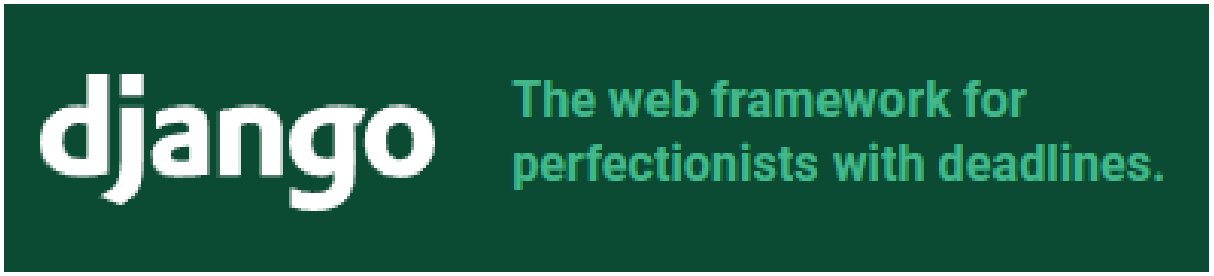
PROGRAMAÇÃO
BACK-END



Conectar o Django no PostgreSQL

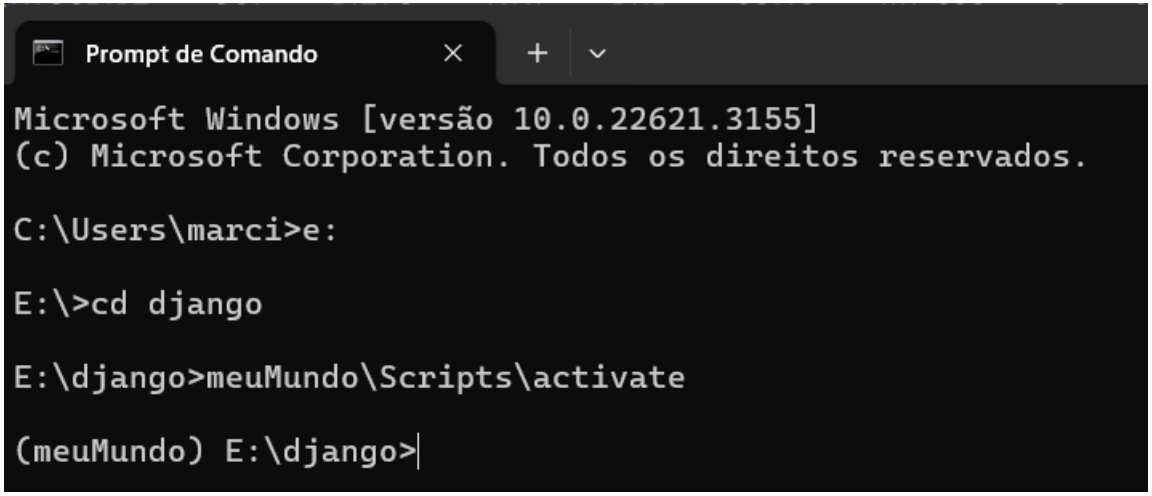
PROGRAMAÇÃO BACK-END

Objetivos desta apresentação

Preparar um aplicativo dvdrentalapp para conectar no banco dvdrental do PostgreSQL e apresentar os dados da tabela customer.

Iniciando

Rodar o script activate.bat

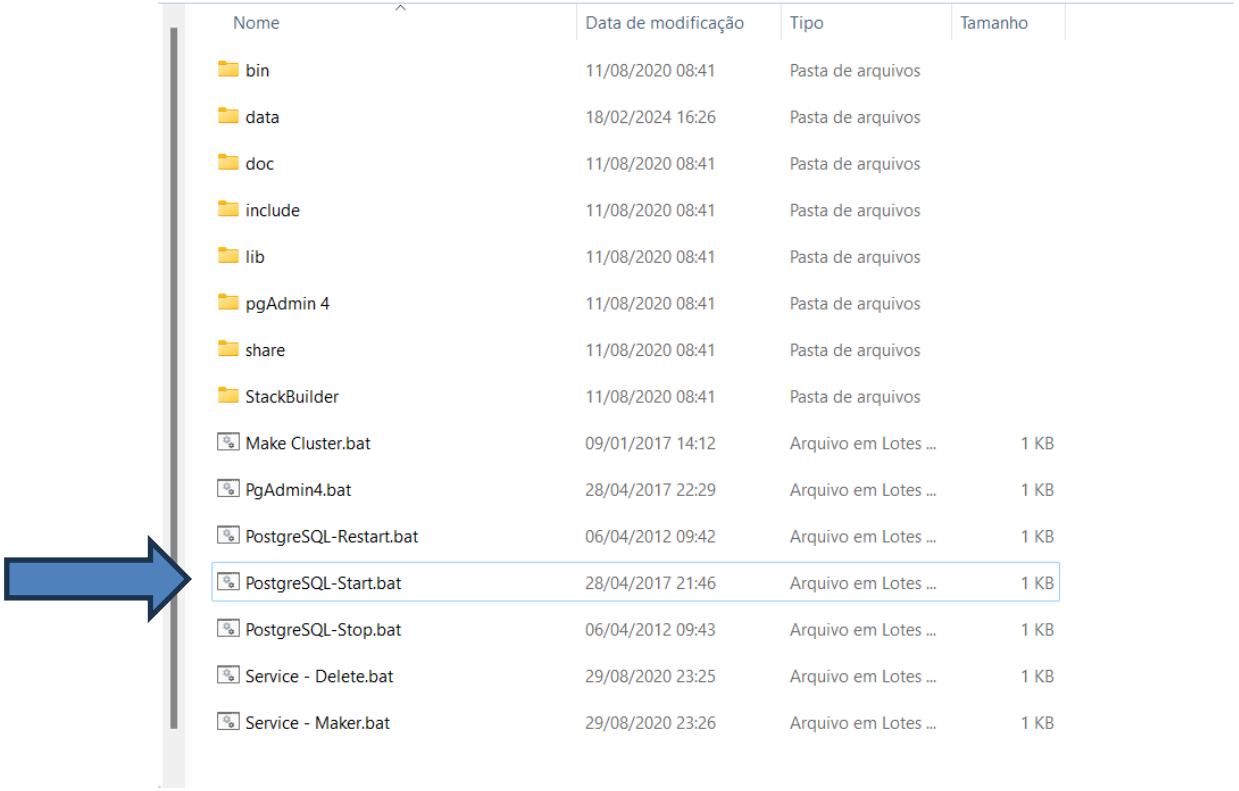


Este passo é importante para configurar as variáveis de ambiente.

