Brief description of the API design:

This API enables easy automatic management of orders, track of them and updates of the restaurant menu. Also, the API is flexible to future updates and improvements to implement in an easy and fast way.

The actual API enables the creation of basic client profiles, management of different client addresses, history of commands and a skeleton to client recommendations function.

Also, it provides to the restaurant a structure easy to maintain, scale and adapt to future changes. The API enables post dishes, menu, orders track update and notification, a queue of orders, save statistic data, etc.

The project has been developed on python over Django Rest Framework. The database structure is in SQL (SQLite), it has been used this database because is simple and enables a fast to deploy environment when it's cloned temporary to another pc. If it would deploy as a permanent service it would change the database, for example, to PostgreSQL or even to MongoDB (NoSQL) without need to change the code, just adapting the settings connection.

Questions

- What are the elements of your system?

The system has three principal node elements: Client, Dish and Order.
Diagram 1, shows the API's database diagram.

The Client element stores personal, contact and addresses info related to the client. A client can have different addresses assigned to them.
Database table related: Client and ClientAddress.

The Dish element stores the information of the dishes from the restaurant, its stores the info from it cooking steps, ingredients, type of dish, expected time and availability (the plate can be promotional).
Database table related: Dish, DishIngredients, DishAvailable, DishSteps, DishType, DishTemperature, Ingredients and TimeUnits.

The Order element stores the input commands from clients and the relation info with the plates on them. This element interacts with the data of the Client and Dish.
Database table related: Order, OrderDishes and OrderTrack.

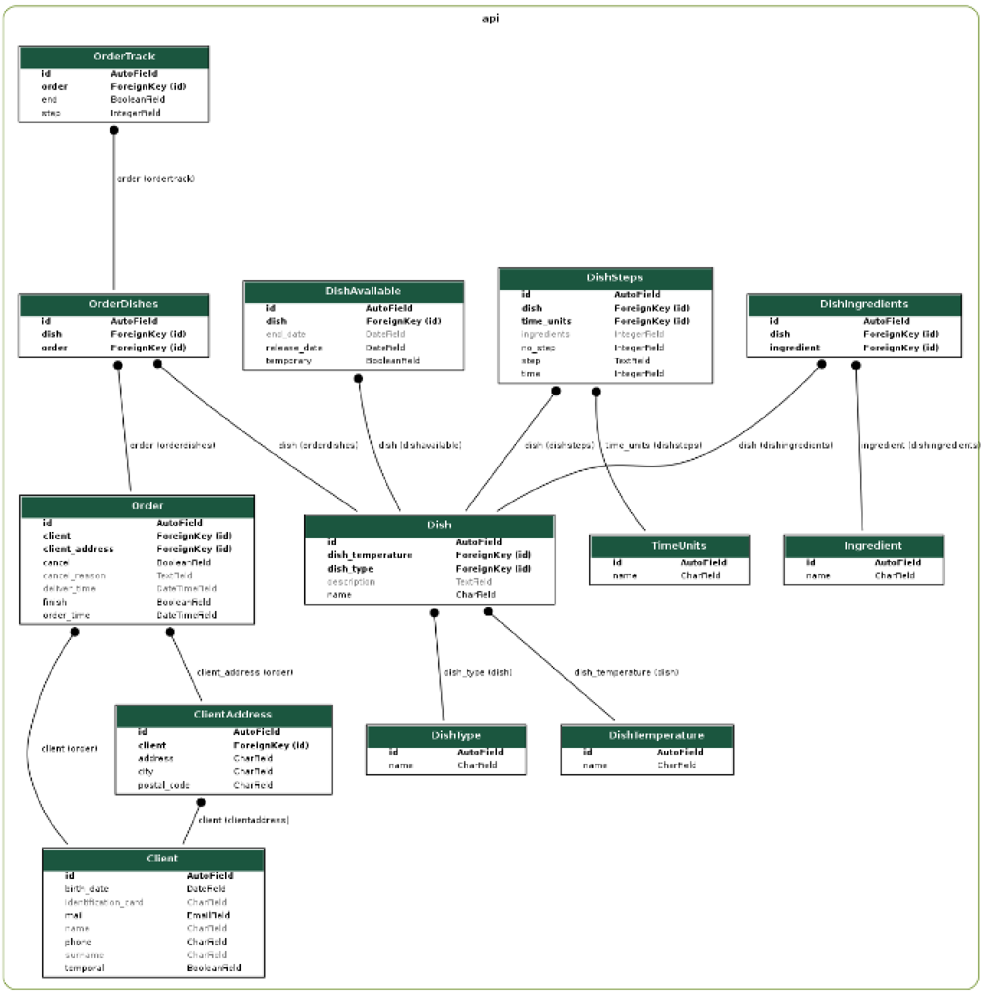
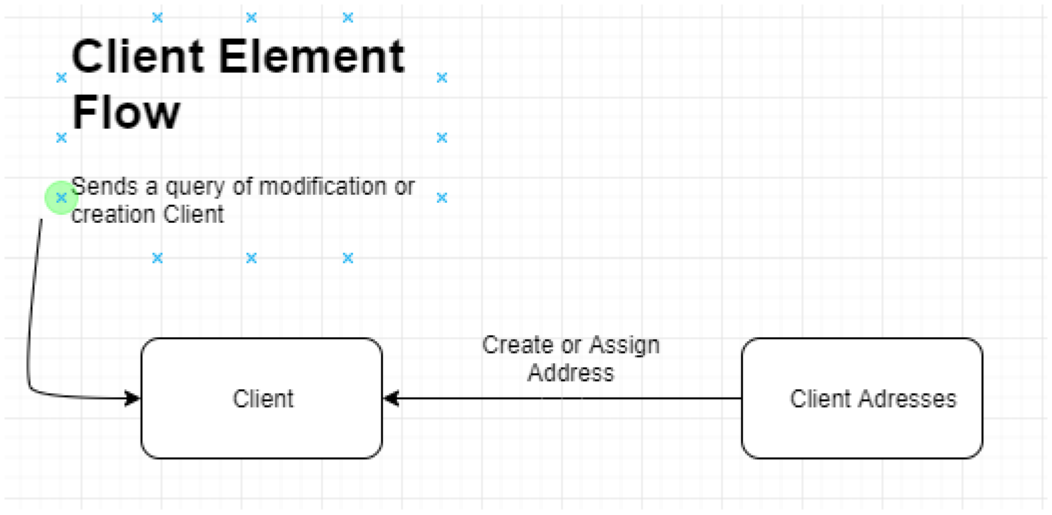


