POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

ITD Document

Author: **Mattia Piccinato, Gabriele Puglisi, Jacopo Piazzalunga**

Academic Year: 2023-24
Link to the repository: Click here

Contents

1 | Introduction

1.1. Purpose

This document outlines the implementation and testing procedures that have been followed to develop a functioning prototype of the service described in the ”Requirements Analysis and Specification Document” and ”Design Document”.

1.1.1. Definitions

- Battle: A Code Kata, that is, a challenge in which teams of players need to solve a problem in a specific coding language and submit their code to get a score according to the rules of the Battle.
- Tournament: A competition composed of many Battles in which participants' overall score is the sum of all the scores obtained in every Battle they participated in, individually or in team with other players.
- Student: The User which takes part into the Tournaments of Battles.
- Educator: The User which organizes Tournaments of Battles to which the Students can participate and who manages every aspect about them.
- Consolidation stage: The Consolidation Stage is the phase of a Battle which starts as soon as the submission deadline expires, during which the Educators who manage the Tournament can eventually assign an additional score to every team, which will be summed to the score previously assigned by the platform.

1.1.2. Acronyms

- CK: Code Kata, that is, a Battle.
- CKB: Code Kata Battle, that is, the name of the platform.

1.2. Revision History

Revised on	Version	Description
4-Feb-2023	1.0	Initial Release of the document

1.3. Reference Documents

- Assignment document A.Y. 2023/2024
  ("Requirement Engineering and Design Project: goal, schedule and rules")
- Software Engineering 2 A.Y. 2023/2024 Slides
  (Lecture slides provided during the course)

1.4. Document Structure

This document is composed of six sections:

- 1st Chapter: We begin by presenting the problem statement and outlining the system's objectives. We provide essential resources for readers, including definitions and abbreviations, to facilitate a comprehensive understanding of this document.
- 2nd Chapter: We present all the functions of the system which the prototype actually implements.

- 3rd Chapter: We introduce the programming languages and frameworks we have chosen to embrace, providing comprehensive justifications for each selection.
- 4th Chapter: We present the structure of the code.
- 5th Chapter: We provide information regarding the testing process, specifically outlining the functions that have undergone testing and elucidating the methodology employed.
- 6th Chapter: We offer instructions that comprehensively guide users on installing and running the prototype.

2 | Product functions and requirements

The CKB platform provides several functions. For every product feature and each requirement outlined in the RASD, we specify whether the prototype incorporates it.

2.1. Product functions

- Battles and Tournaments Creation Educators can create coding challenges (Battles) within Tournaments, specifying details like descriptions, team sizes, deadlines, scoring configurations and, eventually, Badges. Implemented Yes.
- Student Participation On the other hand, Students can join such Tournaments and take part in Battles, individually or in teams. Implemented Yes.
- GitHub Integration The system allows the Students to perform code submissions just by pushing their code on GitHub, thanks to automated workflows triggered by GitHub Actions service. Implemented Yes.
- Automated Evaluation The platform automatically evaluates Student code based on test cases, timeliness, and code quality using external static analysis tools. Implemented Partially. The prototype offers automated functional evaluation exclusively for Java programs. The static analysis has not been implemented.
- Scoring and Ranking It continuously updates team rankings during Battles and provides overall Tournament rankings at the end of each Battle. Implemented Yes.
- Badges and Recognition Educators define Badges which can be obtained by Students, serving as recognition for accomplishments and participation to a certain Tournament. Implemented No. Badges have not been included in the database design.

- Manual Optional Evaluation Educators can manually evaluate Students' work and assign additional scores at the end of every Battle. Implemented Yes.

2.2. Requirements

R1 The system must allow an unregistered Educator to sign up. Implemented Yes.

R2 The system must allow an unregistered Student to sign up. Implemented Yes.

R3 The system must allow a registered User to log in. Implemented Yes.

R4 The system must allow registered Educators to start the creation process of a Tournament of Code Kata Battles. Implemented Yes.

R5 The system must provide registered Educators of a list of Tournament-related statistics for the Badges definition, during the Tournament creation process. Implemented No. Badges have not been implemented.

R6 The system must provide registered Educators of a specific language which lets them define the Badges, during the Tournament creation process. Implemented No. Badges have not been implemented.

