



ARGENTINO SIMONE

M63001555

si.argentino@studenti.unina.it

BORRELLI FRANCESCO

M63001559

francesco.borrelli6@studenti.unina.it

MAURIELLO AUGUSTO

M63001545

au.mauriello@studenti.unina.it

ROMANO VINCENZO

M63001557

vincenzo.romano31@studenti.unina.it

PROGETTO ENACTEST, CORSO DI SYSTEM ARCHITECTURE DESIGN

Sommario

1. Introduzione	3
1.1. Scenario.....	3
1.2. Requisito Specifico Assegnato	3
1.3. Metodologie di Lavoro	4
1.4. Unified Process	4
2. Specifica dei Requisiti.....	6
2.1. Attori, Funzionalità e Attributi	6
2.2. Glossario dei Termini	7
3. User Stories.....	9
3.1. Individuazione delle Storie Utente	9
4. Analisi dei Requisiti	10
4.1. Descrizione Informale dei Requisiti	10
4.2. Requisiti Funzionali	11
4.3. Requisiti sui Dati.....	12
4.4. Requisiti non Funzionali	12
5. Scelte di Progetto	14
5.1. Pattern Architetture Utilizzati	14
5.2. Stili Architetture Utilizzati	15
6. Tecniche di Sviluppo.....	17
7. Modello dei Casi d'Uso.....	22
7.1. Componenti Principali.....	22
7.2. Diagramma dei Casi d'Uso	23
7.3. Scenari Casi d'Uso	24
8. System Domain Model	29
9. Modellazione dei Dati	31
10. Context Diagram.....	32
11. Component Diagram	33
12. Package Diagram	34
13. API.....	36
PROGETTO	1
ENACTEST	1

13.1.	Introduzione e Proprietà.....	36
13.2.	Descrizione formale delle REST API	37
13.3.	TEST REST API in AdminClassController	38
13.4.	TEST REST API in UserClassController	41
14.	Sequence Diagrams	42
14.1.	VisualizzaClassi e DownloadCodiceClasse.....	43
14.2.	RicercaAvanzata	44
14.3.	GestioneClasse	45
14.4.	AggiuntaClasse	46
14.5.	Login	47
14.6.	Avvio Partita	48
15.	Stima dei Costi	49
15.1.	Unadjusted Use Case Weight (UUCW).....	50
15.2.	Unadjusted Actor Weight (UAW).....	51
15.3.	Technical Complexity Factor (TCF).....	51
15.4.	Environmental Complexity Factor (ECF)	52
15.5.	Total Use Case Points (UCP).....	52
16.	Deployment	54
16.1.	Install View	55
16.2.	Diagramma di Deployment	56
17.	Testing.....	57

1. INTRODUZIONE

1.1. SCENARIO

Il Progetto ENACTEST (European Innovation Alliance for Testing Education) nasce dalla volontà di alcuni partner europei in Italia, Spagna, Portogallo, Belgio e Svezia di valorizzare l'attività di collaudo del software (Software Testing). Partendo dall'assunzione che quest'ultima sia di vitale importanza ma spesso noiosa e poco stimolante, sono stati proposti alcuni moduli educativi uniti alla gamification (ludicizzazione) per avvicinare gli studenti al testing, unendo istruzione e divertimento. È stato quindi proposto come capsula un educational game circa "Man vs automated Testing Tools challenges" e diretta a studenti di Ingegneria triennali e magistrali con prerequisito l'avere conoscenze base di tecniche di design e di implementazione di test tramite il framework JUnit. L'obiettivo di tale lavoro di gruppo è far sì che studenti (o team) possano competere contro tool, come Randoop ed EvoSuite, in grado di generare automaticamente casi di test in JUnit. I relativi obiettivi di apprendimento sono correlati all'architettura di un software, il relativo testing adoperando frameworks e l'analisi dei punti di forza e di debolezza dei maggiori strumenti di generazione automatica di codici di test.

1.2. REQUISITO SPECIFICO ASSEGNATO

Il requisito specifico assegnato al gruppo, composto dagli studenti **Argentino Simone, Borrelli Francesco, Mauriello Augusto e Romano Vincenzo**, frequentanti il primo anno magistrale di Ingegneria Informatica presso l'Università di Napoli Federico II, è relativo all'unione di due diversi task:

- **[T1]** L'applicazione deve **mantenere un insieme di Classi Java da testare** e deve offrire la possibilità ai giocatori di **consultare l'elenco delle classi disponibili** e di **fare il download del codice** di una di esse. L'applicazione deve permettere ad un amministratore anche di **aggiornare l'insieme di classi disponibili** mediante **aggiunta di classi** e relativo **salvataggio del file di codice**. Sarebbe auspicabile anche prevedere funzioni per la **ricerca di classi in base a specifici requisiti**, come ad esempio la complessità della classe, o altri attributi.
- **[T5]** Il giocatore (dopo essersi autenticato) **avvia una nuova partita** del Primo Scenario, l'applicazione gli **mostra un elenco di classi da testare ed un elenco di Robot disponibili**, il giocatore **sceglie la classe ed il Robot** contro cui confrontarsi. A questo punto il sistema **crea la partita con tutte le scelte fatte**, le associa un IDPartita, e la salva. Successivamente l'applicazione **avvia l'ambiente di editing** in cui visualizza la classe da testare e gli offre una finestra in cui può scrivere la classe di test.

1.3. METODOLOGIE DI LAVORO

Un Processo Software è un insieme strutturato di attività necessarie per lo sviluppo di un sistema software (specifica, progettazione, sviluppo, validazione, evoluzione dopo il rilascio, ...). Al fine di poter realizzare il progetto, è stato necessario utilizzare strumenti e pratiche ad hoc per organizzare al meglio il lavoro. In particolare, si è deciso di adottare un processo di sviluppo di tipo Agile (e quindi non plan-based/guidato dai piani) per cui la pianificazione è incrementale e risulta più semplice modificare il processo in modo tale da riflettere e adattarsi alle mutevoli esigenze del cliente. A supporto di questa metodologia, durante il processo di progettazione, sono stati realizzati dei prototipi ad-hoc allo scopo di realizzare al meglio i requisiti e per prendere familiarità con le tecnologie utilizzate.

1.4. UNIFIED PROCESS

UP è un framework di processo di sviluppo software iterativo ed incrementale. Infatti, è una metodologia che prevede lo sviluppo del software come un'attività guidata dalla definizione dei requisiti funzionali, espressi attraverso i casi d'uso. L'analisi dei casi d'uso, di conseguenza, permette di definire le caratteristiche dell'architettura software che li realizza in modo integrato. Esso è basato sull'ampliamento e sul raffinamento di un sistema attraverso diverse iterazioni, con feedback e adattamenti ciclici. Il sistema è sviluppato in maniera incrementale col passare del tempo, iterazione per iterazione, e infatti questo approccio è anche conosciuto come sviluppo di software iterativo ed incrementale. Le iterazioni sono divise su quattro fasi ciascuna delle quali consiste in una o più iterazioni:

❖ Inception

La prima è la fase più breve nel progetto. È utilizzata per preparare la base del progetto, che include: stabilire lo scope del progetto, definire i vincoli, creare la tabella AttoriObiettivi, delineare i requisiti chiave e le possibili soluzioni architetturelle insieme ai compromessi di progettazione. Una durata eccessivamente prolungata della fase di inception potrebbe essere sintomo di una mancata chiarezza, da parte degli stakeholders, della visione e degli obiettivi del progetto. Senza obiettivi e visione chiari, il progetto molto probabilmente è destinato a fallire. In questo scenario è meglio prendere una pausa all'inizio del progetto per raffinare visione e obiettivi. In caso contrario, ciò potrebbe portare a ritardi di organizzazione non benevoli per le fasi successive.

❖ Elaborazione

Durante questa fase, il team deve elencare la maggior parte dei requisiti di sistema (per esempio, nella forma di use case), eseguire una analisi dei rischi identificati e definire un piano di risk management per ridurre o eliminarne l'impatto sulla schedule finale e sul prodotto, stabilire la progettazione e l'architettura (utilizzando class diagram di base, package diagram o deployment diagram), creare un piano (calendario, stime dei costi, ecc.) per la fase successiva (costruzione).

❖ Costruzione

La fase più lunga e più ampia di UP. Durante questa fase, la progettazione del sistema viene finalizzata e perfezionata e il sistema viene costruito utilizzando le basi create durante la fase di

elaborazione. La fase di costruzione è suddivisa in più iterazioni, ognuna delle quali deve portare a un rilascio eseguibile del sistema. L'iterazione finale della fase di costruzione permette di ottenere il sistema completo, che deve essere distribuito durante la fase di transizione.

❖ **Transizione**

Fase finale del progetto che consegna il nuovo sistema agli utenti finali. Chiaramente questo modello di sviluppo non è restrittivo né sulla realizzazione di queste fasi, né sul numero di iterazioni da compiere in ogni fase. UP è quindi aperto all'uso di pratiche agili e prevede l'uso di VCS (Version Control System) per mantenere sempre una versione funzionante del software.

2. SPECIFICA DEI REQUISITI

2.1. ATTORI, FUNZIONALITÀ E ATTRIBUTI

Per prima cosa, risulta fondamentale elencare gli attori, le funzionalità e gli attributi del Sistema sviluppato.

Per attore si intende un ruolo coperto da un insieme di entità che interagiscono col sistema. Il sistema da sviluppare sono richiesti i seguenti attori:

➤ Attori

- Utente
- Admin

Per funzionalità si intendono le operazioni che un sistema è in grado di effettuare. Nel sistema da sviluppare sono richieste le seguenti funzionalità:

➤ Funzionalità

- Mantenere un insieme di classi da testare
- Consultare l'elenco di classi disponibili
- Download delle classi disponibili
- Aggiornare l'insieme di classi
- Aggiunta di classi
- Salvataggio del file di codice
- Ricerca di classi in base a specifici requisiti
- Avviare una partita
- Visualizzare elenco classi da testare e robot disponibili
- Scelta della classe da testare e robot avversario
- Avvio dell'ambiente di Editing

Per attributi si intendono le caratteristiche specifiche che il sistema deve memorizzare. Nel sistema da sviluppare sono richiesti i seguenti attributi:

➤ Attributi

- Complessità della classe
- Altri attributi della classe da testare (Come da Traccia)

2.2. GLOSSARIO DEI TERMINI

Segue il glossario dei termini, definito al fine di offrire definizioni e spiegazioni chiare dei termini comuni utilizzati in questa relazione.

