ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

UNIVERSIDAD

Computing Networks

Laboratory No. 5
Databases and Network Protocols

Students:

Andrea Camila Torres González

Jorge Andrés Gamboa Sierra

Presented to:
Fabian Eduardo Sierra Sánchez

Semester 2024-2

ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

UNIVERSIDAD

Table of Contents

Abstract

In this laboratory, we will use Wireshark to capture packages of Network Protocols, and we will analyze their content. Then we will install database server in Slackware, Windows Server and Azure where with those databases, we will create users and databases. Into each database we will create tables, insert data and give permissions to each user who can only access their own database. Finally, we will make a remote connection with DBeaver to connect our database with a client.
Key words: query, database, user, data, remote

Objective

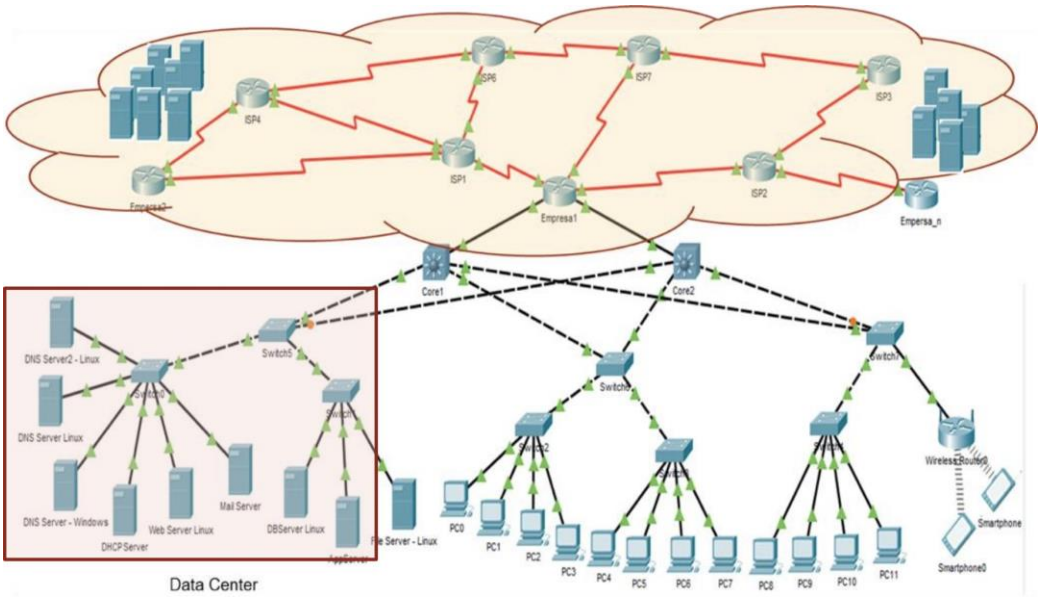
Continue learning the installation of basic software and the operation of network protocols.

Tools to be used

- Computers
- Internet access
- Virtualization software
- Wireshark

Introduction

We are continuing to work within a company's infrastructure, which typically includes various IT services. This infrastructure features both wired and wireless user workstations and servers (both physical and virtual- ized), all connected through switches (Layer 2 and Layer 3), wireless equipment, and routers connecting them to the Internet. Cloud infrastructures are also common, provisioning resources as needed by the organization. The servers may host services such as web, DNS, email, databases, storage, and applications, among others.



In this part of the lab, we will focus on continuing to set up our servers.

Database Management System (DBMS)

A DBMS is software that facilitates the creation and management of databases, ensuring data integrity and security. It allows defining, manipulating, and retrieving stored data. It is essential for organizing and handling large amounts of information in businesses. DBMS includes tools for managing transactions and ensuring data consistency. Examples of DBMS include MySQL, PostgreSQL, and Oracle.

PostgreSQL

PostgreSQL is an open-source relational and object-relational DBMS known for its robustness and extensibility. It supports ACID transactions, referential integrity, and stored procedures. It allows for custom data types and extensions. It is used in both small applications and large enterprise systems. Its active community ensures continuous improvements and support.

SQL Server

SQL Server is a relational DBMS developed by Microsoft, known for its security and ability to handle large volumes of data. It offers advanced tools like SQL Server Management Studio (SSMS). It supports ACID transactions and provides replication and high availability. It is used for data analysis and report generation. Its integration with other Microsoft products makes it popular in enterprise environments.

SQL Database Azure

SQL Database Azure is a cloud-based database service from Microsoft Azure, based on SQL Server. It offers high availability, scalability, and security without the need to manage the underlying infrastructure. It integrates artificial intelligence and machine learning for automatic optimization. It is ideal for applications requiring flexibility and simplified management. It facilitates disaster recovery and continuous monitoring.

DNS Protocol

DNS translates domain names into IP addresses, allowing access to online resources using human-readable names. It is essential for Internet functionality, delegating authority and caching to improve efficiency. DNS servers resolve domain name queries and direct Internet traffic. It provides a hierarchical structure for name management. It facilitates web navigation and online communication.

HTTP Protocol

HTTP is the protocol used to transmit data on the web, defining how messages are formatted and transmitted. It operates at the application layer of the OSI model. It is a stateless protocol, where each request is independent. HTTP/1.1 and HTTP/2 versions improve

transmission efficiency and speed. It is the foundation of communication on the World Wide Web, allowing the transfer of web documents.

Wireshark

Wireshark is a network analysis tool that captures and examines real-time network traffic. It allows users to see all data packets sent and received by a network. It is invaluable for troubleshooting and performance analysis. It offers a graphical interface for detailed analysis of captured data. It facilitates packet filtering and classification for precise diagnostics.

Review of Network Protocols

1. Review of DNS Messages

We perform the query of the URL www.google.com through a web browser and obtain four packets. The first one makes the DNS query request to a server with the IP 10.1.0.17.

IP client

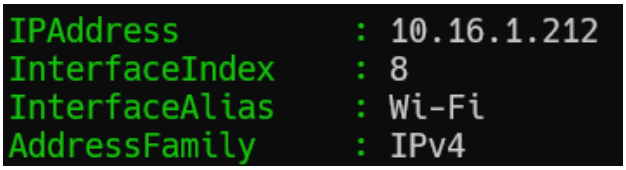


Figure 1 IP client

DNS server client

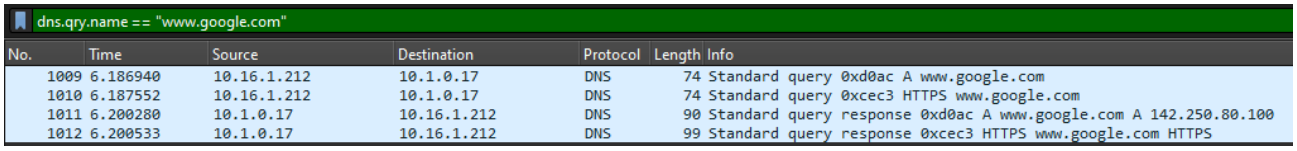


Figure 2 DNS Query

In packet No. 1009, the client with IP 10.16.1.212 sends a DNS query to the server at 10.1.0.17, requesting the A record for www.google.com. This query allows the client to obtain the IP address necessary to connect to the server.

In packet No. 1010, the client (10.16.1.212) makes another DNS query to the same server (10.1.0.17), this time requesting the HTTPS record for www.google.com. This query seeks information on how to establish a secure connection to the server.

In the 1011 packet, the DNS server (10.1.0.17) responds to the client with the IPv4 address (142.250.80.100) for www.google.com. This response is crucial for the client to connect to the server.

The packet No. 1012 is a response from the DNS server to the client, confirming the HTTPS record for www.google.com. It provides details on how the client should make a secure connection to the server.

The two packets we will analyze are 1009 and 1011, which are the most relevant for the DNS query.

In packet No. 1009, the first field shows the next information about the interface used, kind of Encapsulation, transmission time and information about the frame

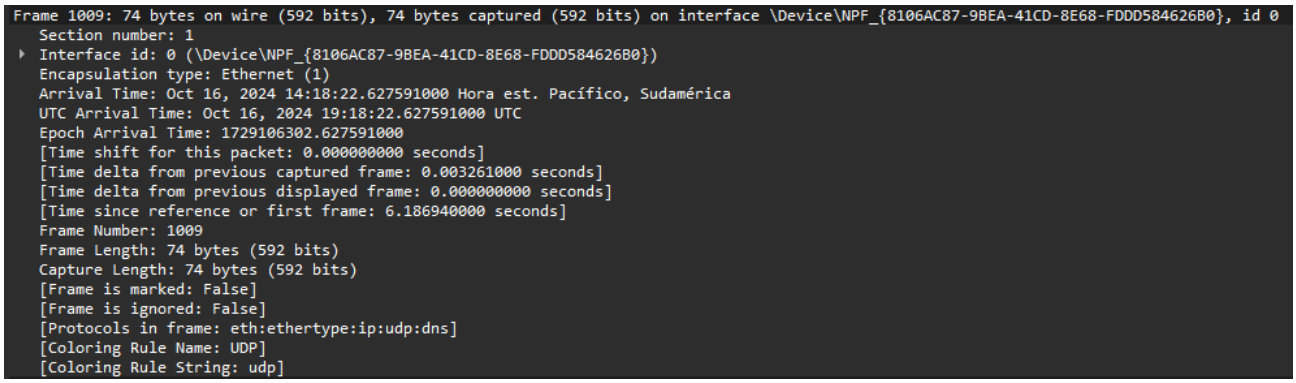


Figure 3 first field DNS query

The second field provides information about the MAC address of both the source and destination devices, as well as the type of IP used, which is IPv4.

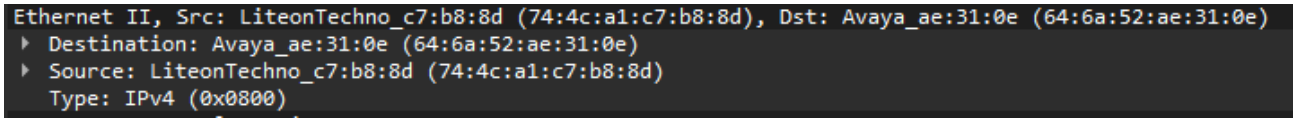


Figure 4 second field DNS query

In the third field, the packet information reveals that it is an IPv4 packet with a source IP address of 10.16.1.212 and a destination IP address of 10.1.0.17. The packet uses the UDP transport layer protocol, has a total length of 60 bytes, and is not fragmented. The IP header is 20 bytes long.

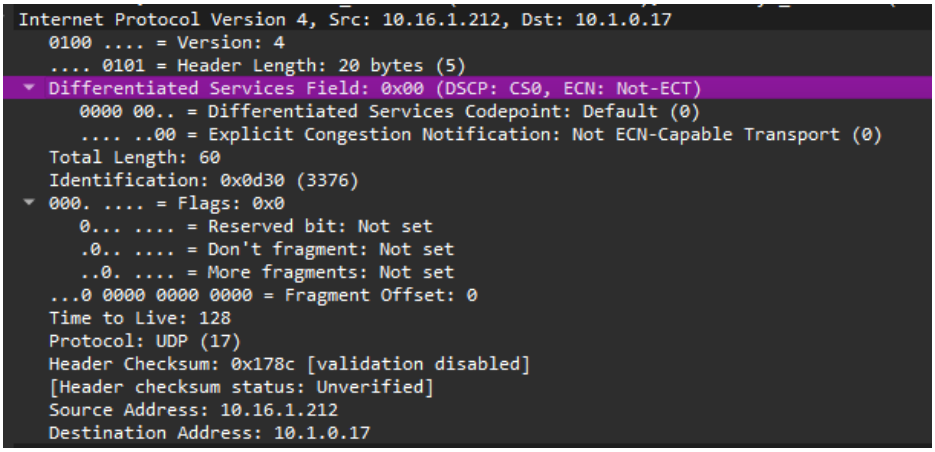


Figure 5 third field DNS query

The fourth field shows a source port of 60634 and a destination port of 53, indicating that it is a DNS query. The packet has a total length of 40 bytes and a UDP payload of 32 bytes. The UDP checksum is 0x9d44.

```
▼ User Datagram Protocol, Src Port: 60634, Dst Port: 53
    Source Port: 60634
    Destination Port: 53
    Length: 40
    Checksum: 0x9d44 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 29]
  ▼ [Timestamps]
      [Time since first frame: 0.000000000 seconds]
      [Time since previous frame: 0.000000000 seconds]
    UDP payload (32 bytes)
```

Figure 6 fourth field DNS query

The fifth field shows that the query is of type A for the domain "www.google.com" and is made recursively (flag RD: recursion desired). The packet is not truncated and does not contain authenticated data. It also indicates that this is an initial request expecting a response from the DNS server.

```
▼ Domain Name System (query)
    Transaction ID: 0xd0ac
  ▼ Flags: 0x0100 Standard query
      0... .... .... .... = Response: Message is a query
      .000 0... .... .... = Opcode: Standard query (0)
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... .0.. .... = Z: reserved (0)
      .... .... ...0 .... = Non-authenticated data: Unacceptable
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ▼ Queries
    ▶ www.google.com: type A, class IN
    [Response In: 1011]
```

Figure 7 fifth field DNS query

Package No. 1011, the first field shows the next information about the interface used, kind of Encapsulation, transmission time and information about the frame

```
▼ Frame 1011: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface \Device\NPF_{8106AC87-9BEA-41CD-8E68-FDDD584626B0}, id 0
    Section number: 1
  ▼ Interface id: 0 (\Device\NPF_{8106AC87-9BEA-41CD-8E68-FDDD584626B0})
      Interface name: \Device\NPF_{8106AC87-9BEA-41CD-8E68-FDDD584626B0}
      Interface description: Wi-Fi
    Encapsulation type: Ethernet (1)
    Arrival Time: Oct 16, 2024 14:18:22.640931000 Hora est. Pacífico, Sudamérica
    UTC Arrival Time: Oct 16, 2024 19:18:22.640931000 UTC
    Epoch Arrival Time: 1729106302.640931000
    [Time shift for this packet: 0.000000000 seconds]
    [Time delta from previous captured frame: 0.012728000 seconds]
    [Time delta from previous displayed frame: 0.012728000 seconds]
    [Time since reference or first frame: 6.200280000 seconds]
    Frame Number: 1011
    Frame Length: 90 bytes (720 bits)
    Capture Length: 90 bytes (720 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ip:udp:dns]
    [Coloring Rule Name: UDP]
    [Coloring Rule String: udp]
```

Figure 8 first field DNS response

The second field provides information about the MAC address of both the source and destination devices, as well as the type of IP used, which is IPv4.

```
Ethernet II, Src: Avaya_ae:31:0e (64:6a:52:ae:31:0e), Dst: LiteonTechno_c7:b8:8d (74:4c:a1:c7:b8:8d)
  Destination: LiteonTechno_c7:b8:8d (74:4c:a1:c7:b8:8d)
  Source: Avaya_ae:31:0e (64:6a:52:ae:31:0e)
  Type: IPv4 (0x0800)
```

Figure 9 second field DNS response

In the third field, the packet information reveals that it is an IPv4 packet with a source IP address of 10.1.0.17 and a destination IP address of 10.16.1.212. The packet uses the UDP transport layer protocol, has a total length of 76 bytes, and is not fragmented. The IP header is 20 bytes long

```
Internet Protocol Version 4, Src: 10.1.0.17, Dst: 10.16.1.212
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      0000 00.. = Differentiated Services Codepoint: Default (0)
      .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    Total Length: 76
    Identification: 0xfb5d (64349)
  000. .... = Flags: 0x0
      0... .... = Reserved bit: Not set
      .0.. .... = Don't fragment: Not set
      ..0. .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 61
    Protocol: UDP (17)
    Header Checksum: 0x6c4e [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 10.1.0.17
    Destination Address: 10.16.1.212
```

Figure 10 third field DNS response

The fourth field shows a source port of 53 and a destination port of 60634, indicating that it is a DNS response. The packet has a total length of 56 bytes and a UDP payload of 48 bytes. The UDP checksum is 0x7c61.

```
User Datagram Protocol, Src Port: 53, Dst Port: 60634
    Source Port: 53
    Destination Port: 60634
    Length: 56
    Checksum: 0x7c61 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 29]
  [Timestamps]
    UDP payload (48 bytes)
```

Figure 11 fourth field DNS response

The fifth field shows that the response is of type A for the domain "www.google.com" and is made recursively (flag RD: recursion desired). It also indicates that this is a response from the DNS server, the response..

```
Domain Name System (response)
    Transaction ID: 0xd0ac
  Flags: 0x8180 Standard query response, No error
      1... .... .... .... = Response: Message is a response
      .000 0... .... .... = Opcode: Standard query (0)
      .... .0.. .... .... = Authoritative: Server is not an authority for domain
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... 1... .... = Recursion available: Server can do recursive queries
      .... .... .0.. .... = Z: reserved (0)
      .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by the server
      .... .... ...0 .... = Non-authenticated data: Unacceptable
      .... .... .... 0000 = Reply code: No error (0)
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  Queries
    www.google.com: type A, class IN
        Name: www.google.com
        [Name Length: 14]
        [Label Count: 3]
        Type: A (1) (Host Address)
        Class: IN (0x0001)
  Answers
    www.google.com: type A, class IN, addr 142.250.80.100
    [Request In: 1009]
    [Time: 0.013340000 seconds]
```

Figure 12 fifth field DNS response

2. Review of HTTP Messages

When querying the URL http://profesores.is.escuelaing.edu.co/˜csantiago/, we obtained the following results that show the requests made to the server with IP 45.239.88.86 via GET requests. We can also observe the responses given to the client with IP 192.168.1.3.

http

No.	Time	Source	Destination	Protocol	Length	Info
307	8.027576	192.168.1.3	45.239.88.86	HTTP	524	GET /~csantiago/ HTTP/1.1
310	8.048998	45.239.88.86	192.168.1.3	HTTP	492	HTTP/1.1 200 OK (text/html)
313	8.076681	192.168.1.3	45.239.88.86	HTTP	492	GET /~csantiago/foto.jpg HTTP/1.1
348	8.116241	45.239.88.86	192.168.1.3	HTTP	1131	HTTP/1.1 200 OK (PNG)
354	8.138556	192.168.1.3	45.239.88.86	HTTP	484	GET /favicon.ico HTTP/1.1
359	8.152942	45.239.88.86	192.168.1.3	HTTP	474	HTTP/1.1 404 Not Found (text/html)

Figure 13 HTTP query

When viewing the request made in the first packet, we can see that a GET request is being sent in the header. It requests the host **profesores.is.escuelaing.edu.co**, specifically requesting the resource **~csantiago**.

```
Hypertext Transfer Protocol
  GET /~csantiago/ HTTP/1.1\r\n
  Host: profesores.is.escuelaing.edu.co\r\n
  Connection: keep-alive\r\n
  Upgrade-Insecure-Requests: 1\r\n
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7\r\n
  Accept-Encoding: gzip, deflate\r\n
  Accept-Language: es-US,es-419;q=0.9,es;q=0.8\r\n
  \r\n
  [Full request URI: http://profesores.is.escuelaing.edu.co/~csantiago/]
  [HTTP request 1/3]
  [Response in frame: 310]
  [Next request in frame: 313]
```

Figure 14 first request GET

In the third captured packet, a response from the server is observed, sending an affirmative reply with a **200 OK** message. It shows the request made to the URI **http://profesores.is.escuelaing.edu.co/csantiago/**. Additionally, we can see the resource obtained, which is an HTML document consisting of 9 lines and has a

size of 146 bytes.

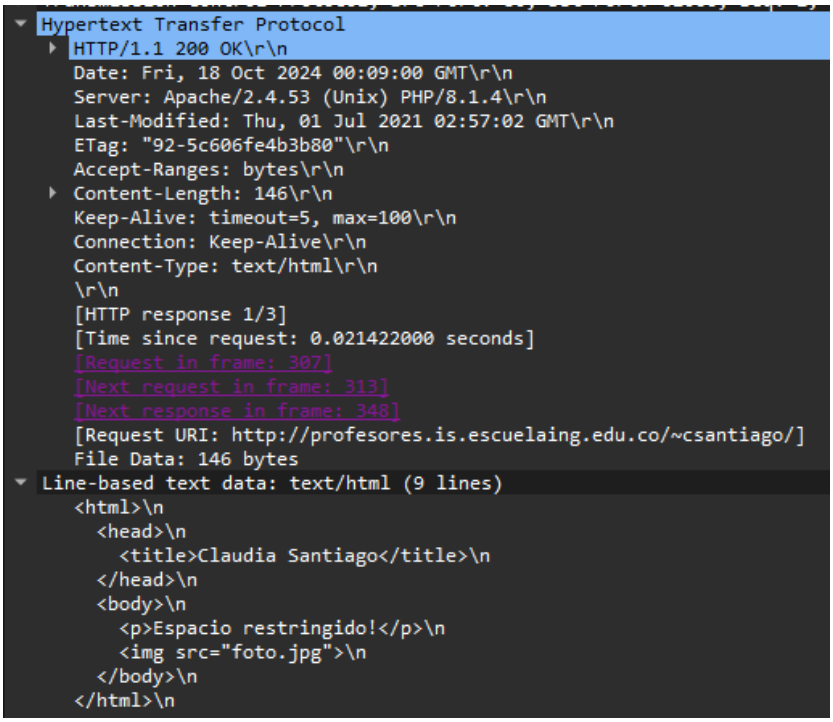


Figure 15 first response HTTP

In the third packet, another request made to the server using a **GET** method can be seen. This is the second of three requests that will be made in total. The request is for an image resource, specifically **foto.png**, with the host being the same as in the previous request, located at **~csantiago**.

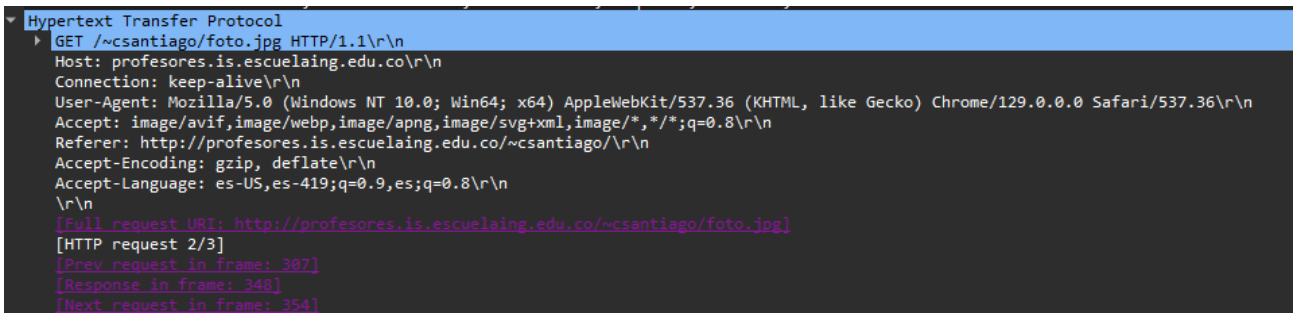


Figure 16 second request HTTP

In the fourth packet, another positive response from the server is observed, indicated by another **200 OK** header. It shows that the requested resource weighs **24,141 Bytes**, along with the URI to which the request was made.

```
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
  Date: Fri, 18 Oct 2024 00:09:00 GMT\r\n
  Server: Apache/2.4.53 (Unix) PHP/8.1.4\r\n
  Last-Modified: Thu, 01 Jul 2021 02:58:24 GMT\r\n
  ETag: "5e4d-5c607032e7400"\r\n
  Accept-Ranges: bytes\r\n
  Content-Length: 24141\r\n
  Keep-Alive: timeout=5, max=99\r\n
  Connection: Keep-Alive\r\n
  Content-Type: image/jpeg\r\n
  \r\n
  [HTTP response 2/3]
  [Time since request: 0.039560000 seconds]
  [Prev request in frame: 307]
  [Prev response in frame: 310]
  [Request in frame: 313]
  [Next request in frame: 354]
  [Next response in frame: 359]
  [Request URI: http://profesores.is.escuelaing.edu.co/~csantiago/foto.jpg]
  File Data: 24141 bytes
  [Expert Info (Note/Malformed): HTTP body subdissector failed, trying heuristic subdissector]
     [HTTP body subdissector failed, trying heuristic subdissector]
     [Severity level: Note]
     [Group: Malformed]
Portable Network Graphics
```

Figure 17 second response HTTP

In the following packet, the last request made to the server is observed, which is for the resource **favicon.ico**. The request was made to the same host, **profesores.is.escuelaing.edu.co**.

```
Hypertext Transfer Protocol
  GET /favicon.ico HTTP/1.1\r\n
    [Expert Info (Chat/Sequence): GET /favicon.ico HTTP/1.1\r\n]
       [GET /favicon.ico HTTP/1.1\r\n]
       [Severity level: Chat]
       [Group: Sequence]
    Request Method: GET
    Request URI: /favicon.ico
    Request Version: HTTP/1.1
  Host: profesores.is.escuelaing.edu.co\r\n
  Connection: keep-alive\r\n
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36\r\n
  Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8\r\n
  Referer: http://profesores.is.escuelaing.edu.co/~csantiago/\r\n
  Accept-Encoding: gzip, deflate\r\n
  Accept-Language: es-US,es-419;q=0.9,es;q=0.8\r\n
  \r\n
  [Full request URI: http://profesores.is.escuelaing.edu.co/favicon.ico]
  [HTTP request 3/3]
  [Prev request in frame: 313]
  [Response in frame: 359]
```

Figure 18 third request HTTP

In the last response, there is a **404 error**, which corresponds to a request made by the client indicating that the requested resource was not found.

```
Hypertext Transfer Protocol
  HTTP/1.1 404 Not Found\r\n
    [Expert Info (Chat/Sequence): HTTP/1.1 404 Not Found\r\n]
    Response Version: HTTP/1.1
    Status Code: 404
    [Status Code Description: Not Found]
    Response Phrase: Not Found
  Date: Fri, 18 Oct 2024 00:09:00 GMT\r\n
  Server: Apache/2.4.53 (Unix) PHP/8.1.4\r\n
  Content-Length: 196\r\n
  Keep-Alive: timeout=5, max=98\r\n
  Connection: Keep-Alive\r\n
  Content-Type: text/html; charset=iso-8859-1\r\n
  \r\n
  [HTTP response 3/3]
  [Time since request: 0.014386000 seconds]
  [Prev request in frame: 313]
  [Prev response in frame: 348]
  [Request in frame: 354]
  [Request URI: http://profesores.is.escuelaing.edu.co/favicon.ico]
  File Data: 196 bytes
Line-based text data: text/html (7 lines)
```

Figure 19 third response HTTP

3. Review of Ethernet Frames

Next, we applied the filter http.request.method == "GET" to the captured data. This filter isolates all GET requests made to specific URLs on the website, enabling us to analyze the details of each request and the corresponding server responses. This focused approach enhances our understanding of the HTTP traffic and interactions between the client and the server.

```
http.request.method == "GET"
```

No.	Time	Source	Destination	Protocol	Length	Info
307	8.027576	192.168.1.3	45.239.88.86	HTTP	524	GET /~csantiago/ HTTP/1.1
313	8.076681	192.168.1.3	45.239.88.86	HTTP	492	GET /~csantiago/foto.jpg HTTP/1.1
354	8.138556	192.168.1.3	45.239.88.86	HTTP	484	GET /favicon.ico HTTP/1.1

Figure 20 Ethernet Frames query

During the traffic capture, we observed three packets requesting different HTML elements. The server has a destination IP address of 45.239.88.86, to which a specific query is being made for the directory "~csantiago."

```
Ethernet II, Src: LiteonTechno_c7:b8:8d (74:4c:a1:c7:b8:8d), Dst: ARRISGroup_d2:bd:1c (50:75:f1:d2:bd:1c)
  Destination: ARRISGroup_d2:bd:1c (50:75:f1:d2:bd:1c)
  Source: LiteonTechno_c7:b8:8d (74:4c:a1:c7:b8:8d)
  Type: IPv4 (0x0800)
```

Figure 21 All Ethernet fields results

In the analysis of the Ethernet frames, the same information is observed across the three packets due to their belonging to a single communication session. This means that the packets follow the same path between the local PC and the server, utilizing the same devices, resulting in unchanged source and destination MAC addresses. Additionally, the protocol type, in this case, IPv4, remains constant across all messages, ensuring smooth and coherent communication throughout the interaction.

Base Software Installation

Another key component of basic IT infrastructure is database management systems (DBMS). These DBMS can be hosted either in a company's datacenter or on a cloud server. They store structured data for the organization and are used by different applications supporting business operations.

1. PostgreSQL - Linux Slackware

a. Installation process

- First, we create the account for the PostgreSQL superuser, using the name 'postgres'. To do this, we write the command **'adduser postgres'** and complete all the information requested by the console.

```
Shell [ /bin/bash ]

Expiry date (YYYY-MM-DD) []:

New account will be created as follows:

---------------------------------------
Login name.......:  postgres
UID..............:  123
Initial group....:  users
Additional groups:  [ None ]
Home directory...:  /home/postgres
Shell............:  /bin/bash
Expiry date......:  [ Never ]

This is it... if you want to bail out, hit Control-C.  Otherwise, press
ENTER to go ahead and make the account.


Creating new account...


Changing finger information for postgres.
Name []: postgres
Office []:
Office Phone []:
Home Phone []:


Finger information changed.
New password:
Retype new password:
passwd: password updated successfully


Account setup complete.
root@andrea:~#
```

Figure 22.Adding the Postgres user to the system

- We create the directory where we will download, unpack and decompress the sources, which will be saved in /usr/src/pgsql. Then, we create the directory where we will install the server, in our case under /usr/local/pgsql. Finally, we make the postgres user the owner of these directories.

```
Account setup complete.
root@andrea:~# mkdir /usr/src/pgsql
root@andrea:~# chwon postgres:users /usr/src/pgsql
-bash: chwon: command not found
root@andrea:~# chown postgres:users /usr/src/pgsql
root@andrea:~# mkdir /usr/local/pgsql
root@andrea:~# chown postgres:users /usr/local/pgsql
root@andrea:~# exit_
```

Figure 23. Creating permissions for the postgres user

- Now we must install some packages that are essential to avoid issues with the PostgreSQL server and the C compiler:
  - ✓ Kernel-headers
  - ✓ gcc
  - ✓ gc
  - ✓ g++
  - ✓ binutils
  - ✓ make
  - ✓ glibc

```
2024-10-12 18:40:06 (262 KB/s) - ■//var/cache/packages/./slackware64/d/kernel-headers-6.10.14-x86-1.
txz■ saved [1200208/1200208]

                        Downloading http://ftp.slackware-brasil.com.br/slackware64-current/./slackwa
re64/d/kernel-headers-6.10.14-x86-1.txz.asc...
--2024-10-12 18:40:06--  http://ftp.slackware-brasil.com.br/slackware64-current/slackware64/d/kernel
-headers-6.10.14-x86-1.txz.asc
Resolving ftp.slackware-brasil.com.br (ftp.slackware-brasil.com.br)... 200.137.217.134
Connecting to ftp.slackware-brasil.com.br (ftp.slackware-brasil.com.br)|200.137.217.134|:80... conne
cted.
HTTP request sent, awaiting response... 200 OK
Length: 195 [text/plain]
Saving to: ■//var/cache/packages/./slackware64/d/kernel-headers-6.10.14-x86-1.txz.asc■

//var/cache/packages/./s 100%[==================================>]     195  --.-KB/s    in 0s

2024-10-12 18:40:11 (16.6 MB/s) - ■//var/cache/packages/./slackware64/d/kernel-headers-6.10.14-x86-1
.txz.asc■ saved [195/195]

        Package kernel-headers-6.10.14-x86-1.txz is already in cache - not downloading
        Installing kernel-headers-6.10.14-x86-1...
Verifying package kernel-headers-6.10.14-x86-1.txz.
Installing package kernel-headers-6.10.14-x86-1.txz:
PACKAGE DESCRIPTION:
# kernel-headers (Linux kernel include files)
#
# These are the include files from the Linux kernel.
#
# You'll need these to compile most system software for Linux.
#
Executing install script for kernel-headers-6.10.14-x86-1.txz.
Package kernel-headers-6.10.14-x86-1.txz installed.
Searching for NEW configuration files...
               No .new files found.

root@andrea:~#
```

Figure 24. kernel-headers installation

```
2024-10-12 18:40:56 (187 KB/s) - ■//var/cache/packages/./slackware64/l/gc-8.2.8-x86_64-1.txz■ saved
[303800/303800]

                        Downloading http://ftp.slackware-brasil.com.br/slackware64-current/./slackwa
re64/l/gc-8.2.8-x86_64-1.txz.asc...
--2024-10-12 18:40:56--  http://ftp.slackware-brasil.com.br/slackware64-current/slackware64/l/gc-8.2
.8-x86_64-1.txz.asc
Resolving ftp.slackware-brasil.com.br (ftp.slackware-brasil.com.br)... 200.137.217.134
Connecting to ftp.slackware-brasil.com.br (ftp.slackware-brasil.com.br)|200.137.217.134|:80... conne
cted.
HTTP request sent, awaiting response... 200 OK
Length: 195 [text/plain]
Saving to: ■//var/cache/packages/./slackware64/l/gc-8.2.8-x86_64-1.txz.asc■

//var/cache/packages/./s 100%[===================================>]     195  --.-KB/s    in 0s

2024-10-12 18:41:01 (30.1 MB/s) - ■//var/cache/packages/./slackware64/l/gc-8.2.8-x86_64-1.txz.asc■ s
aved [195/195]

       Package gc-8.2.8-x86_64-1.txz is already in cache - not downloading
       Installing gc-8.2.8-x86_64-1...
Verifying package gc-8.2.8-x86_64-1.txz.
Installing package gc-8.2.8-x86_64-1.txz:
PACKAGE DESCRIPTION:
# gc (garbage collector library)
#
# The Boehm-Demers-Weiser conservative garbage collector can be used as
# a garbage collecting replacement for C malloc or C++ new.
#
# Homepage: http://www.hboehm.info/gc/
#
Executing install script for gc-8.2.8-x86_64-1.txz.
Package gc-8.2.8-x86_64-1.txz installed.
Searching for NEW configuration files...
               No .new files found.

root@andrea:~# _
```

Figure 25. gc installation

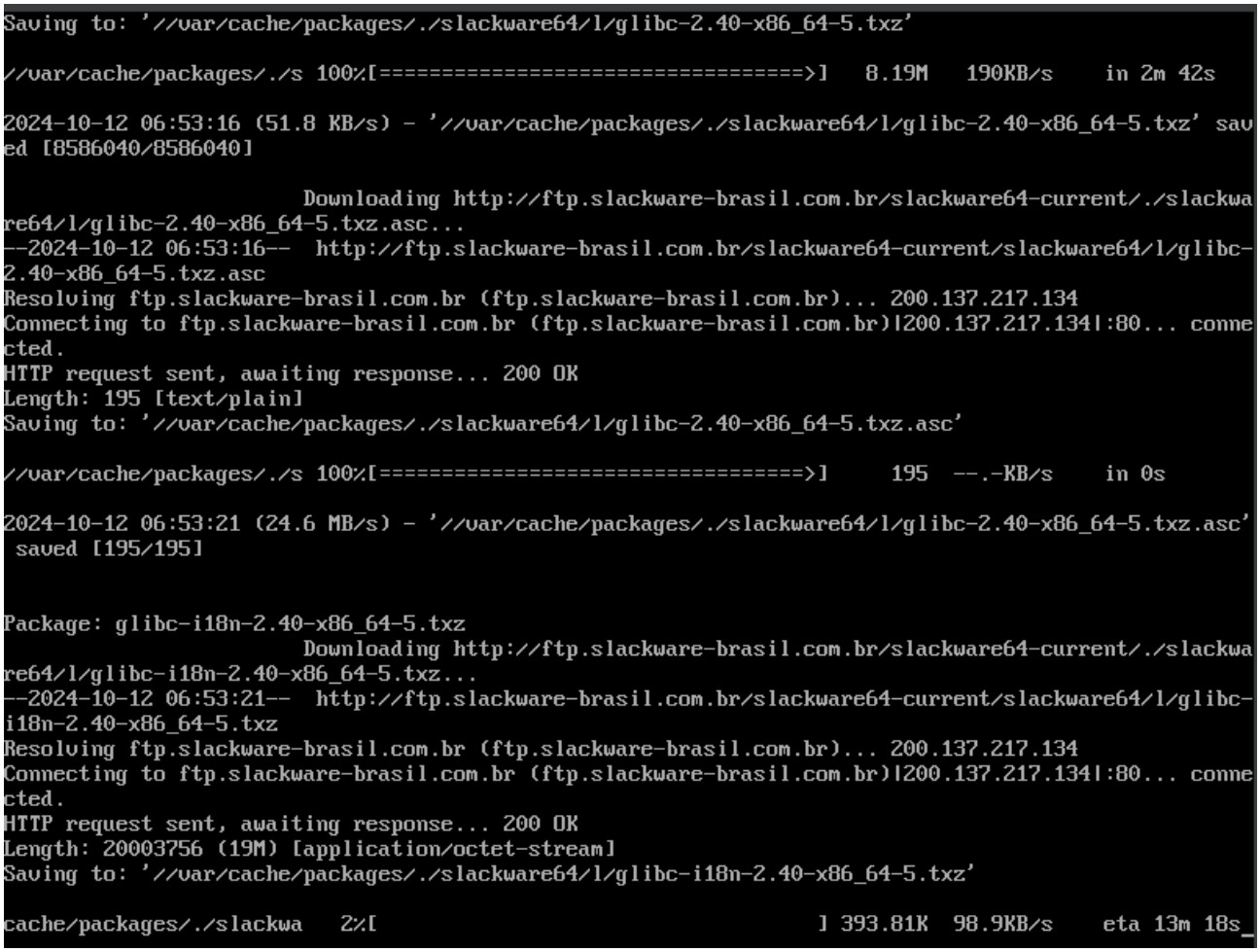


Figure 26. glibc installation

```
--2024-10-11 16:48:40--  http://ftp.slackware-brasil.com.br/slackware64-current/slackware64/d/gcc-14
.2.0-x86_64-2.txz
Resolving ftp.slackware-brasil.com.br (ftp.slackware-brasil.com.br)... 200.137.217.134
Connecting to ftp.slackware-brasil.com.br (ftp.slackware-brasil.com.br)|200.137.217.134|:80... conne
cted.
HTTP request sent, awaiting response... 200 OK
Length: 35743464 (34M) [application/octet-stream]
Saving to: '//var/cache/packages/./slackware64/d/gcc-14.2.0-x86_64-2.txz'

//var/cache/packages/./s 100%[===================================>]  34.09M  2.99MB/s    in 37s

2024-10-11 16:49:23 (933 KB/s) - '//var/cache/packages/./slackware64/d/gcc-14.2.0-x86_64-2.txz' save
d [35743464/35743464]

                        Downloading http://ftp.slackware-brasil.com.br/slackware64-current/./slackwa
re64/d/gcc-14.2.0-x86_64-2.txz.asc...
--2024-10-11 16:49:23--  http://ftp.slackware-brasil.com.br/slackware64-current/slackware64/d/gcc-14
.2.0-x86_64-2.txz.asc
Resolving ftp.slackware-brasil.com.br (ftp.slackware-brasil.com.br)... 200.137.217.134
Connecting to ftp.slackware-brasil.com.br (ftp.slackware-brasil.com.br)|200.137.217.134|:80... conne
cted.
HTTP request sent, awaiting response... 200 OK
Length: 195 [text/plain]
Saving to: '//var/cache/packages/./slackware64/d/gcc-14.2.0-x86_64-2.txz.asc'

//var/cache/packages/./s 100%[===================================>]     195  --.-KB/s    in 0s

2024-10-11 16:49:29 (35.3 MB/s) - '//var/cache/packages/./slackware64/d/gcc-14.2.0-x86_64-2.txz.asc'
 saved [195/195]


Package: gcc-g++-14.2.0-x86_64-2.txz
                        Downloading http://ftp.slackware-brasil.com.br/slackware64-current/./slackwa
re64/d/gcc-g++-14.2.0-x86_64-2.txz...
--2024-10-11 16:49:29--  http://ftp.slackware-brasil.com.br/slackware64-current/slackware64/d/gcc-g+
+-14.2.0-x86_64-2.txz
Resolving ftp.slackware-brasil.com.br (ftp.slackware-brasil.com.br)... _
```

Figure 27. g++ installation

- We log out of the root account and log in with the postgres user account. After that, we navigate to the /usr/src/pgsql directory and use wget to download the .tar.gz file that contains PostgreSQL sources

```
Welcome to Linux 5.15.19 x86_64 (tty1)

andrea login: postgres
Password:
Last failed login: Sat Oct 12 07:59:49 -05 2024 on tty1
There was 1 failed login attempt since the last successful login.
Linux 5.15.19.
postgres@andrea:~$ cd /usr/src/pgsql
postgres@andrea:/usr/src/pgsql$ wget ftp://ftp.postgresql.org/pub/source/v15.3/postgresql-15.3.tar.g
z_
```