Iniciar Servidor Postgresql

PROGRAMAÇÃO BACK-END

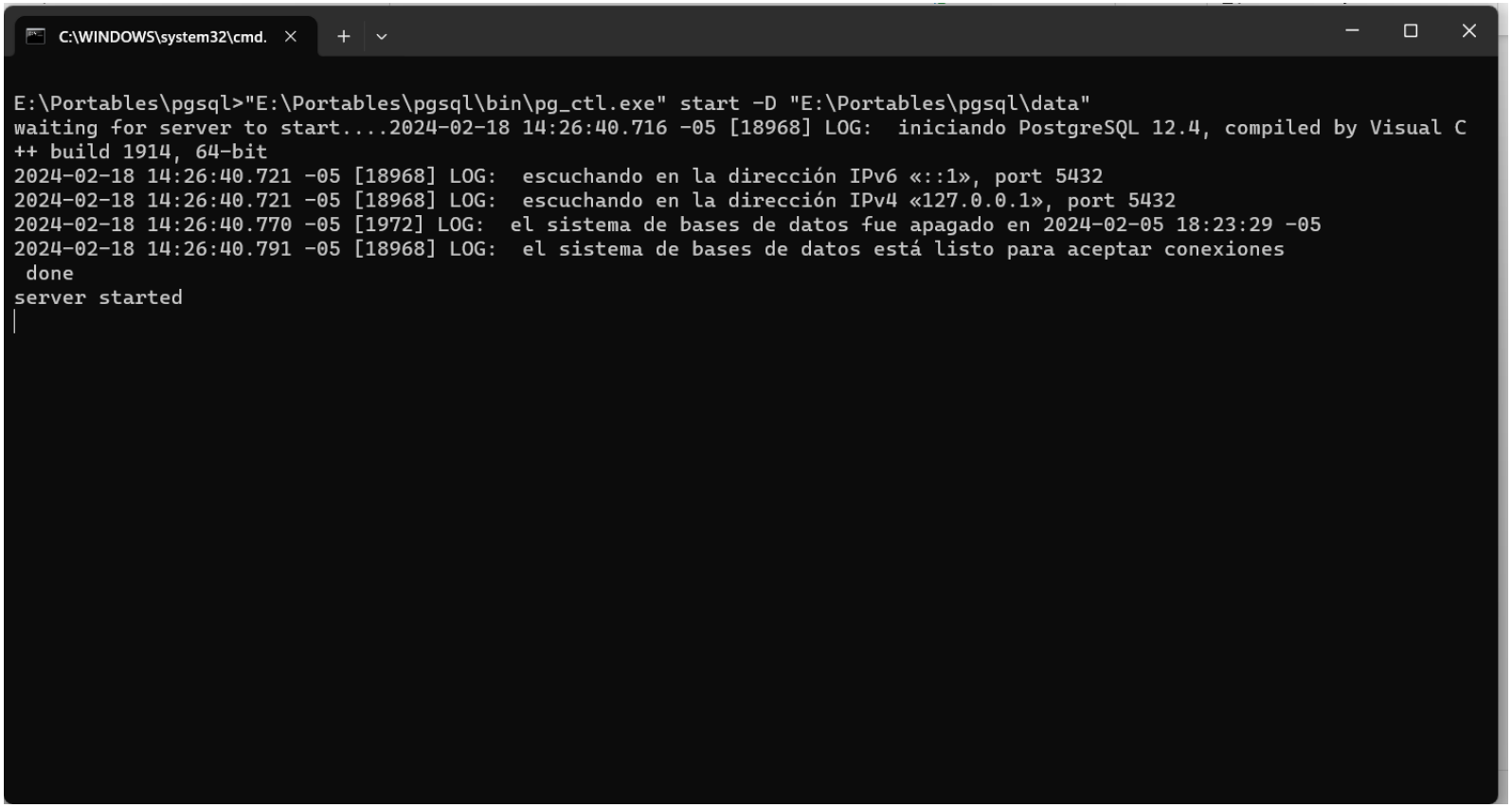
Na pasta ../pgsql execute o arquivo PostgreSQL-Start.bat para iniciar o servidor



Nome	Data de modificação	Tipo	Tamanho
bin	11/08/2020 08:41	Pasta de arquivos	
data	18/02/2024 16:26	Pasta de arquivos	
doc	11/08/2020 08:41	Pasta de arquivos	
include	11/08/2020 08:41	Pasta de arquivos	
lib	11/08/2020 08:41	Pasta de arquivos	
pgAdmin 4	11/08/2020 08:41	Pasta de arquivos	
share	11/08/2020 08:41	Pasta de arquivos	
StackBuilder	11/08/2020 08:41	Pasta de arquivos	
Make Cluster.bat	09/01/2017 14:12	Arquivo em Lotes ...	1 KB
PgAdmin4.bat	28/04/2017 22:29	Arquivo em Lotes ...	1 KB
PostgreSQL-Restart.bat	06/04/2012 09:42	Arquivo em Lotes ...	1 KB
PostgreSQL-Start.bat	28/04/2017 21:46	Arquivo em Lotes ...	1 KB
PostgreSQL-Stop.bat	06/04/2012 09:43	Arquivo em Lotes ...	1 KB
Service - Delete.bat	29/08/2020 23:25	Arquivo em Lotes ...	1 KB
Service - Maker.bat	29/08/2020 23:26	Arquivo em Lotes ...	1 KB

