
**Área Departamental de Engenharia de Eletrónica e Telecomunicações e de Computadores**

# Open source e-learning platform

## Beta Report

Authors: 44598 André L. A. Q. de Oliveira
44847 João Eduardo Santos
44823 Rodrigo Mogárrio F. Leal

Beta Report for Unidade Curricular de Projecto e Seminário
Licenciatura em Engenharia Informática e de Computadores

Advisors: Cátia Vaz, José Simão

15 – June – 2020

# Index

# List of Figures

# List of Tables

# 1. Introduction

In today's competitive job market, programming jobs are amongst the most desirable careers. The ability to innovate, create and troubleshoot all kinds of technologies on a daily basis is what drives many individuals to seek experience and pursue a future in computer science or coding.

To accomplish this ambition one can be a self-taught enthusiast, or one can seek the knowledge of professionals through all sort of courses and universities to acquire a considerable high amount of skills sets that will allow to succeed in whichever field of choice they commit to program. But one thing is for sure, we live in a fast-paced world, and Information Technology (IT) is no different. It is constantly changing and evolving, and new trends appear every day, along with new technologies and marketing strategies. To this is clear that no matter how long one codes, eventually will be faced with the need to keep learning new skills and improve oneself, so prevent becoming outdated, or to be better prepared for a new job interview, or even if just to improve the academic performance.

For this reason, there are out there some platforms that provide an environment for defining algorithms and testing [1]. However, many of them are not open source, or they do not have such an appealing environment or just do not allow multi-language. Therefore, this project intends to combine all the strengths mentioned above with none of barriers and create the IS E-Learning platform, an e-learning platform to help other programmers achieve their goals.

Being an e-learning platform brings to the table certain inherent aspects, like allowing to be accessed from anywhere provided that exists an internet connection. This specific trait gives a very attractive perk to the client, which is, the code-execution environment. The idea is to deliver an uncomplicated and easy-going experience to the user where it is possible to write solutions, build test cases, run the code and check the output directly on our website without having to configure an environment, or download endless libraries. Another positive aspect of this single attribute is that it has the potential to serve as a powerful tool to ISEL, if in the future it could be implemented in school computers, allowing not so wealthy students that can't afford tech gear, a way to help them thrive through their academic studies.

Coding out solutions to algorithm problems is the best way to practice and learn, but the truth is, that doing so with just a tool to run code without any structured guidance makes the process of learning more challenging. Understanding the inner workings of complex algorithms is no easy task, and even experience programmers nowadays struggle with coding interviews for the simple fact that they are hard and go beyond algorithms and data structures. Companies want to hire the best of the best, and they value someone who can develop an high class product, which means the programmer must be able to create something that is performant, stable, scalable and bug free, and to be able to deliver such a system or product, one must be proficient and understand algorithms and have mastery in programming languages. For this reason, and because the best way to learn is from examples of

someone who understands the subject, the IS E-Learning platform comes with a service that provides *Challenges*, which are programming problems that needs to be solved. And because in our own path we learn much by reaching out to the coding community, through forums or other people examples, we also want to foment this concept of community, by allowing any registered user to make good use of the his own gathered expertise and create his own challenge and share it on the platform so that others might learn from it.

But despite of how much an individual studies or practices, he/she will only know if he/she has mastered the topic when put to the test. Sometimes is not all about the smartness or skills, but flexibility, stress-resistance, and the ability to iterate approaches fast. To validate this preparation state, IS E-Learning platform has a service where it is possible to create *Questionnaires*. Questionnaires are a selected number of pre-existent challenges all grouped up and put together to create a single assessment.

One beautiful thing about programming, is that it is everywhere, and that it can be used in any field area to help solve a problem. But with that, comes that not everyone speaks the same programming language, mainly because languages were created to better suit a specific theme, like web development, machine learning or data analysis. As engineers it is not enough to be only good at one thing, since that will not only limit our work opportunities but as our own problem-solving skills. For that reason, we wanted our platform to support multilanguage, and currently it provides 5 popular ones.

In the end, our goal is simple and honest, not only we want to provide an appellative e-learning platform that can be useful in academic environment, professional interviews, or even for just a programming enthusiast who wants to learn more, where every user can solve and create coding challenges, as well as questionnaires to put to test the best of his abilities, but also we want to do it in openly manner, so that it can be freely accessed, used, changed, and shared by everyone.

## 1.1. Outline

This project is divided into 6 chapters.

Chapter 2 describes both functional requirements, where the technical details are explained in order to illustrate what our platform is supposed to accomplish, and the non-functional ones, which are mainly focused on specific design and implementation concerns of the solution, so it can meet the requirements, with great performance and a solid security.

Chapter 3 briefly describes the current state of art regarding similar platforms, and a comparison them and our own solution is performed.

Chapter 4 introduces some of the technologies that support are used for the development of the solution.

Chapter 5 addresses focus on the architecture, implementation details regarding introducing each component that composes IS E-Learning platform, their functionalities, and their interactions.

Chapter 6 addresses on how the modules discussed in chapter 5 were designed, explaining their functionalities and details of their implementation.

Chapter 7 explains what changes need to occur when adding support to a new language and indicates which documentation should be consulted on the matter.

Chapter 8 gives an overview about the progress of the project, what has been completed, the road ahead and some considerations for the planning of the remaining activities.

# 2. Requirements

For this project a series of functional and non-functional requirements [2] were identified and are listed below.

## 2.1. Functional Requirements

- Challenges – Challenges are a programming problem that needs to be solved. Every challenge has a built-in solution that will be compared with the user submitted answer to determinate its "correctness" through unit tests.
    - Challenges can be solved on one or more programming language;
    - To respond to a Challenge a user doesn't need to be logged in;
    - Only logged in users can create Challenges;
    - To create a Challenge a solution and unit tests must be provided;
    - To create a Challenge the code must compile and the tests must pass;
    - Challenges can be associated with tags, which can be used to search specific topics;
    - Only a logged in user can consult the Challenges he/she submitted;
    - A user can create a private Challenge that is unreachable as a single Challenger and can only be visible in a Questionnaire created by the Challenge's creator;
    - Only a logged in user can track and consult previously answered Challenges;
    - A Challenge's solution can only be edited by the Challenge's creator;
    - A public Challenge's solution can be always consulted;
    - A private Challenge's solution can only be seen by its creator;
    - Challenge's answer can only be consulted by the user that submit it;
- Execute Code: users will have a UI element where they run code in a multitude of languages:
    - Users don´t have to be logged in to use this functionality;
    - Users can choose a language to write code;
    - Users can run then written code and verify the output;
- Questionnaire: is a group of Challenges.
    - Only logged in users can create Questionnaires;
    - Can have public and private Challenges;
    - Only the creator can edit the Questionnaire;
    - Questionnaires can be shared through a link created by the platform.
    - Questionnaire can be associated with tags, which can be used for searching
    - Questionnaire can have a timer associate with it;
    - Questionnaire's timer starts when link is accessed;

  - Questionnaire's creator can define what programming language can be used in any challenge;
  - Questionnaire's creator can decide whether the user responding can view the final evaluation or not;
  - Submitted answers for a Questionnaire challenges can only be viewed by the Questionnaire's creator;
  - Submitted answers cannot be modified or deleted;
  - Non submitted answers are considered as wrong;
- Authentication:
  - Users can create an account;
  - Authentication uses a basic username/password scheme;
  - When creating an account, user must provide username, password, name, email and an avatar;
- Multi-Language:

Platform must provide an environment to run code for multiple programming languages (Java, Kotlin, C#, JavaScript and Python)

## 2.2. Non-Functional Requirements

- Scalability: This platform may be accessed by hundreds or even thousands of users. As such the platform must be able maintain a high level of performance.
- Security: Executing third party code in a machine raises security 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.
- Efficiency:
  - Hosting the solution in a cloud-based environment improves efficiency of the solution.

