CodeKataBattle

Implementation and Test
deliverable documentation

Software Engineering 2 project
Academic year 2023 - 2024

27 december 2023
Version 2.0

Authors:
Tommaso Pasini
Elia Pontiggia
Michelangelo Stasi

Professor:
Matteo Camilli

Revision History

Date	Revision	Notes
27/12/2023	v.0.0	Document creation
4/2/2024	v.1.0	First release
9/2/2024	v.2.0	Completition of installation guide after a peer review

Contents

1. Description of the software

This document contains the description of the implementation of the CodeKataBattle platform.

Its purpose is to provide a complete and detailed explanation of the system and its structure to the developers who will work on the project (in our case, the members of other teams and our professors). It is also a useful reference for the developers who will maintain the system in the future.

1.1 Links

In the following list are provided the links to all the documents that are part of the CodeKataBattle project's implementation

- Source Code
- executable ready to be run

1.2 Implemented features

In the following, we list the features and the requirements that have been implemented in the system.

Since the group that has implemented the system is composed of three people, as the assignment required, we have not implemented the features regarding the ganmification aspects of the system

EDU functional requirements

ID	Description	Implemented	Notes
R1	The software shall allow the unregistered EDUs to create an account.	yes	

R2	The software shall allow the registered EDUs to log in.	yes	
R3	The software shall allow the authenticated EDUs to create new tournaments.	yes	
R4	The software shall allow an authenticated EDU to permit other authenticated colleagues to create battles in its tournament, those become owners of the tournament as much as who added them.	yes	
R5	The software shall allow the authenticated EDUs to create coding battles in their tournaments by letting them upload the Code Kata, setting the minimum and maximum number of STUs per group, registration and final submission deadlines, and the score configurations.	yes	
R6	The software shall allow the authenticated EDUs to manually evaluate the work done by STUs subscribed to their own Code Kata battle.	yes	
R7	The software shall allow the authenticated EDUs to see the sources produced by each team participating in their tournaments.	yes	directly visibile on the GitHub repository
R8	The software shall allow the authenticated EDUs to see the personal tournament score of each STU (which is the sum of all battle scores received in that tournament).	yes	
R9	The software shall allow the authenticated EDUs to see a rank that measures how a STU's performance compares to other STUs in the context of that tournament.	yes	

R10	The software shall allow the authenticated EDUs to see the list of ongoing and finished tournaments as well as the corresponding tournament rank.	yes	
R11	The software shall allow the authenticated EDUs to see the list of ongoing and finished battles as well as the corresponding battle rank within a tournament.	yes	
R12	The software shall allow an authenticated EDU to close a tournament iff it is one of the owners of that tournament.	yes	
R13	When the authenticated EDU creates a tournament, the software shall allow it to define gamification badges concerning that specific tournament.	No	Not requested
R14	The software shall allow the authenticated EDU to create new badges and define new rules as well as new variables associated with them.	No	Not requested
R15	The software shall allow all authenticated EDUs to visualize the badges created in CKB by other EDUs.	No	It is a consequence of the badge feature
R16	The software shall allow all authenticated EDUs to visualize STUs' profile where they can see their collected badges, and some personal information.	Not quite	The profile just shows the list of tournaments and battles the student is subscribed to

Table 1.1: EDU's requirements

STU functional requirements

ID	**Description**	**Implemented**	**Notes**

R17	The software shall allow the unregistered STUs to create an account.	yes		
R18	The software shall allow the registered STUs to log in.	yes		
R19	The software shall allow the authenticated STUs to form teams by inviting other STUs respecting the minimum and maximum number of STUs per group set for that battle.	yes		
R20	The software shall allow the authenticated STUs to join a team by an invite.	yes		
R21	The software shall allow the authenticated STUs to subscribe to a tournament until a certain deadline.	yes		
R22	The software shall allow the authenticated STUs subscribed to a coding battle to upload their work until the final submission deadline of that Code Kata battle.	yes		The GitHub pull gets ignored
R23	When the registration deadline of a battle expires, the software shall create a GitHub repository containing the Code Kata.	yes		
R24	The software shall send to all authenticated STUs who are members of subscribed teams to a battle the link to a GitHub repository containing the Code Kata.	yes		
R25	The software shall run the tests on executables pushed by a team, it shall also calculate and update the battle score of the corresponding team.	yes		
R26	At the end of the consolidation stage of a specific battle *'b'*, the software shall send a notification to all authenticated STUs participating to *'b'* when the final battle rank becomes available.	yes		

