Revisão da aula passada:
Na aula passada vimos como instalar e configurar o framework django.
Vimos como criar o nosso primeiro projeto

Django-admin startproject <nome do projeto>

Vimos como criar os apps em um projeto (e registra-la nas configurações)

python3 manage.py startapp <nome do app>

Colocar o nome da app na lista "INSTALLED_APPS"

Vimos como criar um banco de dados para o projeto com os aplicativos padrões do django

python3 manage.py migrate

O banco de dados usado depende da configuração descrita no arquivo settings.py
Vimos como criar um super usuario para gerenciar as tabelas do banco de dados

python3 manage.py createsuperuser

Na aula de hoje iremos ver como criar os models usando o framework
Models:

Os models são as informações as quais voce quer persistir no banco de dados.

Cada model tem um conjunto de atributos e comportamentos das entidades que estão sendo armazenadas.

O framework facilita a interação com o banco de dados pois os atributos de uma classe serão vistos como campos nas tabelas. Cada classe geralmente será uma tabela.

Em Django, todos os modelos devem herdar da classe django.db.models.Model
Existem vários tipos de campos que podem ser usados nos fields, além de suas opções.
Alguns são

CharField – Para strings de até 255 caracteres

DateField – Para datas

IntegerField – Para valores inteiros

EmailField – String de e-mail com validacao

DecimalField – Usado para representações decimais com ponto fixo

FloatField – Usado para representacoes de ponto flutuante

Entre outros, que são listados na documentação do django

Além disso, os campos tem parametros que podem ser utilizados:

primary_key = [True/false] – define o campo como chave primaria

unique = [True/false] – define se o campo será único na tabela (ou seja, outros não podem ser inseridos)

blank = [True/False] – define se pode ou não ficar branco

max_length = <int> - define o tamanho máximo de caracteres do campo em campos de texto

max_digits= <int> - usado para definir o máximo de digitos em numeros decimais

decimal_places = <int> - usado para definir quantas casas decimais serão armazenadas num valor DecimalField

Entre outros.

É Possível definir relacionamentos 1 para 1, 1 para N e N para M.
Para a definição de relacionamentos usamos alguns tipos especiais de campos:
- django,db.models.OneToOne(<classe relacionada>, on_delete=<metodo executado ao deletar objeto relacionado>): Relacionamento 1 para 1
- django.db.models. ForeignKey(<classe relacionada>, on_delete=<metodo executado ao deletar objeto relacionado>): Relacionamento 1 para N
-django.db.models. ManyToManyField(<classe relacionada>): relacionamento N para M

É importante notar que tanto o relacionamento 1 para 1 quanto o 1 para N precisam de um parametro chamado on_delete, que será um procedimento executado na exclusão de um elemento que está relacionado com a entidade em questão.

Toda vez que são criados novos models, ou são feitas alterações nos existentes, é necessário que sejam executadas ações de gerenciamento:

Python3 manage.py makemigrations <app> (o app é opcional)

Python3 manage.py migrate

Para testar é possível alterar o arquivo admin.py do app “registrando” as classes de modelo no espaço do administrador. Isso é feito importando o arquivo model.py (ou as classes do arquivo) no admin.py e chamando o método register:

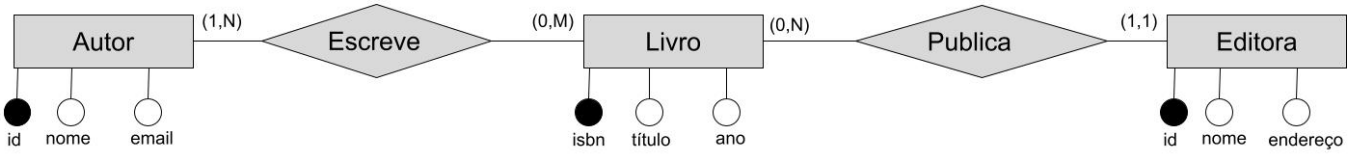
```
From .model import <classe 1>, <classe 2>, ....
....
admin.site.register(<classe de modelo>)
```

É possível passar uma lista (ou qualquer outra estrutura iteravel) como argumento

Vale ressaltar que o site admin usa o método __str __ para listar os objetos, sendo assim, para uma visualização mais amigavel é necessário reescrever o método para a classe.

Exemplo:

Considerando o seguinte diagrama de enteidade relacionamento, elabore as classes de modelo em django.




```
class Autor(models.Model):
   id = models.PositiveIntegerField(primary_key=True)
   nome = models.CharField(max_length=100, blank=False)
   email = models.EmailField(max_length=100, blank=True)

class Editora(models.Model):
   id = models.PositiveIntegerField(primary_key=True)
   nome = models.CharField(max_length=100, blank=False)
   Endereco = models.CharField(max_length=200, blank=True)

class Livro(models.Model):
   isbn = models.PositiveIntegerField(primary_key=True)
   titulo = models.CharField(max_length=100)
   autores = models.ManyToManyField(Autor)
   editora = models.ForeignKey(Editora, on_delete=models.CASCADE)
   ano = models.PositiveIntegerField()

   def __str__(self):
      return self.nome
```

Na nossa proxima aula iremos explorar um pouco mais as questões relacionadas a parte de view do framework, vamos ver como criamos paginas html usando django e como transmitir informações das classes para a parte de visualização, explorando um pouco mais a parte de front end.