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

This document contains information about the project of software architectures.

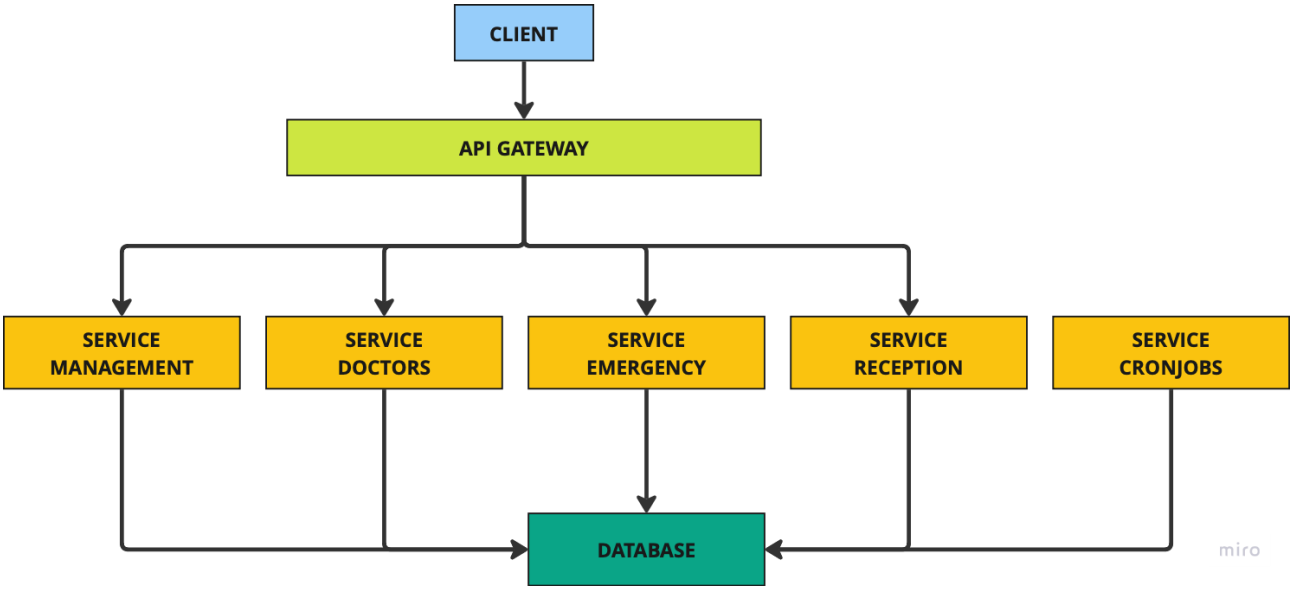
Follow the steps inside the section “How to run the application” in order to build correctly the application and its architecture.

ARCHITECTURES

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

1. **Client**: the user interface of the application, developed in Angular 17.
2. **API Gateway**: It acts as a mediator between clients and services. Each client request is sent to it, and it redirects the request to the correct service based on the URL. Once the service processes the request, the API Gateway receives the response and returns it to the client. Additionally, for requests requiring an authenticated user, the API Gateway verifies whether the provided JWT token is still valid by interacting with the management service. Thanks to the API Gateway, we reduce coupling in the architecture by providing a single entry point.
3. **Services**:
    a. MANAGEMENT: This component handles all aspects of session and user management. It provides APIs for creating, editing, enabling/disabling users, as well as functionalities for login, logout, and password changes. The source code is built using **PHP 8.3**, **Symfony 6.4**, and **Doctrine ORM 3.2**.
    b. DOCTOR: It is used for everything related to patients, doctors, and nurses. Specifically, it provides APIs for managing medical procedures (create, edit, delete, and list), patient vitals (create, edit, delete, and list), and assigning patients to doctors. The source code is developed using **Java 17** and **Spring Boot 3.4.0**.
    c. EMERGENCY: This component is used for managing emergencies. Specifically, it provides APIs for handling hospital beds, emergency visits, patient vitals, and invoices. The source code is developed using **Java 17** and **Spring Boot 3.4.0**.
    d. RECEPTION: This component is used by secretaries and provides APIs for retrieving patient information. The source code is developed using **C# .NET 8.0 – ASP.NET Core** for the API and **Entity Framework Core** for ORM.
    e. CRONJOB: This service consists of a set of **Python 3.12** scripts designed for report generation and database backups.
4. **Database**: The application's data is stored in a **PostgreSQL 16** database. All services interact with this database, as it is shared across the entire system.

