussões em Grupo:</b> Facilitar discussões sobre as vantagens e desvantagens de diferentes frameworks e como escolher o framework adequado para diferentes tipos de projetos.</p>	<ul style="list-style-type: none"> <li>• Laboratório</li> <li>• Projetor multimídia</li> <li>• Notebook</li> <li>• Computador</li> </ul>

## ESTRATÉGIAS DE ENSINO, APRENDIZAGEM E AVALIAÇÃO

Nº horas / aulas	Capacidades a serem trabalhadas	Conhecimentos relacionados	Estratégias de ensino e aprendizagem e instrumentos de avaliação	Recursos e ambientes pedagógicos
10	<b>Técnicas</b> <p>9. Utilizar interações com base de dados para desenvolvimento de sistemas web</p> <p>12. Desenvolver API (web services) para integração de dados entre plataformas</p> <b>Socioemocionais</b> <p>1. Demonstrar autogestão</p> <p>2. Demonstrar pensamento analítico</p> <p>3. Demonstrar inteligência emocional</p> <p>4. Demonstrar autonomia</p>	8. Persistência de dados <p>8.1. Conexão com base de dados</p> <p>8.2. CRUD</p> <p>8.3. Transferência de arquivos locais para ambiente servidor</p> <p>8.4. Geração de relatórios</p> <p>8.5. Manipulação de dados utilizando XML</p> <p>8.6. Manipulação de dados utilizando JSON</p> <p>8.7. Banco de dados de memória</p> <p>8.8. Versionamento do banco de dados</p>	<p><b>Aulas Expositivas:</b> Introduzir conceitos teóricos sobre persistência de dados, conexão com bases de dados, operações CRUD, e manipulação de arquivos e dados em diferentes formatos.</p> <p><b>Demonstrações Práticas:</b> Mostrar como realizar operações de CRUD, conectar-se a bancos de dados, transferir arquivos e gerar relatórios na prática.</p> <p><b>Laboratórios Práticos:</b> Proporcionar atividades práticas onde os alunos possam executar operações de CRUD, manipular dados em XML e JSON, e trabalhar com bancos de dados em memória.</p> <p><b>Tutoriais e Guias:</b> Fornecer tutoriais e guias detalhados sobre como realizar tarefas específicas, como conexão com bases de dados, transferência de arquivos, e geração de relatórios.</p> <p><b>Estudos de Caso:</b> Analisar exemplos reais de persistência de dados para entender como os conceitos são aplicados em projetos reais.</p> <p><b>Discussões em Grupo:</b> Facilitar discussões sobre melhores práticas para a persistência de dados, incluindo gerenciamento de dados e desafios comuns.</p>	<ul style="list-style-type: none"> <li>• Laboratório</li> <li>• Projetor multimídia</li> <li>• Notebook</li> <li>• Computador</li> </ul>

## ESTRATÉGIAS DE ENSINO, APRENDIZAGEM E AVALIAÇÃO

Nº horas / aulas	Capacidades a serem trabalhadas	Conhecimentos relacionados	Estratégias de ensino e aprendizagem e instrumentos de avaliação	Recursos e ambientes pedagógicos
10	<p><b>Técnicas</b></p> <p>5. Preparar o ambiente necessário ao desenvolvimento back-end para a plataforma web</p> <p>8. Definir os frameworks a serem utilizados no desenvolvimento da aplicação web</p> <p>13. Desenvolver sistemas web de acordo com as regras de negócio estabelecidas</p> <p>14. Publicar a aplicação web</p> <p><b>Socioemocionais</b></p> <p>1. Demonstrar autogestão</p> <p>2. Demonstrar pensamento analítico</p> <p>3. Demonstrar inteligência emocional</p> <p>4. Demonstrar autonomia</p>	9. Publicação da aplicação desenvolvida	<p><b>Aulas Expositivas:</b> Introduzir conceitos teóricos sobre a publicação de aplicações, incluindo opções de hospedagem, configuração de servidores, segurança e monitoramento.</p> <p><b>Demonstrações Práticas:</b> Mostrar passo a passo como configurar servidores, serviços de hospedagem, e implementar medidas de segurança e monitoramento.</p> <p><b>Laboratórios Práticos:</b> Oferecer atividades práticas onde os alunos configurem e publiquem suas próprias APIs em um ambiente de produção.</p> <p><b>Tutoriais e Guias:</b> Fornecer tutoriais e guias detalhados sobre a publicação de aplicações, incluindo exemplos de configuração e melhores práticas.</p> <p><b>Estudos de Caso:</b> Analisar exemplos reais de publicação de APIs para entender desafios comuns e soluções eficazes.</p> <p><b>Discussões em Grupo:</b> Facilitar discussões sobre melhores práticas para a publicação de aplicações, incluindo gestão de servidores, segurança e monitoramento.</p>	<ul style="list-style-type: none"> <li>• Laboratório</li> <li>• Projetor multimídia</li> <li>• Notebook</li> <li>• Computador</li> </ul>

