

PROGRAMAÇÃO BACK-END

The Django logo, featuring the word "django" in a white, lowercase, sans-serif font with a small dot over the 'i', set against a dark green rectangular background.

The web framework for
perfectionists with deadlines.

Pesquisas com QuerySet

PROGRAMAÇÃO BACK-END

Referência de pesquisas de campo QuerySet

Em Django, um **QuerySet** é uma coleção de objetos de banco de dados que correspondem a um determinado conjunto de parâmetros de consulta.

Em outras palavras, é a maneira de Django de representar e trabalhar com consultas de banco de dados, permitindo **filtrar**, **ordenar** e **manipular** os **dados** de forma eficiente e abstrata.

PROGRAMAÇÃO BACK-END

Referência de pesquisas de campo QuerySet

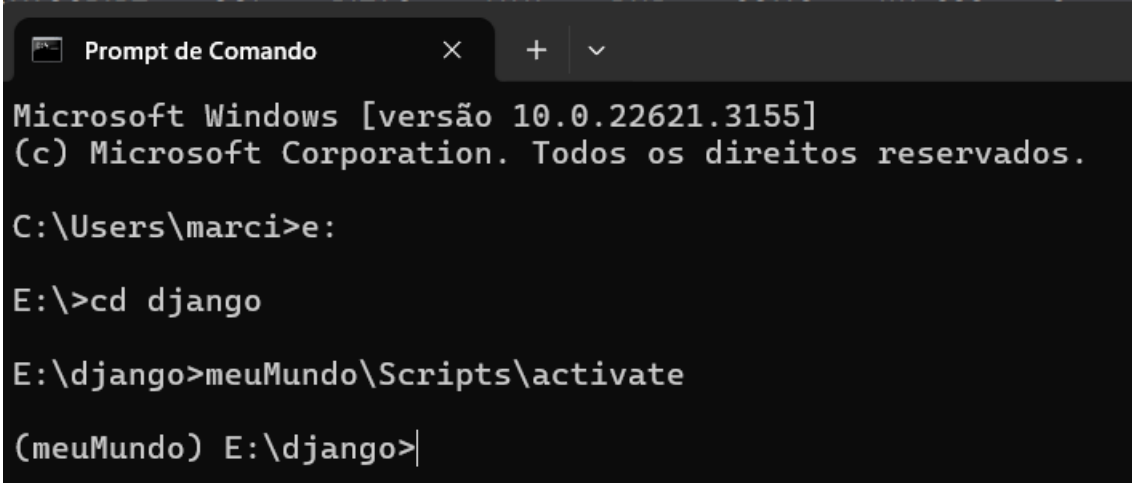
As pesquisas de campo em um **QuerySet** referem-se ao uso de **campos específicos** de modelos em consultas ao banco de dados.

Django permite realizar operações complexas de pesquisa, como filtrar registros por valores de campo, ordenar resultados por campos específicos, e até mesmo realizar **consultas** que **abrangem relacionamentos** entre diferentes **modelos**.

