CS 319 - Object Oriented Software Engineering

Instructor: Eray Tüzün, TA: Muhammad Umair Ahmed & Elgun Jabrayilzade

BilHealth

Final Report

Implementation

Mehmet Alper Çetin
21902324

Vedat Eren Arıcan
22002643

Uygar Onat Erol
21901908

Recep Uysal
21803637

Efe Erkan
21902248

May 3, 2022

Contents

1 Introduction

Our project is a health center management software system, built in the form of a web application. The main goal is to provide online, remote attention to patients to increase productivity for all parties involved. The primary method of interaction is through *case*s, which contain all relevant information for a given medical situation. Patients use the system to open cases and request appointments, and the health center staff acts on these requests to provide medical services. These interactions also function as a variety of medical records for the health center.

1.1 Current Implementation State

2 Lessons Learned

3 Build & Execution Instructions

There are two build flavors in which the application can run: development and production. The production build is more stable, but also requires more initial effort to run. Both builds are highly recommended to be run through **Docker**.

The project runs best on a Linux-based environment, despite using Docker. Through WSL and its integration with Docker, Windows is also able to run the project reasonably well. Note that the project source must be located within the WSL filesystem. In some instances, Docker causes WSL to consume unnecessarily high amounts of RAM. This can be solved through the WSL configuration as seen here.

For both of the build flavors, the prerequisites are that:

- you obtain the project source through tools such as *git clone*.
- you have installed Docker and docker-compose.

Then, begin by changing your working directory to the root of the project directory.

For the most up to date version of these instructions, see the README on the project's repository.

3.1 Development Build

1. Build the Docker image:

```
docker-compose -f docker-compose.devel.yml build
```

2. Any time after the first image build, run the application via:

```
docker-compose -f docker-compose.devel.yml up
```

Once the containers are fully up, the project should be visible at `https://localhost:7257/`. This development image mounts the project folder into the container, allowing changes to the code to be seen in real-time.

Note that the HTTPS support is through self-signed certificates, and your web browser may warn you about it. Ideally, your browser should give you the option to add an exception for the certificate. This is completely safe to do.

3.2 Production Build

The production build process is slightly more involved, at least on the first run. You will need to have additionally installed .NET Core SDK v6.0.x and EF Core tools.

Then;

```
# Build image
docker-compose -f docker-compose.prod.yml build

# Perform database migration (do this only on first run or new migration)
## Bring up only the database container
docker-compose -d -f docker-compose.prod.yml up dbpostgres
## Generate idempotent SQL script and copy into container
dotnet ef migrations --idempotent -o migrate.sql
docker cp migrate.sql bilhealth_db_postgres_1:/migrate.sql
## Enter the database environment and execute migration
docker exec -it bilhealth_dbpostgres_1 bash
cd / && psql -U postgres -d bilhealthprod -f migrate.sql && exit
## Bring down database
docker-compose -f docker-compose.prod.yml down

# Bring up all containers
docker-compose -d -f docker-compose.prod.yml up
```

Once the containers are fully up, the project should be visible at `http://localhost:5000/`. Note that unlike the development image, this one may need to be rebuilt upon every change to the project. The database migration process needs to be done only if the database is not up to date with the latest migration.

There is currently no support for HTTPS in the production build. If desired, a reverse proxy (such as Nginx) may be used to support HTTPS.

4 User's Guide

5 Work Allocation

- **Mehmet Alper Çetin:**
- **Vedat Eren Arıcan:** Interview with the Bilkent University health center. Implementation of the backend and frontend code bases. DevOps tasks such as integrating Docker and a GitHub CI workflow. Writing textual report content through LaTeX. Access matrix and deployment diagram.
- **Uygar Onat Erol:**
- **Recep Uysal:**
- **Efe Erkan:**