PROGRAMAÇÃO BACK-END

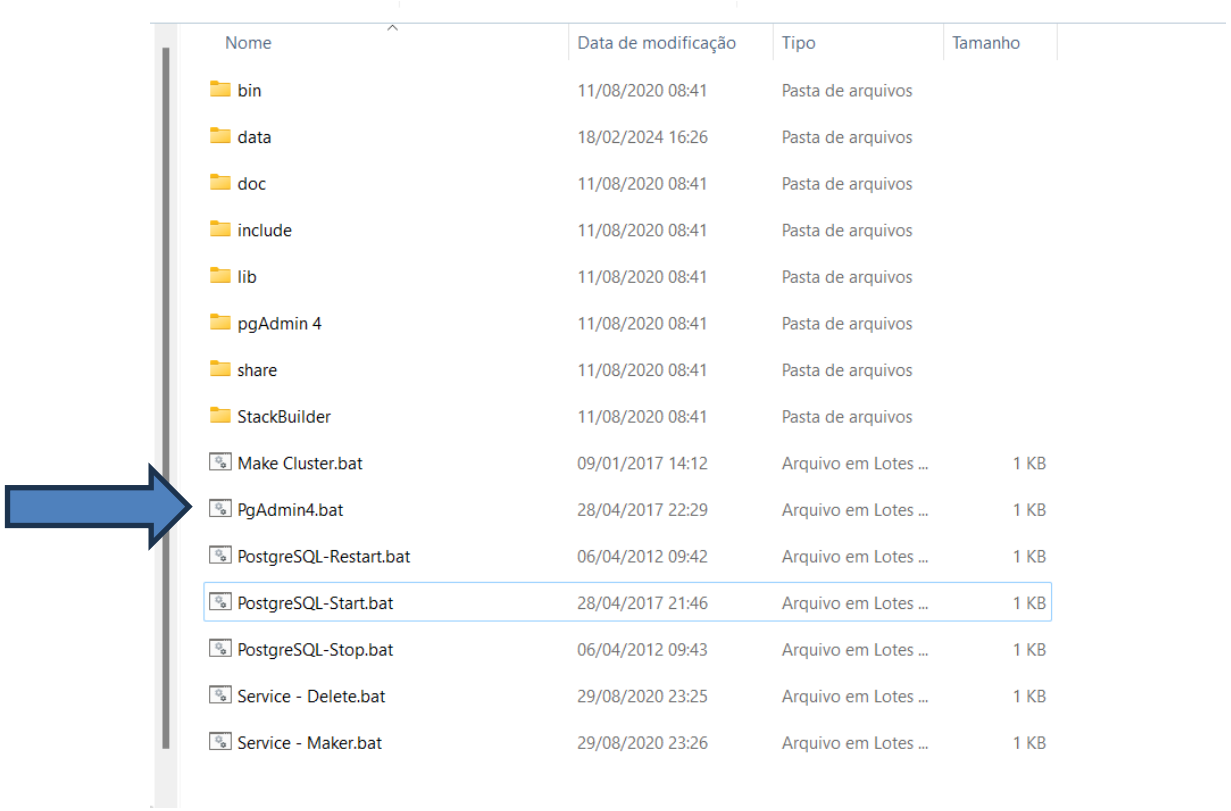
Será aberta uma janela em Texto informando que o servidor iniciou (server started).
Não feche esta janela



Iniciar o ambiente de gerenciamento pgAdmin

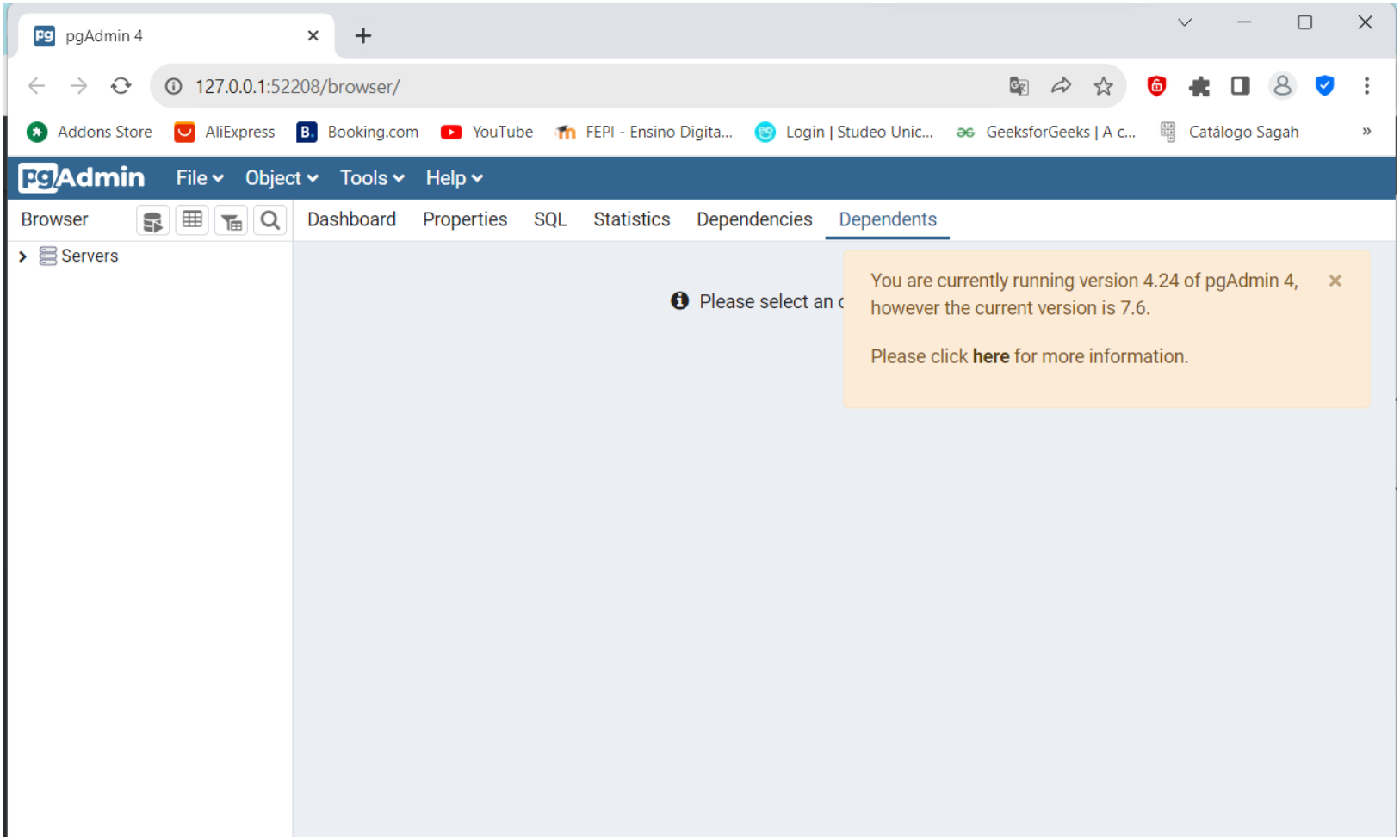
PROGRAMAÇÃO BACK-END

Na pasta ../pgsql execute o arquivo pgAdmin4.bat



PROGRAMAÇÃO BACK-END

O Navegador será aberto conforme a imagem.