## 2.3. User Journeys

With the previous enumerated requirements, it was possible to define a series of user journeys that will help to identify and implement functionalities.

Figure 1 represents the user journey for registration that reflects the account creation and information needed for Authentication's requirements.

Figure 2 represents the user journey for solving a Challenge that reflects the basic authentication scheme and the multiple language, everyone can solve a challenge, read only solution and public solution Challenge's requirements.

Figure 3 represents the ability to create Challenges and reflects the user's permissions, code compilation, mandatory fields, tags and challenge tracking Challenge's requirements.

Figure 4 represents the ability to create a Questionnaire and reflects the user's permissions, Challenge's privacy, shareable link creation, timer and final evaluation visibility requirements present in the Questionnaire section.

Figure 5 represents the ability to run a piece of code in the platform and reflects all requirements present in the Run Code section.

Figure 6 represents the ability to answer a Questionnaire in the platform and reflects the shareable link, timer, final evaluation visibility and submit answers visibility requirements present in the Questionnaire section. With these user journey we get a good coverage of the requirements and base usability of the platform.

### 2.3.1. Registration

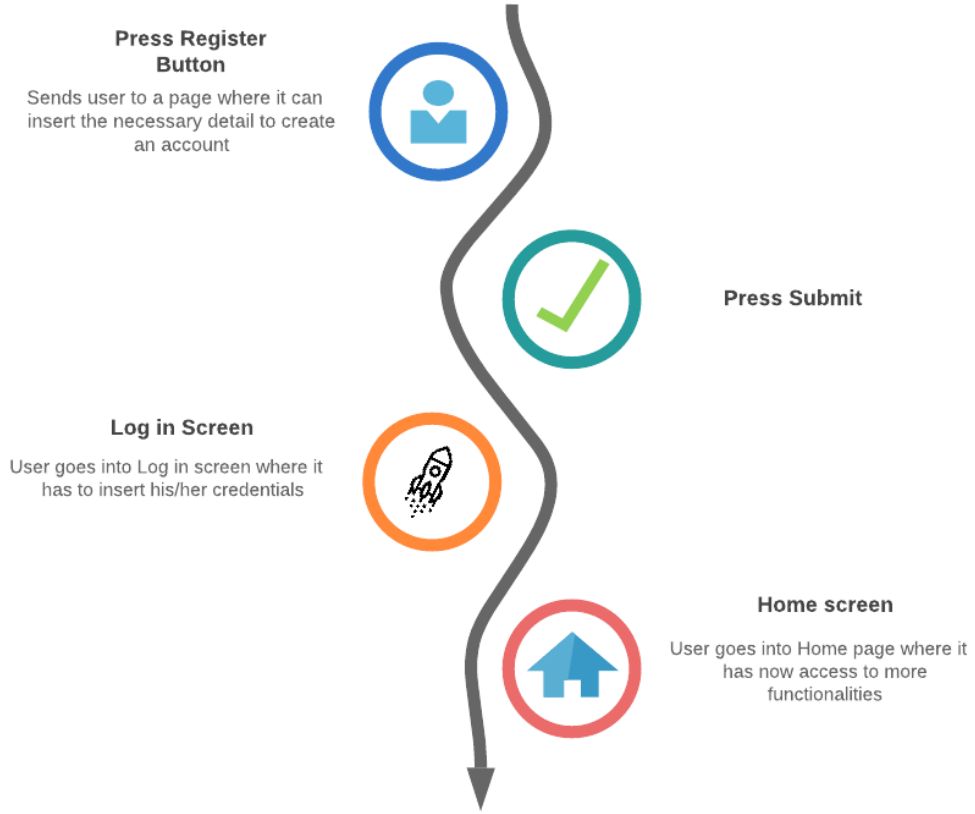


Figure 1 - User Journey for user's registration

## 2.3.2. Solving a Challenge


Figure 2 - User Journey for solving a Challenge

## 2.3.3. Create a Challenge


Figure 3 - User Journey for creating a Challenge

## 2.3.4. Create a Questionnaire


Figure 4 - User Journey for creating a Questionnaire

### 2.3.5. Run code

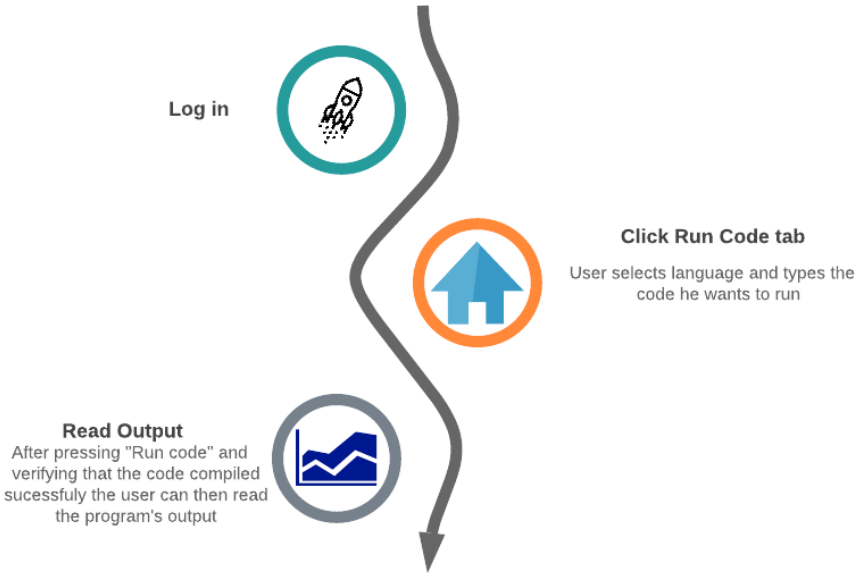


Figure 5 - User Journey for running a piece of code

## 2.3.6. Answer a Questionnaire


Figure 6 - User Journey for answering a Questionnaire

# 3. Related work

As emphasized in the introduction, there are plenty of e-learning platforms out there, each one them serving their own purpose and having unique characteristics. In this chapter we aim to briefly showcase some of them, as a way to demonstrate what are the most common features between them and our own platform, and their differences weighing their pros and cons.

