Introduction:

This report describes the progress of the design of a database for the administration of a residential complex. The main objective is to manage the logistics of residents, their guests, common spaces and communication with the administration. For this, a survey was carried out on 15 students and based on these responses, a list of “user stories” was built in order to more clearly define the requirements of the project. From the latter, the methodology of the entity-relationship model was executed. (MER) to define a complete database design

Survey Responses

1. Notify general meetings
2. Pay for administration services, internal requests (groceries, parking, common areas)
3. Parking assignment
4. A button which facilitates communication with the administration
5. Registration of residents, with guests who may arrive occasionally
6. Pay for administration services, sending PQRs and information on whether a package is found at the gate
7. Pay for administration services, separate the event hall or soccer field
8. List the blocks of those eligible to file complaints and pay for services and administration
9. Reserve common spaces (terrace, pool, garden), maintenance request, pipe closure request (for maintenance or reason x), request for a neighborhood meeting (to discuss important issues).
10. Community news, requests and suggestions, user support
11. Have the possibility of paying the administration and at the time of receiving the visit, be able to reserve the parking spaces for visitors in advance or check if they are free. Be able to review the administration payment statuses. If the residential complex has any court or public space, it can be reserved and also the administration event room
12. Logistics and information about parking, payment of services, information about the residential complex as if they were global communications, site of interests

13. Pay for administration services, be able to see resources and a list of expenses, group information, complaints box and establish communication with the administrator

14. That you have to pay for administration services

15. Pay for management services and an option to report problems with neighbors

Taking into account the previous responses, the main needs of the respondents were discussed and a list of User Stories was made following the format shown below:

User stories

As a <**role**>, i want <**action**>, so what <**impact**>

1. As a resident, I want to receive notifications about general meetings, so I am informed of important topics that will be discussed.

2. As a resident, I want to be able to pay for administration services and make internal requests (groceries, parking, common areas), to streamline administrative procedures and keep my obligations up to date.

3. As a resident, I want a button that makes it easy to communicate with the administration, to resolve questions or problems quickly.

4. As a resident, I want to register my occasional guests, so that they can enter the residential complex without complications.

5. As a resident, I want to send PQRs to resolve any issues that arise in the community.

6. As a resident, I want to reserve common spaces such as the events room or the soccer field, to be able to organize family or sporting events without any setbacks.

After defining the needs of the “client”, the necessary components for the database were defined.

Database Design - Entity Relationship Model (MER)

Definitive components - Step 1 MER

1. Reservation and assignment of common spaces (terraces, auditoriums, swimming pools, parks, parking lots)
2. Payment of administration services
3. List apartment blocks
4. Registration of residents and common guests
5. Sending PQRs (includes news)

Having defined the components, the discussion began to find the entities that would be part of the database.

Define entities - Step 2 MER

1. **Residents**
2. **Apartments**
3. **Common Spaces**
4. **PQR (buzon)**
5. **Guests**
6. **Personal**
7. **Vehicle**
8. **Block**
9. **Time**

From here the respective attributes were defined for each of them.

Define Attributes -Step 3 MER

1. **Residents**: Name, ID-pk
2. **Apartments**: Number, number-of-people, minimum-amount, extra-costs
3. **Spaces** Common: Name, id, reserved
4. **PQR** (mailbox): id, message, date
5. **Guests**: name, ID, visit-time
6. **Personal**: name, ID, role
7. **Vehicle**: license plate, type, length of stay, excess charge
8. **Block**: number, number-of-apartments
9. **Time**: date, start-time, end-time, beneficiary

To have more clarity about the relationships between each of the entities, a table was built in which it is marked with a green box if the two entities in question have any

relationship, and it was complemented with a phrase with which one could be sure. What type of relationship do they present at first?

Relationship Phrases - Step 4 MER

	Resident	Apartments	Common Spaces	RCC	Guests	Personal	Vehicle	Block	Time
Resident									
Apartments									
Common Spaces									
RCC									
Guests									
Personal									
Vehicle									
Block									
Time									

1. **Resident - Apartments**:

   A resident can have multiple apartments, but an apartment only belongs to one resident.

