POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Software Engineering 2
Design Document

Author(s): **Ballabio Giacomo - 10769576**
Benelle Francesco - 10727489
Cavallotti Alberto - 10721275

Academic Year: 2023-2024
Version: 1.0
Release date: 07/01/2024

Contents

1 | Introduction

1.1. Purpose

1.2. Scope

1.3. Definition, Acronyms, Abbreviations

Acronyms	**Definition**
DD	Design Document
ST	Student
ED	Educator
STG	Student Group
CKB	CodeKataBattle
GH	GitHub
User	All STs and EDs
API	Application Programming Interface
RX	Requirement X
CMP	Component

Table 1.1: Acronyms used in the document.

1.4. Revision History

1.5. Reference Documents

1.6. Document Structure

The document is structured in seven sections, as described below.

In the first section, the chapter elucidates the significance of the Design Document, providing comprehensive definitions and explanations of acronyms and abbreviations. Additionally, it recalled the scope of the CodeKateBattle system.

The second section shows the main components of the system and their relationships. This section also focuses on design choices and architectural styles, patterns and paradigms.

The next section, the third, describes the user interface of the system, providing mockups and explanations of the main pages.

The fourth section describes the requirements of the system, showing how they are satisfied by the design choices.

This fifth part provides an overview of the implementation of the various components of the system, it also shows how they are integrated and it gives a plan for testing them all.

In the sixth section are included information about the number of hours each group member has worked for this document.

The last section contains the list of the documents used to redact this Design Document.

2 | Architectural Design

2.1. Overview

Here we represents an overview of how the entire CKB architecture is composed of:

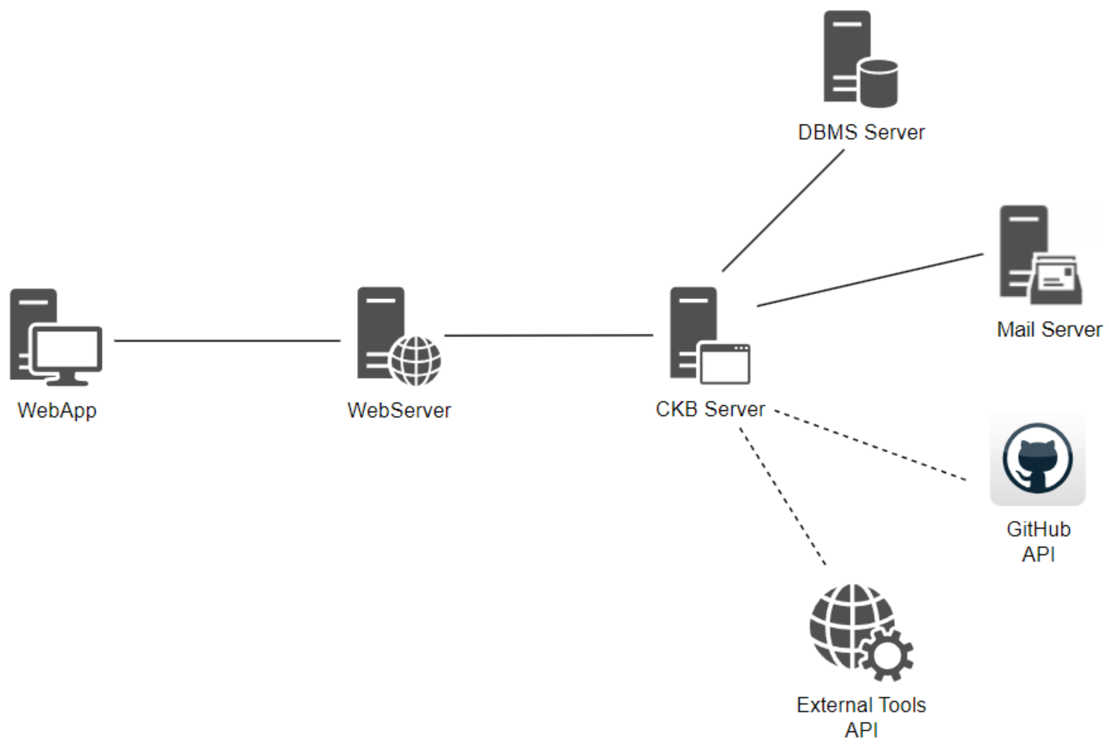

Figure 2.1: CKB Overview.

