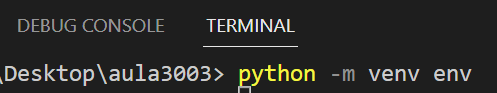
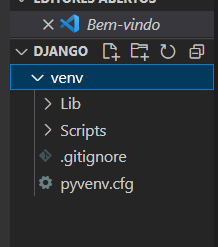
**CRIAR PROJETO NO DJANGO**

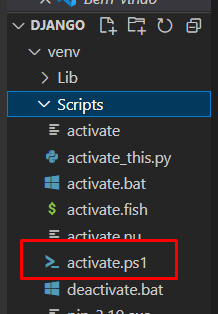
Escolha uma pasta para abrir com editor de texto VSCode e crie o ambiente virtual utilizando pip com o seguinte comando:

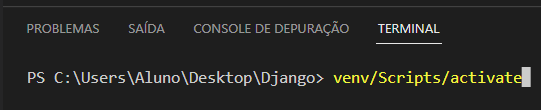


Aguarde até o final da criação e configuração dos arquivos e verifique se o diretório já está disponível em seu projeto.

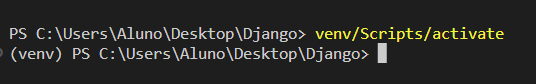


**Ativar ambiente Virtual**

Dentro da pasta criada no passo anterior existe um arquivo chamado activate.ps1 que é responsável por ativar o ambiente virtual.

Este arquivo deve ser executado para que os recursos do ambiente virtual estejam disponíveis. Execute o seguinte comando:

Logo em seguida, se o comando for bem-sucedido, o terminal ficará da seguinte forma:



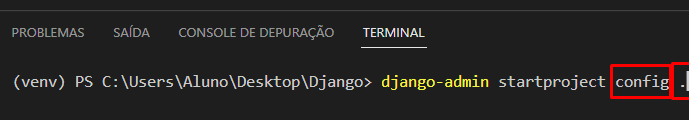
O nome do ambiente virtual será mostrado antes do endereço do projeto.

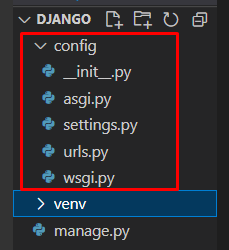
**INSTALAR O FRAMEWORK DJANGO**

Com o gerenciador de dependências *pip* execute o seguinte comando:

**CRIAR O PROJETO**

Após a instalação do framework, execute o seguinte comando:

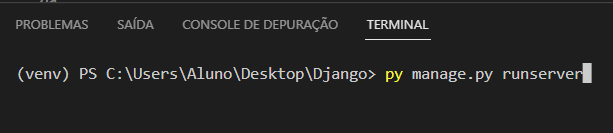
*Django-admin startproject* ***<nome> .***

Após ***startproject*** você definirá o nome das configurações do seu projeto e adicionando um ponto ao final ‘.’ As configurações do seu projeto ficarão em uma única pasta.

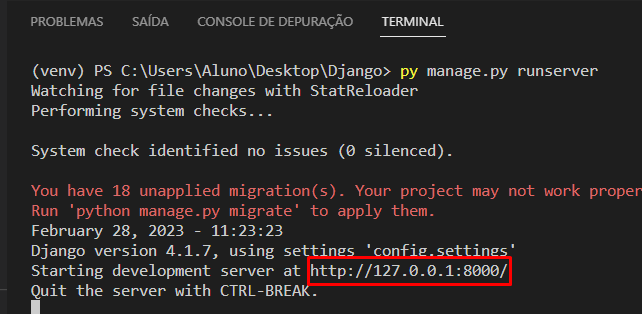
**EXECUTANDO O PROJETO PELA PRIMEIRA VEZ**

Para visualizar o projeto no navegador de internet é necessário executar o seguinte comando:

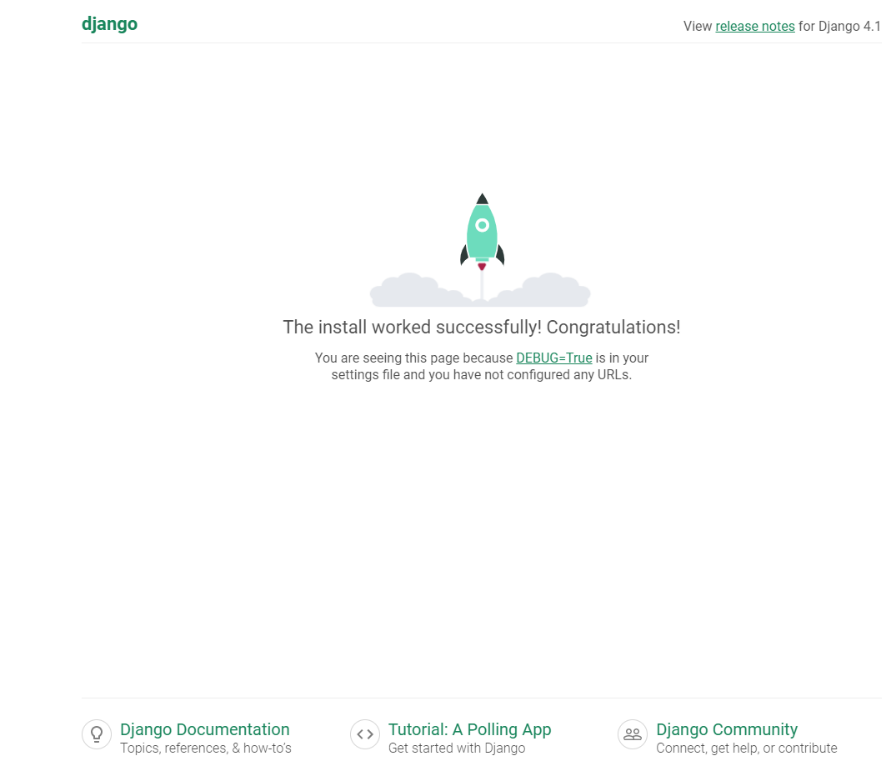
*py manage.py runserver*



O arquivo manage.py é criado no início do projeto e nos permite iniciar o servidor e executar o projeto.



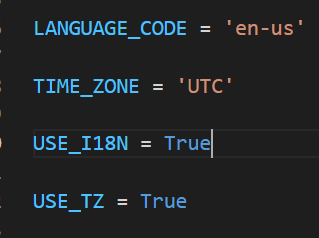
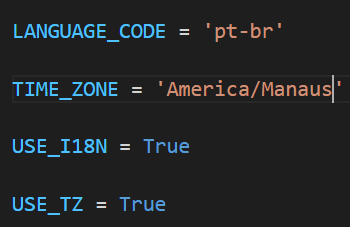
Após a execução do comando é possível acessar a página inicial do projeto digitando o IP no navegador ou utilizando o atalho CTRL + clique no endereço destacado na imagem acima.

Caso as configurações estejam corretas, será exibida a seguinte página:

CTRL + C encerra a atividade do servidor.

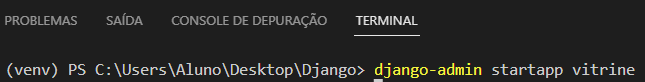
**AJUSTE DE IDIOMA E FUSO-HORÁRIO**

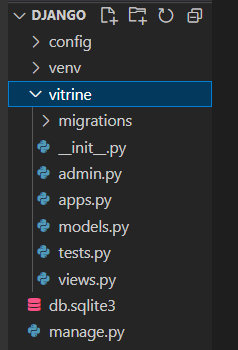
O arquivo settings.py é base para várias modificações que serão feitas no projeto, incluindo o ajuste de idioma e fuso-horário.

Basta alterar os seguintes dados:

https://en.wikipedia.org/wiki/List\_of\_tz\_database\_time\_zones

**CRIAR UM APP**



Ao executar o comando será criada um novo diretório com arquivos para configuração da lógica de funcionamento do app dentro do projeto.