R27	The software shall allow the authenticated STUs to see the list of ongoing and finished battles as well as the corresponding battle rank, iff they are part of the tournament.	yes	
R28	The software shall allow all authenticated STUs to see the personal tournament score of each STU (which is the sum of all battle scores received in that tournament).	yes	
R29	The software shall allow all authenticated STUs to see a rank that measures how a STU's performance compares to other STUs in the context of that tournament.	yes	
R30	The software shall allow all authenticated STUs to see the list of ongoing and finished tournaments as well as the corresponding tournament rank.	yes	
R31	The software shall notify all authenticated STUs involved in a closed tournament when the final tournament rank becomes available.	yes	
R32	The software shall allow all authenticated STUs to visualize other STUs' profile where they can see their collected badges, and some personal information.	Not quite	The profile just shows the list of tournaments and battles the student is subscribed to

Table 1.2: STU's requirements

1.3 Adopted development framework

For a thorough exposition of the pros and cons associated with the languages in question, it is recommendend to search directly on the official websites of said languages, where such aspects are explicated in detail

Backend

The system has been developed using the **Spring Boot** framework, which is a framework that allows to create web applications in Java.
The main reason for this choice is that it is a framework often used in this context since it allows to create a web application with REST APIs in a simple way.
In addiction, there are a lot of libraries that can be used to integrate the framework with other technologies, such as the database, which is very useful.

In order to perform the static analysis of the code, we have used **SonarQube**, which is a tool that allows to perform a static analysis of the code and to find the bugs and the vulnerabilities.

Frontend

For the development of the frontend, we have implemented a single page application simply using **HTML, SCSS and JavaScript**.
Even it is a dated approach, we have chosen this solution because it is the one that we are most familiar with and because it is the one that we have used in the previous projects. Moreover, since the application is very simple, we have not felt the need to use a more complex framework.

Database

The database has been implemented using **MySQL**.
The main reason for this choice is that it is the most used database in the world and it can be easily integrated with the Spring Boot framework thanks to the **Spring Data JPA** library.

1.4 Structure of the source code

The source code of the system can be found in the ITD folder of the repository.
Its structure is the classical one of a Spring Boot application, which is the following:

- `src/main/java/ckb/platform/`: contains the source code of the spring application, divided in the following subfolders:
  - `advices/`: contains the classes that are used to insert into the response the error messages in case of exceptions
  - `controllers/`: contains the classes that are used to handle the requests and are the responsible of the communication between the APIs and the database
  - `entities/`: contains the JPA classes that represent the entities of the database
  - `exceptions/`: contains the classes that are used to handle the exceptions

  - `formParser/`: contains the java beans that are used to parse the forms of the POST requests
  - `githubAPI/`: contains the class that is used to communicate with the GitHub API
  - `repositories/`: contains the interfaces that are used to communicate with the database
  - `scheduler/`: contains the classes that handle the additional threads that keeps the time updated and, whenever a deadline is reached, that send the emails to the interested users
  - `testRepo/` contains the class that is the responsible of the building of the code and the execution of the tests

- `src/main/resources/static/`: contains the files of the frontend application, i.e. the HTML, SCSS and JavaScript files

2. Performed test

The software testing procedure has been performed mainly thorough unit testing; below are listed the most relevant tests that have been performed.
Due to the limited time, we have focused on the backend system, since the frontend is very simple and the majority of the functionalities are implemented in the backend.

Note: Regarding any kind of misuse of the system, the frontend already has some all the necessary checks to avoid any kind of misuse of the system (for instance, it is not possible to subscribe to a tournament that is already closed, it is not possible to subscribe to a battle that is already closed, but also all form is equipped with the necessary type checks and length checks).
Furthermore, in the RASD we have already assumed from the domain that the user won't try to misuse the system.
Nonetheless, the backend provides all the checks already implemented in the frontend, so that the system is safe from any kind of misuse, and returns the right error messages in case of misuse.

2.1 Tournament

1. an EDU creates a tournament, with a new name or a name that already exists
2. a STU subscribes to a tournament
3. an EDU closes a tournament

2.2 GitHub / Static Analysis / mail sending

1. the system sends an email to the right users when a deadline is reached or a significant event happens
2. the system creates a GitHub repository at the end of the registration phase of a battle
3. after a push, the system gets notified by GitHub, retrieves the code in the form of a zip file, unzips it, runs the tests and updates the score

2.3 Battle

1. an EDU creates a battle, selecting one or more options regarding static analysis and manual evaluation
2. a STU subscribes to a battle, both alone and inviting other STUs
3. a STU invites another STU to a battle after being subscribed
4. a STU joins a battle by an invite
5. a STU uploads the code on GitHub and sees his score updated, taking into account the tests, the timeliness and the eventual static analysis
6. an EDU performs the manual evaluation of the code of the teams in a battle

2.4 Profile

1. an user inspects the profile of a STU's profile showing the list of tournaments and battles the student is subscribed to

3. Installation instructions

3.1 Java

In order to run the application, it is necessary to have Java installed on the machine.