Termine	Descrizione	Sinonimi
Classe Java	Descrizione degli attributi di un file <code>.java</code> che si vuole caricare nel database, contenente la classe da testare	Specifiche file
File Java	File con estensione <code>.java</code> contenente il codice Java di una classe da testare	Codice della classe
Last Update	Data dell'ultima modifica della classe Java	Data ultimo Aggiornamento
LOC	Numero di linee di codice del file Java	Linee di Codice
Complexity	Complessità del file Java. Le possibilità previste sono: <ul style="list-style-type: none"> - Simple - Moderate - Complex - Insane 	Complessità della classe
Recommended Opponent	Avversario consigliato. Le possibilità previste sono: <ul style="list-style-type: none"> - RANDOOP - EVOSUITE 	Avversario consigliato
Download	Trasferimento di un file in locale a partire dalla rete	Scaricamento
Test	Parte del ciclo di vita di un software che mira ad individuare la correttezza, la completezza e l'affidabilità dello stesso	Software Testing, collaudo del software
Admin	Utente che può aggiornare l'insieme delle classi disponibili mediante l'aggiunta di classi e relativo salvataggio del file di codice	Gestore, Amministratore
User	Utente che può visualizzare l'elenco delle classi e farne il relativo download	Competitore, Utente registrato, Giocatore

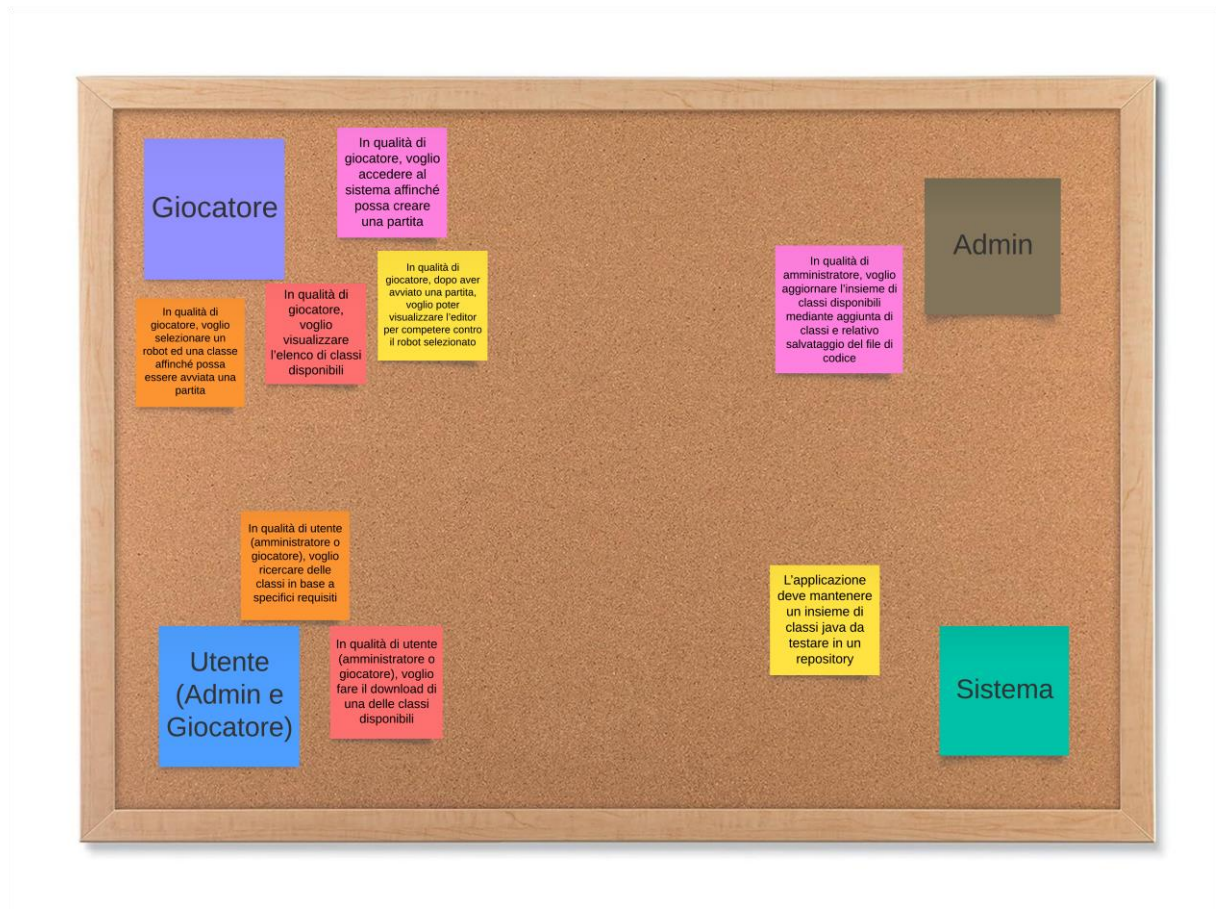
Termine	Descrizione	Sinonimi
Repository	Archivio digitale di informazioni centralizzato	Archivio, database
Partita	Sfida avviata dall'utente, che può portare a vittoria, pareggio o sconfitta	Sfida, Competizione

3. USER STORIES

3.1. INDIVIDUAZIONE DELLE STORIE UTENTE

La storia utente è un concetto chiave nell'Ingegneria del Software, in particolare nel contesto dello Sviluppo Agile. Una storia utente rappresenta un requisito funzionale o un'unità di lavoro che descrive una determinata funzionalità o caratteristica dal punto di vista dell'utente finale del software. In altre parole, definisce cosa deve essere fatto dal sistema per soddisfare le esigenze degli utenti.

Solitamente, una storia utente è composta da una breve descrizione del bisogno dell'utente, i criteri di accettazione per valutare la sua realizzazione e un ordine di priorità. Le storie utente sono spesso organizzate in un elenco prioritario (Product Backlog) e vengono selezionate per l'implementazione durante le iterazioni o gli Sprint di sviluppo. Questo approccio permette di focalizzarsi sugli obiettivi più rilevanti e di consegnare valore in modo incrementale agli utenti durante tutto il processo di sviluppo del software. In figura vengono mostrate le 8 storie utente alla base del task:



4. ANALISI DEI REQUISITI

In questo capitolo vengono descritti i requisiti che il programma deve rispettare. Questi vengono classificati in tre tipi:

- **Requisiti Funzionali**
Rappresentano funzionalità e servizi che il sistema deve fornire e descrivono il comportamento del sistema a fronte di determinati input
- **Requisiti sui Dati**
Caratteristiche che i dati devono rispettare ai fini del corretto funzionamento del programma
- **Requisiti non Funzionali**
Rappresentano vincoli e caratteristiche relative al sistema, al processo di sviluppo e agli standard che essi devono rispettare

4.1. DESCRIZIONE INFORMALE DEI REQUISITI

Si vuole realizzare un'applicazione ove sia possibile effettuare le seguenti operazioni:

- ❖ Mantenere un insieme di classi facilmente consultabili e scaricabili in locale dagli utenti. Tali file hanno alcuni parametri di fondamentale importanza:
 - **ID**
Identificativo utile alla persistenza sulla base dati, individua univocamente un file
 - **Path**
Percorso adoperato per raggiungere un determinato file
 - **Nome del file**
 - **Complexity**
Difficoltà nominale dello studente nell'implementazione dei casi di test relativi alla classe
 - **LOC**
Linee di codice della classe
 - **Last update**
Data di ultima modifica della classe
 - **Recommended opponent**
Tool che generano automaticamente test case in JUnit, con il quale lo studente compete. Quello consigliato si addice alla difficoltà.
- ❖ Gestire le classi da parte di un amministratore al fine di modificare, inserire o rimuovere classi Java. Tale modifica impatterà la visione delle classi da parte dell'utente

- ❖ Realizzare funzionalità avanzate di ricerca che sfruttino particolari filtri e aiutino uno studente a cercare la giusta classe con cui confrontarsi, così come l'amministratore a trovare immediatamente il file eventualmente da modificare
- ❖ Memorizzare le partite avviate dagli utenti caratterizzate dai seguenti parametri:
 - **ID**
Identificativo utile alla persistenza sulla base dati, individua univocamente un file
 - **Classe di test**
Classe che l'utente ha scelto di testare
 - **Opponent**
Tool che generano automaticamente test case in JUnit, con il quale lo studente compete

4.2. REQUISITI FUNZIONALI

Si elencano in seguito i requisiti funzionali.

- ❖ **RF-1.** Il sistema deve consentire il mantenimento di tutte le classi di testing caricate dall'amministratore.
- ❖ **RF-2.** Il sistema deve offrire all'amministratore la possibilità di effettuare l'upload di un file .java contenente la classe da testare.
- ❖ **RF-3.** Il sistema deve offrire all'amministratore la possibilità di specificare le caratteristiche di ciascuna classe di testing caricata.
- ❖ **RF-4.** Il sistema deve offrire all'amministratore la possibilità di modificare gli attributi della classe di testing caricata.
- ❖ **RF-5.** Il sistema deve consentire all'amministratore di eliminare una classe precedentemente caricata, ed il relativo file .java.
- ❖ **RF-6.** Il sistema deve consentire all'utente (giocatore e amministratore) la possibilità di visualizzare tutte le classi di testing collezionate.
- ❖ **RF-7.** Il sistema deve consentire all'utente (giocatore e amministratore) di effettuare il download di un file precedentemente caricato.
- ❖ **RF-8.** Il sistema deve consentire all'utente (giocatore e amministratore) di ricercare un sottoinsieme di classi in base a degli specifici filtri.
- ❖ **RF-9.** Il sistema deve permettere ad un giocatore di poter avviare una partita.
- ❖ **RF-10.** Il sistema deve permettere al giocatore di poter scegliere la classe da testare.
- ❖ **RF-11.** Il sistema deve permettere al giocatore di poter scegliere il robot con il quale competere.

- ❖ **RF-12.** Il sistema deve permettere al giocatore di poter salvare la partita, la quale verrà identificata mediante l'ausilio di un Id.
- ❖ **RF-13.** Il sistema deve permettere al giocatore di poter avviare l'editor una volta avviata la partita.

4.3. REQUISITI SUI DATI

Si elencano in seguito i requisiti sui dati.

- ❖ **RD-1.** Ad ogni file .java devono essere associate le seguenti informazioni:
 - Nome del file
 - Contenuto
 - Tipo di file
- ❖ **RD-2.** Per ogni classe di testing si deve specificare:
 - Complessità
 - Avversario consigliato
 - Data dell'ultima modifica degli attributi
 - Numero di linee di codice
- ❖ **RD-3.** Ad ogni classe di testing deve essere associato il corrispondente file .java.
- ❖ **RD-4.** Per ogni partita avviata dall'utente si devono specificare:
 - Classe di test
 - Avversario

4.4. REQUISITI NON FUNZIONALI

Si elencano in seguito i requisiti non funzionali.

- ❖ **RNF-1: Evolvibilità**
 - Essendo il software implementato parte di un progetto di dimensioni più ampie, deve essere pensato per poter essere esteso, integrando ulteriori componenti.
- ❖ **RNF-2: Usabilità**
 - Il sistema deve risultare intuitivo e semplice da usare, comprensibile sia al giocatore che all'amministratore.
 - Le operazioni consentite agli attori con cui interagisce il sistema devono essere immediate, per cui è opportuno che sia implementata un'interfaccia utente chiara e coerente.

❖ **RNF-3: Installabilità**

- L'applicazione deve risultare semplice da installare.
- Il processo di installazione deve essere opportunamente guidato dalla documentazione

❖ **RNF-4: Testabilità**

- Il testing dell'applicazione deve essere semplice, in modo da favorire il lavoro di altri task che sono intenzionati ad integrare il presente sistema dopo averlo testato.

❖ **RNF-5: Integrità**

- Il sistema deve garantire l'integrità dei dati contenuti nel database.
- Il sistema deve verificare l'integrità degli input prima di effettuare l'inserimento di questi nel database.
- Il sistema deve garantire la consistenza dei dati e che gli identificativi siano univoci.

❖ **RNF-6: Modificabilità**

- Il sistema deve garantire la possibilità di modificare i suoi componenti e metodi facilmente.
- L'architettura software deve essere modulare al fine di semplificare un'operazione di modifica e manutenzione.

❖ **RNF-7: Comprensibilità**

- Il codice sorgente deve essere ben strutturato e ben organizzato.
- Il codice sorgente deve inoltre essere descritto da una accurata documentazione.

5. SCELTE DI PROGETTO

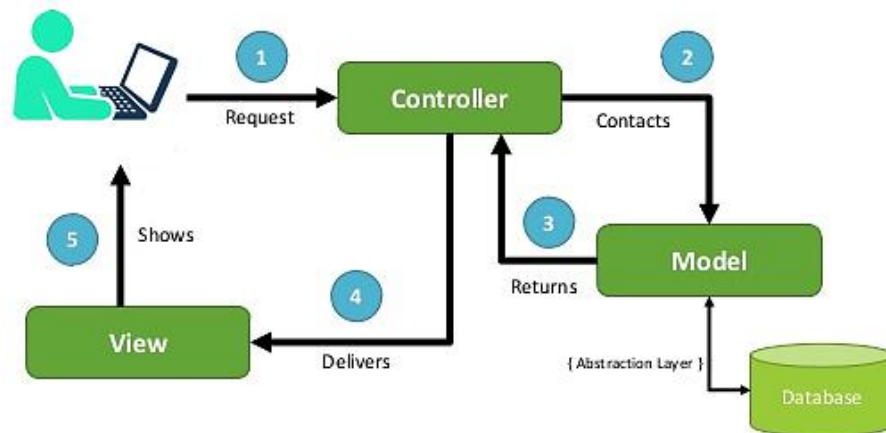
5.1. PATTERN ARCHITETTURALI UTILIZZATI

❖ Model-View-Controller

Il pattern MVC è una popolare architettura per la progettazione di applicazioni web. Definisce tre componenti principali:

- **Model**
Definisce i metodi per accedere e modificare i dati; rappresenta lo stato dell'applicazione. Può avere la responsabilità di notificare ai componenti View gli aggiornamenti dati da parte del Controller, in modo che si aggiornino continuamente.
- **View**
Delega al Controller i processi conseguenti l'input dell'utente. Ha la responsabilità della presentazione dei dati, in modo che siano sempre aggiornati.
- **Controller**
Realizza l'associazione tra model e View. Incapsula la logica della applicazione e seleziona le schermate della View da utilizzare.

