

PROGRAMAÇÃO BACK-END

django

The web framework for
perfectionists with deadlines.

PROGRAMAÇÃO BACK-END

Conectar o Django
no PostgreSQL

PROGRAMAÇÃO BACK-END

Objetivos desta apresentação

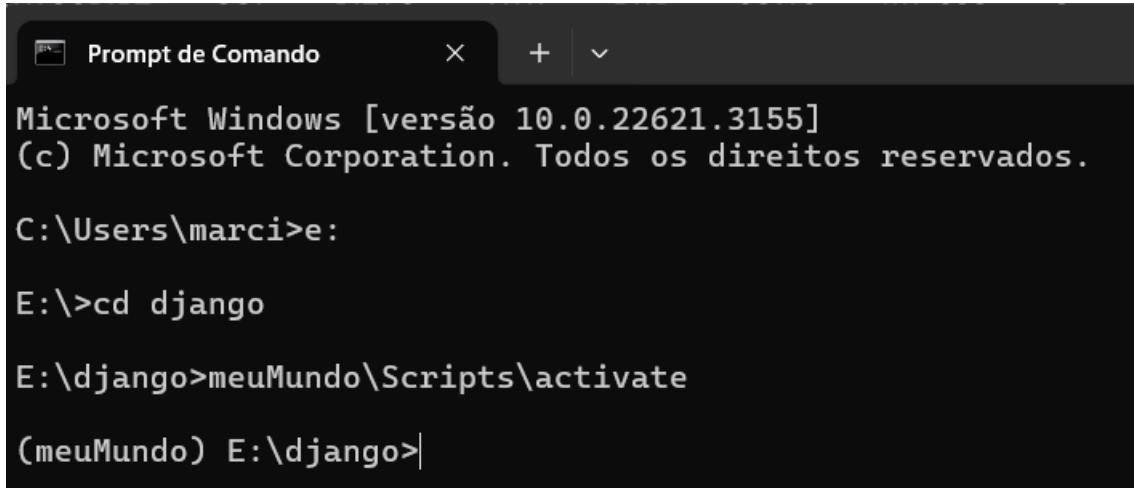
Preparar um aplicativo **dvdrentalapp** para conectar no banco **dvdrental** do PostgreSQL e apresentar os dados da tabela **Customer** e realizar a edição de alguns campos.

PROGRAMAÇÃO BACK-END

Iniciando

PROGRAMAÇÃO BACK-END

Rodar o script **activate.bat**



```
Prompt de Comando
Microsoft Windows [versão 10.0.22621.3155]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\marci>e:

E:\>cd django

E:\django>meuMundo\Scripts\activate

(meuMundo) E:\django>
```

Este passo é importante para configurar as variáveis de ambiente.