Preparando para os testes

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

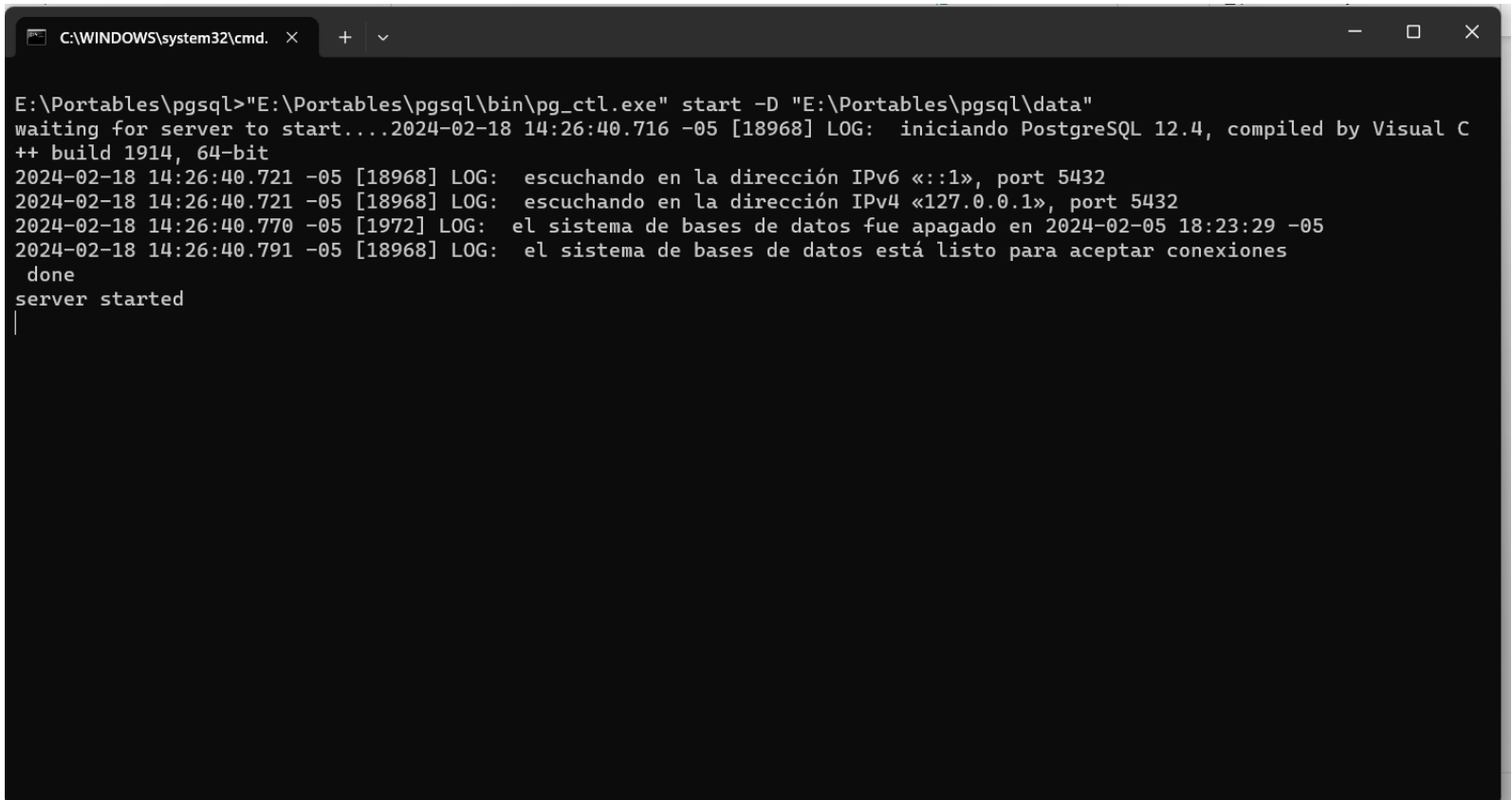
Na pasta `../pgsql` execute o arquivo **PostgreSQL-Start.bat** para iniciar o servidor



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. The title bar shows 'C:\WINDOWS\system32\cmd.' with standard window controls. The command prompt shows the execution of 'pg_ctl.exe' to start a PostgreSQL server. The output includes log messages indicating the server is starting, listening on IPv6 and IPv4, and that the database system is ready for connections. The final output is 'server started' followed by a cursor.

```
C:\WINDOWS\system32\cmd. x + v

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 direcci3n IPv6 «::1», port 5432
2024-02-18 14:26:40.721 -05 [18968] LOG:  escuchando en la direcci3n 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 est3 listo para aceptar conexiones
done
server started
|
```

Templates

PROGRAMAÇÃO BACK-END

Utilizando a App **dvdrentalapp**. Na pasta **templates** crie/verifique o arquivo **list_customers.html** e digite o seguinte conteúdo:

```
urls.py  list_customers.html x  list_customers1.html  views.py ●

dvdrentalapp > templates > list_customers.html > html
1  <!DOCTYPE html>
2  <html>
3  <body>
4
5  <h1>Customers</h1>
6
7  <ul>
8      {% for x in listcustomer %}
9          <li>{{ x.customer_id }} - {{ x.first_name }} {{ x.last_name }}</li>
10         {% endfor %}
11     </ul>
12
13 </body>
14 </html>
```