Figure 28. Starting PostgreSQL installation

```
6 packets transmitted, 6 received, 0% packet loss, time 25167ms
rtt min/avg/max/mdev = 10.969/12.335/14.940/1.706 ms
root@andrea:~# exit
logout


Welcome to Linux 5.15.19 x86_64 (tty1)

andrea login: postgres
Password:
Last login: Sun Oct 13 15:27:04 on tty1
Linux 5.15.19.
postgres@andrea:~$ cd /usr/src/pgsql
postgres@andrea:/usr/src/pgsql$ wget ftp://ftp.postgresql.org/pub/source/v15.3/postgresql-15.3.tar.g
z
--2024-10-13 15:32:44--  ftp://ftp.postgresql.org/pub/source/v15.3/postgresql-15.3.tar.gz
           => ■postgresql-15.3.tar.gz■
Resolving ftp.postgresql.org (ftp.postgresql.org)... 217.196.149.55, 147.75.85.69, 87.238.57.227, ..
.
Connecting to ftp.postgresql.org (ftp.postgresql.org)|217.196.149.55|:21... ^C
postgres@andrea:/usr/src/pgsql$ wget https://ftp.postgresql.org/pub/source/v15.3/postgresql-15.3.tar
.gz
--2024-10-13 15:33:34--  https://ftp.postgresql.org/pub/source/v15.3/postgresql-15.3.tar.gz
Resolving ftp.postgresql.org (ftp.postgresql.org)... 217.196.149.55, 72.32.157.246, 87.238.57.227, .
..
Connecting to ftp.postgresql.org (ftp.postgresql.org)|217.196.149.55|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 29946539 (29M) [application/octet-stream]
Saving to: ■postgresql-15.3.tar.gz■

postgresql-15.3.tar.gz   100%[===================================>]  28.56M  10.8MB/s    in 2.6s

2024-10-13 15:33:42 (10.8 MB/s) - ■postgresql-15.3.tar.gz■ saved [29946539/29946539]

postgres@andrea:/usr/src/pgsql$ ls
postgresql-15.3.tar.gz
postgres@andrea:/usr/src/pgsql$ _
```

Figure 29. PostgreSQL-15.3.tar.gz downloaded

- We unpack and decompress the tarball

```
postgresql-15.3/doc/src/sgml/tsm-system-time.sgml
postgresql-15.3/doc/src/sgml/plpgsql.sgml
postgresql-15.3/doc/src/sgml/README.links
postgresql-15.3/doc/src/sgml/pgstatstatements.sgml
postgresql-15.3/doc/src/sgml/xaggr.sgml
postgresql-15.3/doc/src/sgml/appendix-obsolete.sgml
postgresql-15.3/doc/src/sgml/stylesheet.xsl
postgresql-15.3/doc/src/sgml/contrib.sgml
postgresql-15.3/doc/src/sgml/man-stamp
postgresql-15.3/doc/src/sgml/basic-archive.sgml
postgresql-15.3/doc/src/sgml/bki.sgml
postgresql-15.3/doc/src/sgml/unaccent.sgml
postgresql-15.3/doc/src/sgml/btree.sgml
postgresql-15.3/doc/src/sgml/external-projects.sgml
postgresql-15.3/doc/src/sgml/problems.sgml
postgresql-15.3/doc/src/sgml/arch-dev.sgml
postgresql-15.3/doc/src/sgml/pageinspect.sgml
postgresql-15.3/doc/src/sgml/info.sgml
postgresql-15.3/doc/src/sgml/installation.sgml
postgresql-15.3/doc/src/sgml/xplang.sgml
postgresql-15.3/doc/src/Makefile
postgresql-15.3/doc/KNOWN_BUGS
postgresql-15.3/doc/Makefile
postgresql-15.3/doc/TODO
postgresql-15.3/doc/MISSING_FEATURES
postgresql-15.3/.cirrus.yml
postgresql-15.3/HISTORY
postgresql-15.3/Makefile
postgresql-15.3/README
postgresql-15.3/COPYRIGHT
postgresql-15.3/GNUmakefile.in
postgresql-15.3/.gitattributes
postgresql-15.3/aclocal.m4
postgresql-15.3/INSTALL
postgres@andrea:/usr/src/pgsql$ ls
postgresql-15.3/  postgresql-15.3.tar.gz
postgres@andrea:/usr/src/pgsql$ _
```

Figure 30. Decompressing the .tar.gz file

- The above creates a directory called postgresql-15.3, we move into this directory.
  First, we run the configuration using the script: **`./configure –prefix=/usr/local/pgsql –with-openssl`,** where prefix is the installation directory that we created earlier

```
checking for builtin __sync int64 atomic operations... yes
checking for builtin __atomic int32 atomic operations... yes
checking for builtin __atomic int64 atomic operations... yes
checking for __get_cpuid... yes
checking for __cpuid... no
checking for _mm_crc32_u8 and _mm_crc32_u32 with CFLAGS=... no
checking for _mm_crc32_u8 and _mm_crc32_u32 with CFLAGS=-msse4.2... yes
checking for __crc32cb, __crc32ch, __crc32cw, and __crc32cd with CFLAGS=... no
checking for __crc32cb, __crc32ch, __crc32cw, and __crc32cd with CFLAGS=-march=armv8-a+crc... no
checking which CRC-32C implementation to use... SSE 4.2 with runtime check
checking for library containing sem_init... none required
checking which semaphore API to use... unnamed POSIX
checking which random number source to use... OpenSSL
checking for xmllint... /usr/bin/xmllint
checking for xsltproc... no
checking for fop... no
checking for dbtoepub... no
checking whether gcc supports -Wl,--as-needed... yes
configure: using compiler=gcc (GCC) 14.2.0
configure: using CFLAGS=-Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -We
rror=vla -Wendif-labels -Wmissing-format-attribute -Wimplicit-fallthrough=3 -Wcast-function-type -Wf
ormat-security -fno-strict-aliasing -fwrapv -fexcess-precision=standard -Wno-format-truncation -Wno-
stringop-truncation -O2
configure: using CPPFLAGS= -D_GNU_SOURCE
configure: using LDFLAGS=  -Wl,--as-needed
configure: creating ./config.status
config.status: creating GNUmakefile
config.status: creating src/Makefile.global
config.status: creating src/include/pg_config.h
config.status: creating src/include/pg_config_ext.h
config.status: creating src/interfaces/ecpg/include/ecpg_config.h
config.status: linking src/backend/port/tas/dummy.s to src/backend/port/tas.s
config.status: linking src/backend/port/posix_sema.c to src/backend/port/pg_sema.c
config.status: linking src/backend/port/sysv_shmem.c to src/backend/port/pg_shmem.c
config.status: linking src/include/port/linux.h to src/include/pg_config_os.h
config.status: linking src/makefiles/Makefile.linux to src/Makefile.port
postgres@andrea:/usr/src/pgsql/postgresql-15.3$ _
```

Figure 31. Configure script

- We proceed with the compilation using the command `**make**`

```
make -C access all
make[3]: Entering directory '/usr/src/pgsql/postgresql-15.3/src/backend/access'
make -C brin all
make[4]: Entering directory '/usr/src/pgsql/postgresql-15.3/src/backend/access/brin'
gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Werror=vla -Wendif-lab
els -Wmissing-format-attribute -Wimplicit-fallthrough=3 -Wcast-function-type -Wformat-security -fno-
strict-aliasing -fwrapv -fexcess-precision=standard -Wno-format-truncation -Wno-stringop-truncation
-O2 -I../../../../src/include  -D_GNU_SOURCE   -c -o brin.o brin.c
gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Werror=vla -Wendif-lab
els -Wmissing-format-attribute -Wimplicit-fallthrough=3 -Wcast-function-type -Wformat-security -fno-
strict-aliasing -fwrapv -fexcess-precision=standard -Wno-format-truncation -Wno-stringop-truncation
-O2 -I../../../../src/include  -D_GNU_SOURCE   -c -o brin_bloom.o brin_bloom.c
gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Werror=vla -Wendif-lab
els -Wmissing-format-attribute -Wimplicit-fallthrough=3 -Wcast-function-type -Wformat-security -fno-
strict-aliasing -fwrapv -fexcess-precision=standard -Wno-format-truncation -Wno-stringop-truncation
-O2 -I../../../../src/include  -D_GNU_SOURCE   -c -o brin_inclusion.o brin_inclusion.c
gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Werror=vla -Wendif-lab
els -Wmissing-format-attribute -Wimplicit-fallthrough=3 -Wcast-function-type -Wformat-security -fno-
strict-aliasing -fwrapv -fexcess-precision=standard -Wno-format-truncation -Wno-stringop-truncation
-O2 -I../../../../src/include  -D_GNU_SOURCE   -c -o brin_minmax.o brin_minmax.c
gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Werror=vla -Wendif-lab
els -Wmissing-format-attribute -Wimplicit-fallthrough=3 -Wcast-function-type -Wformat-security -fno-
strict-aliasing -fwrapv -fexcess-precision=standard -Wno-format-truncation -Wno-stringop-truncation
-O2 -I../../../../src/include  -D_GNU_SOURCE   -c -o brin_minmax_multi.o brin_minmax_multi.c
gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Werror=vla -Wendif-lab
els -Wmissing-format-attribute -Wimplicit-fallthrough=3 -Wcast-function-type -Wformat-security -fno-
strict-aliasing -fwrapv -fexcess-precision=standard -Wno-format-truncation -Wno-stringop-truncation
-O2 -I../../../../src/include  -D_GNU_SOURCE   -c -o brin_pageops.o brin_pageops.c
gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Werror=vla -Wendif-lab
els -Wmissing-format-attribute -Wimplicit-fallthrough=3 -Wcast-function-type -Wformat-security -fno-
strict-aliasing -fwrapv -fexcess-precision=standard -Wno-format-truncation -Wno-stringop-truncation
-O2 -I../../../../src/include  -D_GNU_SOURCE   -c -o brin_revmap.o brin_revmap.c
gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Werror=vla -Wendif-lab
els -Wmissing-format-attribute -Wimplicit-fallthrough=3 -Wcast-function-type -Wformat-security -fno-
strict-aliasing -fwrapv -fexcess-precision=standard -Wno-format-truncation -Wno-stringop-truncation
-O2 -I../../../../src/include  -D_GNU_SOURCE   -c -o brin_tuple.o brin_tuple.c
_
```

Figure 32. Compilation process

```
re64/d/guile-3.0.9-x86_64-2.txz.asc...
--2024-10-12 18:48:10--  http://ftp.slackware-brasil.com.br/slackware64-current/slackware64/d/guile-
3.0.9-x86_64-2.txz.asc
Resolving ftp.slackware-brasil.com.br (ftp.slackware-brasil.com.br)... 200.137.217.134
Connecting to ftp.slackware-brasil.com.br (ftp.slackware-brasil.com.br)|200.137.217.134|:80... conne
cted.
HTTP request sent, awaiting response... 200 OK
Length: 195 [text/plain]
Saving to: ■//var/cache/packages/./slackware64/d/guile-3.0.9-x86_64-2.txz.asc■

//var/cache/packages/./s 100%[===================================>]     195  --.-KB/s    in 0s

2024-10-12 18:48:15 (8.28 MB/s) - ■//var/cache/packages/./slackware64/d/guile-3.0.9-x86_64-2.txz.asc
■ saved [195/195]

       Package guile-3.0.9-x86_64-2.txz is already in cache - not downloading
       Installing guile-3.0.9-x86_64-2...
Verifying package guile-3.0.9-x86_64-2.txz.
Installing package guile-3.0.9-x86_64-2.txz:
PACKAGE DESCRIPTION:
# guile (GNU extension language library)
#
# This is Guile, Project GNU's extension language library. Guile is an
# interpreter for Scheme, packaged as a library that you can link into
# your applications to give them their own scripting language. Guile
# will eventually support other languages as well, giving users of
# Guile-based applications a choice of languages.
#
Executing install script for guile-3.0.9-x86_64-2.txz.
Package guile-3.0.9-x86_64-2.txz installed.
Searching for NEW configuration files...
               No .new files found.

root@andrea:~/postgresql-15.3# make
make: error while loading shared libraries: libffi.so.8: cannot open shared object file: No such fil
e or directory
root@andrea:~/postgresql-15.3# _
```

Figure 33. Compilation error

- As we saw in *figure 33*, we encountered a compilation error because we didn't install the essential libraries that `make` requires, so we proceed to install them:
  - ✓ libffi
  - ✓ libguile-3.0.so.1
  - ✓ libunistring.so.5
  - ✓ libfl.so.2

```
                     Downloading http://ftp.slackware-brasil.com.br/slackware64-current/./slackwa
re64/l/libffi-3.4.6-x86_64-1.txz.asc...
--2024-10-12 18:49:14--  http://ftp.slackware-brasil.com.br/slackware64-current/slackware64/l/libff
i-3.4.6-x86_64-1.txz.asc
Resolving ftp.slackware-brasil.com.br (ftp.slackware-brasil.com.br)... 200.137.217.134
Connecting to ftp.slackware-brasil.com.br (ftp.slackware-brasil.com.br)|200.137.217.134|:80... conne
cted.
HTTP request sent, awaiting response... 200 OK
Length: 163 [text/plain]
Saving to: ■//var/cache/packages/./slackware64/l/libffi-3.4.6-x86_64-1.txz.asc■

//var/cache/packages/./s 100%[===================================>]     163  --.-KB/s    in 0s

2024-10-12 18:49:19 (6.32 MB/s) - ■//var/cache/packages/./slackware64/l/libffi-3.4.6-x86_64-1.txz.as
c■ saved [163/163]

       Package libffi-3.4.6-x86_64-1.txz is already in cache - not downloading
       Installing libffi-3.4.6-x86_64-1...
Verifying package libffi-3.4.6-x86_64-1.txz.
Installing package libffi-3.4.6-x86_64-1.txz:
PACKAGE DESCRIPTION:
# libffi (A Portable Foreign Function Interface Library)
#
# FFI stands for Foreign Function Interface. A foreign function
# interface is the popular name for the interface that allows code
# written in one language to call code written in another language.
# The libffi library really only provides the lowest, machine dependent
# layer of a fully featured foreign function interface.
#
# Homepage: https://sourceware.org/libffi/
#
Executing install script for libffi-3.4.6-x86_64-1.txz.
Package libffi-3.4.6-x86_64-1.txz installed.
Searching for NEW configuration files...
               No .new files found.

root@andrea:~/postgresql-15.3# _
```

Figure 34. Installing libraries for 'make'

- We run the compilation again and wait until it finishes

```
../../src/interfaces/libpq -lpq -L../../../src/port -L../../../src/common   -Wl,--as-needed -Wl,-rp
ath,'/usr/local/pgsql/lib',--enable-new-dtags  -lpgcommon -lpgport -lssl -lcrypto -lz -ledit -lm  -o
 isolationtester
gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Werror=vla -Wendif-lab
els -Wmissing-format-attribute -Wimplicit-fallthrough=3 -Wcast-function-type -Wformat-security -fno-
strict-aliasing -fwrapv -fexcess-precision=standard -Wno-format-truncation -Wno-stringop-truncation
-O2 -I. -I. -I../../../src/interfaces/libpq -I./../regress -I../../../src/include  -D_GNU_SOURCE   -
c -o isolation_main.o isolation_main.c
make -C ../../../src/test/regress pg_regress.o
make[3]: Entering directory '/usr/src/pgsql/postgresql-15.3/src/test/regress'
make -C ../../../src/port all
make[4]: Entering directory '/usr/src/pgsql/postgresql-15.3/src/port'
make[4]: Nothing to be done for 'all'.
make[4]: Leaving directory '/usr/src/pgsql/postgresql-15.3/src/port'
make -C ../../../src/common all
make[4]: Entering directory '/usr/src/pgsql/postgresql-15.3/src/common'
make[4]: Nothing to be done for 'all'.
make[4]: Leaving directory '/usr/src/pgsql/postgresql-15.3/src/common'
make[3]: Leaving directory '/usr/src/pgsql/postgresql-15.3/src/test/regress'
rm -f pg_regress.o && ln -s ../../../src/test/regress/pg_regress.o .
gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Werror=vla -Wendif-lab
els -Wmissing-format-attribute -Wimplicit-fallthrough=3 -Wcast-function-type -Wformat-security -fno-
strict-aliasing -fwrapv -fexcess-precision=standard -Wno-format-truncation -Wno-stringop-truncation
-O2 isolation_main.o pg_regress.o -L../../../src/port -L../../../src/common   -Wl,--as-needed -Wl,-r
path,'/usr/local/pgsql/lib',--enable-new-dtags  -lpgcommon -lpgport -lssl -lcrypto -lz -ledit -lm  -
o pg_isolation_regress
make[2]: Leaving directory '/usr/src/pgsql/postgresql-15.3/src/test/isolation'
make -C test/perl all
make[2]: Entering directory '/usr/src/pgsql/postgresql-15.3/src/test/perl'
make[2]: Nothing to be done for 'all'.
make[2]: Leaving directory '/usr/src/pgsql/postgresql-15.3/src/test/perl'
make[1]: Leaving directory '/usr/src/pgsql/postgresql-15.3/src'
make -C config all
make[1]: Entering directory '/usr/src/pgsql/postgresql-15.3/config'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/usr/src/pgsql/postgresql-15.3/config'
postgres@andrea:/usr/src/pgsql/postgresql-15.3$ _
```

Figure 35. Compilation process

- Once the compilation of the packages is complete, we proceed with the installation using the command **`make install`**

```
make[4]: Nothing to be done for 'all'.
make[4]: Leaving directory '/usr/src/pgsql/postgresql-15.3/src/port'
make -C ../../../src/common all
make[4]: Entering directory '/usr/src/pgsql/postgresql-15.3/src/common'
make[4]: Nothing to be done for 'all'.
make[4]: Leaving directory '/usr/src/pgsql/postgresql-15.3/src/common'
make[3]: Leaving directory '/usr/src/pgsql/postgresql-15.3/src/interfaces/libpq'
make -C ../../../src/port all
make[3]: Entering directory '/usr/src/pgsql/postgresql-15.3/src/port'
make[3]: Nothing to be done for 'all'.
make[3]: Leaving directory '/usr/src/pgsql/postgresql-15.3/src/port'
make -C ../../../src/common all
make[3]: Entering directory '/usr/src/pgsql/postgresql-15.3/src/common'
make[3]: Nothing to be done for 'all'.
make[3]: Leaving directory '/usr/src/pgsql/postgresql-15.3/src/common'
/usr/bin/mkdir -p '/usr/local/pgsql/lib/pgxs/src/test/isolation'
/usr/bin/ginstall -c  pg_isolation_regress '/usr/local/pgsql/lib/pgxs/src/test/isolation/pg_isolatio
n_regress'
/usr/bin/ginstall -c  isolationtester '/usr/local/pgsql/lib/pgxs/src/test/isolation/isolationtester'
make[2]: Leaving directory '/usr/src/pgsql/postgresql-15.3/src/test/isolation'
make -C test/perl install
make[2]: Entering directory '/usr/src/pgsql/postgresql-15.3/src/test/perl'
make[2]: Nothing to be done for 'install'.
make[2]: Leaving directory '/usr/src/pgsql/postgresql-15.3/src/test/perl'
/usr/bin/mkdir -p '/usr/local/pgsql/lib/pgxs/src'
/usr/bin/ginstall -c -m 644 Makefile.global '/usr/local/pgsql/lib/pgxs/src/Makefile.global'
/usr/bin/ginstall -c -m 644 Makefile.port '/usr/local/pgsql/lib/pgxs/src/Makefile.port'
/usr/bin/ginstall -c -m 644 ./Makefile.shlib '/usr/local/pgsql/lib/pgxs/src/Makefile.shlib'
/usr/bin/ginstall -c -m 644 ./nls-global.mk '/usr/local/pgsql/lib/pgxs/src/nls-global.mk'
make[1]: Leaving directory '/usr/src/pgsql/postgresql-15.3/src'
make -C config install
make[1]: Entering directory '/usr/src/pgsql/postgresql-15.3/config'
/usr/bin/mkdir -p '/usr/local/pgsql/lib/pgxs/config'
/usr/bin/ginstall -c -m 755 ./install-sh '/usr/local/pgsql/lib/pgxs/config/install-sh'
/usr/bin/ginstall -c -m 755 ./missing '/usr/local/pgsql/lib/pgxs/config/missing'
make[1]: Leaving directory '/usr/src/pgsql/postgresql-15.3/config'
postgres@andrea:/usr/src/pgsql/postgresql-15.3$ _
```

Figure 36. Installation process finished

- We must initialize the database cluster to start using the PostgreSQL Server by running the **initdb** command

```
postgres@andrea:/usr/src/pgsql/postgresql-15.3$ /usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with this locale configuration:
  provider:    libc
  LC_COLLATE:  C
  LC_CTYPE:    en_US.UTF-8
  LC_MESSAGES: en_US.UTF-8
  LC_MONETARY: en_US.UTF-8
  LC_NUMERIC:  en_US.UTF-8
  LC_TIME:     en_US.UTF-8
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are disabled.

creating directory /usr/local/pgsql/data ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... America/Bogota
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok

initdb: warning: enabling "trust" authentication for local connections
initdb: hint: You can change this by editing pg_hba.conf or using the option -A, or --auth-local and
 --auth-host, the next time you run initdb.

Success. You can now start the database server using:

    /usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data -l logfile start

postgres@andrea:/usr/src/pgsql/postgresql-15.3$
```

Figure 37. Initializing the database

- We use **/usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data** to start the server

```
postgres@andrea:/usr/src/pgsql/postgresql-15.3$ /usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data
 -l logfile start
waiting for server to start.... done
server started
postgres@andrea:/usr/src/pgsql/postgresql-15.3$
```

Figure 38. Starting the server

- We access the server using **/usr/local/pgsql/bin/psql**

```
postgres@andrea:/usr/src/pgsql/postgresql-15.3$ /usr/local/pgsql/bin/psql
psql (15.3)
Type "help" for help.

postgres=# _
```

Figure 39. Logging into the PostgreSQL server

b. User and database creation process

- Into the server, we create the users Jorge and Camila and assign a password to each one. Then we create databases for each user: jorge_tourist_sites and camila_tourist_sites

```
postgres=# CREATE USER jorge WITH PASSWORD 'rec0.79.';
CREATE ROLE
postgres=# CREATE USER camila WITH PASSWORD 'rec0.79.';
CREATE ROLE
postgres=# CREATE DATABASE jorge_tourist_sites;
CREATE DATABASE
postgres=# CREATE DATABASE camila_tourist_sites;
CREATE DATABASE
postgres=#
```

Figure 40. Creating users and databases into PostgreSQL

c. Permissions

- We use GRANT to assign permissions to each user

```
postgres=# GRANT CREATE ON SCHEMA public TO camila;
GRANT
postgres=# GRANTE CREATE ON SCHEMA public TO jorge;
ERROR:  syntax error at or near "GRANTE"
LINE 1: GRANTE CREATE ON SCHEMA public TO jorge;
        ^
postgres=# GRANT CREATE ON SCHEMA public TO jorge;
GRANT
postgres=# _
```

Figure 41. Granting permissions to each user

- Before that, we make each user the owner of their respective database

```
postgres=# GRANT CREATE ON SCHEMA public TO jorge;
GRANT
postgres=# ALTER DATABASE camila_tourist_sites OWNER TO camila;
ALTER DATABASE
postgres=# ALTER DATABASE jorge_tourist_sites OWNER TO jorge;
ALTER DATABASE
postgres=# _
```

Figure 42. Making each user the owner of their respective database

d. Table creation and data insertion

i. Database camila_tourist_sites

- To access the database, we use **psql -U [username] -d [database]**

```
postgres@andrea:~$ psql -U camila -d camila_tourist_sites
psql (15.3)
Type "help" for help.

camila_tourist_sites=> _
```

Figure 43. Logging into camila_toruist_sites

- We create the tables based on tourist sites in Colombia. In our case, we create the table cities, which contains the id and name; the table sites, which contains id, name, description, address, schedule, cost, and a foreign key that references the respective city; and finally, the table activities, which contains name, description, max occupation, requirements, and a foreign key that references the sites.

```
camila_tourist_sites=> CREATE TABLE cities (
camila_tourist_sites(> city_id SERIAL PRIMARY KEY,
camila_tourist_sites(> name VARCHAR(100)
camila_tourist_sites(> );
CREATE TABLE
camila_tourist_sites=> CREATE TABLE sites (
camila_tourist_sites(> site_id SERIA PRIMARY KEY,
camila_tourist_sites(>
camila_tourist_sites=> );
ERROR:  syntax error at or near ")"
LINE 1: );
        ^
camila_tourist_sites=> CREATE TABLE sites (
camila_tourist_sites(> site_id SERIAL PRIMARY KEY,
camila_tourist_sites(> name VARCHAR(100),
camila_tourist_sites(> description VARCHAR(250),
camila_tourist_sites(> address VARCHAR(150),
camila_tourist_sites(> schedule VARCHAR(50),
camila_tourist_sites(> cost DECIMAL(10,2)
camila_tourist_sites(> );
CREATE TABLE
camila_tourist_sites=> CREATE TABLE activities (
camila_tourist_sites(> activity_id SERIAL PRIMARY KEY,
camila_tourist_sites(> name VARCHAR(100),
camila_tourist_sites(> description VARCHAR(250),
camila_tourist_sites(> max_occupation INTEGER,
camila_tourist_sites(> requirements VARCHAR(300)
camila_tourist_sites(> );
CREATE TABLE
camila_tourist_sites=> _
```

Figure 44. Creating tables based on Colombia tourist sites into camila_tourist_sites

- In *Figure 23*, we didn't add the respective foreign key. To include it, we use ADD COLUMN to add the attribute and ALTER TABLE to assign it as a foreign key

```
camila_tourist_sites=> ALTER TABLE sites ADD COLUMN city INT;
ALTER TABLE
camila_tourist_sites=> ALTER TABLE activities ADD COLUMN site INT;
ALTER TABLE
camila_tourist_sites=> ALTER TABLE sites
camila_tourist_sites-> ADD CONSTRAINT city
camila_tourist_sites-> FOREIGN KEY (city)sja;
ERROR:  syntax error at or near "sja"
LINE 3: FOREIGN KEY (city)sja;
                          ^
camila_tourist_sites=> ALTER TABLE sites
camila_tourist_sites-> ADD CONSTRAINT FK_site_city
camila_tourist_sites-> FOREIGN KEY (city)
camila_tourist_sites-> REFERENCES cities(city_id)
camila_tourist_sites-> ON DELETE SET NULL;
ALTER TABLE
camila_tourist_sites=> ALTER TABLE activities
camila_tourist_sites-> ADD CONSTRAINT FK_activity_site
camila_tourist_sites-> FOREIGN KEY (site)
camila_tourist_sites-> REFERENCES sites(site_id)
camila_tourist_sites-> ON DELETE CASCADE;
ALTER TABLE
camila_tourist_sites=> _
```

Figure 45. Adding foreign keys into camila_tourist_sites

- Then, we insert data into each table as shown in the following images:

```
camila_tourist_sites=> INSERT INTO cities (name) VALUES ('Medellín');
INSERT 0 1
camila_tourist_sites=> INSERT INTO cities (name) VALUES ('Zipaquirá');
INSERT 0 1
camila_tourist_sites=> INSERT INTO cities (name) VALUES ('Cartagena');
INSERT 0 1
camila_tourist_sites=> INSERT INTO cities (name) VALUES ('Eje Cafetero');
INSERT 0 1
camila_tourist_sites=> SELECT * FROM cities;
 city_id |     name
---------+--------------
       1 | Bogotá
       2 | Medellín
       3 | Zipaquirá
       4 | Cartagena
       5 | Eje Cafetero
(5 rows)

camila_tourist_sites=> _
```

Figure 46. Inserting data into cities table from camila_tourist_sites database

```
INSERT 0 1
camila_tourist_sites=> INSERT INTO sites (name, description, address, schedule, cost, city)
camila_tourist_sites-> VALUES('Catedra de Sal', 'Recinto construido en el interior de las minas de s
al', 'Calle 1 #6 - 00', 'D-D 9:00-17:40',140000,3);
INSERT 0 1
camila_tourist_sites=> INSERT INTO sites (name, description, address, schedule, cost, city)
camila_tourist_sites-> VALUES('Castillo de San Felipe', 'Fortaleza militar', 'Cra 17', 'L-D 7:00-18:
00',33000,4);
INSERT 0 1
camila_tourist_sites=> INSERT INTO sites (name, description, address, schedule, cost, city)
camila_tourist_sites-> VALUES('Murallas de Cartagena', 'Centro historico de la ciudad', 'Cl. de la S
errezuela', '24 horas',0,4);
INSERT 0 1
camila_tourist_sites=> INSERT INTO sites (name, description, address, schedule, cost, city)
camila_tourist_sites-> VALUES('Parque del café', 'Parque temático', 'Km 6, La Tebaida-Montenegro', '
L-D 9:00-18:00',77000,5);
INSERT 0 1
camila_tourist_sites=> SELECT * FROM sites;
 site_id |          name          |                        description                         |
  address           |    schedule    |   cost    | city
---------+------------------------+------------------------------------------------------------+---------
--------------------+----------------+-----------+------
       1 | Monserrate             | Es el más conocido de los cerros Orientales de Bogotá | Cra. 2da
 este #21-48        | L-S 6:30-12:00 |      0.00 |    1
       2 | Jardin Botánico        | Centro de investigación cientifica                    | Calle 73
 #51 D 14           | M-D 9:00-16:00 |      0.00 |    2
       3 | Catedra de Sal         | Recinto construido en el interior de las minas de sal | Calle 1
#6 - 00             | D-D 9:00-17:40 | 140000.00 |    3
       4 | Castillo de San Felipe | Fortaleza militar                                     | Cra 17
                    | L-D 7:00-18:00 |  33000.00 |    4
       5 | Murallas de Cartagena  | Centro historico de la ciudad                         | Cl. de l
a Serrezuela        | 24 horas       |      0.00 |    4
       6 | Parque del café        | Parque temático                                       | Km 6, La
 Tebaida-Montenegro | L-D 9:00-18:00 |  77000.00 |    5
(6 rows)

camila_tourist_sites=>
```

Figure 47. Inserting data into sites table from camila_tourist_sites database

```
camila_tourist_sites=> INSERT INTO activities (name, description, max_occupation, requirements,site)

camila_tourist_sites-> VALUES('Teleférico','Subida a monserrate por teleférico',40,'No consumir comi
da, ropa cómoda y calzado adecuado, personas mayores a 14 años',1);
INSERT 0 1
camila_tourist_sites=> INSERT INTO activities (name, description, requirements,site)
camila_tourist_sites-> VALUES('Grupos sociales', 'Grupos que planean realizar actividades en espacio
s libres', 'Diligenciar el formulario, no ingreso de mascotas, menores de edad con adulto',2);
INSERT 0 1
camila_tourist_sites=> INSERT INTO activities (name, description, requirements,site)
camila_tourist_sites-> VALUES('Ruta del minero', 'Vive la experiencia de trabajar como minero', 'No
tabaco, alcohol o drogas, menores acompañados por adultos',3);
INSERT 0 1
camila_tourist_sites=> INSERT INTO activities (name, description, max_occupation,requirements,site)
camila_tourist_sites-> VALUES ('Recorrido','Recorre los lugares y conoce el castillo',40,'Ropa cómod
a, menores de edad acompañados por un adulto',4);
INSERT 0 1
camila_tourist_sites=> INSERT INTO activities (name, description, max_occupation,requirements,site)
camila_tourist_sites-> VALUES ('Karts','Compite con tus familiares las clasicas carreras de Karts',1
,'Ropa cómoda, estatura minima 150 centimetros, velocidad maxima de 30Kms/h',6);
INSERT 0 1
camila_tourist_sites=>
```

Figure 48. Inserting data into activities table from camila_tourist_sites database

ii. Database jorge_tourist_sites

- We create the same tables as in the camila_tourist_sites database

```
jorge_tourist_sites=> CREATE TABLE cities (
jorge_tourist_sites(> city_id SERIAL PRIMARY KEY,
jorge_tourist_sites(> city_name VARCHAR(100)
jorge_tourist_sites(> );
CREATE TABLE
jorge_tourist_sites=>
```

Figure 49. Creating cities table based on Colombia tourist sites into jorge_tourist_sites

```
jorge_tourist_sites=> CREATE TABLE sites (
jorge_tourist_sites(> site_id SERIAL PRIMARY KEY,
jorge_tourist_sites(> name VARCHAR(100),
jorge_tourist_sites(> description VARCHAR(250),
jorge_tourist_sites(> address VARCHAR(150),
jorge_tourist_sites(> schedule VARCHAR(50),
jorge_tourist_sites(> cost DECIMAL(10,2),
jorge_tourist_sites(> city INT REFERENCES cities(city_id)
jorge_tourist_sites(> );
CREATE TABLE
jorge_tourist_sites=> CREATE TABLE activities (
jorge_tourist_sites(> activity_id SERIAL PRIMARY KEY,
jorge_tourist_sites(> name VARCHAR(100),
jorge_tourist_sites(> description VARCHAR(250),
jorge_tourist_sites(> max_occupation INTEGER,
jorge_tourist_sites(> requirements VARCHAR(300),
jorge_tourist_sites(> site INT REFERENCES sites(site_id)
jorge_tourist_sites(> );
CREATE TABLE
jorge_tourist_sites=>
```

Figure 50. Creating tables based on Colombia tourist sites into jorge_tourist_sites

- Then, we insert data into each table as shown in the following image:

```
jorge_tourist_sites=> INSERT INTO cities (city_name) VALUES ('Bogotá');
INSERT 0 1
jorge_tourist_sites=> INSERT INTO cities (city_name) VALUES ('Cali');
INSERT 0 1
jorge_tourist_sites=> INSERT INTO cities (city_name) VALUES ('San Andrés');
INSERT 0 1
jorge_tourist_sites=> INSERT INTO sites (name, description, address, schedule, cost, city)
jorge_tourist_sites-> VALUES('Museo del Oro', 'Colecciones arqueológicas patrimoniales', 'Cra.6 #15-
88', 'M-S 9:00-19:00 y D y festivos 10:00-17:00',5000,1);
INSERT 0 1
jorge_tourist_sites=> INSERT INTO sites (name, description, address, schedule, cost, city)
jorge_tourist_sites-> VALUES('Zoológico', 'Cuidado y consevación de la vida', 'Cra.2a Oe.', 'D-D 9:0
0-16:30',33000,2);
INSERT 0 1
jorge_tourist_sites=> INSERT INTO sites (name, description, address, schedule, cost, city)
jorge_tourist_sites-> VALUES('Playa Spratt Bight', 'Mar de 7 colores, playa de relajación', 'Cl. 1 #
2-24', 'D-D 8:00-12:00 y 14:00-17:00',0,3);
INSERT 0 1
jorge_tourist_sites=> INSERT INTO activities (name, description, max_occupation,requirements,site)
jorge_tourist_sites-> VALUES ('Visita guiada','Conoceremos la coleccion del Museo del Oro y recorrem
os el barrio de la candelaria',10,'No ingreso de mascotas,no se incluye almuerzo y reserva 36 horas
antes',1);
INSERT 0 1
jorge_tourist_sites=> INSERT INTO activities (name, description, max_occupation,requirements,site)
jorge_tourist_sites-> VALUES ('Tour de Bote','Navega en una lancha hasta Jhonny Cay',8,'Calzado como
do, gafas de sol, sombrero, tarjeta de identidad, no mascotas ni fumar',3);
INSERT 0 1
jorge_tourist_sites=>
```

Figure 51. Inserting data into each table from jorge_tourist_sites database

2. SQL Server - Windows Server

a. Installation process

- We begin the installation of SQL Server Express to compile the databases and manage the server

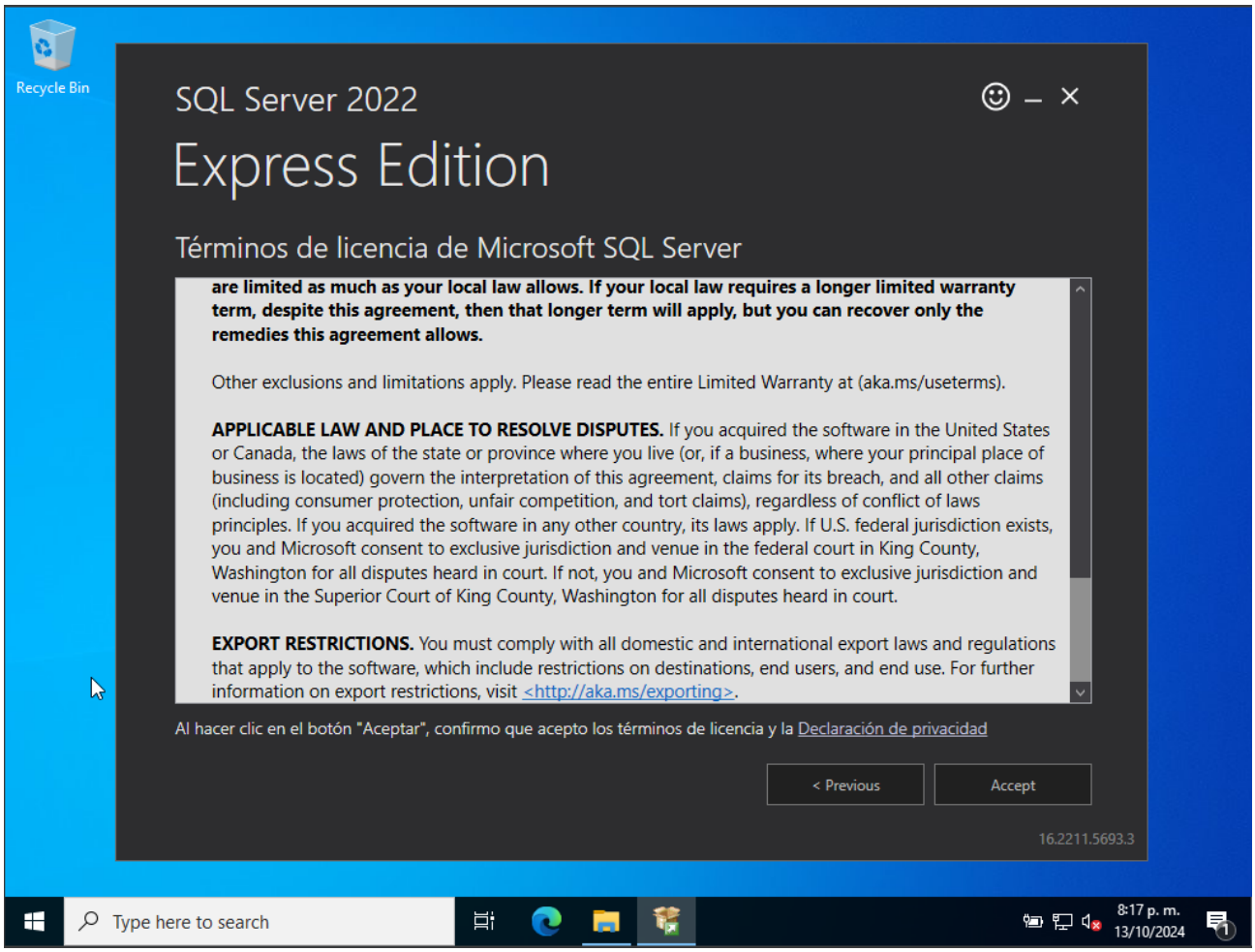


Figure 52. Installation of SQL Server Express

- We need to wait until the installation is complete

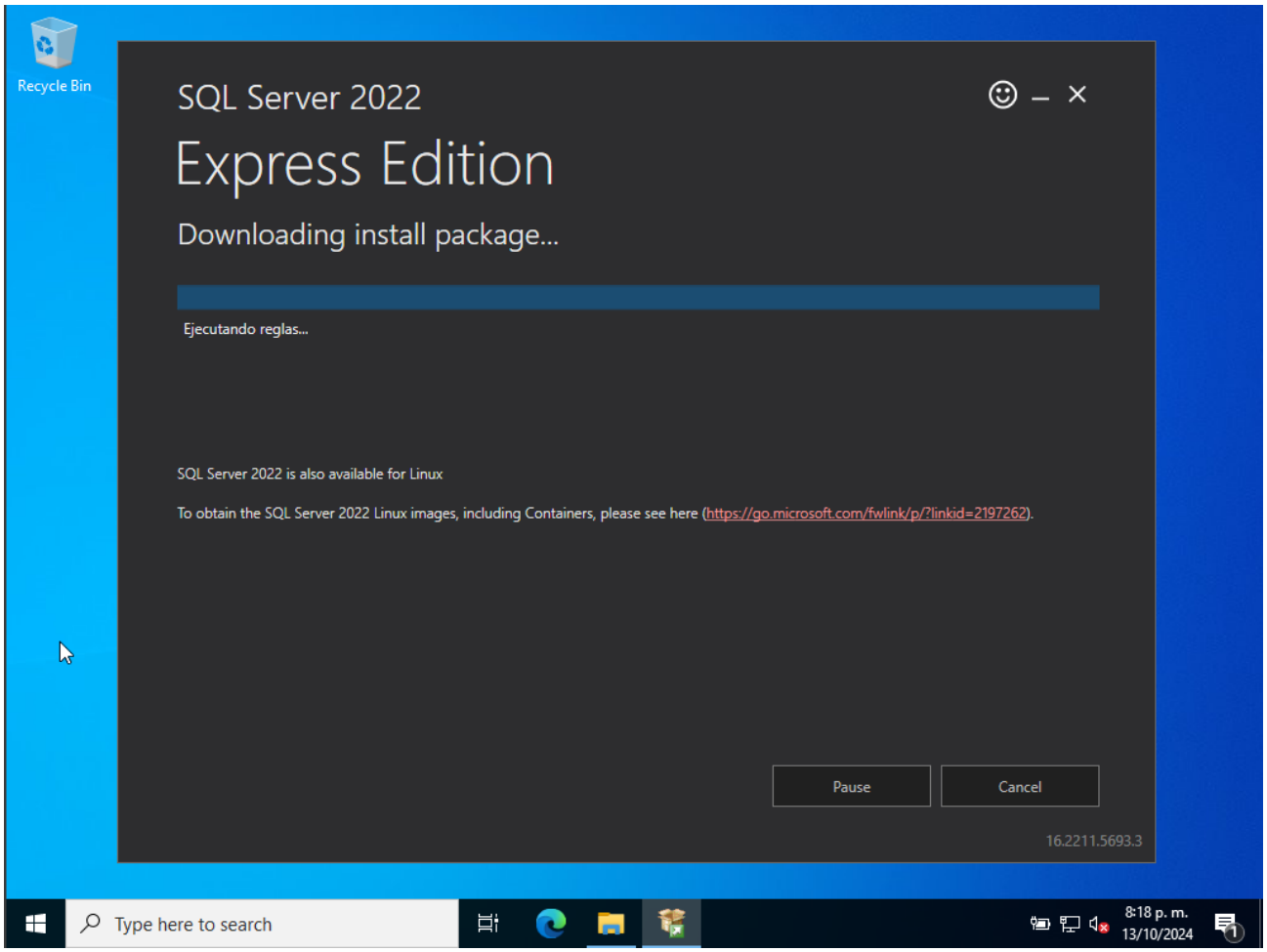


Figure 53. Installation process of SQL Server Express

- When the installation finishes, we can see the credentials to access the server. Then, we choose 'Install SSMS'