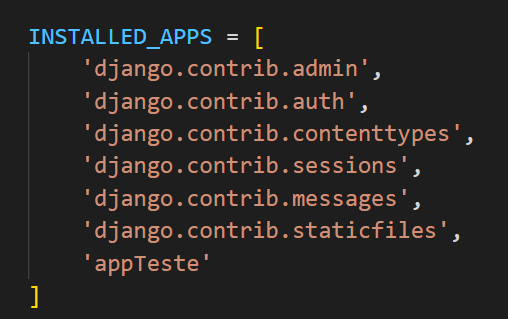
**TEMPLATES**

**Interface gráfica do usuário, Aplicativo

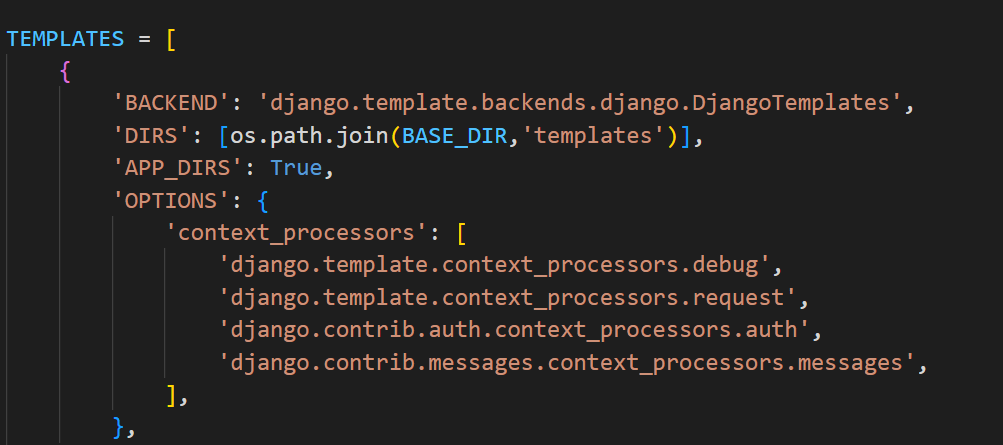
Descrição gerada automaticamente**Para personalizar as páginas do app é necessário criar um diretório na raiz do projeto armazenando os arquivos HTML do sistema.

**Editar arquivo settings.py**

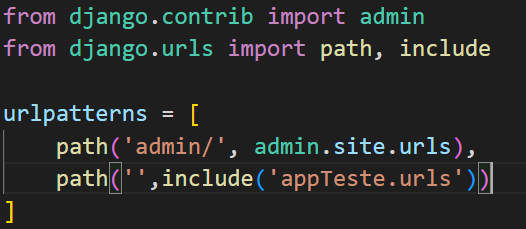
Registrar novo app nas configurações do projeto



Associar a pasta de *templates* as configurações do projeto



Incluir **URLS do app** no arquivo de **URLS principal do projeto**

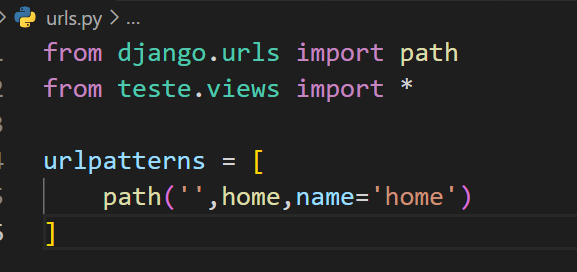


Após configurar o diretório e as páginas e as rotas do projeto e aplicativo, crie uma função no arquivo ‘*views’* do app para renderizar a página.

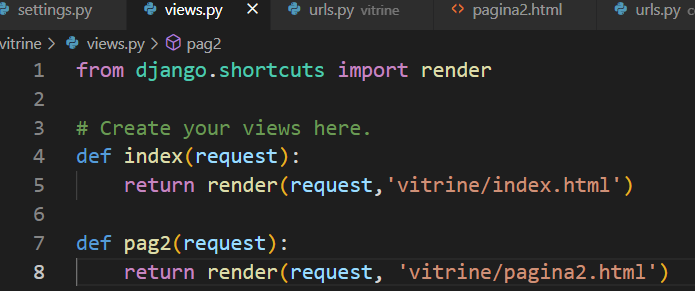
Interface gráfica do usuário

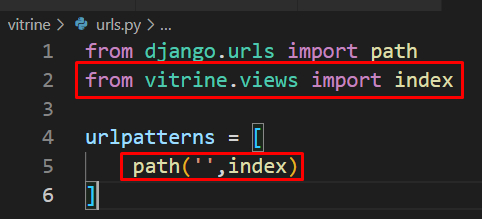
Descrição gerada automaticamente

Em seguida crie um arquivo, na pasta do app, chamado urls.py e adicione o path com nome da função de renderização da página

