Enunciado do Projeto: Sistema de Agendamento de Shows

Objetivo:

O objetivo deste projeto é criar uma API utilizando FastAPI para gerenciar um sistema de agendamento de shows. A API será responsável por realizar operações CRUD (Create, Read, Update, Delete) para a gestão de shows.

Requisitos do Banco de Dados

O sistema de agendamento de shows utilizará o banco de dados MySQL.

1. **Criação do Banco de Dados: O banco de dados deve ser criado no MySQL com o nome eventos_db.**
2. **Estrutura das Tabelas: O sistema terá uma tabela chamada shows com as seguintes colunas:**
    - **id (INT, AUTO_INCREMENT): Identificador único do show (chave primária).**
    - **nome (VARCHAR(255)): Nome do show.**
    - **data (DATE): Data do show (formato YYYY-MM-DD).**
    - **descricao (TEXT): Descrição do show.**
    - **created_at (DATETIME): Data e hora de criação do show (registro automático).**

SQL para criar a tabela:

```
CREATE DATABASE eventos_db;


USE eventos_db;


CREATE TABLE shows (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nome VARCHAR(255) NOT NULL,
  data DATE NOT NULL,
  descricao TEXT,
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

Rotas da API

A API terá as seguintes rotas, que permitirão realizar as operações CRUD no banco de dados.

1. Rota GET - Listar Shows

- **Objetivo: Retornar todos os shows cadastrados no banco de dados.**
- **Método HTTP: GET**
- **Rota: /shows/**
- **Resposta Esperada: Uma lista de objetos JSON com os dados dos shows.**

Exemplo de Resposta:


```
[
  {
    "id": 1,
    "nome": "Show de Rock",
    "data": "2025-06-20",
    "descricao": "Show de rock com várias bandas",
    "created_at": "2025-05-01T10:00:00"
  },
  {
    "id": 2,
    "nome": "Show de Jazz",
    "data": "2025-06-25",
    "descricao": "Show de jazz no centro cultural",
    "created_at": "2025-05-02T15:00:00"
  }
]
```


2. Rota GET - Obter Show Específico

- **Objetivo: Retornar os detalhes de um show específico.**

- **Método HTTP: GET**
- **Rota: /shows/{id}**
- **Resposta Esperada: Dados do show com o ID especificado.**

Exemplo de Resposta:


```
{
  "id": 1,
  "nome": "Show de Rock",
  "data": "2025-06-20",
  "descricao": "Show de rock com várias bandas",
  "created_at": "2025-05-01T10:00:00"
}
```


3. Rota POST - Criar Show

- **Objetivo: Criar um novo show no banco de dados.**
- **Método HTTP: POST**
- **Rota: /shows/**
- **Corpo da Requisição: Os dados do show a ser criado (nome, data e descrição).**

Exemplo de Corpo da Requisição:


```
{
  "nome": "Show de Pop",
  "data": "2025-07-10",
  "descricao": "Show de pop com artistas internacionais"
}
```


Exemplo de Resposta:


```
{
  "id": 3,
  "nome": "Show de Pop",
  "data": "2025-07-10",
```

"descricao": "Show de pop com artistas internacionais",

"created_at": "2025-05-03T12:30:00"

}

4. Rota PUT - Atualizar Show

- **Objetivo: Atualizar os detalhes de um show existente.**
- **Método HTTP: PUT**
- **Rota: /shows/{id}**
- **Corpo da Requisição: Dados atualizados do show (nome, data, descrição).**

Exemplo de Corpo da Requisição:

{

"nome": "Show de Pop Internacional",

"data": "2025-07-15",

"descricao": "Show de pop com artistas internacionais, novas atrações!"

}

Exemplo de Resposta:

{

"id": 3,

"nome": "Show de Pop Internacional",

"data": "2025-07-15",

"descricao": "Show de pop com artistas internacionais, novas atrações!",

"created_at": "2025-05-03T12:30:00"

}

5. Rota DELETE - Deletar Show

- **Objetivo: Deletar um show do banco de dados.**
- **Método HTTP: DELETE**
- **Rota: /shows/{id}**
- **Resposta Esperada: Mensagem de confirmação de exclusão.**

Exemplo de Resposta:

{

"message": "Show com ID 3 deletado com sucesso."

}

Requisitos Adicionais

- **Validação de Dados: A API deve garantir que os dados inseridos ou atualizados sejam válidos (por exemplo, a data do show não pode ser no passado).**
- **Autenticação (Opcional): Se necessário, você pode adicionar autenticação para garantir que apenas usuários autorizados possam criar, atualizar ou excluir shows.**
- **Tratamento de Erros: A API deve retornar mensagens de erro adequadas caso algum dado inválido seja fornecido ou caso o show não exista.**

Estrutura de Arquivos Sugerida:

- **app.py: Arquivo principal onde a aplicação FastAPI e as rotas são definidas.**
- **database.py: Arquivo para a configuração do banco de dados e conexão com MySQL.**
- **models.py: Arquivo para definir o modelo de dados (tabela shows).**
- **schemas.py: Arquivo para definir os modelos de dados que a API vai consumir e gerar (Pydantic models).**