2. **Resident - Common Spaces**:

   One resident may reserve multiple common spaces, but a common space may be reserved by only one resident at a time.

3. **Resident - RCC**:

   A resident can submit multiple PQRs, and a PQR can be submitted by a single resident.

4. **Resident - Guests**:

   A resident can host multiple guests, and a guest can be associated with multiple residents.

5. **Resident - Personal**:

   A resident may interact with multiple staff members, and a staff member may be associated with multiple residents.

6. **Resident - Vehicle**:

   A resident can have multiple vehicles registered, but a vehicle can only be registered to one resident.

7. **Apartments - guests**

An apartment is associated with multiple guests, and a guest can be associated with multiple apartments

8. **Apartments - Block**:

   A block can contain multiple apartments, but an apartment belongs to a single block.

9. **Common Spaces - Schedule**:

   A common space can have multiple times available, and a time can be associated with multiple common spaces.

10. **RCC - Guests**

    A PQR can be sent by a guest, and a guest can send multiple PQRs

11. **RCC - Personal**

    One PQR can be answered by staff, but staff can answer multiple PQRs

12. **Guest - vehicle**

    One guest can register multiple vehicles, and one vehicle can be registered by a single guest

13. **Guest - schedule**

    One guest can be assigned to one time slot, but one time slot can be assigned to multiple guests.

14. **Vehicle - Schedule**:

    A vehicle is assigned to a schedule, but a Schedule is assigned to many vehicles

With the relationships already well defined, it was established what type of relationship each of them was.

