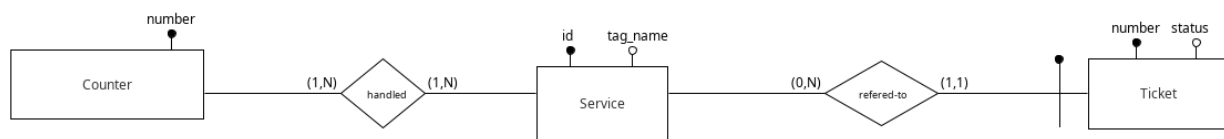


Office Queue Management System

Conceptual design

In this phase, the entity-relationship model of the system was developed, where it was necessary to represent all the entities involved and the relationships required between them. Specifically, in this sprint, the system required the presence of two particular types of users, namely, Customer and Officer, each of whom interacts with the system differently. Furthermore, customer registration was deemed redundant as it was considered "anonymous," as it is closely tied to the ticket entity, and thus, only the ticket would be sufficient to represent the action associated with a user. We can tell the same for the officer, because each counter is managed by an officer so it would be implicit that counter has an officer (it's not required, in this moment, to access as an officer).



Logical design

Translation of entities and associations

- Counter(**number**)
- Handled(*counter_number*, *service_id*)
- Service(**id**, tag_name)
- Ticket(**number**, status, *service_id*)

attribute: in bold => primary key

attribute: in italics => foreign key

Referential constraints

- Handled(counter_number) \subseteq Counter(number)
- Handled(service_id) \subseteq Service(id)
- Ticket(service_id) \subseteq Service(id)

Counter

Attribute	Tipology	Description
number	Integer	The number attribute describes the number that is assigned to that particular counter, and being a primary key is unique to each counter.

Handled

This particular table traces all the correspondences between service and counter, that is identifies for every counter all the services that can manage.

Attribute	Tipology	Description
counter_number	Integer	Reference to the counter number
service_id	Integer	Reference to the service id

Service

Attribute	Tipology	Description
id	Integer	Unique indentifier for a service
tag_name	String	Descriptive name for that service

Ticket

Contains all tickets generated by the system. There is a particular field called "status" that has the responsibility to track the current status of the ticket that can take one of the following values:

- 0: waiting
- 1: being served
- 2: served

This field is very important for the implementation of the queue, because it would be enough to update this state in order to update the queue for that service.

Attribute	Tipology	Description
number	Integer	Number assigned to that ticket
status	Integer	Status of the ticket
service_id	Integer	Service id reference

Installation

For installation, it is recommended to use the latest version of [PostgreSQL](#). A highly recommended tool for advanced and functional database management is [DBeaver](#). This tool provides a wide range of features for working with the respective database.

How to install

It is necessary to change the password of the "postgres" user to establish a successful connection between the JavaScript backend and the database on your machine. The user should be configured with the following credentials:

