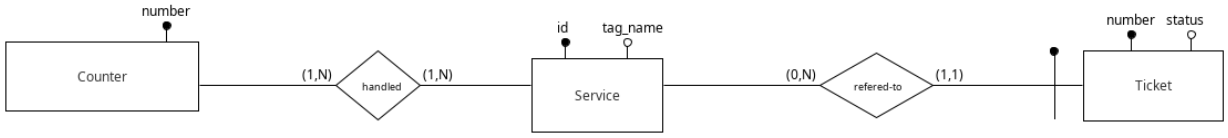
Office Queue Management System

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) ⊆ Counter(number)
- Handled(service_id) ⊆ Service(id)
- Ticket(service_id) ⊆ 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.

When the installation is complete, you must run the file . sql in the folder "db-project/sprint_1" which will create the database with the properties described above.

Furthermore, it is necessary to create a special user to enable 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”