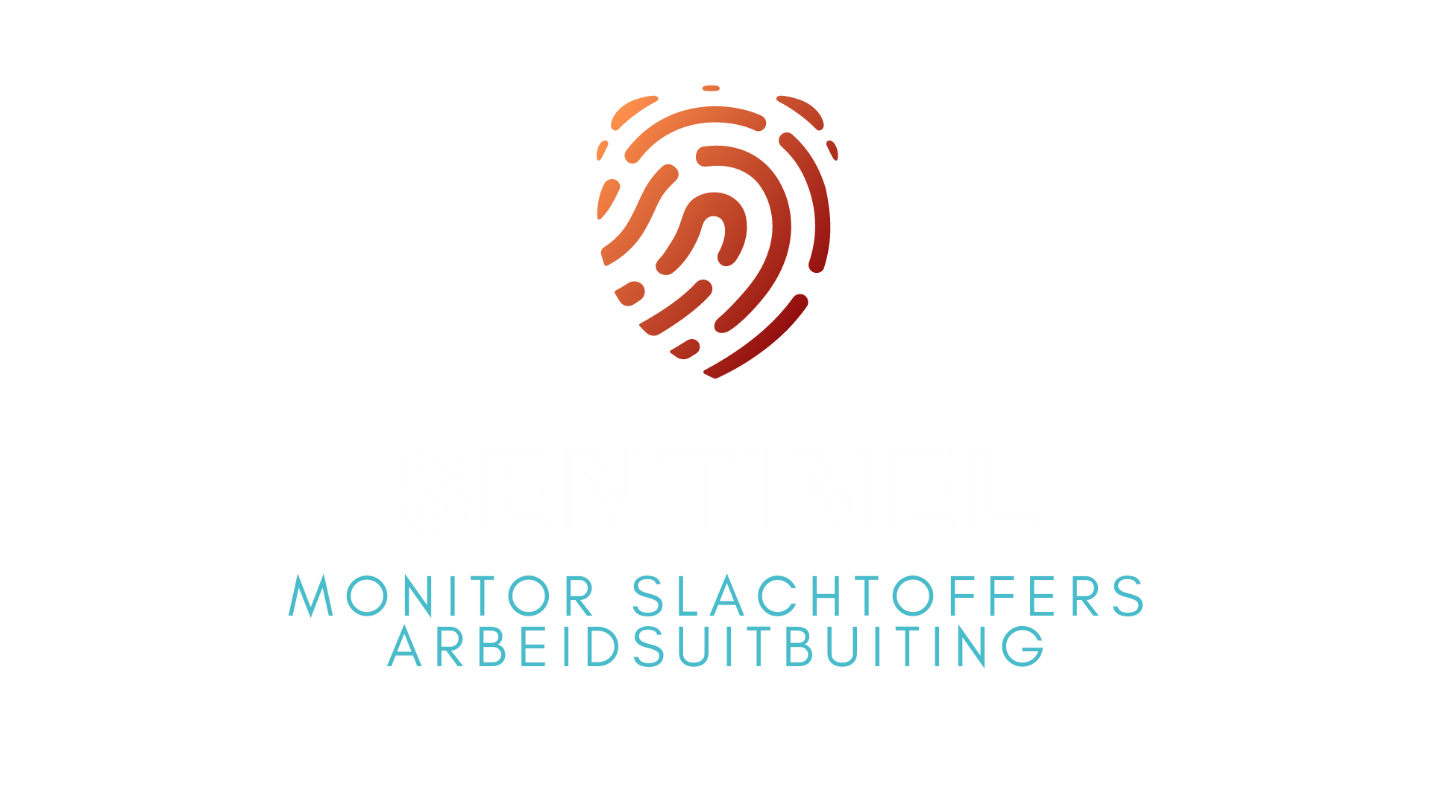
**SENTINEL**

**Development and Setting Guide**



Author: Felipe Ebert

Version: 1.0

Date: 12/08/2022

**Abstract:**

This document is intended to explain how to set the development environment of SENTINEL and the rationale of the design.