## 1.1. AlgoExpert

AlgoExpert [3] was made to serve as a resource to prepare for coding interviews, by providing everything someone needs in one streamlined platform. It has 90 hand-picked questions, where only 4 of them are free, but it this possible to get the full platform content for the price of 115€ per year. Despite only having 7 programming languages, they differ from other e-learning platforms by providing over 60 hours of video content. Each question is accompanied by a two-part video, explaining a conceptual overview of the algorithm in how to approach, implement, optimize and how to analyze its space-time complexity, followed by code walkthrough in order to maximize learning. They also have coding interview tips videos to help coders stand out from other software engineers and publicize full projects contests for their clients to promote their programming skills.

## 1.2. HackerRank

HackerRank [4] is one of the most famous websites platforms for aspiring developers and its highly rich feature wise. On our comparison list it is also the most expensive one, where the individual package costs 230€ per month, but then again it offers beyond the basic coding challenges. It has a clean design, and it is possible to learn on over 25 languages by resolving coding problems, where each problem has a unique leaderboard as well as a solution that provides an explanation of how to approach. It also gives the possibility for a user to create his own problems and share them with friends and participate in challenge competitions. Additionally it also provides the ability for users to submit applications and apply to jobs by solving company-sponsored coding challenges, offering a win-win service not only for developers that want to practice their coding skills and prepare for interviews, but also for companies through their interview platform that helps identify and hire developers with the right skills.

## 1.3. LeetCode

LeetCode [5] has a very intuitive and appealing interface that makes the navigation on their website very satisfying. They have over 1500 problems, categorized by tags and difficulty, and available in 14 programming languages. Most of the problems are free for the common user, but there is premium

content to subscribers that pay 36€ per month or 147€ per year, which includes more questions commonly asked in famous companies like Google or Amazon, solutions and premium solutions to the problems, and other features like possibility to write with autocomplete or debug the code. They also have an online judge for the problems as well as a service that mocks interviews, where a session is launched for a certain amount of time where the users have to submitted the correct answer for each question before the time expires or they end the session manually. Not only does LeetCode prepare candidates for technical interviews, but also help companies identify talent through sponsoring contests.

## 1.4. Codewars

Different from all other platforms, Codewars [6] makes learning programming a lot of fun. Offering a huge repository of over 8600 problems in more than 56 programming languages, and ranking system as well as the ability to form coding clans, this platform has a strong active community. A user with a certain amount of ranking points, obtainable by solving problems, may help the platform grow by creating his own he unique problem. This problem may enter the Codewars repository collection if it receives a high positive feedback, which is also given by the community, and may later be translated to other languages, also with the efforts of the community. Each problem has its own feedback comment session where users may discuss about their implementations, and it is possible to always see others users solutions as long as one has already completed the challenge or if it "give ups" and loses ranking points. Although it is not an e-learning platform *per se*, it accomplishes the same effect by making people addicted to coding by making it a stimulant friendly competition with an excellent user interface experience. Codewars also works with tech companies to find good problems solvers and has an optional subscription for 4.5€, that offers not so substantial features such as profile badges, ad-free experience or member-only cluster environments to get faster results.

## 1.5. CodeChef

CodeChef [7] was born as non-profit educational initiative with the aim to providing a platform for students and young software professionals to practice and hone their skills through online contests. Even having over 4000 problems to practice in more than 55 languages, and a big community, the platform itself is simple and does not offer many features. The reason being that Codechef exists more like an initiative. It excels at promoting coding events in schools, hosting various contests and competitions with not only cash wining prizes but also teach gear, organizing workshops and doubt sessions. There is also the "CodeChef For Schools" program that aims to reach out to young students and encourage them a culture of programming in Indian schools.

On Table 1 it is possible to look at an overview of the most conventional features on each platform.

Table 1 - Feature comparison of select platforms

| *Platform* | *Problems* | *Unit tests* | *Languages* | *Community Spirit* | *Design* | *Price* | *Driven* | *Open Source* |
|---|---|---|---|---|---|---|---|---|
| *AlgoExpert* | 90 | Yes | 7 | Good | Clean | 115€/yr | Job | No |
| *HackerRank* | > 100 | Yes | 35+ | Good | Clean | 230€/mo | Job | No |
| *Leetcode* | > 1500 | Yes | 14 | Good | Good | 147€/yr | Job | No |
| *Codewars* | > 8800 | Yes | 55+ | Very Good | Very Good | Free | Entertainment | No |
| *CodeChef* | > 4300 | No | 55+ | Good | Poor | Free | Educative | No |
| *IS E-Learning* | N/A | Yes | 5 | N/A | Clean | Free | Educative | Yes |

# 4. Related Technologies

For the development of this application specific technologies were selected. Due to the nature of this projects the number of used technologies is vast, in this chapter a subset of the most relevant technologies were selected to be described in more detail.

## 4.1. React

React is a JavaScript library for building user interfaces. Create by Facebook, it is currently a widely used library used for front end development [8] [9].

One of the big advantages of using react is being able to build components which can be independent from each other and can be reused across all application's components, this dramatically improves modularity, provides loose coupling between components and facilitates maintenance of the solution.

The initial configuration of the project is done with the help of a npm package, create-react-app. This package creates the barebones of the client-side code including the first component to be rendered. That component can be edited, and other components can be built using the JSX language. JSX is a syntax extension to JavaScript, it looks like HTML but has the full power of JavaScript [10].

React Router is a library which enables route handling using dynamic routing. This allows developers to build a single-page web application with navigation without the page refreshing as the user navigates.

## 4.2. Spring

Spring is one of the most popular application development frameworks. This lightweight and open source framework enables high performance, easily testable and reusable code [11].

Spring offers several core functionalities like inversion of control (specifically dependency injection), aspect-oriented programming, database access, transaction management, web service development through Spring MVC, amongst many others [11] [12].

Besides the core functionalities, Spring has several projects which allow to extend these functionalities for specific needs. Two projects worthy of mention are Spring Boot and Spring Security which are used on this project.

### 4.2.1. Spring Boot

Spring boot makes it easier to develop Spring applications. Includes embedded Tomcat, Jetty or Undertown as web application servers allowing the development of standalone applications, automatically configure Spring and 3rd party libraries when possible, offers a set of dependencies to

help build the application (starter dependencies), requires no XML configurations and no code generation [13].

Adding to this, Spring Boot is a widely used project which has a very active community.

## 4.3. Swagger

Swagger enables developers to describe their API's structure in such a way that it is possible to build both beautiful and interactive API documentation [14]. Swagger UI enables automatic generation of a rich user interface with the API documentation, this UI is generated from documentation compliant with the Open API standard.

## 4.4. Docker

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers [15] [16] [17].

Containers are a standardized unit of software that allows developers to isolate their app from its environment, solving the "it works on my machine". Includes everything needed to run an application: code, runtime, system tools, system libraries and settings. Any docker client will be able to run the container in any machine. For developers, it means that they can focus on writing code without worrying about the system that it will ultimately be running on.

Another advantage of the containers is that they are lightweight, require fewer resources and have very quick start up times, and secure, the container provides isolation from other containers.

Docker containers are built from Docker images, in order to run an application inside a container an image with the application needs to be built, build a container from that image and only then can the image with the application be executed. A Docker image is an immutable file which contains the source code, libraries, dependencies, tools, and other files needed for an application to run.

There are several images available for use in docker in image registries like Docker Hub [18]. For most cases custom docker images need to be built and these can be built recurring to Docker files.