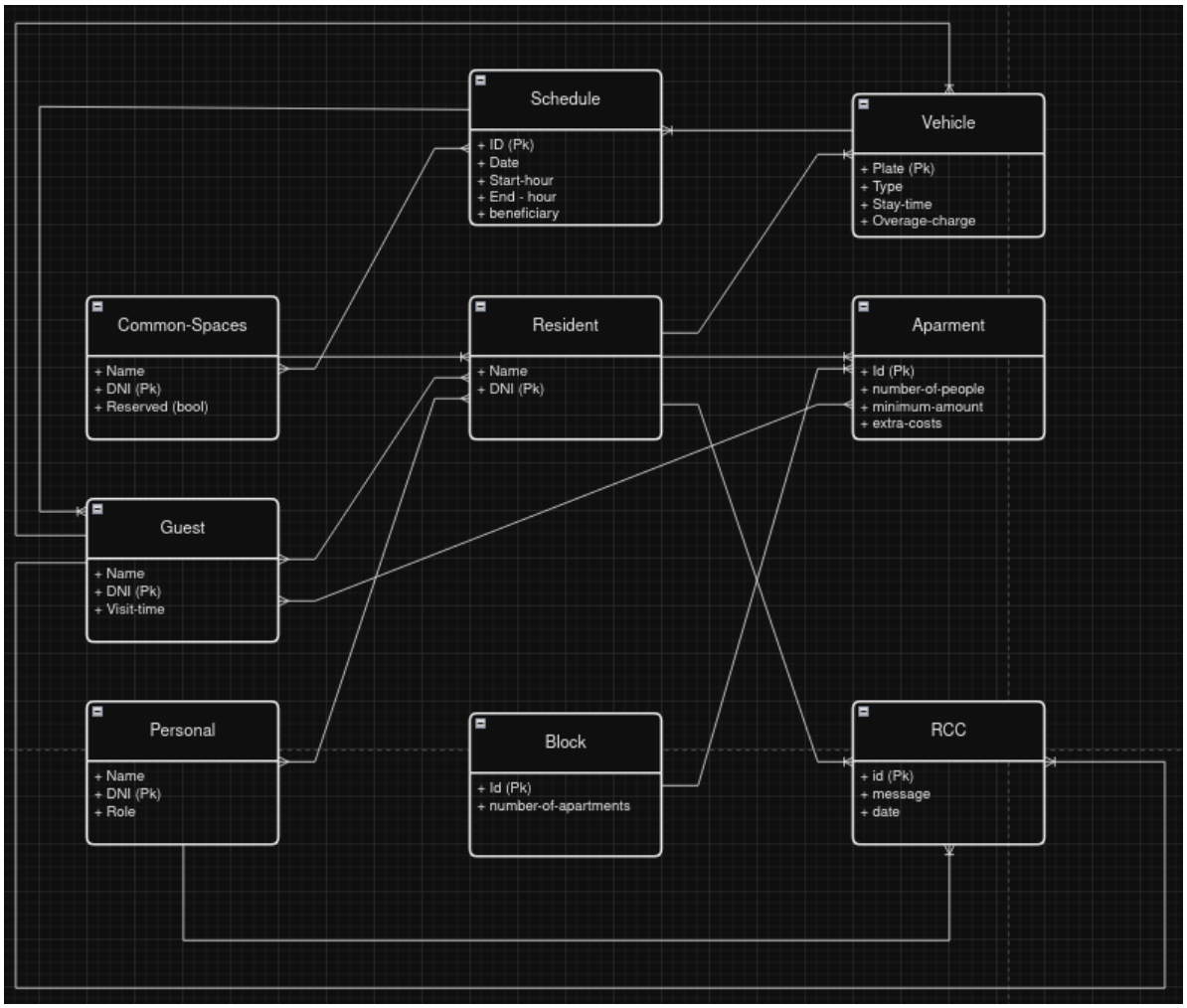
Define relationship types - Step 5 MER

1. **Resident - Apartments**:
   Relationship: One to many (1 -> *)
2. **Resident - Common Spaces**:
   Relationship: One to many (1 -> *)

3. **Resident - RCC**:
   Relationship: One to many (1 -> *)
4. **Resident - Guests**:
   Relationship: Many to many (* -> *)
5. **Resident - Personal**:
   Relationship: Many to many (* -> *)
6. **Resident - Vehicle**:
   Relationship: One to many (1 -> *)
7. **Apartments - Guests**:
   Relationship: Many to many (* -> *)
8. **Apartments - Block**:
   Relationship: One to many (1 -> *)
9. **Common Spaces - Schedule**:
   Relationship: Many to many (* -> *)
10. **RCC - Guests**:
    Relationship: One to many (1 -> *)
11. **RCC - Personal**:
    Relationship: One to many (1 -> *)
12. **Guest - Vehicle**:
    Relationship: One to many (1 -> *)
13. **Guest - Schedule**:.
    Relationship: Many to many (* -> *)
14. **Vehicle - Schedule**:
    Relationship: One to many (1 -> *)

And with the latter we would be ready to make the first entity relationship diagram, although with several missing details.

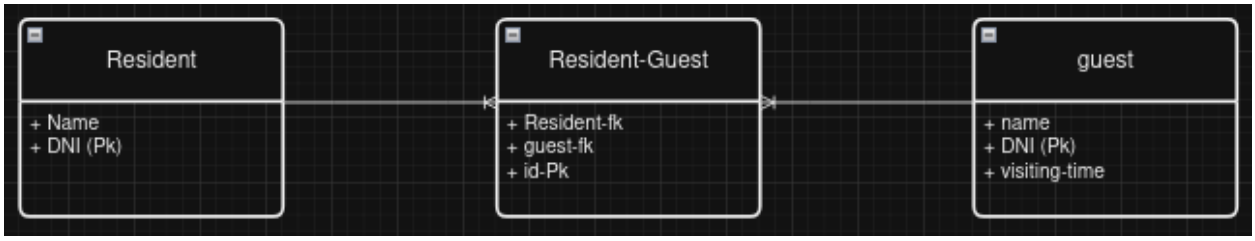
First diagram Entity - Relationship - Step 6 MER (Subject to change)