**INSTRUMENTO DE REGISTRO DE AVALIAÇÃO**

Natureza dos Critérios	Capacidades Técnicas	Critérios de avaliação	Avaliação	
			Aluno	Professor
Competências Técnicas	Utilizar o paradigma da programação orientada a objetos	O aluno implementou uma classe com atributos e métodos, demonstrando a compreensão do conceito de encapsulamento na programação orientada a objetos.		
		O aluno foi capaz de criar hierarquias de classes utilizando herança para promover a reutilização de código e a extensão de funcionalidades.		
		O aluno aplicou o conceito de polimorfismo ao sobrecarregar métodos em diferentes classes, permitindo que o código manipulasse diferentes tipos de objetos de maneira uniforme.		
		O aluno usou interfaces ou classes abstratas para definir contratos que outras classes implementaram, demonstrando a aplicação do princípio da abstração.		
		O aluno projetou e implementou um sistema com classes e objetos que interagem entre si de maneira a seguir os princípios da programação orientada a objetos, como coesão e acoplamento.		
		O aluno foi capaz de utilizar a composição para criar objetos complexos a partir de objetos mais simples, exemplificando o conceito de composição sobre herança quando apropriado.		
	Elaborar diagramas de classe	O aluno criou diagramas de classe detalhados que representam todas as entidades do sistema, suas propriedades e métodos, garantindo uma visão clara da estrutura do software.		
		O aluno identificou e representou corretamente as relações entre as classes, incluindo associações, heranças e composições, no diagrama de classes.		
	Aplicar técnicas de código limpo (clean code)	O aluno escreveu código que utiliza nomes de variáveis e funções descritivos e significativos, facilitando a compreensão do propósito de cada elemento do código.		
		O aluno refatorou trechos de código para remover duplicação, aplicando o princípio DRY (Don't Repeat Yourself) e promovendo a reutilização e manutenção do código.		

<b>Competências Técnicas</b>	Aplicar técnicas de código limpo (clean code)	O aluno aplicou a prática de dividir funções longas em funções menores e mais coesas, melhorando a legibilidade e a testabilidade do código.		
	Aplicar técnicas de código limpo (clean code)	O aluno organizou o código em classes e módulos de maneira que cada um tenha uma única responsabilidade, seguindo o princípio SRP (Single Responsibility Principle).		
		O aluno utilizou comentários e documentação apenas quando necessário, garantindo que o código seja autoexplicativo e reduzindo a necessidade de explicações externas.		
	Identificar as características de programação back-end em ambiente web	O aluno aplicou práticas de segurança no desenvolvimento back-end, incluindo autenticação e autorização		
		O aluno implementou a comunicação entre o servidor e o cliente, demonstrando a capacidade de configurar e gerenciar servidores web no ambiente de desenvolvimento.		
	Preparar o ambiente necessário ao desenvolvimento back-end para a plataforma web	O aluno instalou e configurou ferramentas de desenvolvimento e bibliotecas de back-end, como Django para iniciar o desenvolvimento do sistema.		
		O aluno configurou sistemas de controle de versão, como Git, e integrou o ambiente de desenvolvimento com repositórios para gerenciar e versionar o código do projeto.		
		O aluno configurou ambientes de desenvolvimento local e remoto, incluindo a configuração de máquinas virtuais ou servidores na nuvem para suportar o desenvolvimento e a implantação do sistema.		
		O aluno configurou ambientes de desenvolvimento utilizando ambientes isolados, como virtualenv para Python ou nvm para Node.js, para gerenciar dependências específicas do projeto.		
	Definir os elementos de entrada, processamento e saída para a programação da aplicação web	O aluno especificou os elementos de entrada da aplicação web, identificando e configurando os formulários e campos de entrada necessários para a coleta de dados dos usuários.		
		O aluno implementou a lógica de processamento no back-end da aplicação, incluindo a validação e manipulação dos dados recebidos dos usuários conforme as regras de negócios definidas.		
		O aluno definiu os elementos de saída da aplicação web, criando e configurando as respostas e visualizações que serão apresentadas aos usuários após o processamento dos dados.		
		O aluno desenvolveu a integração entre a interface do usuário e o back-end, assegurando que os dados de entrada sejam corretamente enviados para o servidor e que as respostas sejam apresentadas de forma adequada.		