PROGRAMAÇÃO BACK-END

Iniciar Servidor Postgresql

PROGRAMAÇÃO BACK-END

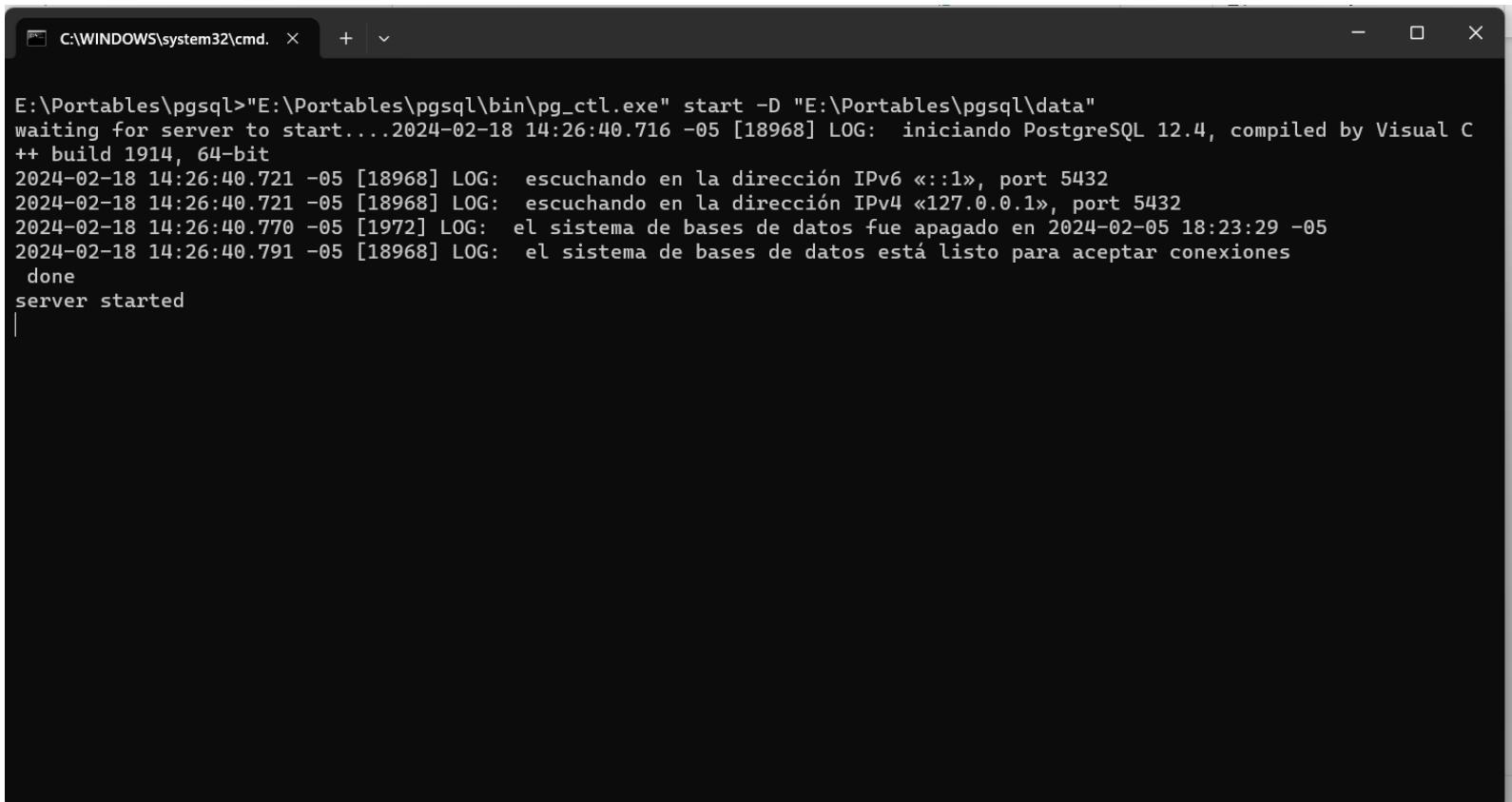
Na pasta `../pgsql` execute o arquivo `PostgreSQL-Start.bat` para iniciar o servidor



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**



A screenshot of a Windows Command Prompt window titled "C:\WINDOWS\system32\cmd.". The window contains the following text output from a PostgreSQL server start command:

```
E:\Portables\pgsql>"E:\Portables\pgsql\bin\pg_ctl.exe" start -D "E:\Portables\pgsql\data"
waiting for server to start....2024-02-18 14:26:40.716 -05 [18968] LOG:  iniciando PostgreSQL 12.4, compiled by Visual C
++ build 1914, 64-bit
2024-02-18 14:26:40.721 -05 [18968] LOG:  escuchando en la dirección IPv6 «::1», port 5432
2024-02-18 14:26:40.721 -05 [18968] LOG:  escuchando en la dirección IPv4 «127.0.0.1», port 5432
2024-02-18 14:26:40.770 -05 [1972] LOG:  el sistema de bases de datos fue apagado en 2024-02-05 18:23:29 -05
2024-02-18 14:26:40.791 -05 [18968] LOG:  el sistema de bases de datos está listo para aceptar conexiones
  done
server started
```