A Dockerfile is a text file which includes the instructions to build a Docker image. A Dockerfile specifies the operating system, the runtimes, environmental variables, file locations, network ports, other components it needs and what the container will be doing once we run it. With a Dockerfile a Docker client can build an image, build a container from that image and execute it.

# 5. Architecture

To develop the IS E-Learning platform three 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 as a single page application.

The Services module will provide a REST API [19] which is the core of the platform. This REST API can be used standalone or with the UI module and will be used to support the UI module.

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.

On Figure 7 is shown how these modules interact, the Frontend module only communicates with the services module which in turn communicates with the execution environments, increasing the solution's modularity.

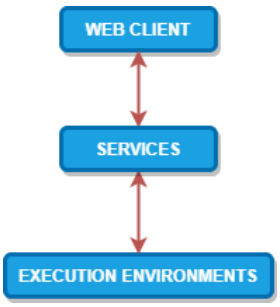


Figure 7 – Architectural Layered Module view, with inter module interactions

## 5.1. Web Client

The web client end will be a Single Page Application (SPA) enabling a user to interact with the application through an UI. This module will be implemented with React and React Router.

For this use case, the web application was built using NodeJS. Adding to this some external libraries/frameworks were used to support the UI development: Material UI; CodeMirror; Formik; Yup. These are explained in more detail below.

To take advantage of React and follow good development practices, on this project there is a concern to implement components with a modular design. With this approach the components can be reused in different pages making the application more modular and loosely coupled.

### 5.1.1 Material-UI

Material-UI [20] is one of the most famous React UI frameworks, and uses grid-based layouts, responsive animations and transitions, padding, and depth effects such as lighting and shadows. It provides React components that implement Google Material Design [21], that are inspired by the physical world objects and their textures on how they reflect light and cast shadows.

It provides comprehensive, modern UI components that work across the web, mobile and desktop, that are fast and consistent and that were fully tested across modern browsers. Material-UI components also work without any additional setup, working in isolation and they are self-supporting, injecting only the styles they need to display, i.e. the global scope is not affected.

All web client front-end was built using Material-UI components.

### 5.1.2 CodeMirror

CodeMirror [22] is a versatile text editor implemented in JavaScript. It is specialized for editing code, and comes with a number of languages modes and addons that implement custom UI themes or more advanced editing functionality such as auto close brackets or auto matching tags that will cause the tags around the cursor to be highlighted.

Some of these features seem only “nice to haves”, but they fulfill an important role in the concept of our application, which is be educational driven and to make the learning process easier.

In the context of the developed e-learning platform, the CodeMirror library is used in all the built-in components that use a text editor, such as the Challenges interface where one can write the code to be executed.

### 5.1.2 Formik

Formik is a lightweight, easy to use form helper library which is concerned in helping with 3 particular points: getting values in and out of form state, validation and error messages; handling form submission [23]. In React forms are usually verbose with a lot of boilerplate, with Formik this issue is mitigated making the code more readable and easier to maintain or change.

Another advantage is the integration this library has with Material UI, which is used on this project. This requires the use of another library “formik-material-ui” and allows the use and configuration of some Material UI components when building a form [24].

### 5.1.2 Yup

The Yup library is a lightweight, widely used, client-side JavaScript schema builder for value parsing and validation [25]. This allows validation of any type of object in javascript, with plenty out of the

box validations for number types and string types, and also allows the creation of custom validators making this library very flexible.

On the context of this project it is used for validation when building Formik scripts making the validations even less verbose [26].

## 5.2. Services

This module is an application which exposes a REST API, enabling its clients to interact with the domains identified during the requirement section.

In Figure 8 it is shown in more detail how the service modules are structured. There are 5 main sub modules: Users; Authentication; Execute code; Challenges; Questionnaires.

This module is developed as a Spring Boot application using the Kotlin language and the Gradle framework as a build and dependency management tool. The database is a Postgres relational database and the API is documented with the Open API 3.0 standard [27] hosted on Swagger UI [28].

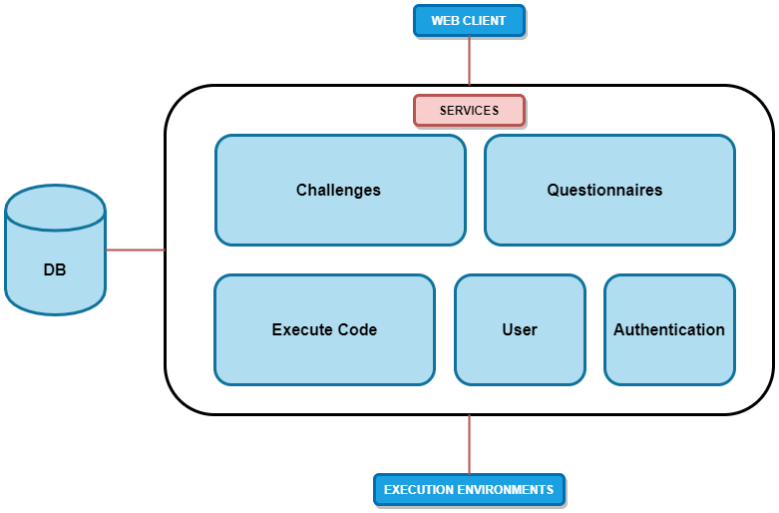


Figure 8 – Detailed view of Services Module including DB

The Users submodule is responsible for interaction and business logic with Users domain.

The Authentication module is responsible for allowing user basic authentication and managing endpoint authentication for the whole application.

The Challenges submodule is responsible for interaction and business logic with Challenges and Challenge Answers domains.

The Questionnaires submodule is responsible for interaction and business logic with Questionnaire and Questionnaire Answers domains.

The Execute Code submodule is responsible for handling code execution requests redirecting the requests to the correct execution environment depending on the language.

Another detailed shown on the picture above is the Database. The services module is the only module with access to the database and it is responsible for directly connecting this database which maintains the state application for the different domains.

To store the platform data we are relying in PostgreSQL, which is an open source object-relational database, well known for its strong reliability, feature robustness and performance.

### 5.2.1 Data Model

The data model reflects the necessary structure to comply with the functional requirements and other support structures necessary to the application. Since the database is relational, the design of the data model was done using an Entity Relationship Diagram, which can be seen on Annex B.

To follow good practices of data model design this data model follows 3NF rules for normalization [29] [30], below is a more detailed explanation of what represents each entity.

App user - This entity represents a user on the platform.

Code Language - This entity contains the different coding languages supported by the platform

Challenge - This entity on the data model maps directly to the challenges identified on the requirements section Requirements. The challenge has a many to one relationship with the user table, this relationship represents the creator of the challenge which can create multiple challenges. This is a weak entity of user, i.e., does not exist if there is no user.

Challenge Solution - This entity represents the solution to the challenge. It is on a separate table to allow defining multiple solutions per Challenge (many to one relationship), one for each language. This is a weak entity of Challenge, i.e., does not exist if there is no Challenge. This table has a one to many relationship with the code language table because each answer is written for a specific supported language.