Client side:

- WebApp: serves as the User interface, allowing all Users to connect to CKB. It enables Users to perform operations such as registration, login, creating or joining Tournaments and Battles, creating or modifying Badges and searching for other Users.

Server side:

- **Web Server:** handles communication with Users, receiving and processing their inputs. Additionally, it provides load balancing for requests, distributing them among various replicas of the CKB Server.
- **CKB Server:** is the central component where interfaces are located, facilitating communication between the Web Server and databases/APIs. It serves as the primary server for the entire website and is replicated across multiple machines to handle a high volume of requests.
- **DBMS Server:** stores data related to Users, Tournaments, and Battles. It acts as the repository for essential information.
- **Mail Server:** is responsible for sending confirmation eMails when a new User registers on CKB, enhancing the User registration process.
- **GitHub API:** is utilized for communication with GitHub, facilitating the creation of the Battle repository and allowing STGs to fork the repository to push their code.
- **External Tools API:** used to automatically test the STG code when a new push is made. It is also used to retrieve the results of the tests and update the Battle dashboard.

2.2. Component View

2.2.1. High Level Diagram

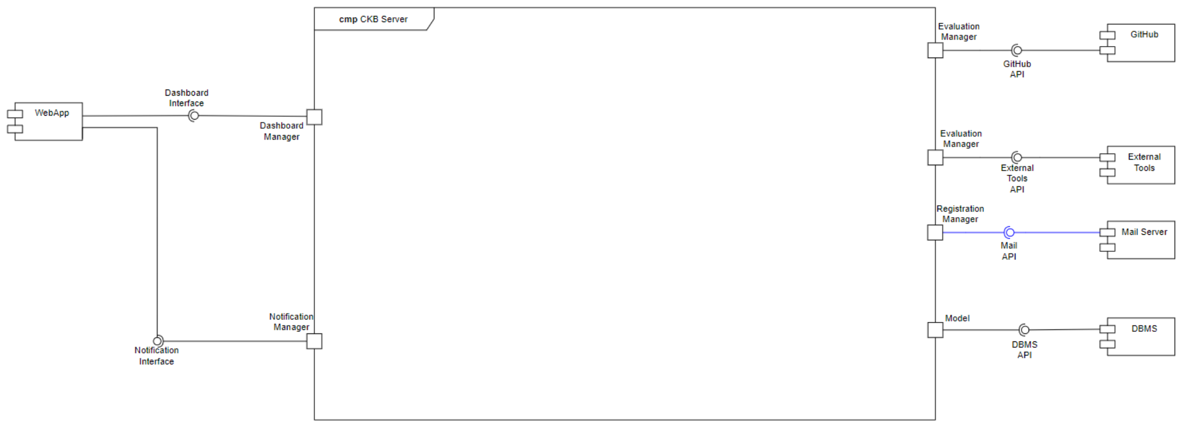

Figure 2.2: High Level Diagram.

In the figure above is the high level component diagram of CKB where it's represented the external components of CKB and how they communicate with the CKB server, in particular:

- **WebApp**: serves as the external access point for Users, allowing communication with the CKB Server through the Dashboard Interface—the sole means for Client-Server interaction from the User side. The CKB Server can relay notifications, such as Tournament or Battle creation, to Users through the Notification Interface.
- **DBMS:** is the storage repository for all User, Tournament, and Battle data. It communicates with the CKB Server via the DBMS API, which is connected to the Model component.
- **Mail Server:** responsible for sending registration confirmation eMails, the Mail Server communicates with the CKB Server using the Mail API interface. This interface is linked to the Registration Manager component, which oversees the User registration process..
- **External Tools:** external application used for testing the code submitted by STGs on GitHub. It communicates with the CKB Server through the External Tools API, connecting to the Evaluation Manager component. The Evaluation Manager handles the evaluation process for STG-submitted code.
- **GitHub:** external website used to create repositories for the code katas of Battles. Each STG, after forking the main repository, pushes their code for evaluation. GitHub communicates with the CKB Server through the GitHub API, linked to the Evaluation Manager.