Split many-to-many relationships - Step 7 MER

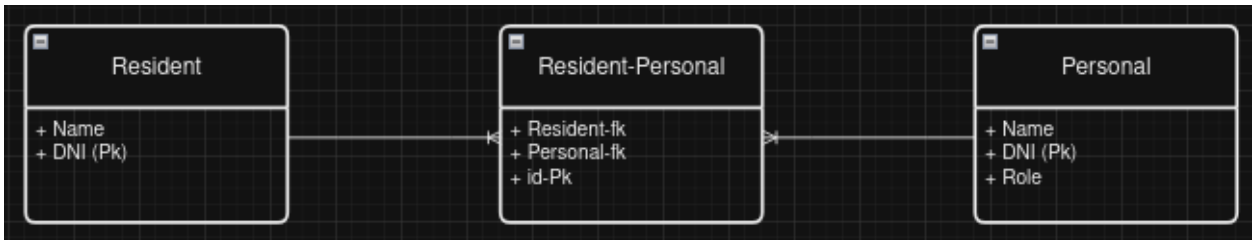
We have 4 relations that feature a many-to-many type (* -> *):

1. **Resident - Guests**:
   Relationship: Many to many (* -> *)



2. **Resident - Personal**:

   Relationship: Many to many (* -> *)

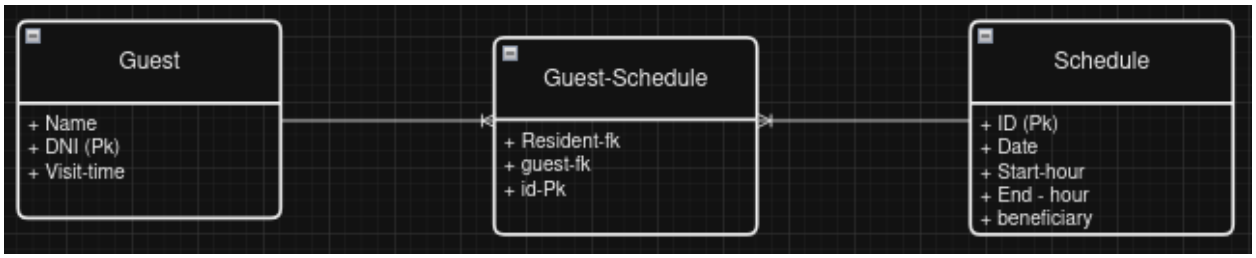


3. **Apartments - Guests**:

   Relationship: Many to many (* -> *)

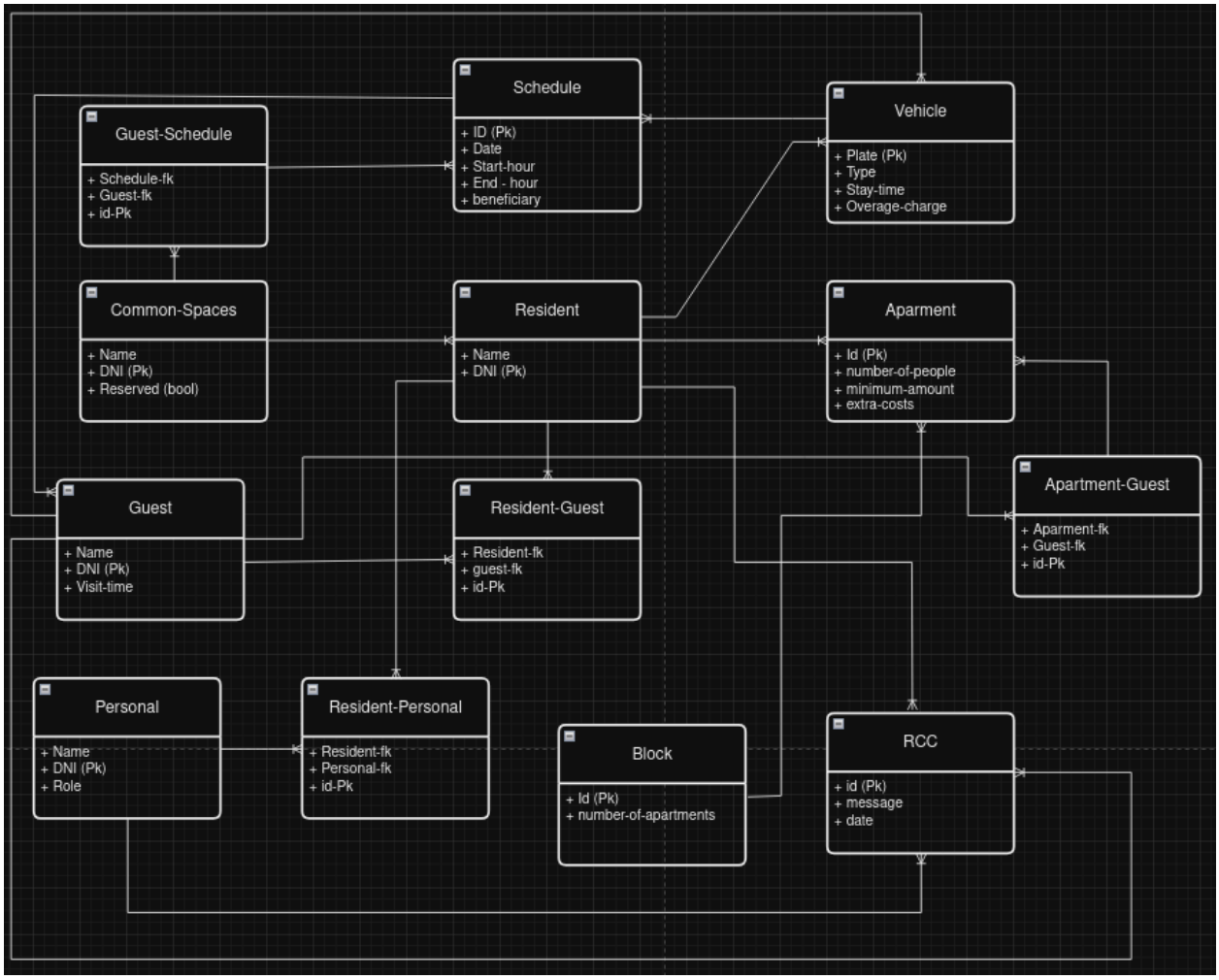
4. **Guests - Schedule**:.

