INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Open source e-learning plataform

Projeto e Seminário

Licenciatura em Engenharia Informática e Computadores

André de Oliveira nº44580
917135594 `alaqo@hotmail.com`

Rodrigo Leal nº44823
961604272 `rodrigomfl@hotmail.com`

João Santos nº44847
926366577 `j.ers.santos@outlook.com`

Advisors:
Cátia Vaz, cvaz@cc.isel.ipl.pt
José Simão jsimao@cc.isel.ipl.pt

March 9, 2020

1 Introduction

Nowadays there are some platforms that provide an environment for defining algorithms and testing, but not all of them are open source, don't have such an appealing environment or they don't allow multi-language.
Therefore, this project intends to develop an open source e-learning platform, dedicated to the definition and testing of algorithms in a multi-language environment.
This platform will have several challenges which can be solved for study or evaluation purposes. This can be useful in academic environments, professional interviews or programming enthusiast.

2 Analysis

To develop the e-learning platform 3 main modules were identified: UI, Services and Execution Environments.
The UI module is the presentation layer, with which the final user will interact. This interface will be developed with React, a JavaScript library for building user interfaces, as a single page application.

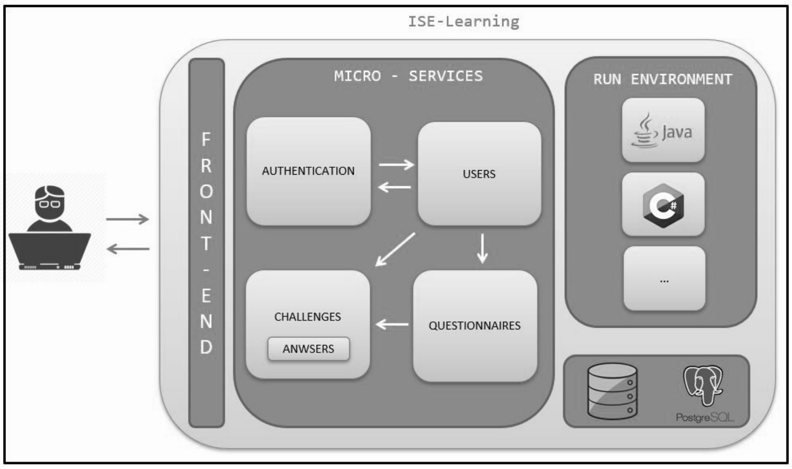


Figure 1: Architecture

The Services module will provide a REST API which is the core of the platform, this module will be developed using a microservice architecture. This REST API can be used standalone or with the UI module. In this project the REST API will be used to support the UI module. This module will be developed with the following technological stack: Kotlin; Gradle; Spring; Docker; Postgres; Swagger; Cloud based hosting solution.
Kotlin is the programming language which will be used to develop the microservices, Gradle will be used for dependency management, Spring will be used as a support framework for dependency injection and REST application development.
Docker will be used to build and run containers for each microservice and the database. Postgress will be used as the SQL client to enable data persistence across the platform. Swagger will be used to document the REST API and a cloud based solution will be used to host the platform, on a later date a comparative analysis will be done to find the right solution for this project with the major cloud providers (GCP, Azure, AWS).

The Execution Environment module will be responsible for executing code provided by an external source. This module will support several runtime environments, where each application will be developed and hosted on a separate container.

Some risk were identified during the project analysis: scalability, requires extensive load tests with the solution deployed on cloud environment; lack of diverse userbase for testing, complex solutions required thorough testing to minimize bugs; group members are currently working and studying at same time; the learning process required for new tools has inherent uncertainty; estimates may not meet proposed values.
The major constraint of this project's execution is time management. Taking this into account several strategies were devised as contingency plans if need be: scope reduction, removing questionnaire functionality; group members taking personal vacation days; drop one of the other courses.

2.1 Requirements

After analysing the objectives of this project we identified functional and non-functional requirements.

