

PONTÍFICA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Ensino Superior

Engenharia de Computação

LÂMPADA DE LAVA

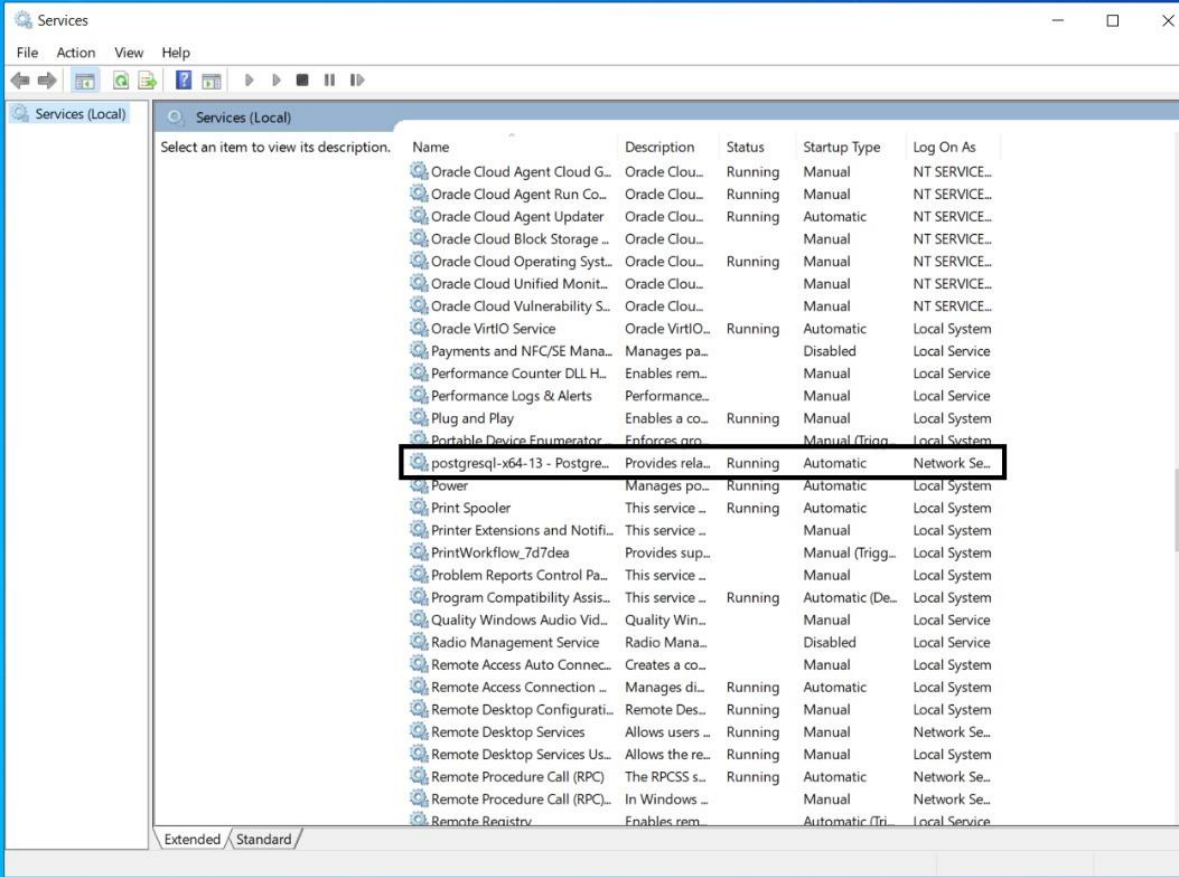
Relatório 6

Grupo: Ana Beatriz, Marcos Victor, Mariana Aram, Paula Talim, Yago Garzon

Belo Horizonte, 2023

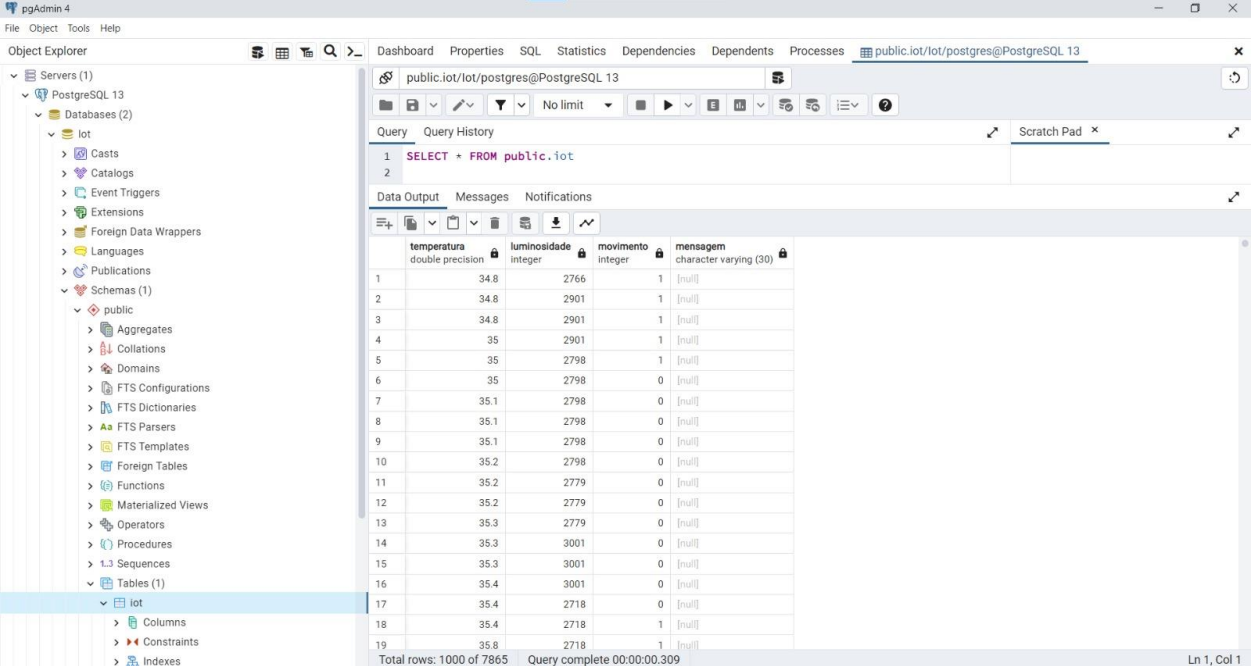
Objetivo: Demonstrar banco de dados

Banco de Dados



The screenshot shows the Windows Services console. The 'Services (Local)' window is open, displaying a list of services. The service 'postgresql-x64-13 - PostgreSQL' is highlighted with a black box. The service is running, has an automatic startup type, and is configured to run as the Network Service.

Name	Description	Status	Startup Type	Log On As
Oracle Cloud Agent Cloud G...	Oracle Clou...	Running	Manual	NT SERVICE...
Oracle Cloud Agent Run Co...	Oracle Clou...	Running	Manual	NT SERVICE...
Oracle Cloud Agent Updater	Oracle Clou...	Running	Automatic	NT SERVICE...
Oracle Cloud Block Storage ...	Oracle Clou...		Manual	NT SERVICE...
Oracle Cloud Operating Syst...	Oracle Clou...	Running	Manual	NT SERVICE...
Oracle Cloud Unified Monit...	Oracle Clou...		Manual	NT SERVICE...
Oracle Cloud Vulnerability S...	Oracle Clou...		Manual	NT SERVICE...
Oracle VirtIO Service	Oracle VirtIO...	Running	Automatic	Local System
Payments and NFC/SE Mana...	Manages pa...		Disabled	Local Service
Performance Counter DLL H...	Enables rem...		Manual	Local Service