The following diagram summarizes the architecture 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. If you are on Windows, remove from docker-compose.yml the following row: “gateway” -> “platform” -> “linux/x86_64”. We had to add it in order to run on Mac OS with Apple Chip.
    b. Launch “docker-compose up –build -d”
    c. When it finishes, launch the following command:
        i. docker exec --workdir /var/www/html service-management composer install --no-interaction --optimize-autoloader
3. API:
    a. If you are on Windows, remove from docker-compose.yml the following row: “gateway” -> “platform” -> “linux/x86_64”. We had to add it in order to run on Mac OS with Apple Chip.
    b. Launch “docker-compose up –build -d”
4. EMERGENCY:
    a. If you are on Windows, remove from docker-compose.yml the following row: “gateway” -> “platform” -> “linux/x86_64”. We had to add it in order to run on Mac OS with Apple Chip.
    b. Launch “docker-compose up –build -d”
5. DOCTORS:
    a. Launch “docker-compose up –build -d”
6. RECEPTION:
    a. Launch “docker-compose up –build -d”
7. CRONJOBS:
    a. Launch “docker-compose up –build -d”
8. CLIENT:
    a. Launch “docker-compose up –build -d”

TODO: When everything has been launched, you can access the application at http://localhost:4200.

CREDENTIALS

The database has been seeded with many users of each type. In particular, you can access with the following credentials:

- ADMIN:
  - Username: admin
  - Password: P4ssword1@
- DOCTOR:
  - Username: doctor1
  - Password: P4ssword1@

  NB: If you want another doctor, just change 1 with 2, 3, 4, 5 and so on
- NURSE:
  - Username: nurse1
  - Password: P4ssword1@

  NB: If you want another nurse, just change 1 with 2, 3, 4, 5 and so on
- SECRETARY:
  - Username: secretary1
  - Password: P4ssword1@

  NB: If you want another secretary, just change 1 with 2, 3, 4, 5 and so on
- PATIENT:
  - Username: patient1
  - Password: P4ssword1@

  NB: If you want another patient, just change 1 with 2, 3, 4, 5 and so on

Each time you create a user, it has **P4ssword1@** as default password. After the first login of a new user, it needs to change the password.

STEPS TO TRY APPLICATION

1. ADMIN:
    a. Login with an admin user: **admin P4ssword1@**
    b. Click the various buttons to show the list of users, filtered by type.
    c. Try to create/edit users of each type.
    d. When you create a user, try to login with it and default password **P4ssword1@** the app should ask you to change the password.
2. PATIENT:
    a. Login with a patient user (ie: patient1)
3. DOCTOR:
    a. Login with a doctor user (ie: doctor1)
    b. Click Medical History button on the first row and you will be redirect to patient detail with list of procedures and vitals signs
    c. Click to Add New Procedure
    d. Click to Add New Vital (body temperature should be valid, under 40 degrees)
4. NURSE:
    a. Login with a nurse user (ie: nurse1)
    b. Free bed that has status OCCUPIED
    c. Click Assign button on the second row and select a Patient and Assign Bed
    d. Click Assign bed to patient button and Assign Bed
    e. Go to Billing Invoice by clicking item in the navbar
    f. Click detail button and check the Payment Received to update payment
    g. Create billing invoice if there's a patient with no billing invoice yet and has status discharged
5. SECRETARY:
    a. Login with a nurse secretary (ie: secretary1)
    b. Try to create a register patient;
    c. Show the detail of a patient and try to edit it;
    d. Change the password of a patient;
    e. Try to edit the triage form of some patients.
    f. Click on Billing invoice in the top menu and try to add a new or edit an existing one.

LINK TO GITHUB: https://github.com/orgs/Hospital-CM0639-1/repositories

LINK TO GITHUB README: https://github.com/Hospital-CM0639-1/.github