2.2.2. Low Level Diagram

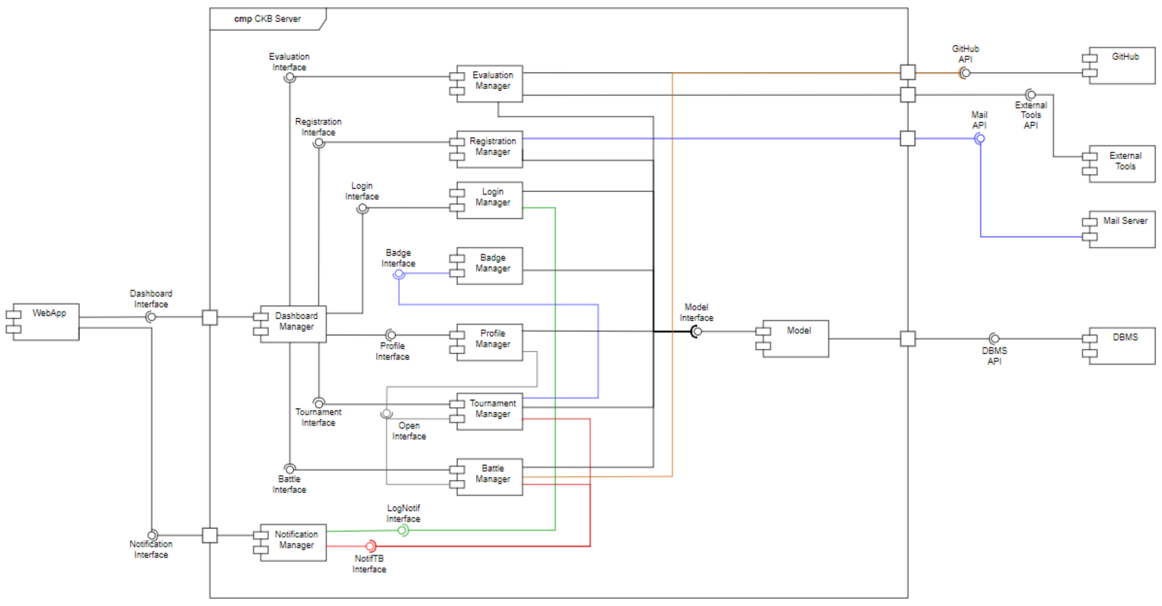

Figure 2.3: Low Level Diagram.

The figure above represents the complete architecture of CKB website with each components inside the CKB Server:

- **Dashboard Manager:** pivotal component that orchestrates all communication between users and the CKB website. Users interact with CKB through the Dashboard Interface, and the Dashboard Manager directs requests to the appropriate components. It serves as the central hub for user interactions.
- **Model:** This high-level component represents the data on the server and acts as an interface to the database server. It acts as a mask to the database server, and every component needs to interface with it to access data from the DBMS through the DBMS API.
- **Evaluation Manager:** component that handles the code evaluation both when a new push is made by a STG on GH or when an ED wants to manually evaluate a STG code during the consolidation stage of a Battle, more detailed information in the following section in Figure 2.4. This component communicates with the Dashboard manager through the Evaluation Interface, with the Model component through the Model Interface to add and modify the STG evaluation in the DBMS, with GitHub through the GitHub API when a new push is made by a STG and

with the External Tools through the External Tools API to automatically evaluate a STG code.

- **Registration Manager:** component that handles the registration of a new User. When a new User wants to create an account on CKB system it communicates with the Dashboard Manager that forwards the request to the Registration Manager through the registration interface. Then the Registration Manager handles the request and communicates through the Mail API to the Mail Server, to send a confirmation mail to the new User, and through the Model Interface to the Model component to add the new User's information to the DBMS. This component also gives the permission to the User that registered as an Educator to create Tournaments, Battles and Badges.

- **Login Manager:** component that handles the login process for registered Users. When a User attempts to log in, the Dashboard Manager forwards the request to the Login Manager through the Login Interface. The Login Manager communicates with the Model component through the Model interface to retrieve the User's data from the DBMS.

- **Badge Manager:** component that handles the Badges creation and modification when a new Tournament is created. When the ED creates a new Tournament, the Badge Manager component receives a request from the Tournament Manager through the badge interface and lets the ED create new Badges or modify existing ones, more details in Figure 2.5. The new Badges are added in the DBMS through the communication between the Badge Manager and the Model component through the Model Interface.