Performance Logs & Alerts	Performance...		Manual	Local Service
Plug and Play	Enables a co...	Running	Manual	Local System
Portable Device Enumerator	Enforces pro...		Manual (Trigg...	Local System
postgresql-x64-13 - Postgre...	Provides rela...	Running	Automatic	Network Se...
Power	Manages po...	Running	Automatic	Local System
Print Spooler	This service ...	Running	Automatic	Local System
Printer Extensions and Notifi...	This service ...		Manual	Local System
PrintWorkflow_7d7dea	Provides sup...		Manual (Trigg...	Local System
Problem Reports Control Pa...	This service ...		Manual	Local System
Program Compatibility Assis...	This service ...	Running	Automatic (De...	Local System
Quality Windows Audio Vid...	Quality Win...		Manual	Local Service
Radio Management Service	Radio Mana...		Disabled	Local Service
Remote Access Auto Connec...	Creates a co...		Manual	Local System
Remote Access Connection ...	Manages di...	Running	Automatic	Local System
Remote Desktop Configurati...	Remote Des...	Running	Manual	Local System
Remote Desktop Services	Allows users ...	Running	Manual	Network Se...
Remote Desktop Services Us...	Allows the re...	Running	Manual	Local System
Remote Procedure Call (RPC)	The RPCSS s...	Running	Automatic	Network Se...
Remote Procedure Call (RPC)...	In Windows ...		Manual	Network Se...
Remote Registry	Enables rem...		Automatic (Tri...	Local Service



