README.md

Medical-Test-Records

Introduction

A Health care secure system and network that allows secure and confidential access to patient medical records of certain healthcare organizations. It also demonstrates the secure interconnection (sending and receiving of medical records) of partner healthcare organizations. In such manner, a patient will be able to access his medical records of every healthcare organization where he is registered.

Problem

- Health care institutions gather and store sensitive information from patients such as medical test records;
- Patients can not always access their medical records in a secure manner;
- Information systems do not always ensure fine-grained and contextualized access to medical records for relevant staff;
- Lack of connection and access to medical records of different partner institutions, with different infrastructures.

Solution

- Provide a secure system to allow safe and confidential access to patient's medical records;
- Ensure authentication and access control to certain resources for authorized personnel;
- Ensure secure interconnection (sending and receiving medical records) between partner health organizations.

Built With

- Java - Programming Language;
- Maven - Build Tool and Dependency Management;
- Spring Boot - Create Java stand-alone Spring applications;
- Spring Boot Security - A highly customizable authentication and access-control framework;

- Mongo DB Atlas - Cloud Database;
- Ubuntu Server - Virtual Machines to run the Servers;
- UFW - Uncomplicated Firewall - Firewall.

Getting Started

These instructions will get you a copy of the project up and running on your local machine for development and testing purposes. See deployment for notes on how to deploy the project on a live system.

Prerequisites

If testing via vms:

- VirtualBox installed on your machine. Install here VirtualBox

if testing via jar files:

- Install here java 17 (*)
- Install here Maven and select the version "apache-maven-3.8.4-bin.tar.gz",

(*) for Macbook M1 select "macOS/AArch64"

Setup

VM's

To prepare the environment it is necessary to import the vms. Firstly a new NAT Network must be created in order to successfully setup the network interfaces.

To create a Nat Network go to:

- **File > preferences > Network**.
- Click on the green icon to add a new Nat Network.
- The name should be "**rede**", Network CIDR: **10.0.0.0/24** and **DHCP** should be **enabled**.

With that done we can procede to import the vms.

In order to import the vm you should go to:

- **Import**, Choose the correct .ova file and import it into virtualbox.

After setting up the network, by choosing the created network interface the vm can be turned on and changing MACs. Finally By starting the vm, the code will be running and the server will start without any help.

To access the frontend for both laboratory and hospital, the vm patient should be started.

The login is **patient** and password is **patient**.

After logging in, when after opening the browser it is possible to see two bookmarks:

For hospital:

```
https://10.0.0.4:8443/
```

For laboratory

```
https://10.0.0.104:8443/
```

The **credentials for each vm** are the the same as the **hostname**, this is, if hostname is hosf, then **username** is hosf and **password** is hosf. The **root crentials** are root:root

VM's List of IP's

Hospital Frontend 10.0.0.4

Hospital Backend 10.0.0.5

Laboratory Frontend 10.0.0.104

Laboratory Backend 10.0.0.105

Hospital Backend Server 10.0.0.6

Note: Links to vm's can be provided if needed

Setup in your local machine

First of all, maven and java are required. So let's install them

Maven Instalation

We're gonna use maven. The folder is inclued in our project, so ignore this step.

But there are the instructions:

Installing **maven**, compatible with java 17:

0. Please click here to download maven;
1. Download Binary tar.gz archive: `apache-maven-3.8.4-bin.tar.gz`;
2. Unpack the archive with tar/unzip;

Java 17 Instalation

LINUX

To run the code in a local linux system java 17 and maven must be installed.

Installation of **java 17**:

```
sudo apt install openjdk-17-jre
```

```
sudo apt install openjdk-17-jdk
```

Change java version to the newer one:

```
sudo update-alternatives --config java
```

MAC OS

In order to make the correct installation of **java 17** follow the following instructions:

0. Download java 17 from here `NOTE: for Macbook M1 select "macOS/AArch64"`;
1. Now go to your Downloads `cd Downloads/`;
2. Make `sudo mv openjdk-17.0.2_macos-aarch64_bin.tar.gz /Library/Java/JavaVirtualMachines/`;
3. `cd /Library/Java/JavaVirtualMachines/`;
4. `sudo tar -xzf openjdk-17.0.2_macos-aarch64_bin.tar.gz`;
5. `sudo rm openjdk-17.0.2_macos-aarch64_bin.tar.gz`;
6. `export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk-17.0.2.jdk/Contents/Home`.

After the previous installation is complete in your OS, please read the following notes **before** running.

❗ Notes about running the application ❗ Mandatory reading ❗

If you are testing in your local machine, **Do not run all jars at the same time**. Please note, Hospital and Laboratory frontends are running on port 8443, and their respective backends are also running in port 3000. Due to this, it is **impossible** to run both Hospital and Laboratory at the same time. It is still possible to run Hospital Backened server, that is running on port 4000, in order to test the custom protocol.

But if you deploy the project in the vms, you can run all at the same time (there is no issue with localhost used ports).