R7 The system must allow registered Educators to grant other registered Educators the permission to manage the Tournament, during the Tournament creation process. Implemented Yes.

R8 The system must allow registered Educators to end the creation process of a Tournament that they started themselves. Implemented Yes.

R9 The system must be able to send notifications to every registered User. Implemented Yes. The system uses an e-mail service to notify Users.

R10 The system must allow a registered Educator to create a Battle in a Tournament if and only if he is the creator of the Tournament or if he was granted the permission to by the latter. Implemented Yes.

R11 The system must allow a registered Student to create a group for a Battle in a Tournament. Implemented Yes.

R12 The system must allow a registered Student to accept an invitation to a group for a Battle in a Tournament. Implemented Yes.

R13 The system must allow registered Students to see the list of ongoing Tournaments and join any of those if its subscription deadline is not expired yet. Implemented Yes.

R14 The system must allow registered Students who are enrolled in a Battle to perform code submissions. Implemented Yes.

R15 The system must be provided of proper APIs to let registered Students perform code submissions through GitHub Actions. Implemented Yes.

R16 The system must update the Battle ranking when a valid code submission is performed. Implemented Yes.

R17 The system must set the Consolidation Stage of a Battle when its submission deadline expires. Implemented Yes.

R18 The system must let Educators to end the Consolidation Stage of a Battle if and only if he is the creator of the Tournament or if he was granted the permission to by the latter. Implemented Yes.

R19 The system must update Tournament ranking when a Battle exits the Consolidation Stage. Implemented Yes.

R20 The system must let Educators who are either the creator of the Tournament or who have been granted the permission to by the latter to close a Tournament if and only if there is no Battle such that either their subscription or submission deadline is not expired yet or such that they are still in the Consolidation Stage. Implemented Yes.

R21 The system must assign an achievements' Badge for a given Tournament to any Student who satisfied the conditions defined by the creator of the Tournament. Implemented No. Badges have not been implemented.

R22 The system must allow every User to see the Badges which were ever obtained by a given Student. Implemented No. Badges have not been implemented.

R23 The system must notify every registered Student about the creation of a new Tournament. Implemented Yes.

R24 The system must notify every registered Student about the creation of a new Battle within a Tournament they are enrolled in. Implemented Yes.

R25 The system must notify every registered Student about the end of a Battle they are participating in. Implemented Yes.

R26 The system must notify every registered Student about the end of a Tournament they are enrolled in. Implemented Yes.

3 | Development and frameworks

3.1. Programming Languages

3.1.1. Java

Java was selected as the backend programming language for our web application due to the following advantages:

- **Platform Independence:** Java's write once, run anywhere (WORA) philosophy allows us to create code that can run on various operating systems without modification, ensuring broad compatibility.
- **Robust Ecosystem:** Java boasts a mature and extensive ecosystem, offering a wide range of libraries and frameworks. This rich set of tools provides a solid foundation for building scalable and maintainable backend solutions.
- **Community Support:** Java enjoys strong community support, with a vast developer community contributing to its continuous improvement. Regular updates from Oracle enhance the language's reliability and security.
- **Object-Oriented Paradigm:** Java's object-oriented programming (OOP) paradigm aligns well with the complexity of web applications. This approach supports the development of modular, extensible, and organized code.
- **Readability and Maintainability:** Java emphasizes readability and maintainability, making it conducive to creating backend code that is not only efficient but also easy to manage over the long term.

Furthermore, the decision to utilize Java for developing the backend of our prototype was influenced by the chance to implement Spring and JPA frameworks, which we anticipated

would expedite the development process.

3.1.2. JavaScript

JavaScript is the standard language used to develop dynamic and interactive web applications. We used it for creating responsive user interfaces and handling client-side scripting for the frontend of out application.

3.2. Frameworks

3.2.1. Spring

The decision to employ the Spring framework for the backend of our application was driven by several advantages:

- **Comprehensive Ecosystem:** Spring provides a comprehensive ecosystem that facilitates the development of robust and scalable applications. Its modular design allows us to selectively use components, enhancing flexibility.
- **Dependency Injection:** The built-in support for dependency injection in Spring promotes loose coupling, making the codebase more maintainable and testable.
- **Aspect-Oriented Programming (AOP):** Spring's AOP capabilities enable us to separate cross-cutting concerns, enhancing code modularity and readability.
- **Integration Capabilities:** Spring offers seamless integration with various technologies and frameworks, providing a versatile platform for building enterprise-level applications.
- **Spring Boot:** The use of Spring Boot simplifies the configuration and deployment processes, allowing for rapid development of production-ready applications.

