Para a criação do site com Django:

Criar e ativar ambiente virtual

Instalar Django

Iniciar o projeto Django

Django-admin startproject {nome do site}

Para cada API que eu for criar, tenho que dar o comando

Python manage.py startapp {nome da API}

Depois de criar a primeira API, que no caso chamei de core, vamos criar uma pasta templates e dentro dela vamos criar uma pasta também chamada core, onde iremos criar um arquivo html como base para todas as páginas que serão criadas.

Após isso, temos que ir criando os arquivos html utilizando a opção de estender do arquivo base, todos dentro desta última pasta chamada core.

Temos que registrar esses arquivos html criados a partir do base no arquivo views.py que se encontra dentro da pasta core

E depois temos que adicionar esses caminhos ao arquivo urls.py que fica dentro da pasta principal do meu site.

Ainda na criação do arquivo base.html nós vamos criar os links que serão acessados e os botões para login e novo usuário, apesar de não criar ainda o caminho que eles devem redirecionar ao clicarmos.

Por fim, criamos um rodapé para a página, com os itens que serão exibidos ao final da página.

<li><a href="#" class="text-lg text-teal-500 hover:text-teal-700">Sobre</a></li>

Aqui onde tem asterisco vamos substituir pelo nome salvo no arquivo urls

Agora vamos criar mais um app chamado item

Python manage.py startapp item

Dentro da pasta item, teremos o arquivo models.py

Vamos criar uma categoria.

Depois que fizermos o makemigrations e o migrate, para realizar alterações (pelo que entendi), temos que ter um usuário criado.

Para isso vamos usar o comando

Python manage.py createsuperuser

Na pasta item temos um arquivo chamado admin e nele vamos registrar o model que acabamos de criar (Category).

Mudei o nome de Category para Categoria. Para isso funcionar temos que parar o servidor, realizar o makemigrations e o migrate e depois rodar o servidor de novo. O que ocorre quando a gente muda somente o nome, ao que parece, é que ele altera no visual da página mas no nome da tabela do banco de dados ele ainda continua com o nome antigo (Category).

Nota: Ao que parece ele coloca o nome que a gente escrever no plural, como se fosse um conjunto daquelas coisas...

O Django faz isso automaticamente, para ajustar vamos no arquivo models.py e vamos adicionar uma classe

    class Meta:

        verbose\_name\_plural = 'Categorias' #Aqui eu altero a forma como o Django vai transformar em plural as palavras que eu escrever.

Ainda no arquivo models.py vamos criar a classe Item, para representar os itens que o usuário quiser adicionar ao site, dentro de cada uma das categorias (são os produtos em si).

Depois de criarmos o campo Item, temos que ir no arquivo admin.py e registrar ele também, assim como foi com o campo Categoria.

Após a criação do item, vamos fazer com que os últimos itens adicionados apareçam na tela inicial do site e que indique quantos itens constam em cada categoria da página.

Após isso, temos que criar uma página para visualização de cada item detalhadamente.

Vamos fazer isso no arquivo views da pasta item.

Agora na pasta item temos que criar uma pasta templates e dentro dela criar uma chamada item, para criarmos, dentro desta última, um arquivo html para ser a página detail.

Como, no exemplo, vamos utilizar imagem também na página da descrição detalhada do produto, vamos ter que criar um arquivo urls.py na pasta item, que será o caminho para acessarmos essas imagens.

Após isso, temos que incluir essa url no arquivo principal de urls, na pasta principal do site.

Dentro desse arquivo html chamado detail vamos escrever todos os campos que quisermos que apareçam na descrição detalhada do produto. Por fim, vamos adicionar um botão que irá redirecionar para o contato com o vendedor do produto.

Após isso, vamos criar um campo abaixo com os itens relacionados, ou seja, aqueles itens cadastrados que estejam na mesma categoria que estamos visualizando a página detalhada.

Agora, vamos criar uma página para o cliente se cadastrar no site.

Na pasta core, que é nossa pasta da página principal, vamos criar o arquivo urls.py

(Esse passo de criar um arquivo de urls dentro do core eu não entendi bem)

Para criar o usuário e tudo o que está envolvido com validação, por exemplo, vamos criar um arquivo chamado forms.py e nele vamos importar a biblioteca do Django chamada forms.

Após criada a tela de cadastro, será criada a tela de login do usuário.