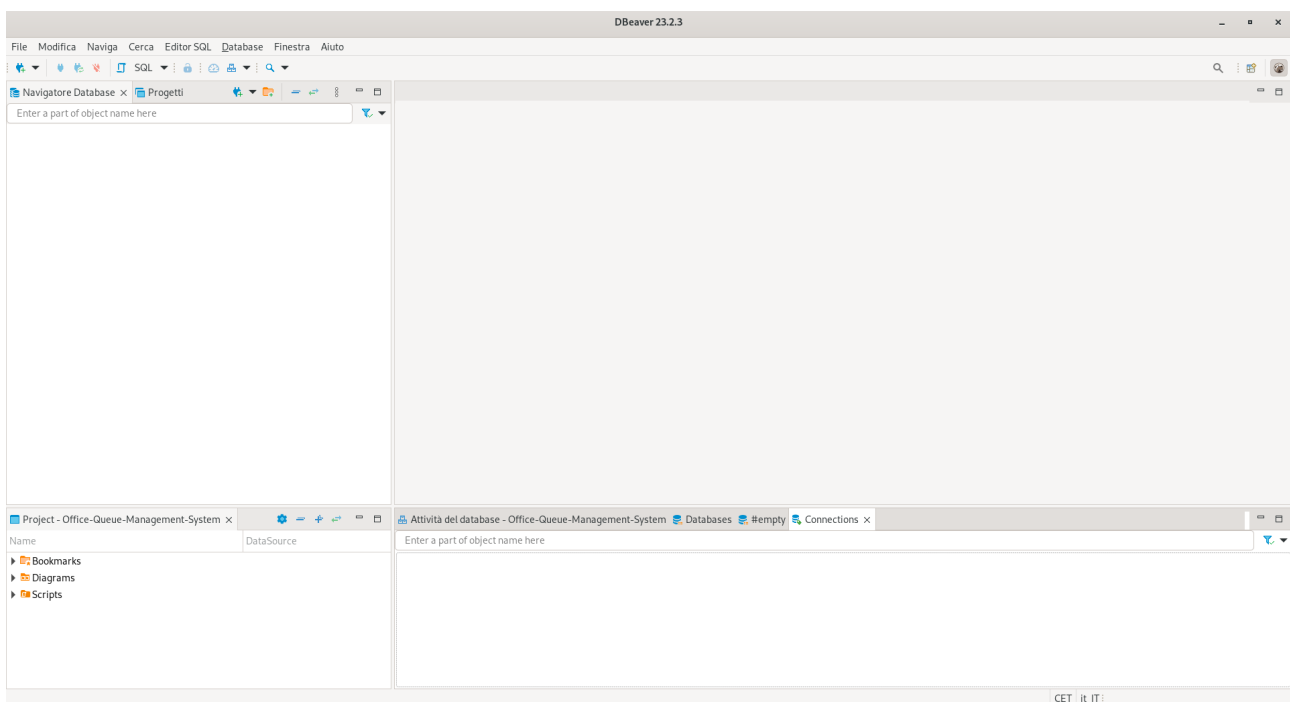
- User: "postgres"
- Password: "group15-postgres"

Once you have completed these steps, you are ready to import the