MVC Design Pattern



5.2. STILI ARCHITETTURALI UTILIZZATI

❖ **Stile Object Oriented**

I componenti sono oggetti, con dati e operazioni annesse; i connettori sono dati scambiati tramite messaggi ed invocazione di metodi. Ogni oggetto è responsabile dell'integrità della loro rappresentazione interna e questa è nascosta ad altri oggetti. L'adozione di tale stile ci ha consentito di avere interfacce ben definite e di utilizzare principi di progettazione come l'information hiding e l'incapsulazione che aumentano la modificabilità.

❖ **Stile Client-Server**

L'architettura client-server è un particolare tipo di stile architetturale a due livelli che suddivide un'applicazione in due componenti principali: il client e il server. Il client implementa la presentation logic, mentre il server gestisce la logica di business, l'elaborazione delle richieste e l'accesso ai dati. Nel contesto dell'architettura client-server, il pacchetto View rappresenta il lato del client dell'applicazione, che si occupa dell'interfaccia utente e della generazione delle pagine HTML da visualizzare. Il client fa richieste al server per ottenere i dati o eseguire operazioni. I pacchetti Controller, Service e Repository rappresentano il lato del server dell'applicazione. Il pacchetto Controller riceve le richieste del client e coordina l'esecuzione delle operazioni appropriate. Il pacchetto Service contiene la logica di business dell'applicazione, mentre il pacchetto Repository gestisce l'accesso ai dati nel database. In questa architettura, il client interagisce con il server tramite richieste API REST. Il server elabora le richieste, recupera i dati necessari, esegue le operazioni richieste e restituisce le risposte al client. Alcuni vantaggi che questa scelta architetturale ha fornito sono:

➤ **Separazione delle responsabilità**

L'architettura client-server permette di separare chiaramente le responsabilità tra i due componenti. Il client si occupa dell'interazione con l'utente, della presentazione dei dati e dell'interfaccia utente, mentre il server gestisce la logica di business, l'elaborazione delle richieste e l'accesso ai dati.

➤ **Riutilizzo del codice**

La separazione tra client e server consente il riutilizzo del codice. La logica di business e le operazioni complesse possono essere implementate nel server e utilizzate da diversi client. Questo riduce la duplicazione del codice e favorisce la manutenibilità dell'applicazione.

➤ **Manutenibilità e Evolvibilità**

L'architettura client-server semplifica la manutenibilità dell'applicazione. I cambiamenti nella logica di business possono essere effettuati nel server senza dover apportare modifiche significative al client. Inoltre, è possibile aggiornare o sostituire il server senza influire sull'interfaccia utente o sui client esistenti.

❖ **Stile REpresentational State Transfer (REST)**

Le API REST sono uno stile architetturale per progettare e sviluppare servizi web. L'approccio REST si basa su un insieme di principi che promuovono l'interazione tra client e server attraverso richieste e risposte HTTP, utilizzando concetti come risorse, URL (Uniform Resource Locator) e metodi HTTP (GET, POST, PUT, DELETE). Nel sistema realizzato le API REST sono utilizzate per accedere ai metodi del package View e ai metodi del Controller. Esse rappresentano lo strumento che connette la parte front-end della applicazione con il rispettivo back-end. Grazie ai suoi standard aperti e alla sua natura leggera, le API REST possono essere facilmente integrate con diverse piattaforme, linguaggi di programmazione e framework. Inoltre consentono di realizzare il principio di separazione tra client e server che permette una maggiore modularità e manutenibilità dell'architettura.

Le scelte di progetto nello sviluppo software si riferiscono alle decisioni prese durante la pianificazione e progettazione di un'applicazione. Queste decisioni riguardano l'architettura del sistema, la scelta del linguaggio di programmazione, delle librerie e dei framework da utilizzare, il design dell'interfaccia utente e molti altri aspetti. Le scelte di progetto influenzano la scalabilità, la manutenibilità e le prestazioni del software, nonché la sua facilità di utilizzo da parte degli utenti finali.

6. TECNICHE DI SVILUPPO

Conclusa la progettazione, risulta importante focalizzarsi sulla scelta delle tecnologie da adoperare al fine di muovere i primi passi verso l'implementazione.

❖ HTML, JAVASCRIPT E CSS



HTML, CSS e JavaScript sono fondamentali per lo sviluppo del Front-End di un'applicazione web.

HTML (HyperText Markup Language) viene utilizzato per creare la struttura della pagina, definendo elementi come titoli, paragrafi, tabelle e form.

JavaScript permette di rendere la pagina interattiva, manipolando gli elementi HTML, gestendo gli eventi e implementando logiche di business.

CSS permette di rendere la pagina più chiara e facile da utilizzare, modificando il layout e la formattazione degli elementi HTML e JS.

❖ VISUAL STUDIO CODE



Visual Studio è uno strumento di sviluppo potente che è possibile usare per completare l'intero ciclo di sviluppo in un'unica posizione. Si tratta di un ambiente di sviluppo integrato completo (IDE) che è possibile usare per scrivere, modificare, eseguire il debug, compilare codice e distribuire l'app. Oltre alla modifica e al debug del codice, Visual Studio include compilatori, strumenti di completamento del codice, controllo del codice sorgente, estensioni e molte altre funzionalità per migliorare ogni fase del processo di sviluppo software.

❖ THYMELEAF



Thymeleaf è un motore di template per Java che consente di integrare la logica del Back-End con il Front-End in modo semplice ed efficace. Utilizzando Thymeleaf, è possibile incorporare tag ed espressioni all'interno di file HTML per generare dinamicamente il contenuto della pagina. Questo permette di manipolare i dati provenienti dal server e renderizzarli direttamente nel template HTML. Thymeleaf supporta anche funzionalità avanzate come cicli, condizioni e iterazioni, consentendo una gestione dinamica e flessibile del Front-End. Inoltre, Thymeleaf offre un'ottima integrazione con framework come Spring, rendendolo una scelta popolare per lo sviluppo web basato su Java.

❖ MYSQL



MySQL è un sistema di gestione di database relazionali ampiamente utilizzato nel Back-End delle applicazioni web. Per utilizzare MySQL nel Back-End, è necessario stabilire una connessione al database utilizzando le credenziali di accesso corrette. Una volta stabilita la connessione, è possibile eseguire query per inserire, recuperare, aggiornare o eliminare dati dal database. MySQL offre un'ampia gamma di funzionalità, tra cui l'indicizzazione dei dati, le transazioni, le viste e molto altro, che consentono di gestire e manipolare i dati in modo efficiente. È possibile utilizzare librerie o framework come JDBC o ORM (Object-Relational Mapping) per semplificare l'interazione con il database MySQL nel Back-End delle applicazioni web.

❖ XAMPP



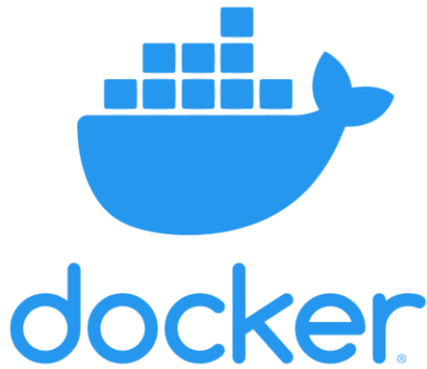
XAMPP è una distribuzione di software open-source che semplifica l'installazione e la configurazione di un ambiente di sviluppo per il Back-End. Utilizzando XAMPP, è possibile eseguire localmente un server web (Apache), un database (MySQL), un interprete di scripting (PHP) e altro ancora, il tutto in un'unica applicazione. Questo consente agli sviluppatori di creare e testare applicazioni Back-End senza dover configurare manualmente ciascun componente separatamente. XAMPP offre un'interfaccia utente intuitiva per la gestione dei servizi e delle configurazioni, semplificando l'avvio e la sospensione del server. Inoltre, XAMPP è multiplatforma e supporta sistemi operativi come Windows, macOS e Linux, rendendolo uno strumento versatile per lo sviluppo Back-End su macchine diverse.

❖ POSTMAN



Postman è uno strumento di sviluppo API che consente di testare, documentare e collaborare con le API nel processo di sviluppo Back-End. Utilizzando Postman, è possibile inviare richieste HTTP personalizzate agli Endpoint delle API, visualizzarne le risposte e analizzare i dati restituiti. Inoltre, offre funzionalità avanzate come l'automazione dei test, la generazione di documentazione API e la condivisione di collezioni di richieste con il team di sviluppo. È uno strumento essenziale per la fase di testing e debugging delle API, consentendo agli sviluppatori di garantire la correttezza e la robustezza delle loro applicazioni Back-End.

❖ DOCKER



Docker è una piattaforma di containerizzazione che consente di creare e gestire facilmente ambienti di sviluppo e produzione per il Back-End delle applicazioni. Utilizzando Docker per il Back-End, è possibile creare immagini dei componenti software, inclusi il server web, il database e altre dipendenze, in modo da poterli eseguire in modo isolato e riproducibile su diversi ambienti. Docker semplifica il processo di distribuzione delle applicazioni, garantendo che l'ambiente di sviluppo sia identico a quello di produzione, riducendo così problemi di compatibilità. Inoltre, Docker permette di scalare facilmente l'applicazione in base al carico di lavoro, sia verticalmente che orizzontalmente, grazie alla gestione dei container.

❖ HIBERNATE



Hibernate è un framework di Object-Relational Mapping (ORM) che semplifica l'interazione con il database nel Back-End delle applicazioni. Utilizzando Hibernate, è possibile mappare gli oggetti Java alle tabelle del database in modo trasparente, senza la necessità di scrivere query SQL manualmente. Hibernate gestisce automaticamente la persistenza degli oggetti nel database, permettendo di eseguire operazioni CRUD(Create, Read, Update, Delete) in modo semplice e intuitivo. Inoltre, Hibernate gestisce le relazioni tra gli oggetti, consentendo di definire associazioni come uno-a-uno, uno-a-molti e molti-a-molti. Grazie a Hibernate, le operazioni di accesso al database diventano più semplici e manutenibili, consentendo agli sviluppatori di concentrarsi sulla logica di business dell'applicazione.

❖ MAVEN



Maven è uno strumento di gestione delle dipendenze e di automazione della compilazione per i progetti software. Utilizzando Maven, è possibile definire il progetto tramite un file di configurazione denominato "pom.xml", specificando le dipendenze esterne necessarie per il progetto. Maven scaricherà automaticamente le dipendenze richieste e le includerà nel progetto. Inoltre, esso semplifica la compilazione, il testing e il packaging del progetto, consentendo di generare file JAR, WAR o altri formati. Maven offre anche la possibilità di definire script per eseguire compiti personalizzati, come la pulizia del progetto o l'esecuzione di test specifici. In generale, Maven semplifica notevolmente il processo di gestione e compilazione dei progetti software, fornendo una struttura coerente e prevedibile per lo sviluppo.

❖ VISUAL PARADIGM



Visual Paradigm è uno strumento di modellazione e progettazione che consente di creare diagrammi UML (Unified Modeling Language) e diagrammi di flusso per il processo di sviluppo software.

Utilizzando Visual Paradigm, è possibile visualizzare l'architettura e la struttura del sistema, definire classi, interfacce, relazioni e altre entità concettuali. Inoltre, esso supporta la generazione automatica di codice da modelli UML, facilitando l'implementazione del software. Il tool offre anche funzionalità per la collaborazione e la gestione dei requisiti, consentendo a team di sviluppatori di lavorare in modo coordinato e di mantenere traccia delle modifiche nel ciclo di sviluppo. In sintesi, Visual Paradigm aiuta a creare modelli e documentazione di qualità per il processo di sviluppo software.