- **Profile Manager:** component that allows User profile search and profile open in both Tournament and Battle dashboard. When a User initiates a search, the Dashboard Manager forwards the request to the Profile Manager using the Profile Interface. The Profile Manager communicates with the Model component through the Model Interface to retrieve relevant information from the DBMS.This component also manages the open profile operation within a Tournament or Battle dashboard, more detail inFigure 2.6.

- **Tournament Manager:** component that manages all user actions related to Tournaments; more detailed information in Figure 2.7. It communicates with the Dashboard Manager through the Tournament Interface, with the Notification Manager through the NotifTB Interface when a new Tournament is created and with the Model component through the Model Interface to add or retrieve data from the

DBMS.

- **Battle Manager:** component that manages all user actions related to Battles; more detailed information in Figure 2.8. It communicates with the Dashboard Manager through the battle interface, with the Notification Manager through the NotifTB Interface when a new Battle is created or when a ST invites other STs to join his STG and with the Model component through the Model Interface to add or retrieve data from the DBMS. It also communicates with GitHub through the GitHub API to create repositories and upload code katas for new Battles.
- **Notification Manager:** component that handles each notification that has to be sent to the Users in particular when a new Tournament is created it sends a notification to all STs registered in CKB, when a new Battle is created it sends a notification to all the STs that have joined that specific Tournament, when a ST invites another ST to join his STG for a battle a notification is sent to the second ST and when a Tournaments is closed and the score are updated it sends a notification to all the STs that have joined the Tournament. All the communication from the Battle Manager and the Tournament Manager with the Notification manager is made through the NotifTB interface and the communication with the WebApp is made through the Notification Interface.

2.2.3. Evaluation Manager

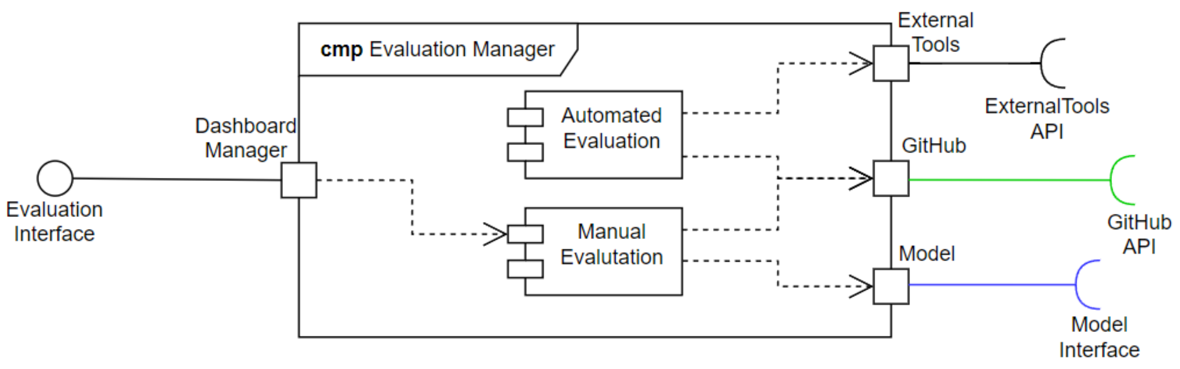

Figure 2.4: Evaluation Manager.

The Evaluation Manager is composed by two other sub-components that handles the two different evaluation method of a STG code:

- **Automated Evaluation component:** utilized by the CKB system to automatically evaluate STG code whenever a new push is made to the STG's forked repository on GitHub. When a code update occurs, GitHub communicates with the Automated Evaluation component through the GitHub API. Subsequently, the Automated Evaluation component sends the code to External Tools via the External Tools API, where the code undergoes testing and evaluation. Upon receiving the evaluation results, the Automated Evaluation component, through the Model interface, communicates with the Model component. The Model component utilizes the DBMS API to update the new score in the relevant DBMS section associated with the corresponding Battle.

- **Manual Evaluation component:** comes into play when an ED wishes to manually assess an STG code. The process begins with the WebApp, which, through the Dashboard Interface, requests the STG code from the Dashboard Manager. The Dashboard Manager then communicates this request to the Manual Evaluation component through the Evaluation Interface. Subsequently, the Manual Evaluation component communicates with the GitHub API to retrieve the source code from the STG's forked repository, allowing the ED to analyze it. Once the evaluation is complete and the ED decides to update the score, the Manual Evaluation component, through the Model Interface, communicates with the Model component. The Model component, utilizing the DBMS API, updates the score in the DBMS, ensuring the manual evaluation results are recorded appropriately.