database on your machine. There are two methods to import the dump file

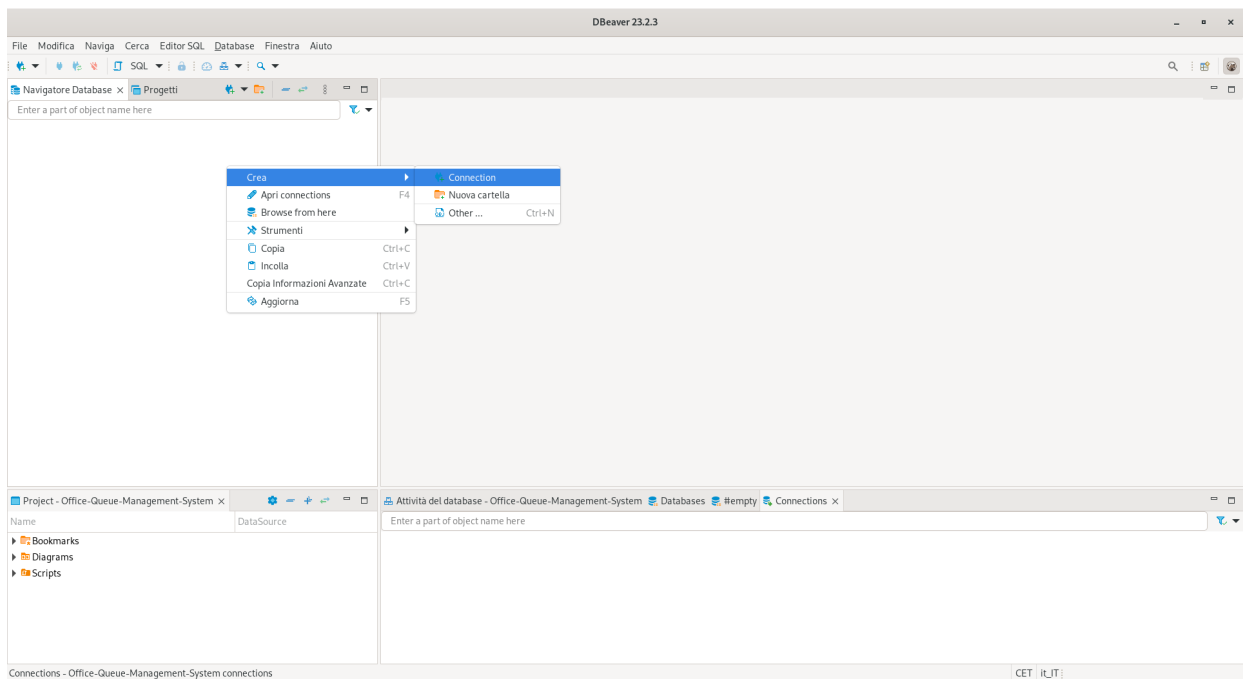
- Import it with Dbeaver
- Import it from terminal

