AY 2023/2024

POLITECNICO
MILANO 1863

DD: Design Document

Tommaso Capacci Gabriele Ginestroni

Professor
Elisabetta Di Nitto

Version 1
December 8, 2023

Contents

1 Introduction

1.1 Scope

Traditional software programming education often lacks of hands-on experience and continuous evaluation. CodeKataBattle addresses these issues by providing a platform for competitive programming challenges which promotes teamwork and emphasizes the test-first approach in software development. It allows students to apply theoretical knowledge in tournaments with an automated scoring system. Instructors benefit from closer mentorship through code reviews and manual evaluation, creating a cycle of learning through practice, feedback, and community engagement.

HERE

2 Architectural Design

2.1 Overview

2.2 Component view

The following component diagram highlights the main components of the system and their interaction with external entities and services. In the diagram, the components have been organized to highlight the logical grouping of the system elements.
The WebApplication component represents the presentation layer of the system, being the only entry point for the users. The application and integration logic are represented together due to their tight interaction, while the data layer contains the databases accessed by the respective microservices. Different colors are used to highlight components that share similar roles in the system.
Orange components represents the system's microservices. Some complex microservices have been further decomposed into subcomponents, for a more fine grained representation.
Yellow components represents the model of the database accessed by its microservice. The model offers to the microservice an abstraction of the database, allowing it to access the data without knowing the underlying database implementation technology.
The violet has been used to highlight components that cover an important role in the integration between some of the main microservices of the system. Specifically, it has been used for the queues subcomponents, which are used to implement the asynchronous and concurrent communication between specific microservices. Some microservices have an important role in the integration with external entities as well, but have been depicted with their orange color used for microservices. This aspect will be clarified in the detailed description of the components that follows the diagram.
Red components represent the external services that interact with the system.
Finally, the green color has been used to highlight the databases components that are used to store the data of the system.