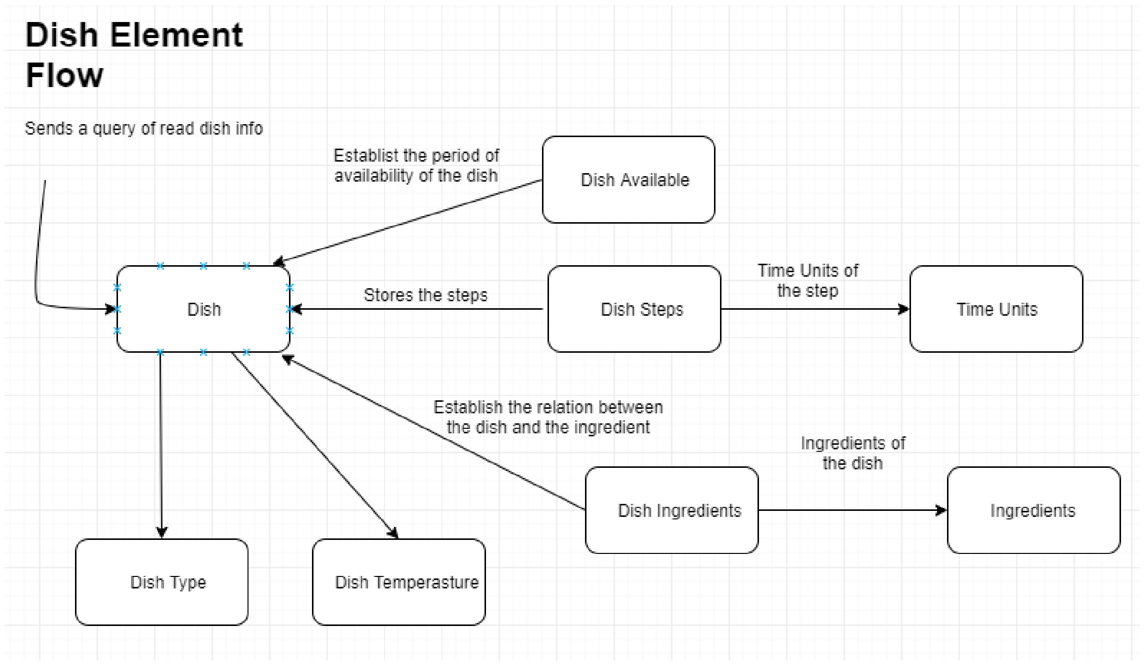
Diagram 1: Design of the database

- What are the interactions between those elements?