Dbeaver Method

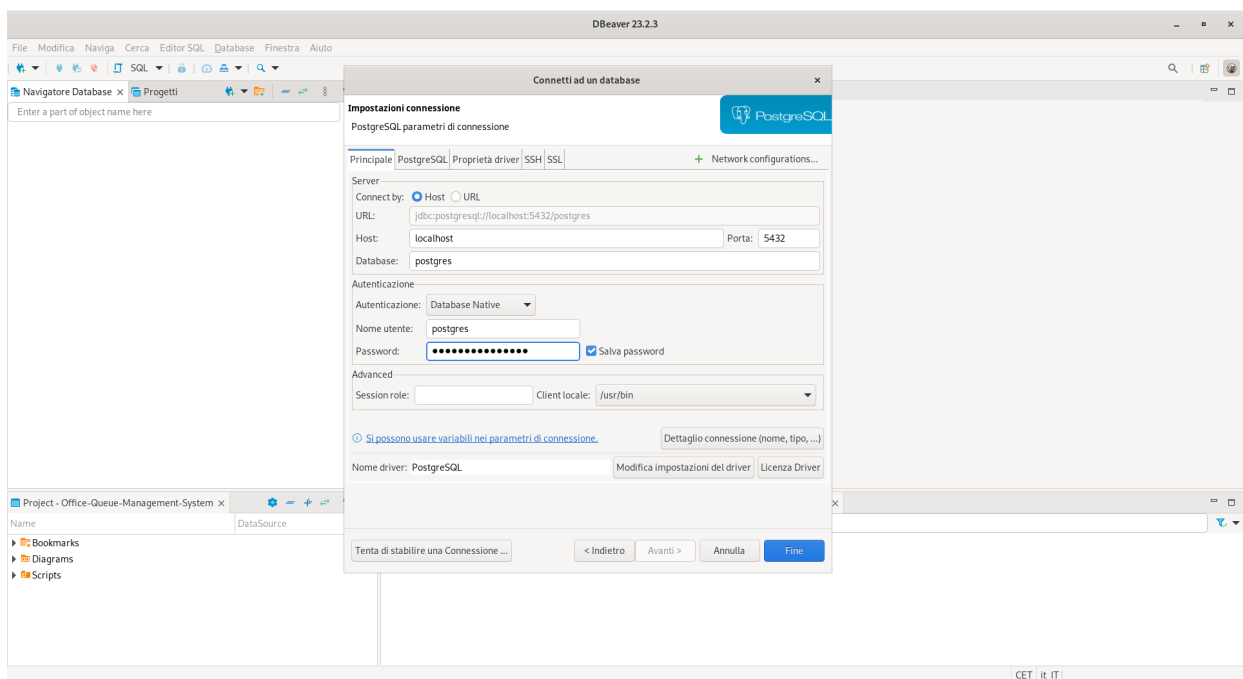
1. Open Dbeaver



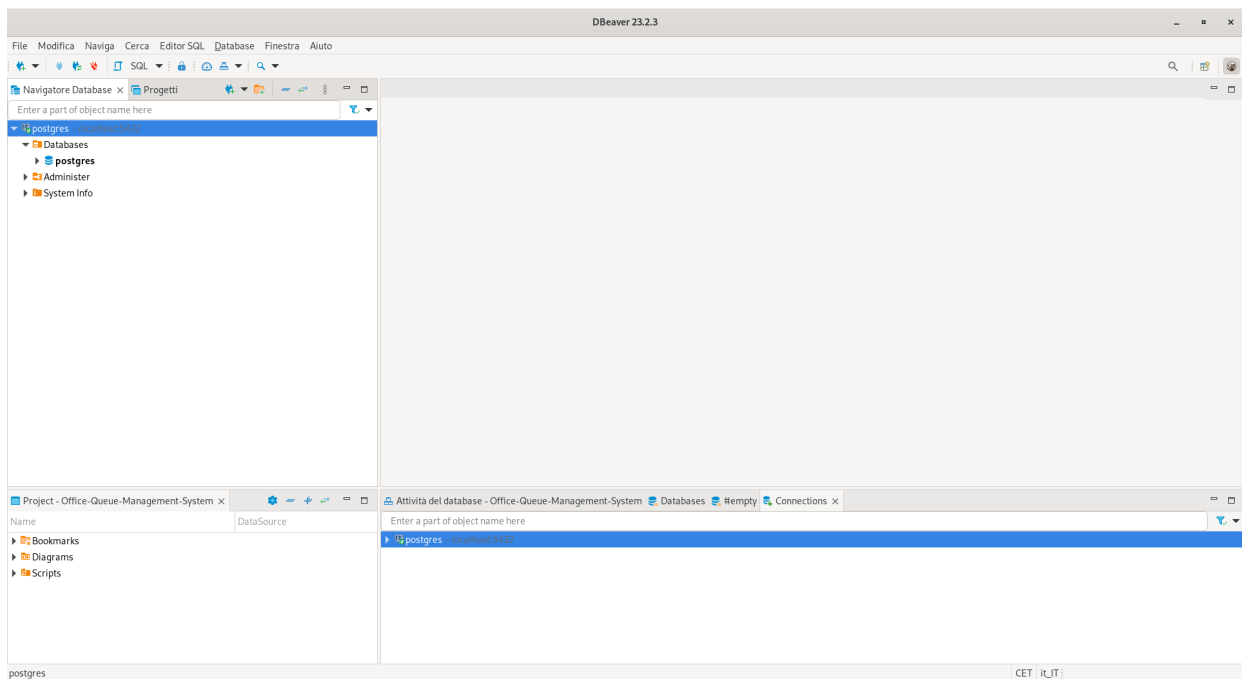
2. Create a new connection, choose PostgreSQL and click on Next