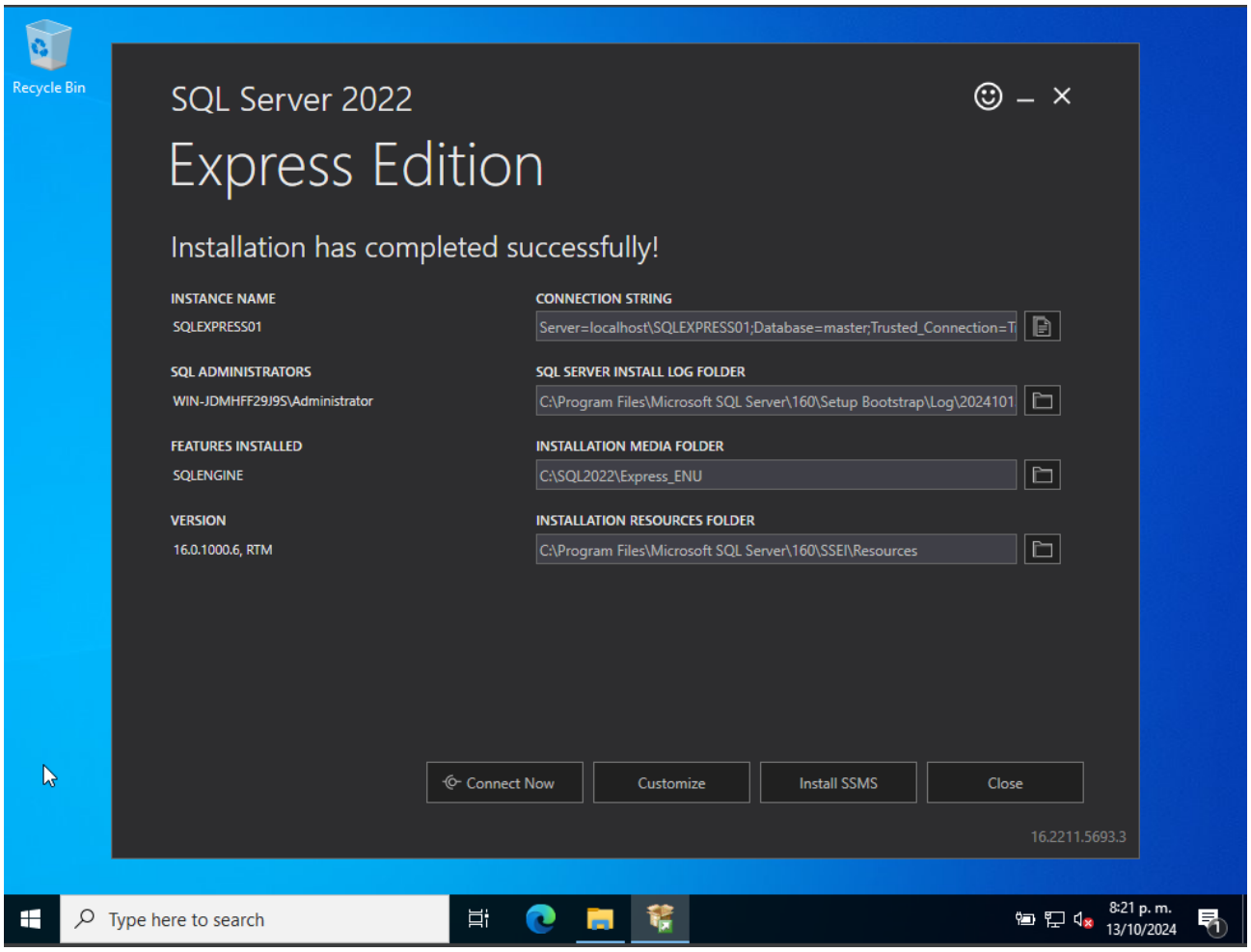


Figure 54. Credentials to access the server

- When we click on the button, it opens a window to download Server Management Studio (SSMS)

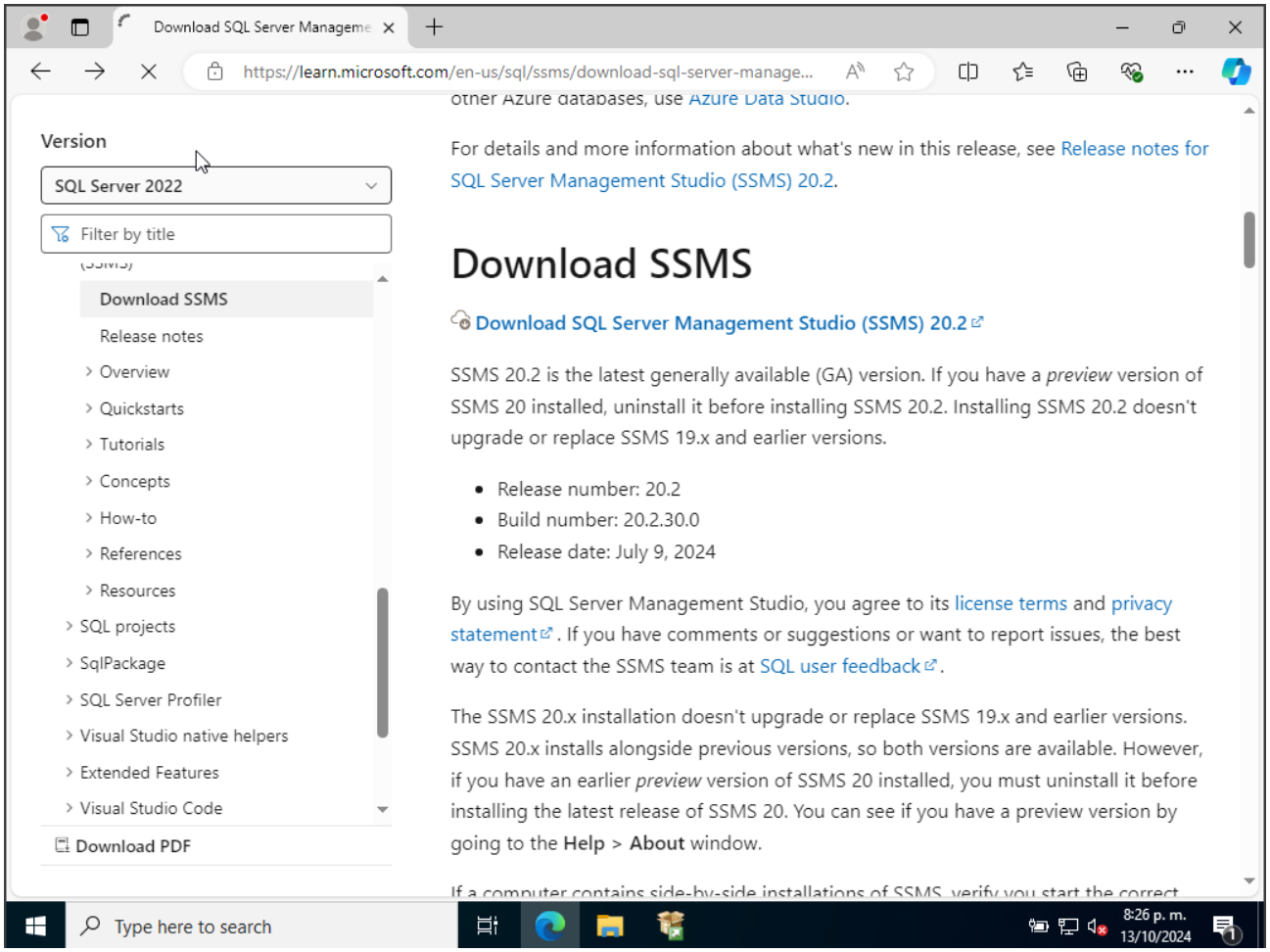


Figure 55. SSMS Download Page

- We download SSMS 20.2

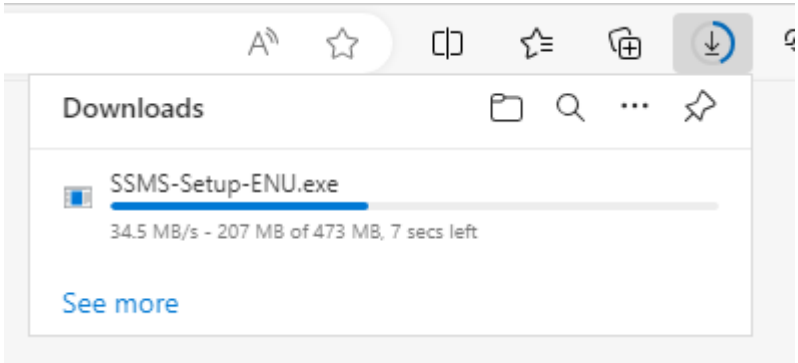


Figure 56. Downloading the SSMS application

- When the application installation opens, we click on 'install' and wait until it finishes

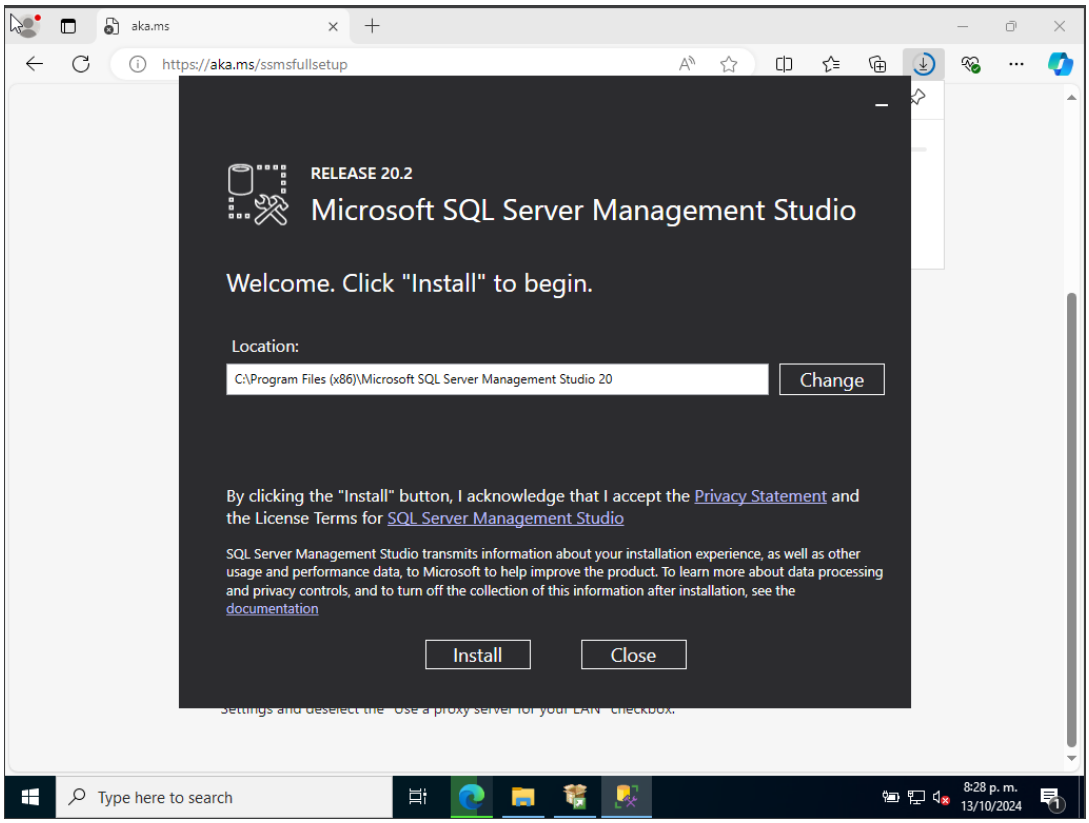


Figure 57. Start of the application installation

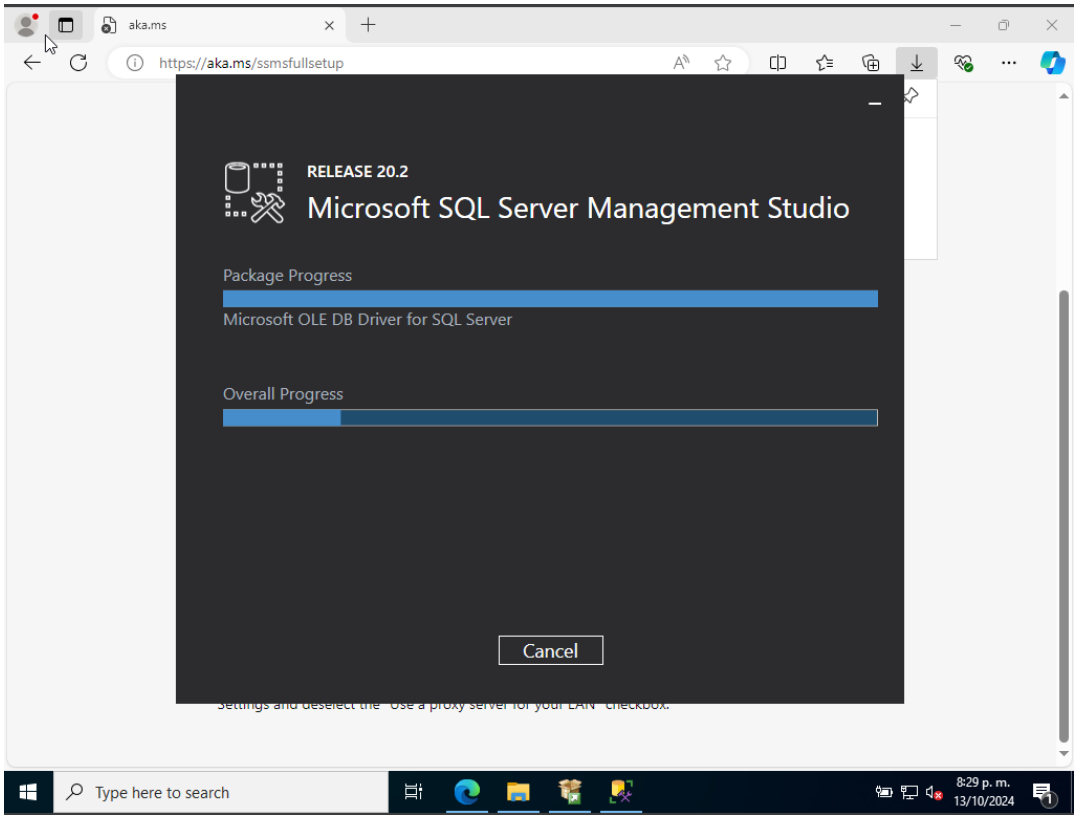


Figure 58. SSMS Installation process

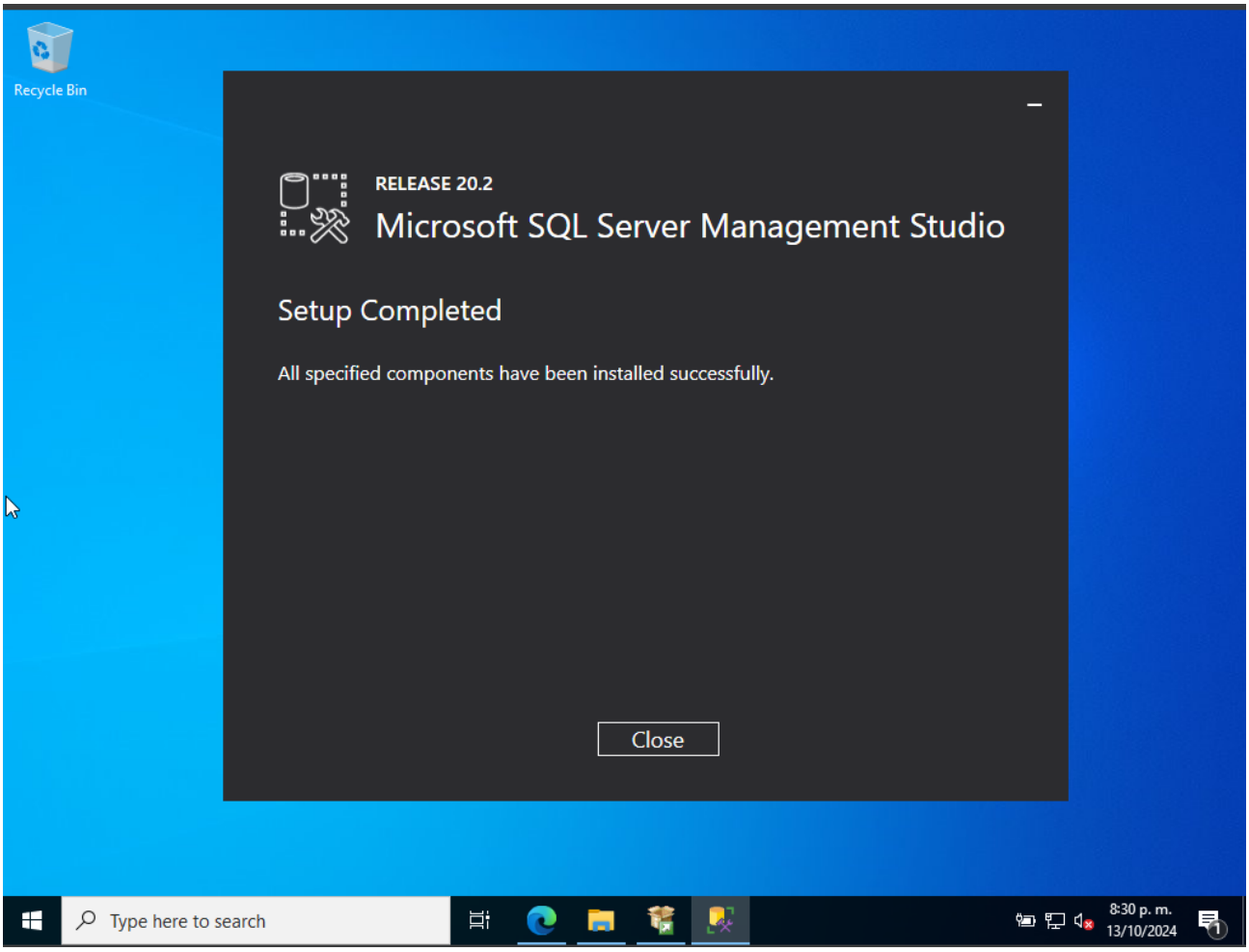


Figure 59. SSMS Installation completed

b. User creation process

- We open the SSMS application and enter the credentials shown in *Figure 54*

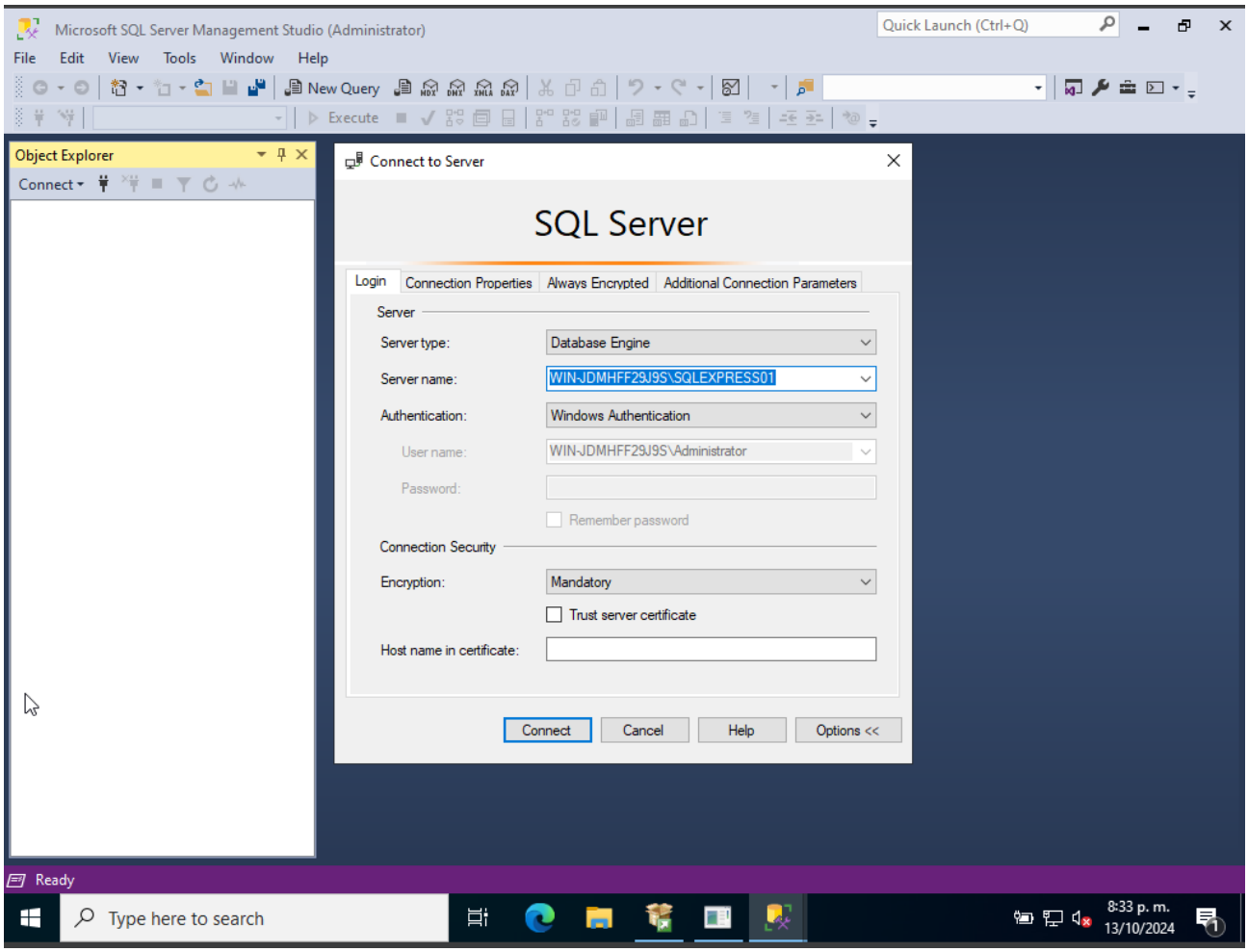


Figure 60. Logging into SQL Server

- We click on Security, then Login, right click, and choose New Login

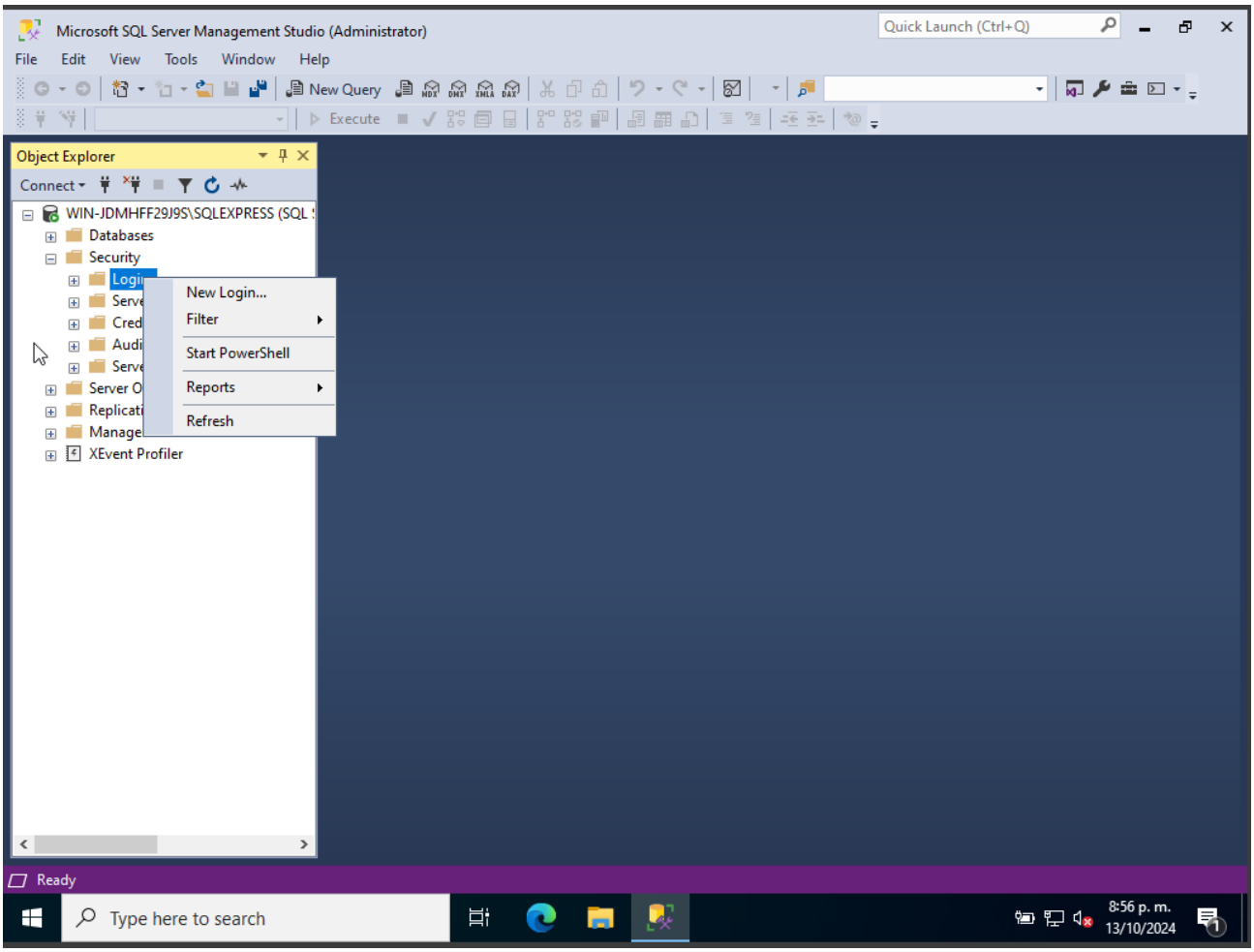


Figure 61. Creating new users in SQL Server

- We enter the user's name, choose the 'SQL authentication' option, and then we write a password

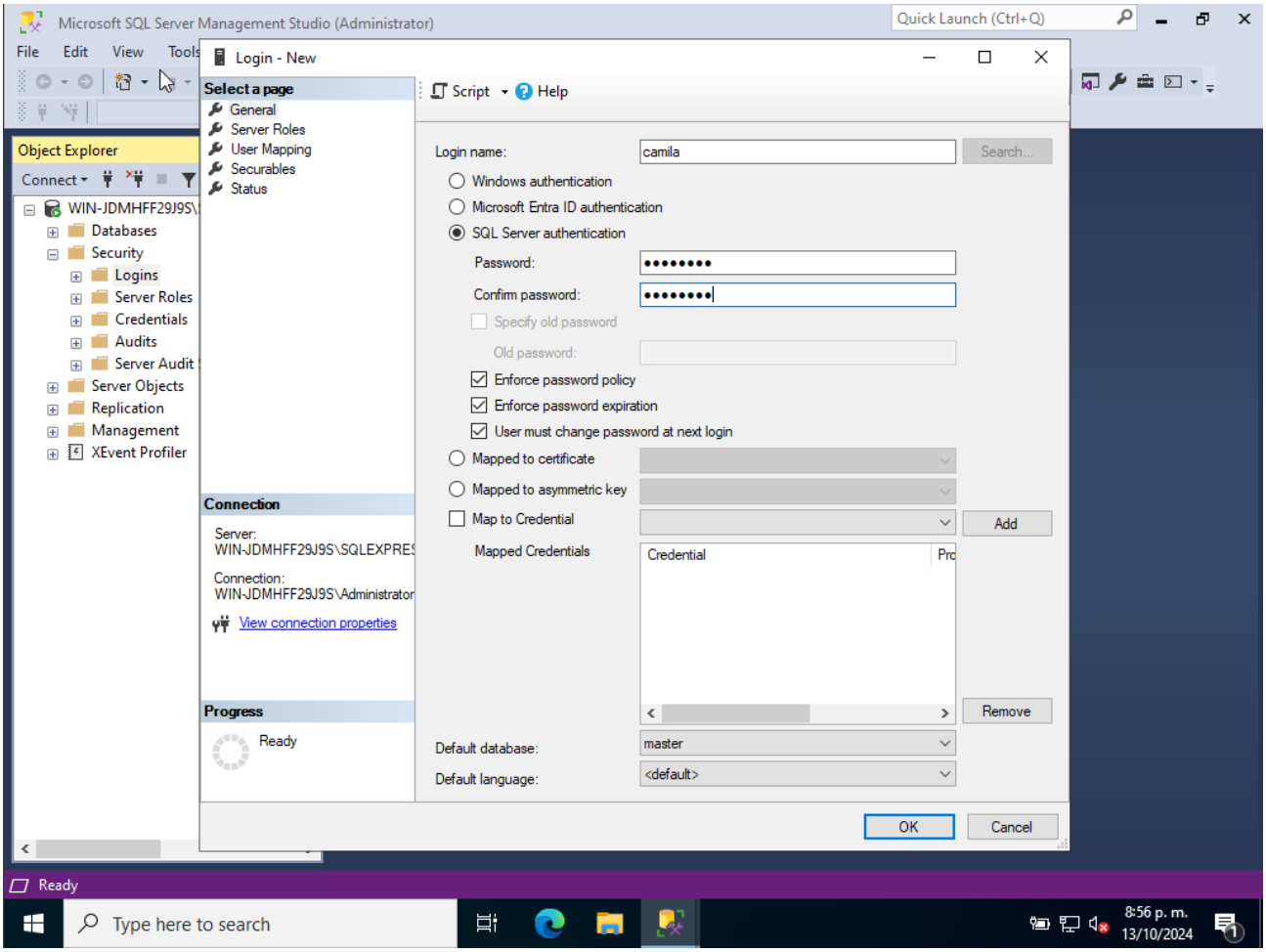


Figure 62. Creating user Camila in SQL Server

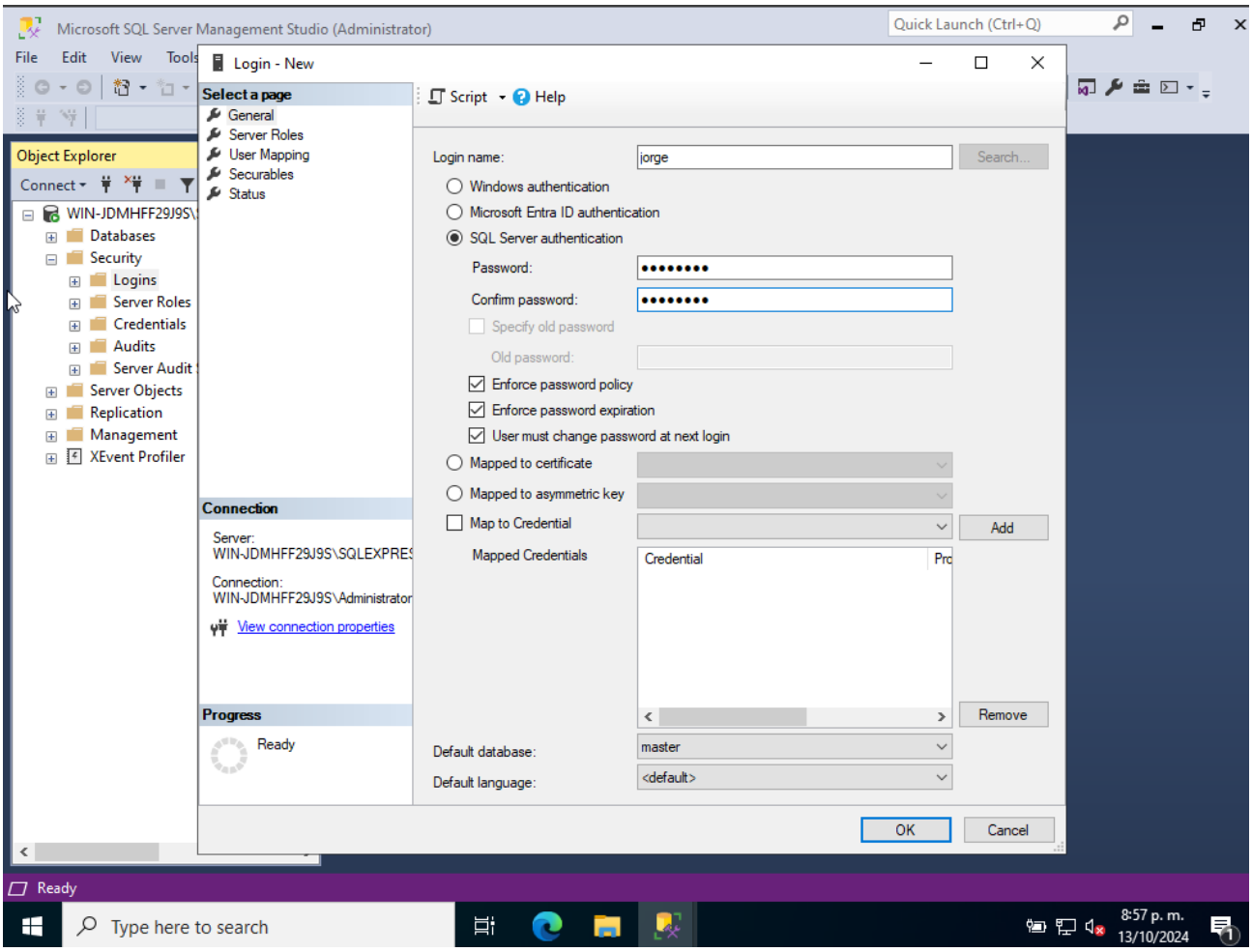


Figure 63. Creating user Jorge in SQL Server

c. Permissions

- To create new databases, click on ‘Database’, then select ‘New Database’

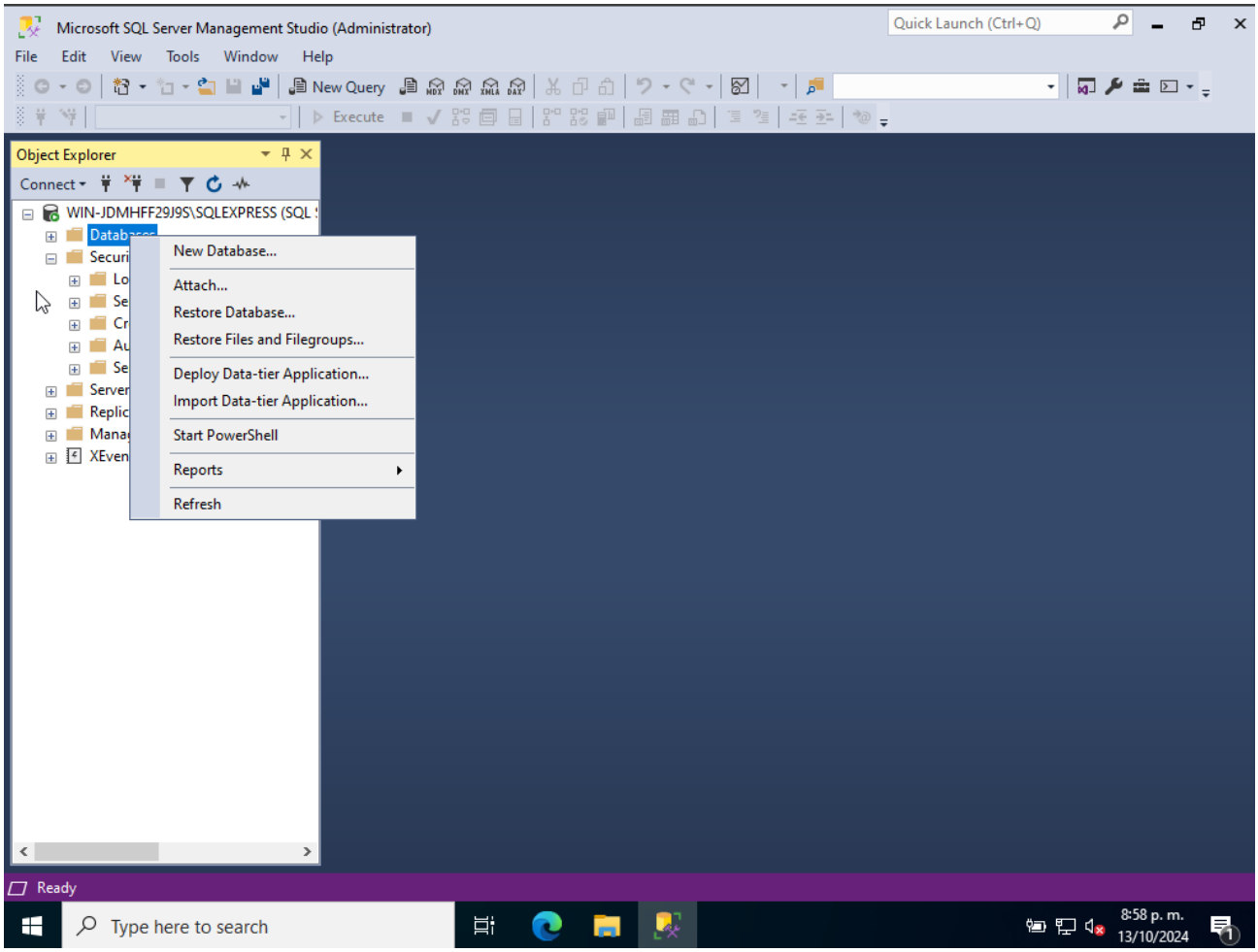


Figure 64. Creating databases in SQL Server

- We enter the database name, in this case 'camila_activity_schedule' then we click on 'Owner' and choose the user who will be the owner of the database, in this case the user 'Camila'

