Estudiantes: Jocksan, Christopher, Marvin

Iso Ubuntu: 22.04

Configuración:

Instalar postgresql:

```
root@user:/home/user# sudo apt install wget ca-certificates
```

```
root@user:/home/user# wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -
```

```
root@user:/home/user# sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/
/etc/apt/sources.list.d/pgdg.list'
```

```
root@user:/home/user# sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ $(lsb_release -cs)-pgdg main"
 >> /etc/apt/sources.list.d/pgdg.list'
```

```
root@user:/home/user# sudo apt update
```

```
root@user:/home/user# apt install postgresql postgresql-contrib
```

```
root@user:/home/user# service postgresql status
```

```
root@user:/home/user# sudo -u postgres psql
```

Configuración remota:

```
root@user:/home/user# nano /etc/postgresql/16/main/postgresql.conf
```

```
# - Connection Settings -

listen_addresses = '*'                  # what IP address(es) to listen on;
                                        # comma-separated list of addresses;
```

```
root@user:/home/user# nano /etc/postgresql/16/main/pg_hba.conf
```

```
# IPv4 local connections:
host    all             all             all                     scram-sha-256
```

```
root@user:/home/user# systemctl restart postgresql
```

Extracción del script de la base de datos:

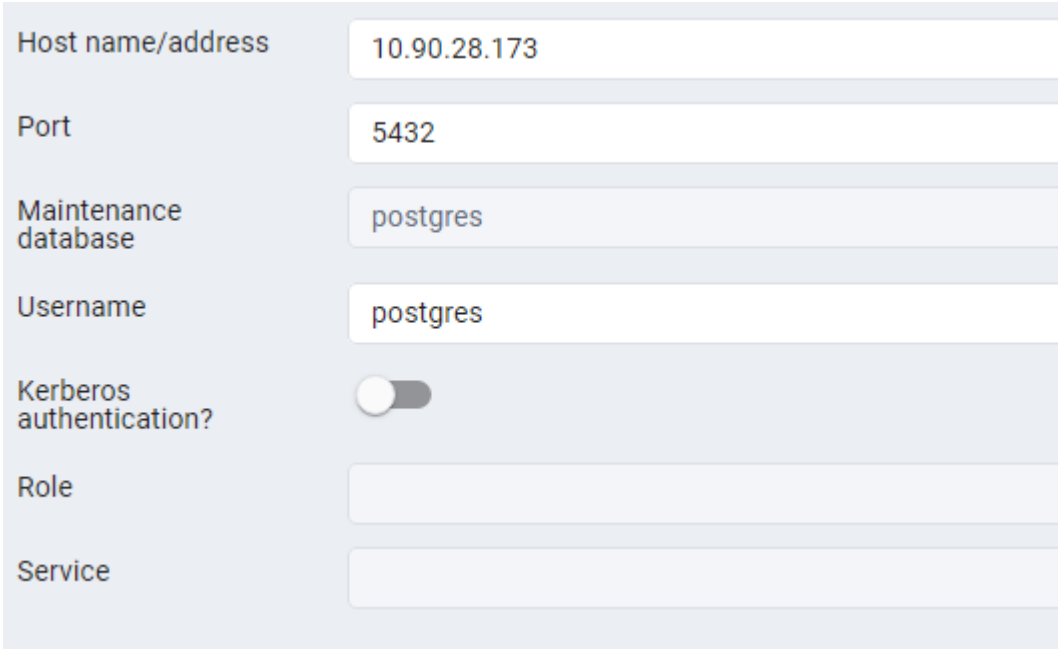
```
root@user:/home/user# su - postgres
postgres@user:~$ wget http://10.90.29.126:8080/isos/demo-medium-en.zip
```

```
postgres@user:~$ ls
16  demo-medium-en.zip
postgres@user:~$ unzip demo-medium-en.zip
Archive:  demo-medium-en.zip
  inflating: demo-medium-en-20170815.sql
postgres@user:~$ ls
16  demo-medium-en-20170815.sql  demo-medium-en.zip
```

Correr el script:

```
postgres@user:~$ psql -f demo-medium-en-20170815.sql
```

pgAdmin 4:



Configuración API:

```
root@user:/home/user# sudo apt install python3 python3-pip
```

Creación de entorno aislado:

```
root@user:/home/user# python3 -m venv myenv
root@user:/home/user# source myenv/bin/activate
```

Instalación de las herramientas:

```
root@user:/home/user# sudo apt install python3 python3-pip
```

```
(myenv) root@user:/home/user# sudo apt-get install libpq-dev python3-dev
```

```
(myenv) root@user:/home/user# python -m pip install --upgrade pip
```

```
(myenv) root@user:/home/user# pip install psycopg2
```

```
(myenv) root@user:/home/user# pip install Flask
```