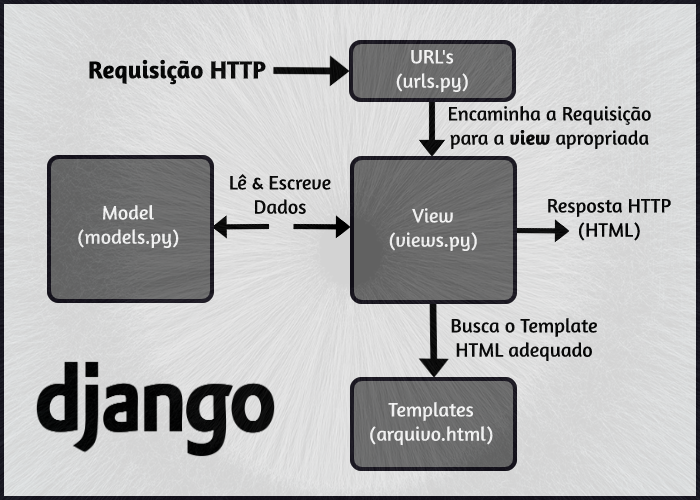


**ASSOCIANDO TEMPLATE HTML AO PROJETO**

Criar funções no arquivo views.py para renderizar as páginas conforme seus diretórios dentro do projeto.

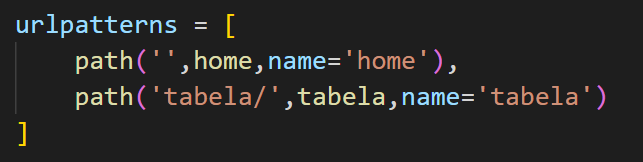
No arquivo de urls.py, dentro do app, listar os paths para cada página usando como base os nomes das funções criadas em views.py

**LINKS ENTRE PÁGINAS**

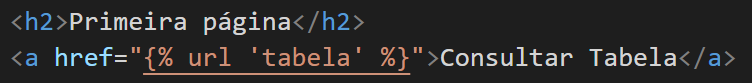


Para estabelecer ligação entre as páginas é necessário usar configurações específicas do framework Django.

Ao definir uma rota, configure, em *urls.py* do app, e o argumento ‘name’ para identificar abrir a página desejada.



O arquivo HTML também será modificado, ao invés do link para arquivo direto será utilizada a marcação de urls do Django.

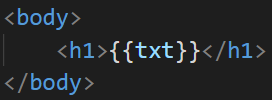


Texto

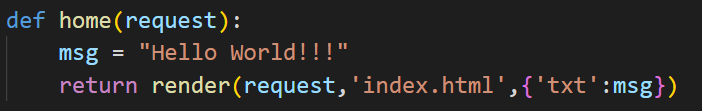
Descrição gerada automaticamente

*Observe o posicionamento das aspas para o redirecionamento funcionar*

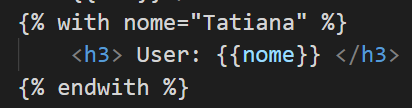
**SINTAXE**

Variáveis podem ser exibidas na página HTML Através de um conjunto de chaves:

As variáveis podem ser instanciadas nas funções do arquivo *views.py* do app e repassadas para a página HTML como um dicionário.

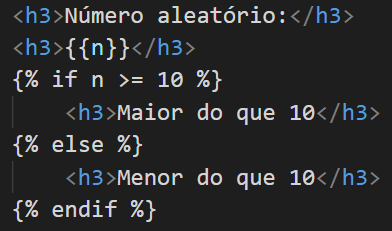


Para enviar dados utilização nas páginas HTML é necessário informar mais um argumento no render.

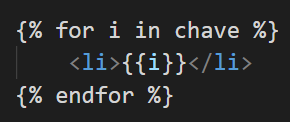
Ou podemos criar variáveis direto no HTML

**TAGS**

Essas palavras-chave, *if* e *for*, são chamadas de "*templates tags*" do Django.

Estrutura de decisão ***if***

Estrutura ***for***

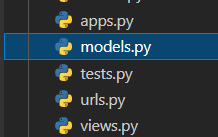


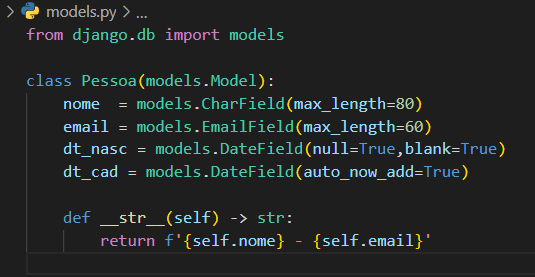
**Models**

Models é a fonte de informações sobre seus dados em cada app. Ele contém os campos e comportamentos essenciais dos dados que você está armazenando.