2.2.4. Badge Manager

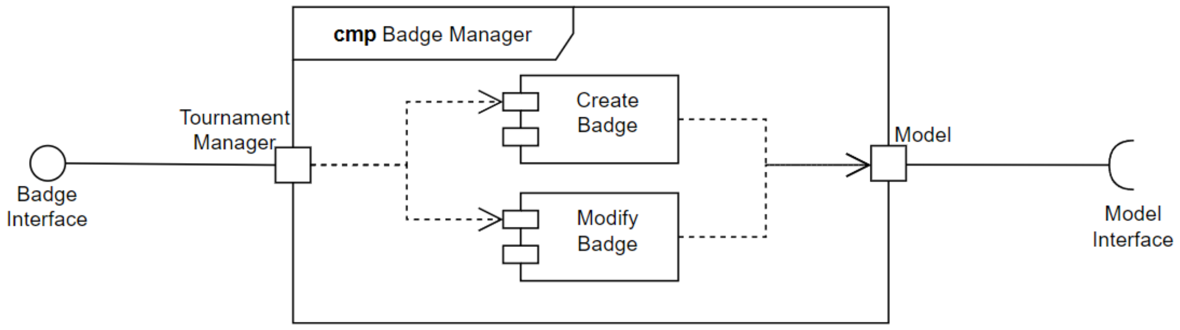

Figure 2.5: Badge Manager.

The Badge Manager is the component used by the CKB system to handle the creation and the modification of the Badges during the Tournament creation:

- **Create Badge component:** activated when an ED intends to create a new Badge for a newly generated Tournament. The initiation of this process is through the Create Tournament component, a sub-component of the Tournament Manager. The Create Tournament component, via the Badge Interface, sends a request to the Create Badge component, allowing the ED to define the Badge with its specific settings and parameters, such as the criteria STs must fulfill to obtain it. Following the Badge creation, the Create Badge component, through the Model Interface, communicates with the Model component, ensuring the newly created Badge is added to the DBMS.
- **Modify Badge component:** engaged when an ED aims to modify an existing Badge that was previously created for another Tournament. The process is initiated by the Create Tournament component, a sub-component of the Tournament Manager. The Create Tournament component, via the Badge Interface, sends a request to the Modify Badge component, allowing the ED to adjust parameters associated with an existing Badge, specifying new criteria for STs to fulfill. Once the Badge is successfully modified, the Modify Badge component, through the Model Interface, communicates with the Model component. This communication ensures that the updated Badge information is reflected in the DBMS.

2.2.5. Profile Manager

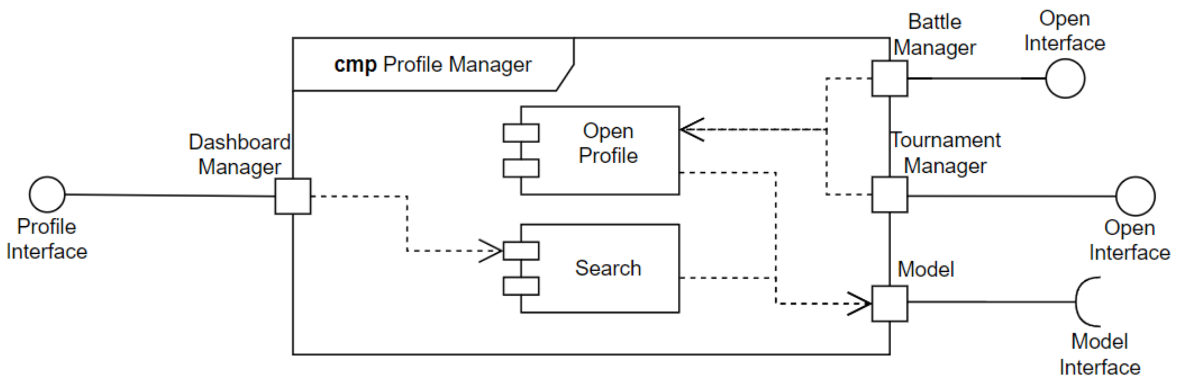

Figure 2.6: Profile Manager.