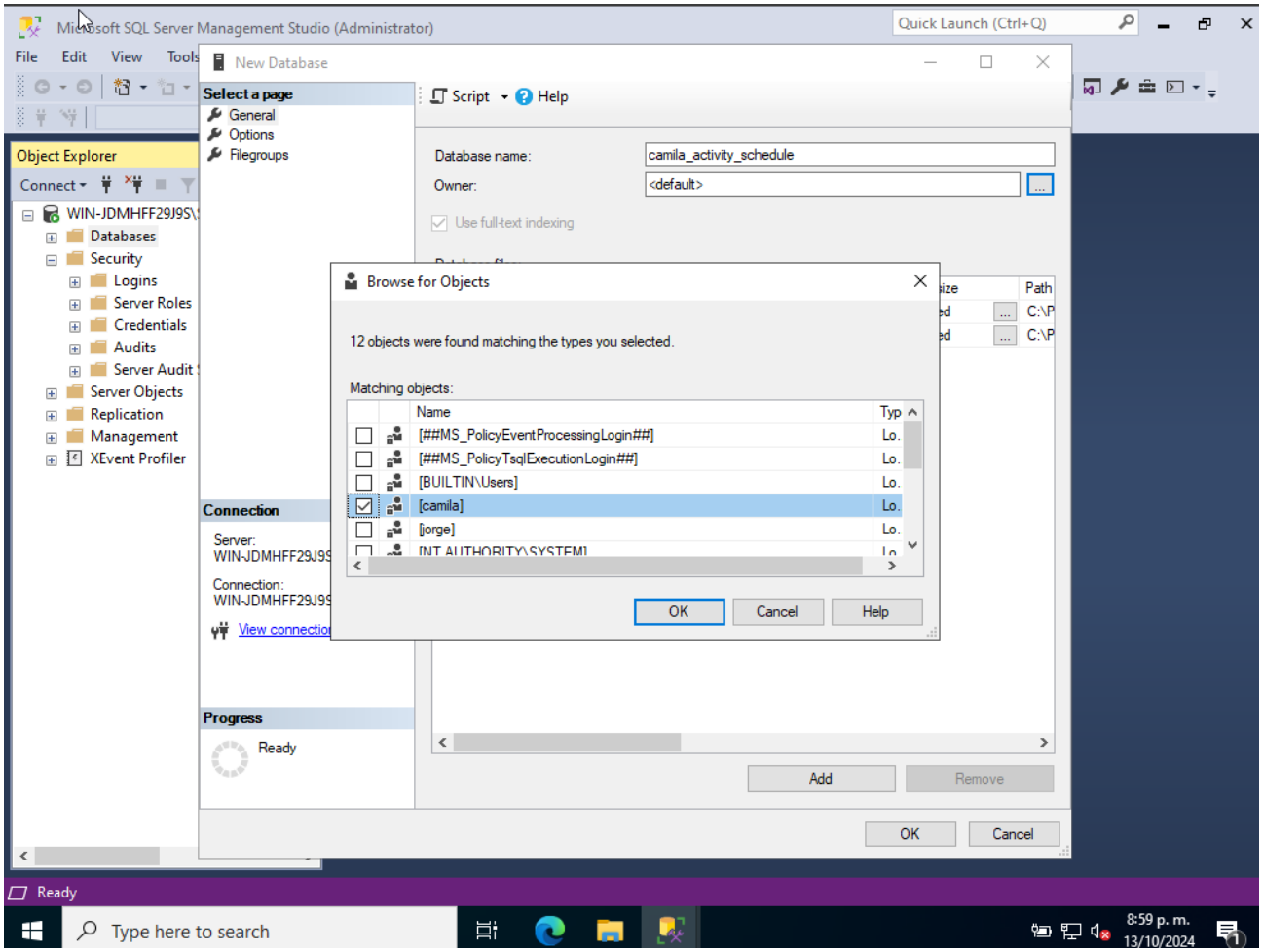


Figure 65. Creating the database camila_activity_schedule

- When we finish, click on 'Ok'

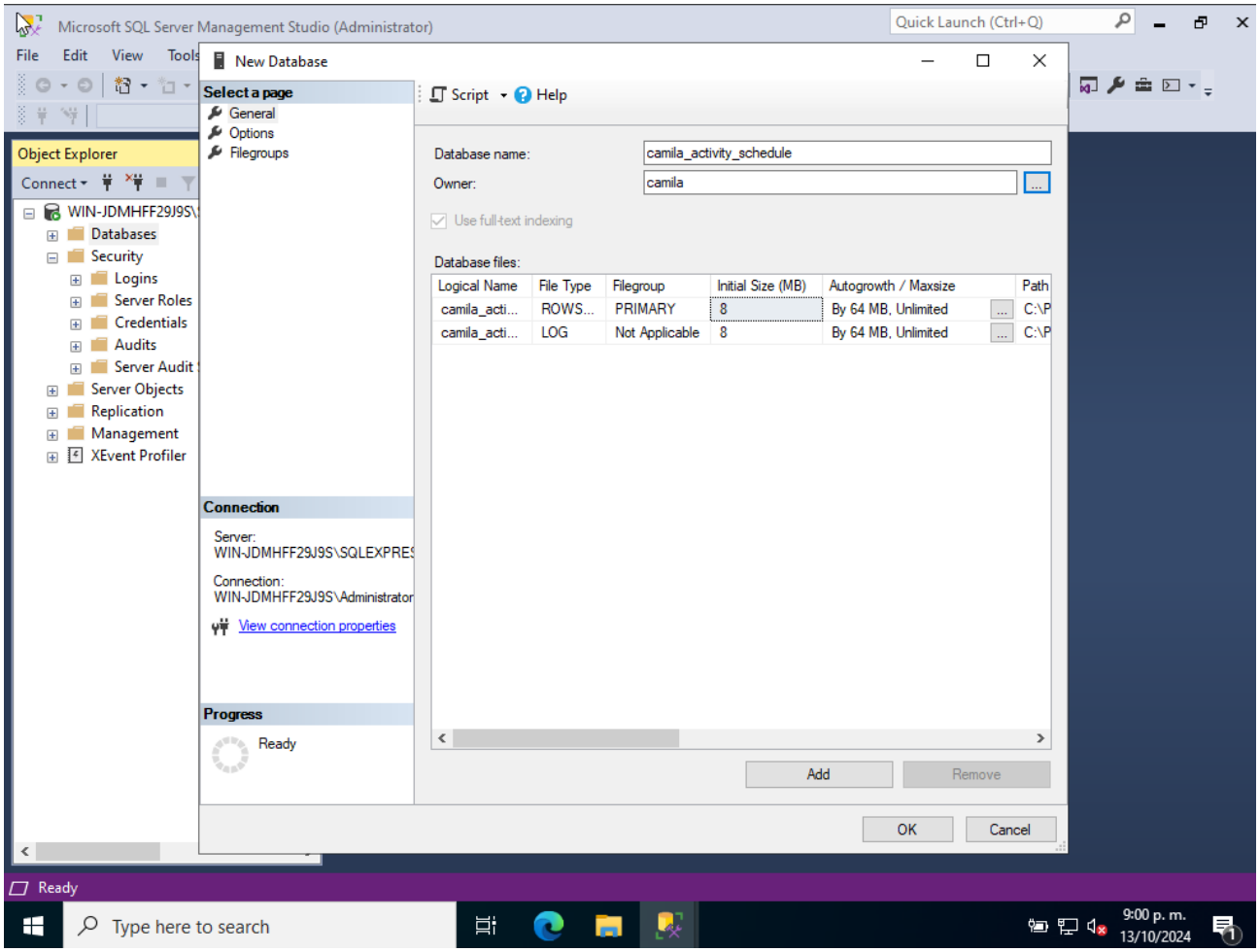


Figure 66. Checking the data creation of camila_activity_schedule

- We follow the same steps as for the last database; in this case, the owner will be Jorge, and the database name is 'jorge_activity_schedule'

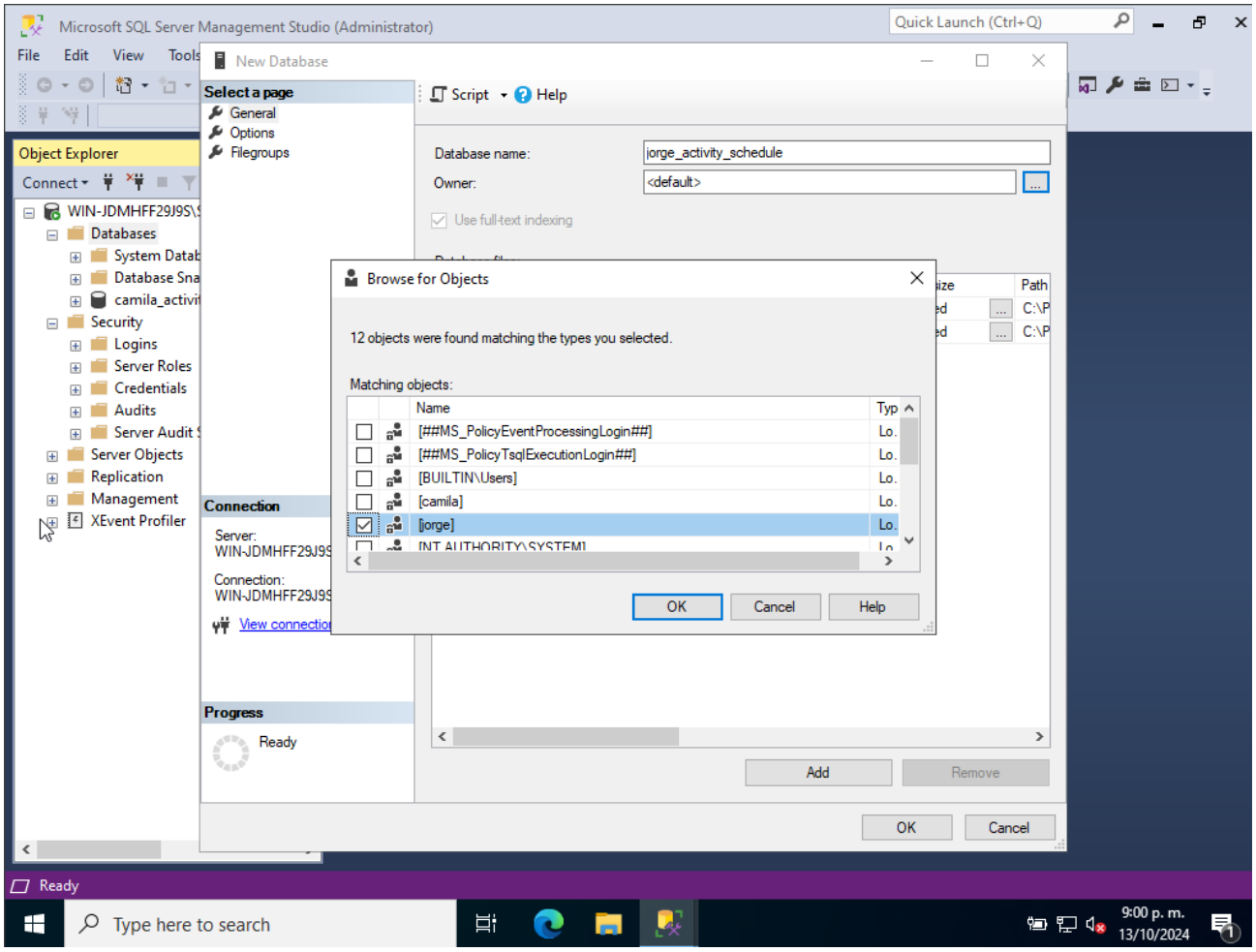


Figure 67. Creating the database jorge_activity_schedule

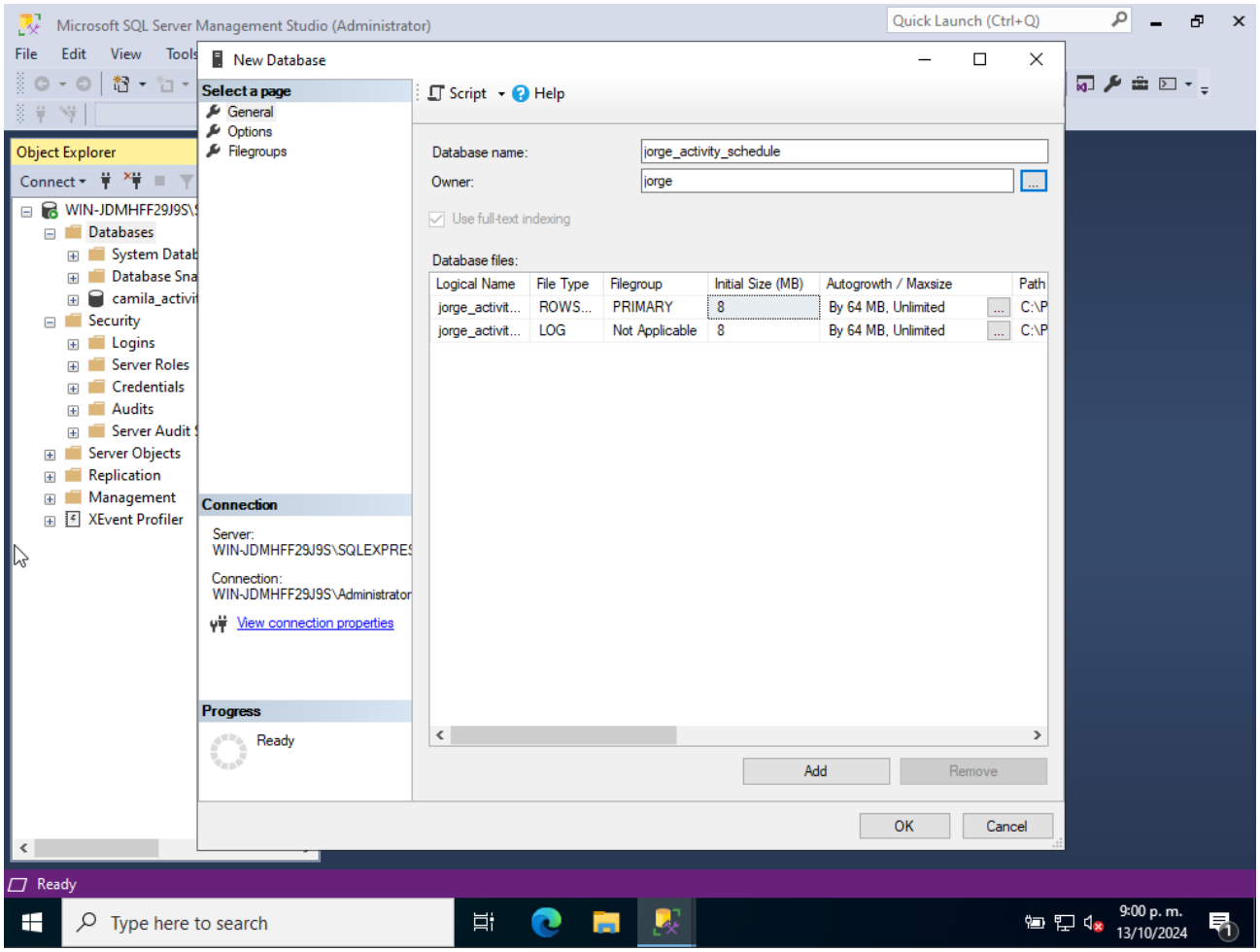


Figure 68. Checking the data creation of jorge_activity_schedule

d. Table creation and data insertion

i. Database camila_activity_schedule

- We click on the database where we want to add data, in this case ‘camila_activity_schedule’ then we click on ‘New Query’

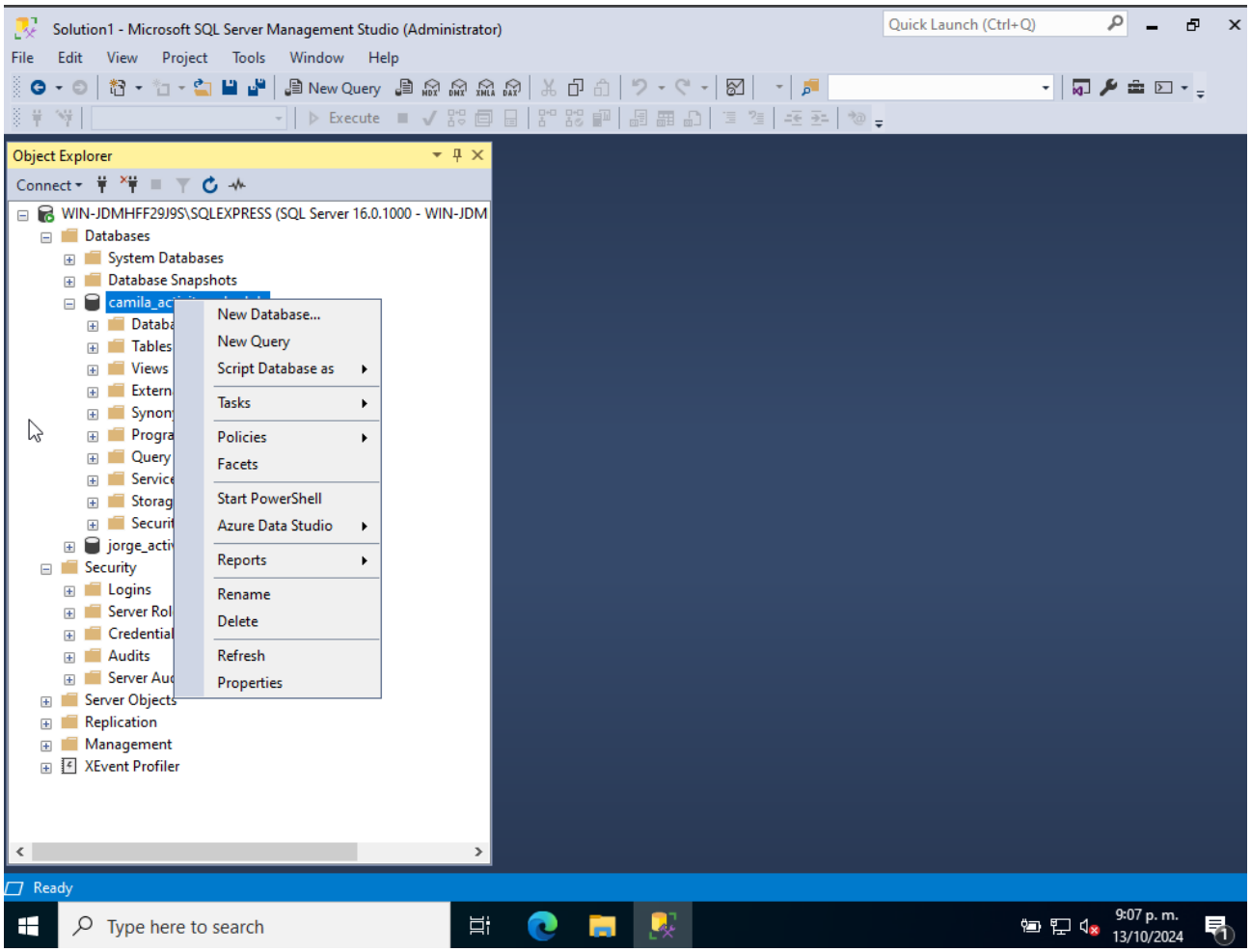


Figure 69. Creating query for camila_activity_schedule

- We create the tables based on our Monthly Activity Schedule. We create the table categories, which contains the id and name; the table activities, which contains id, name, description, date, priority, and a foreign key that references the respective category; and finally, the table reminders, which contains id, reminder date, message, and a foreign key that references the activity

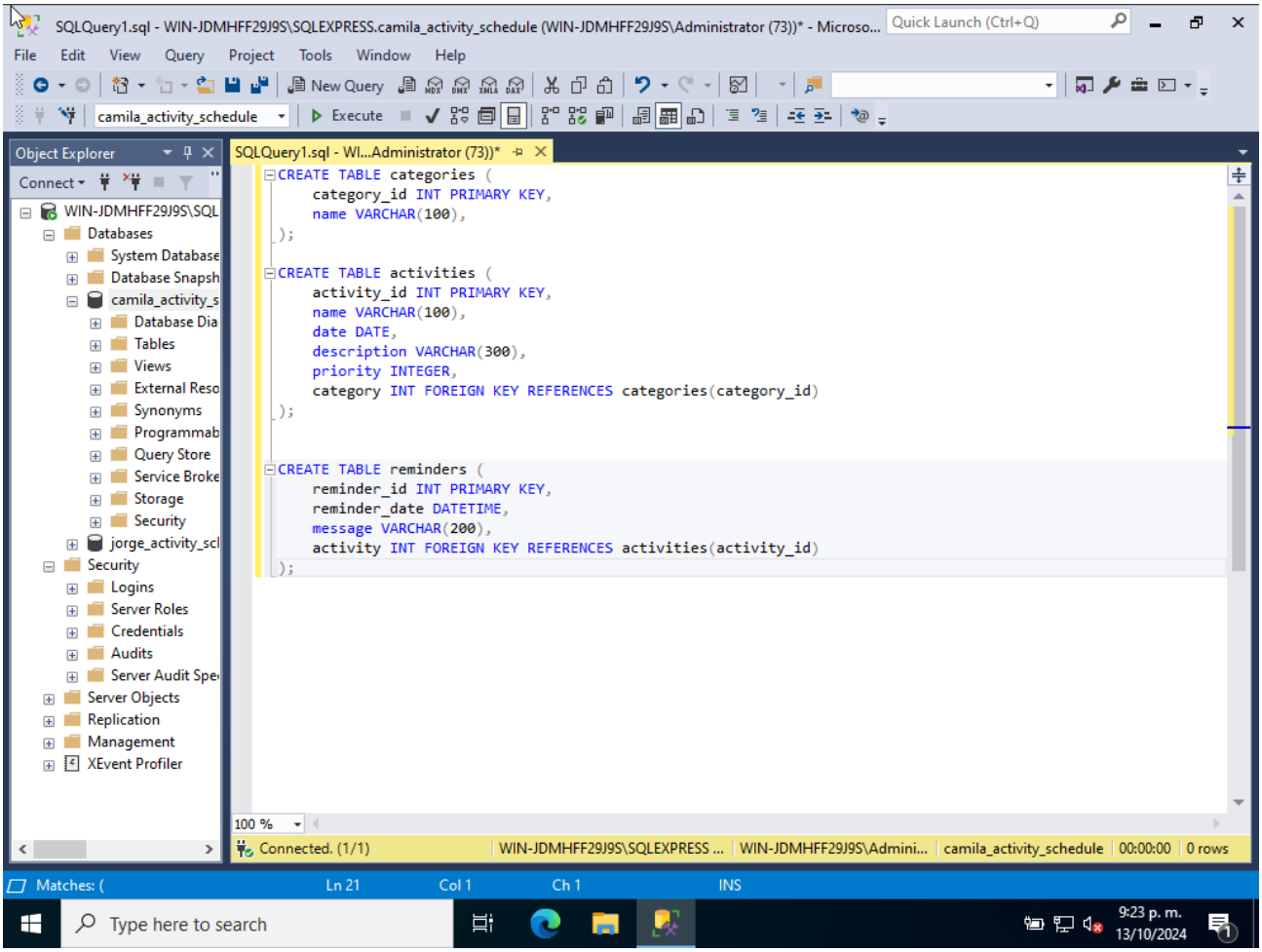


Figure 70. Creating tables for camila_activity_schedule

- Then, we insert data into each table as shown in the following images:

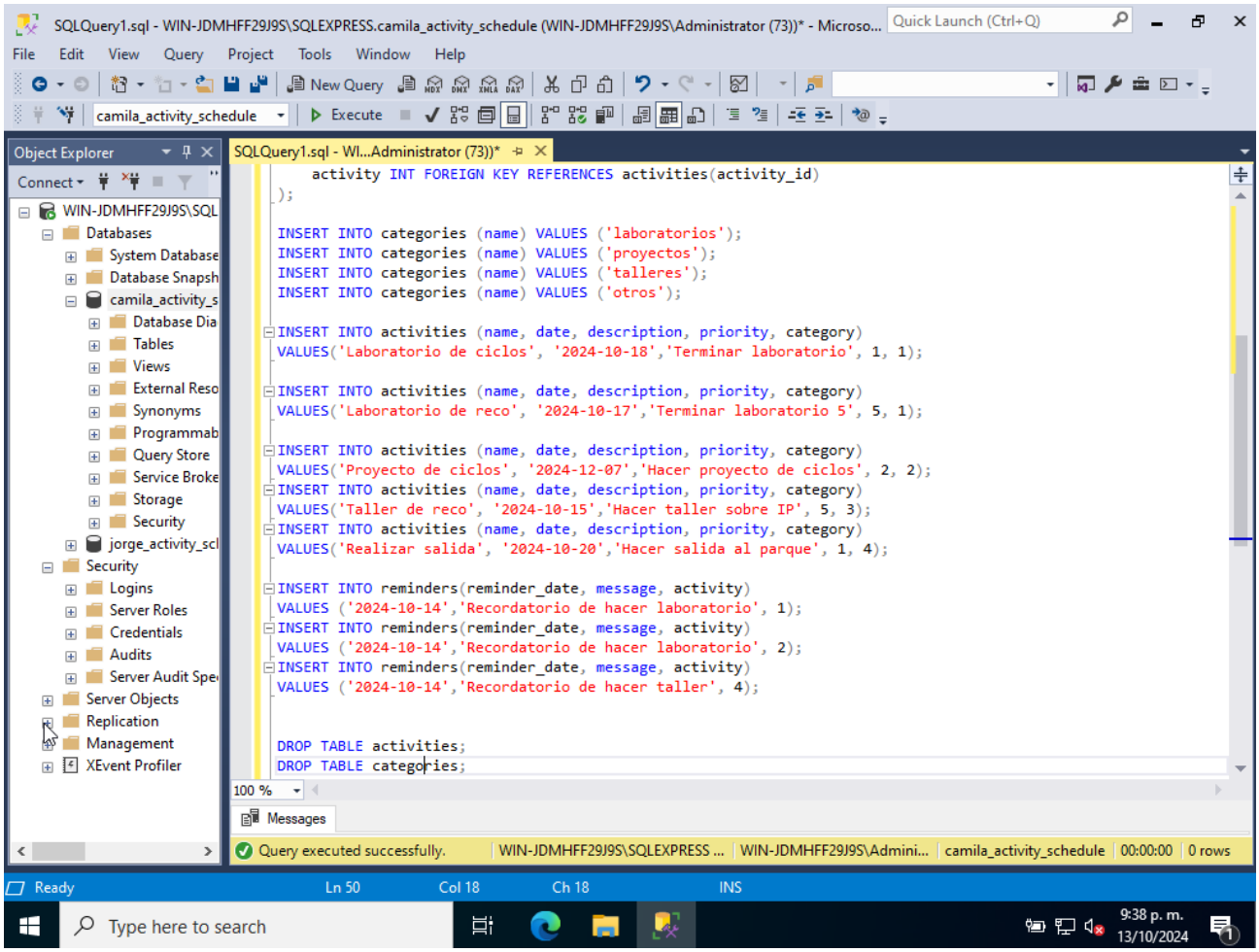


Figure 71. Inserting data into each table from camila_activity_schedule

- Finally, we execute all these instructions

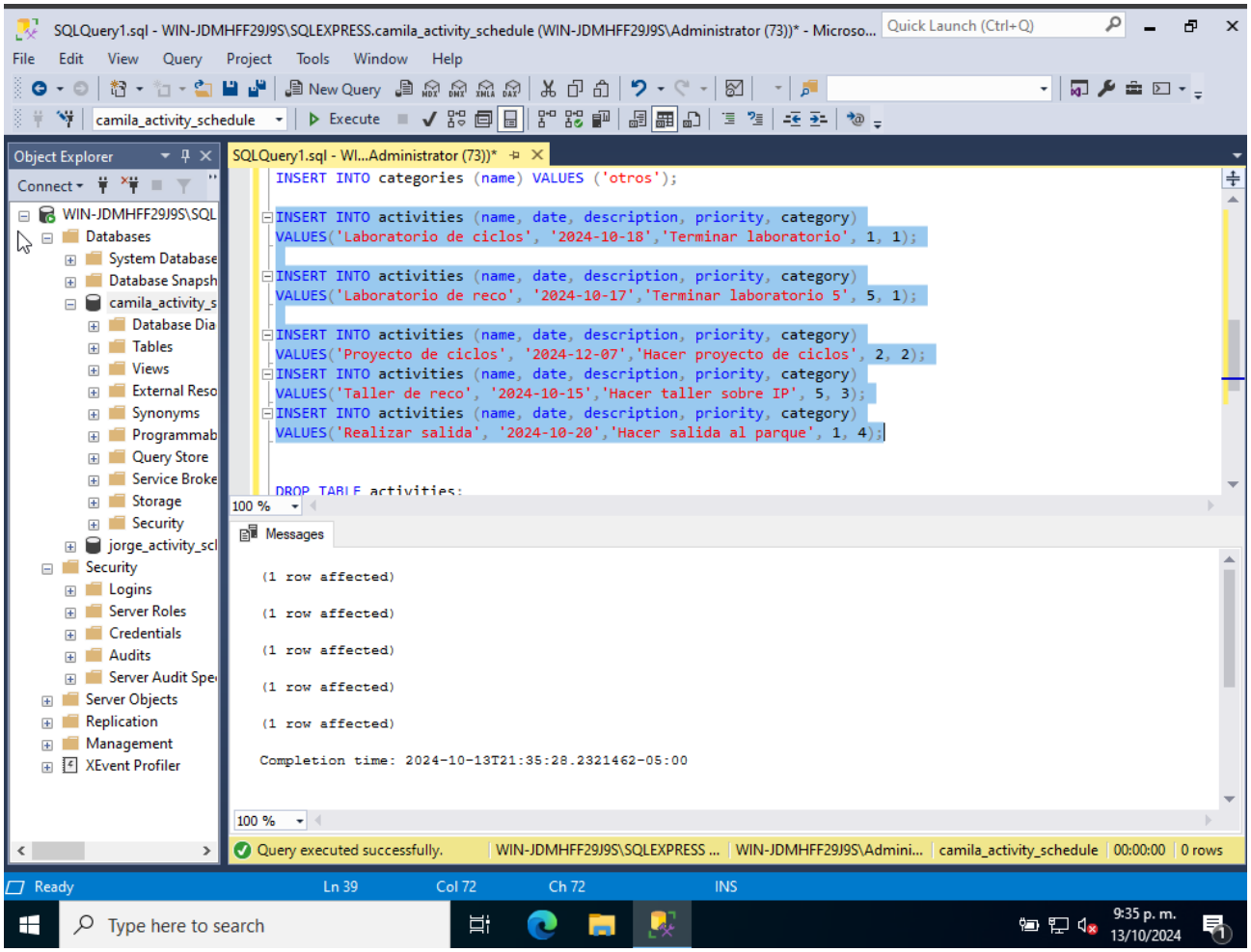


Figure 72. Executing instructions from camila_activity_schedule

ii. Database jorge_activity_schedule

- We create the same tables as in the camila_activity_schedule database

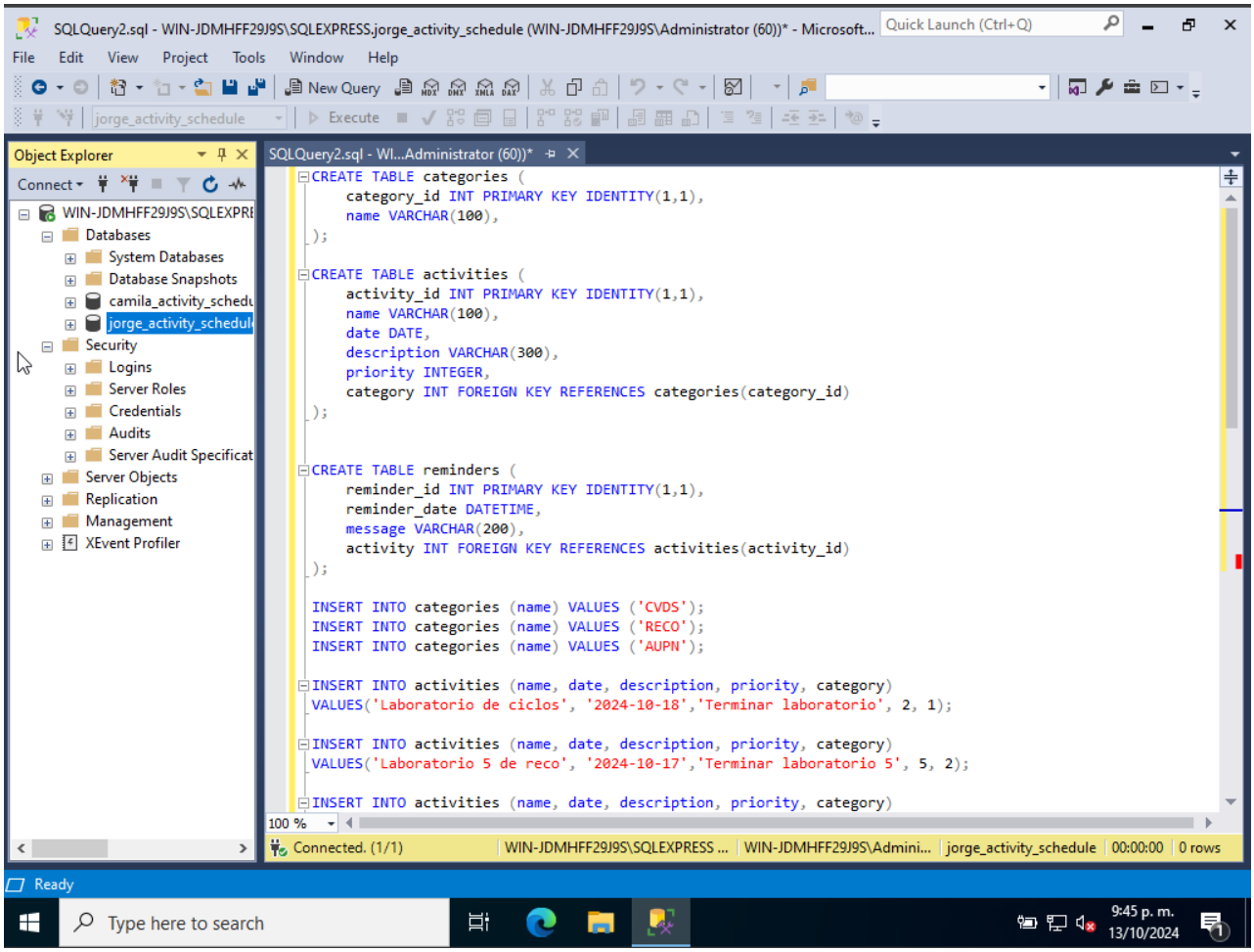


Figure 73. Creating tables for jorge_activity_schedule

- Then, we insert data into each table and execute these instructions as shown in the following image:

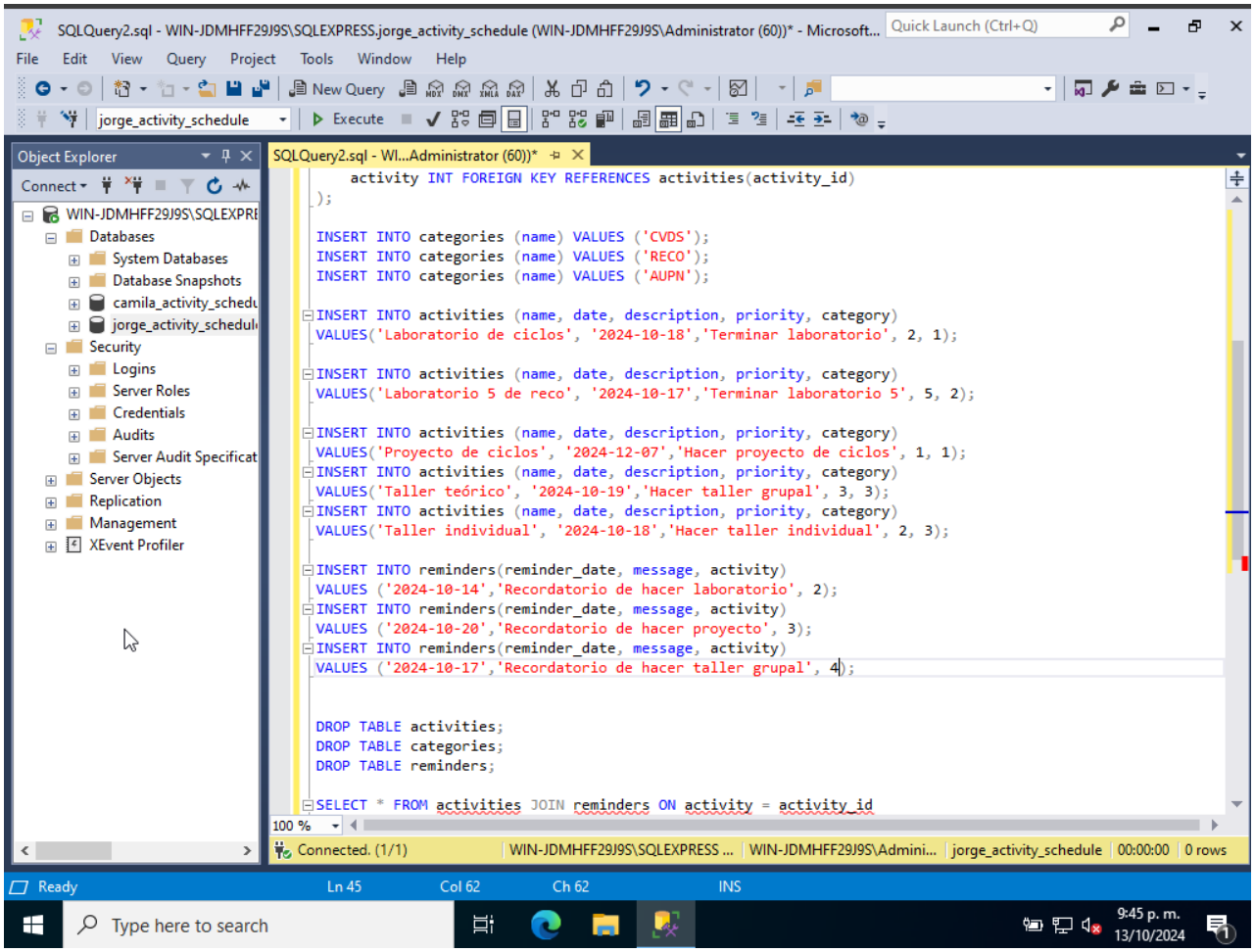


Figure 74. Inserting data into each table from jorge_activity_schedule

3. SQL Database - Microsoft Azure

a. Go to https://azure.microsoft.com/en-us/pricing/purchase-options/azure-account/ and log in with your institutional email.

b. Once logged into the **Azure** portal, explore the available services.

- What is cloud computing, and what are some of the advantages of using a platform like MicrosoftAzure compared to an on-premise infrastructure?

Cloud computing is a model where a company uses computing services provided by an external provider instead of maintaining the infrastructure in their own facilities. This includes servers, storage, databases, networks, software, and more.

The advantages of using a platform like Microsoft Azure compared to an on-premise infrastructure include:

- **Cost**: In a cloud environment, you only pay for the resources you use, eliminating the costs of hardware maintenance and upgrades.
- **Scalability**: The cloud allows easy adjustment of resources according to the current needs of the company, facilitating growth or downsizing.
- **Rapid Deployment**: The provisioning of new resources is almost instant, allowing the integration and availability of new applications quickly.
- **Global Accessibility**: It facilitates connecting with customers and partners globally without the need for physical infrastructure in each location.
- **Data Backup and Recovery**: Data can be regularly backed up and easily recovered in case of failures.
- **Lower Capital Expenditure**: No significant initial investment in hardware is required, as everything is managed through subscriptions and pay-as-you-go models.

- What types of services do Microsoft Azure offer (IaaS, PaaS, SaaS), and how do they differ fromone another?

There are three types of cloud computing services: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Each of these services provides different levels of control, flexibility, and management to meet various customer needs.

1. Infrastructure as a Service (IaaS):

- **Description:** IaaS provides users with virtual computing resources such as virtual machines, storage, and networking.
- **Advantages:** Avoids the expense and complexity of buying and managing physical servers and other data center infrastructure.
- **Ideal For:** Organizations that need to run legacy applications or require greater control over their infrastructure.
- **Examples in Azure:** Virtual Machines, Virtual Networks, and some Storage Accounts features.

2. Platform as a Service (PaaS):

- **Description:** PaaS allows users to avoid the expense and complexity of buying and managing software licenses, application infrastructure, middleware, container orchestrators, development tools, and other resources.
- **Advantages:** Users control the applications and services they develop, while the cloud service provider manages everything else.
- **Ideal For:** Organizations that want to quickly develop and deploy new applications without

worrying about the underlying infrastructure.

- **Examples in Azure:** App Service, Azure Functions, Azure Kubernetes Service.

3. Software as a Service (SaaS):

- **Description:** SaaS provides access to software applications hosted and maintained by Microsoft.
- **Advantages:** Users no longer need to manage the underlying infrastructure, develop, or deploy applications, as Microsoft handles it all.
- **Ideal For:** Organizations that want to use software applications without managing them.
- **Examples in Azure:** Office 365, Dynamics 365, Power BI.

- What is the importance of regions and availability zones in Azure, and how do they affect serviceavailability?

Regions and Availability Zones:

- **Regions**: These are specific geographic areas where Microsoft Azure operates its data centers. Each region has distinct capabilities and services.
- **Availability Zones**: These are physically separate locations within a region that offer high availability and resilience for your applications. By using these zones, you can protect your applications from regional failures.