Geralmente, cada modelo é mapeado para uma única tabela de banco de dados.

* Cada modelo é uma classe Python que subclasse django.db.models.Model.
* Cada atributo do modelo representa um campo do banco de dados.

Na pasta do app do projeto, acesse o arquivo models.py

Crie a classe do modelo de dados

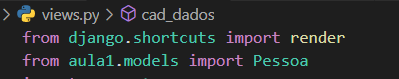
Em seguida, execute os seguintes comandos de migração no terminal

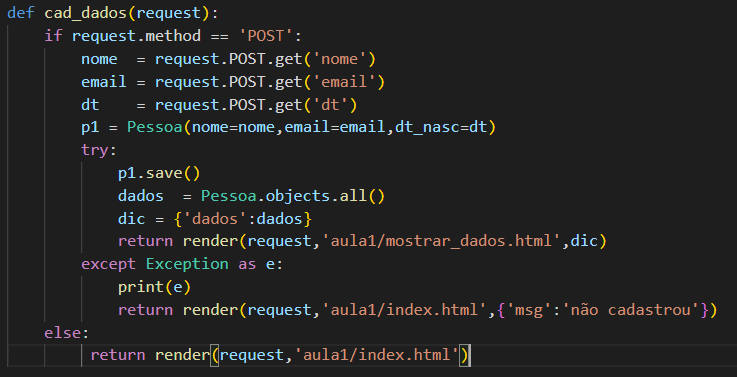


Fields types

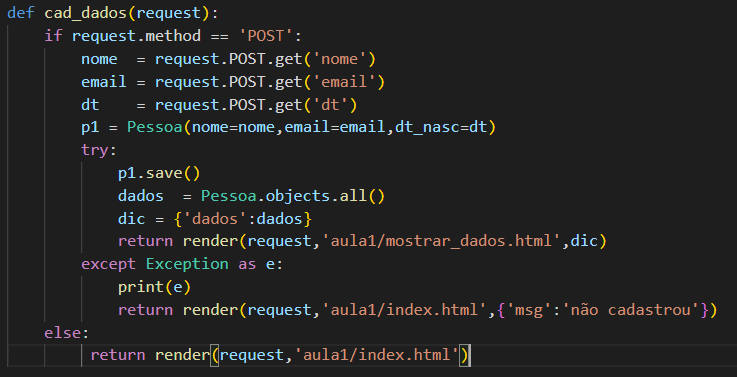
<https://docs.djangoproject.com/en/4.2/ref/models/fields/#field-types>

INSERINDO DADOS DE UM FORM MANUAL

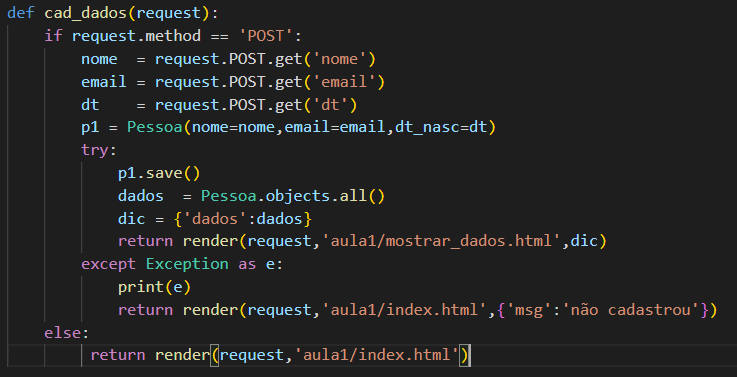
Após criar a tabela do banco de dados com Models, importe o modelo no arquivo de views