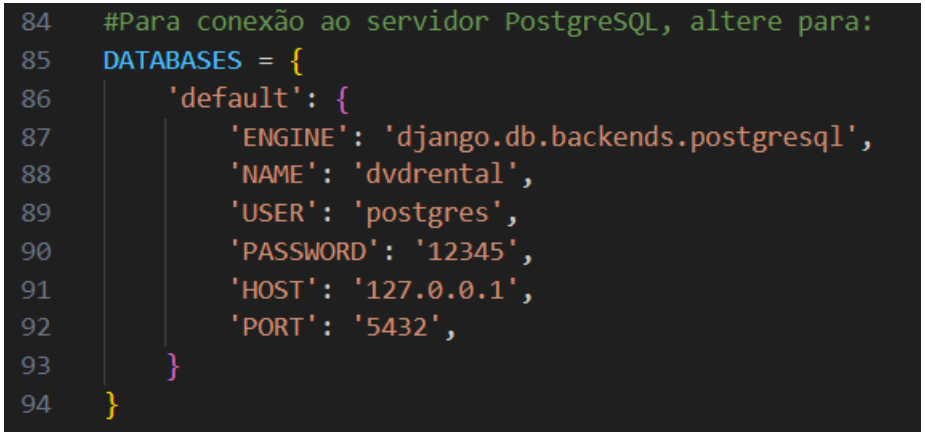


Configurando o arquivo Para conectar no banco

PROGRAMAÇÃO BACK-END

Na pasta my_ads altere a configuração do arquivo settings.py e coloque o conteúdo conforme a seguir:

```
#Para conexão ao servidor PostgreSQL, altere para:
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'dvdrental',
        'USER': 'postgres',
        'PASSWORD': '12345',
        'HOST': '127.0.0.1',
        'PORT': '5432',
    }
}
```



Criando a APP

PROGRAMAÇÃO BACK-END

Criar a APP

Colocaremos o nome do aplicativo de dvdrentalapp.

Comece navegando até o local selecionado onde deseja armazenar o aplicativo, no caso a pasta my_ads, e execute o comando:

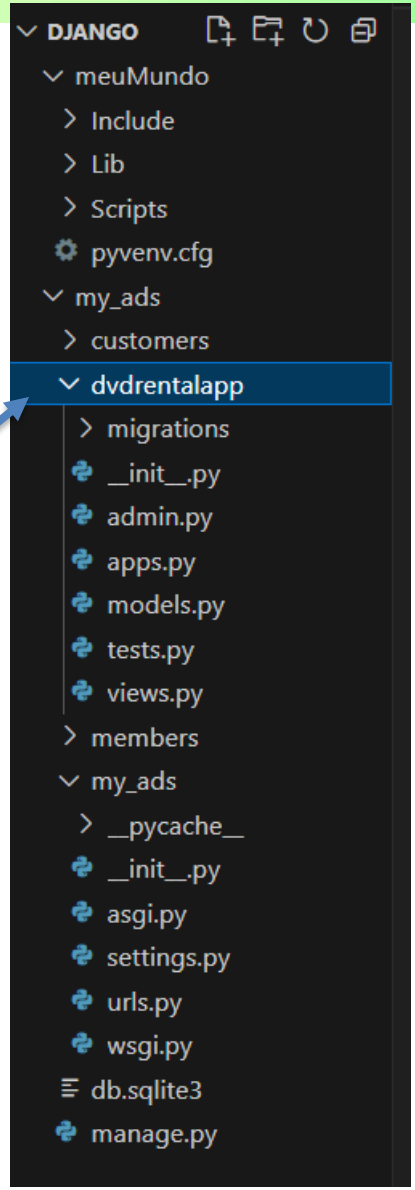
```
python manage.py startapp dvdrentalapp
```

Obs: A(s) outra(s) pasta members criadas para os testes não influenciam a atual criada.

PROGRAMAÇÃO BACK-END

O Comando para criar a app via cmd e seu resultado.

```
(meuMundo) E:\django\my_ads>python manage.py startapp dvdrentalapp

(meuMundo) E:\django\my_ads>
```



Pasta dvdrentalapp criada

Mapeando o banco de dados no Django

PROGRAMAÇÃO BACK-END

Para mapear o banco de dados para o arquivo models.py, faça o seguinte procedimento:

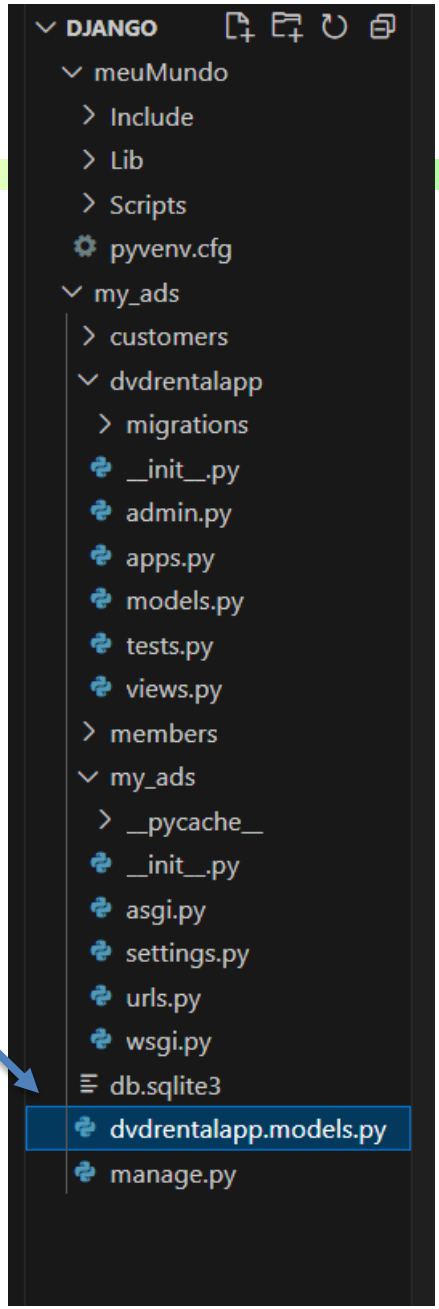
Na pasta my_ads execute o seguinte comando:

python manage.py inspectdb > dvdrentalapp.models.py

```
(meuMundo) E:\django\my_ads>python manage.py inspectdb > dvdrentalapp.models.py

(meuMundo) E:\django\my_ads>
```