Benefits of Regions and Availability Zones:

- **High Availability and Resilience**: Deploying resources across multiple availability zones within a region enhances application availability and minimizes the risk of interruptions.
- **Disaster Recovery**: Paired regions allow data and services to be replicated to a secondary region, improving recovery from catastrophic failures.
- **Capacity and Growth**: Recommended regions typically have higher capacity and are ideal for most workloads, offering lower costs and early access to new features.
- **Compliance and Data Residency**: Some organizations have legal requirements to store data within a specific region to comply with data sovereignty regulations.
- **Proximity and Latency**: Choosing a region close to your users or on-premises systems can reduce latency and improve application performance.
- **Cost Optimization**: Deploying resources in different regions can help optimize costs, especially in development and testing environments.

- What is the difference between vertical scaling and horizontal scaling in Azure, and when would you choose one over the other?

Vertical scaling involves increasing the power of a single system (adding more CPU or memory to a VM) to handle increased workloads. It's simpler and less complex to implement, but it involves downtime during upgrades and poses a single point of failure risk. It has lower initial costs but becomes less cost-effective over time due to upgrade limitations.

Horizontal scaling involves adding more nodes to a system (adding more VMs) to handle

increased workloads. It offers better performance, resilience, and no downtime during scaling but is more complex to implement and maintain, with higher initial costs. It requires load balance and efficient workload distribution across multiple nodes, making it ideal for large-scale applications where distributed processing is essential.

Choosing Between Vertical and Horizontal Scaling:

- Choose **vertical scaling** when simplicity, lower initial costs, and ease of maintenance are priorities, and when the workload can be handled by a single machine's capacity.
- Choose **horizontal scaling** for better performance, resilience, and fault tolerance, especially for applications needing distributed processing and when downtime is not acceptable. Horizontal scaling is preferred for large-scale applications, like web servers, databases, and cloud-based applications, where workloads can be efficiently distributed across multiple nodes.

- How does using technologies like TLS (Transport Layer Security) on the transport layer affect accessing Azure SQL Database compared to a local virtual machine database?

TLS (Transport Layer Security) is a protocol that provides security and privacy in communications over a network. Below is how it affects access to an Azure SQL Database compared to a database on a local virtual machine:

1. **Data Encryption and Maintenance**:
    - **Azure SQL Database**: TLS ensures that all data transmitted between the client and the server is encrypted, protecting against eavesdropping and man-in-the-middle attacks. Azure SQL Database requires all connections to use TLS and manages all maintenance and updates needed to ensure that connections are always secure.
    - **Local Virtual Machine**: TLS can be configured to encrypt communications for a local database, but its implementation and configuration depend on the system administrator. Additionally, the administrator must keep TLS components up to date and manually configure security policies, which can be complex and prone to errors if not done properly

- From a transport layer perspective, how does the handling of TCP connections differ between aSQL database hosted on a local virtual machine and Azure SQL Database?

When comparing TCP connection handling between a SQL database on a local virtual machine and Azure SQL Database, the differences can be summarized as follows:

1. **Connection Management**: Azure SQL Database employs advanced management features such as load balancing and automatic recovery, ensuring efficient distribution of connections and minimizing downtime during failures. In contrast, local virtual machines rely on manual configurations, which can lead to longer downtimes if connections fail

2. **Scalability**: Azure SQL Database can dynamically scale its connections based on demand, allowing it to handle traffic spikes seamlessly. Local virtual machines are limited by the physical hardware resources, requiring manual intervention for scaling, which may lead to performance bottlenecks during high demand

3. **Security and Encryption**: Azure SQL Database enforces encrypted connections using TLS by default, providing robust security against eavesdropping. On the other hand, local databases also support TLS, but it must be configured manually, increasing the risk of misconfiguration and security vulnerabilities

c. Use the Azure SQL Database service to manage records of books and scientific articles. The database must have at least 3 tables.

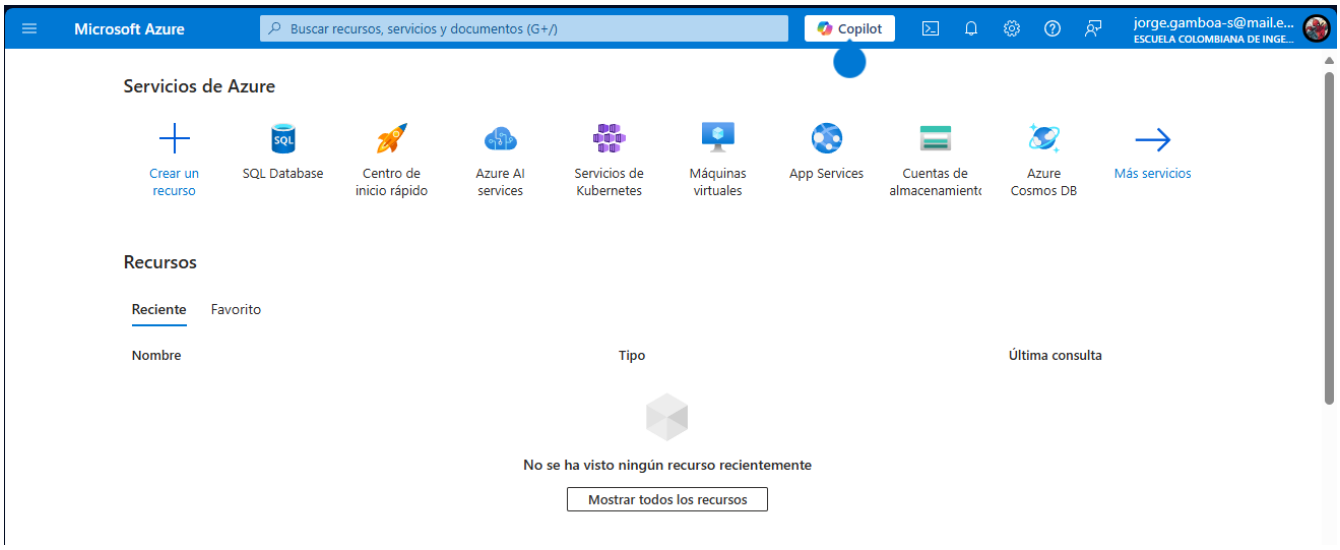


Figure 75 searching data base

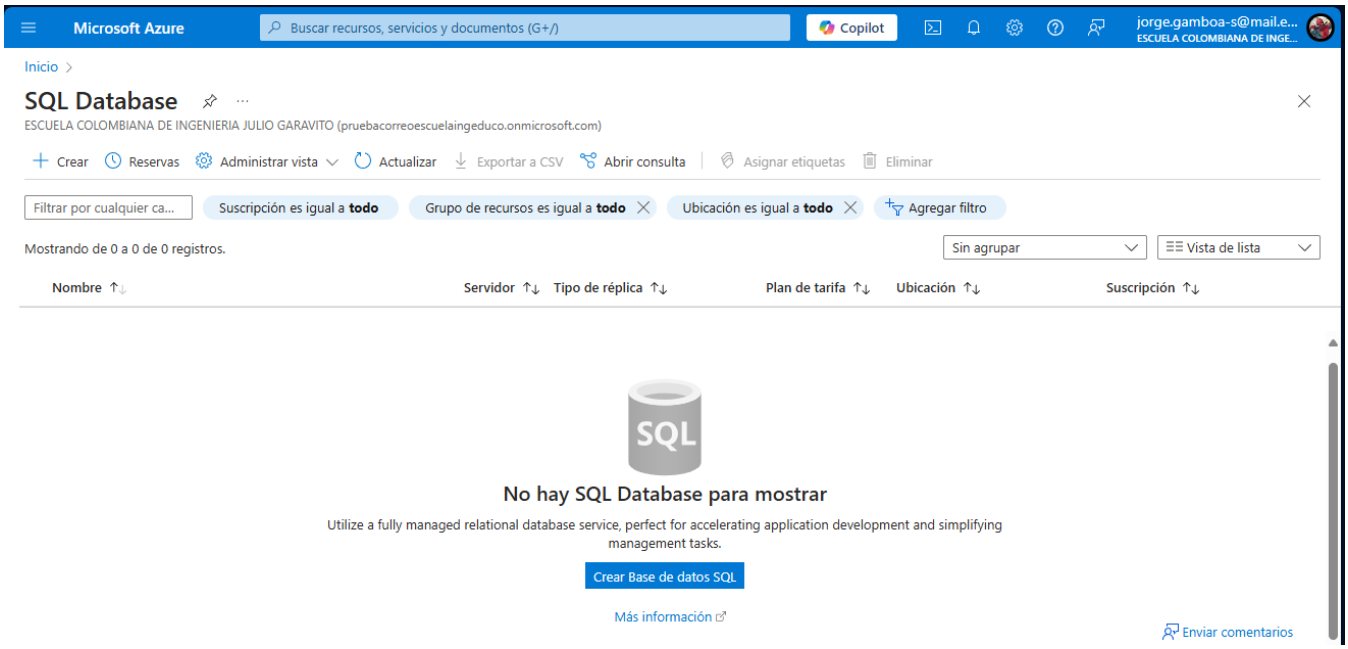


Figure 76 finding SQL DataBase

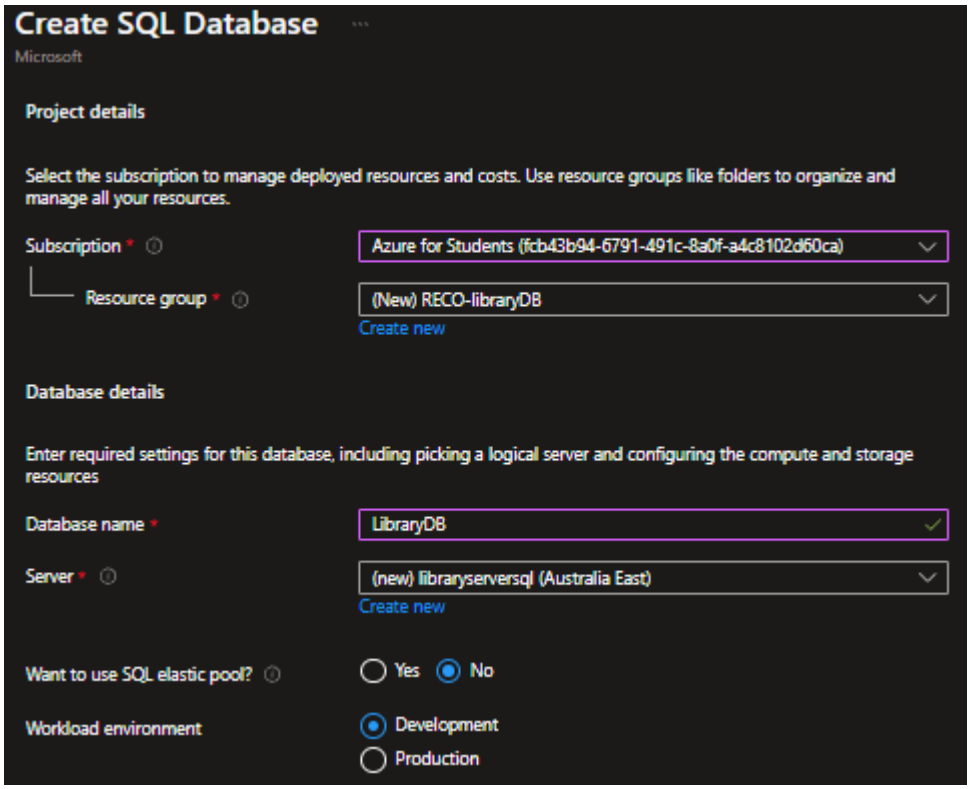


Figure 77 configurating the group and server

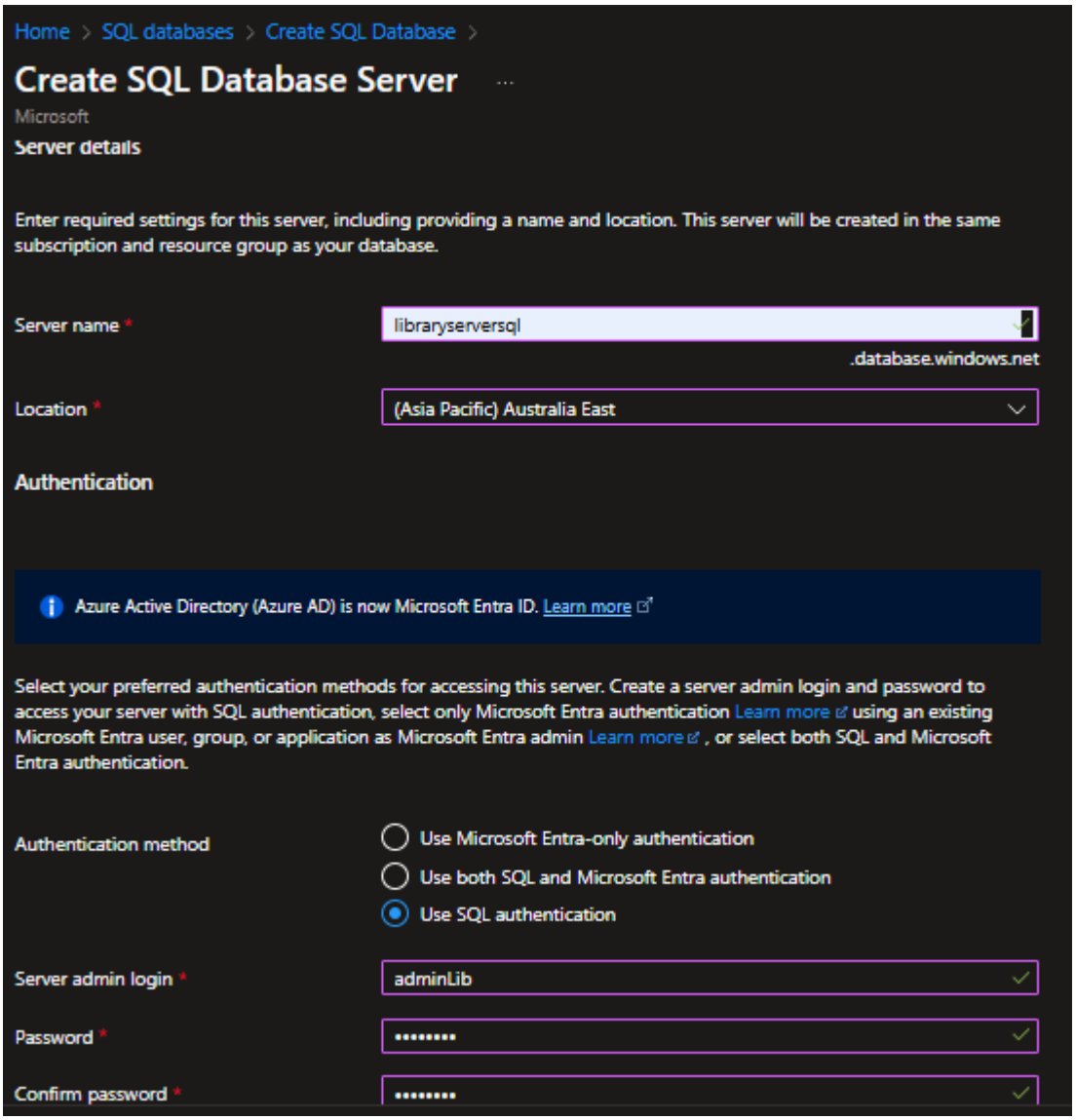


Figure 78 creating a new server

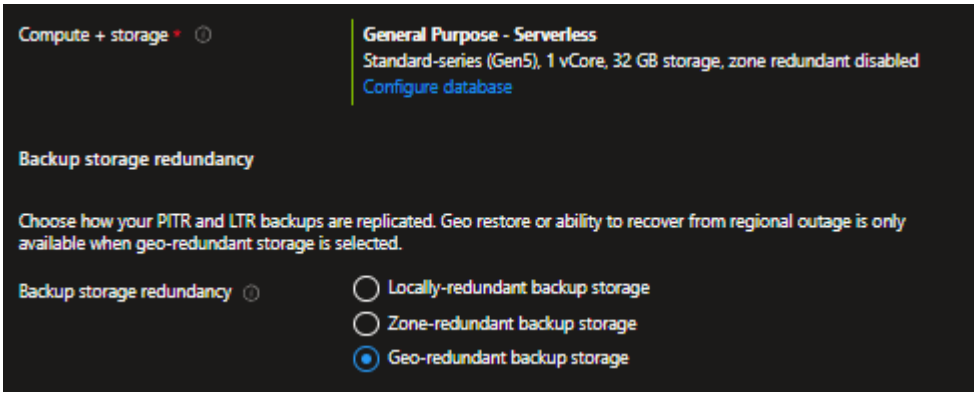


Figure 79 setting the Backup storage

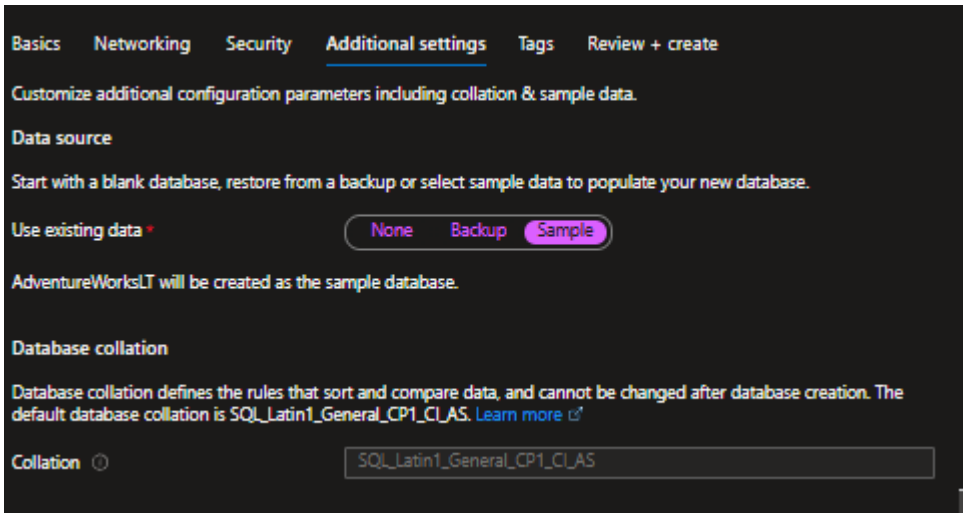


Figure 80 configuration of additional settings

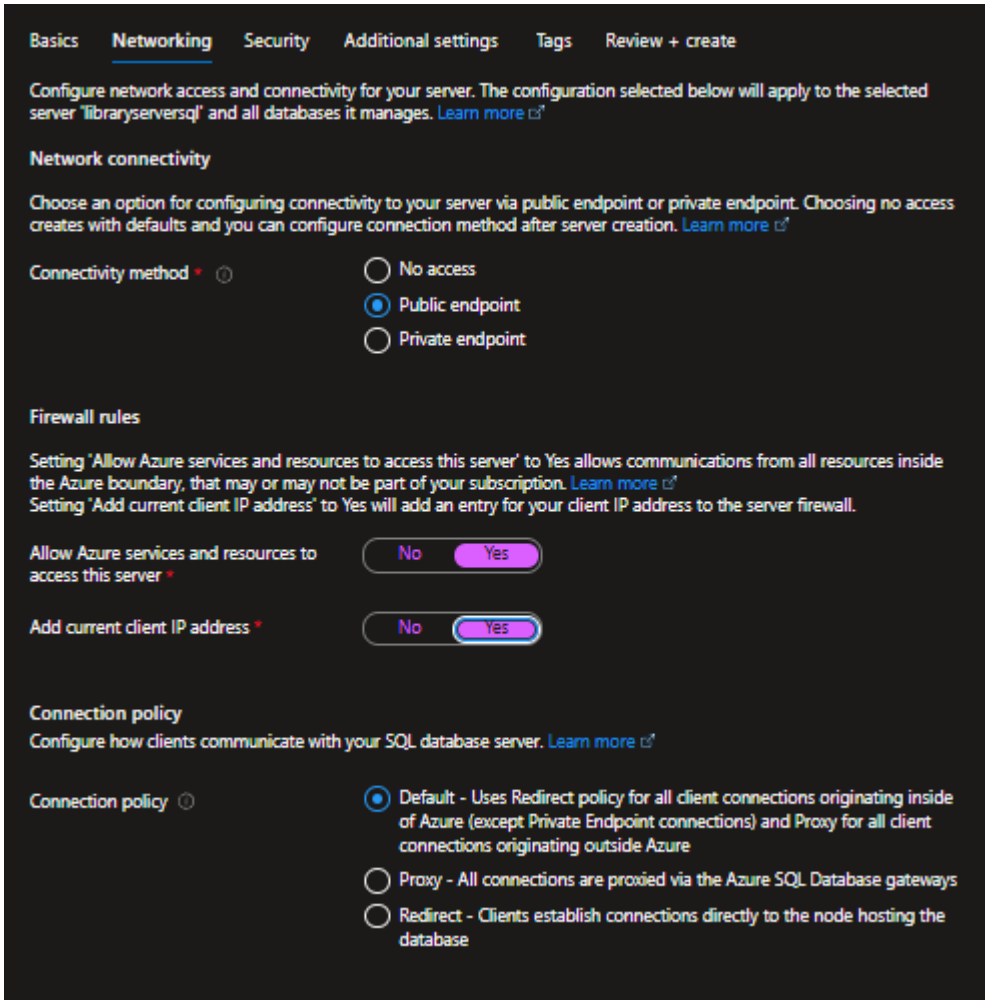


Figure 81 configuration of networking

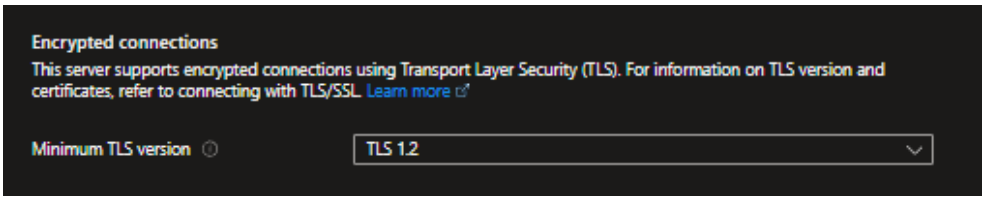


Figure 82 additional configuration

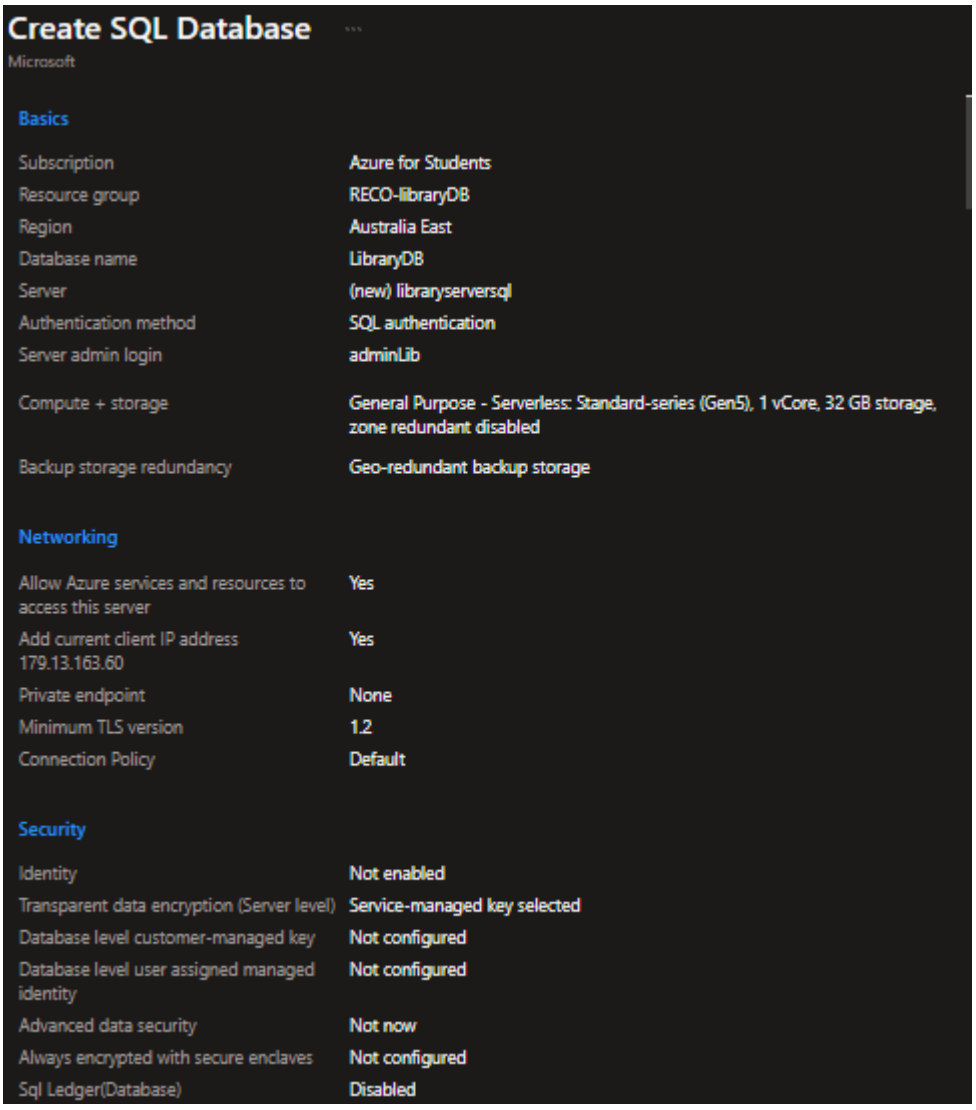


Figure 83 general information

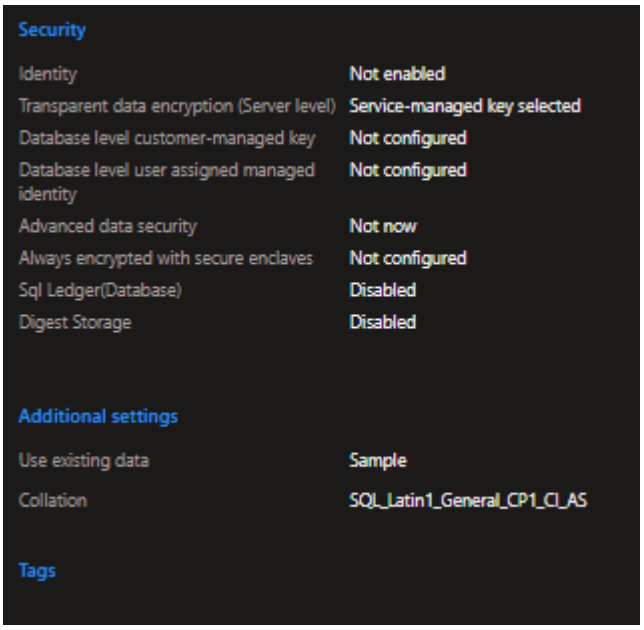


Figure 84 general information two

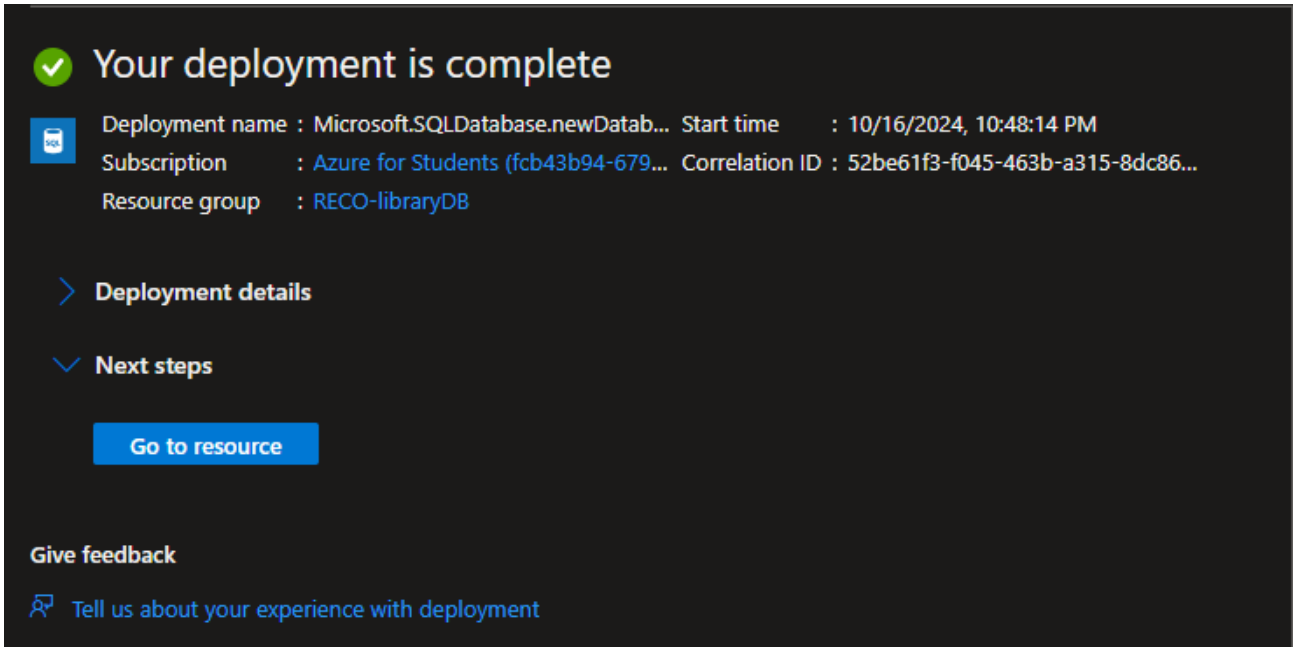


Figure 85 creation process completed

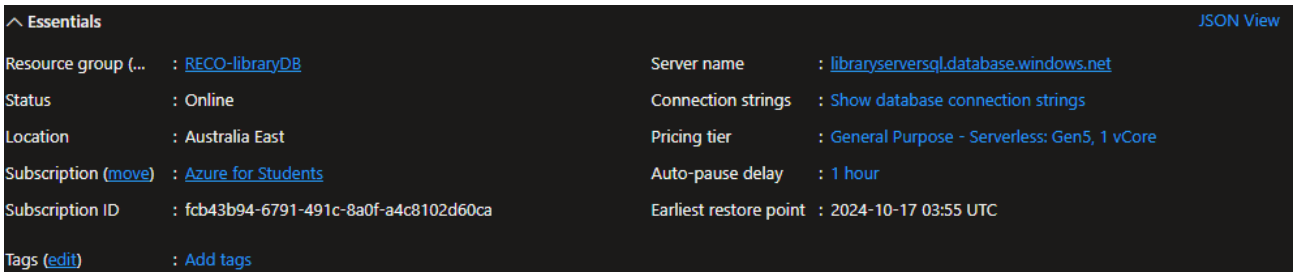


Figure 86 general information about the dataBase

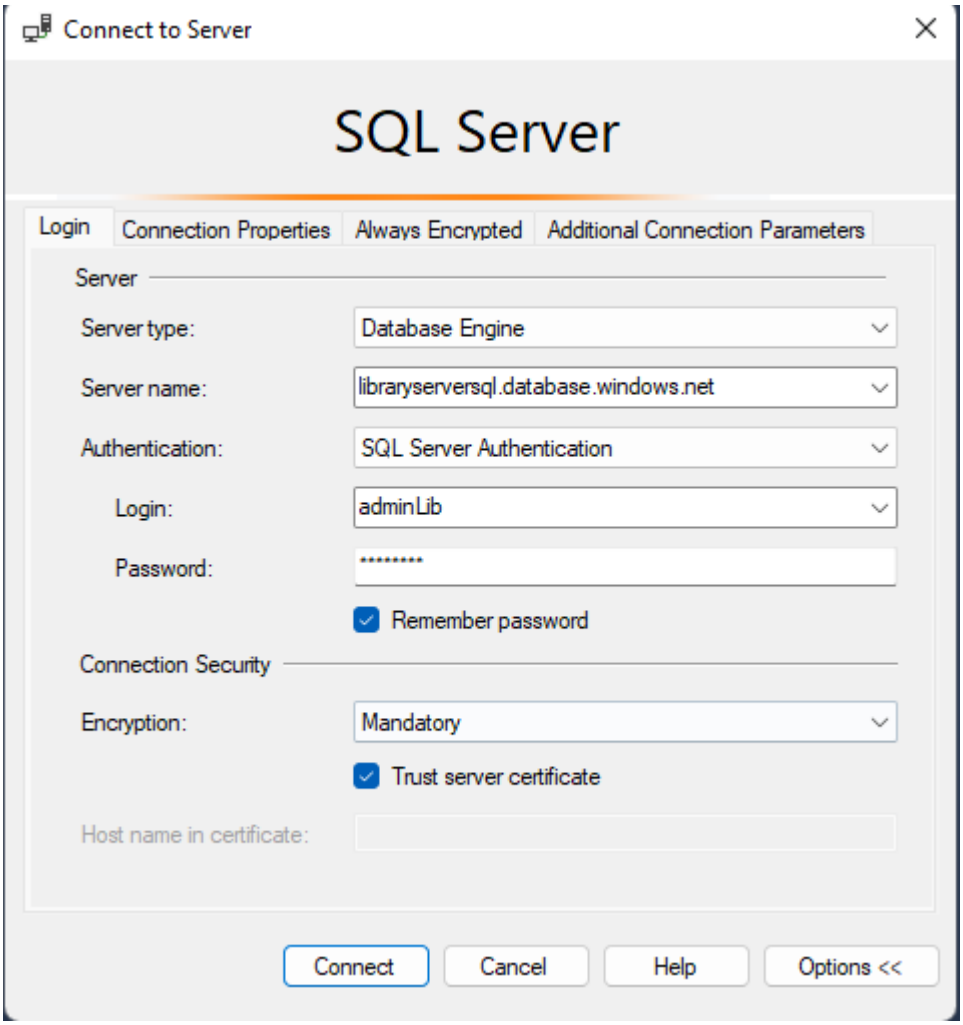


Figure 87 configuration in SQL Server

Now, in the attached video, we will demonstrate the process of recording and creating the database, along with inserting some data. https://youtu.be/gLiavXA9oFg.

Results Messages

	AuthorID	Name
1	1	John Doe
2	2	Jane Smith
3	3	Alice Johnson

	BookID	Title	AuthorID	PublishedDate
1	1	Understanding SQL Server	1	2023-01-15
2	2	Advanced Database Systems	2	2022-05-10
3	3	Data Science for Dummies	1	2023-07-30

	ArticleID	Title	AuthorID	Journal	PublishedDate
1	1	The Future of Databases	2	Database Journal	2023-03-12
2	2	SQL Performance Tuning	1	Tech Review	2023-06-25
3	3	NoSQL Databases Expl...	3	Journal of Com...	2022-11-20

Figure 88 data SQL Azure

4. Other Database Engine Configurations

4.1. Configuration to automatically start the database

4.1.1. Linux Slackware

- We created a startup script to automatically launch the PostgreSQL server during system boot, saving us from having to use the long startup command (**/usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data**) each time. This script will be located at **/etc/rc.d/rc.postgresql**

```
  GNU nano 6.0                          /etc/rc.d/rc.postgresql
#!/bin/sh
#POSTGRESQL startup script for Slackware

case "$1" in
        start)
                echo "Starting PostgreSQL..."
                su - postgres -c "/usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data start"
                ;;
        stop)
                echo "Stopping PostgreSQL..."
                su - postgres -c "/usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data stop"
                ;;
        restart)
                echo "Restarting PostgreSQL..."
                su - postgres -c "/usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data restart"
                ;;
        *)
                echo "Usage: $0 {start|stop|restart}"
                exit 1
esac
exit 0

                                [ Read 21 lines ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut       ^T Execute   ^C Location  M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Paste     ^J Justify   ^_ Go To Line M-E Redo
```

Figure 7689. Script to launch PostgreSQL during system boot

- We add this command to **/etc/rc.d/rc.local** to run during system boot

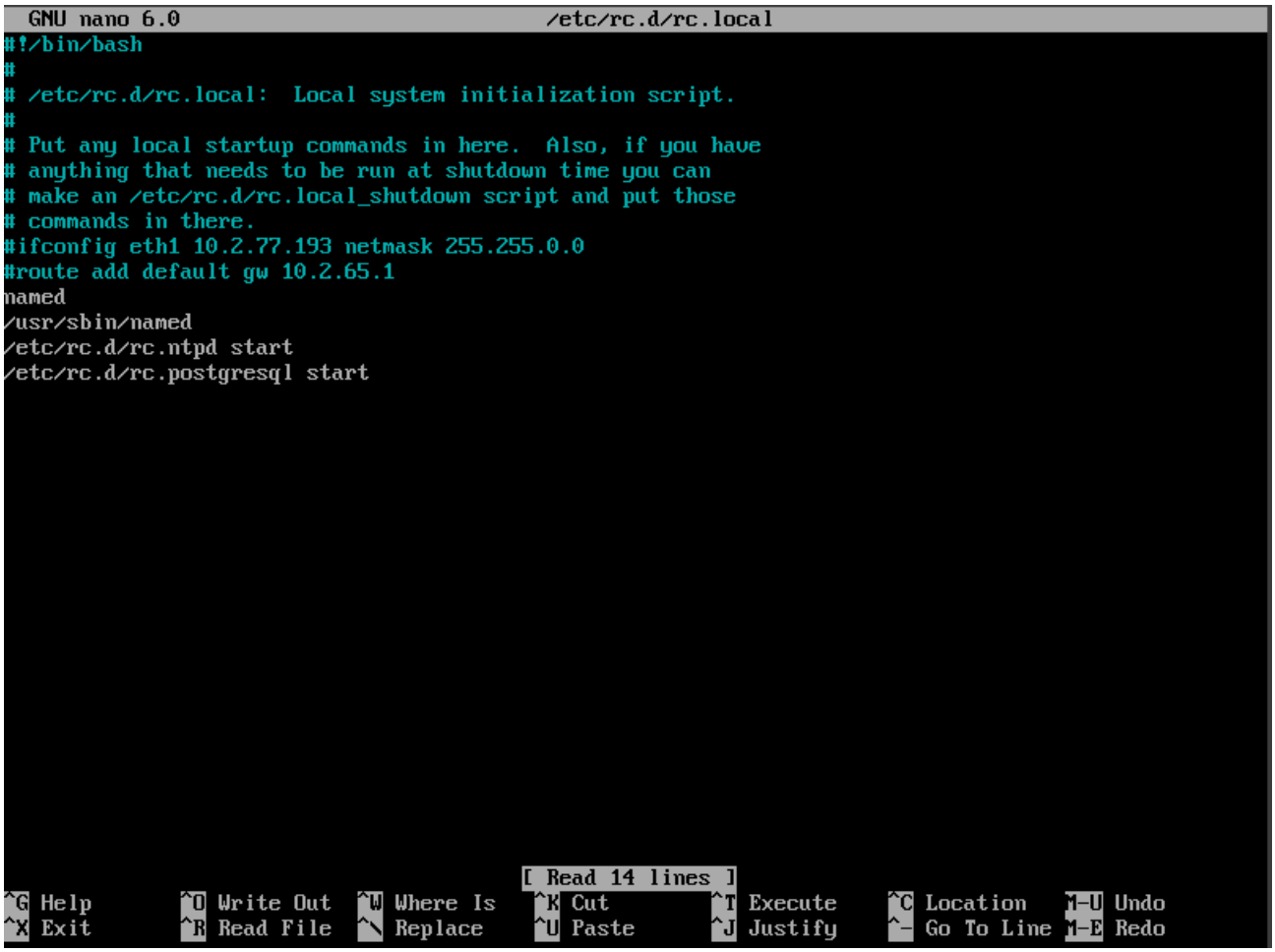


Figure 7790. Adding command into /etc/rc.d/rc.local

- We tested this and can see that the server started

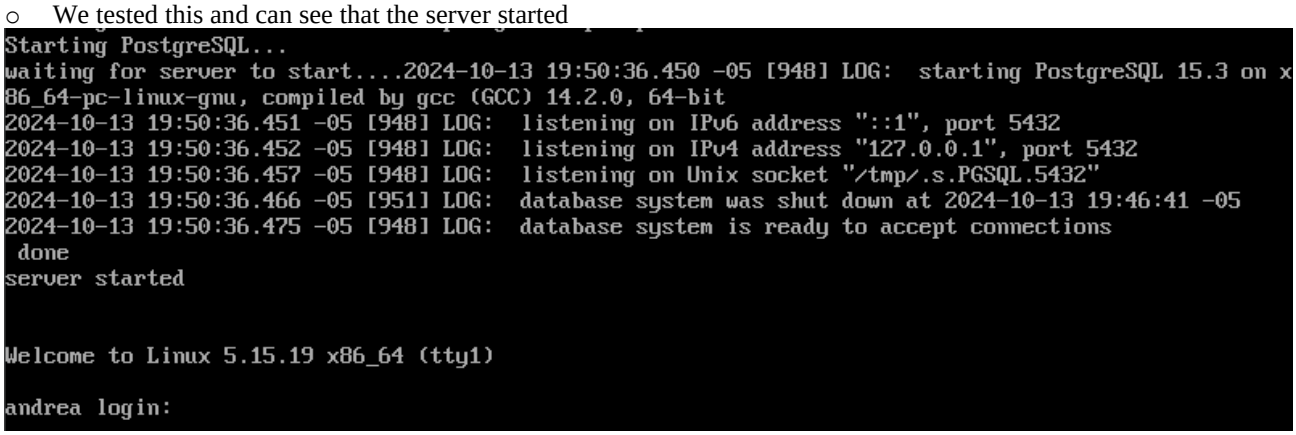


Figure 7891. Starting PostgreSQL during system boot

4.1.2. Windows Server

- We press CTRL+R and then type the command ‘services.msc’