Contents

[1. General Architecture 4](#_Toc111201284)

[2. Gmail Account 5](#_Toc111201285)

[3. Frontend – AngularJS 6](#_Toc111201286)

[4. Backend – Spring Boot 7](#_Toc111201287)

[5. Data Model & Database 9](#_Toc111201288)

[6. Sign-on and Authentication – Keycloak 11](#_Toc111201289)

# General Architecture

The SENTINEL is a prototype developed by JADS in collaboration with DITSS. The goal of the tool is to simplify and automate the process of reporting possible cases of labor exploitation in the Netherlands. The system has a simple architecture with 3 main components as micro-services (Figure 1). Up to this stage, all services are only running on localhost.

All the demos (i.e., to showcase the system running on a mobile device) have been running on the browser Firefox (version 103.0.2) with the extension mobile simulator [Mobile FIRST](https://www.webmobilefirst.com/en/).

We still did not decide the hosting platform for the prototype, at Tilburg University there is a [website](https://eur02.safelinks.protection.outlook.com/?url=https%3A%2F%2Fwww.tilburguniversity.edu%2Fintranet%2Fcommunication%2Fchanges%2Fnew-website&data=05%7C01%7C%7C09f17f80c24e4127d29108da22d31c80%7Ccc7df24760ce4a0f9d75704cf60efc64%7C1%7C0%7C637860587360527036%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=f5WTtAFoHyHC09DpLoIbInxfjpPx%2F1i0%2BMOY2mFAjO8%3D&reserved=0) with some instructions a person for contact is [Mira Bückmann](mailto:M.Buckmann@tilburguniversity.edu) from the Marketing and Communication Division.

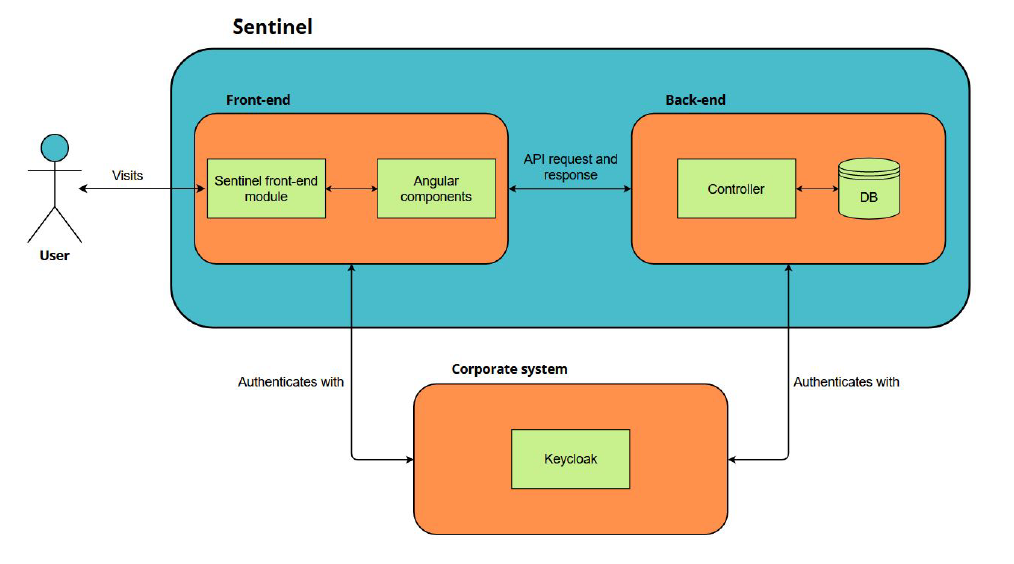


Figure 1 - SENTINEL architecture.