Challenge Answer - This entity represents an answer of a challenge. There is a many to one relationship with the challenges since a challenge can be answered by multiple different people, as a result this table also has a many to one relationship with the user table. This is a weak entity of Challenge, i.e., does not exist if there is no Challenge.

Questionnaire - This entity on the data model maps directly to the questionnaires identified on the requirements section Requirements. The questionnaire has a many to one relationship with the user

table, this relationship represents the creator of the questionnaire which can create multiple questionnaires. This is a weak entity of user, i.e., does not exist if there is no user.

Questionnaire Answer - This entity represents an answer of a questionnaire. There is a many to one relationship with the questionnaires since a questionnaire can be answered by multiple different people. This is a weak entity of Questionnaire, i.e., does not exist if there is no Questionnaire.

Answer - This entity represents the abstract concept of an answer. The type of the answer is determiner through the mandatory mutually exclusive relationships between Answer and it's "children", Challenge Answer and Questionnaire Answer. This was done to normalize answer related data since both challenges and questionnaire answers share data but have specificity to their domain. This was enforced on a database level through the usage of triggers. This table has a one to many relationship with the code language table because each answer is written for a specific supported language.

Tag - This entity supports the tag search functionality identified on the requirements section Requirements. There is a one to many relationship with the challenges since a challenge can have multiple tags.

One special connection is also worthy of note, the many to many connection between the tables Challenge, Questionnaire and Questionnaire Answer. This relationship exists in order to support a questionnaire associating to many Challenges each with a language (it could only be solved for a specific language even if it is available with more) and also associating the questionnaire answer to the challenge connected with questionnaire.

## 5.3. Execution Environments

This module contains multiple containers, these containers are grouped in container execution managers and have grouped containers by runtime, i.e. there will be a container execution manager for each type of runtime environment. The scalability of this module is provided with the execution manager, providing more containers to the manager will increase capacity.

The goal of this module is to make it possible to execute external code send by this module's clients while supporting multiple runtime environments.

As can be seen on Figure 9, this was achieved by having multiple applications running in separate containers, each container supporting a single runtime. Each application is listening to HTTP requests to execute the code, and when it receives a request it compiles the code (if necessary), executes the code and returns the result of the execution.

These applications all share the same API contract, this means the clients only needs to respect the contract and send the request to the correct application (endpoint) depending on the runtime of the code to be executed. This allows the clients to be abstracted from any implementation, increasing to

modularity of the solution. If the need arises to change a specific runtime environment or even add more it would be a seamless change.

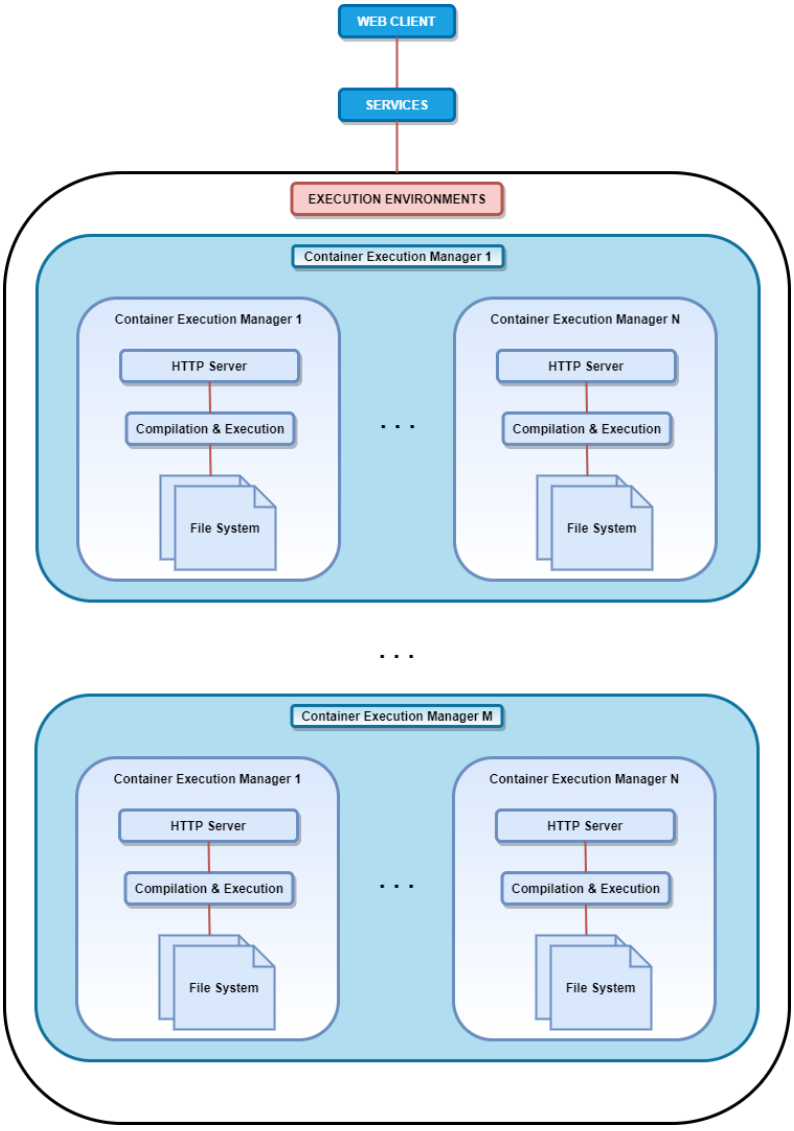


Figure 9 – Detailed view of ExecutionEnvironments Module

Docker will be used to build and run containers for each execution environment application and 5 runtime environments will be supported: Java, Kotlin, JavaScript, C#, Python.

# 6. Implementation Details

This chapter describes some relevant implementation details regarding the modules described in chapter 5.

## 6.1. Services

The services Users, Execute Code, Challenges and Validations have implementation details which are specific of each service but also share other cross-service implementations like for input validation and error handling, below are described such implementation details.

### 6.1.1 Users

User service is responsible to interact with user domain models and enforce business logic. Database records are accessed through an implementation of Spring's interface CrudRepository, that provides several methods to access this information. More details for each method and data structures can be found on this project's Swagger documentation [28].

### 6.1.2 Execute code

The scope of the Execute submodule is very specific, its purpose is to receive requests to execute code remotely and if the request language is supported redirect the request to the execution environment module where the execution of the code will take place. The contract exposed by the controller of this submodule can be checked on the Swagger documentation [28].

This submodule needs a property file to work properly named "executionEnvironments.properties". This properties file has information about the endpoint to which the redirection of the execution requests is sent, more details of each property can be found on the wiki of this project's repository [31].

### 6.1.3 Challenges

The Challenges submodule is responsible for interaction and business logic with Challenges and Challenge Answers domains.

The scope of the Challenges submodule includes the domains of Challenges, Challenge Answers, Challenge Tags and Tags.

Implementation wise each of these domains exposes a different Spring RestController and has a different service to handle the business logic. The possible operations exposed by these controllers can be found in the Swagger documentation [28].

The Challenge domain entity is represented by a combination of the data model entities Challenge and Challenge Solution, i.e. a challenge on the service module contains information not only about the challenge itself but also about its solutions.

