

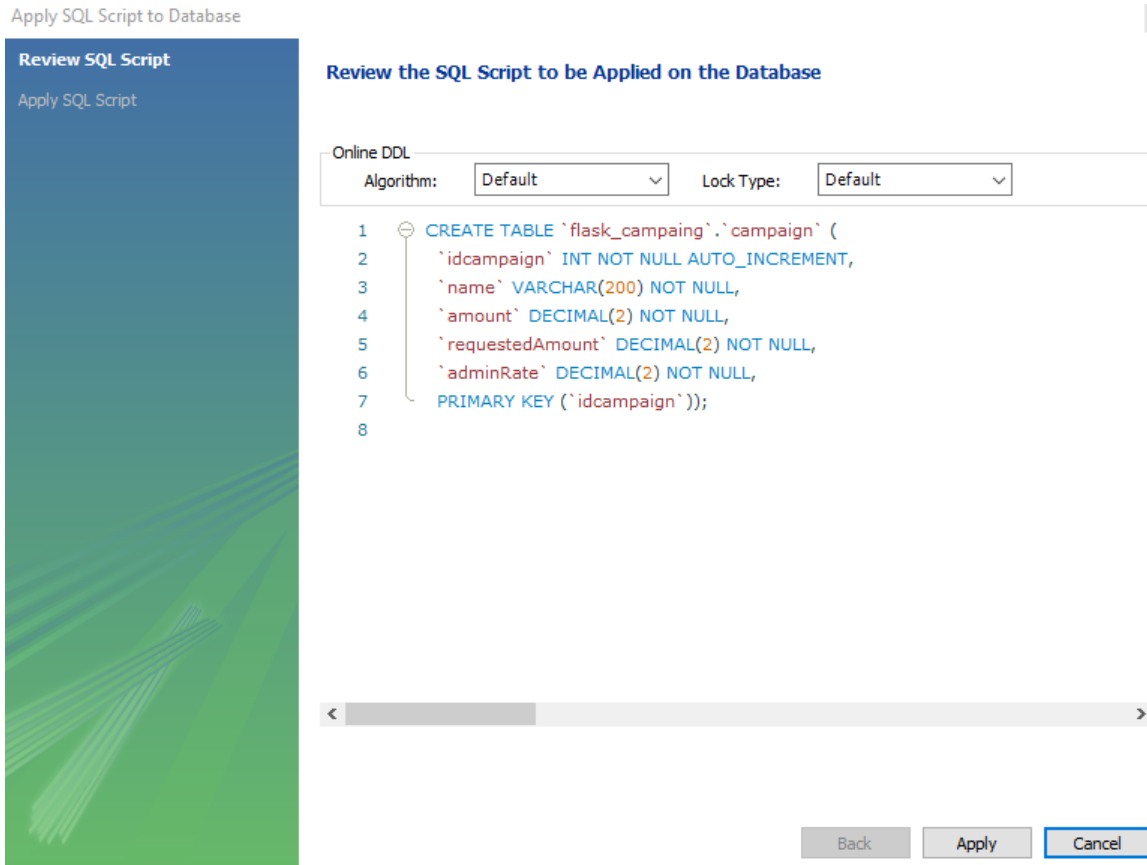
Autor: Jeisson Poveda.

Explicación y documentación.

Microservicio. A2censo.

El servicio debe retornar las campañas de forma ordenada por la cantidad de recursos o recursos solicitados.

1. Se crea la Base de datos usando el motor mysql.



```
INSERT INTO `flask_campaing`.`campaign` (`idcampaign`, `name`, `amount`, `requestedAmount`)
VALUES ('3', 'RobinFood2.0', '200000000', '250000000');
```

```
INSERT INTO `flask_campaing`.`campaign` (`idcampaign`, `name`, `amount`, `requestedAmount`)
VALUES ('1', 'T4 Tea For U', '1000000000', '1200000000');
```

```
INSERT INTO `flask_campaing`.`campaign` (`idcampaign`, `name`, `amount`, `requestedAmount`)
VALUES ('2', 'Smoking Burguers', '200000000', '200000000');
```

La creacion de la base de datos tambien es viable mediante sqlite

2. Mediante el framework FLASK y Python, se realiza la codificación del microservicio.

Se tiene 3 archivos:

- App.py: el cual contiene el Backend, es decir, la lógica que permite el funcionamiento de la página web.
- Index.html: Se describe el front de la aplicación web.
- Layout.html: se le realiza el diseño de estilos usando bootstrap.

