Technical Evaluation 2 – OPEN SCOPE

This document describes the **second technical evaluation** for the IDP trainees. The main objective is evaluating the creativity and adaptability skills by keeping the scope open. The general guidelines will be provided and the trainee must choose a theme and develop it.

1.Softwares:

The following softwares are mandatory to be used during the application development:

- Java Development Kit 11
- Maven (latest version)
- Docker Engine, WSL 2 and Ubuntu 20.04

Note: The *Docker Desktop for Windows* must be avoided due license restrictions.

As development tools, you can use the same ones used on first evaluation project.

2.Guidelines

The following guidelines will help the trainee to develop his project:

- Search for interesting projects and choose one. The project must integrate to others on-line system;
- The project consists in a back-end application which provides REST endpoints for their clients and consumes remote endpoints from existing online project;
- Write a summarized documentation (provided in README.md) describing the functional requirements in high level;
- Avoid large number of endpoints. About 4 and 6 endpoints are already sufficient;
- Use Spring Boot platform and *Spring Initalizr* (start.spring.io) for initial configuration;
- The project must have the **trainee identification** as prefix. Ex: VitorFigueiredo_MySolutionProject
- Use microservices concept based on containerizations. A suggestion is to use 2 different containers: a) a container for Spring Boot application; b) another container for MySQL database;
- Use Spring cache resource;
- Use docker-compose to integrate the containers;
- You must provide the API documentation following the Swagger and OpenAPI 3 specification;
- All service methods must have the respective unit tests. Success and failure tests must be contemplated;
- Postman can be used for manual functional tests during the development phase (deliver automated scripts for Postman is optional). For the demonstration is expected to have some endpoint's automated functional tests using any known test framework (e.g. Cucumber, Karate, Rest Assured). Positive and negative scenarios must be explored and the assertions must be done on both response status code, header (if any) and payload.

3.Important Tips

- At first phase of project, the H2 can be used as local file database and there is no need for container configuration. But, at the final phase, you must configure the container files and switch to MySQL database container;
- Work in incremental way: First, make the **UML class diagram**, then build **entity classes**, then **repository layer**, then **service layer** and finally the **controller layer**. Remember to design the **DTO classes**;
- As you develop each layer, write the unit tests and correct the failures as they are found;
- Remember to document the business rules;
- You are encouraged to propose algorithms, utility classes or "out-of-the-box" architectures as long as they represent real contributions to the final solution;
- The API documentation does not need to enable the user to send requests and receive responses.

Deliverables

The final project has to be delivered as a GitHub repository link. The repository must have the following items:

- The project must be structured as Maven project and so will be able to be compiled, tested, built and packaged by command line (`mvnw`);
- The unit tests covering all business methods and classes;
- The README.md file must have the instructions to run the application with all containers;
- The API documentation based on Swagger and following the OpenAPI 3 Specification. Export the documentation to a PDF file. Create the docs folder and put all documentation files including the exported PDF file;
- Create the modeling folder and put all exported images from UML class diagrams;
- Create the presentation folder and put the respective project´s PowerPoint presentation.

# Excellent project #