Functional:

- Multi-Language Provide run environment for multiple programming languages.(eg. Java, Kotlin, C#, JS, Python)
- Execute Solution Any user may write a code solution, run it and get the result trough the front-end application.
- Basic Authentication Users can create an account, with a profile information and will be able to get access to more features(eg. keep track of the solved challenges, create/solve Questionnaires to/from other users).
- Challenges are a programming problem that needs to be solve. Every challenge has a built in answer that will be compared with the user submitted solution to determinate its "correctness" trough unit tests.
    - Any user may create a Challenge, but also has to create its unit tests, and they have to compile/run successfully before being able to submit it. Only its creator can edit the Challenge.
    - Challenges may have tags associated, and they can be searched by them.
    - A logged in user can track the Challenge he/she submitted.
- Questionnaires are a group of Challenges.
    - Questionnaires can only be submitted or solved by logged in users, and they consist in a group of existing Challenges.
    - A user may choose any number of Challenges to create a Questionnaire and set it a timer. Only its creator can edit the Questionnaire. (eg. user A created a Questionnaire with 5 Challenges that has to be solved by user B in one hour).
    - A user may save its favourite Questionnaires.
    - Questionnaires may be shared trough a link.
    - Only logged in users have access to the Questionnaires.

Non-Functional Requirements:

- Scalability This platform may be accessed by hundreds or even thousands of users. As such the proposed architecture takes into consideration the scalability of its usage.
    - The microservice architecture allows tasks to be broken down into smaller units since the services follow the single responsibility principle.
    - Hosting these services in containers and managed in a Kubernetes cloud based environment makes it easier to scale up instances of those services, meeting high peek demands when necessary.
- Security Executing third party code in a machine raises security concerns.
    - Executing the services which will execute this code in containers adresses some of these concerns. A self-contained run environment limits the impact of malicious code to the container which executes it, protecting the remaining infrastructure.

- Solution Maintenance Maintaining a complex solution requires a balance between many moving parts, as such this project's architecture reflects the principles of loose coupling and modularity which facilitate the solution maintenance.

    - The module separation of the platform maintains loose coupling between the modules.
    - The microservice architecture maintains the services scope smaller and reduces dependence from services implementation.
    - Components will be deployed in containers the infrastructure maintenance will be easier because it will only need to support Docker containers

- Efficiency Hosting the solution in a cloud based environment improves efficiency of the solution.

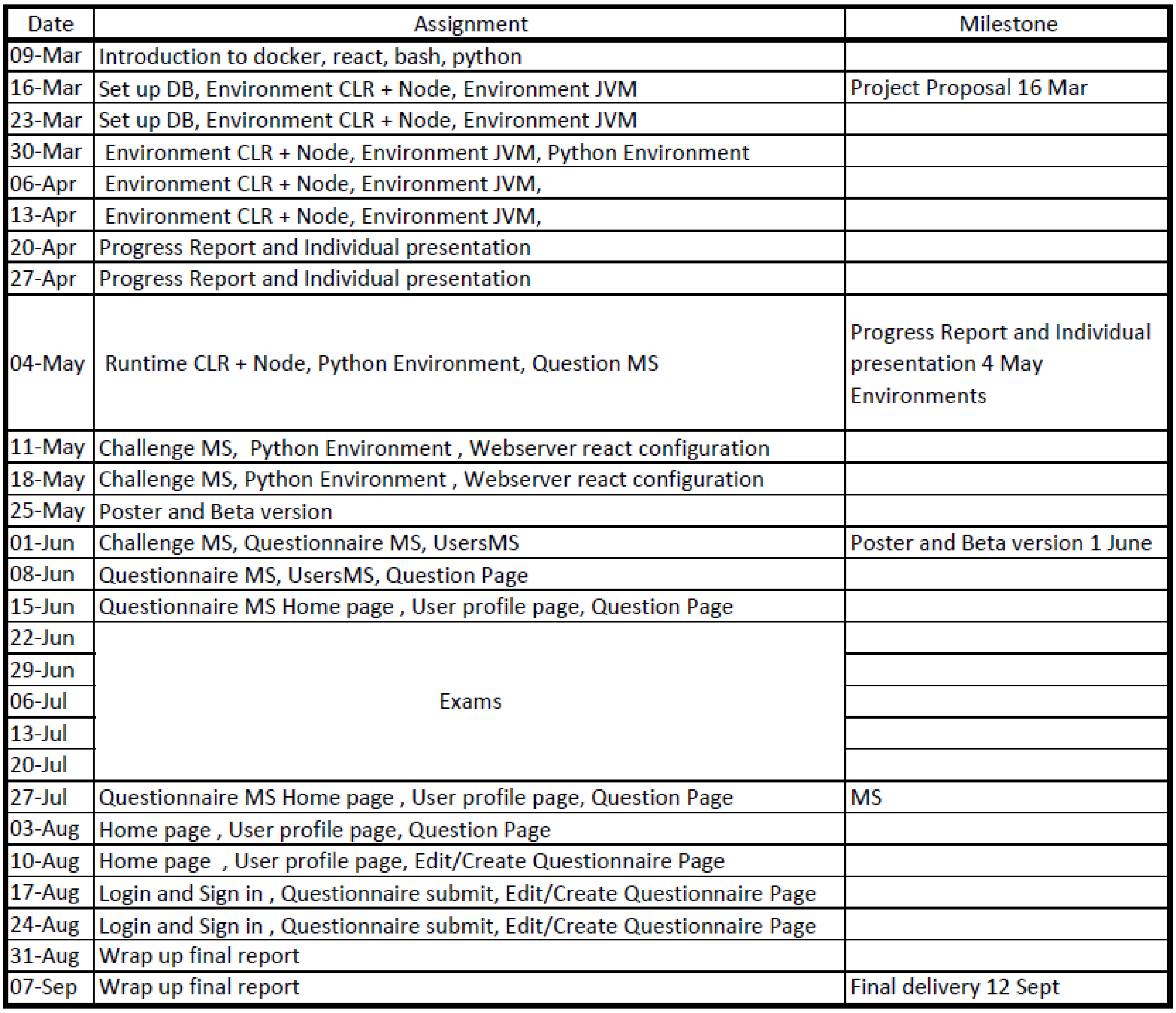
    - Cloud based hosting provides load balancing for services, improving network load efficiently across multiple servers.
    - Cloud based hosting allows for a more efficient resource management, allowing for scaling up or down resource usage adjusting to current demand.