PROGRAMAÇÃO BACK-END

Iniciar o ambiente
de gerenciamento pgAdmin

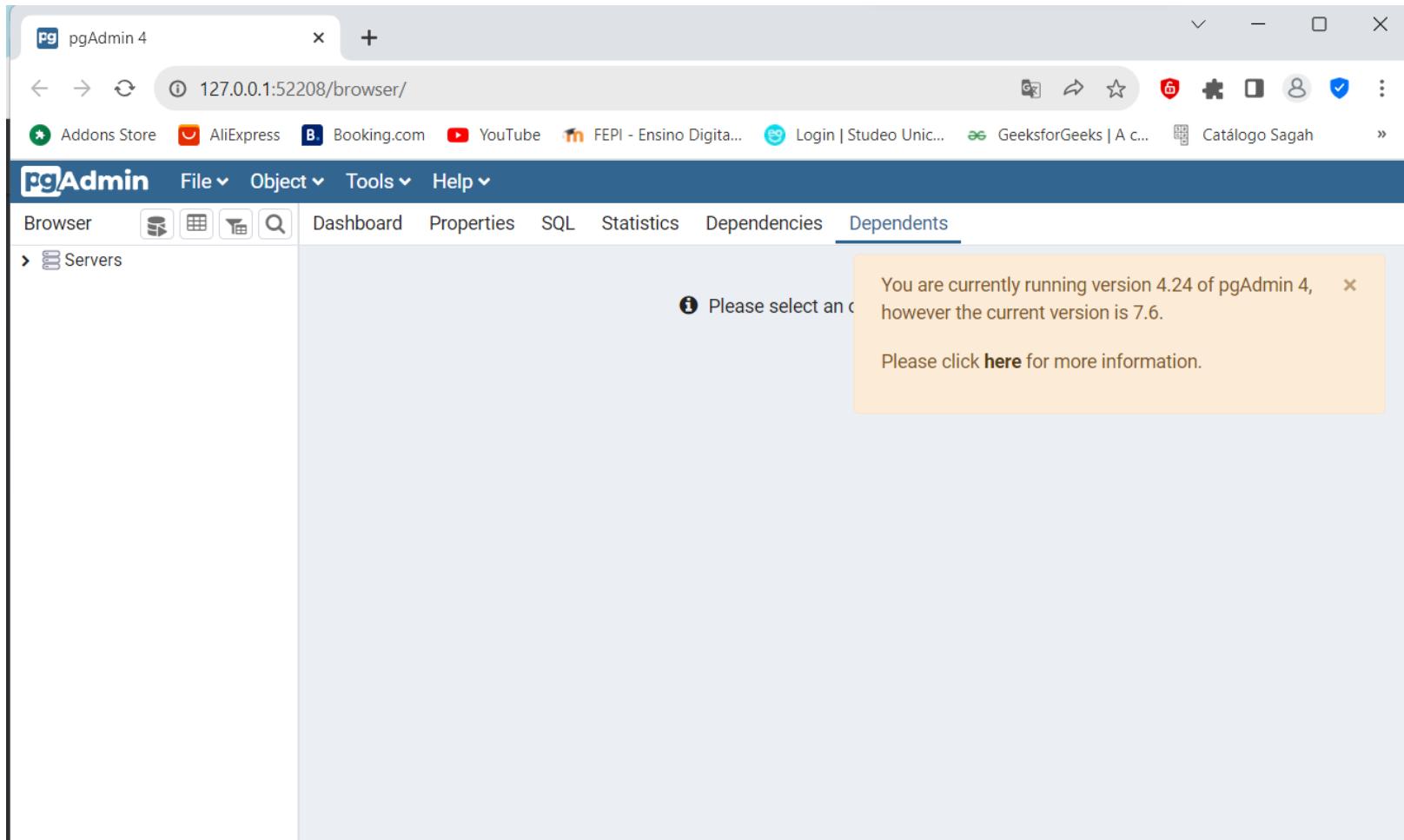
PROGRAMAÇÃO BACK-END

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

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

O Navegador será aberto conforme a imagem.



PROGRAMAÇÃO BACK-END

1. Configurando o arquivo Para conectar no banco

**Caso esta etapa já foi realizada
não é necessário fazer.**

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',  
     }  
 }
```

```
84 #Para conexão ao servidor PostgreSQL, altere para:  
85 DATABASES = {  
86     'default': {  
87         'ENGINE': 'django.db.backends.postgresql',  
88         'NAME': 'dvdrental',  
89         'USER': 'postgres',  