The Badge Manager is the component used by the CKB system to handle the research of a User's profile and the visualization of another User's profile when the User clicks on a nickname within the Tournament or Battle dashboard:

- **Search component:** responsible for managing the profile search process when a User enters a nickname or keyword into the search bar across various CKB pages. When a User initiates a search by entering another User's nickname or a relevant keyword, the Dashboard Manager communicates with the Search component through the Profile Interface. The Search component then forwards the search request to the Model component via the Model Interface. Subsequently, the Model component retrieves the profile information from the DBMS. The retrieved information is then presented to the User, allowing them to visualize the searched User's profile.
- **Open Profile component:** manages the retrieval of a User's profile when a User clicks on a nickname within a Tournament or Battle dashboard. When a User clicks on another User's nickname in a Tournament or Battle dashboard, the Dashboard Manager communicates with the View component within the Tournament or Battle Manager. The View component forwards the request to the Open Profile component through the Open Interface. The Open Profile component, in turn, communicates with the Model component via the Model Interface. This communication with the Model component facilitates the retrieval of the User's profile information from the DBMS. The retrieved profile information is then presented to the User, allowing them to view the selected User's profile.

2.2.6. Tournament Manager

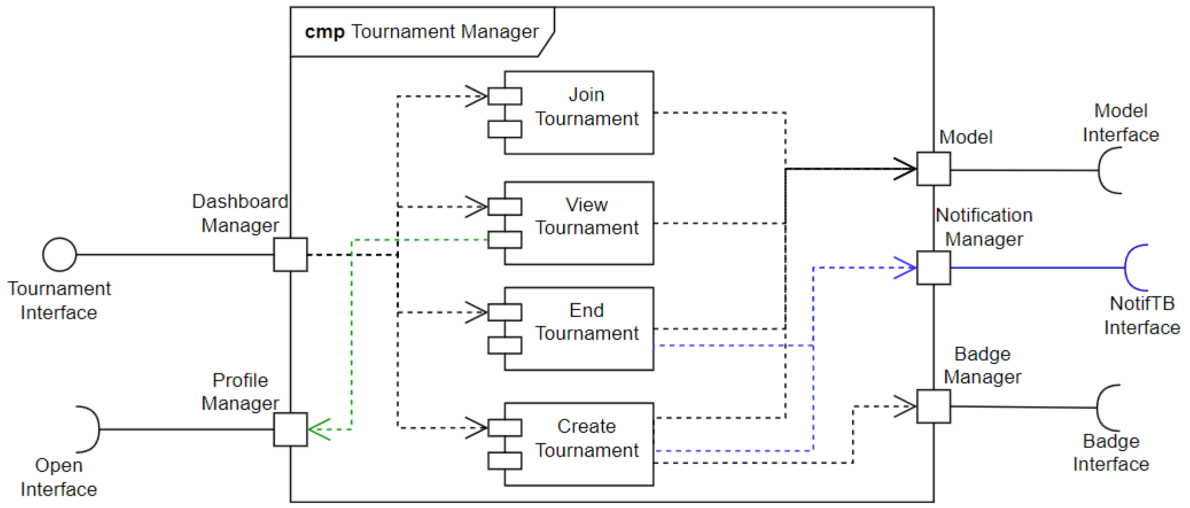

Figure 2.7: Tournament Manager.

The Tournament Manager is the component used by the CKB system to handle every aspect of the Tournament, from the creation to the end going through the Join and the View sub-components:

- **Create Tournament component:** utilized when an ED initiates the creation of a new Tournament. The ED communicates with the Dashboard Manager through the Dashboard Interface, which then directs the request to the Create Tournament component. This component sends back a creation form for the ED to fill. Once completed, the Create Tournament component communicates the data to the Model component through the Model Interface, adding the information to the DBMS via the DBMS API. If the ED grants permissions to other EDs, the Create Tournament component communicates with the Notification Manager through the NotifTB Interface to notify the specified EDs. Additionally, notifications are sent to all registered STs via the Notification Manager, informing them of the new tournament. Throughout the tournament creation process, the Create Tournament component also communicates with the Badge Manager through the Badge Interface, allowing the ED to create or modify Badges associated with the tournament.
- **Join Tournament component:** activated when an ST wishes to join a tournament. The ST communicates with the Dashboard Manager through the Dashboard Interface, and the request is forwarded to the Join Tournament component. This component communicates through the Model Interface with the Model component to add the ST to the participant list in the DBMS through the DBMS API.
- **View Tournament component:** used by the CKB system to let the User visualize the Tournament page with all the information, like the available Battles and the Dashboard with the STs score . When a User wants to search a Tournament it writes the Tournament name or a keyword of it in search bar and it communicates with the Dashboard Manager through the Dashboard Interface that forwards the request to the View Tournament component that communicates with the Model component through the model interface to retrieve all the information from the DBMS through the DBMS API and let the User visualize the Tournament page. the same communication is made when a User clicks on a Tournament name in another User's profile or in his main dashboard page. This component also manages the open profile operation when a User wants to visualize another User's profile from the Tournament dashboard; when a User clicks on another User's nickname in the Tournament dashboard the Dashboard Manager communicates through the Dashboard Interface with the View Tournament component that forwards the request to the the Profile Manager through the Open Interface.

- **End Tournament component:** triggered when an ED decides to close a Tournament, preventing further ST participation and ED creation of Battles within it. The ED communicates with the Dashboard Manager through the Dashboard Interface, which forwards the request to the Close Tournament component. The Close Tournament component communicates with the Model component through the Model Interface, modifying the Tournament's status to make it non-joinable in the DBMS through the DBMS API. Additionally, the Close Tournament component communicates through the NotifTB Interface to the Notification Manager, which sends notifications to all STs who participated in the Tournament, informing them that final scores are ready for viewing.

2.2.7. Battle Manager

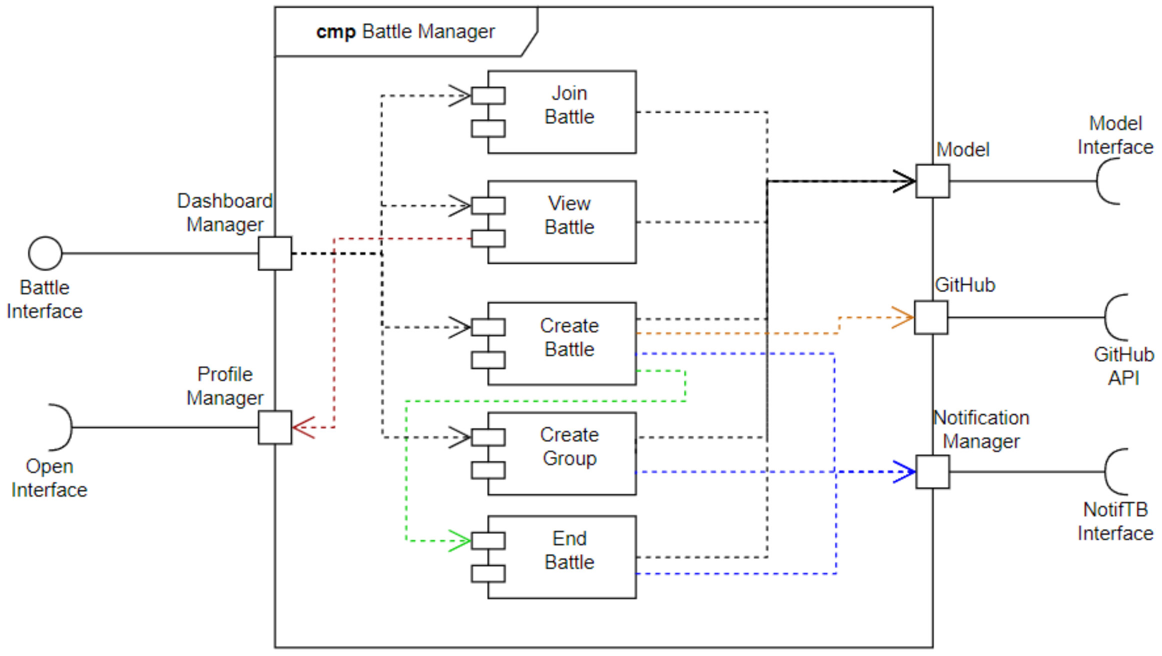

Figure 2.8: Battle Manager.