# Gmail Account

I have created a Gmail account for the project. This account used for two scenarios: i) for the mail server (in the user registration process to send the verification email), and ii) for the Google Maps API key. The details of the account are as follows:

Name: Sentinel  
Last name: Monitor  
Email: [sentinel.jads@gmail.com](mailto:sentinel.jads@gmail.com)  
Password: JADS@DenBosch  
Phone: 06 13513964 (Felipe Ebert phone number)  
Recovery email: [felipe.ebert@gmail.com](mailto:felipe.ebert@gmail.com)  
Date of birth: 15-05-1986

**PS1**: For the mail server work properly the “two-step verification” needs to be activated, then add one "app passwords" for “e-mail”. This password is the one used on Keycloak configuration. This is already configured on Gmail and the app password is (the spaces should be included):

App password: lzqg uuuv kgod qudm

**PS2: bellow I describe some TODOs needed to have the account properly configured:**

* **TODO-1**: change Felipe’s personal information (phone, recovery email, and date of birth) on the Gmail account for another responsible person of the project. Please feel free to contact to make these changes.
* **TODO-2**: set up the Gmail account to use the [Maps api](https://mapsplatform.google.com/intl/en/pricing/). A payment method (either credit or debit card) should be informed (this is something I did not receive any response from [Angelique Penners Wouters](mailto:a.c.l.penners@jads.nl)). The first 3 months are for free. Then the key generated should be added to the index.html file in the frontend (in the script tag shown below). The tool will not work until this key is included because the system completely depends on the location from Google API.

<script async src="https://maps.googleapis.com/maps/api/js?key=**TODO**=places"></script>

# Frontend – AngularJS

**Rationale:** we decided to use the framework AngularJS for the frontend because of the complex requirements from the client for the UI, i.e., the functionality to provide several multistep forms in the same report flow. Another important decision we made for the frontend was to make the least number of calls to the backend and database. Thus, the frontend only connects to the backend and database only in two scenarios: i) when the user clicks to create a new report, so the location is verified for previous reports in that location, and ii) to save the report in the end of the process. Therefore, a lot of data is hard coded in the frontend, for example, the text and id of the situations and the offenses and laws which both are also recorded in the database. Additionally, the geo location functionality is handled in the frontend (in the component location) with the usage of the [Google Maps API](https://mapsplatform.google.com/intl/en/pricing/). We use three services: 1) Maps JavaScript API (for the map), 2) Geocoding API (for the markers), and 3) Places API (for the search box). The key used for the Google Maps API should be indicated in the <script> tag in the index.html file. Another important aspect is that the location (i.e., addresses saved in the database) are unique, so the relation between address and report is one-to-many. This means one address cannot be repeated in the database (if this happens for some reason it can crash the system), and one address can have multiple reports, but one report is linked to only one address. This can be observed in the report flow, where after the user filling in the location, the system checks in the database if there are previous reports in that specific address – that is the reason the user can only edit the number and addition of the address, all other fields are not editable for the user (they are obtained from the Google API). The stepper component is the main one where the user fills in the number of reports previously indicated, it also shows the summary of the reports (and the offenses and laws of each situation indicated­­).

**AngularJS version:** ~13.3.0

**Development hosting:** the code is located on TU/e GitLab at [*https://gitlab.tue.nl/febert/sentinel-frontend*](https://gitlab.tue.nl/febert/sentinel-frontend). The access was already granted to [Indika Weerasingha Dewage](mailto:i.p.k.weerasingha.dewage@tue.nl).

**Design:** the system is designed with one Angular component per system screen: there are 7 components. One is the header, and the remaining are the sequential screens: welcome -> location -> previousreport -> numberpeople -> stepper -> reportsaved. There are only 2 Angular services: one for handling the login/registration with Keycloak and another for connecting to the backend and database. Finally, there are 3 basic classes to help modeling the Address, Location, and Report. The code of Keycloak is located in the utility folder.