   Relationship: Many to many (* -> *)



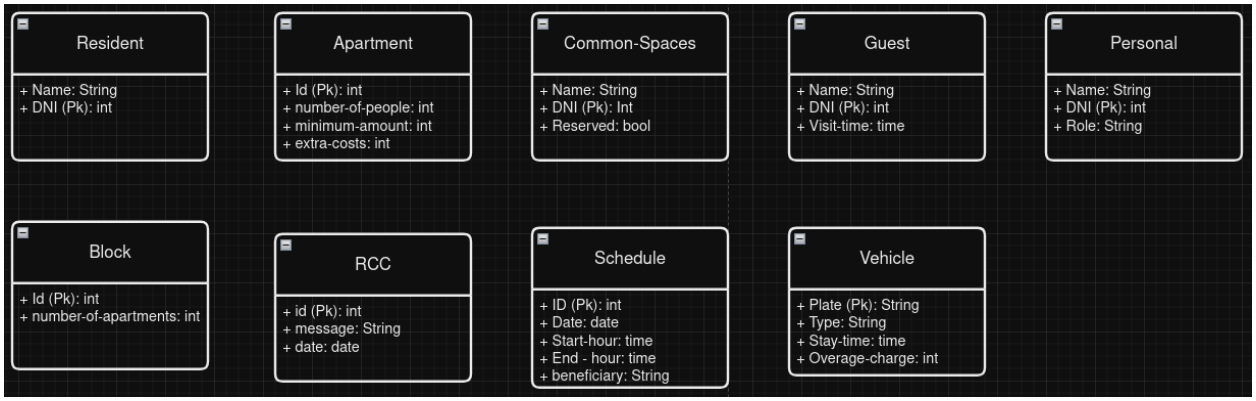
Second Entity-Relationship Draw - Step 8 MER

After dividing the many-to-many relationships, the entity-relationship diagram was redone.