The Challenge Answers domain entity is represented by a combination of the data model entities Challenge Answer and Answer, this is only natural since the Answer data model entity exists as part of a mandatory mutually exclusive relationship with different types of answers, i.e. every type of answer is related to an Answer data model entity.

The Challenge Tags domain entity is represented by a combination of the data model entity Tag and a record from the many to many entity CT, which represents the associations between a Tag and a challenge data model entities.

The Tags domain entity is represented by a data model entity Tag.

### 6.1.4 Questionnaires

The Questionnaires submodule is responsible for interaction and business logic with Questionnaire and Questionnaire Answers domains.

The scope of the Questionnaires submodule includes the domains of Questionnaire, Questionnaire Instance, Questionnaire Answers and Questionnaire-Challenge.

Implementation wise each of these domains exposes a different Spring RestController and has a different service to handle the business logic. The possible operations exposed by these controllers can be found in the Swagger documentations [28].

The Questionnaire domain entity represents the wrapper of all questionnaire components. It contains the information about the questionnaire domain but also references all the challenges in the questionnaire. The questionnaire itself cannot be solve, but it serves as a template in which instances of it can be created and send to multiple users to be solved.

The Questionnaire Instance, extends its parent Questionnaire, and it is this resource that is going to be sent to a user for further operations. Multiple Questionnaires Instance can derive from a single parent Questionnaire.

The Questionnaire-Challenge domain entity is represented by a combination of the data model entity Questionnaire and a record from the many to many entity QC, which represents the associations between a questionnaire and a challenge data model entities, and the answer to the challenge present in the questionnaire.

The Questionnaire Answers domain entity is represented by a combination of the data model entities Questionnaire Answer and Answer. The questionnaires answer in data are equal to the challenges answer ones since they both aim to resolve a challenge problem. They also respect the same mandatory mutually exclusive relationship with different types of answers, with the difference that they are bound to a different resource.

### 6.1.5. Validations

The parameter validation is done on the service layer using the Javax validators. For these validators to work the service classes and methods need to be annotated with `@Validated` annotation. The validations were done on the service classes's methods, for each input parameter. This was done with annotations from package `javax.validation.constraints` for specific validations on String and Number types, e.g. `@Positive` , and using the annotation `@Valid` for other custom Reference types. For these custom reference types the class also had annotations from package `javax.validation.constraints` on fields which were to be validated.

### 6.1.6. Error Handling

Error handling responsibility in this solution is divided in several classes.

The class ControllerAdviceHandler was marked with `@ControllerAdvice` annotation to enable application wide exception handling in a single class. This class has method to capture two application specific Exceptions (ServerException and ConstraintViolationException) and the most generic "Exception".

The basic flow in this class is as follows: an exception will be caught and then will be mapped to ApiError's instance that maps to the json+problem standard [28].

The class ServerExcepton extends Exception type, and its instantiated when an error occurs. This object is constructed with a message for the client, a message for the developer and an error code, which is mapped to an http error code when returning to the client.

### 6.1.7 Postman

For documentation and test purposes a Postman collection was developd containing examples of working requests for developed services. This enables the developers to test services easily and provides a documented example of a working request to each service endpoint.

## 6.2. Data base

To configure the database a single master script “CreateDB.sql” was created, and can be found on the Wiki of the project repository [31].

This script includes not only the creation of the model with all its tables and dependencies, but also all the triggers and store procedures that allow a robust consistency on the database on the insertion of new data.

The single master script was created through the merge of several individual ones used on the development phase such as scripts to create, delete, drop and fill the database. Adding to this there is also a test script for the database. The other scrips can be found as store procedures on the project repository.

### 6.2.1. Data base access

For database access to be possible a data source been was created to configured database related configurations, allowing a database connection to be used by the application to communicate with a given DB.

To configure the database connection a properties file with the name "application.properties" must be added to the resource folder. An example of its configuration can be found on the wiki repository [31].

On the application spring Repositories were used to interact with the database, specifically CrudRepositorty and PagingAndSortingRepository. The CrudRepositorty is a repository with some CRUD operations already implemented and the PagingAndSortingRepository extends the CrudRepositorty supporting sorting and paging for DB operations.

To enable the repositories to perform database operations each repository was associated with a given class which represented the DB tables with which the repository would interact. As such these classes needed some configuration so the repository would be able to know the table name, the field names and other related entities were there. To this effect the package `pt.iselearning.services.domain` contains several classes which were annotated with annotations from `org.hibernate.annotations` and `javax.persistence` packages in order to identify DB table names, column names, which of the properties were primary keys, identity keys and relations to other domains line one to many or many to one.

## 6.3. Execution Environments

The implemented execution environments share some implementation details but for the most part require different dependencies and configurations.

In order to have fewer dependencies inside the container each container has only one runtime execution, which means the environment in which the code is meant to be executed will have to be the same on which the application will have to run. To specify, for the Java execution environment

there will be a Java application listening to HTTP requests, for the C# execution environment there will be a C# application and so on. As a result, each application will use specific technologies. The technology in common between every application is Swagger, which will be used to document the REST API shared amongst every application [28].

Of the supported languages 3 of them already have the execution environments developed and are described in more detail below: Java, Kotlin and JavaScript.

The remaining 2, C# and Python are not yet developed but are planned to be supported by the end of this project.

### 6.3.1. Java & Kotlin

Because both Java and Kotlin can be compiled to be executed on the JVM, the same application was used for both execution environment, with minor changes for each.

For these execution environments the application executed inside a Docker container is a Spring Boot application developed in Java using Maven as a build and dependency management tool.

The application is a simple one by design, once the application receives an HTTP request determines if there is the need to execute the code or the unit tests, writes the code to the file system, compiles the files and executes it. Both the compilation and the execution processes are done by executing bash or command line commands depending if the system is running on windows or Linux system. After the execution is complete with error or not, the result of the execution which was dumped to a text file is returned.

One of the main differences between the Java and Kotlin execution Environments is the environment on which the applications are executed, i.e. the docker container have different dependencies.

For the Java execution environment, the container is built on top of the OpenJDK 13 docker image, this allows the java application to run and the commands to compile and execute Java code to work.

For the Kotlin execution environment is not as simple, besides needing the JDK to run the Java application and executing Kotlin code compiled to the JVM it also needs the Kotlin compiler. This container was also built on top of the OpenJDK 13 docker image but the docker file also contained instructions to download and install the Kotlin JVM compiler.

### 6.3.2. JavaScript

For this execution environment was developed a NodeJS application that receives an HTTP message and based on the request body creates a file with to code that need to be executed and a file that contains the unit tests, the latter imports the function exported in the running code file to run the tests. With the file(s) created, a child process is spawn to run the defined code or the unit tests if defined in the message body.

The container is built on top of the node 14 image to enable said docker to run the mentioned application.

### 6.3.3 Postman

The execution environments all share the same HTTP API contract, even so each environment is a different application and handles the requests differently. With this in mind a Postman collection was created containing a documented example of a working request for each of the developed execution environments.

# 7. Additional language support

Adding support to a new language requires changes on two elements of this solution: Execution Environments and Service layer.