90         'PASSWORD': '12345',  
91         'HOST': '127.0.0.1',  
92         'PORT': '5432',  
93     }  
94 }
```

PROGRAMAÇÃO BACK-END

2. Criando a APP

**Caso esta etapa já foi realizada
não é necessário fazer.**

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

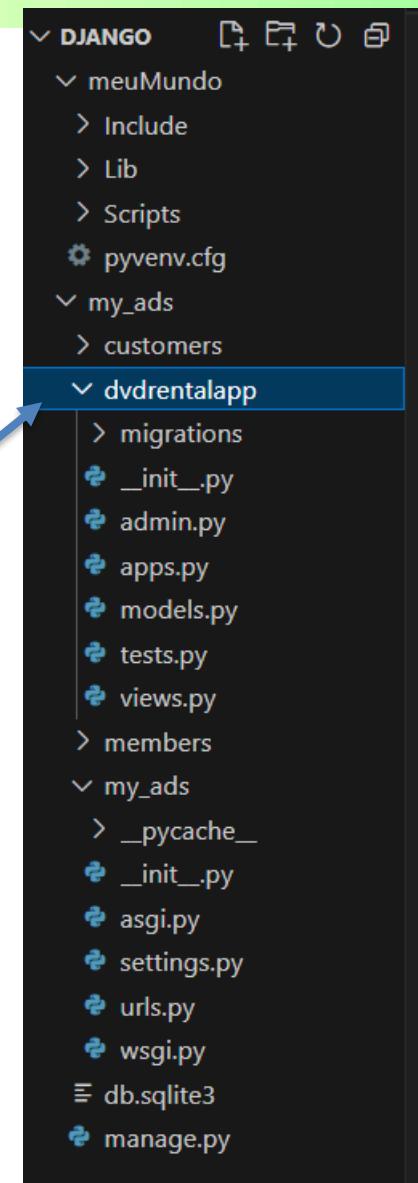
Obs: A(s) outra(s) pasta members criadas para os testes não influenciam a atual criada.

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



PROGRAMAÇÃO BACK-END

3. Mapeando o banco de dados no Django

**Caso esta etapa já foi realizada
não é necessário fazer.**

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:

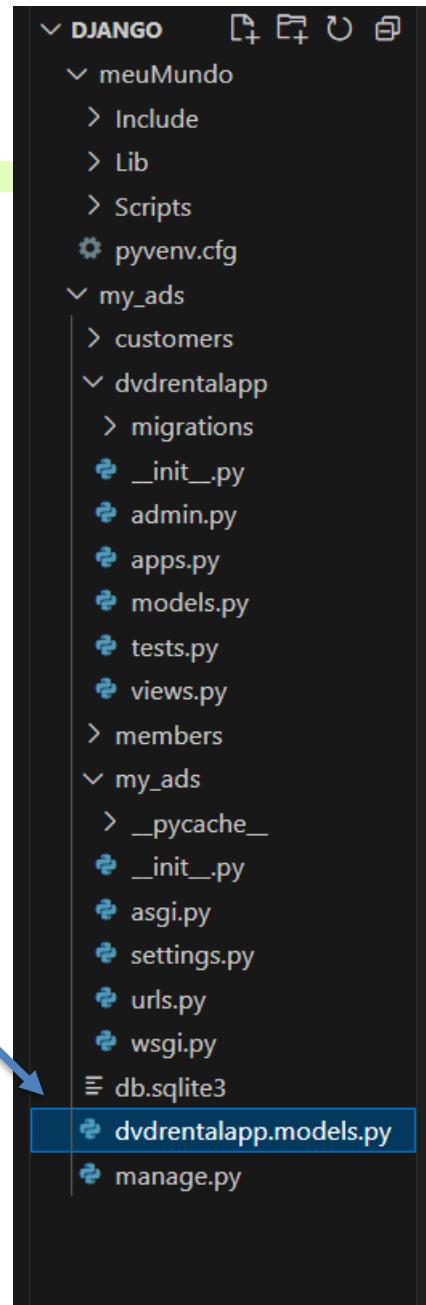
`python manage.py inspectdb > dvdrentalapp.models.py`