PROGRAMAÇÃO BACK-END

Arquivos `views.py` e `urls.py`

PROGRAMAÇÃO BACK-END


Na pasta **dvdrentalapp** adicione/verifique o seguinte conteúdo no arquivo **views.py** (linhas 9 a 14)



```
8 def listacustomer(request):
9     mycustomers = Customer.objects.all().values()
10    template = loader.get_template('list_customers.html')
11    context = {
12        'listcustomer': mycustomers,
13    }
14    return HttpResponse(template.render(context, request))
```

PROGRAMAÇÃO BACK-END

Na pasta **dvdrentalapp** adicione o seguinte conteúdo no arquivo **urls.py** (linha 10)



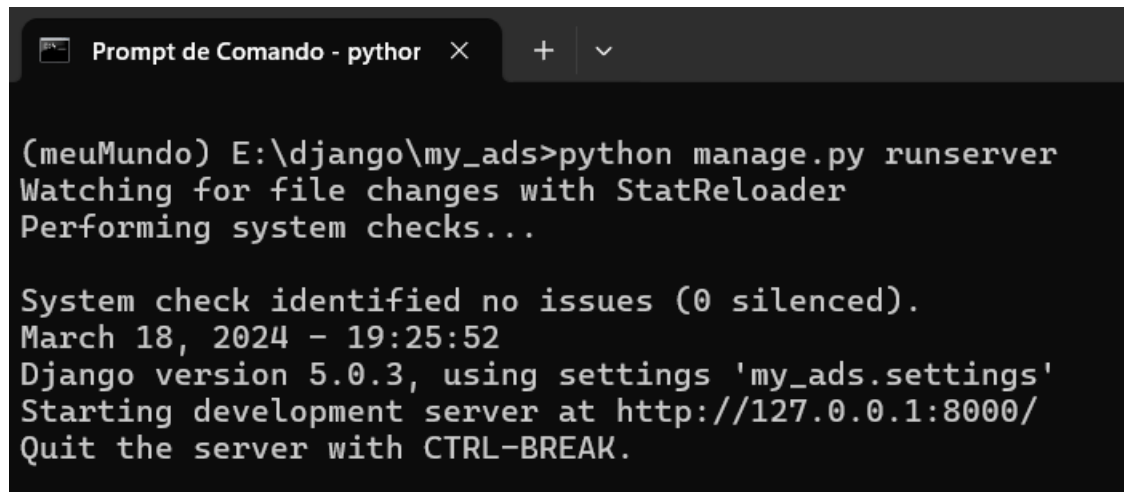
```
urls.py  x  list_customers.html  list_customers1.html  views.py  ●
dvdrentalapp > urls.py > ...
1  from django.urls import path
2  from . import views
3
4  urlpatterns = [
5      path('customer/', views.customer, name='mycustomer'),
6      path('detalhes/<int:id>', views.detalhes, name='myDetalhe'),
7      path('edit_customer/<int:customer_id>', views.edit_customer, name='edit_customer'),
8      path('add_category/', views.add_category, name='add_category'),
9      path('categories/', views.list_categories, name='list_categories'),
10     path('listacustomer/', views.listacustomer, name='listacustomer'),
11     # Adicione outras URLs conforme necessário
12 ]
```

**Iniciando o servidor
e testando**

PROGRAMAÇÃO BACK-END

Execute na pasta my_ads o comando:

python manage.py runserver

A screenshot of a Windows Command Prompt window titled "Prompt de Comando - pythor". The window shows the command prompt at "E:\django\my_ads" where the command "python manage.py runserver" has been entered. The output shows that the development server is starting, with messages about file change watching, system checks, and the server URL "http://127.0.0.1:8000/".