Summary:

- Hospital Frontend: port 8443
- Hospital Backened: port 3000
- Laboratory Frontend: port 8443
- Laboratory Backened: port 3000
- Hospital Backend Server: port 4000

It's time to run our project:

0. Unzip project;
1. Run the following instructions.

To run the LABORATORY

Go to `Medical-Test-Records/jars` directory and run:

```
java -jar Laboratory-Frontend.jar
```

Go to `Medical-Test-Records/Code/Laboratory-Backend` directory and run:

```
java -jar Laboratory-Backend.jar
```

Go to `Medical-Test-Records/Code/Hospital-Backend-Server` directory and run:

```
apache-maven-3.8.4/bin/mvn spring-boot:run
```

Now open your browser in `https://localhost:8443/` and enjoy our laboratory system.

To run the HOSPITAL

Go to `Medical-Test-Records/jars` directory and run:

```
java -jar Hospital-Frontend.jar
```

```
java -jar Hospital-Backend.jar
```

Now open your browser in `https://localhost:8443/` and enjoy our hospital system.

Citizen card ID and passwords

The credentials to access the hospital and laboratory can be found below.

✓ HOSPITAL

❗ **Note: You can not register in the web site. Only Admin can register personnel. Only Doctor/Nurse can create Medical Records. Only Ward Clerks can register Patients.**

Admin	Citizen Card ID	Password
André Proenza	12345678	Password123!

Doctor	Citizen Card ID	Password
Ana Albuquerque	19573526	Fl4%8!Hd10k4Zc*!

Ward Clerk	Citizen Card ID	Password
Hélder Costa	15378965	V322!!P25KM4&f6b

Patient	Citizen Card ID	Password
Amanda Júlio	17645234	pB0K!*vF85!0&@60

Porter	Citizen Card ID	Password
Henrique Jota	17564920	7u$%4n&B90%8U!6!

Volunteer	Citizen Card ID	Password
Madalena Afonso	17564532	`*Mj0T**p*?Z!yv0u`

Patient_Assistant	Citizen Card ID	Password
Alexandre Pinto	14789078	l&u5lhmt*nO$f?!1

Clinical_Assistant	Citizen Card ID	Password
Mariana Rita	18509738	p$h4M$xcHk10q@

✓ LABORATORY

❗ **Note: You can not register in the web site. Only Admin can register personnel. Only Responsible can create Clinical Records and register Patients.**

Admin	Citizen Card ID	Password
André Proenza	12345678	Password123!

Responsible	Citizen Card ID	Password
Paulo Marques	15288625	y20lH%7pk*1@h0Ou

limitations

- Due to lack of time, we were unable to output a error/sucess message to laboratory frontened html, for the sucess/insucess in putting the medical Record of the laboratory in the Hospital database. The message can be seen in the System.Out of the Laboratory backend.

System Features

- Ensures confidentiality and integrity of medical records;
- Ensures confidentiality and integrity of communications with the web browser;
- Ensure successful authentication of citizens;
- Authenticates citizens in a secure way;
- Ability to change password if citizen is authenticated;
- Authenticated user credentials confirmation sent to email;
- Ensures that only authorized staff and patients have an account;
- Ensures different “roles” have access to different privileges;
- Ensures there is only one account per citizen, using citizen ID card number;
- Prevents access to medical records if the citizen does not have privileges;
- Allows user A to change the privileges of user B, if user A is a system administrator;
- Validates and sanitize form input;
- Uses HTTPS to encrypt communications;
- Stores and manage symmetric and asymmetric keys;
- Defines a restricted set of rules on the firewall;
- Establishes mutual authentication and shares medical records with partner institutions;
- Minimizes the impact of attacks inside the system.

Deployment

For deploymend on a live system you should have access to a mongodb database account and a suficient ammount of ram. The upgrade from iptables into firewall will greatly benefit your network, since with them there comes a better capability to obtain logs and information about the network.

For virtualization any hyphervisor should suffice, you just need to be able to create isolated internal networks and and have access to the internet.

For a more isolated control of any possible vulnerabilities, the use of docker could add an extra layer of protection against attacks to the network, this is, it creates an extra difficulty to obtain higher/relevant credentials on the computar that manages the docker/s.

Regarding the system, there is a necessity to create the first admin manually, since the option of creating new user/admins is not available. This is a great advantage in terms of security, but can become a problem in usability, even if it must only be done once.

The Hospital Backend server is able to accept multiple clients (via threads) therefore it possible to easily escalate the system, create multiple laboratories.

The credentials for each vm are the the same as the hostname, this is, if hostname is **hosf**, then username is **hosf** and password is **hosf**. The root crentials are root:root

Regarding vm and networking, although we had a solution that forwarded all packets from both interfaces via a router, the vms had no internet access, therefore we had to remove the router and put every vm in the same network.

Additional Information

Authors

- **Ana Albuquerque** - ** ist1102209 ** - GitHub
- **André Proença** - ** ist1102327 ** - GitHub
- **Joel Russo** - ** ist1102098 ** - GitHub

Versioning

We used GitHub for versioning.