7. MODELLO DEI CASI D'USO

7.1. COMPONENTI PRINCIPALI

Relativamente al task da implementare, è possibile definire la seguente classificazione:

❖ Attori primari

- Amministratore
- Utente Registrato
- Utente Autenticato

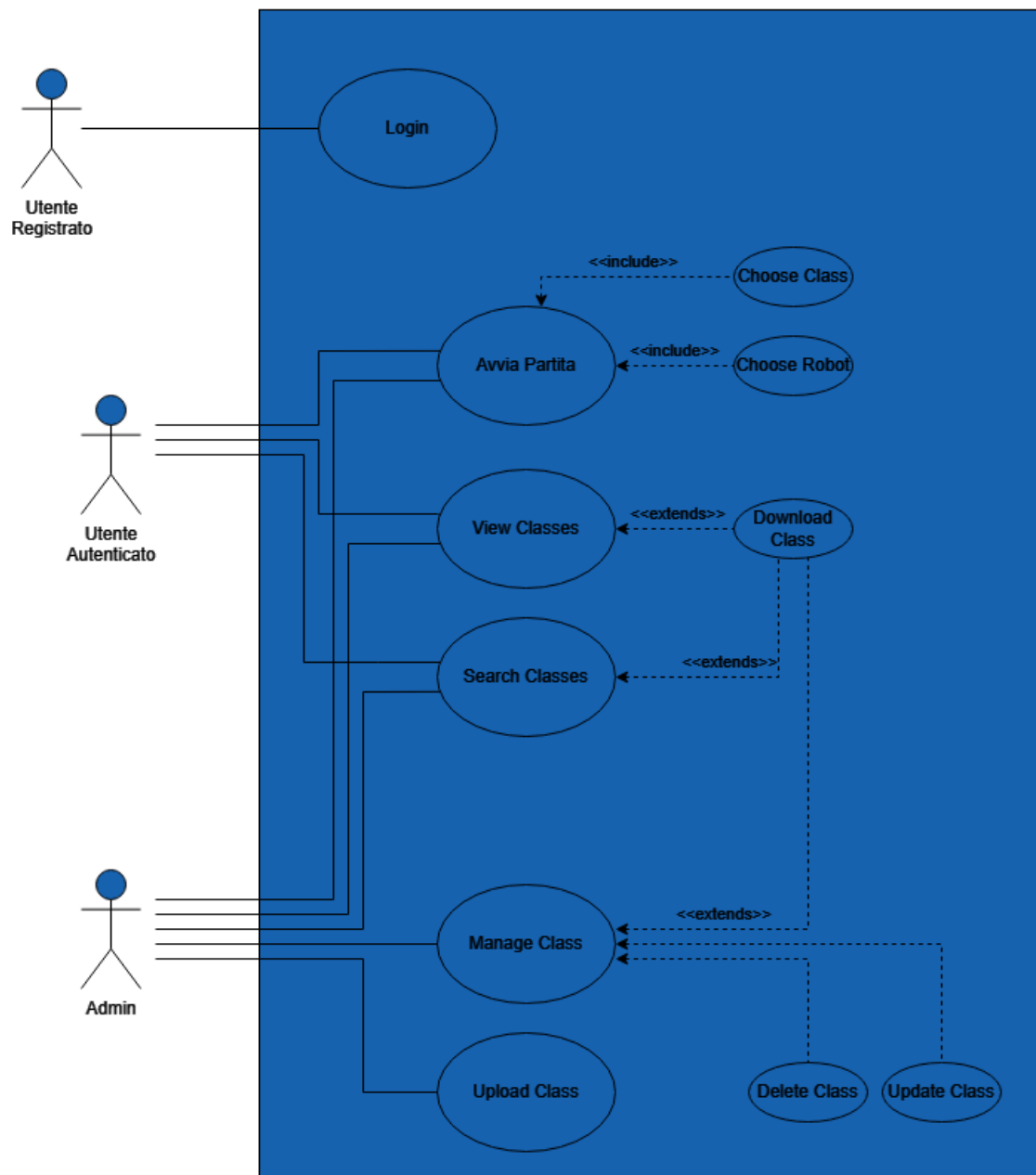
❖ Attori secondari

- Browser Web
- Repository
- Sistema

❖ Casi d'uso

- **UC1:** Registrazione
- **UC2:** Login
- **UC3:** Visualizza elenco
- **UC4:** Avvia partita
- **UC5:** Salva partita
- **UC6:** Scelta robot
- **UC7:** Scelta classe
- **UC8:** Download classe
- **UC9:** Gestione classi
- **UC10:** Modifica classe
- **UC11:** Aggiungi classe
- **UC12:** Elimina classe

7.2. DIAGRAMMA DEI CASI D'USO



7.3. SCENARI CASI D'USO

A conclusione di tale capitolo, segue un'analisi approfondita di ciascun caso d'uso con indicazioni sugli attori (primari e secondari), una breve descrizione, precondizioni e post-condizioni, sequenze di eventi principali e alternative.

➤ UC1: Login

Caso d'uso:	Login
Attore primario	Utente registrato
Attore secondario	Browser Web, Repository
Descrizione	L'utente inserisce i dati per il login
Pre-Condizioni	L'utente deve essere registrato [Mock]
Sequenza di eventi Principale	1. L'utente inserisce username e password per il login
Sequenza di eventi alternativi	1. Se l'utente inserisce un username o password sbagliata, l'applicazione restituisce un messaggio di errore
Post-Condizioni	L'utente registrato effettua il login con successo.

➤ UC2: View Classes

Caso d'uso:	View Classes
Attore primario	Utente autenticato
Attore secondario	Browser Web, Repository
Descrizione	Il giocatore visualizza l'elenco delle classi Java da testare
Pre-Condizioni	L'elenco delle classi da testare non è vuoto
Sequenza di eventi Principale	1. Il giocatore seleziona l'opzione "View Classes" sulla homepage dell'applicazione 2. Il sistema mostra l'elenco delle Classi Java da testare presenti nel database
Post-Condizioni	L'elenco viene correttamente mostrato dall'interfaccia web dell'applicazione.

➤ **UC3: Avvia Partita**

Caso d'uso:	Avvia Partita
Attore primario	Utente autenticato, Amministratore
Attore secondario	Browser Web, Repository
Descrizione	Il giocatore seleziona le impostazioni della partita.
Pre-Condizioni	L'elenco delle classi da testare non è vuoto.
Sequenza di eventi Principale	<ol style="list-style-type: none"> 1. Il giocatore seleziona la classe Java da testare 2. Il giocatore seleziona il robot con cui competere 3. Il giocatore clicca su "Play" con le impostazioni da lui selezionate
Post-Condizioni	La partita viene avviata con successo.

➤ **UC4: Choose Robot**

Caso d'uso:	Choose Robot
Attore primario	Utente autenticato, Amministratore
Attore secondario	Browser Web, Repository
Descrizione	Il giocatore seleziona il robot con cui competere.
Pre-Condizioni	//
Sequenza di eventi Principale	<ol style="list-style-type: none"> 1. Il giocatore seleziona un robot tra quelli disponibili
Post-Condizioni	Il robot è stato selezionato correttamente.

➤ **UC5: Choose Class**

Caso d'uso:	Choose Class
Attore primario	Utente autenticato, Amministratore
Attore secondario	Browser Web, Repository
Descrizione	Il giocatore seleziona la classe da testare.
Pre-Condizioni	La lista delle classi non è vuota.
Sequenza di eventi Principale	<ol style="list-style-type: none"> 1. Il giocatore seleziona una classe tra quelle disponibili
Post-Condizioni	La classe è stata selezionata correttamente.

➤ **UC6: Download Class**

Caso d'uso:	Download Class
Attore primario	Utente autenticato, Amministratore
Attore secondario	Browser Web, Repository, Sistema
Descrizione	Il giocatore effettua il download di una delle classi Java disponibili
Pre-Condizioni	L'elenco delle classi da testare non è vuoto
Sequenza di eventi Principale	<ol style="list-style-type: none"> 1. Il giocatore o l'amministratore seleziona la classe Java da scaricare 2. Il giocatore o l'amministratore sceglie l'opzione "Download" per la classe selezionata 3. Il sistema fornisce il file .java della classe scelta.
Post-Condizioni	Il file .java viene correttamente scaricato, dal repository, nella cartella Download.

➤ **UC7: Update Class**

Caso d'uso:	Update Class
Attore primario	Amministratore
Attore secondario	Browser Web, Repository
Descrizione	L'amministratore modifica una classe presente nel database
Pre-Condizioni	La classe è già presente nel database
Sequenza di eventi Principale	<ol style="list-style-type: none"> 1. L'amministratore seleziona la classe da modificare 2. L'amministratore apporta le modifiche alla classe e le salva
Post-Condizioni	La lista delle classi è aggiornata con la classe modificata

➤ **UC8: Upload Class**

Caso d'uso:	Upload Class
Attore primario	Amministratore
Attore secondario	Browser Web, Repository
Descrizione	L'amministratore aggiunge una nuova classe
Pre-Condizioni	La classe non è già presente del database
Sequenza di eventi Principale	<ol style="list-style-type: none"> 1. L'amministratore inserisce le caratteristiche della nuova classe: <ul style="list-style-type: none"> • LOC • Complexity • Last Update • Opponent Recommended 2. L'amministratore carica la classe .java e clicca su "Submit"
Post-Condizioni	La lista delle classi è aggiornata con la nuova classe.

➤ **UC9: Delete Class**

Caso d'uso:	Delete class
Attore primario	Amministratore
Attore secondario	Browser Web, Repository
Descrizione	L'amministratore elimina una classe tra quelle disponibili
Pre-Condizioni	La classe è già presente del database
Sequenza di eventi Principale	<ol style="list-style-type: none"> 1. L'amministratore seleziona una classe 2. L'amministratore clicca sull'icona del cestino
Post-Condizioni	La lista delle classi è aggiornata con la classe eliminata

➤ **UC10: Search Classes**

Caso d'uso:	Search Classes
Attore primario	Utente autenticato
Attore secondario	Browser Web, Repository
Descrizione	Ricerca di una classe in base a determinate caratteristiche
Pre-Condizioni	La classe è già presente del database
Sequenza di eventi Principale	<ol style="list-style-type: none"> 1. L'utente clicca su "Search Classes" 2. L'utente inserisce le caratteristiche della classe da cercare 3. L'applicazione gli mostra le classi con quelle caratteristiche
Post-Condizioni	L'applicazione restituisce le classi con le caratteristiche inserite dall'utente

➤ **UC11: Manage Classes**

Caso d'uso:	Manage Classes
Attore primario	Amministratore
Attore secondario	Browser Web, Repository
Descrizione	Il giocatore visualizza l'elenco delle classi Java per modificarle, eliminarle o scaricarle
Pre-Condizioni	//
Sequenza di eventi Principale	<ol style="list-style-type: none"> 1. L'amministratore seleziona l'opzione "Manage Classes" sulla home page dell'applicazione. 2. Il sistema mostra l'elenco delle Classi Java da testare presenti nel repository.
Post-Condizioni	L'elenco viene correttamente mostrato dall'interfaccia web dell'applicazione.

8. SYSTEM DOMAIN MODEL

Il System Domain Model aiuta a comprendere e definire i concetti chiave, le entità, le relazioni e i comportamenti all'interno del dominio specifico del sistema. Un System Domain Model è un modello concettuale di un sistema che descrive le varie entità che fanno parte o hanno rilevanza nel sistema e le loro relazioni. Il modello di dominio è utile per mettere a fuoco i concetti fondamentali di un sistema e definire un vocabolario specifico. In particolare, il System Domain Model rappresenta il modello di dominio di un sistema software.