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

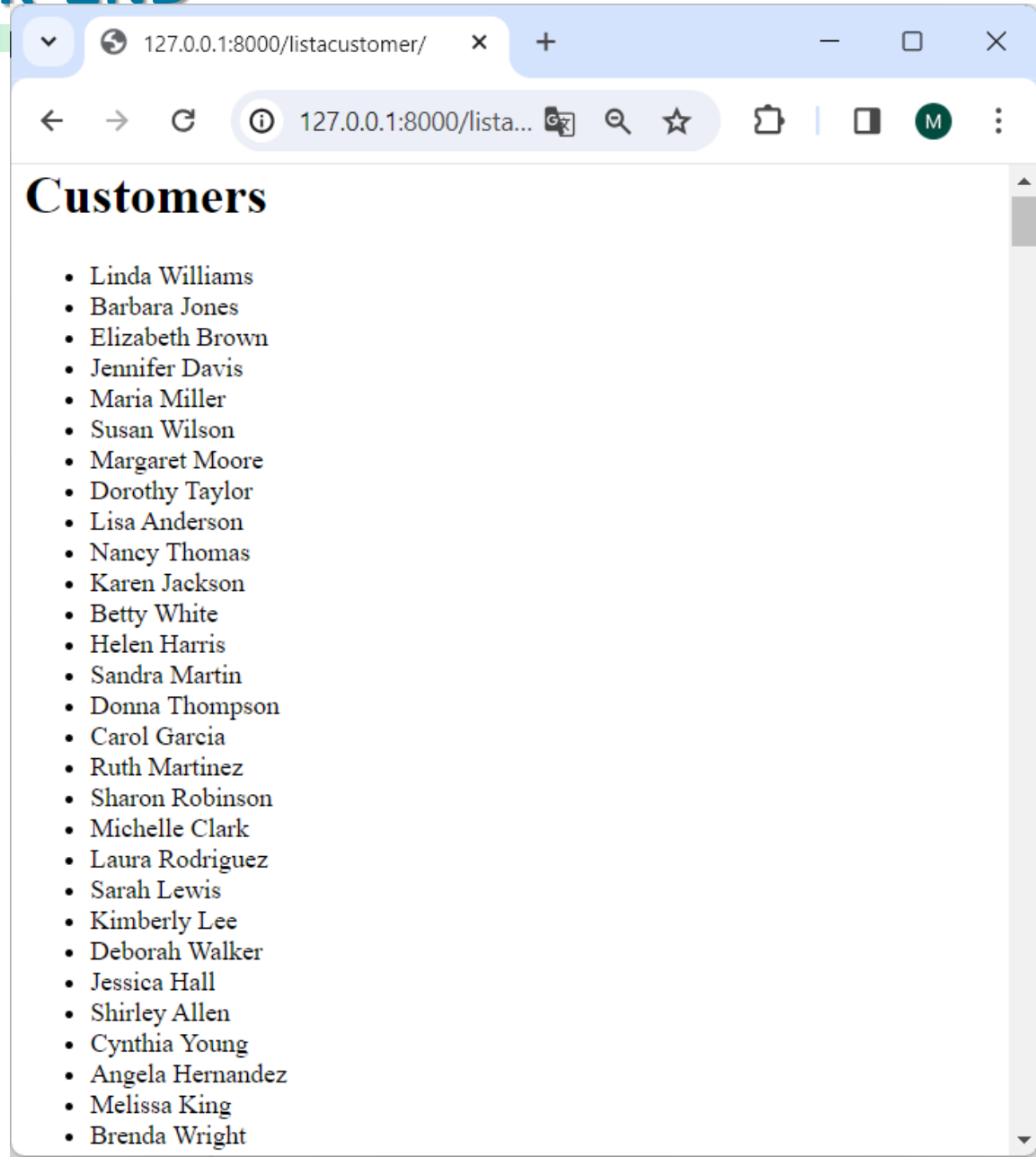
System check identified no issues (0 silenced).
March 18, 2024 - 19:25:52
Django version 5.0.3, using settings 'my_ads.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Vá ao navegador e digite:

<http://127.0.0.1:8000/listacustomer/>

PROGRAMAÇÃO BACK-END

O resultado da página será a listagem de todos customers.



**Método filter()
Operação AND**


PROGRAMAÇÃO BACK-END

Método filter()

O método **filter()** aceita os argumentos como ****kwargs** (argumentos de palavras-chave), para que você possa filtrar mais de um campo **separando-os** por **vírgula**.