The Battle Manager is the component used by the CKB system to handle every aspect of the Battle, from the creation to the end going through the Join, the View and the Create Group sub-components:

- **Create Battle component:** engaged when an ED wishes to create a new Battle within a Tournament. The ED communicates with the Dashboard Manager through the Dashboard Interface, forwarding the request to the Create Battle component. The Create Battle component responds by sending a creation form to be filled by the ED. Upon completion, the component communicates the data to the Model component through the Model Interface, adding the information to the DBMS through the DBMS API. Additionally, the Create Battle component communicates with the Notification Manager through the NotifTB Interface, notifying all STs who have joined the relevant Tournament about the new Battle. It also collaborates with the End Battle component to create a timer, ensuring that after the consolidation stage concludes, the End Tournament component notifies all STs about the availability of final grades. The Create Battle component also communicates with GitHub through the GitHub API to create a new repository and upload the code kata for the battle. This repository is later forked by all STGs to submit their code.

- **Join Battle component:** activated when an ST intends to join a Battle. The ST communicates with the Dashboard Manager through the Dashboard Interface, and the request is forwarded to the Join Battle component. The Join Battle component, through the Model Interface, communicates with the Model component, adding the ST to the participant list of the battle in the DBMS through the DBMS API.
- **View Battle component:** used by the CKB system to let the User visualize the Battle page with the dashboard including all the STGs score. When a User wants to visualize the Battle page it communicates with the Dashboard Manager through the Dashboard Interface that forwards the request to the View Battle component that communicates with the Model component through the Model Interface to retrieve all the information from the DBMS through the DBMS API and finally let the User visualize the Battle page. This component also manages the open profile operation when a User wants to visualize another User's profile from the Battle dashboard; when a User clicks on another User's nickname in the Battle dashboard the Dashboard Manager communicates through the Dashboard Interface with the View Tournament component that forwards the request to the the Profile Manager through the Open Interface.
- **Create Group component** engaged when STs want to create a new STG for a Battle. After joining a Battle before the registration deadline expires, an ST can create an STG to participate in the battle. The ST communicates with the Dashboard Manager through the Dashboard Interface, initiating a request to create a new STG. The request is forwarded to the Create Group component through the Battle Interface, allowing the ST to decide the STG name and invite other STs. Notifications are sent through the Notification Manager, which communicates with

the Battle Manager through the NotifTB interface and with the WebApp through the notification interface. When the STG is confirmed, the Create Group component communicates through the Model Interface with the Model component to add the newly created STG to the DBMS through the DBMS API.

- **End Battle component:** activated to notify all STGs that the final scores of the battle are available. When the consolidation stage concludes, and the timer created by the Create Battle component expires, the End Battle component communicates through the Model Interface with the Model component, updating the scores in the DBMS through the DBMS API. It also notifies all participating STs through the Notification Manager, communicating through the NotifTB Interface, that the final scores are accessible on the Battle page. Communication between the Notification Manager and the WebApp is facilitated through the Notification Interface.

2.3. Deployment View

2.4. Runtime View

2.5. Component Interfaces

2.6. Selected Architectural Styles and Patterns

2.7. Other Design Decisions

3 User Interface Design

4 Requirements Traceability

5 Implementation, Integration and Test Plan

6 | Effort Spent

Member of group	Effort spent	
Ballabio Giacomo	Introduction	$0h$
	Architectural Design	$0h$
	User Interface Design	$0h$
	Requirements Traceability	$0h$
	Implementation Integration Test Plan	$0h$
	Reasoning	$0h$
Benelle Francesco	Introduction	$0h$
	Architectural Design	$0h$
	User Interface Design	$0h$
	Requirements Traceability	$0h$
	Implementation Integration Test Plan	$0h$
	Reasoning	$0h$
Cavallotti Alberto	Introduction	$0h$
	Architectural Design	$0h$
	User Interface Design	$0h$
	Requirements Traceability	$0h$
	Implementation Integration Test Plan	$0h$
	Reasoning	$0h$

Table 6.1: Effort spent by each member of the group.

7 | References

7.1. References

- The Requirement Engineering and Design Project specification document A.Y. 2023–2024.

7.2. Used Tools

- GitHub for project versioning and sharing.
- LaTeX and *Visual Studio Code* as editor for writing this document.
- *sequencediagram.org* for the sequence diagrams' design.
- *draw.io* for the other diagrams' design.
- *Google Documents* for collaborative writing, notes and reasoning.

List of Figures

List of Tables