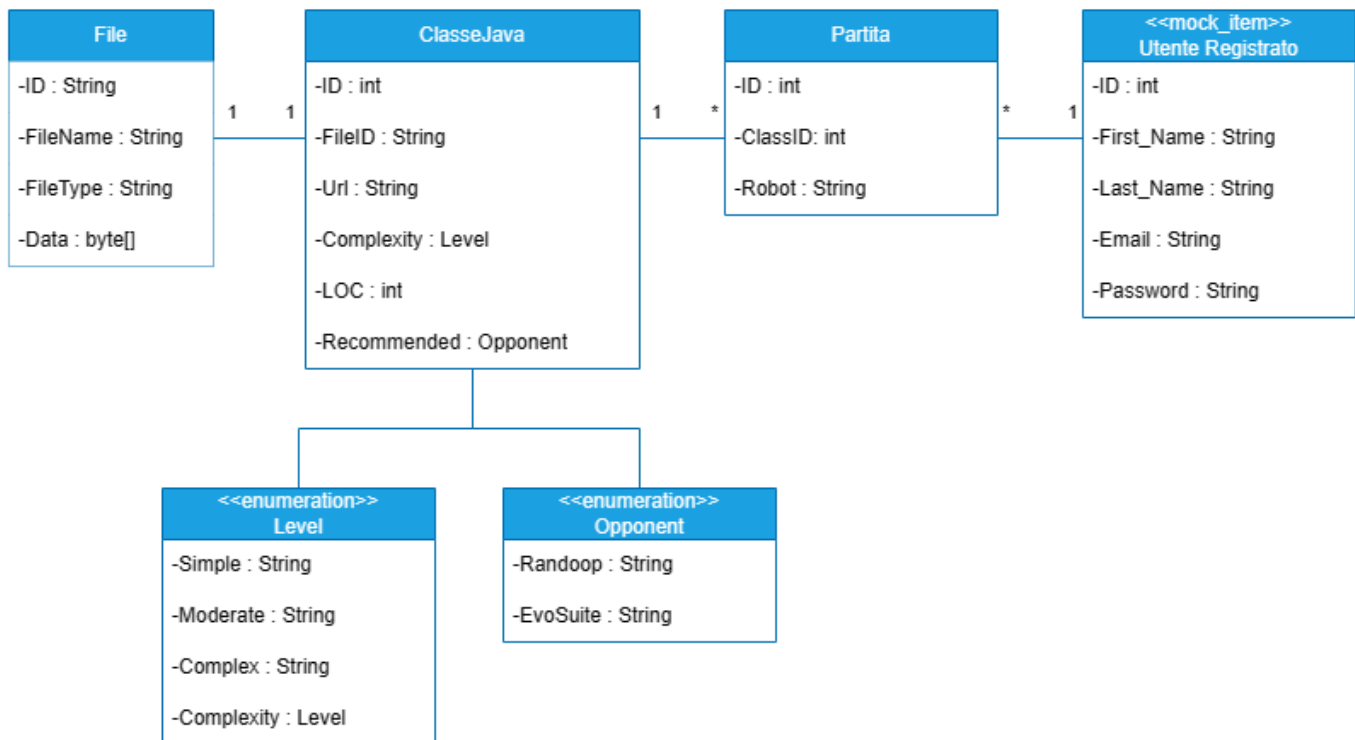
Il System Domain Model rappresentato comprende:

- ClasseJava
- File
- Partita
- Utente Registrato

Le relazioni presenti sono:

- ClasseJava – File [1:1]
Collegata tramite la chiave esterna: FileID – ID
- ClasseJava – Partita [1:*]
Collegata tramite la chiave esterna: ID – ID
- Partita – Utente Registrato [*:1]
Collegata tramite la chiave esterna: ID – ID

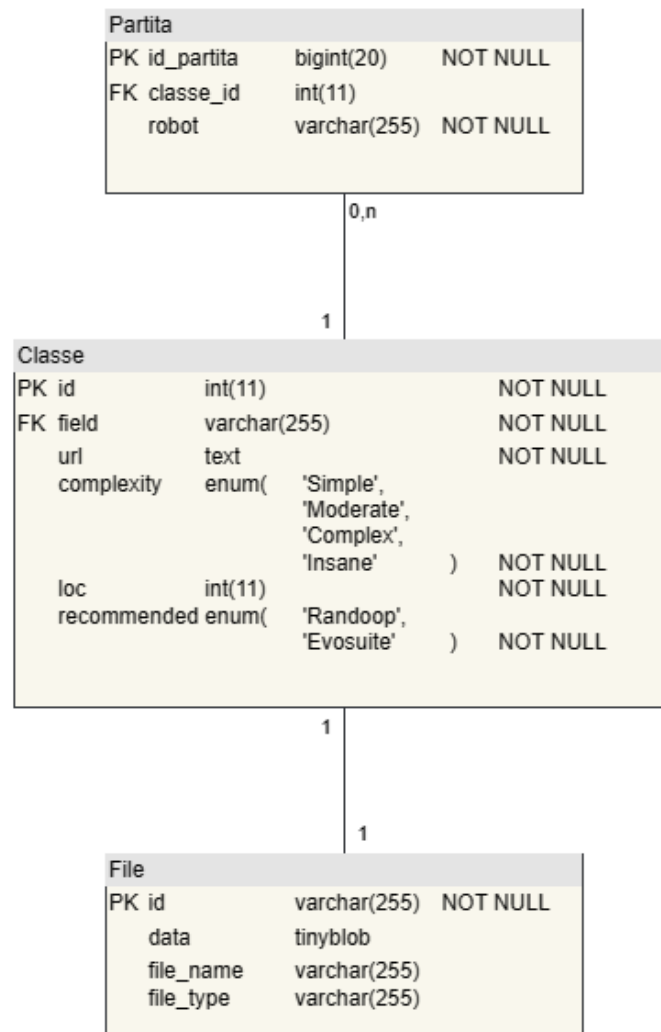
L'entità ClasseJava presenta due attributi di tipo enumerazione: Level e Opponent. Entrambi si compongono di una stringa selezionabile fra quattro alternative per il tipo Level e due alternative per il tipo Opponent.



9. MODELLAZIONE DEI DATI

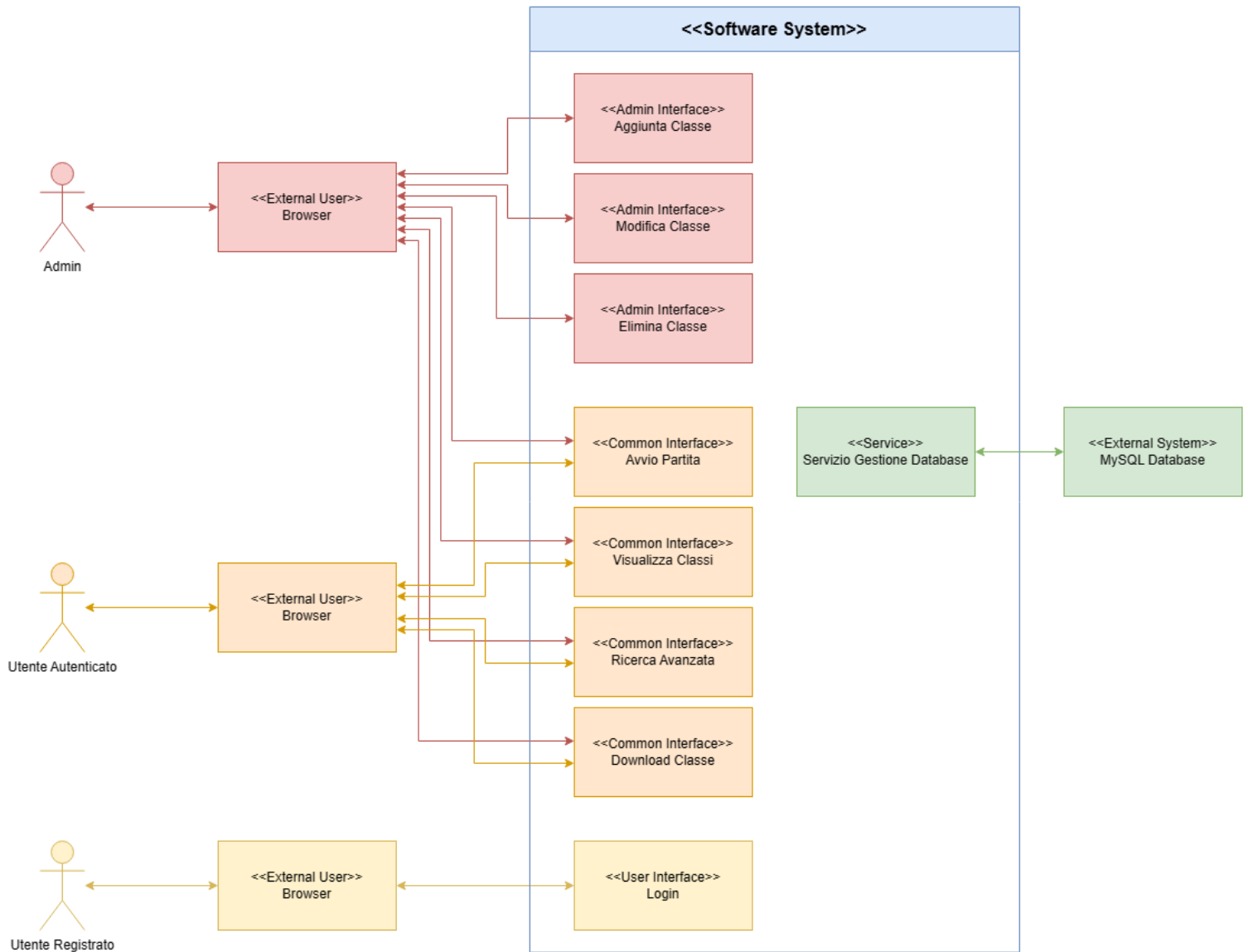
Il diagramma di modellazione dei dati è un tipo di diagramma che rappresenta la struttura di un database e le relazioni tra le tabelle. Questo tipo di diagramma è utilizzato per rappresentare i dati e le relazioni tra di essi in modo intuitivo. In particolare, il diagramma di modellazione dei dati può essere utilizzato per definire la struttura di un database relazionale e per descrivere le relazioni tra le tabelle.

Il database risulta essere composto da tre tabelle, ossia da collezioni di dati organizzati in righe e colonne possedenti una chiave primaria.



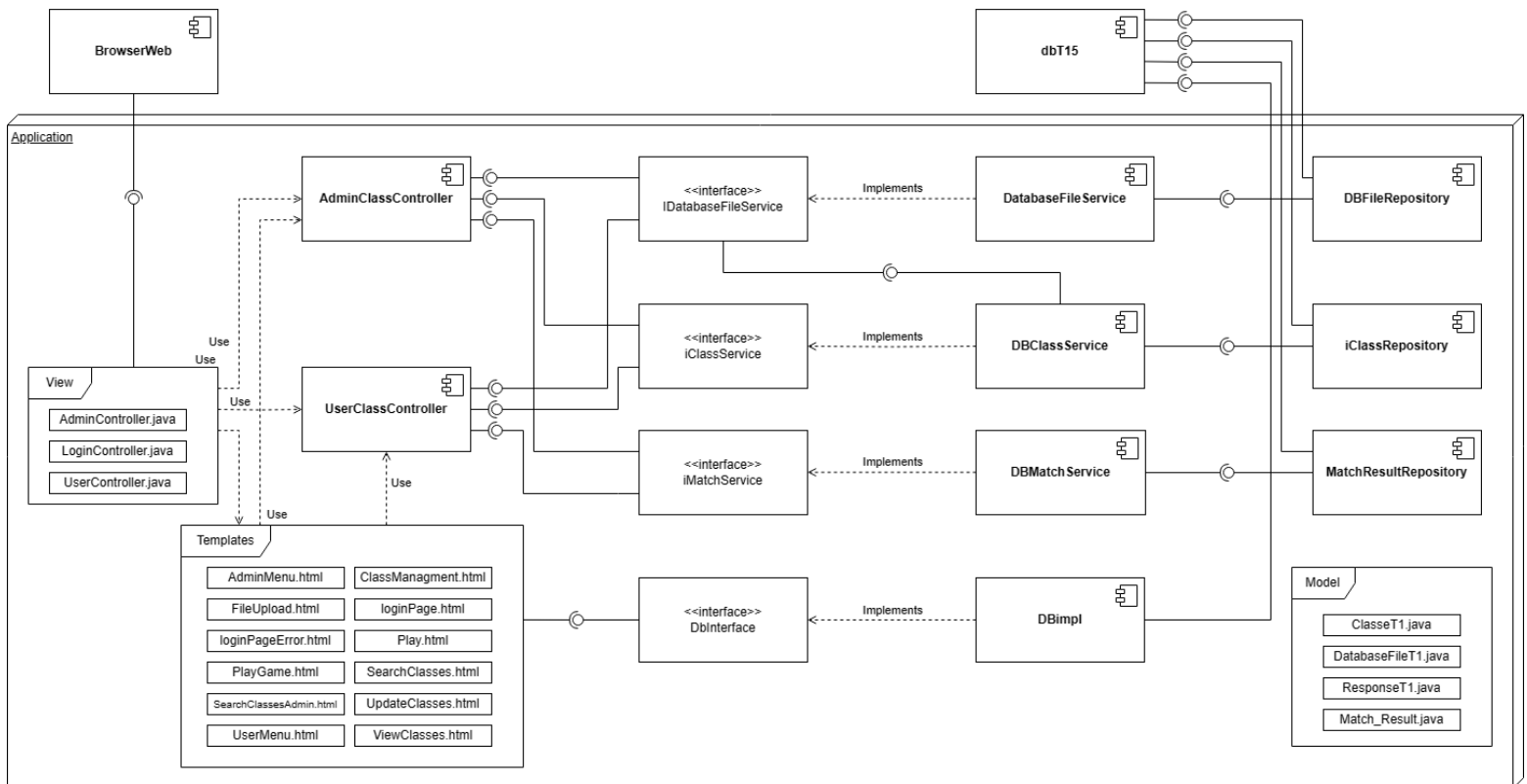
10.CONTEXT DIAGRAM

Il Context Diagram è un ulteriore diagramma ad alto livello che rappresenta il contesto operativo del sistema e mette in evidenza le interazioni con gli attori esterni. Fornisce, dunque, una visione panoramica dell'architettura senza entrare nei dettagli interni del sistema.