PROGRAMAÇÃO BACK-END

Na pasta **dvdrentalapp** altere o seguinte conteúdo no arquivo **views.py** (linha 9):



```
8 def listacustomer(request):
9     mycustomers = Customer.objects.filter(first_name = 'Maria', customer_id=7 ).values()
10    template = loader.get_template('list_customers.html')
11    context = {
12        'listcustomer': mycustomers,
13    }
14    return HttpResponse(template.render(context, request))
```

Equivalente SQL

O SQL equivalente ao exemplo acima será:

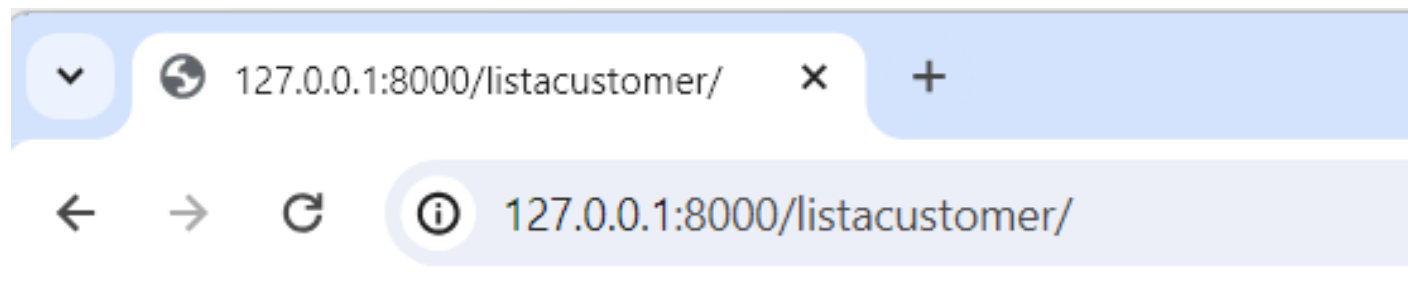
```
SELECT * FROM customers WHERE first_name
= 'Maria' AND customer_id = 7;
```

PROGRAMAÇÃO BACK-END

Vá ao navegador e digite:

<http://127.0.0.1:8000/listacustomer/>

O resultado da página será a **listagem** a seguir.



Customers

- 7 - Maria Miller

**Método filter()
Operação OR**

PROGRAMAÇÃO BACK-END


Método filter()

Podemos usar o método **filter()**, utilizando o separador que corresponde a uma barra vertical **|**. Este separador é equivalente a operação OR.

Os resultados serão mesclados em um modelo.

PROGRAMAÇÃO BACK-END

Na pasta **dvdrentalapp** altere o seguinte conteúdo no arquivo **views.py** (linha 9):



```
8 def listcustomer(request):
9     mycustomers = Customer.objects.filter(first_name = 'Maria').values() | Customer.objects.filter(customer_id=8).values()
10    template = loader.get_template('list_customers.html')
11    context = {
12        'listcustomer': mycustomers,
13    }
14    return HttpResponse(template.render(context, request))
```

Equivalente SQL

O SQL equivalente ao exemplo acima será:

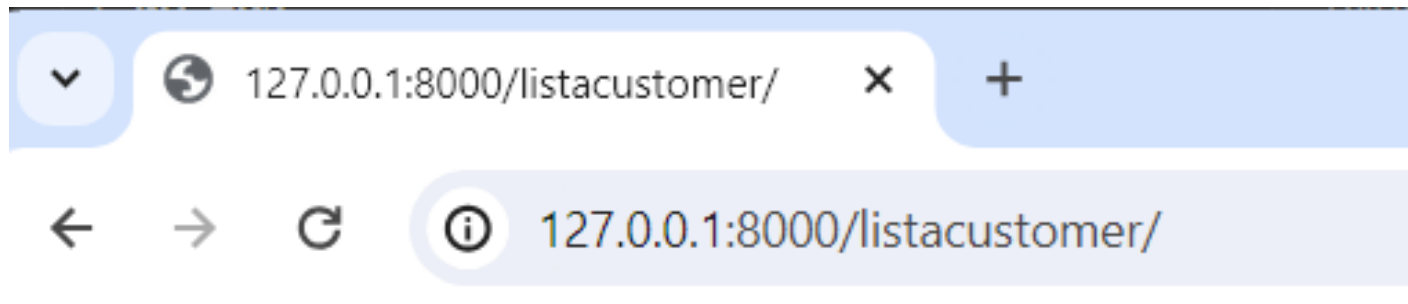
```
SELECT * FROM customers WHERE first_name
= 'Maria' OR customer_id = 8;
```


PROGRAMAÇÃO BACK-END

Vá ao navegador e digite:

<http://127.0.0.1:8000/listacustomer/>

O resultado da página será a **listagem** a seguir.