PROGRAMAÇÃO BACK-END

O Comando cria o arquivo: dvdrentalapp.models.py na pasta my_ads



PROGRAMAÇÃO BACK-END

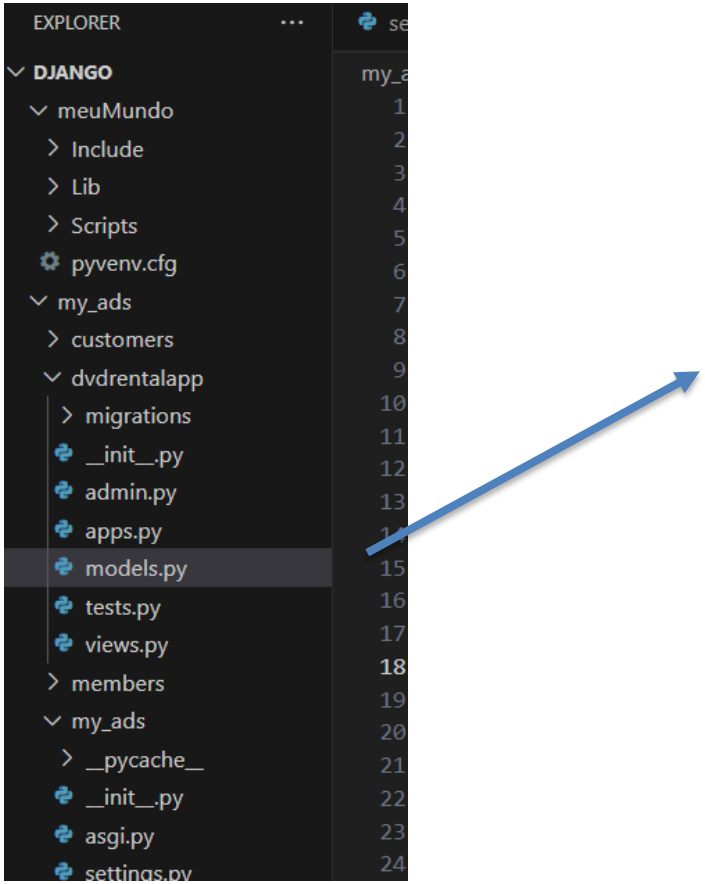
Verificando o conteúdo do arquivo dvdrentalapp.models.py pode-se observar as diversas tabelas que compõe o banco dvdrental e a estrutura de campos de cada tabela.

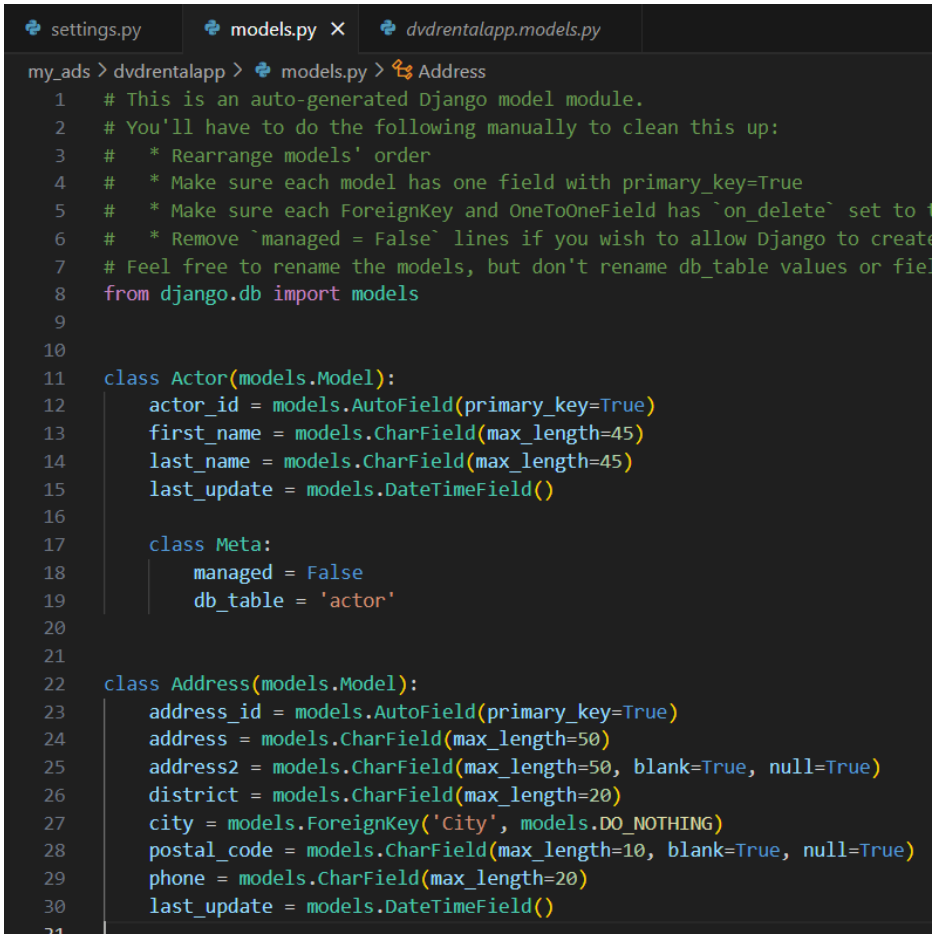
my_ads > dvdrentalapp.models.py > ...

```python
# This is an auto-generated Django model module.
# You'll have to do the following manually to clean this up:
#   * Rearrange models' order
#   * Make sure each model has one field with primary_key=True
#   * Make sure each ForeignKey and OneToOneField has `on_delete` set to the desire
#   * Remove `managed = False` lines if you wish to allow Django to create, modify,
# Feel free to rename the models, but don't rename db_table values or field names.
from django.db import models


class Actor(models.Model):
    actor_id = models.AutoField(primary_key=True)
    first_name = models.CharField(max_length=45)
    last_name = models.CharField(max_length=45)
    last_update = models.DateTimeField()

    class Meta:
        managed = False
        db_table = 'actor'


class Address(models.Model):
    address_id = models.AutoField(primary_key=True)
    address = models.CharField(max_length=50)
    address2 = models.CharField(max_length=50, blank=True, null=True)
    district = models.CharField(max_length=20)
    city = models.ForeignKey('City', models.DO_NOTHING)
    postal_code = models.CharField(max_length=10, blank=True, null=True)
    phone = models.CharField(max_length=20)
    last_update = models.DateTimeField()

    class Meta:
        managed = False
        db_table = 'address'
```

PROGRAMAÇÃO BACK-END

Faça a cópia do conteúdo do arquivo dvdrentalapp.models.py para o arquivo models.py que está na pasta dvdrentalapp.