- Infrastructure Agnostic Because every component will be deployed in separate containers the deployment will be platform independent and fault tolerant to component failure when managed in a Kubernetes cloud based environment.

- Open source Open source software refers to something people can modify and share because its design is publicly accessible.

    - Improved control since the is publicly available.
    - Improved security since the code is accessible to be tested and verified stimulating error detection.
    - Improved stability since the source code is always available
    - Fosters community, group of people invested in producing, testing, using and promoting the software.

3 Schedule



Date	Assignment	Milestone
09-Mar	Introduction to docker, react, bash, python	
16-Mar	Set up DB, Environment CLR + Node, Environment JVM	Project Proposal 16 Mar
23-Mar	Set up DB, Environment CLR + Node, Environment JVM	
30-Mar	Environment CLR + Node, Environment JVM, Python Environment	
06-Apr	Environment CLR + Node, Environment JVM,	
13-Apr	Environment CLR + Node, Environment JVM,	
20-Apr	Progress Report and Individual presentation	
27-Apr	Progress Report and Individual presentation	
04-May	Runtime CLR + Node, Python Environment, Question MS	Progress Report and Individual presentation 4 May Environments
11-May	Challenge MS, Python Environment , Webserver react configuration	
18-May	Challenge MS, Python Environment , Webserver react configuration	
25-May	Poster and Beta version	
01-Jun	Challenge MS, Questionnaire MS, UsersMS	Poster and Beta version 1 June
08-Jun	Questionnaire MS, UsersMS, Question Page	
15-Jun	Questionnaire MS Home page , User profile page, Question Page	
22-Jun	Exams	
29-Jun		
06-Jul		
13-Jul		
20-Jul		
27-Jul	Questionnaire MS Home page , User profile page, Question Page	MS
03-Aug	Home page , User profile page, Question Page	
10-Aug	Home page , User profile page, Edit/Create Questionnaire Page	
17-Aug	Login and Sign in , Questionnaire submit, Edit/Create Questionnaire Page	
24-Aug	Login and Sign in , Questionnaire submit, Edit/Create Questionnaire Page	
31-Aug	Wrap up final report	
07-Sep	Wrap up final report	Final delivery 12 Sept

Figure 2: Requirements