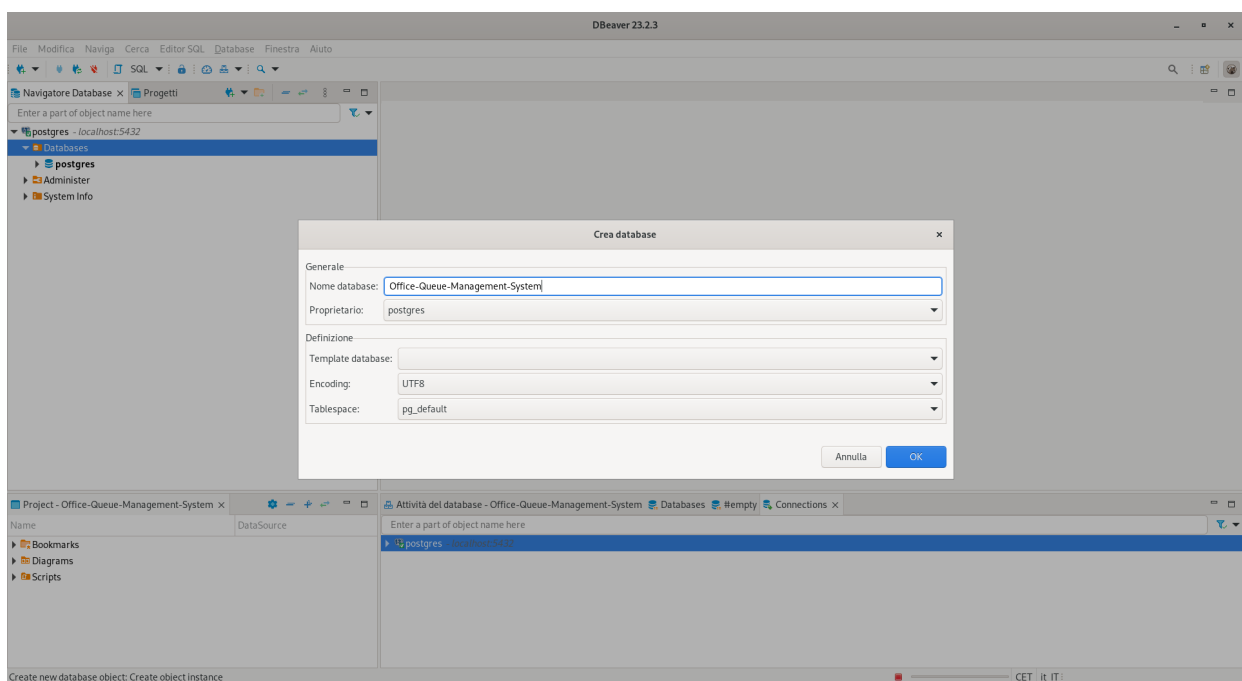
3. Insert “postgres” as username and “group15-postgres” as password (important: modify the password of the user “postgres” to “group15-postgres” if not previously done) and click on “End”