Creación de archivo:

```
(myenv) root@user:/home/user# nano app.py
```

Configuración del archivo:

```python
# Call external libraries
import psycopg2
import locale
from flask import Flask, jsonify, abort, make_response, request

# Create default flask application
locale.setlocale(locale.LC_ALL, "es")
app = Flask(__name__)

# Function to execute data modification sentence
1 usage
def execute(auxsql):
    data = None
    try:
        # Create data access object
        conex = psycopg2.connect(host='10.90.28.173',
                                 database='demo',
                                 user='postgres',
                                 password='utn')
        # Create local cursor to SQL executor
        cur = conex.cursor()
        # Execute SQL sentence
        cur.execute(auxsql)
        # Retrieve data if exists
        data = cur.fetchall()
        # close cursor
        cur.close()
    except (Exception, psycopg2.DatabaseError) as error:
        print(error)
    finally:
        if conex is not None:
            conex.close()
            print('Close connection.')
    # Return data
    return data

# Error support section
@app.errorhandler(400)
def bad_request(error):
    return make_response(jsonify({'error': 'Bad request....!'}), 400)
```

```python
@app.errorhandler(401)
def unauthorized(error):
    return make_response(jsonify({'error': 'Unauthorized....!'}), 401)

@app.errorhandler(403)
def forbidden(error):
    return make_response(jsonify({'error': 'Forbidden....!'}), 403)

@app.errorhandler(404)
def not_found(error):
    return make_response(jsonify({'error': 'Not found....!'}), 404)

# Get Aircraft
@app.route('/aircraft', methods=['GET'])
def get_aircraft():
    resu = execute("select ad.aircraft_code, ad.model->'en', ad.range from aircrafts_data ad")
    if resu is not None:
        salida = {
            "status_code": 200,
            "status": "OK",
            "data": []
        }
        for cod, modelo, rango in resu:
            salida["data"].append({
                "code": cod,
                "model": modelo,
                "range": rango
            })
    else:
        abort(404)
    return jsonify({'data': salida}), 200

# Create thread app
if __name__ == '__main__':
    #Aqui va la IP de donde se esta ejecutando
    app.run(host='10.90.28.173', port=5001, debug=True)
```

Ejecutar archivo:

```
(myenv) root@user:/home/user# python app.py
```

Apis:

1:

```python
# Get Aircraft
@app.route( rule: '/airports/<string:lala>', methods=['GET'])
def get_aircraft(lala):
 resu = execute("select airport_code,airport_name,city,coordinates,timezone from airports_data")
 if resu != None:
     salida = {"status_code": 200,
     "status": "OK",
     "data":[]
    }
     for cod, nombre, ciudad, coor, timz in resu:
        salida["data"].append({
         "code": cod,
         "name": nombre[lala],
        "city": ciudad[lala],
            "coordinates":coor,
            "timezone":timz,
         })
 else:
    abort(404)
 return jsonify({'AirPorts': salida}), 200
```

2:

```python
@app.route( rule: '/passengers', methods=['GET'])
def get_passenger():
    ress = execute('select t.passenger_id, t.passenger_name,t.contact_data,h.fare_conditions,f.flight_id,f.departure_airport, f.arrival_airport,f.actual_departure, f.actual_arrival  '
                   'from ticket_flights h join tickets t on t.ticket_no = h.ticket_no join flights f on f.flight_id = h.flight_id LIMIT 100;')
    if ress !=None:
        salida = {"status_code": 200,
                  "status": "OK",
                  "data": []
                  }
        for id,nombre,info,cond,id2,ai1,ai2,hor1,hor2 in ress:
            salida["data"].append({
                "Passenger_ID":id,
                "Passenger_name":nombre,
                "Contac_Info":info,
                "Class":cond,
                "Fligths_ID":id2,
                "Airport_of_Departure":ai1,
                "Airport_of_Arrival":ai2,
                "Hour_of_Departure":hor1,
                "Hour_of_Arrival":hor2
            })
    else:
        abort(404)
    return jsonify({'Passengers': salida}), 200
```

3:

```python
@app.route( rule: '/passengers', methods=['GET'])
def get_passenger():
    ress = execute('select t.passenger_id, t.passenger_name,t.contact_data,h.fare_conditions,f.flight_id,f.departure_airport, f.arrival_airport,f.actual_departure, f.actual_arrival  '
                   'from ticket_flights h join tickets t on t.ticket_no = h.ticket_no join flights f on f.flight_id = h.flight_id LIMIT 100;')
    if ress !=None:
        salida = {"status_code": 200,
                  "status": "OK",
                  "data": []
                  }
        for id,nombre,info,cond,id2,ai1,ai2,hor1,hor2 in ress:
            salida["data"].append({
                "Passenger_ID":id,
                "Passenger_name":nombre,
                "Contac_Info":info,
                "Class":cond,
                "Fligths_ID":id2,
                "Airport_of_Departure":ai1,
                "Airport_of_Arrival":ai2,
                "Hour_of_Departure":hor1,
                "Hour_of_Arrival":hor2
            })
    else:
        abort(404)
    return jsonify({'Passengers': salida}), 200
```