**Frontend port:** 4200 (default AngularJS)

**Recommended IDE:** Visual Studio Code

**How to run:** simply run “ng serve --o” in the root folder (the option --o automatically opens the url http://localhost:4200/).

# Backend – Spring Boot

**Rationale:** we decided to use the framework Spring Boot to develop the backend in Java because of its simplicity and safety.

**Spring Boot version:** 2.6.4

**Development hosting:** the code is located on TU/e GitLab at <https://gitlab.tue.nl/febert/sentinel>. The access was already granted to [Indika Weerasingha Dewage](mailto:i.p.k.weerasingha.dewage@tue.nl).

**Design:** the backend is implement as a RESTful API with two main services: “getReportsFromAddress” and “saveReports”. Only authenticated users via Keycloak can access the API. It can be noted that when a report is saved in the database, we manually save the relation between the report and situations in the table report\_has\_situation with the method “saveSituation”. A draft of the class diagram is presented in Figure 2 (the class diagram images are also stored in the “class diagram” folder in the backend).

**Backend port:** 8082

**Recommended IDE:** Spring Tool Suite 4

**Plugin needed:** as we have used [Lombok](https://projectlombok.org/setup/eclipse) dependency (to avoid writing getters, setters, and constructors in the model classes, this plugin should be installed on Eclipse (Spring Tool Suite).

**How to run:** on Spring Tool Suite -> run as -> Spring Boot App

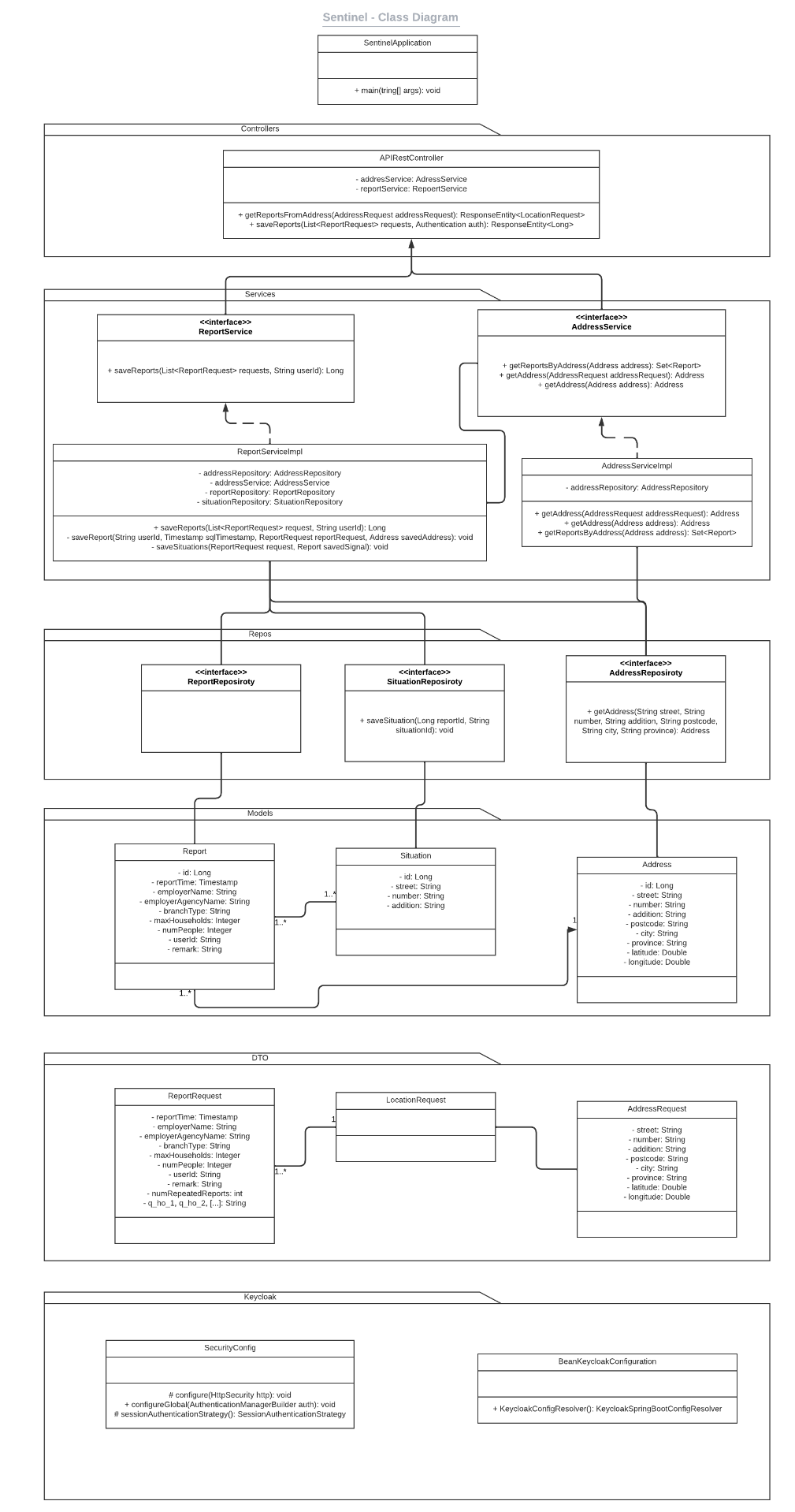
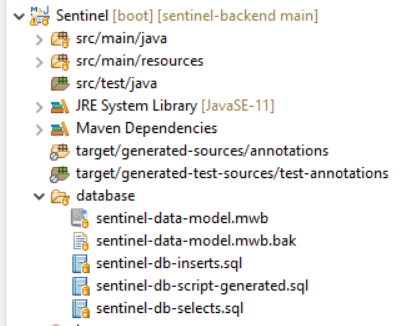
****