<b>Competências Técnicas</b>	Definir os elementos de entrada, processamento e saída para a programação da aplicação web	O aluno criou e testou endpoints da API que recebem dados de entrada, processam essas informações e retornam as saídas esperadas, garantindo a funcionalidade completa da aplicação.		
	Utilizar design patterns no desenvolvimento da aplicação web	O aluno aplicou o design pattern Observer para implementar uma funcionalidade de notificação entre diferentes componentes do sistema, garantindo a atualização automática dos observadores quando o estado do objeto observado muda.		
	Definir os frameworks a serem utilizados no desenvolvimento da aplicação web	O aluno utilizou o design pattern MVC (Model-View-Controller) para separar a lógica de apresentação da lógica de negócios e dados, facilitando a manutenção e escalabilidade da aplicação web.		
		O aluno selecionou e configurou framework para o desenvolvimento front-end, como Angular Native, para a construção da interface da aplicação web.		
		O aluno definiu e integrou frameworks para o desenvolvimento back-end, como Django Rest Framework, para a construção da lógica de servidor da aplicação.		
	Utilizar interações com base de dados para desenvolvimento de sistemas web	O aluno utilizou frameworks para a implementação de autenticação e autorização para garantir a segurança e o controle de acesso na aplicação web.		
		O aluno configurou e conectou a aplicação web a um banco de dados relacional, como MySQL ou SQLite, permitindo a persistência e recuperação de dados.		
		O aluno implementou operações CRUD (Create, Read, Update, Delete) utilizando consultas SQL para interagir com o banco de dados da aplicação web.		
		O aluno configurou e utilizou índices no banco de dados para acelerar a busca e a recuperação de dados em consultas complexas.		
	Transferir arquivos entre cliente e servidor por meio da aplicação web	O aluno criou e gerenciou esquemas de banco de dados, incluindo a definição de tabelas, relações e restrições para atender às necessidades da aplicação web.		
		O aluno implementou uma funcionalidade de upload de arquivos no front-end, permitindo que os usuários enviem documentos para o servidor através da aplicação web.		