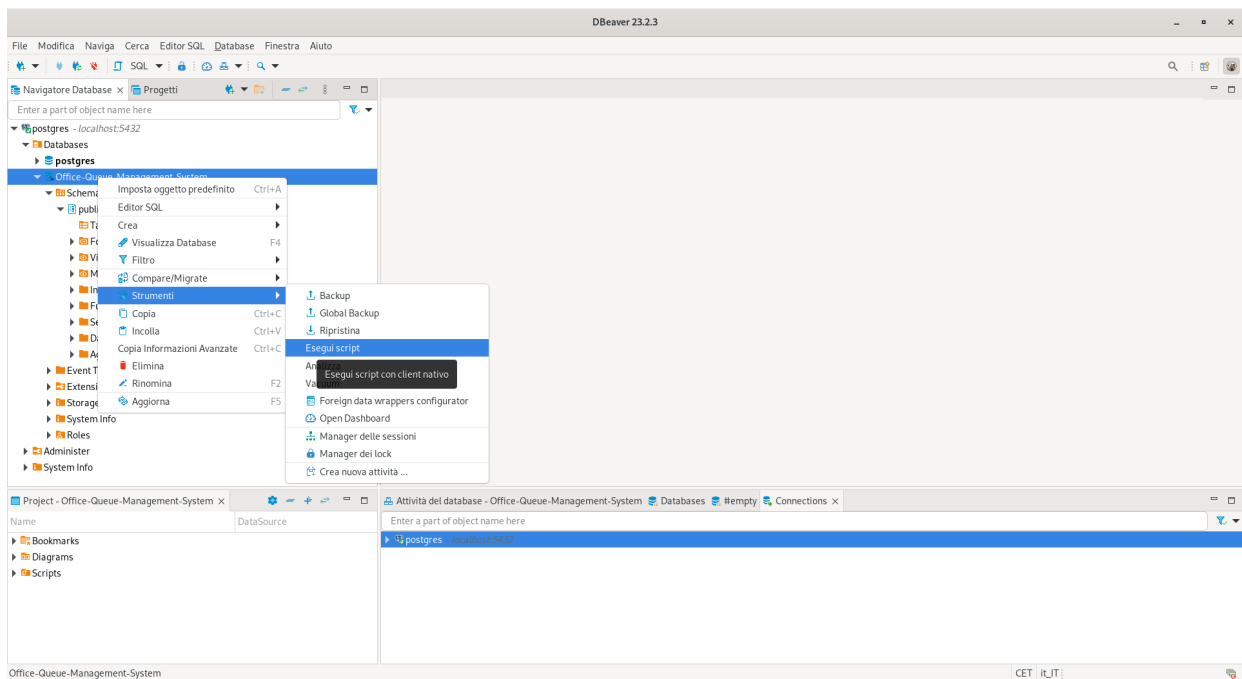
4. Will displayed this page



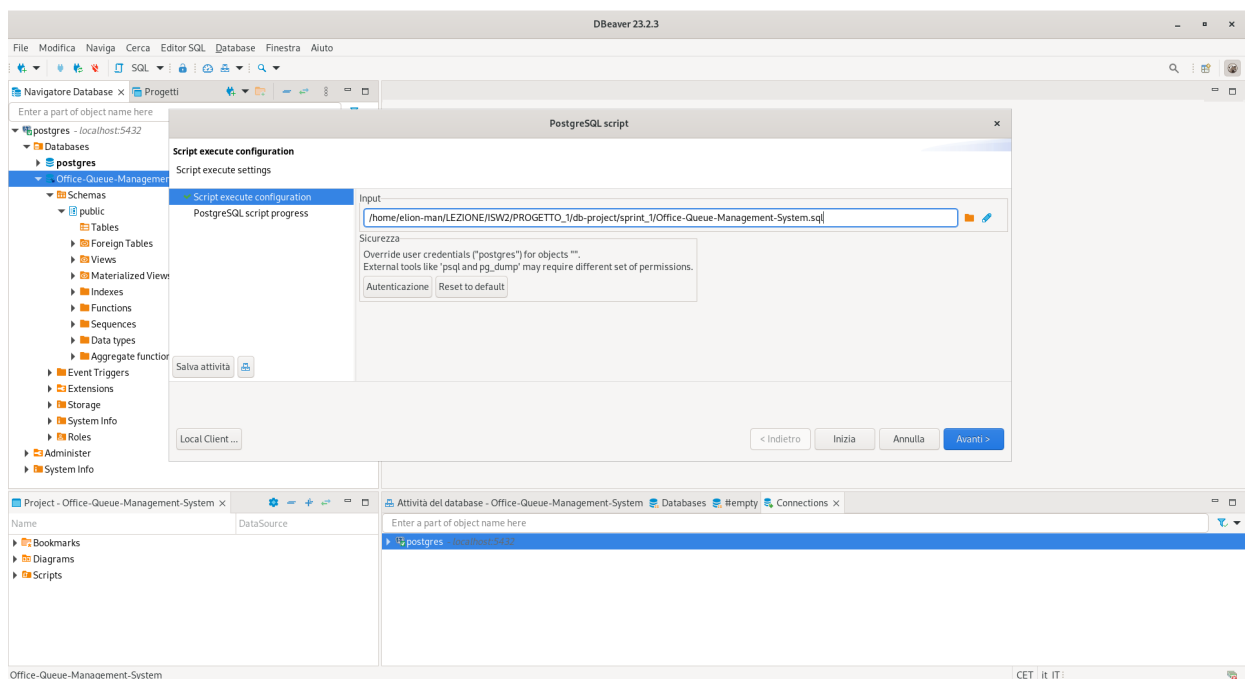
5. Click on Databases, click on Create a new Database and call it “Office-Queue-Management-System”.