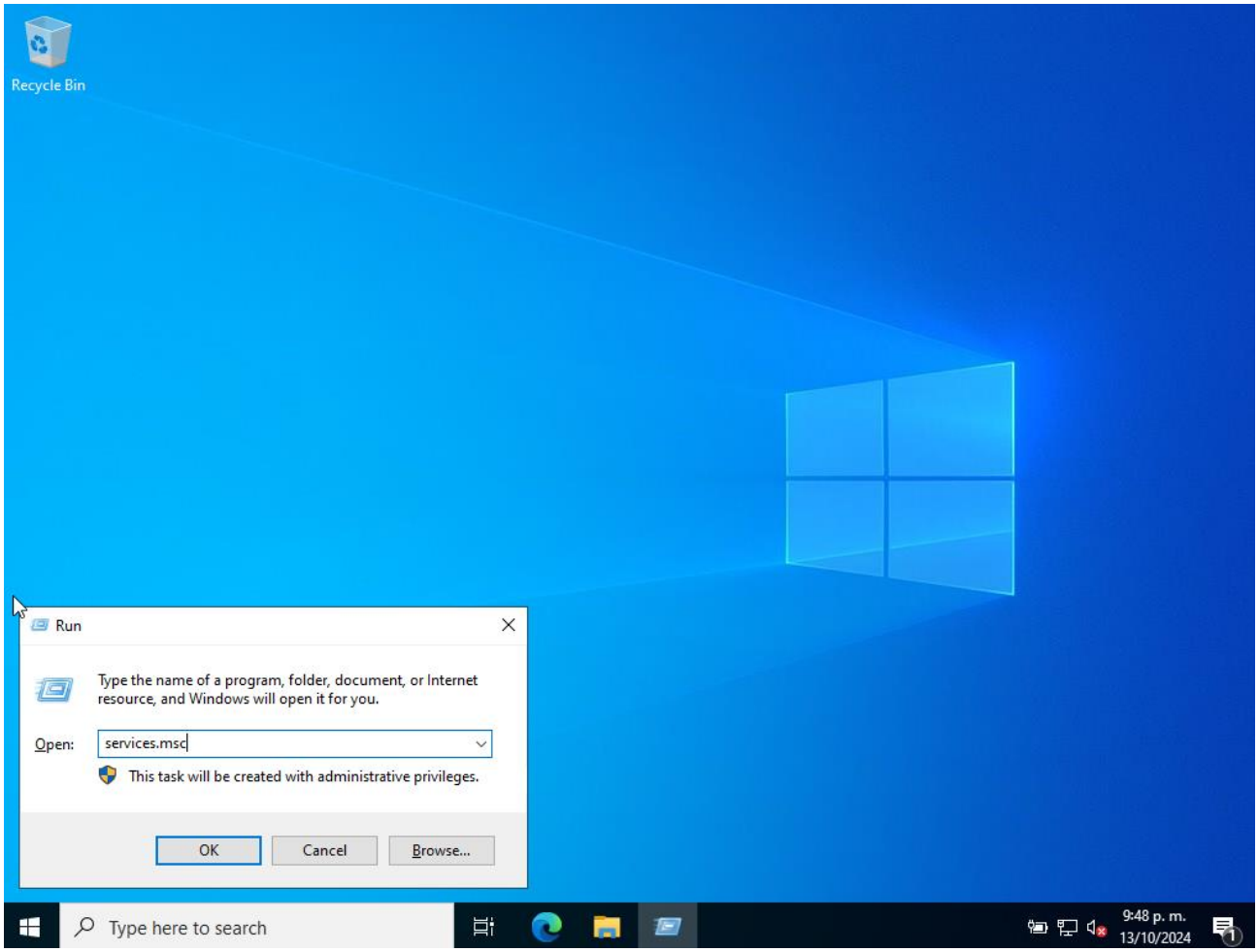


Figure 7992. Opening Microsoft services

- We search for SQL Server, then right click and select ‘Properties’

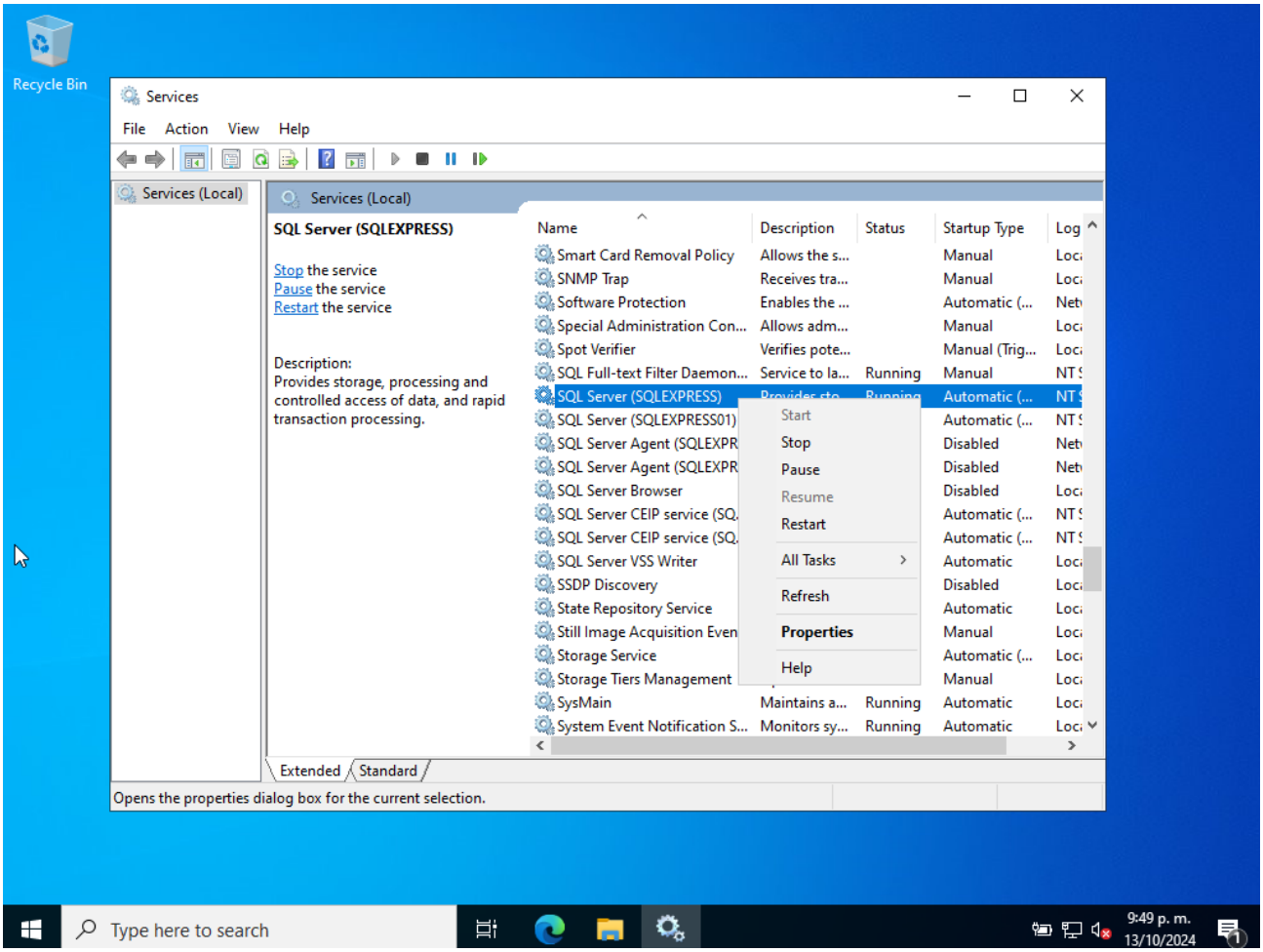


Figure 8093. Properties of SQL Server Service

- In 'Startup Type' we must select 'Automatic' to run the server during system boot

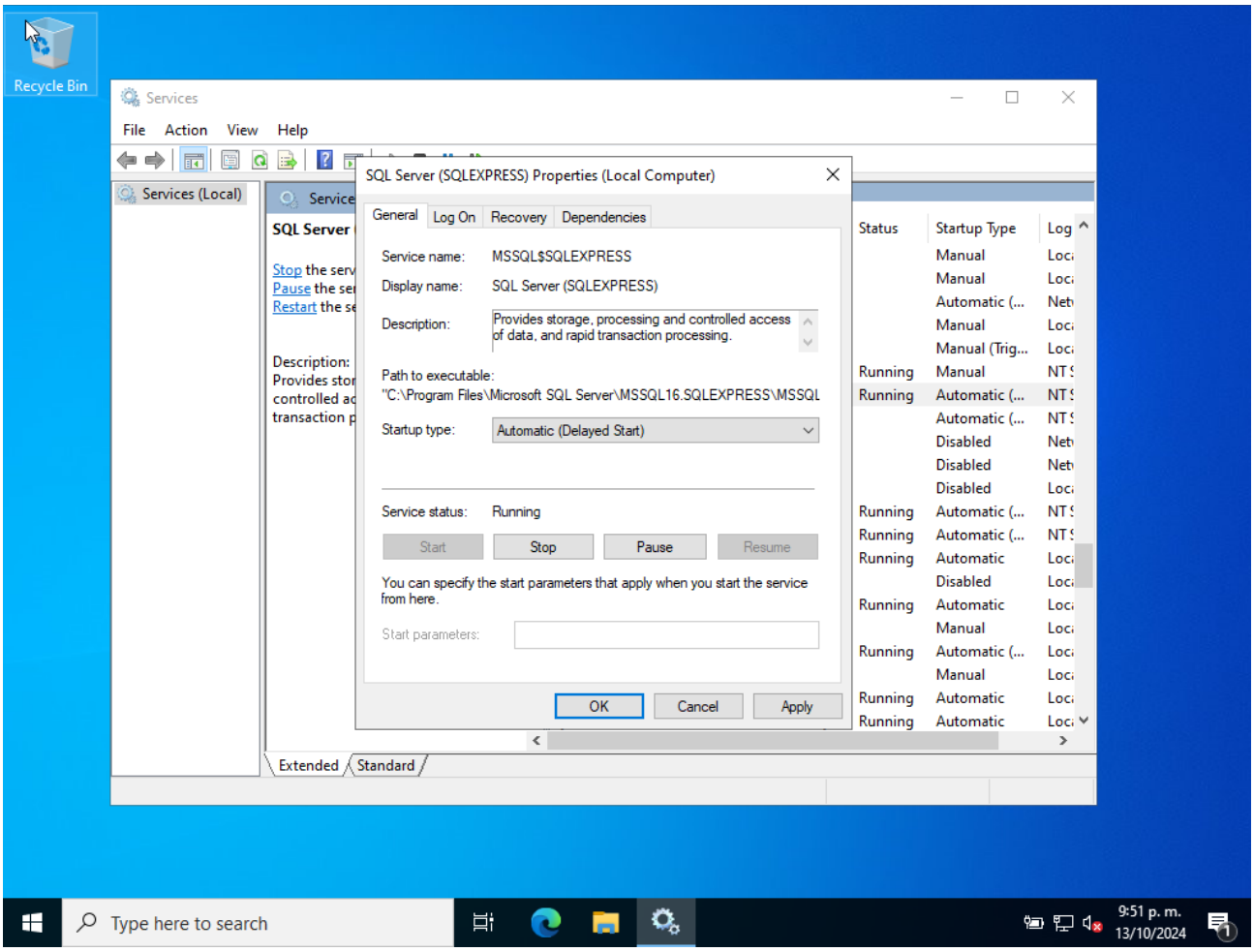


Figure 81 94. Configuring SQL Server to Start during system boot

4.2. Remote Database Connection with DBeaver

4.2.1. Linux Slackware

- We open the file **/usr/local/pgsql/data/postgresql.conf,** then search for the line 'listen_addresses' and uncomment it. Finally, we change the value of this variable to '***'.** This will allow it to listen to all network addresses.

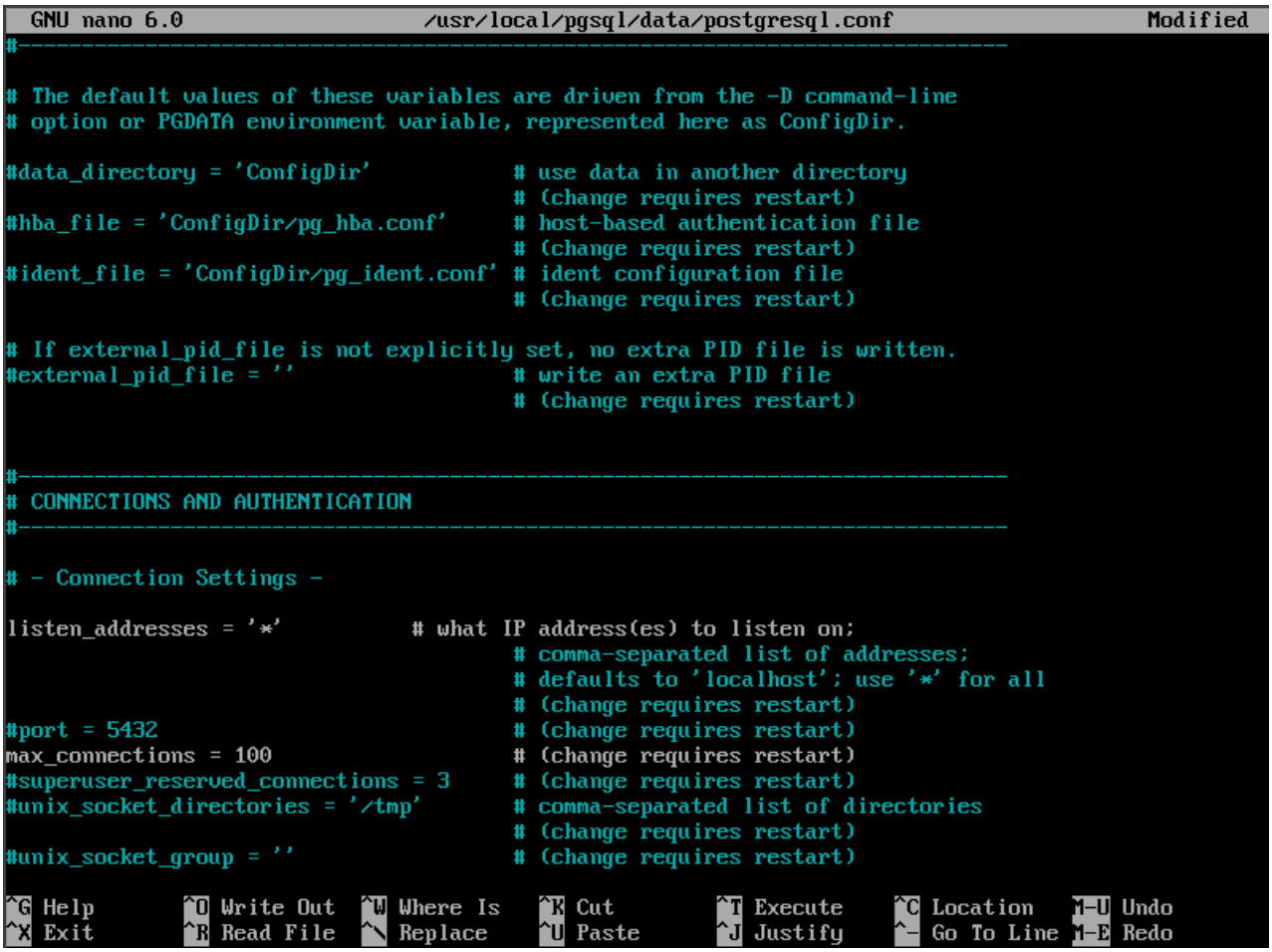


Figure 8295. Changing /usr/local/pgsql/data/postgresql.conf to allow listening on all network addresses

- We open the file **/usr/local/pgsql/data/pg_hba.conf** to add a new line: '**host all all 0.0.0.0/0 md5**'.
  This will allow connections from any IP address

```
  GNU nano 6.0                  /usr/local/pgsql/data/pg_hba.conf                  Modified
# This file is read on server startup and when the server receives a
# SIGHUP signal.  If you edit the file on a running system, you have to
# SIGHUP the server for the changes to take effect, run "pg_ctl reload",
# or execute "SELECT pg_reload_conf()".
#
# Put your actual configuration here
# ----------------------------------
#
# If you want to allow non-local connections, you need to add more
# "host" records.  In that case you will also need to make PostgreSQL
# listen on a non-local interface via the listen_addresses
# configuration parameter, or via the -i or -h command line switches.

# CAUTION: Configuring the system for local "trust" authentication
# allows any local user to connect as any PostgreSQL user, including
# the database superuser.  If you do not trust all your local users,
# use another authentication method.


# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     trust
# IPv4 local connections:
host    all             all             127.0.0.1/32            trust
# IPv6 local connections:
host    all             all             ::1/128                 trust
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication     all                                     trust
host    replication     all             127.0.0.1/32            trust
host    replication     all             ::1/128                 trust
host    all             all             0.0.0.0/0               md5

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location  M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line M-E Redo
```

Figure 8396. Allowing connections from any IP address

- We open DBeaver and select a server, in this case PostgreSQL

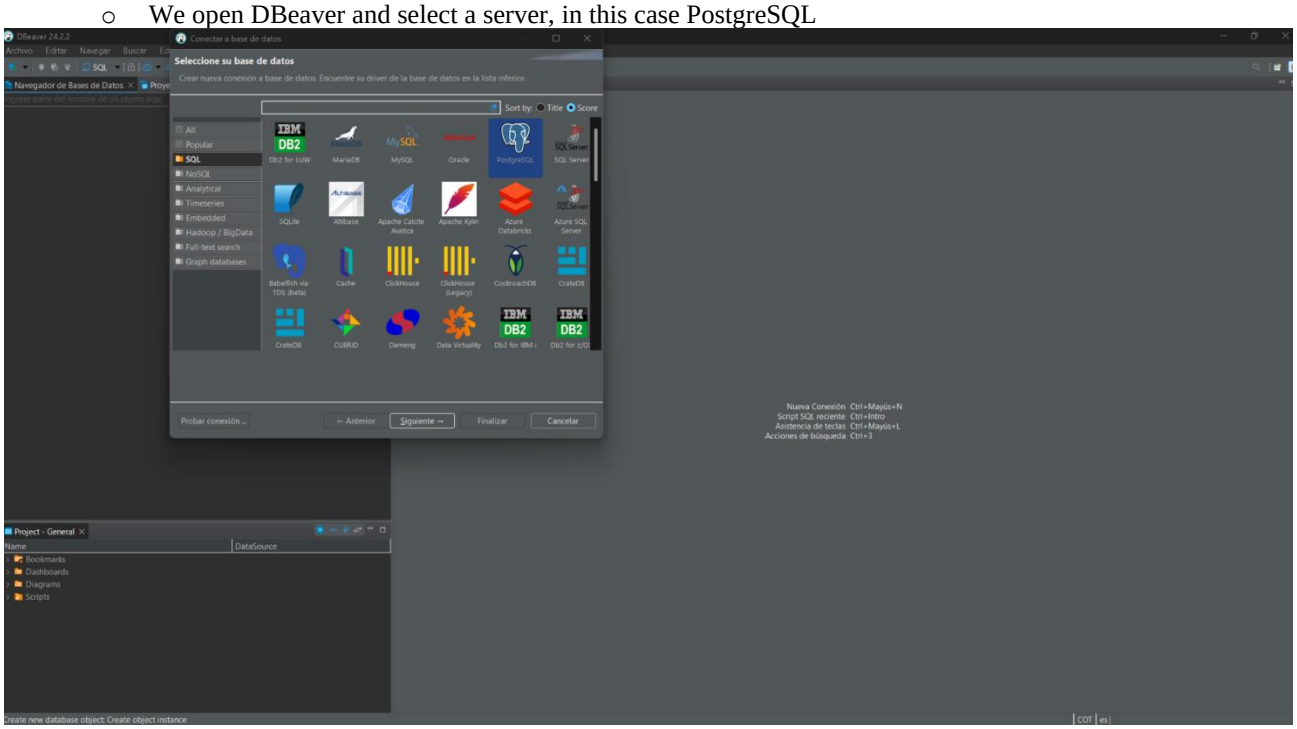


Figure 8497. Selecting PostgreSQL server in DBeaver

- We enter the IP address of Slackware, the database name (in this case camila_tourist_sites) and the username with the password

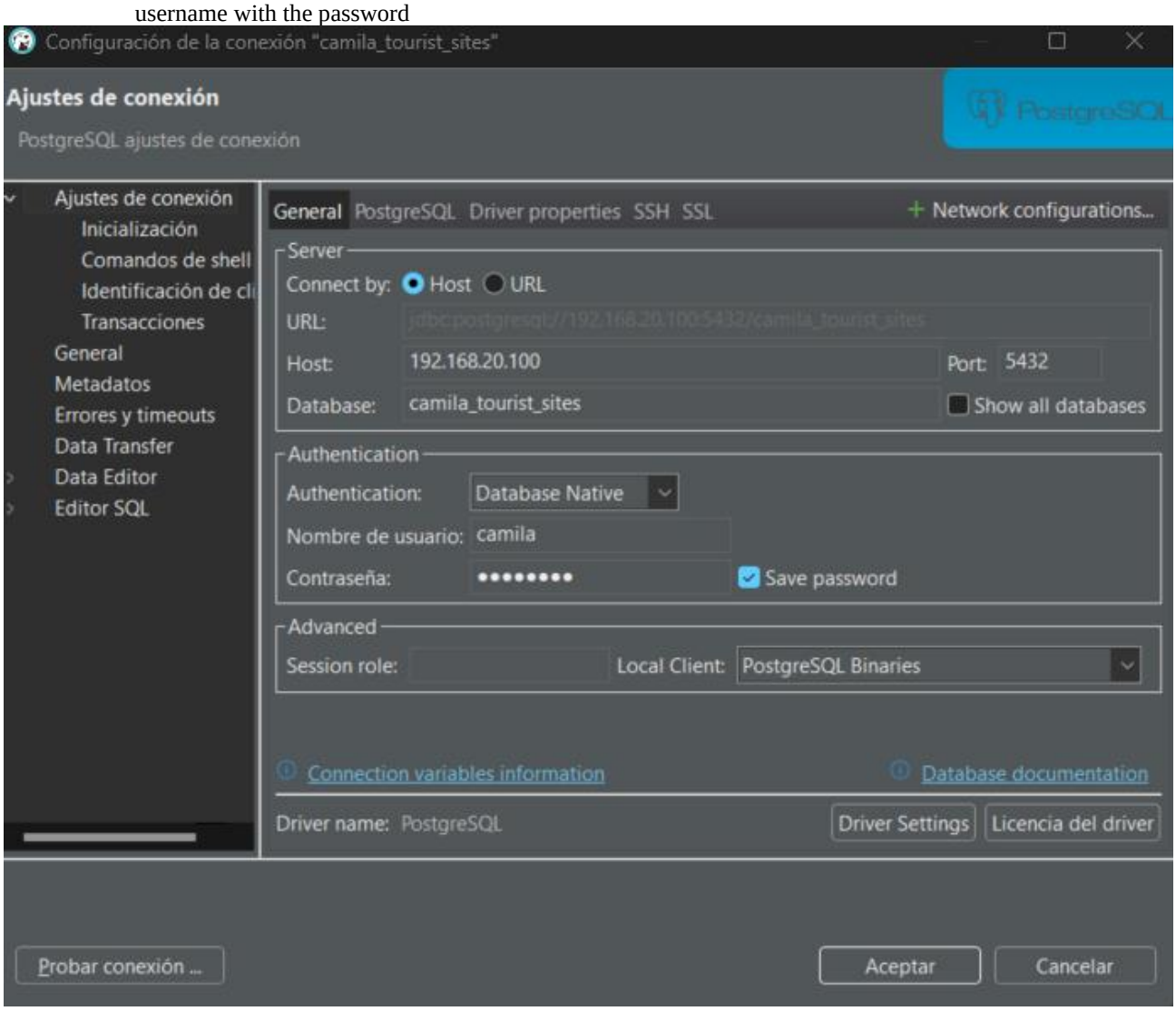


Figure 8598. Credentials to connect to camila_tourist_sites in DBeaver

- We establish the remote connection

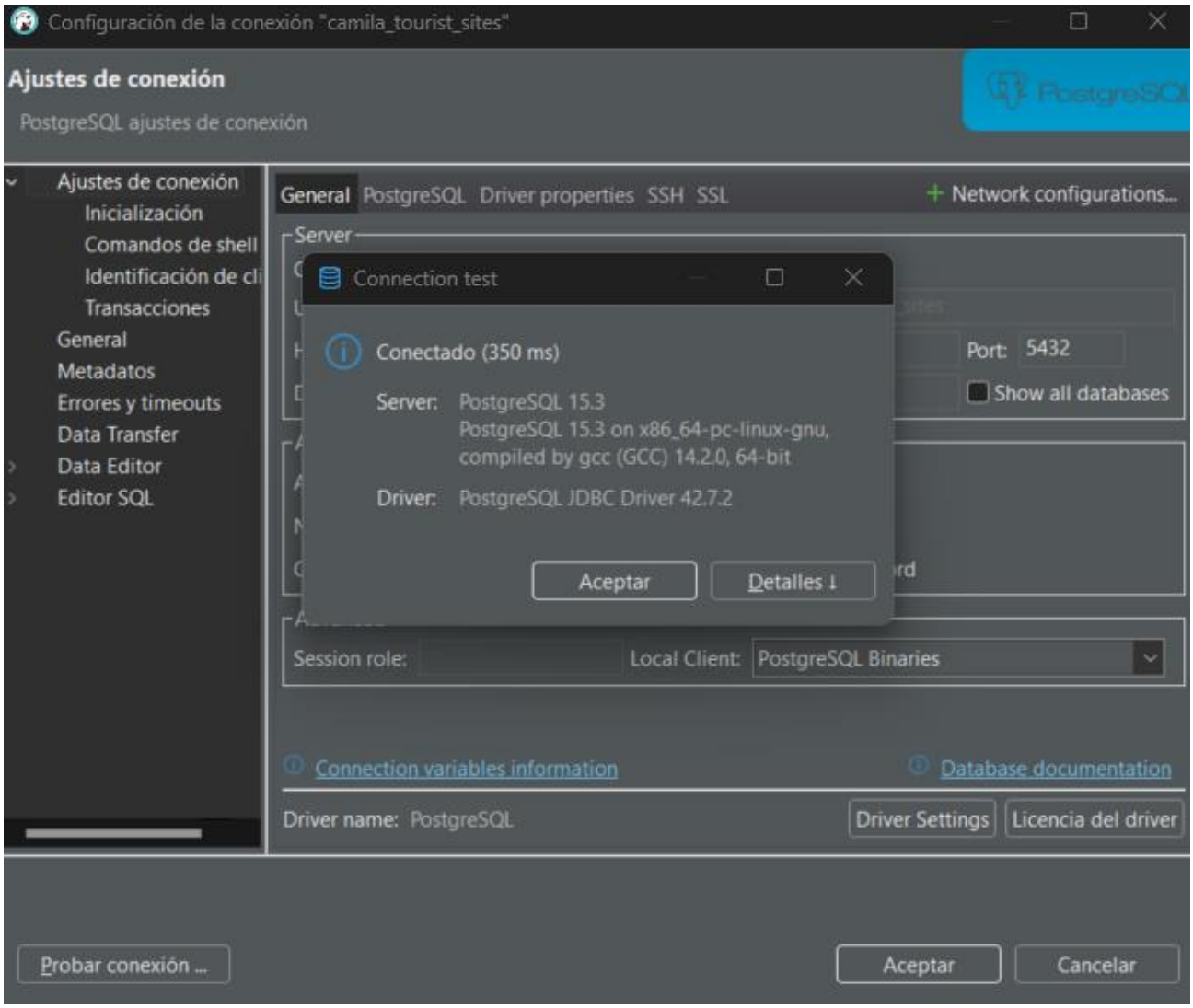


Figure 8699. Remote connection to camila_tourist_sites in DBeaver

- We can see the tables and the data that we inserted earlier

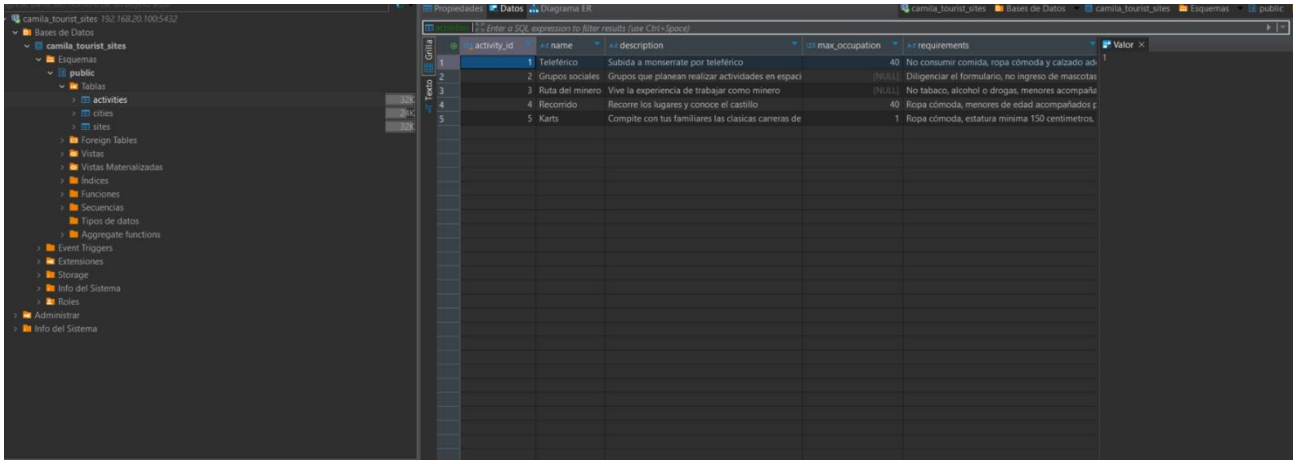


Figure 87100. Activities table of database camila_tourist_sites in DBeaver

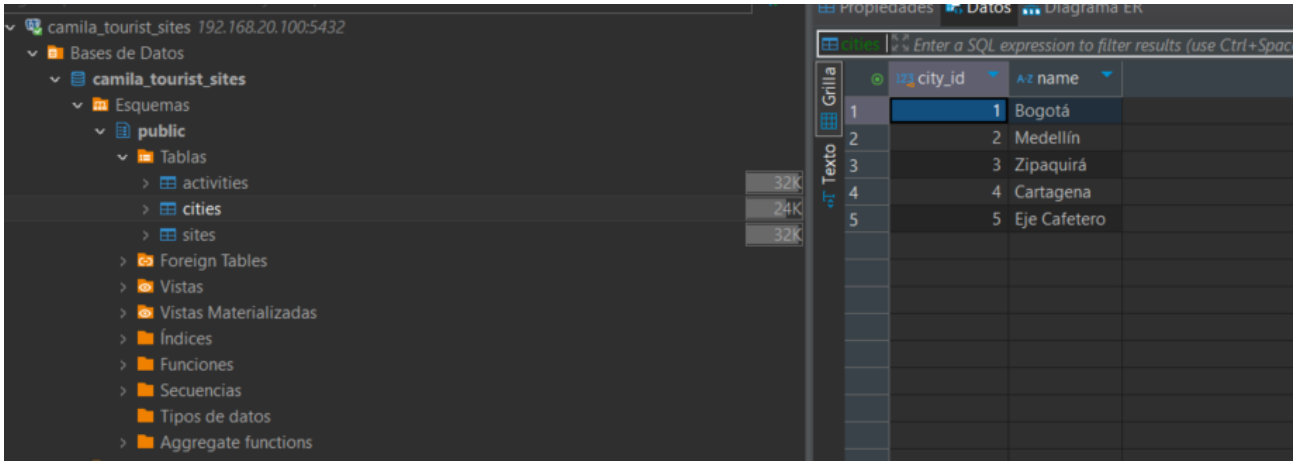


Figure 88101. Cities table of database camila_tourist_sites in DBeaver

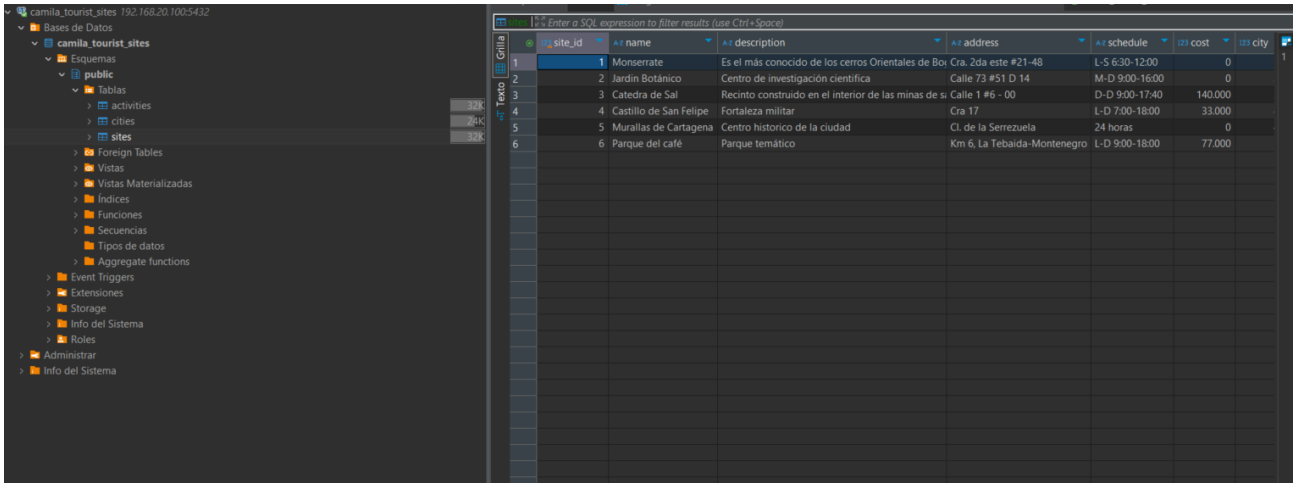


Figure 89102. Sites table of database camila_tourist_sites in DBeaver

- To connect to jorge_tourist_sites, we follow the same steps as before, but with the user Jorge and the respective password

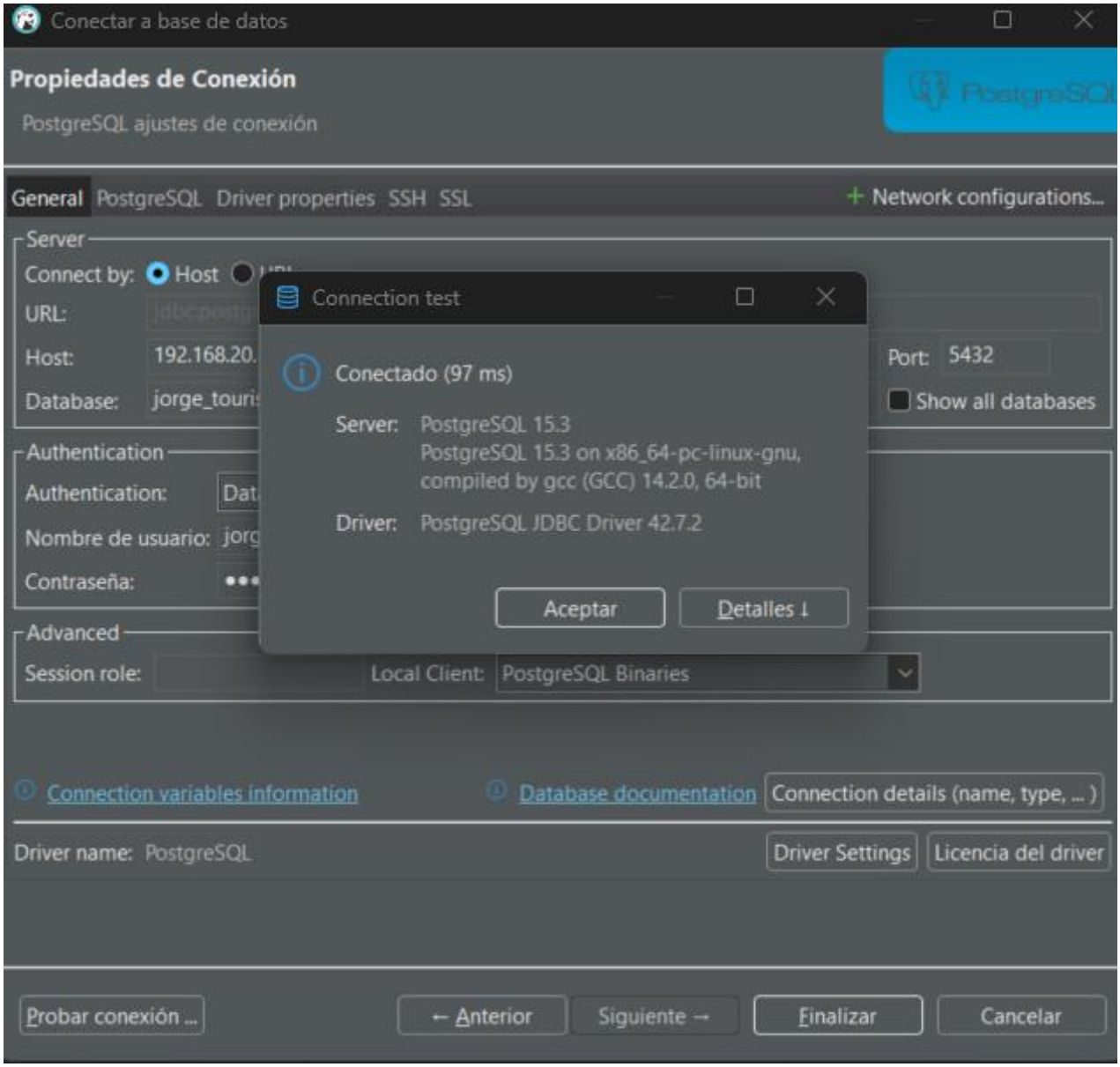


Figure 90103. Remote connection to jorge_tourist_sites in DBeaver

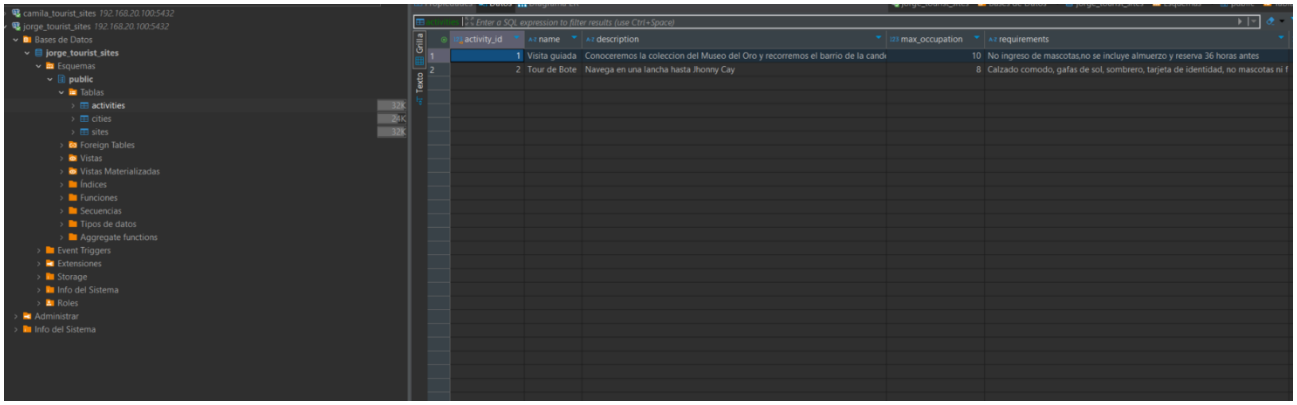


Figure 91104. Activities table of database jorge_tourist_sites in DBeaver

Figure 92105. Cities table of database jorge_tourist_sites in DBeaver

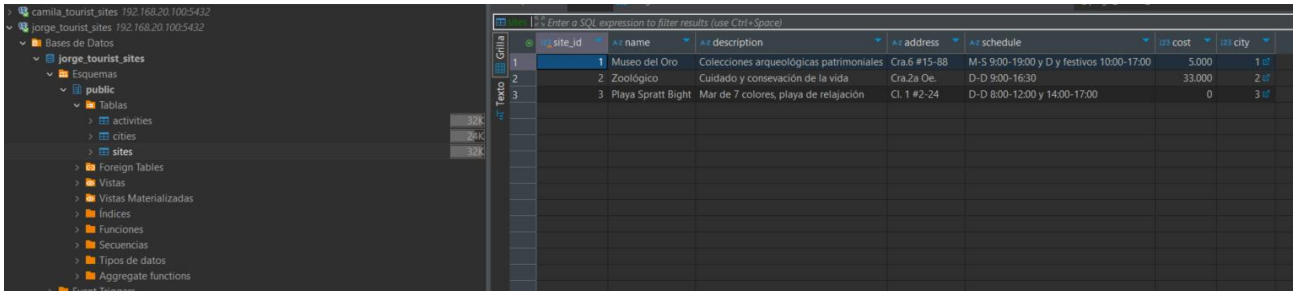


Figure 93106. Sites table of database jorge_tourist_sites in DBeaver

4.2.2. Windows Server

- In SSMS we open the server properties

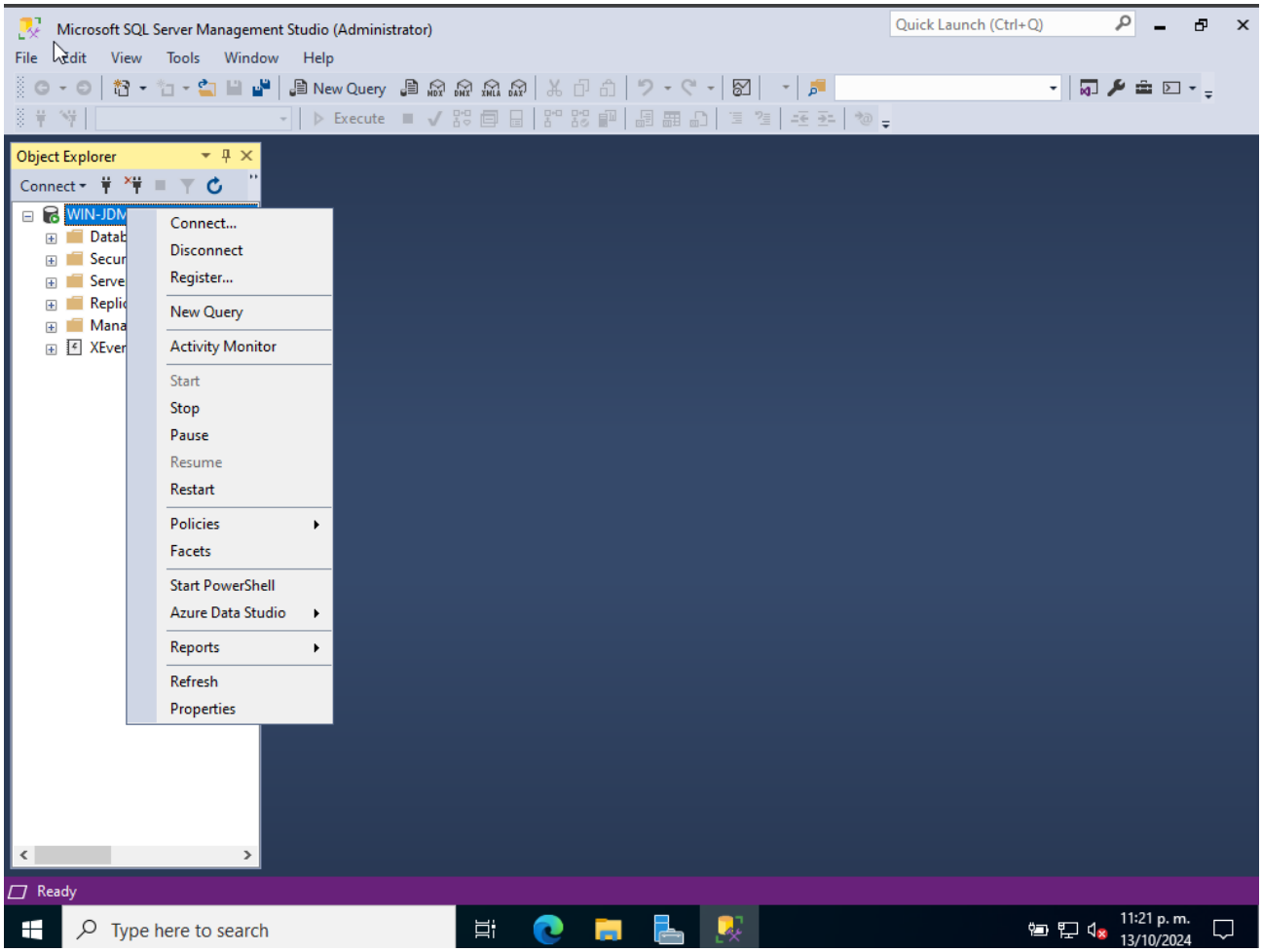


Figure 94107. Opening server properties

- We navigate to 'Connections' and select 'Allow remote connections to this server'