```
(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**



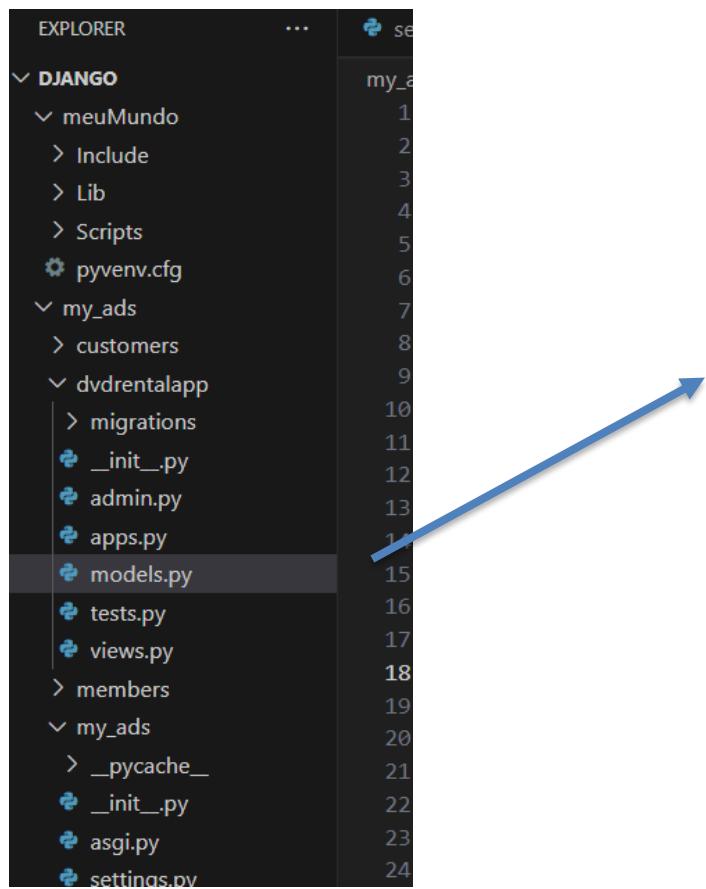
PROGRAMAÇÃO BACK-END

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 > dvrentalapp.models.py > ...
1  # This is an auto-generated Django model module.
2  # You'll have to do the following manually to clean this up:
3  #   * Rearrange models' order
4  #   * Make sure each model has one field with primary_key=True
5  #   * Make sure each ForeignKey and OneToOneField has `on_delete` set to the desire
6  #   * Remove `managed = False` lines if you wish to allow Django to create, modify,
7  #     # Feel free to rename the models, but don't rename db_table values or field names.
8  from django.db import models
9
10
11 class Actor(models.Model):
12     actor_id = models.AutoField(primary_key=True)
13     first_name = models.CharField(max_length=45)
14     last_name = models.CharField(max_length=45)
15     last_update = models.DateTimeField()
16
17     class Meta:
18         managed = False
19         db_table = 'actor'
20
21
22 class Address(models.Model):
23     address_id = models.AutoField(primary_key=True)
24     address = models.CharField(max_length=50)
25     address2 = models.CharField(max_length=50, blank=True, null=True)
26     district = models.CharField(max_length=20)
27     city = models.ForeignKey('City', models.DO_NOTHING)
28     postal_code = models.CharField(max_length=10, blank=True, null=True)
29     phone = models.CharField(max_length=20)
30     last_update = models.DateTimeField()
31
32     class Meta:
33         managed = False
34         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**.



PROGRAMAÇÃO BACK-END

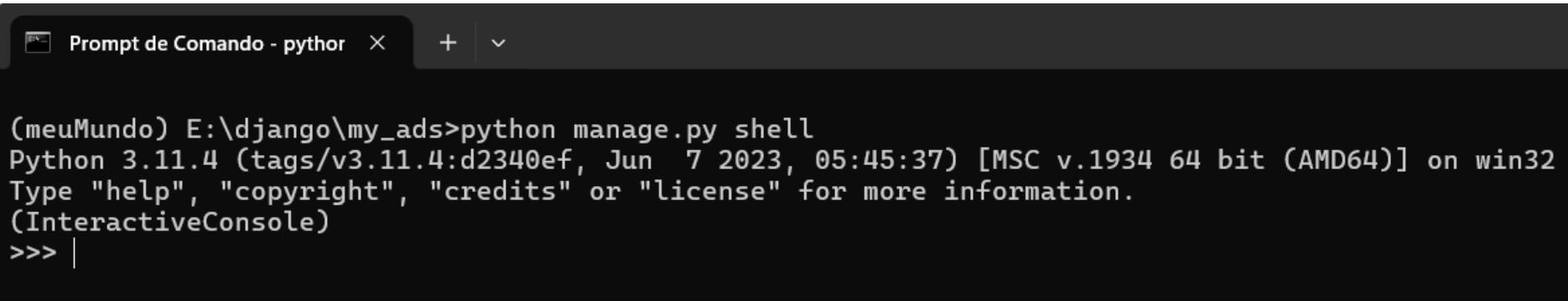
4. Testando a conexão com o banco

**Caso esta etapa já foi realizada
não é necessário fazer.**

PROGRAMAÇÃO BACK-END

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

my_ads> python manage.py shell



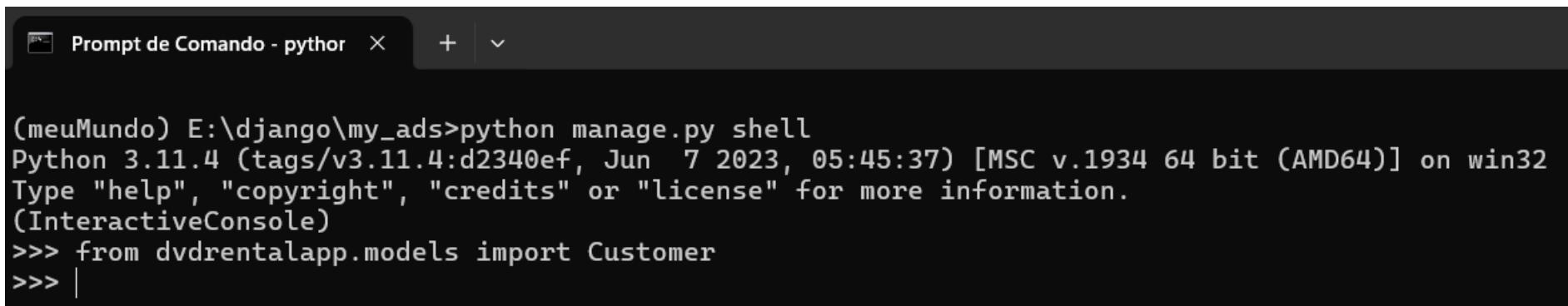
```
(meuMundo) E:\django\my_ads>python manage.py shell
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun  7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> |
```

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`)



A screenshot of a terminal window titled "Prompt de Comando - python". The window shows the following command-line session:

```
(meuMundo) E:\django\my_ads>python manage.py shell
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun  7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from dvdrentalapp.models import Customer
>>> |
```

PROGRAMAÇÃO BACK-END

Execute o seguinte comando:

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

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

Para sair digite

```
>>> exit()
```

PROGRAMAÇÃO BACK-END

5. Criando os arquivos na
pasta templates

PROGRAMAÇÃO BACK-END

5.1. Arquivo all_customers.html

PROGRAMAÇÃO BACK-END

Na pasta `dvdrentalapp` subpasta `templates` crie/altere o arquivo `all_customers.html` com o seguinte conteúdo:

```
all_customers.html X detalhes.html urls.py dvdrentalapp urls.py my_ads views.py
dvdrentalapp > templates > all_customers.html > html
1  <!DOCTYPE html>
2  <html>
3  <body>
4
5  <h1>Customers</h1>
6
7  <ul>
8  |  {% for x in mycustomer %}
9  |  |  <li><a href="/edit_customer/{{x.customer_id}}">{{ x.first_name }} {{ x.last_name }}</a></li>
10 |  |  {% endfor %}
11 </ul>
12
13 </body>
14 </html>
```

PROGRAMAÇÃO BACK-END

Verifique que no arquivo `all_customers.html`, foi acrescentado a tag `href`, do html, que permite direcionar para um nova chamada html (`edit_customer.html`), passando o `customer_id` que server de filtro da tabela e editar os dados deste filtro.

PROGRAMAÇÃO BACK-END

5.2. Arquivo edit_customer.html

PROGRAMAÇÃO BACK-END

Na pasta `dvdrentalapp` subpasta `templates` crie o arquivo `edit_customer.html` com o seguinte conteúdo:

PROGRAMAÇÃO BACK-END

all_customers.html

edit_customer.html X

urls.py

views.py

dvdrentalapp > templates > edit_customer.html > ...

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  |    <title>Editar Cliente</title>
5  </head>
6  <body>
7
8  <h2>Editar Cliente</h2>
9
10 <form method="post">
11     {% csrf_token %}
12     <label for="first_name">Primeiro Nome:</label><br>
13     <input type="text" id="first_name" name="first_name" value="{{ customer.first_name }}"><br>
14     <label for="last_name">Último Nome:</label><br>
15     <input type="text" id="last_name" name="last_name" value="{{ customer.last_name }}"><br>
16     <label for="email">Email:</label><br>
17     <input type="email" id="email" name="email" value="{{ customer.email }}"><br>
18     <!-- Adicione outros campos conforme necessário -->
19     <input type="submit" value="Salvar Alterações">
20 </form>
21
22 </body>
23 </html>
```

PROGRAMAÇÃO BACK-END

O `{% csrf_token %}` no arquivo `edit_customer.html` (ou em qualquer outro template HTML usado em um projeto Django) refere-se a uma medida de segurança importante chamada "Cross Site Request Forgery protection", abreviadamente CSRF. CSRF é um tipo de ataque na web onde um usuário mal-intencionado induz os usuários a executarem ações indesejadas em um site no qual estão atualmente autenticados, sem que eles saibam.

PROGRAMAÇÃO BACK-END

Verifique que no arquivo `edit_customer.html`, foi acrescentado os **campos necessários para a edição com o botão que será utilizado para salvar as alterações.**

PROGRAMAÇÃO BACK-END

Preparando os arquivos
view.py, urls.py

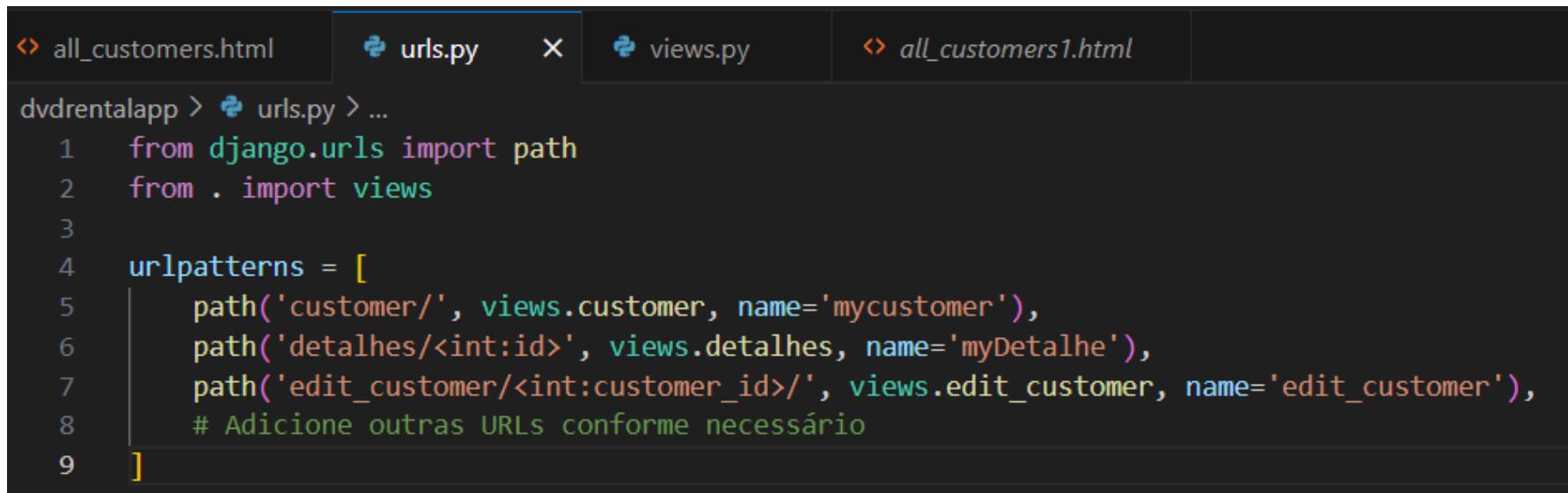
PROGRAMAÇÃO

**Na pasta
dvdrental
app altere
o arquivo
views.py
com o
seguinte
conteúdo:**