Figure 3.1: Spring Framework

3.2.2. JPA

The adoption of the Java Persistence API (JPA) for our backend is based on the following key benefits:

- **Database Independence:** JPA provides a standardized way to interact with databases, allowing us to switch databases easily without major code modifications.
- **Object-Relational Mapping (ORM):** JPA's ORM capabilities simplify the mapping of Java objects to database entities, streamlining database interactions and reducing development time.
- **Portability:** The portability offered by JPA allows our application to run on different JPA-compliant persistence providers, enhancing flexibility and minimizing vendor lock-in.
- **Transaction Management:** JPA's built-in transaction management ensures data integrity by providing a mechanism for handling database transactions.

  In our implementation, we specifically utilized Hibernate as the JPA provider. This choice of Hibernate enhances our ability to leverage JPA benefits while enjoying the specific advantages and capabilities offered by Hibernate as our underlying implementation.

Figure 3.2: Hibernate Framework

3.2.3. JWT Authentication

JSON Web Token (JWT) authentication was chosen as the preferred method for securing our web application due to its numerous advantages and compatibility with our development stack.

- **Stateless Authentication:** JWT is a stateless authentication mechanism, meaning that the server does not need to store any session information. This aligns well with the RESTful architecture of our application, allowing for scalability and easy maintenance.
- **Token-Based Approach:** JWT uses a token-based approach where a compact, URL-safe string is generated, containing information about the user's identity and additional

claims. This token is then sent with each request, eliminating the need for continuous authentication checks against a database.

- **Cross-Origin Resource Sharing (CORS) Compatibility:** JWT is well-suited for applications with cross-origin requests, providing a secure way to include user authentication information in HTTP requests across different domains.
- **Decentralized Authorization:** JWT allows for decentralized authorization by embedding user roles and permissions directly into the token. This decentralization enhances scalability and reduces the need for frequent database queries during authorization checks.
- **Interoperability:** JWT is a standardized, open-source format (RFC 7519), ensuring interoperability across different platforms and technologies. This makes it a suitable choice for our diverse tech stack involving Java, Spring, and React.
- **Expiration and Refresh Mechanism:** JWT supports token expiration, enhancing security by limiting the lifespan of tokens. Additionally, a refresh mechanism can be implemented to obtain new tokens without requiring the user to re-enter credentials, providing a seamless user experience.

Figure 3.3: JSON Web Token

3.2.4. React

For the frontend development, React was chosen for the following reasons:

- **Component-Based Architecture:** React's component-based architecture promotes reusability and modularity, making it easier to manage complex user interfaces.
- **Virtual DOM:** React's use of a virtual DOM enhances performance by minimizing the need for direct manipulation of the actual DOM, resulting in faster updates and improved user experience.
- **Declarative Syntax:** The declarative syntax of React simplifies the process of building interactive user interfaces, making the code more readable and easier to understand.
- **Community and Ecosystem:** React has a vibrant and extensive community, along

with a vast ecosystem of libraries and tools, which facilitates rapid development and problem-solving.

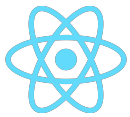

Figure 3.4: React

3.2.5. Further Notes

- Collectively, these choices have steered the development process in accordance with the Spring Web MVC architecture. This approach facilitated seamless implementation of Controller classes, enabling efficient access to the database through JPA Repositories.
- Additionally, we utilized Maven as a project manager to streamline the build and dependency management processes, ensuring a well-organized and maintainable project structure.
- Finally, we integrated Vite.js and NPM into our development workflow, optimizing the frontend development process through rapid prototyping capabilities, efficient dependency management and a streamlined build process.

4 | Source code structure

The structure of the application's source code is depicted, by only highlighting the most important components and pertinent files for description.