For the latter, a new entry must be added to the languageUrlMap, with the key being the new language and the value the address and port of the machine running the new execution environment. It is also necessary to add the supported language to the enum SupportedLanguages and update the "executionEnvironments.properties" file to reflect the new execution environment implemented.

The Execution Environment is basically an application that provides exposes one endpoint, which will be used to run code and/or unit tests remotely.

This endpoint must respect a specific contract. The endpoint must have a parameter that contains a field named "code" which is a string that represents the code that the user wants to run, a field named "executeTests" which is a Boolean that represents if the user wants to test the code being sent against the unit tests defined and a field named "unitTests" which is a string that contains the unit tests to run with the code sent by the user.

This endpoint must return a structure that contains a field named "rawResult" which is a string that represents either the correct result of the code submitted by the user or the errors that appeared while compile/running the code.

For this project it was devised that the Execution Environments would run in a container driven deployment methodology and as such in order for the new execution environment to run the same way a new docker file with the necessary commands to install any dependencies needs created, these commands also need to enable the container to expose a port so that the a new endpoint can be used by the service application.

# 8. Project progress

The project continues to be mostly on schedule. Some time constraints have impacted the planned schedule such as jobs or college projects for other courses, specifically projects for other courses have caused significant delay on some developments, resulting in more tasks behind schedule than was the case on the progress report delivery.

On Figure 10, which represents the planned scheduled up to the progress beta delivery, are highlighted the activities which have not been finished. The development of the Questionnaire service has proven more complex than anticipated, and although it is behind schedule it is already in development, as is the Authentication Service. The Fill Questionnaire page have not begun development yet.

On Table 2 is the necessary information to understand the scheduled plan on Figure 10 and Figure 11.

Table 2 – Assignment Types

| Assignment Type | Description |
|---|---|
| **E(x)** | Design and implementation of execution environment for a given language |
| **P(x)** | Creation of webpage |
| **S(x)** | Design and Implementation of backend service |

This means the remaining activities are finished, summarizing:

1. The Database is set up and documented
2. React framework was configured for development
3. JVM execution environment is developed and documented, including for Java and Kotlin
4. The page on which the user could execute code is developed
5. The home page is developed
6. The progress report was finished and each team member has an individual presentation prepared
7. Challenges Service
8. User Service
9. User's Profile page
10. Beta version was ready for delivery

| Date | Assignment | Milestone |
|---|---|---|
| 09-Mar | Introduction to docker, react, bash, python | |
| 16-Mar | Set up DB, webserver react configuration, E(JVM) | Project Proposal 16 Mar |
| 23-Mar | Set up DB, webserver react configuration, E(JVM) | |
| 30-Mar | E(JVM), E(Node), P(Execute code) | |
| 06-Apr | E(JVM), E(Node), P(Execute code) | |
| 13-Apr | E(JVM), E(Node), P(Home) | |
| 20-Apr | Progress Report and Individual presentation | |
| 27-Apr | Progress Report and Individual presentation | |
| 04-May | S(Challenges), P(Home), S(User) | Progress Report and Individual presentation 4 May |
| 11-May | S(Challenges), S(User), S(Questionnaire) | |
| 18-May | S(Challenges), P(User's Profile), S(Questionnaire) | |
| 25-May | Poster and Beta version, S(Questionnaire) | |
| 01-Jun | S(Challenges), P(User's Profile), S(Questionnaire) | Poster and Beta version 1 June |
| 08-Jun | P(Login and Signin ), P(Fill Questionnaire), S(Authentication) | |

Figure 10 – Planned Schedule before beta delivery

Despite this set back, the plan is on the right path for a full delivery within the specified timeline.

On Figure 11 is the planned schedule from the progress report delivery date onwards. An important detail regarding this plan is the login and sign in page, which was planned to be finished on the June 15th have already been finishing, meaning there is also one task finished ahead of schedule.

| | | |
|---|---|---|
| 15-Jun | P(Login and Signin ), P(Fill Questionnaire), S(Authentication) | |
| 22-Jun | Exams | |
| 29-Jun | | |
| 06-Jul | | |
| 13-Jul | | |
| 20-Jul | | |
| 27-Jul | P(Challenge), E(Python), E(CLR) | |
| 03-Aug | P(Challenge), E(Python), E(CLR) | |
| 10-Aug | P(Challenge), E(Python), E(CLR) | |
| 17-Aug | Cloud environment deployment | |
| 24-Aug | Cloud environment deployment | |
| 31-Aug | Wrap up final report | |
| 07-Sep | Wrap up final report | Final delivery 12 Sept |

Figure 11 – Planned Schedule after progress report delivery

Regarding the remaining tasks there is some uncertainty, specifically on execution environments tasks for Python and CLR since some of the technologies necessary to perform those tasks are not yet well known and regarding the cloud environment deployment for similar reasons, the technology is not yet chosen (cloud provider) and the technology is not well known by the team members.

Also worthy of note are the changes of official dates of the exam season which was moved now starts and finishes one week later (June 29th – July 31st), the beta delivery was also delayed and it was on June 15th and the final delivery which is on the September 26th. While this has caused some re-

planning the extra weeks before the final delivery will provide a cushion for further the current delays and maybe allow extra implementation.

On a brighter note development capacity is predicted to pick up starting on July 31$^{th}$ once the exams are finished and the team will no longer have workload related to other courses.

# 9. Lexicon

API – Application Program Interface

CLR – Common Language Runtime

DB – Database

ISEL – Instituto Superior de Engenharia de Lisboa

IT – Information Technology

HTML – Hypertext Markup Language

HTTP – Hypertext Transfer Protocol

JDK – Java Development Kit

JSX – JavaScript XML

JVM – Java Virtual Machine

REST – Representational State Transfer

SPA – Single Page Application

UI – User Interface

# 10. References


[1] S. S. Matthias Muller-Hannemann, Algorithm Engineering: Bridging the Gap Between Algorithm Theory and Practice, Springer; 2010 edition, 2010.

[2] K. Wiegers, Software Requirements (Developer Best Practices), Microsoft Press, 2013.

[3] "AlgoExpert," [Online]. Available: https://www.algoexpert.io/product. [Accessed 25 04 2020].

[4] "HackerRank," [Online]. Available: https://www.hackerrank.com/. [Accessed 25 04 2020].

[5] "LeetCode," [Online]. Available: https://leetcode.com/. [Accessed 25 04 2020].

[6] "CodeWars," [Online]. Available: https://www.codewars.com/. [Accessed 25 04 2020].

[7] "CodeChef," [Online]. Available: https://www.codechef.com/. [Accessed 25 04 2020].

[8] "React – A JavaScript library for building user interfaces," [Online]. Available: https://reactjs.org. [Accessed 24 04 2020].

[9] A. Banks, Learning React: Functional Web Development with React and Redux, O'Reilly Media, 2017.

[10] "Introducing JSX – React," [Online]. Available: https://reactjs.org/docs/introducing-jsx.html. [Accessed 24 04 2020].

[11] "Spring Framework," [Online]. Available: https://spring.io/projects/spring-framework. [Accessed 24 04 2020].

[12] "2. Introduction to the Spring Framework," [Online]. Available: https://docs.spring.io/spring/docs/4.3.x/spring-framework-reference/html/overview.html. [Accessed 24 04 2020].