```
dvdrentalapp > views.py > ...
1 # Create your views here.
2 from django.http import HttpResponseRedirect
3 from django.template import loader
4 from .models import Customer, Rental
5 from django.shortcuts import render, get_object_or_404, redirect
6
7 def customer(request):
8     mycustomers = Customer.objects.all().values()
9     template = loader.get_template('all_customers.html')
10    context = {
11        'mycustomer': mycustomers,
12    }
13    return HttpResponseRedirect(template.render(context, request))
14
15 def edit_customer(request, customer_id):
16     customer = get_object_or_404(Customer, pk=customer_id)
17     if request.method == "POST":
18         customer.first_name = request.POST.get('first_name')
19         customer.last_name = request.POST.get('last_name')
20         customer.email = request.POST.get('email')
21         # Atualize outros campos conforme necessário
22         customer.save()
23         return HttpResponseRedirect('/customer') #
24     return render(request, 'edit_customer.html', {'customer': customer})
25
26 def detalhes(request, id):
27     myDetalhes = Rental.objects.filter(customer_id=id)
28     customer = get_object_or_404(Customer, pk=id)
29     template = loader.get_template('detalhes.html')
30     context = {
31         'myDetalhe': myDetalhes,
32         'customer_name' : f'{customer.first_name} {customer.last_name}',
33     }
34     return HttpResponseRedirect(template.render(context, request))
35
```

PROGRAMAÇÃO BACK-END

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



The screenshot shows a code editor with four tabs at the top: "all_customers.html", "urls.py", "views.py", and "all_customers1.html". The "urls.py" tab is active. The code in the editor is as follows:

```
dvdrentalapp > urls.py > ...
1  from django.urls import path
2  from . import views
3
4  urlpatterns = [
5      path('customer/', views.customer, name='mycustomer'),
6      path('detalhes/<int:id>', views.detalhes, name='myDetalhe'),
7      path('edit_customer/<int:customer_id>/', views.edit_customer, name='edit_customer'),
8      # Adicione outras URLs conforme necessário
9 ]
```

Verifique que `edit_customer/<int:customer_id>/` será passado um id para filtragem.

PROGRAMAÇÃO BACK-END

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

```
17 from django.contrib import admin  
18 from django.urls import include, path  
19  
20 urlpatterns = [  
21     path('', include('dvdrentalapp.urls')),  
22     path('admin/', admin.site.urls),  
23 ]
```

**Caso esta etapa já foi realizada
não é necessário fazer.**

PROGRAMAÇÃO BACK-END

Na pasta `my_ads` altere/verifique o arquivo `settings.py` acrescentando o APP `dvdrentalapp`

```
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     ]
```

**Caso esta etapa já foi realizada
não é necessário fazer.**

PROGRAMAÇÃO BACK-END

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>
```