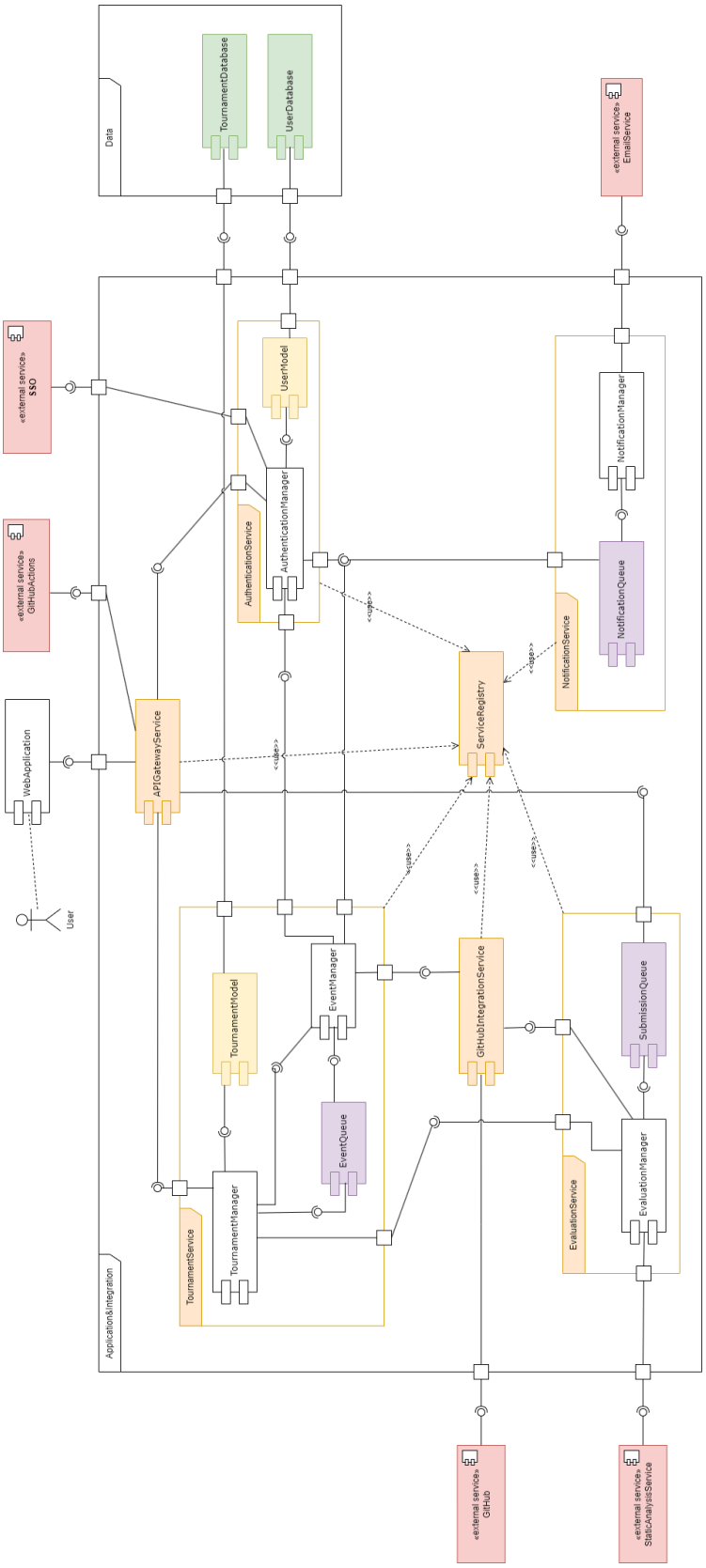

Figure 1: Component diagram

The components in Figure 1 are :

- **WebApplication**: the web app to which users of the CKB platform (students and educators) connect through a modern web browser. It is the front end of the system, and thanks to the interface offered by the **APIGatewayService**, allows users to manage and access the most important aspects of tournaments and battles

- **APIGatewayService**: this component is the microservice that exposes the REST API used by the WebApplication. Indeed, it allows the implementation of the main functionalities needed by the users of the web application. It also offers a REST API, used by the GitHub Actions Service, to notify a submission of a student on the Github repository. It's responsible for the following functionalities:

  - Request validation and access limit: it is responsible for the validation of the requests, limiting the access to the system's APIs.
  - Orchestration: it is responsible for the orchestration of the interaction between different components of the system on behalf of the different clients. It also handles the authentication and authorization of the users, thanks to the interaction with the **AuthenticationService**.

- **AuthenticationService**: this component is the microservice that handles the authentication and authorization of the users of the system. It is responsible also for all the data related to the users of the system, such as their personal information and their roles. It includes the following subcomponents:

  - **AuthenticationManager**: implements the main logical functionalities of the AuthenticationService, exposing APIs used by other microservices to authenticate users and to retrieve their information.
  - **UserModel**: represents the model of the database used by the AuthenticationService to store the data related to the users of the system.

- **TournamentService**: this component is the microservice that implements most of the functionalities needed by the users. It handles:

- Management of tournaments and battles: it allows the creation of battle and tournaments, enrollment of students to tournaments and battles.
- Management of events regarding tournaments and battles.
- Management of the ranking of the students.
- Management of data related to tournaments, battles and submissions

It is composed of the following subcomponents:

- **TournamentManager**: implements the main logical functionalities of the TournamentService, exposing APIs used by other microservices to manage tournaments and battles and their data.
- **TournamentModel**: represents the model of the database used by the TournamentService to store the data related to tournaments and battles.
- **EventQueue**: implements the queue used by the TournamentService to manage the events related to tournaments and battles.
- **EventManager**: periodically checks the EventQueue for events and processes them once the deadlines are reached.

2.3 Deployment view

2.4 Component interfaces

2.5 Runtime view

2.6 Selected architectural styles and patterns

2.7 Other design decisions

3 Effort Spent

Name and Surname	Section 1	Section 2	Section 3	Section 4
Tommaso Capacci	10	10	10	10
Gabriele Ginestroni	10	10	10	10