<b>Competências Técnicas</b>	Transferir arquivos entre cliente e servidor por meio da aplicação web	O aluno configurou o servidor para receber e processar arquivos enviados pelos usuários, garantindo a correta armazenagem e integridade dos dados.		
	Estabelecer envio de notificações entre cliente e servidor por meio de aplicação web	O aluno implementou a funcionalidade de envio de notificações push para usuários, permitindo que o servidor envie alertas diretamente para o navegador ou aplicativo móvel.		
		O aluno desenvolveu a interface no front-end para exibir notificações recebidas, garantindo que as mensagens de notificação sejam apresentadas de forma clara e acessível aos usuários.		
		O aluno configurou o back-end para enviar notificações baseadas em eventos, como atualizações de status ou mudanças em dados importantes, para os usuários relevantes.		
	Desenvolver API (web services) para integração de dados entre plataformas	O aluno desenvolveu uma API RESTful que expõe endpoints para operações CRUD, permitindo a integração de dados entre diferentes plataformas.		
		O aluno desenvolveu endpoints de API que permitem a consulta e a filtragem de dados com base em parâmetros fornecidos pelos usuários, otimizando a interação com a plataforma.		
		O aluno configurou a API para lidar com erros e exceções de maneira adequada, retornando mensagens de erro significativas e códigos de status HTTP apropriados.		
		O aluno desenvolveu a API com suporte a CORS (Cross-Origin Resource Sharing), permitindo que a API seja acessada por diferentes domínios e plataformas.		
	Desenvolver sistemas web de acordo com as regras de negócio estabelecidas	O aluno implementou funcionalidades no sistema web que atendem às regras de negócios especificadas, garantindo que todas as operações de usuários estejam em conformidade com os requisitos estabelecidos.		
		O aluno configurou lógicas de validação e processamento de dados no sistema web, assegurando que os dados inseridos e manipulados estejam de acordo com as regras de negócios.		
		O aluno desenvolveu e testou fluxos de trabalho e processos no sistema web que seguem as diretrizes e políticas de negócios definidas para o funcionamento da aplicação.		
		O aluno configurou notificações e alertas no sistema web que são acionados com base em eventos e condições especificadas pelas regras de negócios, garantindo a comunicação eficiente de mudanças e ações necessárias.		

<b>Competências Socioemocionais</b>	Demonstrar autogestão	O aluno organizou seu tempo de forma eficiente, estabelecendo e cumprindo prazos para todas as etapas do projeto sem necessidade de supervisão constante.		
		O aluno priorizou suas tarefas de acordo com a importância e urgência, demonstrando a capacidade de gerir múltiplos aspectos do projeto simultaneamente.		
		O aluno tomou a iniciativa de identificar e resolver problemas de forma independente, sem depender de orientação externa para encontrar soluções.		
		O aluno documentou e comunicou seu progresso regularmente, mantendo todas as partes interessadas informadas sobre o andamento do projeto e qualquer dificuldade encontrada.		
		O aluno demonstrou responsabilidade ao lidar com erros e falhas, assumindo a responsabilidade e tomando medidas para corrigir e aprender com essas situações.		
	Demonstrar pensamento analítico	O aluno identificou e analisou os principais problemas e desafios do projeto, oferecendo soluções baseadas em uma avaliação cuidadosa dos dados e requisitos.		
		O aluno utilizou técnicas de modelagem e análise de dados para interpretar informações e tomar decisões fundamentadas no desenvolvimento da aplicação.		
		O aluno realizou uma análise detalhada dos requisitos do sistema, desmembrando-os em componentes menores para facilitar a implementação e garantir a conformidade com os objetivos do projeto.		
	Demonstrar inteligência emocional	O aluno demonstrou a capacidade de pensar criticamente sobre as implicações e consequências das decisões tomadas, avaliando como cada escolha impacta o projeto como um todo.		
		O aluno lidou com críticas e feedbacks construtivos de forma positiva e construtiva, demonstrando capacidade de manter a compostura e aprender com as sugestões recebidas.		
		O aluno gerenciou suas emoções eficazmente durante momentos de estresse ou pressão, mantendo a calma e a clareza para resolver problemas de maneira eficaz.		
		O aluno se comunicou de maneira assertiva e respeitosa, expressando suas ideias e preocupações de forma clara e sem agressividade, facilitando a resolução de conflitos.		