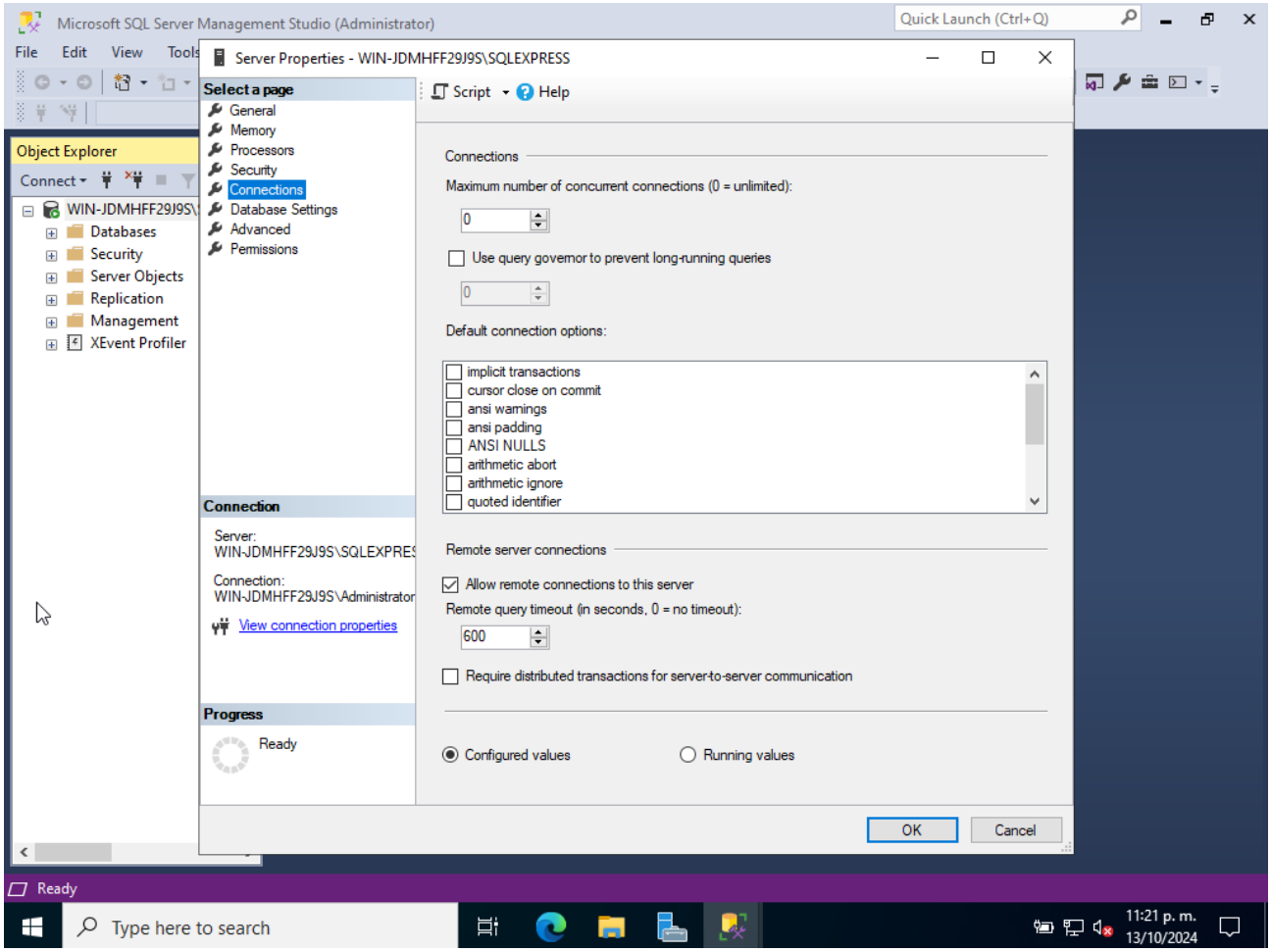


Figure 95108. Allowing remote connections in SSMS

- Then navigate to ‘Security’ and select ‘SQL Server and Windows authentication mode’

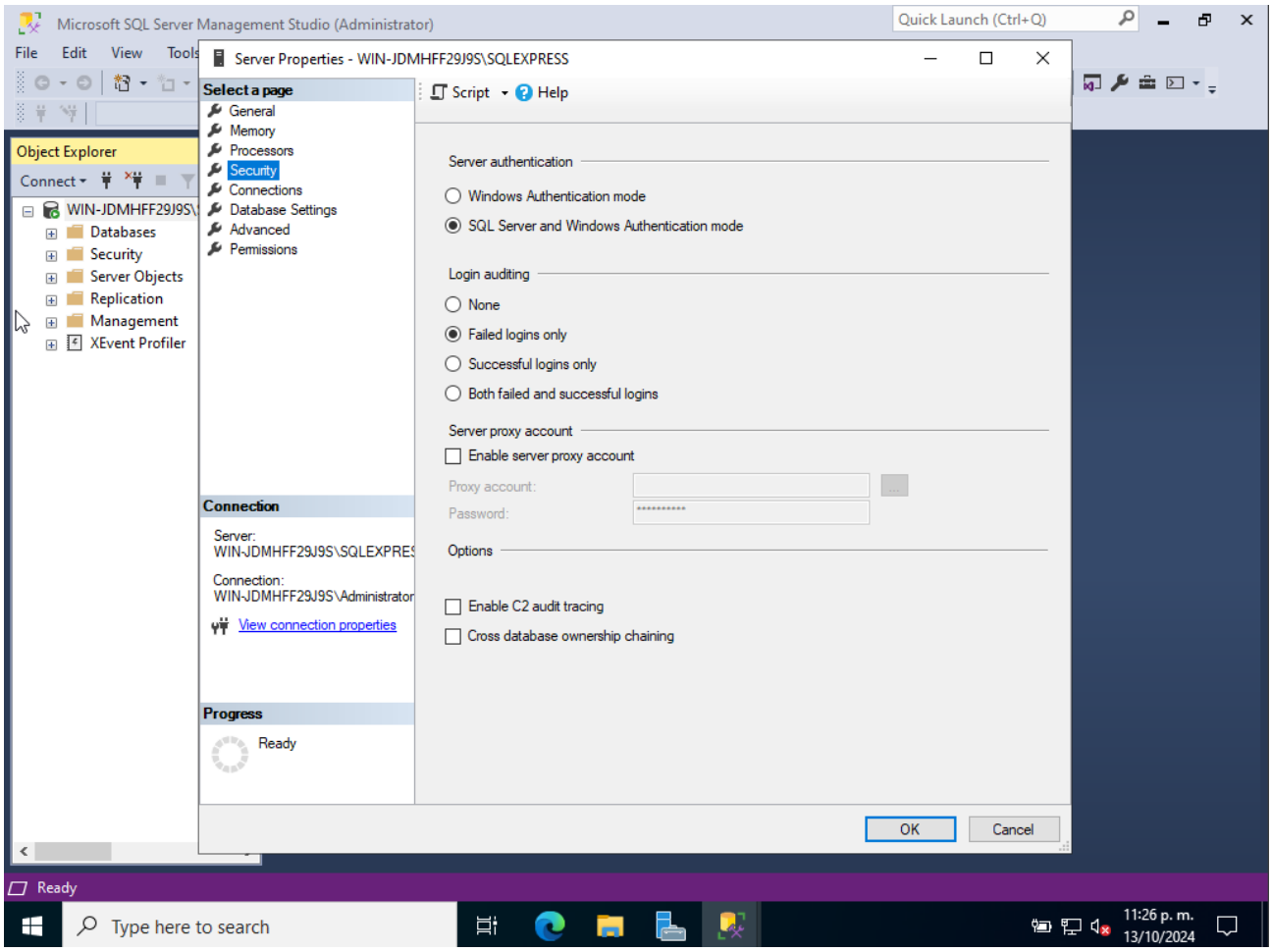


Figure 96109. Selecting SQL Server and Windows authentication to allow remote connections

- We open SQL Server Configuration Manager, navigate to SQL Server Network Configuration, then select the SQL Server instance and open TCP/IP

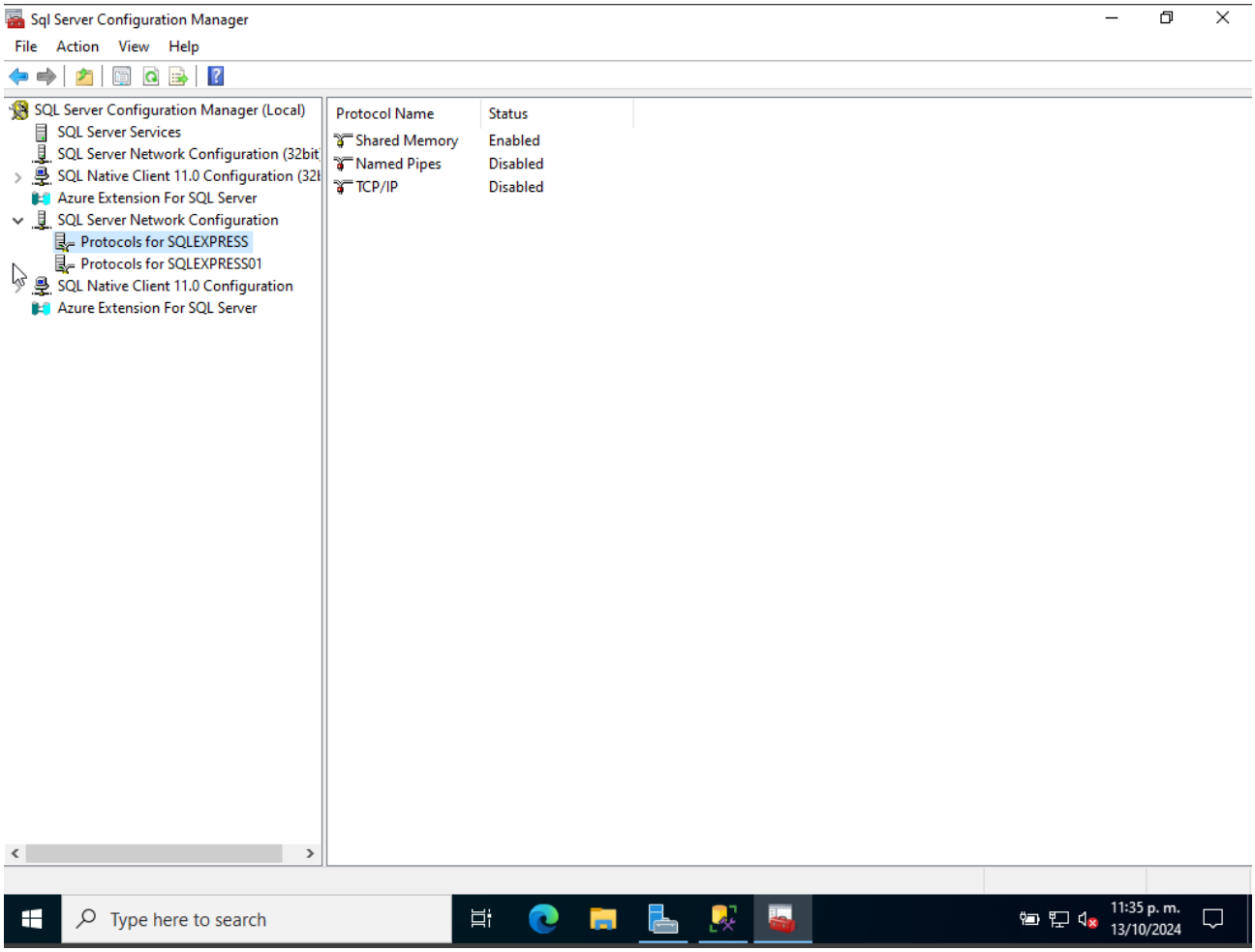


Figure 97110. Configuring TCP/IP settings for SQL Server

- In the ‘Enabled’ option, change it to ‘yes’

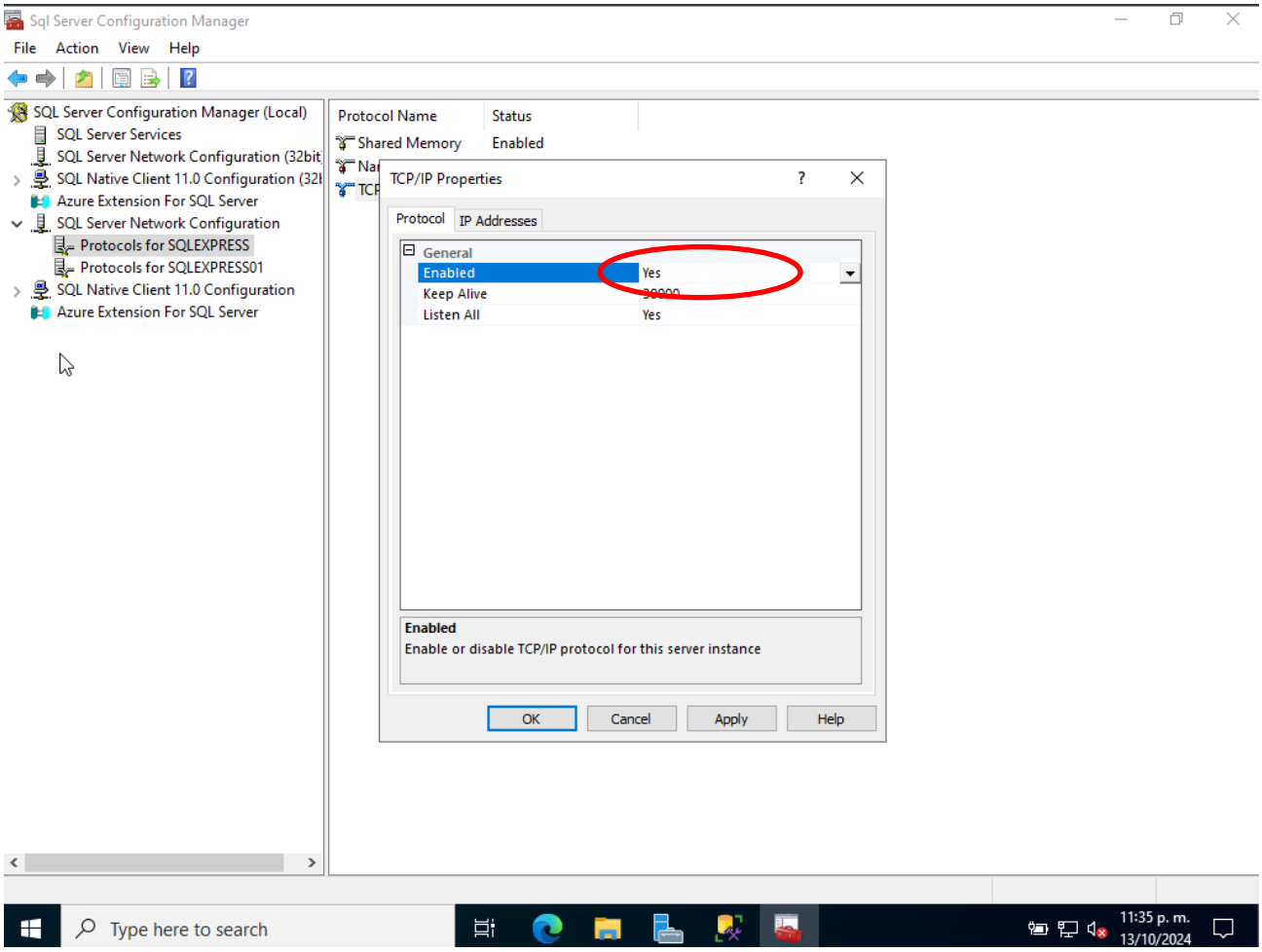


Figure 98111. Enabling TCP/IP protocol for SQL Server

- Then go to the 'IP Addresses' window, and in 'TCP Dynamics Ports' option, enter the SQL Server port: 1433

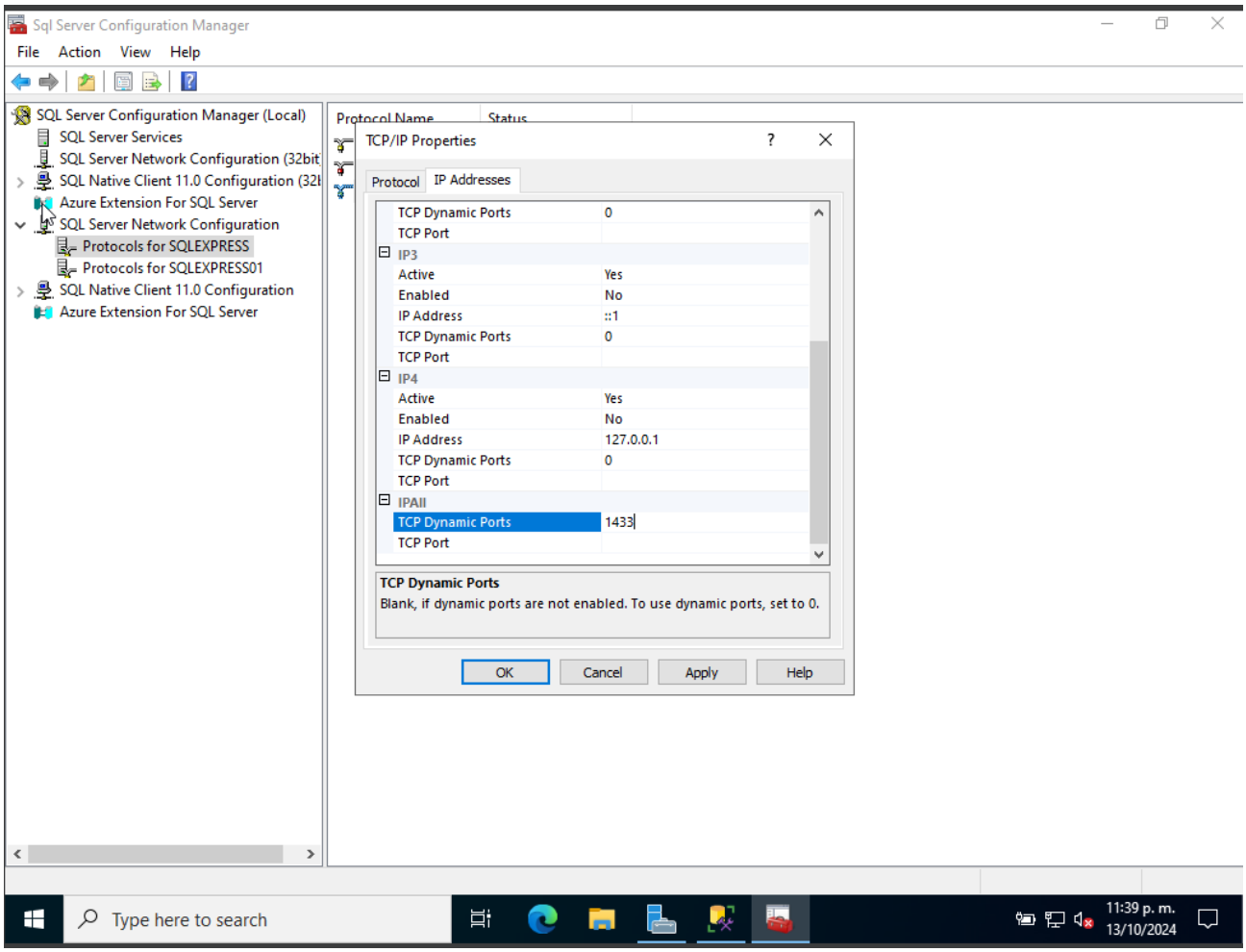


Figure 99112. Configuring SQL Server port

- Restart the service

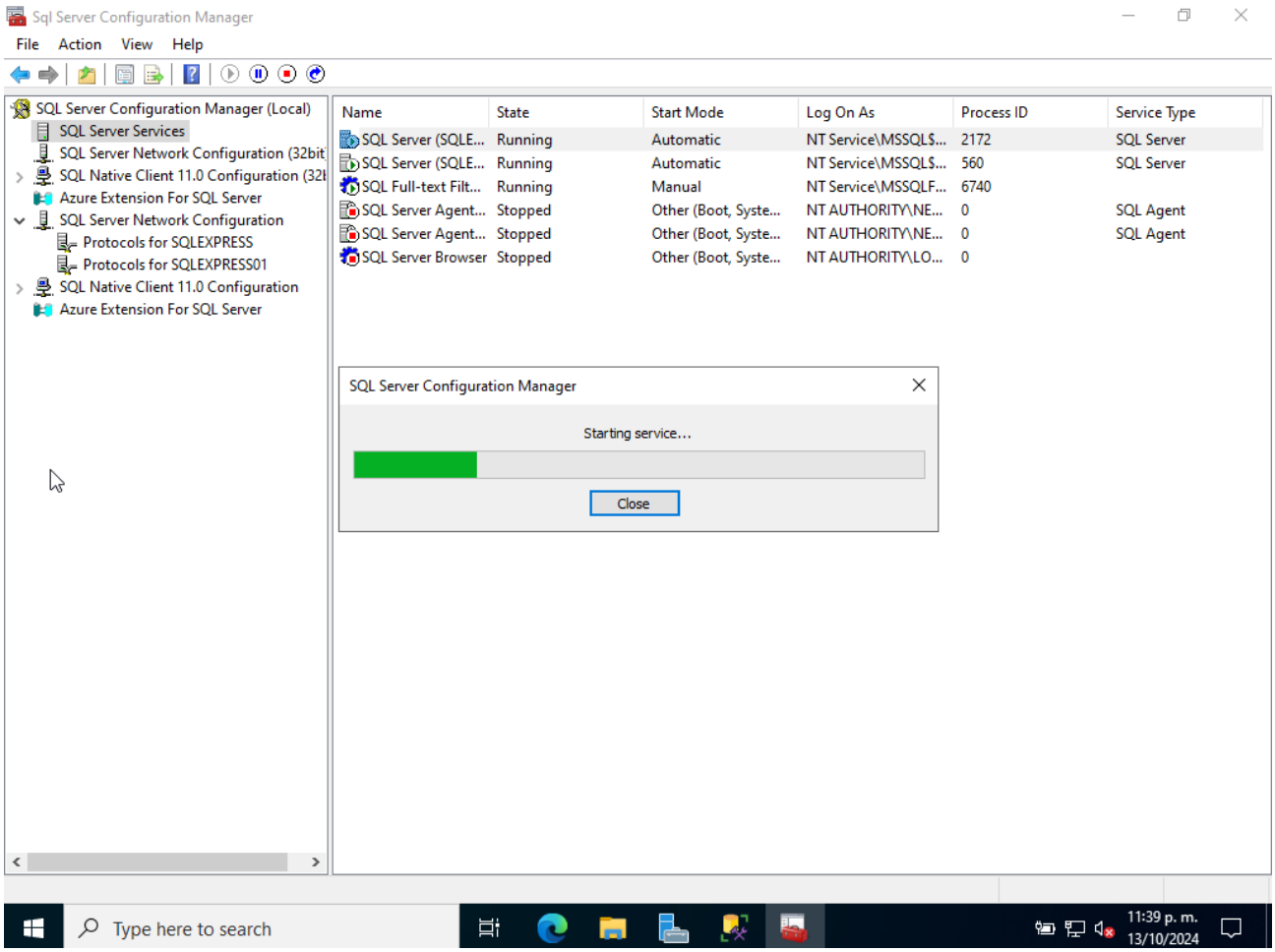


Figure 100113. Restarting SQL Server service

- Open ‘Windows Defender Firewall’, select ‘Inbound Rules’ and then click on ‘New Rule’

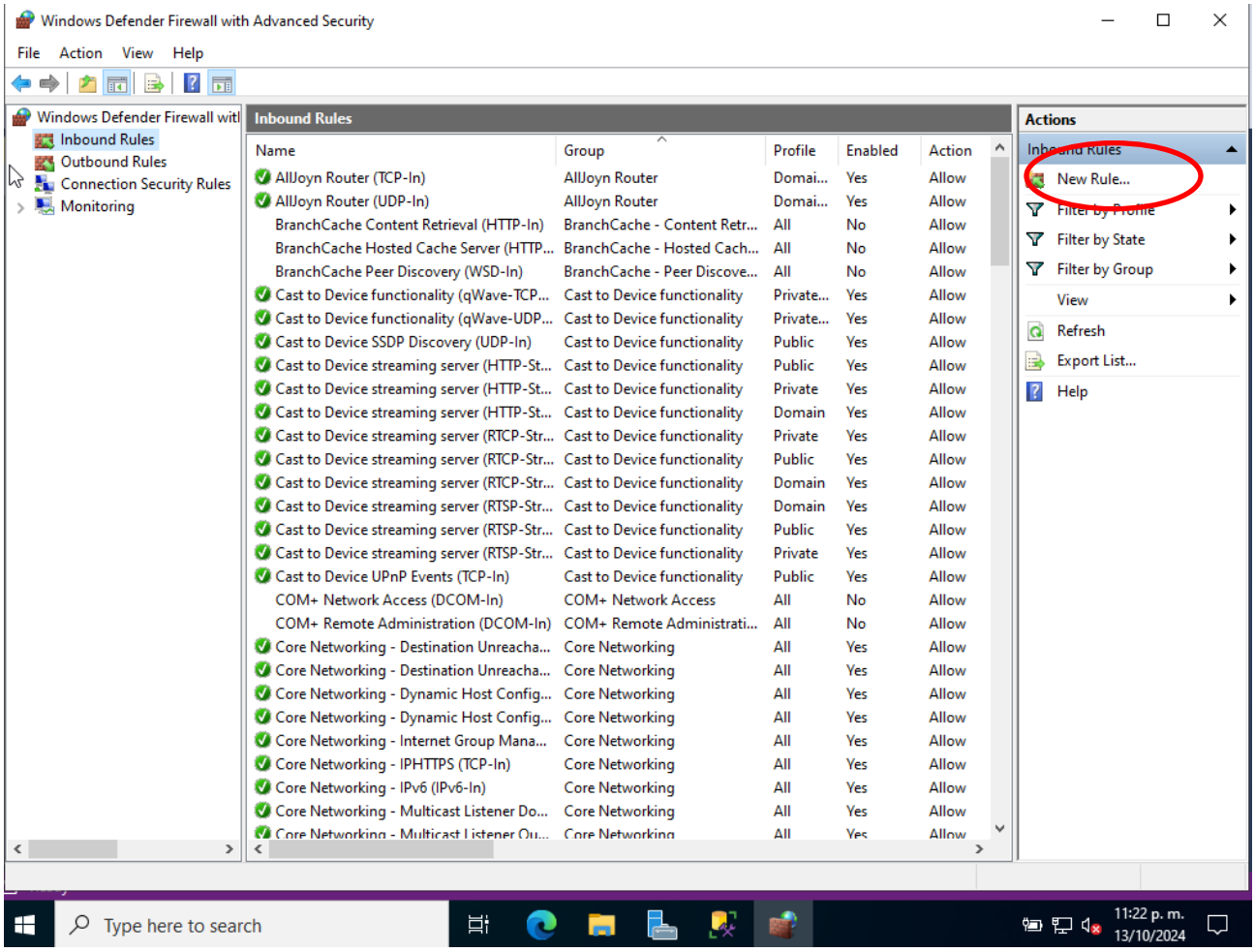


Figure 101114. Configuring firewall

- Select 'Port' and enter the SQL Server port

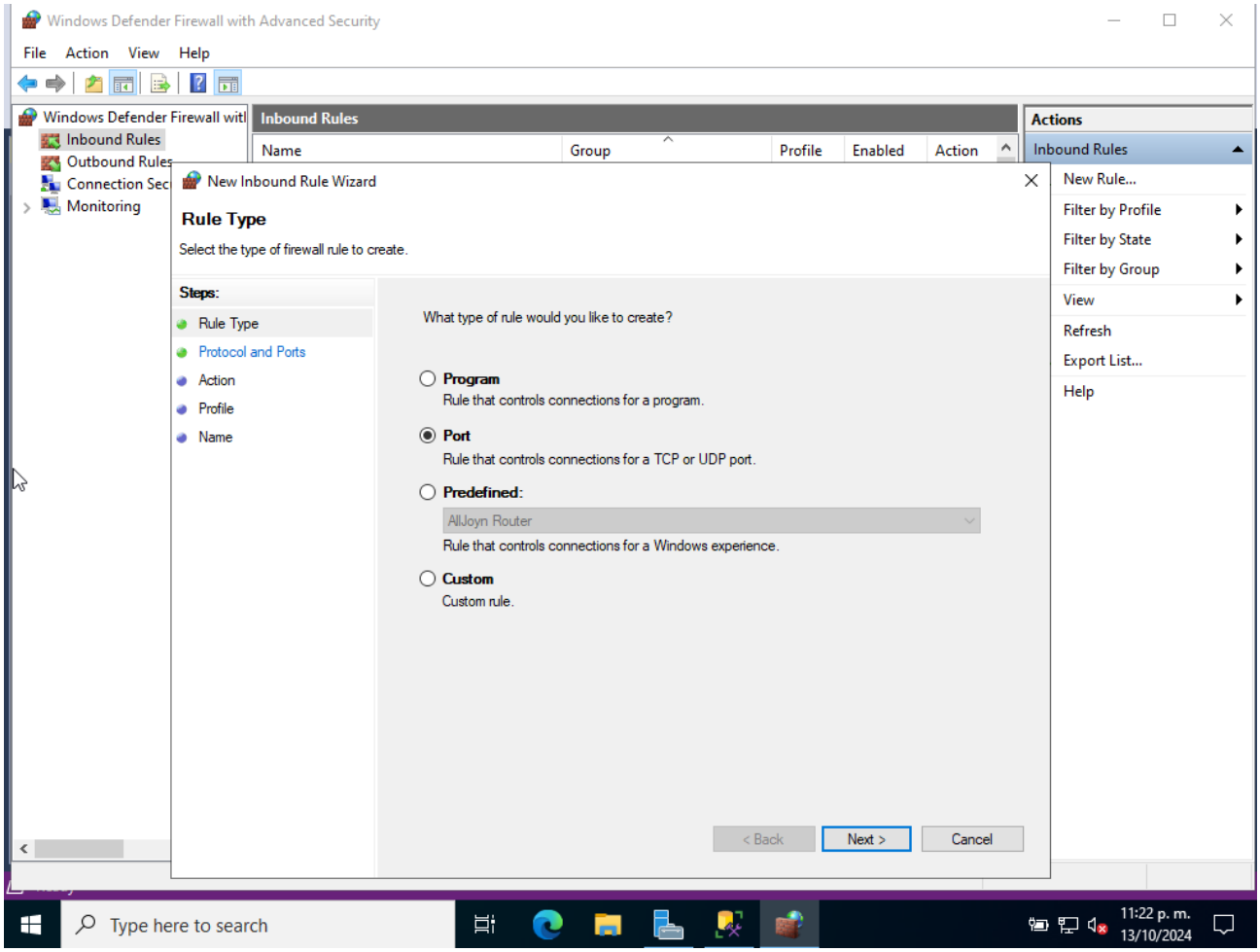


Figure 102115. Configuring a new port in the firewall

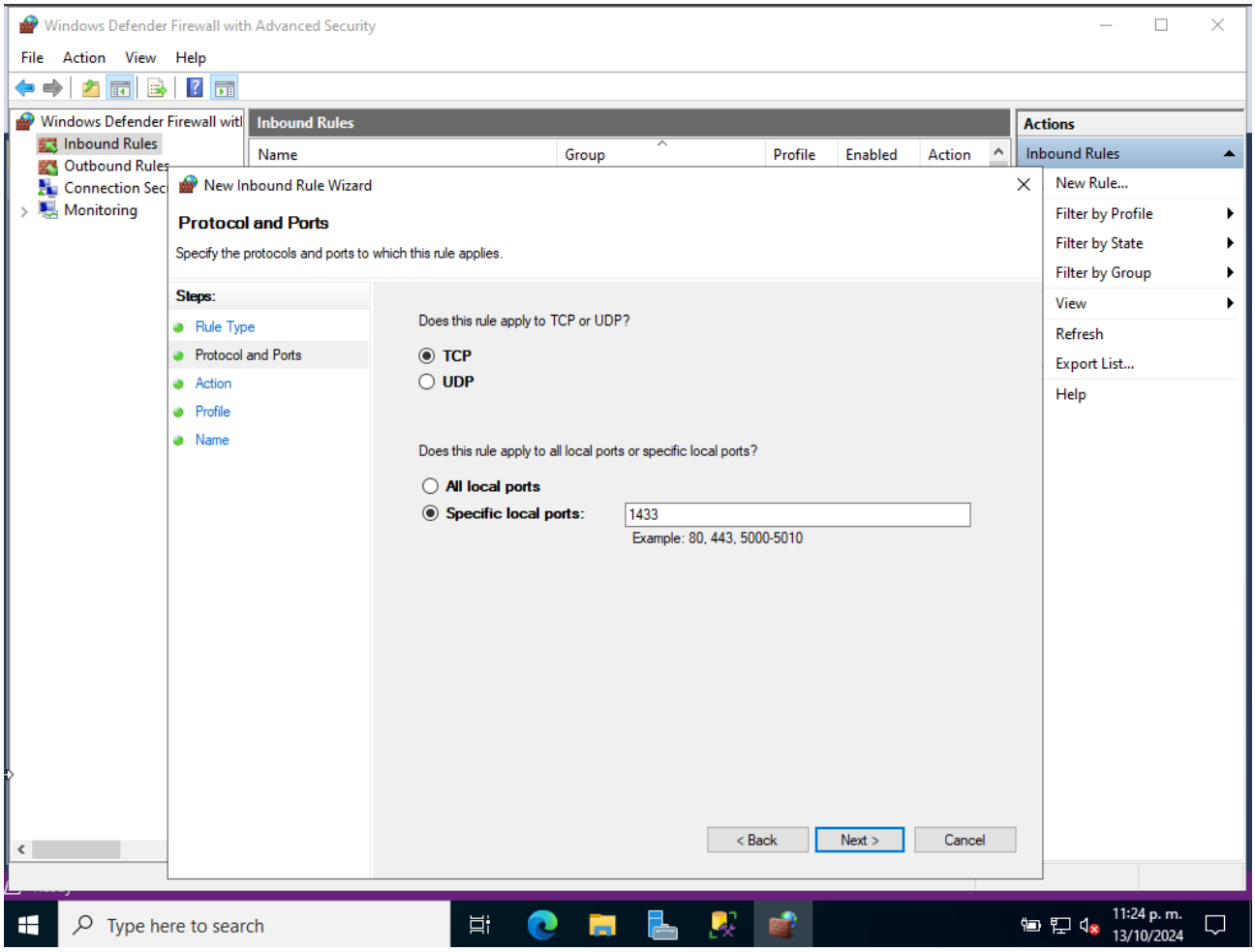


Figure 103116. Configuring the SQL Server Port in the firewall

- Click on ‘Next’ and select ‘Allow the Connection’

