Criação de Models

Documentação Oficial:

https://docs.djangoproject.com/pt-br/5.0/

Django ORM

O Django ORM (Object-Relational Mapping) é uma ferramenta que facilita a interação com bancos de dados relacionais por meio de objetos Python, permitindo operações de banco de dados sem a necessidade de escrever consultas SQL diretamente. Ele mapeia modelos de dados Python para tabelas no banco de dados, simplificando o desenvolvimento de aplicativos web complexos.

O ORM traduz operações em métodos Python, convertendo-os em consultas SQL apropriadas, promovendo a portabilidade do código entre diferentes sistemas de gerenciamento de banco de dados e permitindo que os desenvolvedores se concentrem na lógica da aplicação.

Django ORM

Leia a documentação a seguir:

Models:

https://docs.djangoproject.com/pt-br/5.0/topics/db/models/

Consultas e Querysets:

https://docs.djangoproject.com/pt-br/5.0/topics/db/queries/

Agregação de resultados de consultas:

https://docs.djangoproject.com/pt-br/5.0/topics/db/aggregation/

Django ORM Cookbook – um livro de receitas rápidas “como fazer operação x” :

https://readthedocs.org/projects/django-orm-cookbook/downloads/pdf/latest/

Django ORM – Exemplo Rápido

Neste exemplo o modelo define um Person, o qual tem um first_name``e ``last_name:

```python
from django.db import models


class Person(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
```

first_name e last_name são “fields” do modelo. Cada campo é especificado como um atributo de classe, e cada atributo mapeia uma coluna do banco de dados. O modelo Person criaria uma tabela de banco de dados como esta:

```sql
CREATE TABLE myapp_person (
    "id" bigint NOT NULL PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
    "first_name" varchar(30) NOT NULL,
    "last_name" varchar(30) NOT NULL
);
```

Django ORM – Exemplo Rápido

Algumas notas técnicas:

- O nome da tabela, myapp_person, é automaticamente derivado de alguns metadados do modelo mas pode ser sobrescrito. Veja Table names para mais detalhes.
- Um campo id é adicionado automaticamente, mas este comportamento pode ser sobrescrito. Veja Campos chave-primária automáticos..
- O SQL CREATE TABLE neste exemplo está formatado usando a syntax do PostgreSQL, mas vale a pena notar que o Django usa SQL de acordo com o “backend” do banco de dados especificado no seu arquivo de definições.

Django ORM – Usando Modelos

Uma vez que tenha definido seus modelos, você precisa dizer ao Django que irá usar aqueles modelos. Faça isso editando seu arquivo de definições e alterando a definição do INSTALLED_APPS para adicionar o nome do módulo que contém seu models.py.

Por exemplo, se o modelo para sua aplicação está em um módulo myapp.models (a estrutura de pacote que foi criada para uma aplicação pelo script manage.py startapp), INSTALLED_APPS deve ser lido, em parte:

```
INSTALLED_APPS = [
    # ...
    "myapp",
    # ...
]
```

Quando adicionar novas aplicações ao INSTALLED_APPS, tenha certeza de rodar manage.py migrate, opcionalmente antes fazendo migrações para eles antes com manage.py makemigrations.

Django ORM – Tipos de Campos

Cada campo no seu modelo deve ser instanciado da classe Field apropriada.

O Django usa o tipo de classe do campo para determinar algumas coisas:

- O tipo da coluna, o qual diz ao banco de dados que tipo de dado irá armazenar (e.g. INTEGER, VARCHAR, TEXT).
- O widget HTML padrão para usar quando renderizar o campo de um form (ex. <input type="text">, <select>).
- Os requisitos mínimos de validação, usados no admin do Django e em formulários gerados automaticamente.

Django vem com dúzias de tipos de campos; você pode achar uma lista nas referências de campos de modelo. Você pode facilmente escrever seus próprios campos se os que vêm definidos no Django não resolverem; veja Como criar campos de modelo customizado.

Django ORM – Opções de Campos

Cada campo recebe um certo conjunto de argumentos específicos (documentados na referência de campos de modelo). Por exemplo, a CharField (e suas subclasses) requerem um argumento max_length o qual especifica o comprimento usado para armazenar dados do campo VARCHAR do banco de dados.

Também há um conjunto de argumentos comuns disponíveis para todos os tipos de campos. Todos opcionais. Estes argumentos são explicados na referência, mas aqui um sumário dos argumentos mais usados:

Django ORM – Relacionamentos

Campos de relacionamento

Django também define um conjunto de campos que representam relacionamentos.

ForeignKey

class (to, on_delete, **optionsForeignKey

Uma relação de muitos para um. Requer dois argumentos posicionais: a classe para qual o modelo está relacionado e a opção.

Para criar uma relação recursiva – um objeto que tem um muitos-para-um relação consigo mesmo – use .models.ForeignKey('self', on_delete=models.CASCADE)

Se você precisar criar um relacionamento em um modelo que ainda não foi definido, Você pode usar o nome do modelo, em vez do objeto de modelo em si:

```python
from django.db import models


class Car(models.Model):
    manufacturer = models.ForeignKey(
        "Manufacturer",
        on_delete=models.CASCADE,
    )
    # ...


class Manufacturer(models.Model):
    # ...
    pass
```

Django ORM – Tipos de Campos

Vamos acessar a documentação oficial e dar uma olhada nos tipos de campos e nas opções de campos:

https://docs.djangoproject.com/pt-br/5.0/ref/models/fields/#model-field-types

Exercício

Exercício

Dados os dicionários de dados nos slides a seguir, crie uma tabela MEDICO e uma tabela especialidade utilizando Django ORM.

Você deverá criar os models e fazer o Migrate.

Entregar:

- Print da tela dos Models (o seu nome deve aparece em algum lugar)
- Print da tela do DB Browser com o banco aberto e mostrando as tabelas

Exercício

Tabela: MEDICO

Campo	Tipo de Dados	Descrição
id_medico	Inteiro	Identificador único do médico
nome	Texto	Nome completo do médico
endereco	Texto	Endereço do consultório ou residência do médico
telefone	Texto	Número de telefone do médico
email	Texto	Endereço de e-mail do médico
data_nascimento	Data	Data de nascimento do médico
crm	Texto	Número de registro do Conselho Regional de Medicina
especialidade_id	Inteiro	Chave estrangeira referenciando a tabela de ESPECIALIDADE, indicando a especialidade do médico

Exercício

Tabela: ESPECIALIDADE

Campo	Tipo de Dados	Descrição
id_especialidade	Inteiro	Identificador único da especialidade
nome	Texto	Nome da especialidade
descricao	Texto	Descrição da especialidade