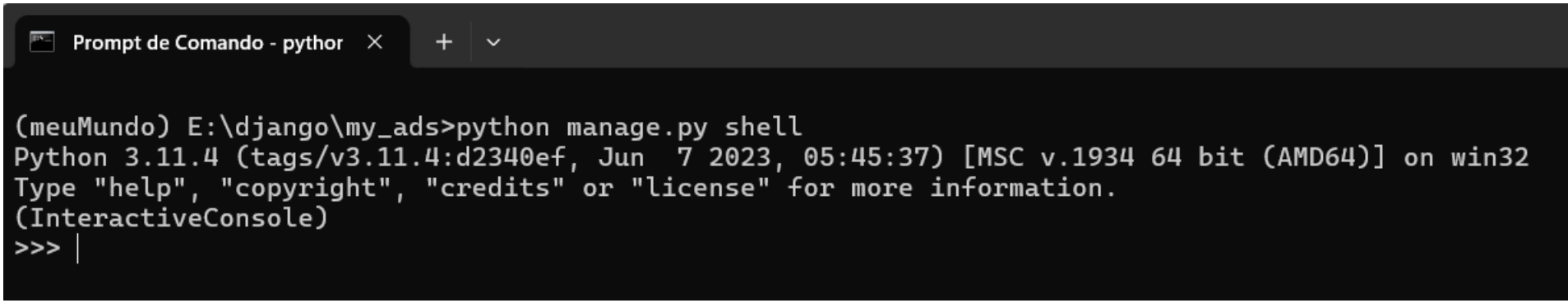


Testando a conexão com o banco

PROGRAMAÇÃO BACK-END

Para testar a conexão, deve-se entrar pelo shell. Execute o seguinte comando:

my_ads> python manage.py shell

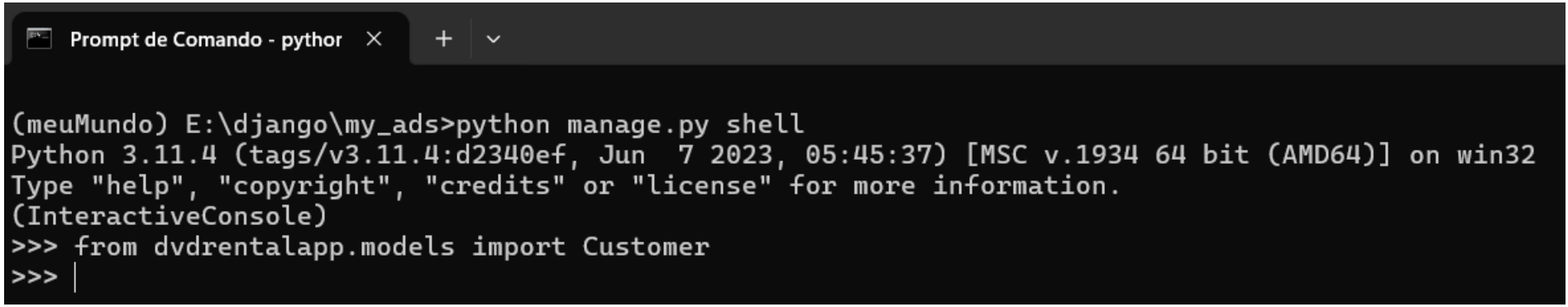


PROGRAMAÇÃO BACK-END

Execute o seguinte comando:

>>> from dvdrentalapp.models import customer

Este comando está importando a tabela Customer da APP dvdrentalapp no models (que está no arquivo models.py)



PROGRAMAÇÃO BACK-END

Execute o seguinte comando:

>>> Customer.objects.all().values()

Este comando irá mostrar todo conteúdo da tabela Customer conforme mostra a seguir.

```
>>> Customer.objects.all().values()
<QuerySet [{'customer_id': 524, 'store_id': 1, 'first_name': 'Jared', 'last_name': 'Ely', 'email':
omer.org', 'address_id': 530, 'activebool': True, 'create_date': datetime.date(2006, 2, 14), 'last_
etime(2013, 5, 26, 14, 49, 45, 738000), 'active': 1}, {'customer_id': 1, 'store_id': 1, 'first_name
': 'Smith', 'email': 'mary.smith@sakilacustomer.org', 'address_id': 5, 'activebool': True, 'create_
2006, 2, 14), 'last_update': datetime.datetime(2013, 5, 26, 14, 49, 45, 738000), 'active': 1}, {'cus
_id': 1, 'first_name': 'Patricia', 'last_name': 'Johnson', 'email': 'patricia.johnson@sakilacustome
 6, 'activebool': True, 'create_date': datetime.date(2006, 2, 14), 'last_update': datetime.datetime(
 45, 738000), 'active': 1}, {'customer_id': 3, 'store_id': 1, 'first_name': 'Linda', 'last_name': '
```

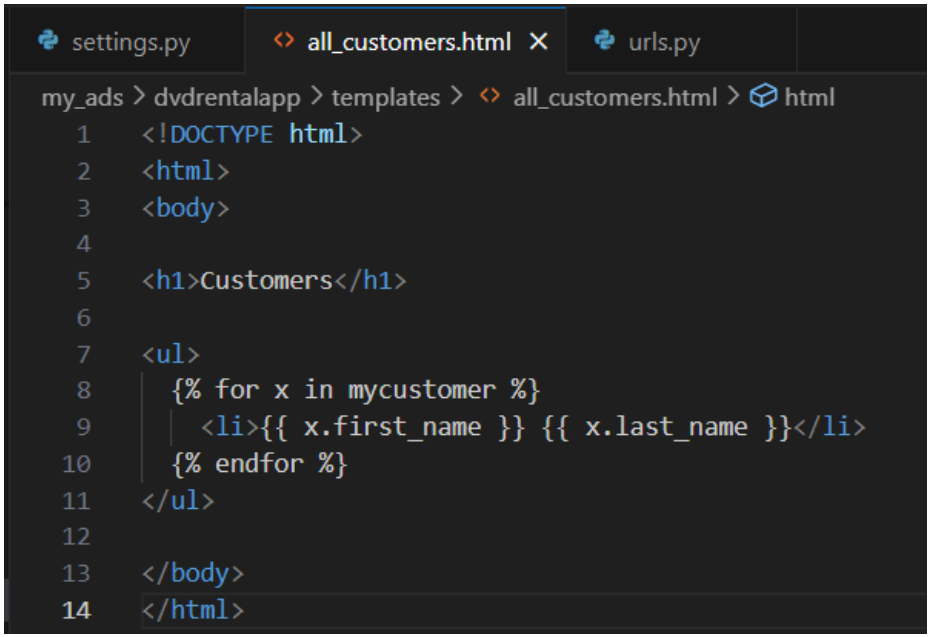
Para sair digite

>>> exit()