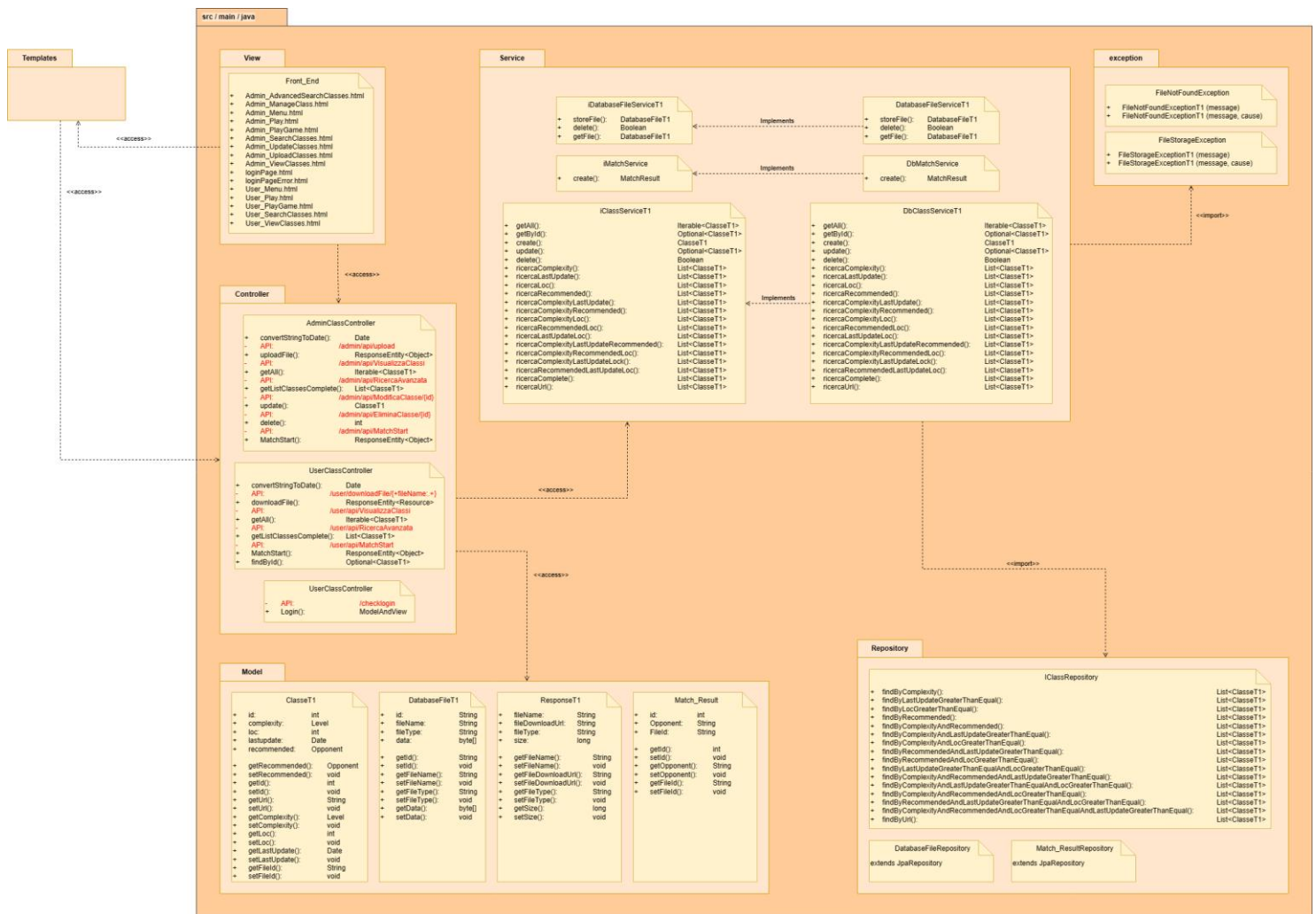
11.COMPONENT DIAGRAM

Il diagramma dei componenti viene utilizzato per descrivere l'architettura software di un sistema, esso mostra i componenti del sistema e le relazioni tra di essi.

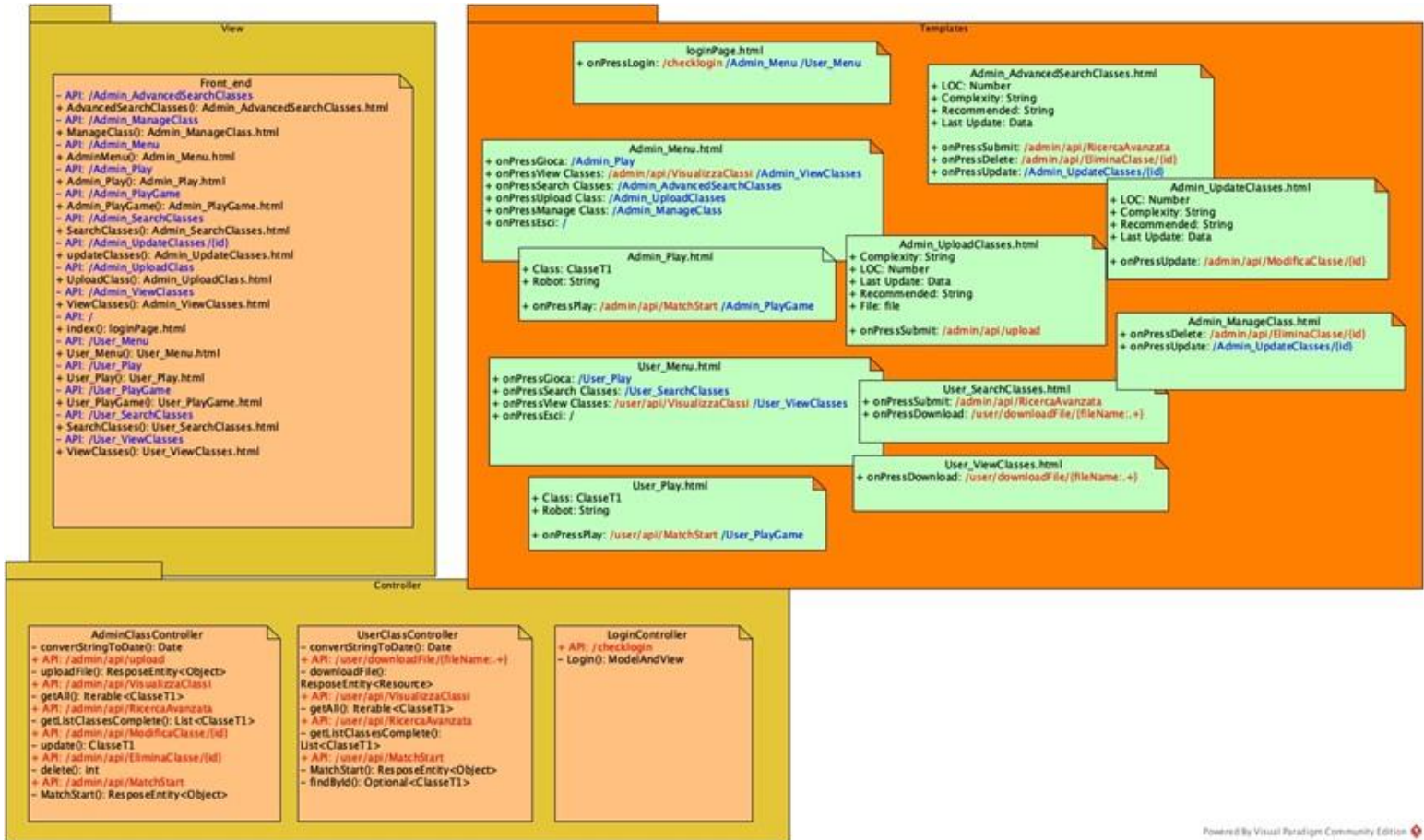


12. PACKAGE DIAGRAM

Un Package Diagram è un tipo di diagramma UML che permette di descrivere la struttura di un sistema attraverso dei grafi in cui i nodi rappresentano i package e gli archi le dipendenze tra di essi. Lo scopo dei diagrammi dei package UML è organizzare gli elementi in gruppi per fornire una struttura migliore per un modello di sistema e illustrare le dipendenze tra i diversi package di questo sistema. Un package è il blocco predefinito di un diagramma e rappresenta un raggruppamento di elementi di un modello. Questi elementi possono essere diagrammi, documenti, classi o anche altri package.



Di seguito si riporta una rappresentazione estesa del contenuto dei package View e Controller e della cartella contenente i file .html, per rendere più chiara l'interazione tra essi. I file .html all'interno contengono script in linguaggio JavaScript che richiamano i metodi del controller o della classe Front_End tramite le API REST. Analogamente i metodi contenuti nella classe Front_End del package View sono richiamati dal Browser Web attraverso le corrispondenti API, e in base al metodo richiamato viene selezionato l'apposito file HTML.



13.API

13.1. INTRODUZIONE E PROPRIETÀ

Le REST API, acronimo di "**Representational State Transfer Application Programming Interface**", sono un insieme di principi e protocolli che consentono la comunicazione e lo scambio di dati tra sistemi software. Le REST API si basano sull'architettura REST, che è un approccio comune nello sviluppo di servizi web. I vantaggi delle REST API nel contesto dello sviluppo software sono diversi:

- **Scalabilità e flessibilità**
Le REST API permettono di creare servizi web altamente scalabili, in grado di gestire un gran numero di richieste simultanee da parte degli utenti. L'architettura REST si basa su risorse indipendenti, consentendo l'aggiunta o la rimozione di risorse senza influire sulle altre. Le REST API consentono di distribuire il carico su più server o risorse. Ciò significa che è possibile allocare le risorse in modo ottimale, garantendo che ogni richiesta venga gestita in modo efficiente e senza sovraccaricare un singolo server.
- **Interoperabilità**
Le REST API utilizzano protocolli di comunicazione standard come HTTP e HTTPS, che sono ampiamente supportati e comprensibili da diverse tecnologie e piattaforme. Ciò consente a sistemi diversi di comunicare e scambiare dati senza problemi.
- **Separazione dei concetti**
Le REST API seguono il principio di separazione tra client e server. Ciò significa che il server fornisce l'accesso ai dati e ai servizi, mentre il client è responsabile della presentazione dei dati e dell'interazione con l'utente. Questa separazione permette di sviluppare e modificare indipendentemente il client e il server, facilitando la manutenzione e l'evoluzione del sistema.
- **Semplicità**
Le REST API utilizzano una serie di operazioni standard, come GET, POST, PUT e DELETE, per accedere e manipolare le risorse. Questo rende l'interfaccia delle API intuitiva e facile da usare per gli sviluppatori, riducendo la curva di apprendimento e accelerando lo sviluppo delle applicazioni.