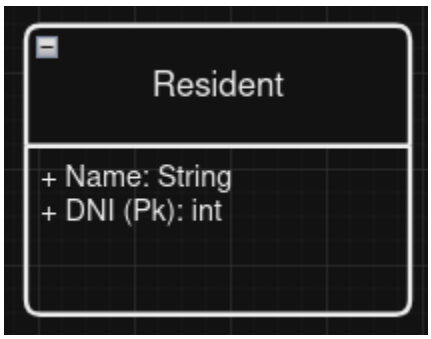


Getting Data Structure - Step 9 MER

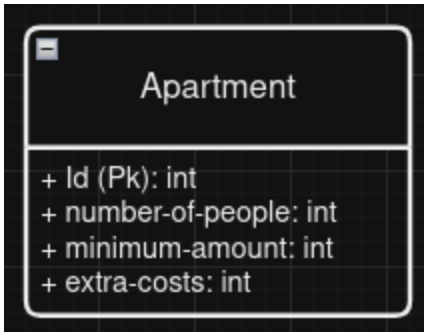
There is little left to finish, so we proceed to define the data structure



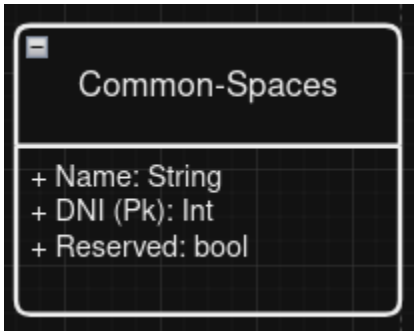
Define Constraints and Properties of Data - Step 10 MER



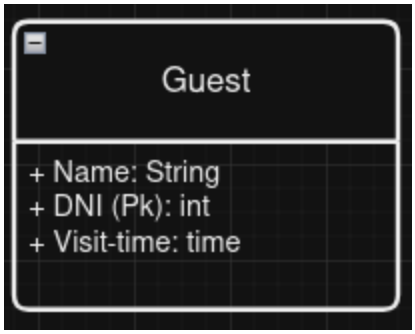
1. **Resident**
    - **Name: varchar(50) Not Null**
    - **DNI: Not Null, unique**



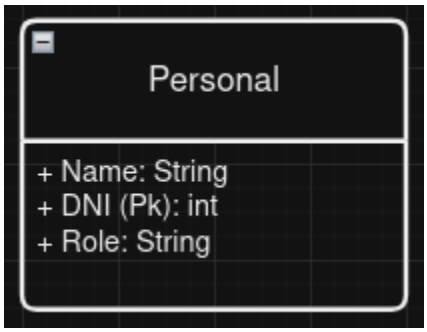
2. **Apartment**
    - **Id: Unique, Not Null, unique, autoIncremental**
    - **Number-of-people: Not Null**
    - **Minimum-amount: Not Null**
    - **Extra-cost: Null**



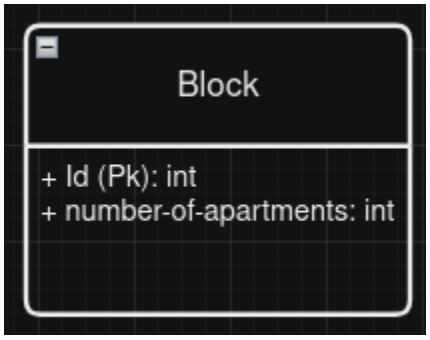
3. **Common-Spaces**
    - **Name: varchar(50), Not Null**
    - **DNI: unique, Not Null, autoIncremental**
    - **Reserved: Not Null, DEFAULT false**



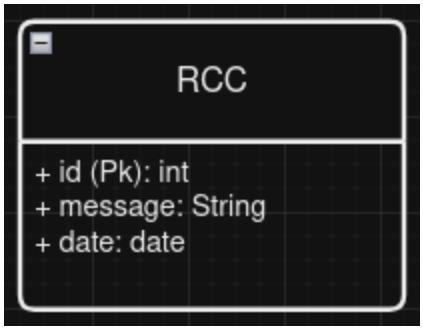
4. **Guest**
    - **Name: varchar(50), Not Null**
    - **DNI: Not Null, Unique**
    - **Visit-time: Null**



