Resumo do Projeto: Sistema de Gerenciamento de Dispositivos

Este projeto é um **sistema web** para gerenciar dispositivos (como servidores, switches, roteadores, etc.), permitindo **listar, adicionar, editar e excluir** dispositivos de um banco de dados MySQL.

1️⃣ Linguagens e Tecnologias Utilizadas

Tecnologia	Função no projeto
Python	Linguagem de programação principal. Toda lógica do backend é escrita em Python.
Flask	Framework web em Python, usado para criar rotas, receber requisições HTTP e renderizar páginas HTML.
MySQL	Banco de dados relacional para armazenar informações sobre os dispositivos.
HTML	Linguagem de marcação usada para criar as páginas web visíveis aos usuários.
CSS	Linguagem de estilo para definir aparência das páginas (cores, fontes, tabelas, etc.).
Jinja2	Sistema de templates do Flask, que permite **inserir dados do Python dentro do HTML**.

2️⃣ Estrutura do Projeto

```
/app.py               -> Código principal do backend (Flask + MySQL)
 /templates/          -> Páginas HTML do sistema
    base.html         -> Layout principal compartilhado
    index.html        -> Página de listagem de dispositivos
    add_device.html   -> Formulário para adicionar dispositivo
    edit_device.html  -> Formulário para editar dispositivo
 /static/
    style.css         -> Estilo visual do sistema (cores, tabelas, formulários)
```

3️⃣ Como o Script Funciona

Backend (Python + Flask + MySQL)

1. **Conexão com o Banco de Dados**

```
db = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    port=7306,
    database="rack_management"
)
```

- Conecta o Python ao MySQL para realizar consultas e alterações na tabela `devices`.

2. **Rotas do Flask**

- Cada função com `@app.route()` é uma **rota** (página ou ação do site):
    - `'/'` → Página inicial, lista todos os dispositivos.
    - `'/add'` → Adiciona novo dispositivo.

- `'/edit/<id>'` → Edita dispositivo existente.
- `'/delete/<id>'` → Deleta dispositivo do banco.

3. **Renderização de HTML**

- Função `render_template('nome.html', variaveis)`:
  - Insere **dados do Python** (como lista de dispositivos) dentro do HTML usando Jinja2.
  - Exemplo: `{{ device.device_name }}` mostra o nome do dispositivo na tabela.

4. **Formulários**

- Usam `method="POST"` para enviar dados do usuário para o servidor.
- O backend lê os dados com `request.form['campo']` e insere ou atualiza no banco com SQL.

5. **Redirecionamento**

- Após adicionar/editar/excluir, o sistema usa `redirect(url_for('index'))` para voltar à lista de dispositivos.

Frontend (HTML + CSS)

1. **Base do HTML (`base.html`)**

- Define o layout padrão: cabeçalho, navegação e rodapé.
- Usa `{% block content %}{% endblock %}` para inserir páginas específicas (`index.html`, `add_device.html`, etc.).

2. **Páginas Específicas**

- **index.html** → Lista os dispositivos em uma tabela e oferece links para editar ou deletar.
- **add_device.html** → Formulário para cadastrar novo dispositivo.
- **edit_device.html** → Formulário para alterar os dados de um dispositivo existente.

3. **CSS (`style.css`)**

- Define estilo das tabelas, formulários, textos e responsividade.
- Exemplo:

```
table, th, td {
    border: 1px solid black;
    padding: 10px;
}
a:hover {
    text-decoration: underline;
}
```

4️⃣ Fluxo de Funcionamento

1. Usuário acessa o navegador → entra na página inicial `/`.
2. Flask executa a função `index()`, busca dispositivos no MySQL e renderiza `index.html`.
3. Usuário clica em "Adicionar" → página `/add`.
4. Preenche o formulário → envia → Flask recebe os dados → insere no banco → redireciona para `/`.
5. Usuário pode **editar ou excluir** qualquer dispositivo usando links na tabela.
6. Todas as páginas usam o mesmo **estilo CSS** e a base do layout (`base.html`).

5️⃣ Conceitos Importantes para Iniciantes

- **Flask**: pequeno e fácil de usar, ótimo para aprender web em Python.
- **Rota (@app.route)**: determina qual função executa em cada URL.
- **Templates (Jinja2)**: misturam HTML com dados do Python.
- **MySQL Connector**: biblioteca Python que conecta e interage com MySQL.
- **POST vs GET**:
  - `GET` → usado para acessar páginas.
  - `POST` → usado para enviar dados de formulários.
- **CRUD**: esse projeto implementa as 4 operações básicas de banco de dados:
  - **C**reate → adicionar (`/add`)
  - **R**ead → listar (`/`)
  - **U**pdate → editar (`/edit/<id>`)
  - **D**elete → deletar (`/delete/<id>`)