The screenshot shows the pgAdmin 4 interface. The 'Object Explorer' on the left displays the database structure for 'PostgreSQL 13'. The 'Query Editor' on the right shows a query: `SELECT * FROM public.iot`. The 'Data Output' tab displays the results of the query, showing a table with 19 rows and 4 columns: `temperatura` (double precision), `luminosidade` (integer), `movimento` (integer), and `mensagem` (character varying (30)).

	temperatura double precision	luminosidade integer	movimento integer	mensagem character varying (30)
1	34.8	2766	1	[null]
2	34.8	2901	1	[null]
3	34.8	2901	1	[null]
4	35	2901	1	[null]
5	35	2798	1	[null]
6	35	2798	0	[null]
7	35.1	2798	0	[null]
8	35.1	2798	0	[null]
9	35.1	2798	0	[null]
10	35.2	2798	0	[null]
11	35.2	2779	0	[null]
12	35.2	2779	0	[null]
13	35.3	2779	0	[null]
14	35.3	3001	0	[null]
15	35.3	3001	0	[null]
16	35.4	3001	0	[null]
17	35.4	2718	0	[null]
18	35.4	2718	1	[null]
19	35.8	2718	1	[null]

Total rows: 1000 of 7865 Query complete 00:00:00.309 Ln 1, Col 1

Listagem do Programa em Python

```

import paho.mqtt.client as mqtt
import psycopg2

dados = {}

config = {
    'host': '168.138.154.70',
    'port': 5432,
    'database': 'Iot',
    'user': 'postgres',
    'password': '123'
}

try:
    # Testando a conexão
    conn = psycopg2.connect(**config)
    print("Conexão bem-sucedida!")

    cursor = conn.cursor()
    try:
        # Definição da tabela
        create_table_query = '''
        CREATE TABLE IF NOT EXISTS iot (
            temperatura DOUBLE PRECISION,
            luminosidade INTEGER,
            movimento INTEGER,
            mensagem VARCHAR(30)
        );
        ...

        # Executando o comando CREATE TABLE
        cursor.execute(create_table_query)

        # Confirmando a transação
        conn.commit()

        print("Tabela criada ou já existente.")

    except psycopg2.Error as e:

```

```

        print(f"Erro ao criar a tabela: {e}")

except psycopg2.Error as e:
    print(f"Erro ao conectar ao banco de dados: {e}")

def print_hi(name):
    # mensagem inicial
    print(f'Oi, {name}')

# esta função é a função callback informando que o cliente se conectou ao servidor
def on_connect(client, userdata, flags, rc):
    print("Connectado com codigo "+str(rc))

    # assim que conecta, assina um tópico. Se a conexão for perdida, assim que
    # que reconectar, as assinaturas serão renovadas
    client.subscribe("lampadalava/sensor/temperatura")
    client.subscribe("lampadalava/sensor/luminosidade")
    client.subscribe("lampadalava/sensor/movimento")
    client.subscribe("lampadalava/sensor/mensagem")

# esta função é a função callback que é chamada quando uma publicação é recebida
do servidor
def on_message(client, userdata, msg):
    print("Mensagem recebida no tópico: " + msg.topic)
    print("Mensagem: "+ str(msg.payload.decode()))

    if str(msg.payload.decode().strip()) == "termina":
        print("-----")
        print("Recebeu comando termina.")
        print("-----")
        if conn.is_connected():
            cursor.close()
            conn.close()
            print("Fim da conexão com o Banco dadosIoT")

```

```

if str(msg.payload.decode().strip()) == "delete":
    print("-----")
    print("Recebeu comando delete e apaga tabela .")
    print("-----")
    cursor.execute("TRUNCATE TABLE iot")

# Armazena a mensagem no dicionário usando o tópico como chave
if msg.topic != "lampadalava/sensor/mensagem":
    dados[msg.topic] = str(msg.payload.decode())

# Verifica se todas as mensagens foram recebidas
if len(dados) == 3: # Número de tópicos que você está esperando receber
    # Insere os dados no banco de dados
    temperatura = dados['lampadalava/sensor/temperatura']
    luminosidade = dados['lampadalava/sensor/luminosidade']
    movimento = dados['lampadalava/sensor/movimento']

    cursor.execute("INSERT INTO iot (temperatura, luminosidade, movimento)
VALUES (%s, %s, %s)", (temperatura, luminosidade, movimento))

    conn.commit()

if __name__ == '__main__':
    print_hi('este é nosso trabalho')

    client = mqtt.Client("IoT_PUC_SG_mqtt")
    client.on_connect = on_connect
    client.on_message = on_message

    client.connect("test.mosquitto.org", 1883, 60)

    # a função abaixo manipula trafego de rede, trata callbacks e manipula
    reconexões.

    client.loop_forever()

```