Le applicazioni delle REST API sono ampie e diverse. Alcuni esempi comuni includono:

- **Integrazione di sistemi**
Le REST API consentono a diverse applicazioni o sistemi di comunicare tra loro, scambiando dati e informazioni. Ad esempio, un'applicazione mobile può utilizzare le REST API per inviare richieste a un server remoto e ottenere i dati necessari per fornire una funzionalità specifica.
- **Sviluppo di servizi web**
Le REST API sono ampiamente utilizzate per lo sviluppo di servizi web che forniscono accesso a dati e funzionalità tramite Internet. Le applicazioni possono utilizzare queste API per ottenere informazioni da diverse fonti, come social media, servizi di pagamento o dati di terze parti.
- **Creazione di applicazioni distribuite**
Le REST API consentono lo sviluppo di applicazioni distribuite, in cui diverse parti del sistema possono essere sviluppate indipendentemente e comunicare tra loro tramite API. Questa

flessibilità consente di creare applicazioni complesse, in cui diverse funzionalità possono essere sviluppate separatamente e integrate successivamente.

In sintesi, le REST API offrono una soluzione efficiente e scalabile per la comunicazione tra sistemi software, consentendo lo sviluppo di applicazioni distribuite, l'integrazione di servizi e l'accesso a dati e funzionalità di terze parti. Grazie alla loro semplicità e flessibilità, le REST API sono diventate uno degli standard più diffusi nello sviluppo di applicazioni web e mobile.

13.2. DESCRIZIONE FORMALE DELLE REST API

In questa sezione si fornisce la descrizione contenente le informazioni essenziali sulle API implementate, inclusi i percorsi (paths) e i relativi metodi HTTP consentiti, nonché i parametri richiesti o opzionali per ciascun percorso.

La documentazione delle API è di fondamentale importanza per aiutare gli sviluppatori a comprendere come interagire correttamente con un software attraverso le API. Essa fornisce una guida chiara e strutturata sulle funzionalità disponibili, sui parametri richiesti, sui formati di dati accettati e sulle eventuali restrizioni. Attraverso questa descrizione delle API, gli sviluppatori possono esplorare i percorsi disponibili, comprendere i metodi consentiti (come GET, POST, PUT e DELETE) e i parametri richiesti per ciascuna richiesta. Questo aiuta gli sviluppatori a integrare correttamente le tue API nel loro software, inviando richieste valide e gestendo le risposte in modo appropriato.

13.3. TEST REST API IN ADMINCLASSCONTROLLER

Di seguito si riporta il testing delle API di AdminClassController:

1. /admin/api/upload

The screenshot shows a Postman interface for a POST request to `http://localhost:8080/admin/api/upload`. The request is configured with the following parameters:

Key	Value
jsonData	<code>{"complexity": "INSANE", "loc": "70", "lastupdate": "2023-09-25", "recommended": "EVOSUL..."}</code>
file	<code>C:\Users\raf19\Desktop\Esempio.json</code>
Key	Value

The response status is **200 OK** with a time of **41 ms** and a size of **193 B**. The response body is displayed as `1 File is uploaded successfully`.

2. /admin/api/VisualizzaClassi

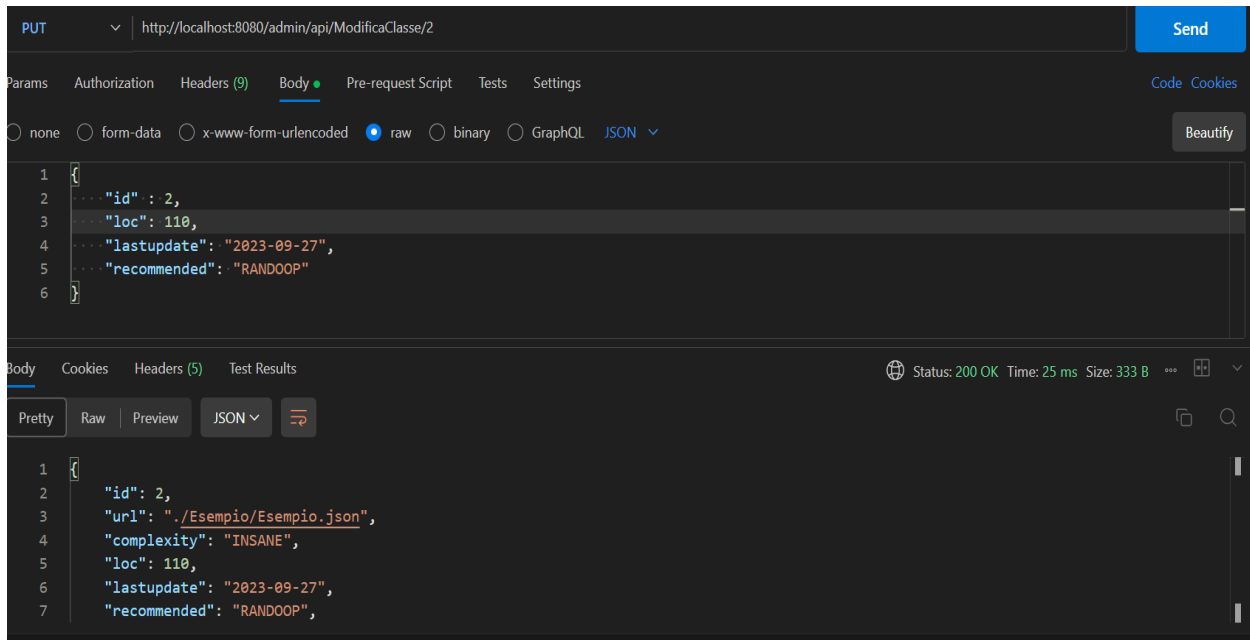
The screenshot shows a Postman interface for a GET request to `http://localhost:8080/admin/api/VisualizzaClassi`. The response status is **200 OK** with a time of **15 ms** and a size of **335 B**. The response body is displayed as a JSON object:

```

1 {
2   "id": 2,
3   "url": "/Esempio/Esempio.json",
4   "complexity": "INSANE",
5   "loc": 70,
6   "lastupdate": "2023-09-25",
7 }

```

3. /admin/api/ModificaClasse/{id}



PUT | http://localhost:8080/admin/api/ModificaClasse/2

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings Code Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL ☐ JSON

```

1 {
2   "id": 2,
3   "loc": 110,
4   "lastupdate": "2023-09-27",
5   "recommended": "Randoop"
6 }

```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 25 ms Size: 333 B

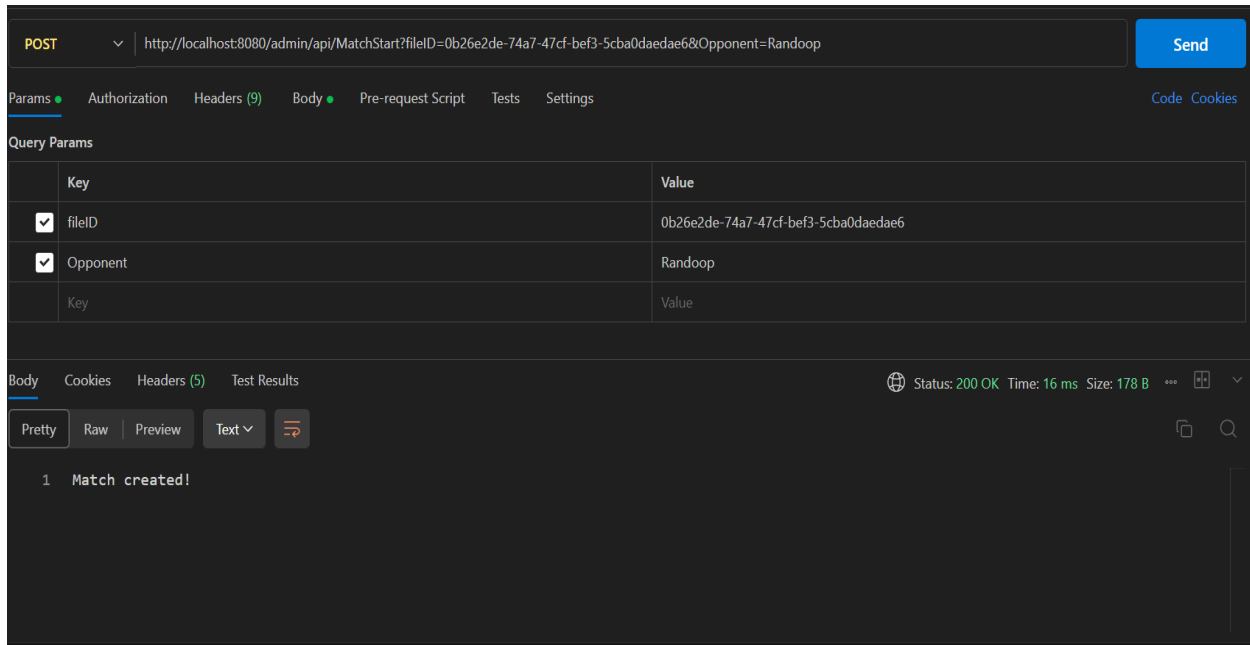
Pretty Raw Preview **JSON**

```

1 {
2   "id": 2,
3   "url": "./Esempio/Esempio.json",
4   "complexity": "INSANE",
5   "loc": 110,
6   "lastupdate": "2023-09-27",
7   "recommended": "Randoop",

```

4. /admin/api/MatchStart



POST | http://localhost:8080/admin/api/MatchStart?fileID=0b26e2de-74a7-47cf-bef3-5cba0daedae6&Opponent=Randoop

Params **Authorization** Headers (9) Body Pre-request Script Tests Settings Code Cookies

Query Params

Key	Value
<input checked="" type="checkbox"/> fileID	0b26e2de-74a7-47cf-bef3-5cba0daedae6
<input checked="" type="checkbox"/> Opponent	Randoop
Key	Value

Body Cookies Headers (5) Test Results Status: 200 OK Time: 16 ms Size: 178 B

Pretty Raw Preview **Text**

```

1 Match created!

```


5. /admin/api/EliminaClasse/{id}

DELETE ▼ http://localhost:8080/admin/api/EliminaClasse/2 Send

Params Authorization Headers (9) Body ● Pre-request Script Tests Settings Code Cookies

Query Params

Key	Value
Key	Value

Body Cookies Headers (5) Test Results ⊕ Status: 200 OK Time: 23 ms Size: 165 B ⋮ ⌵

Pretty Raw Preview JSON ▼ ⌵

```
1 {}
```

6. /admin/api/RicercaAvanzata

GET ▼ http://localhost:8080/admin/api/RicercaAvanzata?loc=110 Send

Params ● Authorization Headers (9) Body ● Pre-request Script Tests Settings Code Cookies

Query Params

Key	Value
<input checked="" type="checkbox"/> loc	110
Key	Value

Body Cookies Headers (5) Test Results ⊕ Status: 200 OK Time: 19 ms Size: 336 B ⋮ ⌵

Pretty Raw Preview JSON ▼ ⌵

```
1 {
2   {
3     "id": 3,
4     "url": "./Esempio/Esempio.json",
5     "complexity": "INSANE",
6     "loc": 110,
7     "lastupdate": "2023-09-25",
8   }
9 }
```

13.4. TEST REST API IN USERCLASSCONTROLLER

7. /user/downloadFile/{filename:.+}

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** http://localhost:8080/user/downloadFile/84b73f5c-e3fa-4e19-a9a6-4b49dcc130c2
- Body Type:** form-data
- Form Data:**