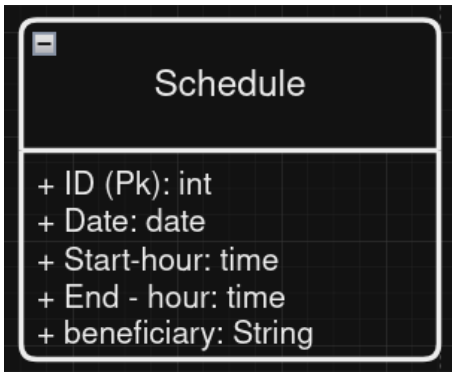
5. **Personal**
    - **Name: varchar(50), Not Null**
    - **DNI: Not Null, Unique**
    - **Role: varchar(25), Not Null**



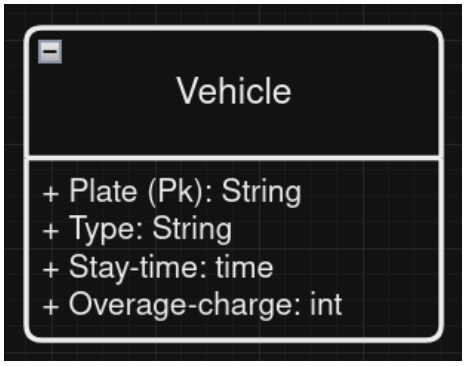
6. **Block**
    - **Id: Not Null, Unique, autoIncremental**
    - **Number-of-apartments: Not Null**



7. **RCC**
    - **Id: Unique, Not Null, autoIncremental**
    - **Message: text (without Limit), Not Null**
    - **Date: Not Null**

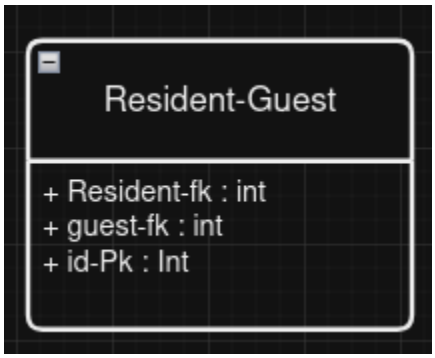


8. **Schedule**
    - **Id: unique, Not Null, autoIncremental**
    - **Date: Not Null**
    - **Start-hour: Not Null**
    - **Beneficiary: varchar(50), Not Null**

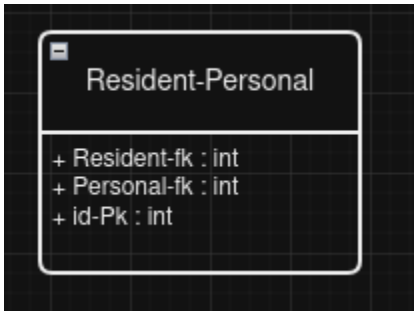


9. **Vehicle**
    - **Plate: varchar(10), unique, Not Null**

- **Type: varchar(15), Not Null**
- **Stay-time: Not Null, DEFAULT 02:00:00.0**
- **Overage-charge: Not Null**

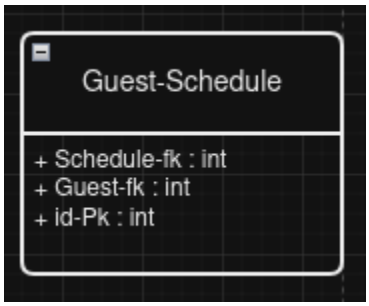


10. **Resident-Guest**
    - **Resident: Not null**
    - **Guest: Not null**
    - **ID : unique, auto increment**



11. **Resident-Personal**
    - **Resident: Not null**
    - **Personal: Not null**
    - **ID : unique, auto increment**

12. **Apartment - Guest**
    - **Apartment: Not null**
    - **Guest: Not null**
    - **ID : unique, auto increment**



13. **Guest - Schedule**
    - **Schedule: Not null**
    - **Guest: Not null**
    - **ID : unique, auto increment**

Link to DRAW.IO:
https://drive.google.com/file/d/1EU8IDxSMWjk6R5YTAUT18K_nAsIa9ckV/view?usp=sharing

Conclusions

Through this report we can see the ease and naturalness that is presented when designing a database with the Entity Relationship Model (ERM), all the requirements requested by the users were able to be completed.