Figure 2 - Class Diagram

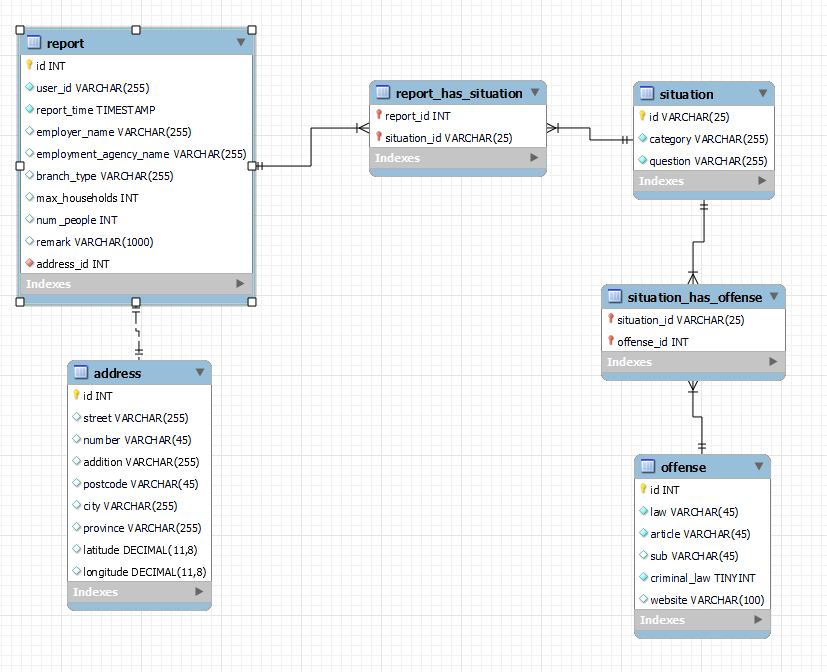
# Data Model & Database

As for the database, we decided to use a simple relational database to store the data: MySQL. All the data model (enhanced entity-relationship ‘EER’ diagram) and SQL scripts are stored in the “database” folder in the backend project:



The “sentinel-db-script-generated.sql” file contains the tables creation script (generated from the model). The “sentinel-db-inserts.sql” file contains the inserts on the situation, offense, and situation\_has\_offense tables. This data is static. In the future, when people from DITSS/DISS provide more and updated information about offenses the database should be updated (also in the frontend).