<b>Competências Socioemocionais</b>	Demonstrar inteligência emocional	O aluno fez um esforço para construir e manter relacionamentos positivos com colegas e stakeholders, mostrando habilidades sociais que ajudam a promover uma boa dinâmica de equipe.			
		O aluno gerenciou suas reações emocionais de forma eficaz ao lidar com mudanças inesperadas ou desafios, adaptando-se rapidamente e mantendo um desempenho estável.			
		O aluno demonstrou habilidades de auto-regulação ao manter uma atitude equilibrada e positiva em situações de conflito, ajudando a resolver desacordos de maneira construtiva e pacífica.			
	Demonstrar autonomia	O aluno assumiu a responsabilidade total por suas tarefas e entregas, gerenciando seu próprio tempo e recursos sem necessidade de supervisão constante.			
		O aluno tomou a iniciativa de identificar e iniciar projetos ou melhorias de forma independente, contribuindo de maneira proativa para o sucesso do projeto.			
		O aluno fez escolhas informadas e bem fundamentadas sobre as abordagens e técnicas a serem usadas, demonstrando capacidade de tomar decisões autônomas no desenvolvimento do projeto.			
		O aluno buscou e utilizou recursos e ferramentas de forma independente para resolver problemas e superar desafios, sem depender de orientação externa.			
		O aluno gerenciou e priorizou suas próprias tarefas e objetivos, mantendo-se organizado e focado sem necessidade de lembretes ou supervisão.			
	Demonstrar autonomia	O aluno resolveu problemas complexos e desafios de maneira autônoma, aplicando suas próprias habilidades e conhecimentos para encontrar soluções eficazes.			
		O aluno desenvolveu e seguiu um plano de ação para atingir suas metas e objetivos, mostrando independência na execução e monitoramento de seu progresso.			
		O aluno fez uso eficaz de sua iniciativa ao propor novas ideias e soluções para o projeto, demonstrando capacidade de pensar e agir de forma independente.			
		O aluno tomou decisões de forma autônoma, baseando-se em análise crítica e avaliação dos dados disponíveis, sem depender da aprovação ou orientação contínua de outros.			
<b>Nível de Desempenho</b>					
<b>Nota</b>					

**TABELA DE NÍVEIS DE DESEMPENHO**

Critérios de Avaliação	Nível de desempenho	Conversão em notas
Atendeu todos os critérios críticos (17) e até 53 desejáveis	25	100
Atendeu todos os critérios críticos (17) e até 52 desejáveis	24	98
Atendeu todos os critérios críticos (17) e até 48 desejáveis	23	96
Atendeu todos os critérios críticos (17) e até 44 desejáveis	22	94
Atendeu todos os critérios críticos (17) e até 40 desejáveis	21	92
Atendeu todos os critérios críticos (17) e até 36 desejáveis	20	90
Atendeu todos os critérios críticos (17) e até 32 desejáveis	19	88
Atendeu todos os critérios críticos (17) e até 28 desejáveis	18	85
Atendeu todos os critérios críticos (17) e até 24 desejáveis	17	80
Atendeu todos os critérios críticos (17) e até 20 desejáveis	16	75
Atendeu todos os critérios críticos (17) e até 16 desejáveis	15	70
Atendeu todos os critérios críticos (17) e até 12 desejáveis	14	65
Atendeu todos os critérios críticos (17) e até 8 desejáveis	13	60
Atendeu todos os critérios críticos (17) e até 4 desejáveis	12	55
Atendeu de 17 critérios críticos e qualquer quantidade de desejáveis	11	50
Atendeu de 15 a 16 critérios críticos e qualquer quantidade de desejáveis	10	45
Atendeu de 13 a 14 critérios críticos e qualquer quantidade de desejáveis	9	40
Atendeu de 11 a 12 critérios críticos e qualquer quantidade de desejáveis	8	35
Atendeu de 9 a 10 critérios críticos e qualquer quantidade de desejáveis	7	30
Atendeu de 7 a 8 critérios críticos e qualquer quantidade de desejáveis	6	25
Atendeu de 5 a 6 critérios críticos e qualquer quantidade de desejáveis	5	20
Atendeu de 3 a 4 critérios críticos e qualquer quantidade de desejáveis	4	15
Atendeu de 2 critérios críticos e qualquer quantidade de desejáveis	3	10
Atendeu de 1 critério crítico e qualquer quantidade de desejáveis	2	5
Não atendeu nenhum critério crítico e nenhum critério desejável	1	0

ELABORAÇÃO	DATA	APROVAÇÃO	DATA
Lindomar	21/07/2025	Recchia	/ /