Enunciado do Projeto: Sistema de Gerenciamento de Serviços de Petshop

Objetivo:

O objetivo deste projeto é criar uma API utilizando FastAPI para gerenciar um sistema de serviços de petshop. A API será responsável por realizar operações CRUD (Create, Read, Update, Delete) para a gestão dos serviços oferecidos no petshop.

Requisitos do Banco de Dados

O sistema de petshop utilizará o banco de dados MySQL.

1. **Criação do Banco de Dados: O banco de dados será criado com o nome petshop_db.**
2. **Estrutura das Tabelas: O sistema terá uma tabela chamada servicos com as seguintes colunas:**
    - **id (INT, AUTO_INCREMENT): Identificador único do serviço (chave primária).**
    - **nome (VARCHAR(255)): Nome do serviço (exemplo: Banho, Tosa, Consulta).**
    - **preco (DECIMAL(10, 2)): Preço do serviço.**
    - **descricao (TEXT): Descrição detalhada do serviço.**
    - **created_at (DATETIME): Data e hora de criação do serviço (registro automático).**

SQL para criar a tabela:

```
CREATE DATABASE petshop_db;


USE petshop_db;


CREATE TABLE servicos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nome VARCHAR(255) NOT NULL,
  preco DECIMAL(10, 2) 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 Serviços

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

Exemplo de Resposta:

```
[
  {
    "id": 1,
    "nome": "Banho",
    "preco": 50.00,
    "descricao": "Banho completo para cães e gatos.",
    "created_at": "2025-05-01T10:00:00"
  },
  {
    "id": 2,
    "nome": "Tosa",
    "preco": 80.00,
    "descricao": "Tosa higiênica e estética para cães.",
    "created_at": "2025-05-02T11:00:00"
  }
]
```

2. Rota GET - Obter Serviço Específico

- **Objetivo: Retornar os detalhes de um serviço específico.**
- **Método HTTP: GET**
- **Rota: /servicos/{id}**
- **Resposta Esperada: Dados do serviço com o ID especificado.**

Exemplo de Resposta:

{

"id": 1,

"nome": "Banho",

"preco": 50.00,

"descricao": "Banho completo para cães e gatos.",

"created_at": "2025-05-01T10:00:00"

}

3. Rota POST - Criar Serviço

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

Exemplo de Corpo da Requisição:

{

"nome": "Consulta Veterinária",

"preco": 100.00,

"descricao": "Consulta veterinária com exame clínico e prescrição."

}

Exemplo de Resposta:

{

"id": 3,

"nome": "Consulta Veterinária",

"preco": 100.00,

"descricao": "Consulta veterinária com exame clínico e prescrição.",

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

}

4. Rota PUT - Atualizar Serviço

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

Exemplo de Corpo da Requisição:

{

"nome": "Consulta Veterinária Completa",

"preco": 120.00,

"descricao": "Consulta veterinária com exame clínico, laboratório e prescrição."

}

Exemplo de Resposta:

{

"id": 3,

"nome": "Consulta Veterinária Completa",

"preco": 120.00,

"descricao": "Consulta veterinária com exame clínico, laboratório e prescrição.",

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

}

5. Rota DELETE - Deletar Serviço

- **Objetivo: Deletar um serviço do banco de dados.**
- **Método HTTP: DELETE**

- **Rota: /servicos/{id}**
- **Resposta Esperada: Mensagem de confirmação de exclusão.**

Exemplo de Resposta:

{

"message": "Serviço 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, o preço não pode ser negativo).**
- **Autenticação (Opcional): Se necessário, você pode adicionar autenticação para garantir que apenas usuários autorizados possam criar, atualizar ou excluir serviços.**
- **Tratamento de Erros: A API deve retornar mensagens de erro adequadas caso algum dado inválido seja fornecido ou caso o serviço 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 servicos).**
- **schemas.py: Arquivo para definir os modelos de dados que a API vai consumir e gerar (Pydantic models).**