Customers

- 7 - Maria Miller
- 8 - Susan Wilson

Pesquisa através de
campo text no html

Templates

PROGRAMAÇÃO BACK-END

Utilizando o App **dvrentalapp**. Na pasta **templates** crie o arquivo **list_customers1.html** e digite o seguinte conteúdo:

```
urls.py  list_customers.html  list_customers1.html X  views.py

dvrentalapp > templates > list_customers1.html > ...
1  <!DOCTYPE html>
2  <html>
3  <body>
4
5  <h1>Customers</h1>
6
7  <!-- Adicione o formulário de pesquisa aqui -->
8  <form action="{% url 'listcustomer1' %}" method="get">
9      <label for="search_name">Procura pelo first_name:</label>
10     <input type="text" id="search_name" name="search_name">
11     <button type="submit">Procurar</button>
12 </form>
13
14 <ul>
15     {% for x in listcustomer1 %}
16         <li>{{ x.customer_id }} - {{ x.first_name }} {{ x.last_name }}</li>
17     {% endfor %}
18 </ul>
19
20 </body>
21 </html>
```

PROGRAMAÇÃO BACK-END

Arquivos views.py e urls.py

PROGRAMAÇÃO BACK-END


Na pasta **dvdrentalapp** adicione o seguinte conteúdo no arquivo **views.py** (linhas 16 a 31)



```
16 def listcustomer1(request):
17     # Obtém o valor de 'search_name' da requisição GET, retorna None se não encontrado
18     search_name = request.GET.get('search_name', '')
19
20     if search_name:
21         # Filtra os clientes pelo nome fornecido
22         mycustomers = Customer.objects.filter(first_name__contains=search_name).values()
23     else:
24         # Retorna todos os clientes se nenhum nome foi fornecido
25         mycustomers = Customer.objects.all().values()
26
27     template = loader.get_template('list_customers1.html')
28     context = {
29         'listcustomer1': mycustomers,
30     }
31     return HttpResponse(template.render(context, request))
```

PROGRAMAÇÃO BACK-END

Na pasta **dvdrentalapp** adicione o seguinte conteúdo no arquivo **urls.py** (linha 11)



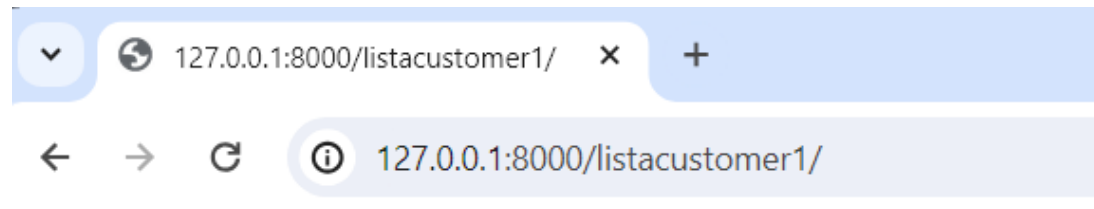
```
urls.py  X  list_customers.html  list_customers1.html  views.py
dvdrentalapp > urls.py > ...
1  from django.urls import path
2  from . import views
3
4  urlpatterns = [
5      path('customer/', views.customer, name='mycustomer'),
6      path('detalhes/<int:id>', views.detalhes, name='myDetalhe'),
7      path('edit_customer/<int:customer_id>', views.edit_customer, name='edit_customer'),
8      path('add_category/', views.add_category, name='add_category'),
9      path('categories/', views.list_categories, name='list_categories'),
10     path('listacustomer/', views.listacustomer, name='listacustomer'),
11     path('listacustomer1/', views.listacustomer1, name='listacustomer1'),
12     # Adicione outras URLs conforme necessário
13 ]
```

PROGRAMAÇÃO BACK-END

Vá ao navegador e digite:

<http://127.0.0.1:8000/listacustomer1/>

O resultado da página será a **listagem** a seguir.