The data model is as follows:



As noted, the user information is not present because it is stored in Keycloak. We just save the user\_id. Also, as the situation and offense data is static (hard coded), when a report is saved in the database, we just add a report, address, and update the table “report\_has\_situation”.

**MySQL version:** 8.0.22 (MySQL Community Server - GPL)

**MySQL port:** 3306 (default MySQL)

**Recommended IDE:** MySQL Workbench (version 8.0.28)

**Database configuration:**

* **Name:** sentinel
* **URL:** *jdbc:mysql://localhost:3306/sentinel*
* **Username:** root
* **Password:** admin

# Sign-on and Authentication – Keycloak

**Rationale:** we used [Keycloak](https://www.keycloak.org/) for the registration, authentication, and authorization of the end users. So, when the user hits the system url and is not logged in, the frontend redirects it to Keycloak service. With Keycloak, the user can register and log in in the system. The idea is that the end users will be able to register themselves in the system, and to do so, we can set specific mail domain address allowed to do such operation (whitelist). In the registration process, the user will receive an email to confirm their identity (verify email). We also created a theme for SENTINEL on Keycloak, and in this theme we added 3 additional fields in the registration: phone, police unit, and organization.

**Keycloak version:** 16.1.1 (standalone)

**Development hosting:** the code is located on TU/e GitLab at <https://gitlab.tue.nl/febert/sentinel-keycloak>. The access was already granted to [Indika Weerasingha Dewage](mailto:i.p.k.weerasingha.dewage@tue.nl).

**Design:** we create one realm “sentinelservice” with two clients: “sentinel-backend” and “sentinel-frontend”. Each of the client is responsible for securing the front and back end. Even though now there is no distinction between different users in the frontend, the backend and Keycloak are already supporting it. Keycloak is configured with different roles: “app-admin” and “app-user”.

**Keycloak port:** 8080

**How to run:** simply run the file “standalone.bat” from “sentinel-keycloak/bin” folder, then access <http://localhost:8080/> and go to “Administrative console”: the user and password are both “admin” for now. They should be changed when on production!

**References for setting a theme on Keycloak:**

<https://www.baeldung.com/spring-boot-keycloak>  
<https://www.baeldung.com/spring-keycloak-custom-themes>  
<https://www.baeldung.com/keycloak-custom-login-page>  
<https://www.baeldung.com/keycloak-user-registration>  
<https://www.baeldung.com/keycloak-custom-user-attributes>

**References for setting a whitelist based on email address on Keycloak:**

<https://github.com/micedre/keycloak-mail-whitelisting>  
<https://keycloak.discourse.group/t/how-to-restrict-registration-to-specific-email-domains/702/8>

**References for setting a mail server for user registration (verify email) on Keycloak:**

<https://dev.to/rounakcodes/keycloak-configure-realm-email-settings-gmail-3dfn>

**References for setting composite roles on Keycloak:**

<https://www.thomasvitale.com/keycloak-configuration-authentication-authorisation>

**References for setting AngularJS and SpringBoot with Keycloak:**

<https://medium.com/@kamleshbadgujar00/secure-spring-boot-angular-9-application-using-keycloak-1-3-b00e801ba693>  
<https://medium.com/@kamleshbadgujar00/secure-spring-boot-angular-9-application-using-keycloak-2-3-bf98ba360d66>  
<https://medium.com/@kamleshbadgujar00/integrating-angular-application-with-keycloak-3-3-8aee940897e>