Crie um método para receber os dados do formulário

Em seguida, crie um objeto da classe para salvar os dados

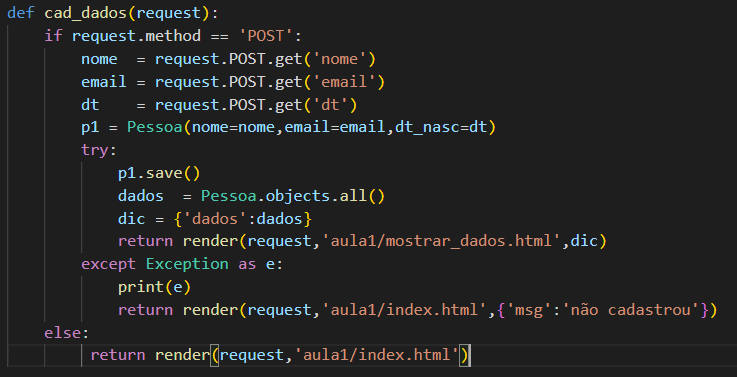


Utilize o método save() para armazenar os dados na tabela do banco de dados

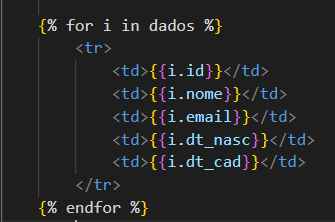


**Mostrar os dados Inseridos no SQLite**

Para obter os dados registrados na tabela utiliza-se o método all(), conforme exemplo abaixo:



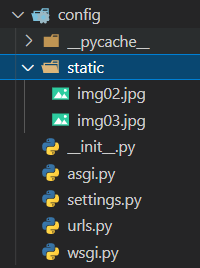
Caso seja necessário mostrar os dados em uma página do site, os dados devem estar organizados em um dicionário no método render()

Por fim, os dados podem ser mostrados na página HTML utilizando a chave do dicionário em uma estrutura de repetição FOR

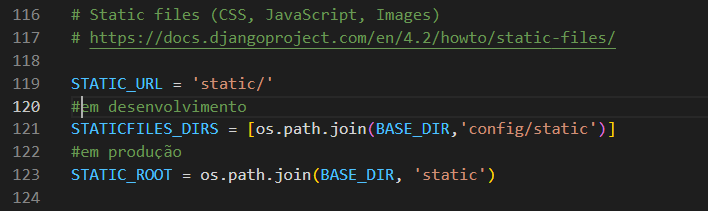
**CARREGANDO ARQUIVOS ESTÁTICOS**

Para associar e localizar arquivos estáticos como imagens, arquivos em pdf, arquivos CSS/JS ou vídeos em um projeto Django é necessário carrega-los em um diretório especifico dentro das pastas do projeto e identifica-lo no arquivo settings.py

Dentro das configurações do seu projeto crie uma pasta para os arquivos estáticos. Salve todos os arquivos necessários nessa pasta criada no passo anterior.

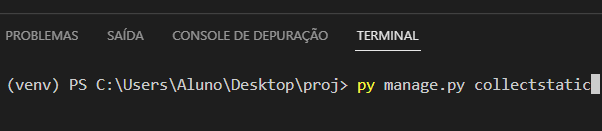


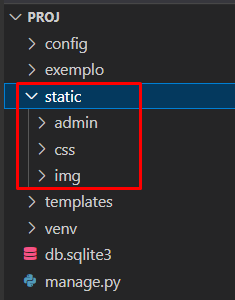
O próximo passo é identificar todo esse conteúdo no arquivo settings.py criando uma constante chamada STATICFILES\_DIRS

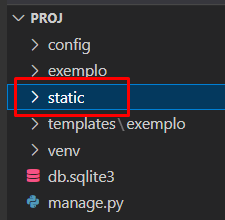


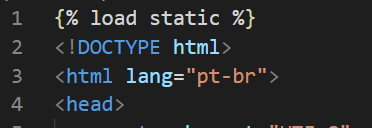
Após configurar o arquivo settings.py com a localização dos arquivos estáticos execute o seguinte comando:

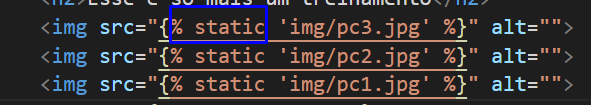
*py manage.py collectstatic*



Caso esteja tudo certo, será criado um novo diretório conforme as configurações de settings.py com a cópia de todos os arquivos estáticos e adicionais do painel administrativo.



Utilize o comando {% load static %} no início dos documentos para referenciar os arquivos estáticos.

Adicione static a referência dos arquivos da seguinte forma: