Gerenciando um CRUD Completo com FastAPI e SQLAlchemy

Nesta aula, aprendemos a criar uma aplicação FastAPI integrada ao SQLAlchemy para gerenciar um CRUD completo. Configuramos o banco de dados com SQLite, definimos modelos ORM para representar tabelas, e esquemas Pydantic para validar dados.

Endpoints para Leitura de Dados

Listando Livros

Implementamos uma rota GET para retornar todos os livros cadastrados, exibindo informações como título, autor e descrição.

Buscando um Livro

Criamos uma rota GET que recebe um ID de livro e retorna os dados correspondentes, caso o livro exista no banco de dados.

Endpoints para Atualização de Dados

1

Atualizando Informações

Implementamos uma rota PUT para atualizar informações de um livro específico, utilizando um ID para identificar o livro a ser modificado.

2

Validando Dados

Utilizamos esquemas Pydantic para validar os dados recebidos, garantindo que as informações atualizadas estejam no formato correto.

Endpoints para Deleção de Dados

Excluindo Livros

Criamos uma rota DELETE para remover livros do banco de dados, utilizando um ID para identificar o livro a ser removido.

Desafio Prático: Implementando Rotas para CRUD de Livros

1

Criando Rotas

Desenvolvemos rotas para as operações CRUD (Create, Read, Update, Delete) para livros, utilizando a biblioteca FastAPI.

2

Definindo Modelos ORM

Criamos modelos ORM com SQLAlchemy para representar tabelas no banco de dados, mapeados para objetos Python.

3

Esquemas Pydantic

Utilizamos esquemas Pydantic para validar os dados de entrada e saída das APIs, garantindo a consistência.

Desafio Prático: Implementando Rotas para CRUD de Livros

Estrutura da Aplicação
Banco de Dados: Utilizaremos SQLite para simplicidade.

Modelos e Esquemas:
Book: Representa um livro.

Endpoints:

- POST /books/: Cria um novo livro.
- GET /books/: Lista todos os livros.
- GET /books/{book_id}: Busca detalhes de um livro.
- PUT /books/{book_id}: Atualiza informações de um livro.
- DELETE /books/{book_id}: Remove um livro.

Um livro deve conter pelo menos:

id
title
author
description

Adicionando Filtros Dinâmicos às Consultas

1

Filtros por Título

Implementamos filtros para buscar livros por título, utilizando parâmetros na URL para definir a pesquisa.

2

Filtros por Autor

Adicionamos filtros para buscar livros por autor, expandindo as opções de pesquisa e personalização.

3

Filtros por Descrição

Criamos filtros para buscar livros por descrição, permitindo buscas mais específicas e relevantes.

Explorando Dependências e Injeção de Dependências

1

Definindo Dependências

Criamos funções para gerenciar o banco de dados e outros recursos, definindo-as como dependências das rotas.

2

Injeção de Dependências

Utilizamos a injeção de dependências para acessar as funções definidas, simplificando o código e promovendo a reutilização.

3

Gerenciando Recursos

A injeção de dependências facilita a organização e o gerenciamento de recursos, como conexões com o banco de dados.

Seguindo Boas Práticas de Organização de Código

Organizamos o código seguindo boas práticas, utilizando pastas para separar funcionalidades e módulos, promovendo a modularidade e a legibilidade.