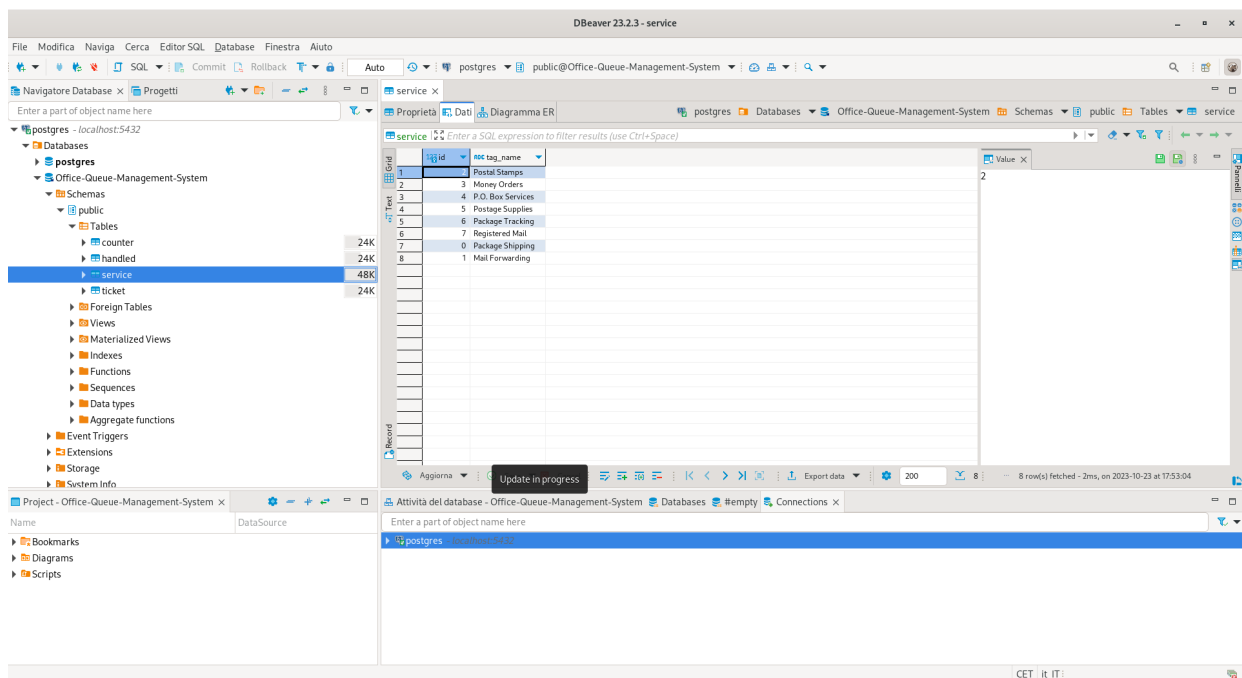
6. Once created the database, we have to run the sql file. So right click on the created database, click on tools and finally run scripts.



7. Insert as input the path of the file “Office-Queue-Management-System.sql” in the Input field requested. And finally, click on Start.



8. If at the end of the execution an error returns "Task execution failed, reason: ERROR: you cannot delete the currently opened database." you can ignore it. The database has been successfully loaded:



Terminal Method

To perform these steps, you need to have PostgreSQL installed on your computer to use the provided commands. Additionally, please note that these steps have only been tested on Linux (Debian). In the same repository folder, a file named 'load_db.sh' has been uploaded, which, when executed, allows you to load the database automatically. You will only need to enter the password “group15-postgres” when prompted.

If the file is not in executable form, type the command “sudo chmod 744 load_db.sh” to make it executable.

To run the file, you only need to execute the following command: “./load_db.sh”.

Database Access via Terminal

To access the database via the command line, we use the “psql -U postgres -W” command, and it will prompt us to enter the password. Once we are granted access, we can type the command “\l” to view the list of all databases in the system, including “Office-Queue-Management-System”.

Using the command '\c Office-Queue-Management-System', we can select the desired database (this may prompt us to enter the password again). With the command '\d', we can retrieve a list of all available tables.

Alternatively, we can log out using “\q” and then enter the command “psql -d Office-Queue-Management-System -U postgres -W”, where, after entering the correct password, we will be directly logged into the specified database.