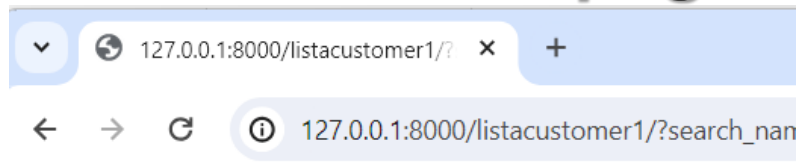
Customers

Procura pelo first_name:

- 3 - Linda Williams
- 4 - Barbara Jones
- 5 - Elizabeth Brown
- 6 - Jennifer Davis
- 7 - Maria Miller
- 8 - Susan Wilson
- 9 - Margaret Moore
- 10 - Dorothy Taylor
- 11 - Lisa Anderson

PROGRAMAÇÃO BACK-END

Digite no campo text as letras 'Mar' e acione o botão procura. O resultado da página será a **listagem a seguir.**



Customers

Procura pelo first_name:

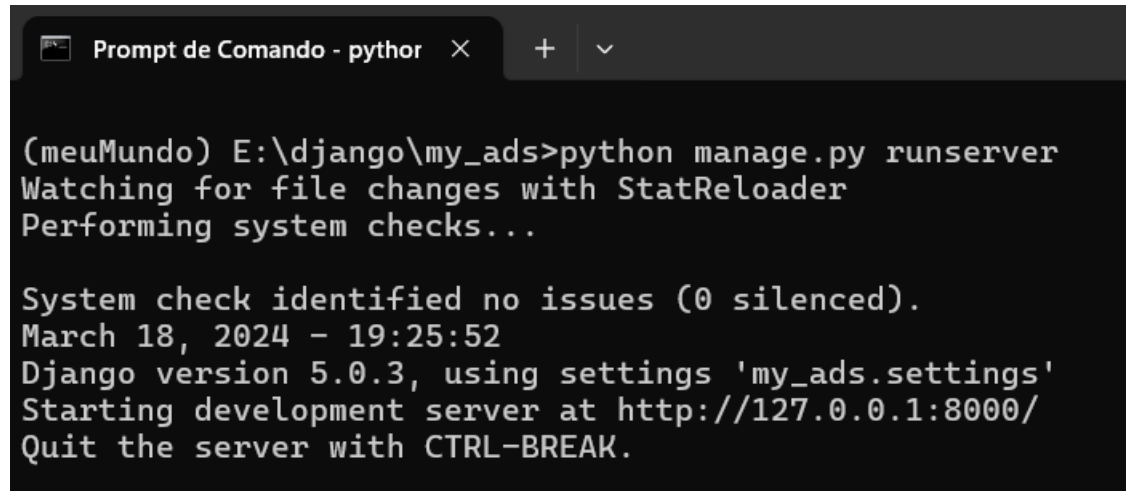
- 7 - Maria Miller
- 9 - Margaret Moore
- 38 - Martha Gonzalez
- 44 - Marie Turner
- 1 - Mary Smith1
- 80 - Marilyn Ross
- 128 - Marjorie Tucker
- 178 - Marion Snyder
- 236 - Marcia Dean
- 240 - Marlene Welch
- 246 - Marian Mendoza
- 257 - Marsha Douglas
- 267 - Margie Wade
- 312 - Mark Rinehart
- 383 - Martin Bales
- 413 - Marvin Yee
- 441 - Mario Cheatham
- 444 - Marcus Hidalgo
- 499 - Marc Outlaw
- 583 - Marshall Thorn
- 588 - Marion Ocampo

PROGRAMAÇÃO BACK-END

Desafio

PROGRAMAÇÃO BACK-END

Implemente a funcionalidade de procura em outra tabela do banco



```
Prompt de Comando - pythor × + ∨  
  
(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.
```