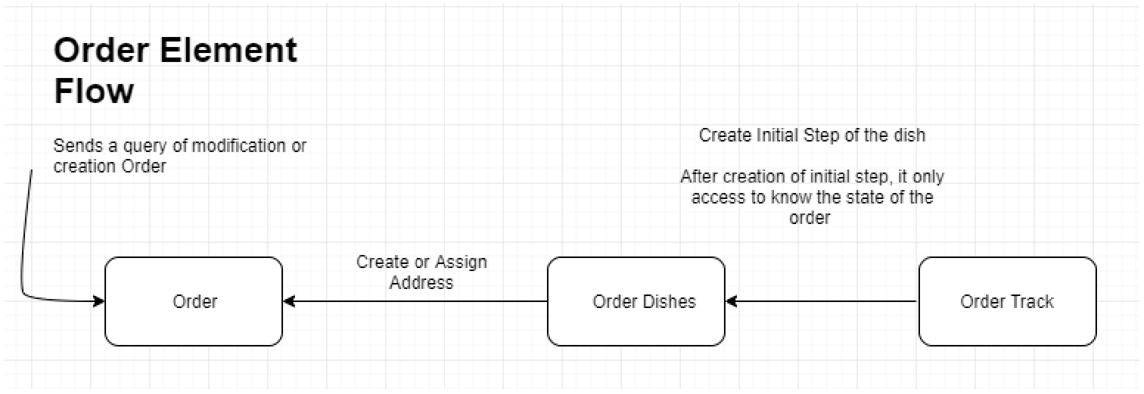
Client elements flow



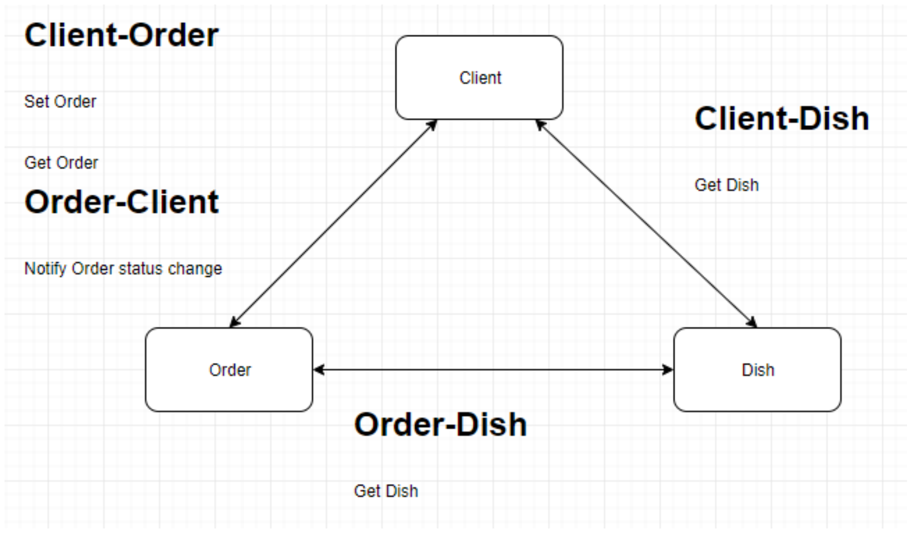
Dish elements flow



Order elements flow


Order Element Flow
Sends a query of modification or creation Order
Create Initial Step of the dish
After creation of initial step, it only access to know the state of the order
Create or Assign Address
Order
Order Dishes
Order Track

Interaction between the elements



- What happens if an element of the system fails?

If an element of the system fails:

- It would create or update an entry in calls when is possible, for example at assigning a new ingredient to a dish.

- Notify an error code to the admin, for example, the order doesn't exist.

- Why is it easy to scale the system with regards to multiple restaurants?

It’s easy to scale because with some minor changes to the actual models, like add a field "restaurant" to the database table "order" and it in the queries it will work without errors. Also, it can pass to a distributed database in order to improve the performance at queries to the database.