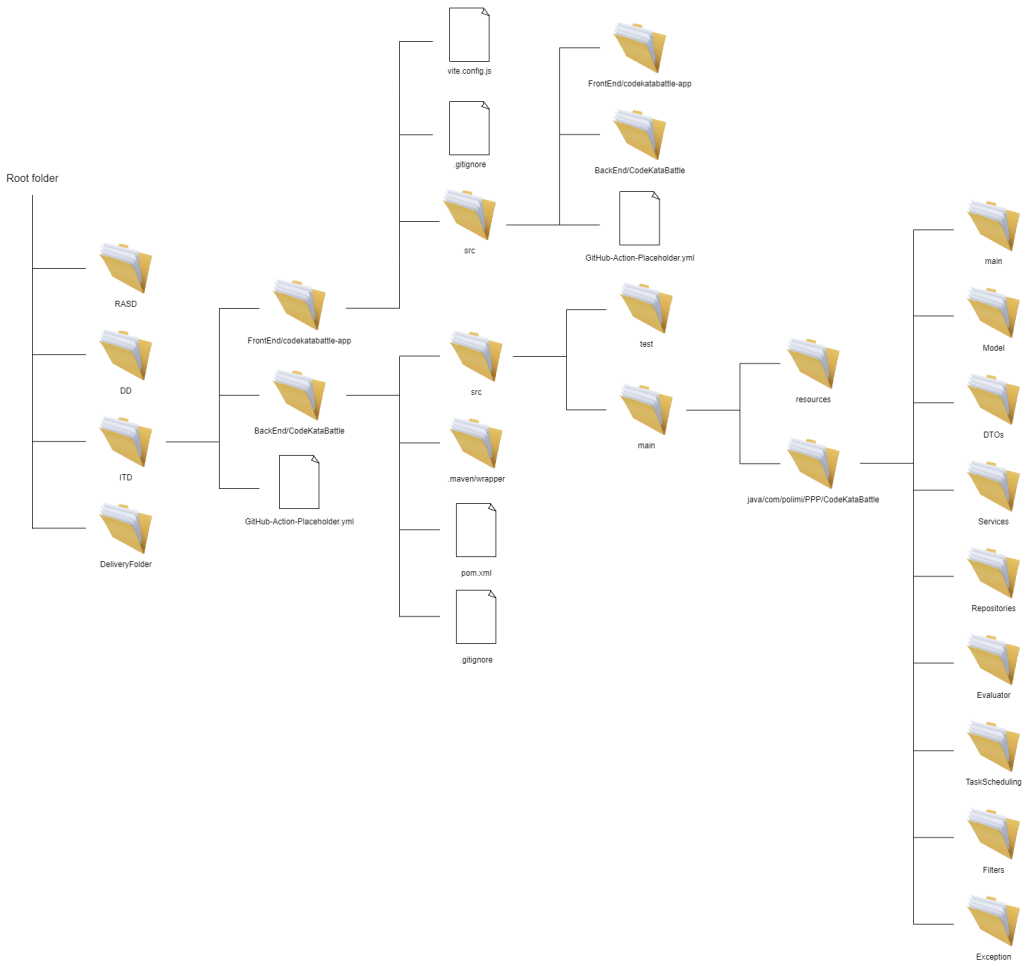

Figure 4.1: Structure of the source code

5 | Testing

During the testing phase, we ensured the reliability and functionality of our system.

First, we conducted unit testing on services to deeply examine the business logic. To achieve this, we employed Mockito to mock dependencies for unit tests of components that required dependency injection, focusing on validating the correctness of individual classes, without assessing the correct integration of the components. This step primarily targeted Services.

Figure 5.1: Mockito

In the subsequent phase, we conducted comprehensive integration testing using JUnit and MockMVC to simulate requests. To emulate real-world scenarios, we utilized H2, a database management system that is compatible with JPA and operates in memory. This strategy guaranteed that our testing environment closely resembled the production setup, eliminating the need for a separate testing database. Notably, we mocked GitHub APIs and the Email Provider to reduce email spam and avoid exceeding GitHub's API request limits.

The testing process included a total of 39 tests, covering various aspects of the system. Furthermore, we thoroughly tested each endpoint using Postman to ensure the overall robustness and reliability of our application.

5.1. Relevant Test Cases

As mentioned, the entire backend of our application was subjected to exhaustive testing.

5.2. Outcomes

The outcome of our testing process are illustrated below.