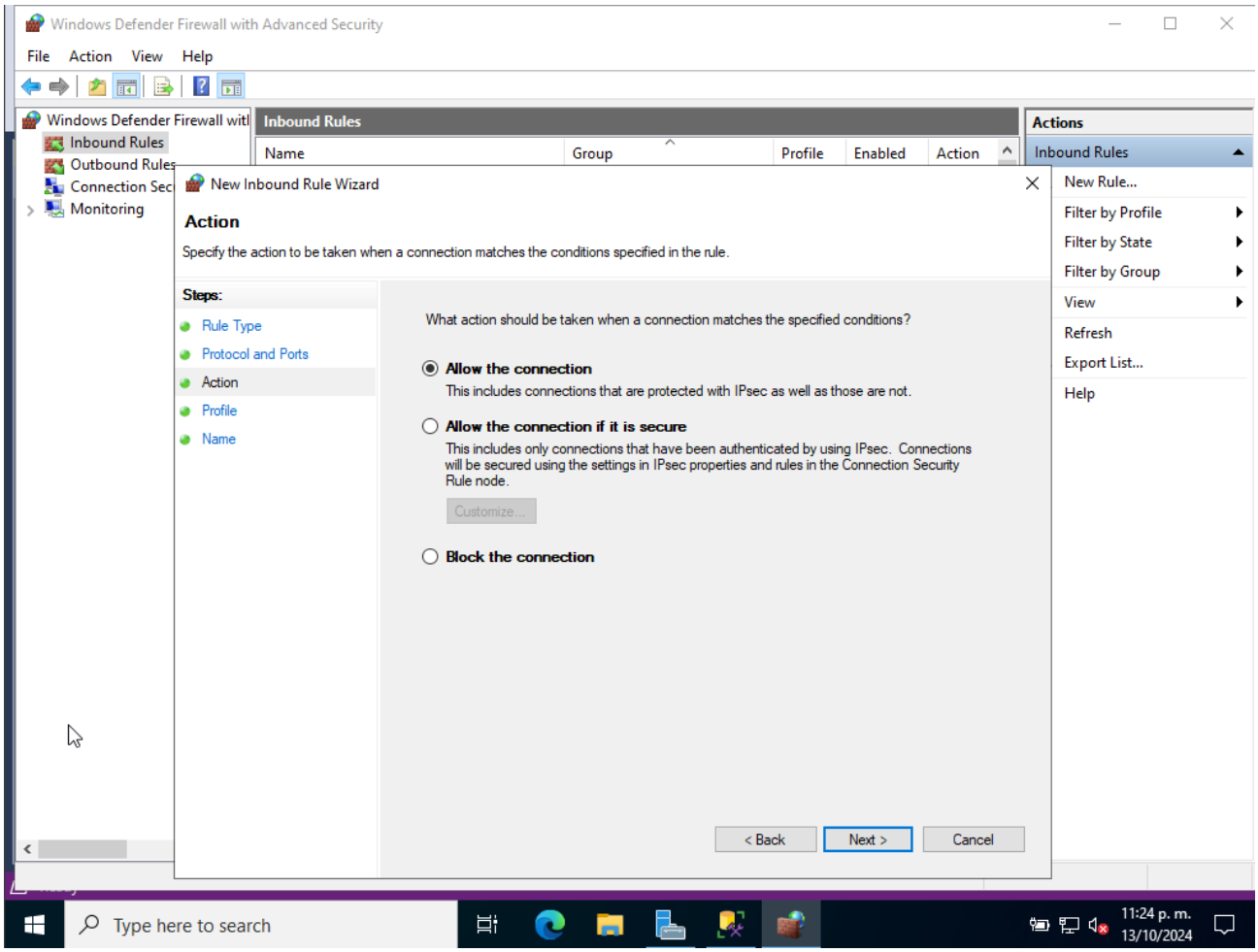


Figure 104117. Allowing the SQL Server port connection through firewall

- Select all the following options:

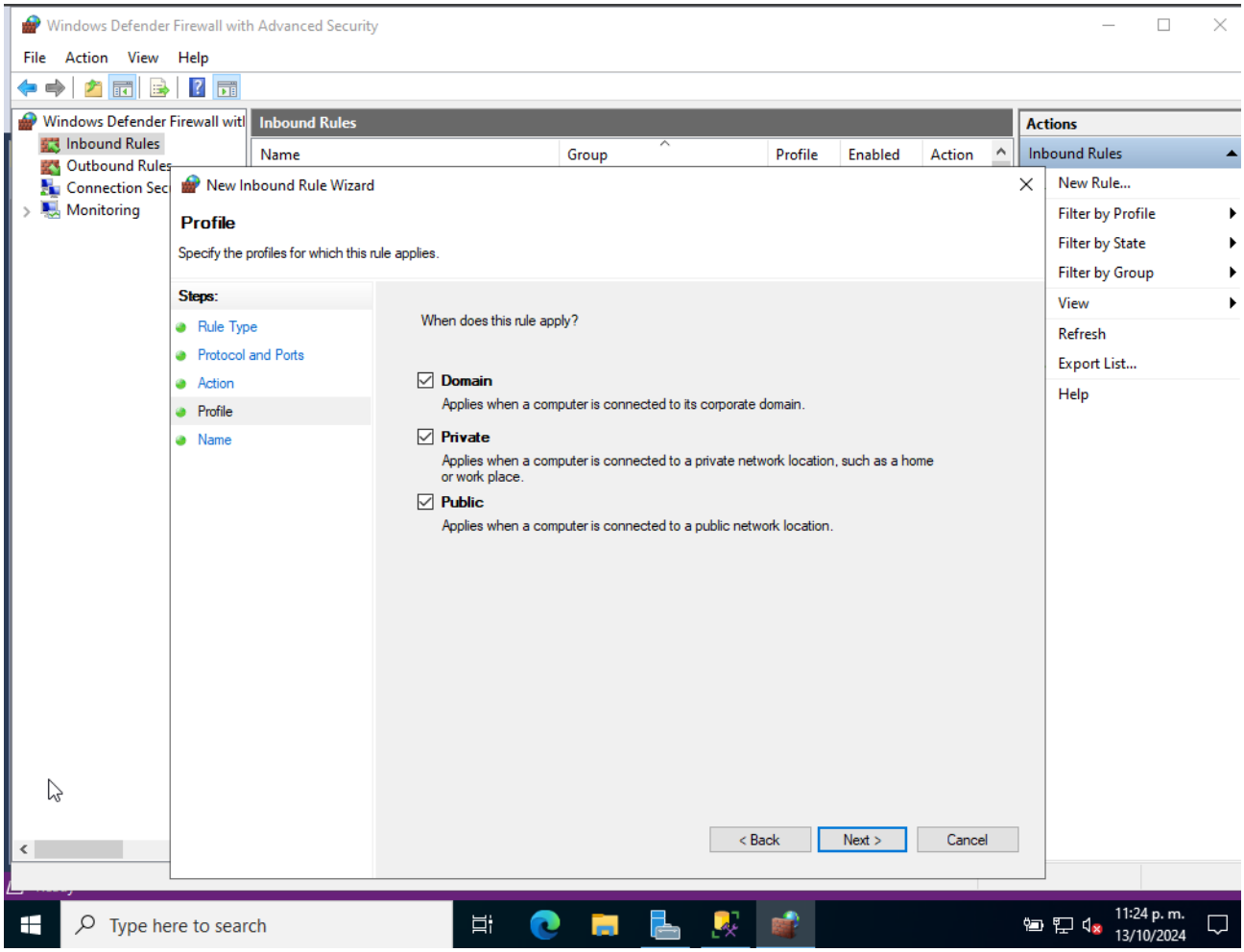


Figure 105118. Adding details to SQL Server Port in firewall

- Finally, give a name to the new rule; in our case we choose ‘database connection’

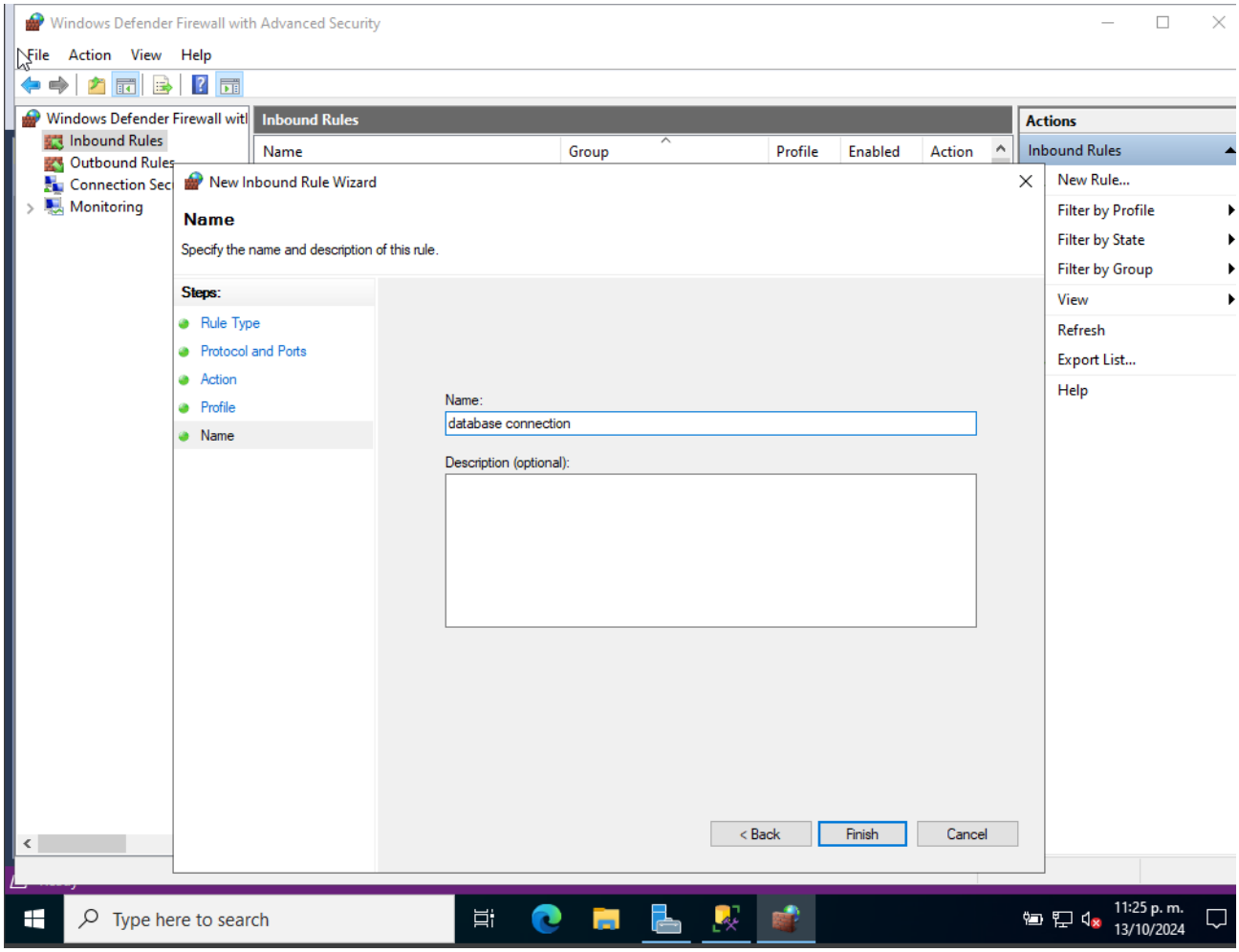


Figure 106119. Naming the new rule

- We open DBeaver to create a new connection

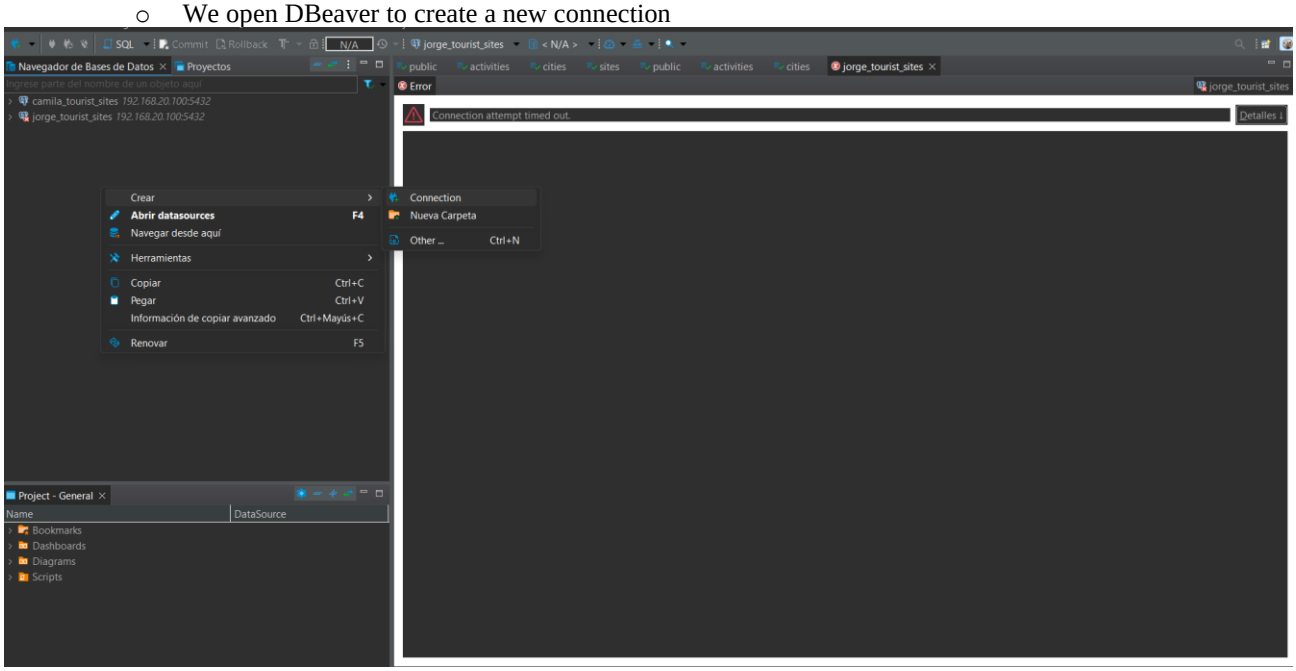


Figure 107120. Creating a new connection in DBeaver

- Select SQL Server

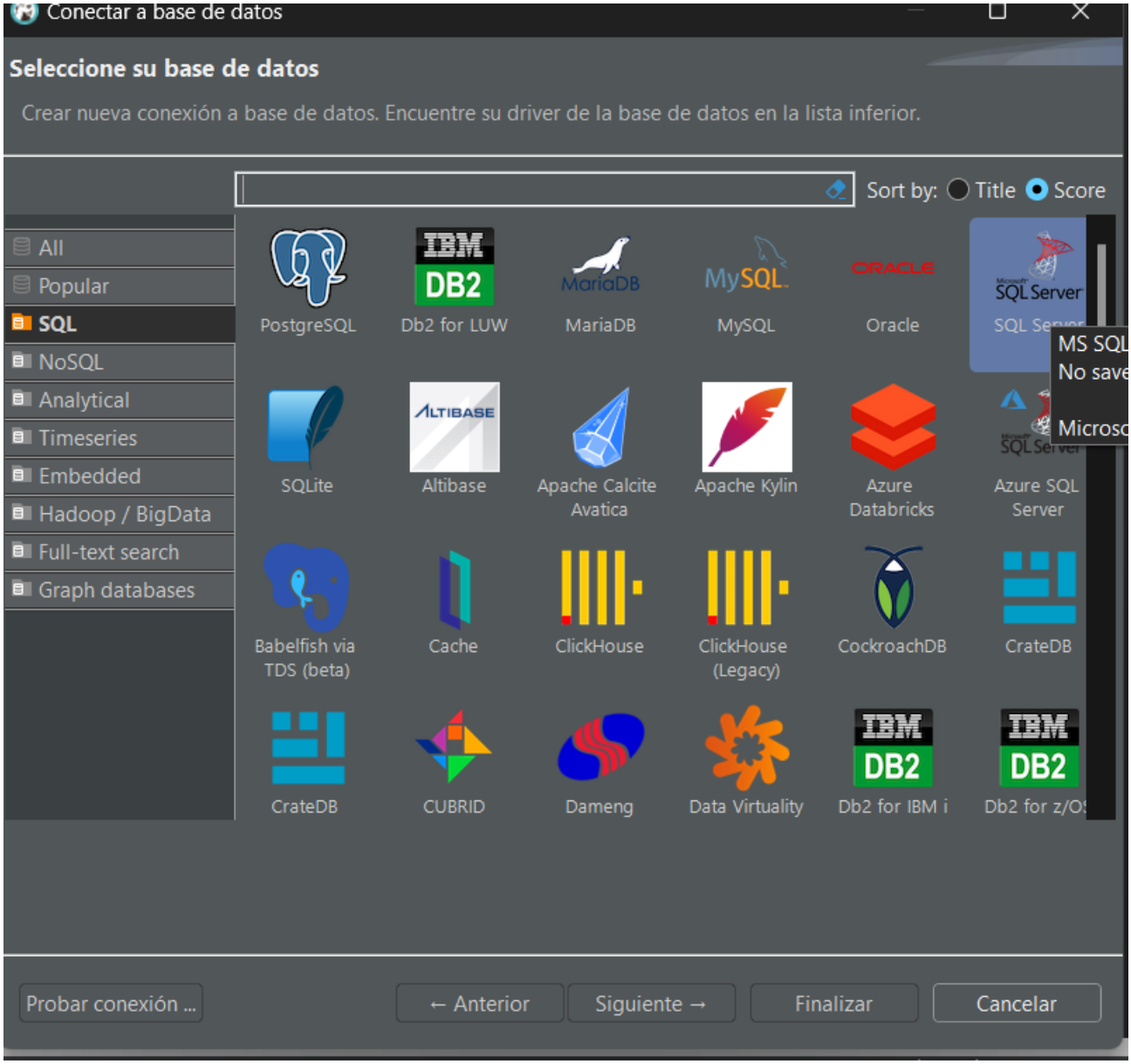


Figure 108121. Selecting SQL Server in DBeaver

- We enter the IP address of Windows, the database name (in this case camila_activity_schedule) and the username with the password

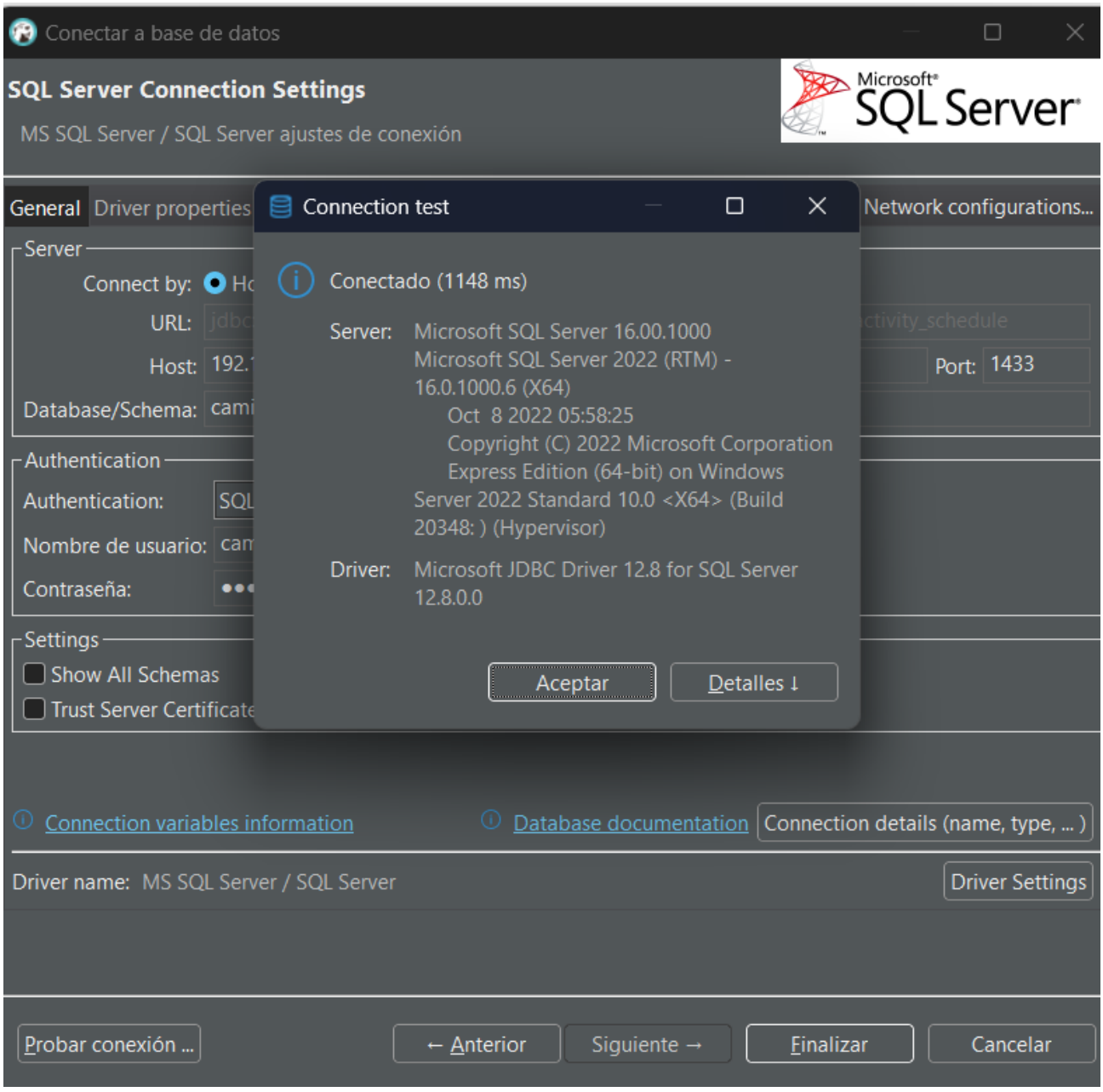


Figure 109122. Remote connection to camila_activity_schedule in DBeaver

- We can see the tables and the data that we inserted earlier

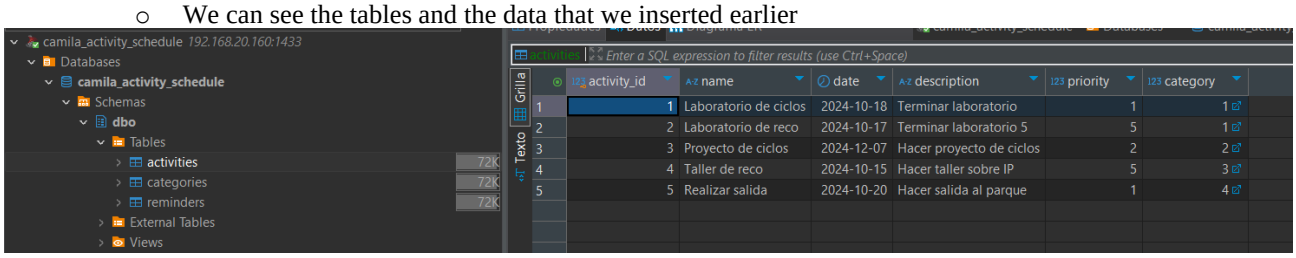


Figure 110123. Activities table of database camila_activity_schedule in DBeaver

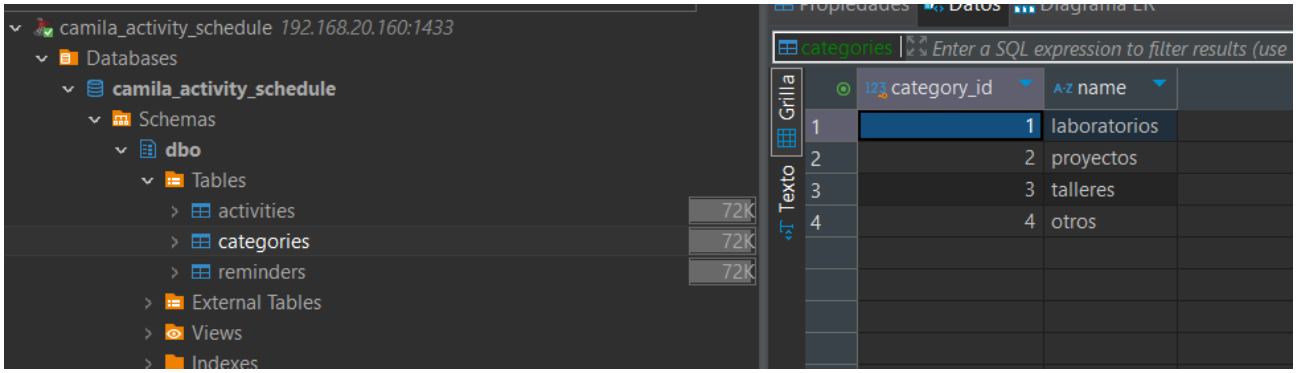


Figure 111124. Categories table of database camila_activity_schedule in DBeaver

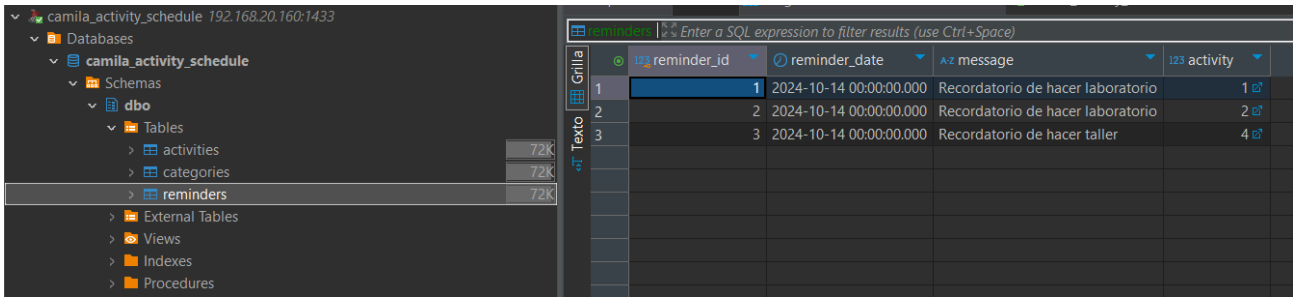


Figure 112125. Reminders table of database camila_activity_schedule in DBeaver

- To connect to jorge_activity_schedule, we follow the same steps as before, but with the user Jorge and the respective password

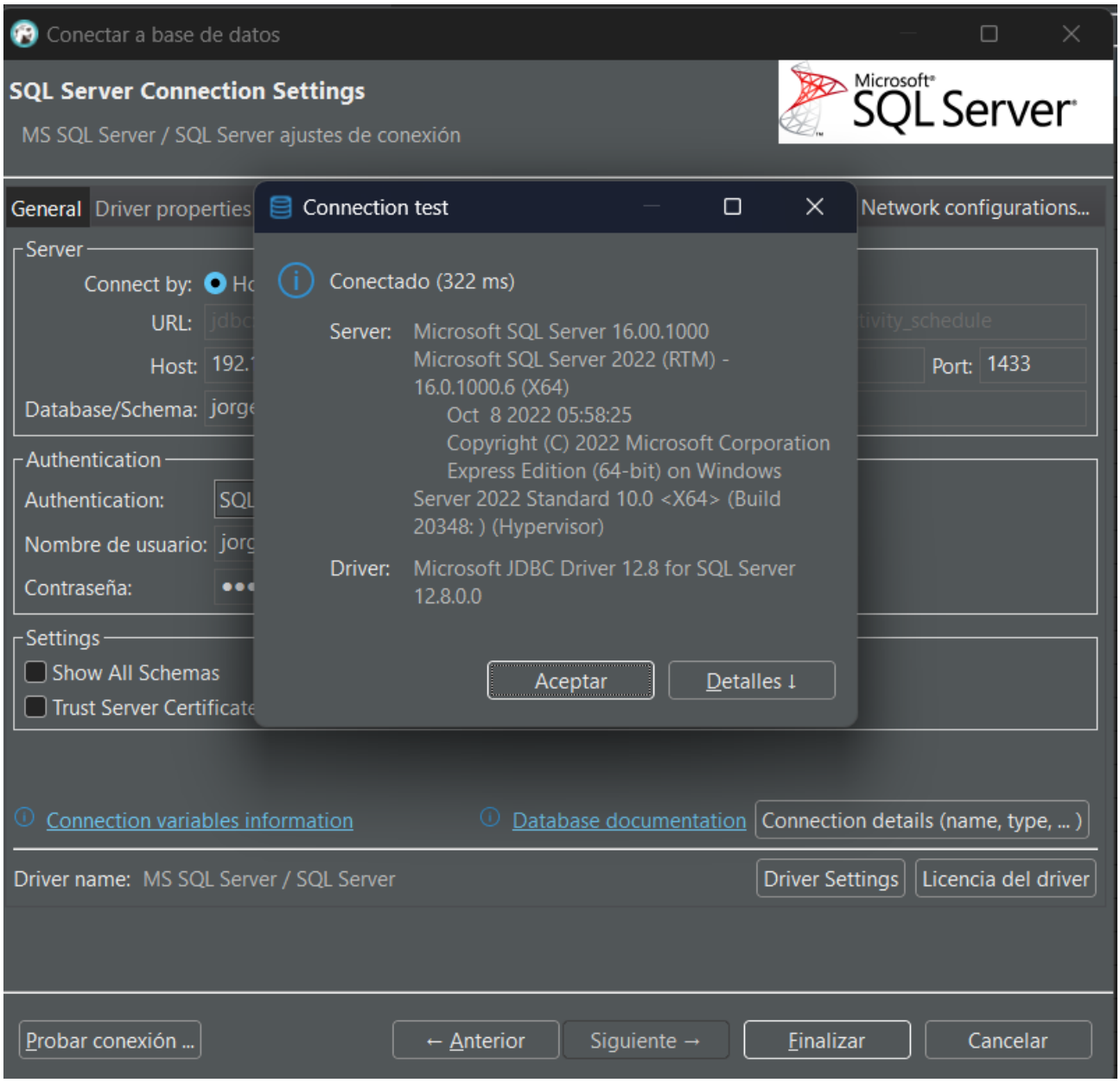


Figure 113126. Remote connection to jorge_activity_schedule in DBeaver

- We can see the tables and the data that we inserted earlier

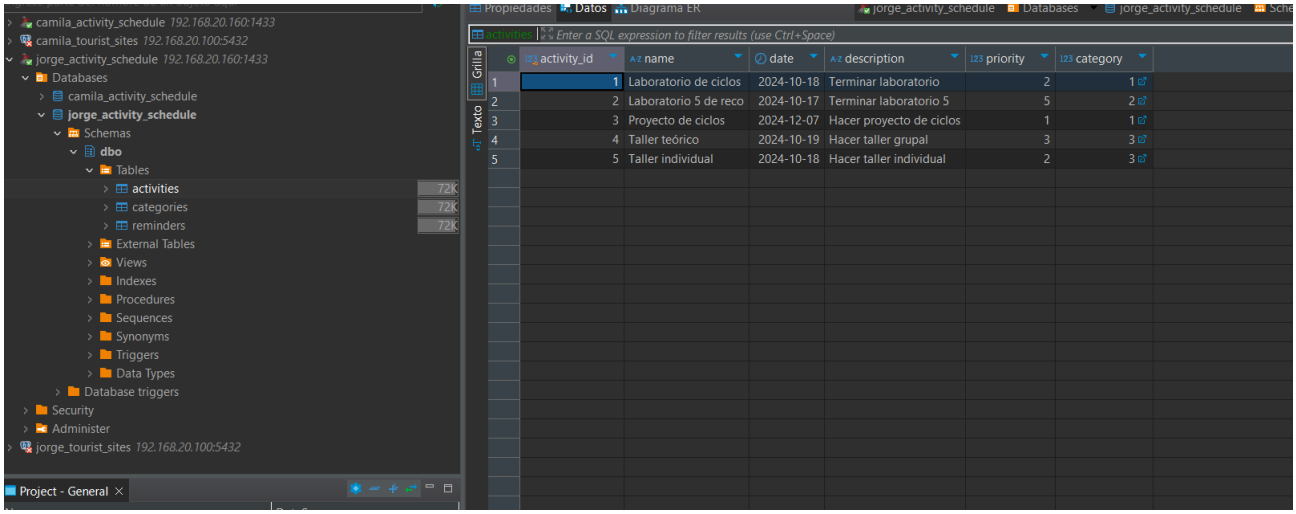


Figure 114127. Activities table of database jorge_activity_schedule in DBeaver

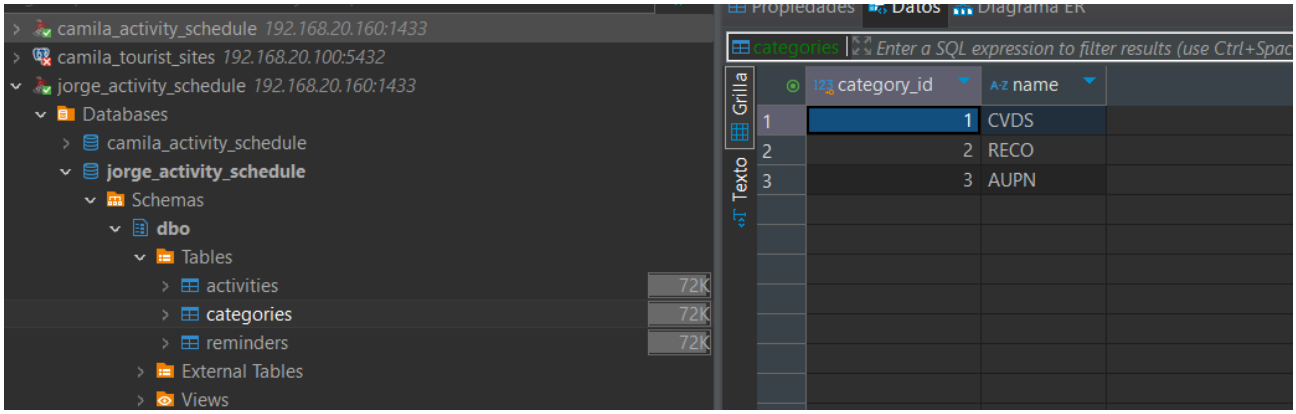


Figure 115128. Categories table of database jorge_activity_schedule in DBeaver

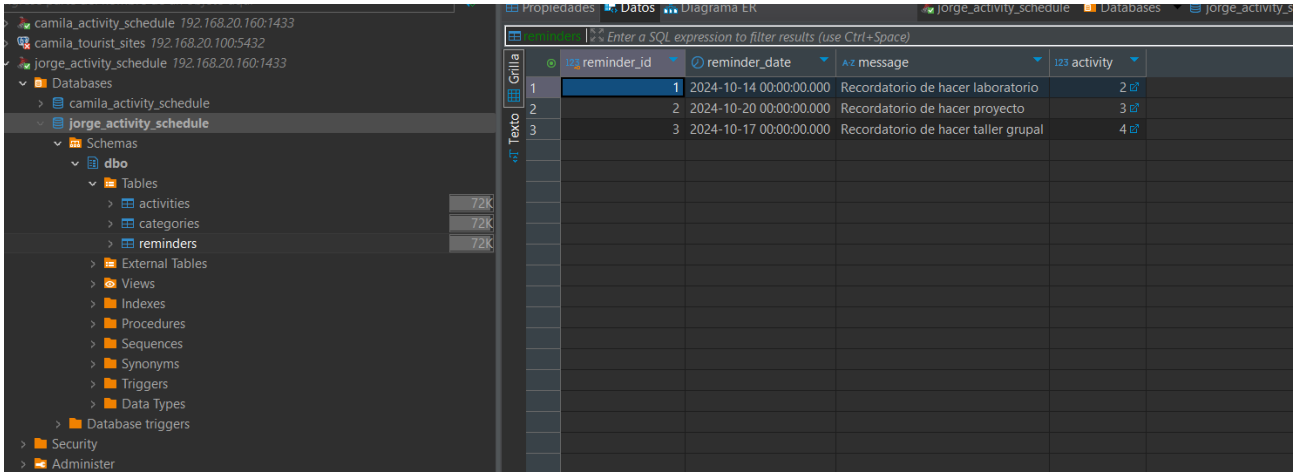


Figure 116129. Reminders table of database jorge_activity_schedule in DBeaver

- If we try to access another database with a different user, we find that we cannot log in with that user. Therefore, we verify that each user has access only to their own database.

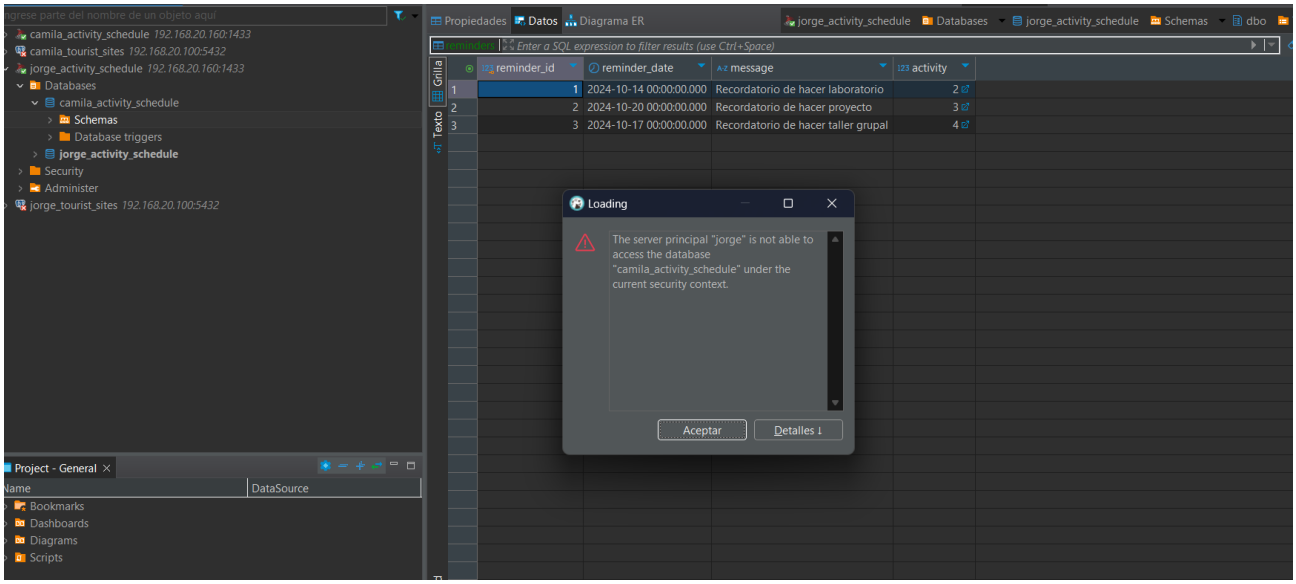


Figure 117130. Verifying that each user has access only to their own database

Conclusions

- **Understanding Network Protocols**: Using Wireshark, we examined and analyzed DNS messages, HTTP messages, and Ethernet frame headers. This activity helped us better understand how network communication and name resolution work. We identified key information in each field of the protocol headers, such as IP addresses, request methods, and content types.
- **Installation of Database Management Systems**: By installing and configuring database management systems like PostgreSQL and SQL Server on virtual machines, we became familiar with the database administration process. We created users and specific databases, which helped us learn about managing structured data and collaboration in shared environments.
- **Exploration of Cloud Computing with Azure**: Practicing in Microsoft Azure allowed us to understand the concepts of cloud computing and its advantages compared to on-premise infrastructures. By exploring the different services offered by Azure, we learned about IaaS, PaaS, and SaaS models, as well as the importance of vertical and horizontal scaling.
- **Connections and Security**: By comparing connections to cloud databases versus local databases, we highlighted the relevance of security in data transfer. We investigated how TLS affects communication, understanding the importance of handling TCP connections properly to secure access to sensitive data in a cloud environment.
- **Development of Practical Skills**: This laboratory not only strengthened our theoretical knowledge about networks and databases but also provided us with essential practical skills for configuring and managing IT infrastructure. We experienced firsthand the importance of installation, management, and monitoring processes in both local and cloud environments.
- **Responsibility in Resource Management**: By removing the resources created in Azure at the end of the laboratory, we understood the importance of properly managing cloud resources to avoid unnecessary costs. This aspect of the laboratory fostered our responsibility in using cloud services and our understanding of resource management in a business setting.

bibliography

- *Azure availability zones*. (n.d.). Microsoft.com. Retrieved October 14, 2024, from https://azure.microsoft.com/en-us/explore/global-infrastructure/availability-zones
- *Connectivity settings for Azure SQL Database and Azure Synapse Analytics*. (n.d.). Microsoft.com. Retrieved October 14, 2024, from https://learn.microsoft.com/en-us/azure/azure-sql/database/connectivity-settings?view=azuresql&tabs=azure-portal
- Es, C. [@campusmvp]. (n.d.). *Instalar SQL Server 2022 gratis desde cero (y bien) [TUTORIAL]*. Youtube. Retrieved October 14, 2024, from https://www.youtube.com/watch?v=_fFz-_O2yvI
- Hidalgo, M. (2023, January 30). *Cloud computing service types: IaaS vs PaaS vs SaaS*. Tutorials Dojo. https://tutorialsdojo.com/cloud-service-types-iaas-vs-paas-vs-saas/
- *Select Azure regions*. (n.d.). Microsoft.com. Retrieved October 14, 2024, from https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-setup-guide/regions
- Slingerland, C. (2024, July 18). *Horizontal vs. Vertical scaling: Which should you choose?* CloudZero. https://www.cloudzero.com/blog/horizontal-vs-vertical-scaling/
- (N.d.). Cleo.com. Retrieved October 14, 2024, from https://www.cleo.com/blog/knowledge-base-on-premise-vs-cloud#:~:text=Cloud%20Computing,-Cloud%20computing%20differs&text=A%20company%20hosts%20everything%20in,the%20growth%20of%20a%20company.
- *Microbuffer*. (2010, septiembre 23). Microbuffer. https://microbuffer.wordpress.com/2010/09/23/instalacion-de-postgresql-en-slackware/