- **Java Runtime Environment**: Download JDK: at least version 21
- **Java Development Kit**: Download JRE: at least version 17

Once the installation is completed, it is necessary to set the environment variables (the way to do this depends on the operating system).

3.2 MySQL

In order to run the application, it is necessary to have MySQL installed on the machine and to create a database.
You can download it from the official website.
Required modules to be installed:

- MySQL Server
- MySQL Workbench (optional, but recommended)
- Connector/J

Once the installation is completed, it is necessary to create a schema called **'ckb-db'**, and create a user with all the privileges on this schema and with the following credentials:

- username: **CKBPlatform**
- password: **CKB202430L!**

In order to work as expected, the database must be running on the default port (3306).

3.3 Static Analysis Tool - SonarQube

In order to use this tool, we need to install SonarQube Community Edition at this link. Then, install a local instance of SonarQube by following these instructions.
In our case we simply run from the zip file, using Windows as a OS and we have installed it in the same directory of the instructions.

Notice that if you are installing it in a different location, you have to insert the correct directory when the application starts.
Java 17 is required to start the local instance of SonarQube.
Last thing, install the SonarScanner for CLI and we can find the download here.

3.4 GitHub API

As our system runs on `https://localhost:8080` and GitHub need to reach a public address, we will use "ngrok". Ngrok generates a tunnel address that need to be used in the WorkFlow Action to reach the End-Point `/ckb_platform/battle/pulls`
In order to make ngrok work properly, follow these instructions:

- Log In with CKB GitHub credentials (mail: **codekatabattle.platform@gmail.com** passeord: **CKB202430l!**) on Ngrok `https://dashboard.ngrok.com/login`
- Go in section Your Authtoken and copy the command: `ngrok config add-authtoken 2bTBjPXyFBY3dy8NlZSJ5LzrC9x_7FRwMrsggt8f68KFpvJfW`
- Ngrok can be downloaded from `https://ngrok.com/download`, unzip it, and open it (CMD will pop up)
- Paste the command you copied before

From now, every time you need to use ngrok, open it and type the command: `ngrok http https://localhost:8080`
A public address will be prompted, this will be used in GitHub Action

In conclusion, whenever a GitHub repo is created and forked, in the readme you will find a template for the action workflow, you simply have to put the tunnel link where you find `«NGROK»`

3.5 pytest

In order to comput the test score in a python battle, the platform uses `pytest`.

The installation is quite straight-forward if you have already python on your machine: it is sufficient to run the command `pip install -upgrade -force-reinstall pytest`

If a warning appears showing a path quute long, it means that that path isn't among the environment variables: it is recommended to add it.

3.6 Running the application

Once all the previous steps are completed, it is possible to run the application.
To run the application, just run the jar file that can be found in the link above and follow the instructions given by the application.

3.6.1 Some notes about the Gmail API

To send mails using gmail to other addresses, Google requires an OAuth authentication system since 2019.
The application from Google is seen as if it were in the testing phase, so we have a token that expires every 7/10 days and must be renewed. We have not made it public because Google requires a series of checks regarding privacy/security etc.

The first time the application is run, it will open a link to which you must access with the credentials provided in the email.
After that, the token will be generated and the application will be able to send emails.
After 7/10 days, the token will expire and the application will show again the link to access to generate a new token.

4. Effort Spent

Team

Topic	**Time**
Final revision before the delivery	4h

Table 4.1: Effort Spent during team meetings

Tommaso Pasini

Topic	**Time**
POST endpoints	20h
External APIs (GitHub and Gmail)	20h
Setting up the test of the uploaded code	30h
Frontend requests refining	10h

Table 4.2: Effort Spent by Tommaso Pasini

Elia Pontiggia

Topic	**Time**
Frontend	35h
http requests exchange	16h
Writing of the document	6h

Table 4.3: Effort Spent by Elia Pontiggia

Michelangelo Stasi

Topic	**Time**
Backend	55h

Table 4.4: Effort Spent by Michelangelo Stasi