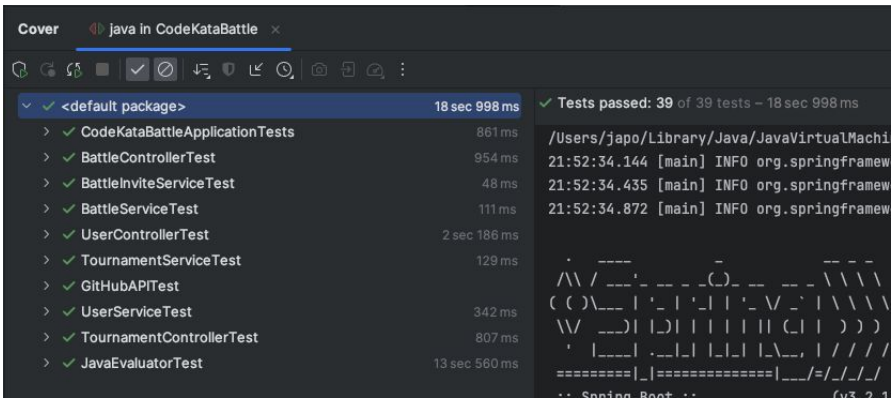

Figure 5.2: Test outcome

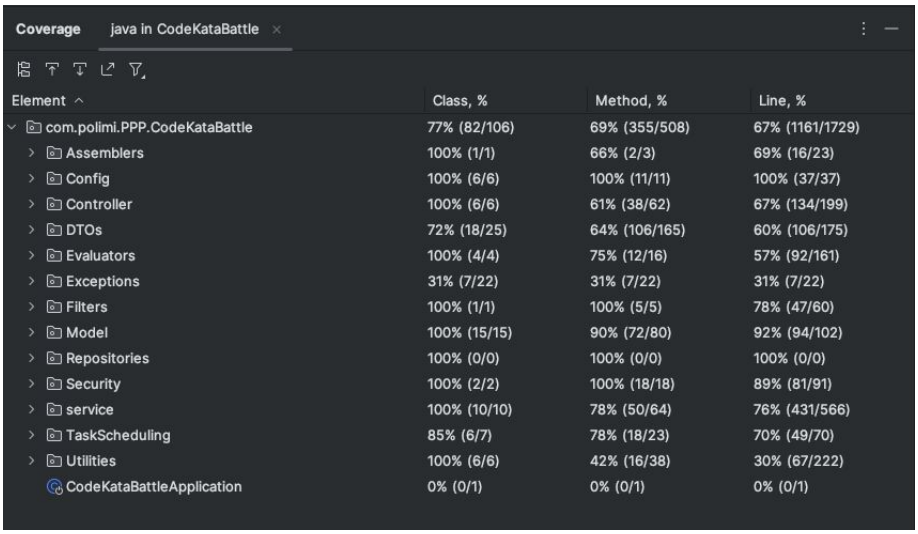

Figure 5.3: Test coverage

6 | Installation guide

Both the frontend and the backend are already hosted on DigitalOcean. There is no need of installing anything locally.

The applications are hosted at the following link:

- Frontend: https://codekatabattle.it
- CKB: https://codekatabattle.it:8443

7 | Effort spent

This section shows the amount of time that each member has spent to produce the document. Please notice that each unit is the result of coordinated work among all the members.

UNIT	**MEMBERS**	**HOURS**
BackEnd	Piazzalunga, Piccinato, Puglisi	120h
FrontEnd	Puglisi, Piazzalunga, Piccinato	80h
Testing	Piazzalunga, Puglisi	20h
Report	Piccinato	10h

8 | References

8.1. References and Tools

1. GitHub: https://www.github.com
2. GitHub Actions: https://github.com/features/actions
3. Diagrams have been made with: http://draw.io
4. React: https://en.wikipedia.org/wiki/React_(software)
5. Spring: https://en.wikipedia.org/wiki/Spring_Framework
6. JPA: https://en.wikipedia.org/wiki/Jakarta_Persistence
7. Hibernate: https://en.wikipedia.org/wiki/Hibernate_(framework)
8. JWT: https://en.wikipedia.org/wiki/JSON_Web_Token
9. DigitalOcean: https://www.digitalocean.com/github-students