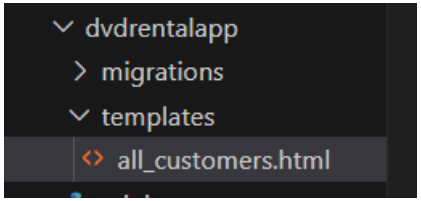
Criando os arquivos na pasta templates

PROGRAMAÇÃO BACK-END

Na pasta dvdrentalapp crie a pasta templates e crie um arquivo all_customers.html com o seguinte conteúdo:




```
<!DOCTYPE html>
<html>
<body>

<h1>Customers</h1>

<ul>
  {% for x in mycustomer %}
    <li>{{ x.first_name }} {{ x.last_name }}</li>
  {% endfor %}
</ul>

</body>
</html>
```



PROGRAMAÇÃO BACK-END

Os colchetes {% %} dentro do documento HTML são Tags do Django, dizendo ao Django para executar alguma lógica de programação dentro desses colchetes.

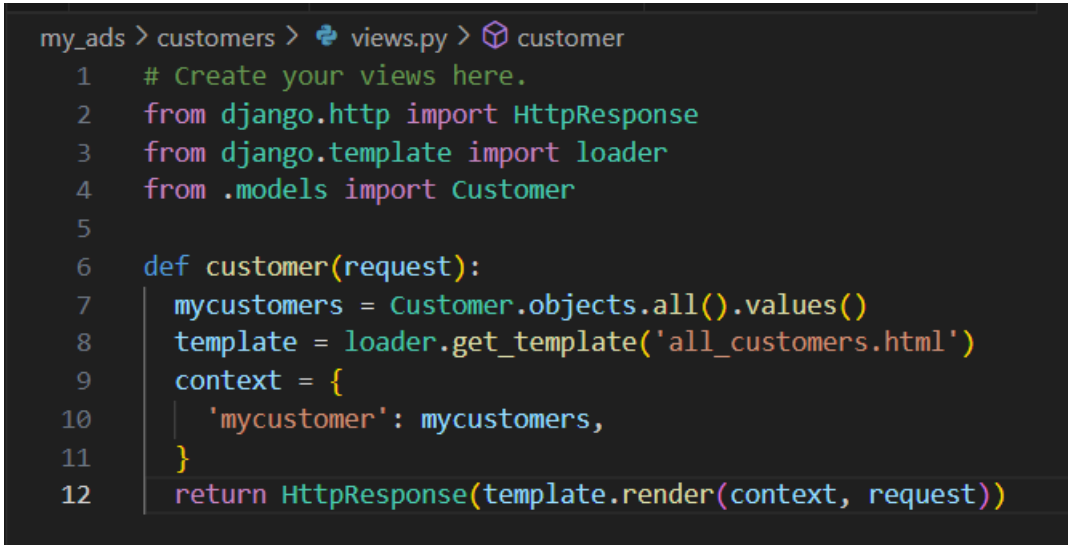
Preparando os arquivos view.py, urls.py

PROGRAMAÇÃO BACK-END

Na pasta dvdreantalapp altere o arquivo views.py com o seguinte conteúdo:

```
#from django.shortcuts import render

# Create your views here.
from django.http import HttpResponse
from django.template import loader
from .models import Customer

def customer(request):
  mycustomers = Customer.objects.all().values()
  template = loader.get_template('all_customers.html')
  context = {
    'mycustomer': mycustomers,
  }
  return HttpResponse(template.render(context, request))
```

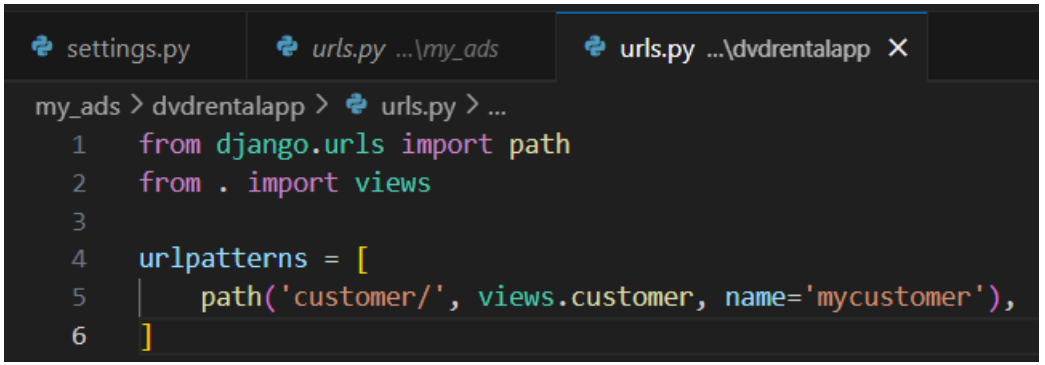
PROGRAMAÇÃO BACK-END



PROGRAMAÇÃO BACK-END

Na pasta dvdreantalapp crie um arquivo urls.py com o seguinte conteúdo:

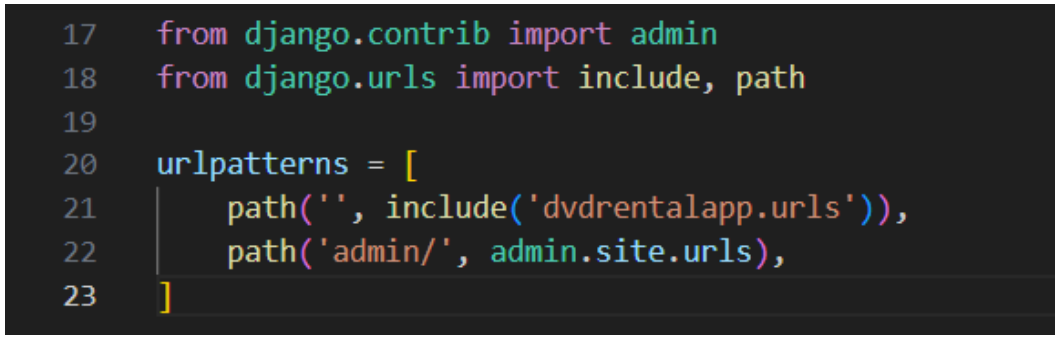
```
from django.urls import path
from . import views

urlpatterns = [
    path('customer/', views.customer,
name='mycustomer'),
]
```



PROGRAMAÇÃO BACK-END

Na pasta my_ads altere o arquivo urls.py com o seguinte conteúdo:

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('', include('dvdrentalapp.urls')),
    path('admin/', admin.site.urls),
]
```



PROGRAMAÇÃO BACK-END

Na pasta my_ads altere o arquivo settings.py acrescentando o APP dvdreantalapp

```
33  INSTALLED_APPS = [
34      'django.contrib.admin',
35      'django.contrib.auth',
36      'django.contrib.contenttypes',
37      'django.contrib.sessions',
38      'django.contrib.messages',
39      'django.contrib.staticfiles',
40      'dvdrentalapp',
41  ]
```

Execute na pasta my_ads o comando:

python manage.py migrate

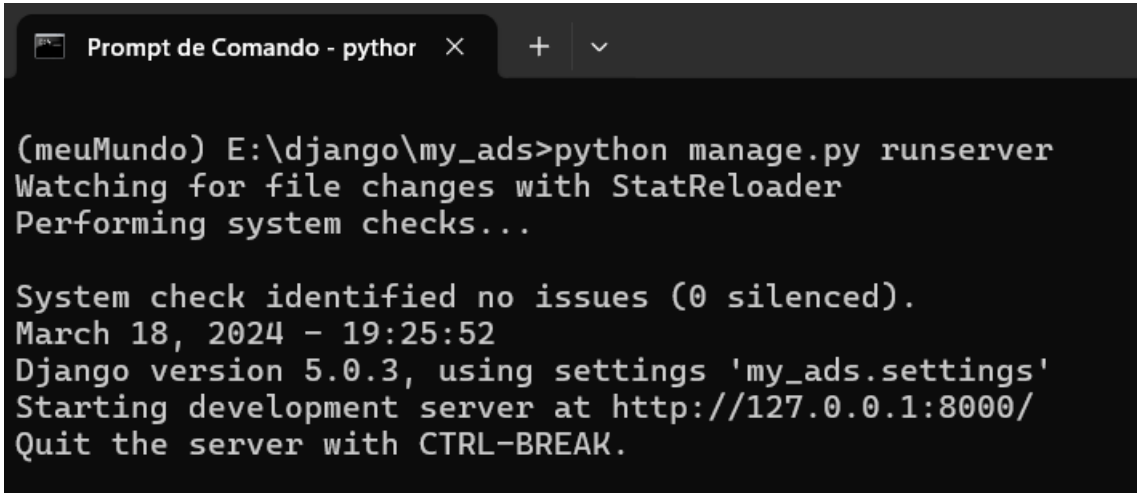
```
(meuMundo) E:\django\my_ads>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK

(meuMundo) E:\django\my_ads>
```

Iniciando o servidor e testando

PROGRAMAÇÃO BACK-END

Execute na pasta my_ads o comando:

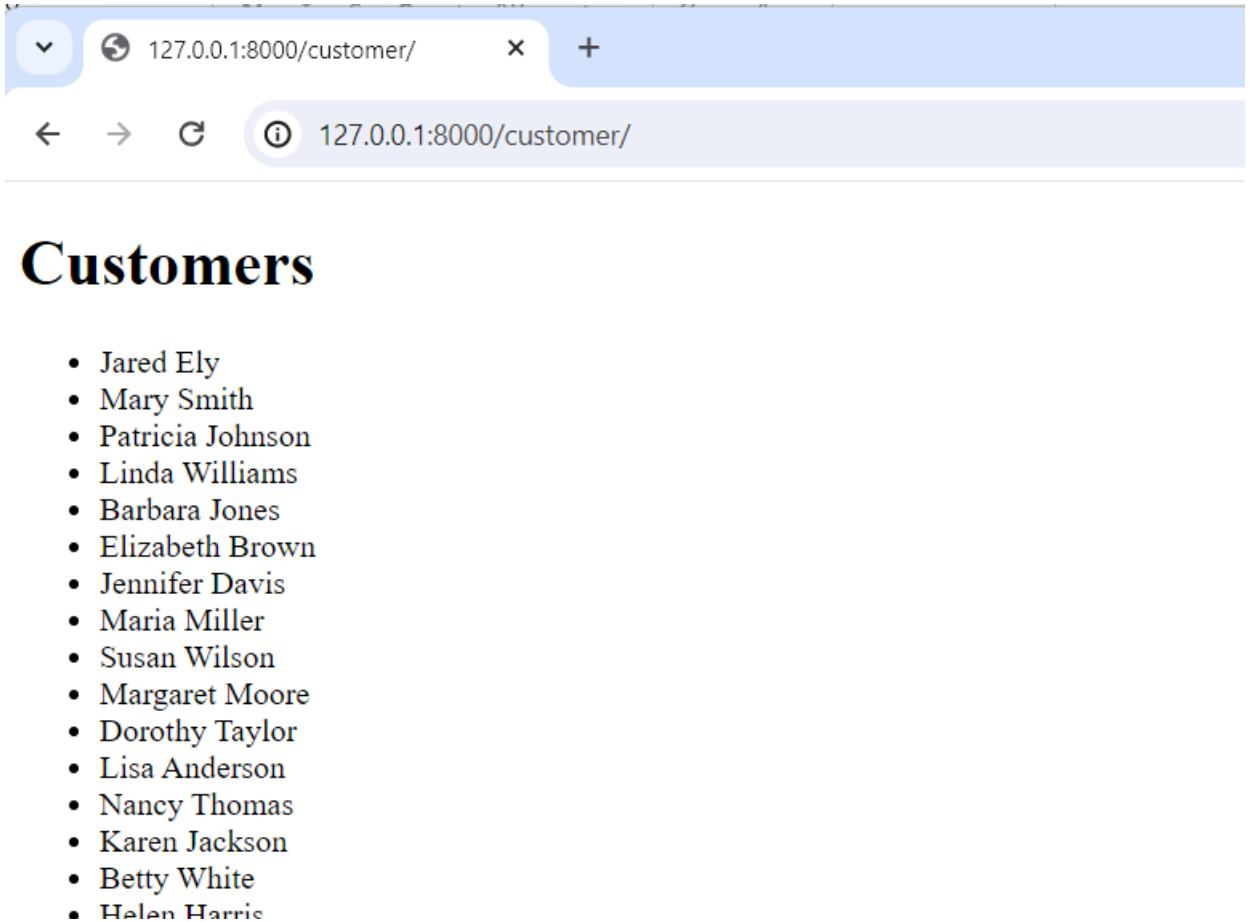
python manage.py runserver



Vá ao navegador e digite:

http://127.0.0.1:8000/customer/

O resultado no navegador



Desafio

PROGRAMAÇÃO BACK-END

Crie outra APP que acesse outra tabela do banco, mostrando os seus dados.