[13] "Spring Boot," [Online]. Available: https://spring.io/projects/spring-boot. [Accessed 04 24 2020].

[14] "API Documentation & Design Tools for Teams | Swagger | Swagger," [Online]. Available: https://swagger.io/. [Accessed 24 04 2020].

[15] "Empowering App Development for Developers | Docker," [Online]. Available: https://www.docker.com/. [Accessed 17 04 2020].

[16] "Docker Documentation | Docker Documentation," [Online]. Available: https://docs.docker.com/. [Accessed 17 04 2020].

[17] S. M. Jain, Linux Containers and Virtualization - A kernel perspective, Independently published, 2019.

[18] "Docker Hub," [Online]. Available: https://hub.docker.com/. [Accessed 17 04 2020].

[19] "What is REST – Learn to create timeless REST APIs," [Online]. Available: https://restfulapi.net/. [Accessed 01 05 2020].

[20] "Material-UI," [Online]. Available: https://material-ui.com/. [Accessed 13 06 2020].

[21] "Material Design," [Online]. Available: https://material.io/design. [Accessed 13 06 2020].

[22] "CodeMirror," [Online]. Available: https://codemirror.net/. [Accessed 13 06 2020].

[23] "Formik · Build forms in React, without the tears.," [Online]. Available: https://jaredpalmer.com/formik/. [Accessed 06 2020].

[24] "Formik Material-UI," [Online]. Available: https://stackworx.github.io/formik-material-ui/. [Accessed 06 2020].

[25] "yup - npm," [Online]. Available: https://www.npmjs.com/package/yup. [Accessed 06 2020].

[26] "Tutorial · Formik," [Online]. Available: https://jaredpalmer.com/formik/docs/tutorial#schema-validation-with-yup. [Accessed 06 2020].

[27] "OpenAPI-Specification/3.0.2.md at master · OAI/OpenAPI-Specification," [Online]. Available: https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.2.md. [Accessed 27 04 2020].

[28] "IS E-Learning Swagger UI," [Online]. Available: https://joaoesantos.github.io/ise_learning/apiDocumentation. [Accessed 27 04 2020].

[29] "Database - Third Normal Form (3NF) - Tutorialspoint," [Online]. Available: https://www.tutorialspoint.com/sql/third-normal-form.htm. [Accessed 27 04 2020].

[30] S. B. N. a. R. Elmasri, FUNDAMENTALS OF DATABASE SYSTEMS, Pearson Education, Inc., 2004.

[31] "Home · joaoesantos/ise_learning Wiki," [Online]. Available: https://github.com/joaoesantos/ise_learning/wiki. [Accessed 2020].

[32] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Algorithm_engineering. [Accessed 20 04 2020].

[33] "Spring Security," [Online]. Available: https://spring.io/projects/spring-security. [Accessed 24 04 2020].

# 11. Annex

## Annex A. Supported versions of container dependencies

| Container | Dependency | Supported Version |
|---|---|---|
| **Java Execution Environment** | Open JDK | 13 |
| **Kotlin Execution Environment** | Open JDK | 13 |
| **Kotlin Execution Environment** | Kotlin compiler | 1.3.71 |
| **Javascript Execution Environment** | Nodejs Runtime | 14.0.0 |

# Annex B.Data Model

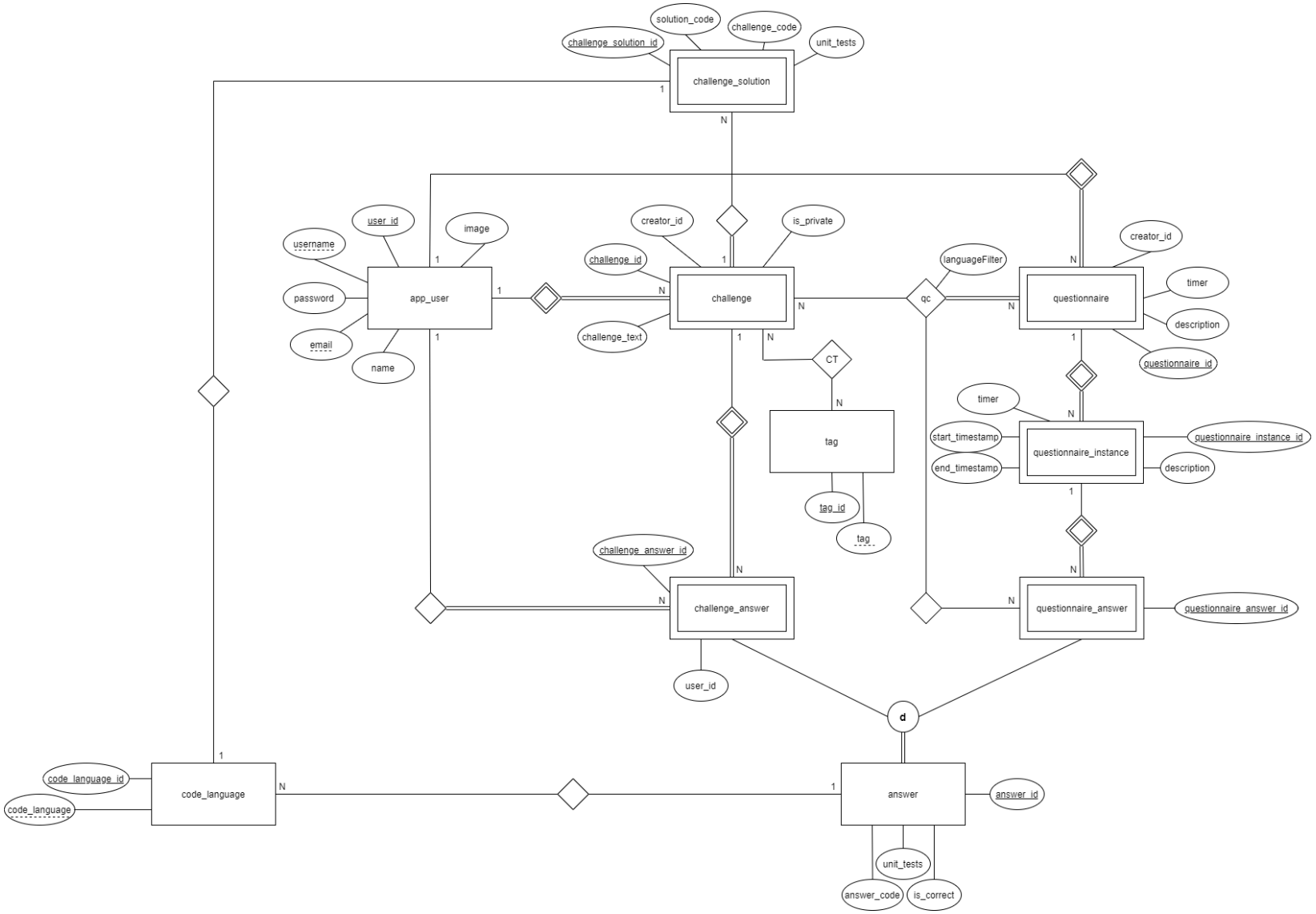


Figure 12 – Data model