SW ARCHITECTURES PROJECT DOCUMENTATION

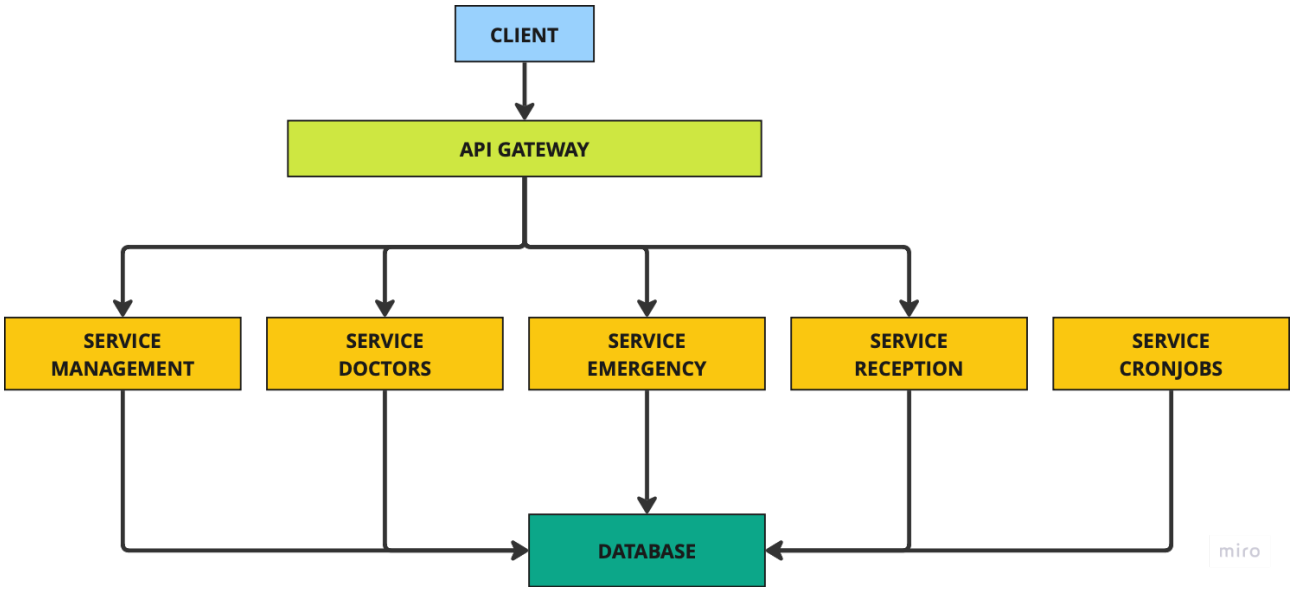
STUDENTE	EMAIL	MATRICOLA
Stefano Zogno	1001900@stud.unive.it	1001900
Alessandro Dussin	881424@stud.unive.it	881424
Lukas Hrmo	1001537@stud.unive.it	1001537
Fedor Kazin	908385@stud.unive.it	908385
Lorenzo Facchin	903942@stud.unive.it	903942

ARCHITECTURES

The software is a modern web application, developed with most popular and modern frameworks. The architecture used for this project is a service one, divided in the following parts:

1. **Client**: the user interface of the application, developed in Angular 17.
2. **API Gateway**: it is a sort of mediator between client and services. Each client requests are sent to it, which redirects to the correct service based on the URL and then it gets response and return to the client. In addition, the API Gateway for each request which requires an authenticated user, verify if the given JWT Token is still valid, interacting with management service. Thanks to the API Gateway, we reduce the coupling in the architectures, since we have just one entrance point.
3. **Services**:
    a. MANAGEMENT: it is used for everything concerns the session and user management. It provided API for create, edit, enable/disable user, login, logout and password changing. The source code is developed using PHP 8.3, Symfony 6.4 and Doctrine ORM 3.2.
    b. DOCTOR: it is used for everything concerns patients, doctor and nurse side. In particular, it provides API for managing medical procedures (create, edit, delete and list), patient vitals (create, edit, delete and list) and assigning patient to a doctor. The source code is developed using Java X.Y and Spring Boot X.Y.
    c. EMERGENCY: it is used for managing the emergency, in particular it provides API for managing hospital beds, emergency visits, patient vitals and invoices.
    d. RECEPTION: it is used by secretaries and provides API for getting information about patients. The source code is developed using C# .net8.0 – ASPNet framework for API and EntityCore framework for ORM.
    e. CRONJOB: this service is a set of Python X.Y scripts for report generation and database backup.
4. **Database**: where data of the application are stored. Each service interacts with it, since it's the same for each one. For it, we chose Postgres 16.

The following picture summarize the architectures of the application.


CLIENT
API GATEWAY
SERVICE MANAGEMENT
SERVICE DOCTORS
SERVICE EMERGENCY
SERVICE RECEPTION
SERVICE CRONJOBS
DATABASE
miro

HOW TO RUN THE APPLICATION

To run the application, you need to follow these steps:

1. DATABASE:
    a. Add a .env file in the root of the folder of this repository with the following rows:
        i. POSTGRES_USER=root
        ii. POSTGRES_PASSWORD=root
    b. Launch “docker-compose up –build -d”
    c. Wait until it finishes, since it also creates the net_storage.
2. MANAGEMENT:
    a. Launch “docker-compose up –build -d”
    b. When it finishes, launch the following command:
        i. docker exec --workdir /var/www/html service-management composer install --no-interaction --optimize-autoloader
3. EMERGENCY:
    a. Launch “docker-compose up –build -d”
4. DOCTORS:
    a. Launch “docker-compose up –build -d”
5. RECEPTION:
    a. Launch “docker-compose up –build -d”
6. CRONJOBS:
    a. Launch “docker-compose up –build -d”
7. CLIENT:
    a. WORK IN PROGRESS