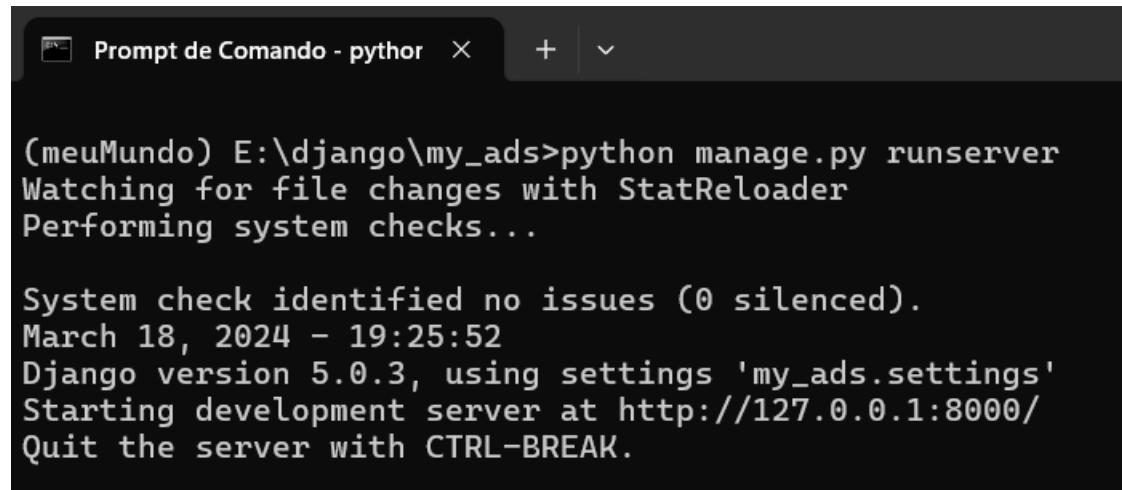
PROGRAMAÇÃO BACK-END

Iniciando o servidor
e testando

PROGRAMAÇÃO BACK-END

Execute na pasta **my_ads** o comando:

python manage.py runserver



```
(meuMundo) E:\django\my_ads>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

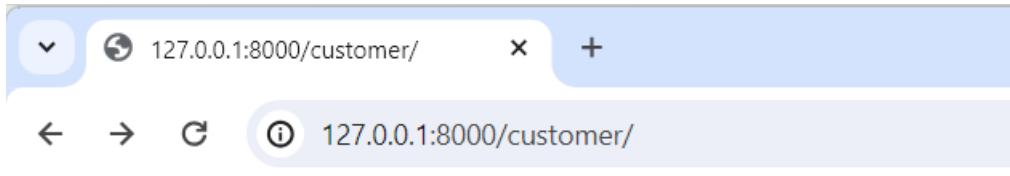
System check identified no issues (0 silenced).
March 18, 2024 - 19:25:52
Django version 5.0.3, using settings 'my_ads.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Vá ao navegador e digite:

http://127.0.0.1:8000/customer/

PROGRAMAÇÃO BACK-END

O resultado no navegador



Customers

- [Linda Williams](#)
- [Barbara Jones](#)
- [Elizabeth Brown](#)
- [Jennifer Davis](#)
- [Maria Miller](#)
- [Susan Wilson](#)
- [Margaret Moore](#)
- [Dorothy Taylor](#)
- [Lisa Anderson](#)
- [Nancy Thomas](#)
- [Karen Jackson](#)
- [Betty White](#)
- [Helen Harris](#)
- [Sandra Martin](#)
- [Donna Thompson](#)
- [Carol Garcia](#)
- [Ruth Martinez](#)
- [Sharon Robinson](#)
- [Michelle Clark](#)



Editar Cliente

Primeiro Nome:

Linda

Último Nome:

Williams

Email:

linda.williams@sakilacustor

Salvar Alterações

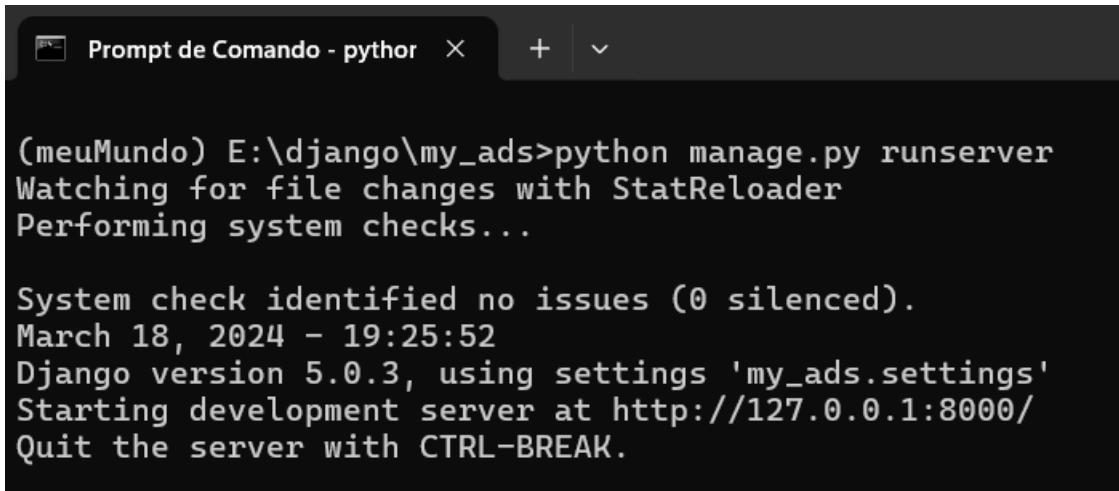
Altere e salve os dados

PROGRAMAÇÃO BACK-END

Desafio

PROGRAMAÇÃO BACK-END

Crie outra rotina que acesse outra tabela do banco, editando os seus dados.



```
(meuMundo) E:\django\my_ads>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
March 18, 2024 - 19:25:52
Django version 5.0.3, using settings 'my_ads.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```