En el Front de la aplicación web, se le da al usuario la opción de elegir por:

- Criterio: Amount o requestedAmount.

- Orden: Mayor a menor, menor a mayor.

a2censo

Opciones seleccion

Elige las opciones

Criterio Choose

Orden Choose

Cons requestedAmount

Up Down

name	amount	requestedAmount
T4 Tea For U	1000000000.0	1200000000.0
Smoking Burguers	200000000.0	200000000.0
RobinFood 2.0	200000000.0	250000000.0

a2censo

Opciones seleccion

Elige las opciones

Criterio Choose

Orden Choose

Cons Mayor a menor

Up Down

name	amount	requestedAmount
T4 Tea For U	1000000000.0	1200000000.0
Smoking Burguers	200000000.0	200000000.0
RobinFood 2.0	200000000.0	250000000.0

En postman se realiza la peticiones al microservicio, comprobando su correcto funcionamiento.

Overview

GET a2censo

No Environment

A2censo_bvc / a2censo

GET

http://127.0.0.1:300/

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

KEY	VALUE	DESCRIPTION	Bulk Edit
Key	Value	Description	

Body

Cookies

Headers (4)

Test Results

200 OK 49 ms 4.01 KB Save Response

Pretty

Raw

Preview

Visualize

HTML

```

1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta http-equiv="X-UA-Compatible" content="IE=edge">
7   <meta name="viewport" content="width=device-width, initial-scale=1.0">
8   <title>a2censo</title>
9   <link rel="stylesheet" href="https://bootswatch.com/5/cerulean/bootstrap.min.css">
10 </head>
11
12 <body>
13
14   <nav class="navbar navbar-expand-lg navbar-dark bg-dark">

```

Bootcamp

Runner

Trash

Overview GET a2censo X + ... No Environment

A2censo_bvc / a2censo Save ...

GET http://127.0.0.1:300/ Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (4) Test Results 200 OK 49 ms 4.01 KB Save Response

Pretty Raw Preview Visualize

```

<tbody>
  <tr>
    <td>T4 Tea For U</td>
    <td>1000000000.0</td>
    <td>1200000000.0</td>
  </tr>
  <tr>
    <td>Smoking Burguers</td>
    <td>2000000000.0</td>
    <td>2000000000.0</td>
  </tr>

```

Bootcamp Runner Trash

Bloque de código que permite traer todos los datos de la Base de datos creada en MySQL.

```

16 #Guardar datos seetings
17 app.secret_key='mysecretkey'
18 @app.route('/') #
19 def Index():
20     cur=mysql.connection.cursor()
21     cur.execute('SELECT *FROM campaign')
22     data=cur.fetchall() #para obtener todos esos datos
23     print(data)
24
25     return render_template('index.html',contacts=data) # cada vez que el usuario visite la ruta
26

```

Bloque de código que realiza la lógica de ordenamiento a partir de la selección por parte del usuario.

```

45 @app.route('/data', methods=['POST'])
46 def data():
47     if request.method=='POST':
48         criterio= request.form.get("sim")
49         orden=request.form.get("config")
50         print("Este es el criterio: " +criterio)
51         print("Orden: "+ orden)
52         print(type(orden))
53         datos=()
54         print(datos)
55
56         if orden=="ASC":
57             cur=mysql.connection.cursor()
58             cur.execute('SELECT * FROM campaign ORDER BY (%s) ASC',(criterio,))
59             mysql.connection.commit()
60             datos=cur.fetchall()
61             print(datos)
62             print("pasa ASC")
63         elif orden=='DESC':
64             cur=mysql.connection.cursor()
65             cur.execute('SELECT * FROM campaign ORDER BY (%s) DESC',(criterio,))
66             mysql.connection.commit()
67             datos=cur.fetchall()
68             print("pasa DESC")
69         print(datos)
70     return render_template('index.html',contacts=datos)
71

```

Testing.

Con selenium es una buena opción para realizar las pruebas del microservicio, y en una metodología screenplay con el uso de cucumber y serenity se tendría una prueba que da una historia de usuario.