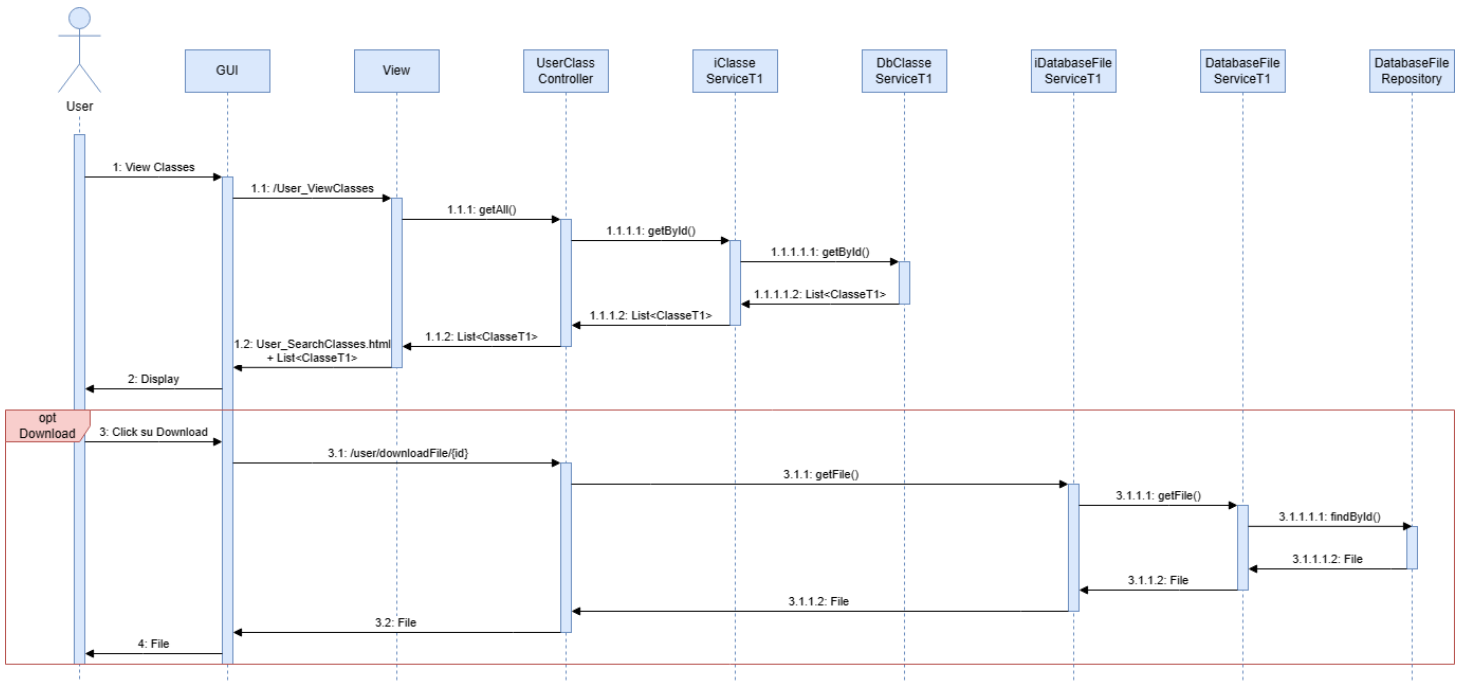
Key	Value
jsondata	{"complexity": "INSANE", "loc": "110", "lastupdate": "2023-09-25", "recommended": "EVOS..."}
file	C:\Users\raf19\Desktop\Esempiodownload.json
Key	Value
- Status:** 200 OK
- Time:** 11 ms
- Size:** 264 B
- Body Content:** 1 Questo è un esempio

14.SEQUENCE DIAGRAMS

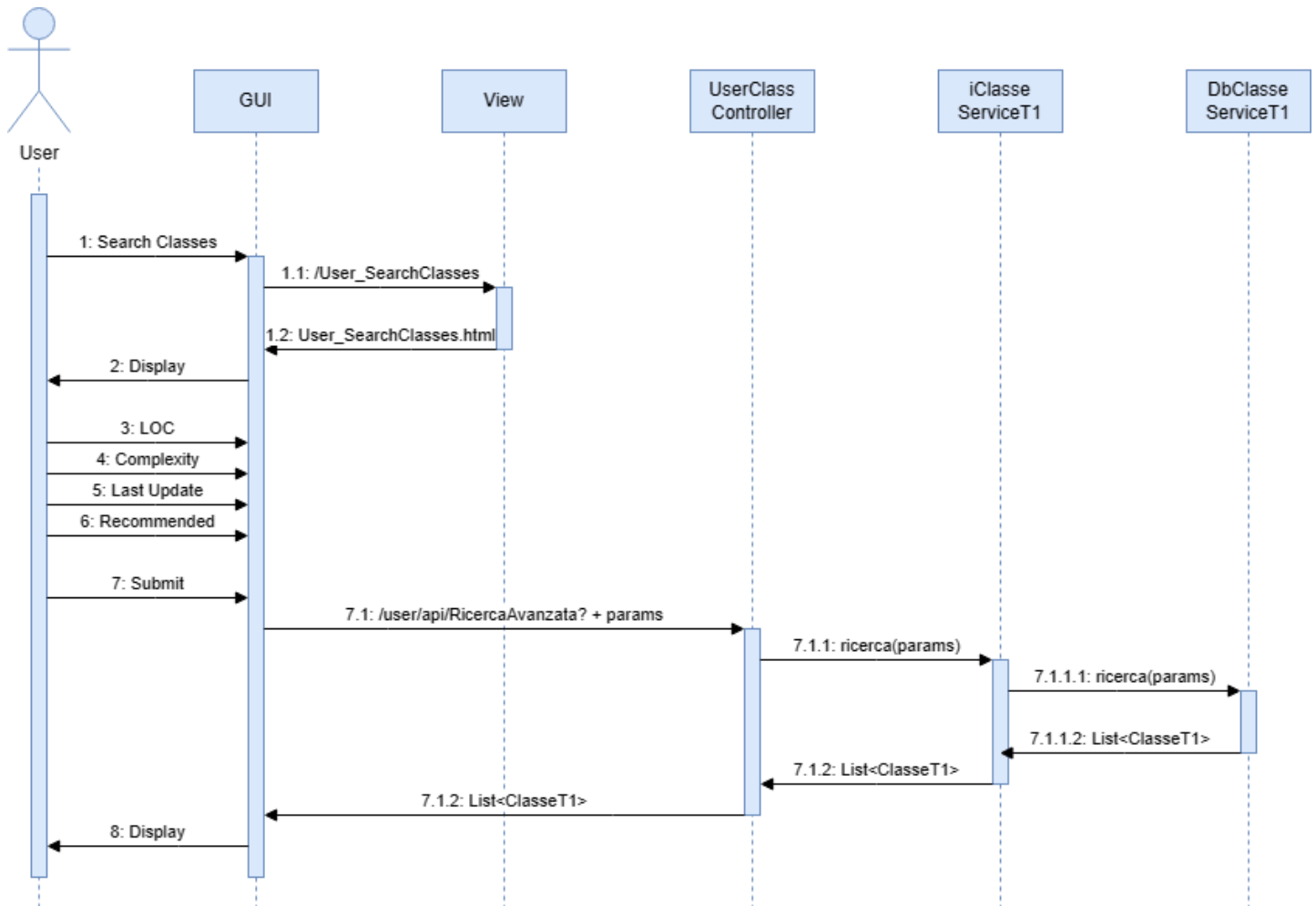
I diagrammi di sequenza sono uno strumento di modellazione utilizzato per rappresentare l'interazione tra oggetti o componenti di un sistema nel tempo. Sono composti da una linea verticale che rappresenta il tempo e da frecce che mostrano il flusso di controllo tra gli oggetti. I diagrammi di sequenza consentono di visualizzare le azioni, i messaggi e le chiamate di metodo tra gli oggetti, fornendo una panoramica chiara del comportamento dinamico del sistema. Sono utili per comprendere l'ordine delle operazioni, le dipendenze temporali e le collaborazioni tra gli elementi del sistema.

Si usano inoltre per rappresentare l'interazione tra i componenti del pattern Model-View-Controller (MVC). Essi consentono di visualizzare chiaramente come avvengono le chiamate di metodo e i messaggi tra il Model, la View e il Controller. Questo aiuta a comprendere il flusso delle operazioni e le dipendenze tra i componenti, facilitando la progettazione e la comprensione del comportamento dinamico del sistema. I diagrammi di sequenza possono evidenziare l'ordine temporale delle azioni, inclusi gli eventi scatenati dall'utente, le richieste del Controller al Model e il successivo aggiornamento della View.

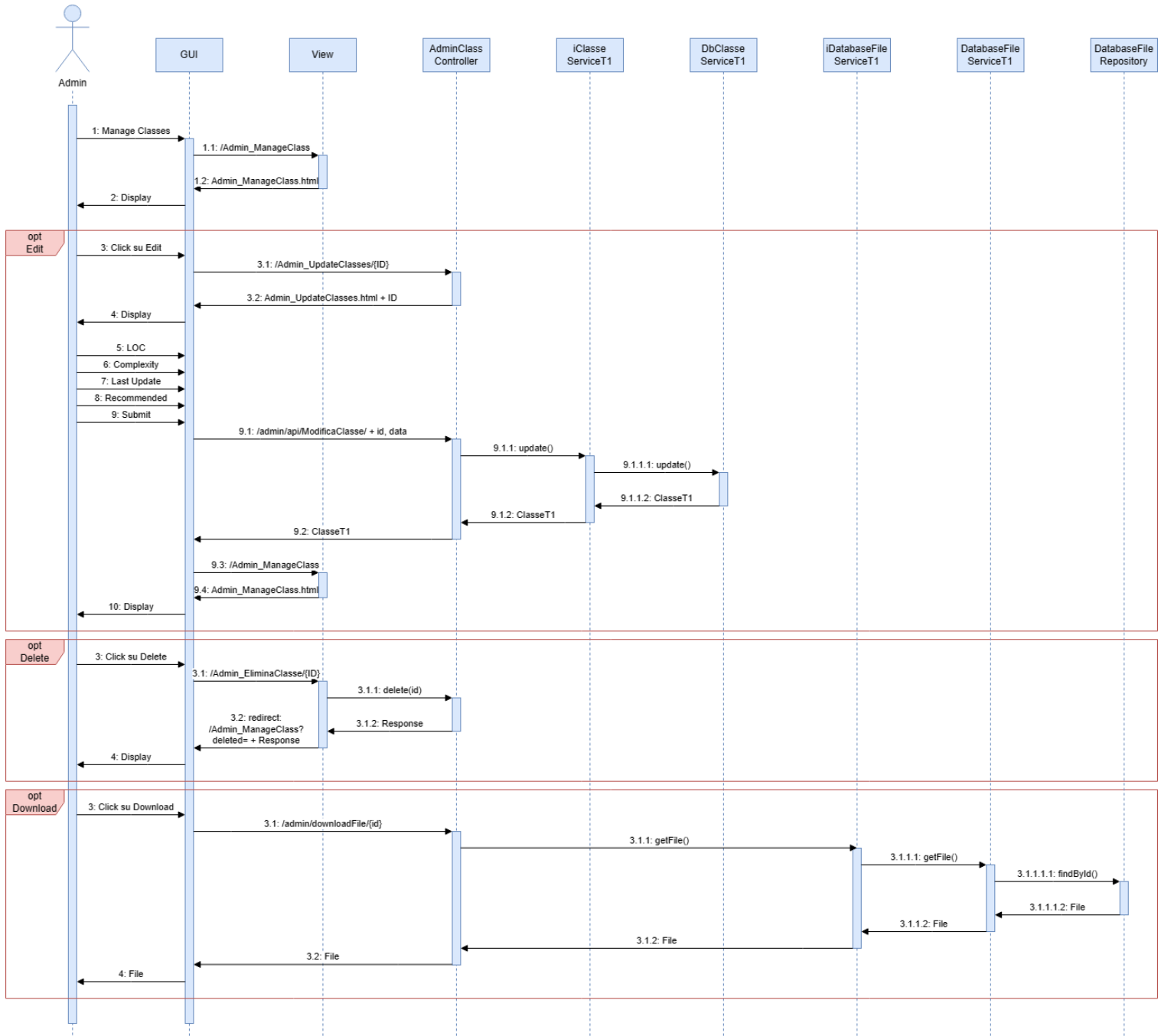
14.1. VISUALIZZACLASSE E DOWNLOADCODICECLASSE



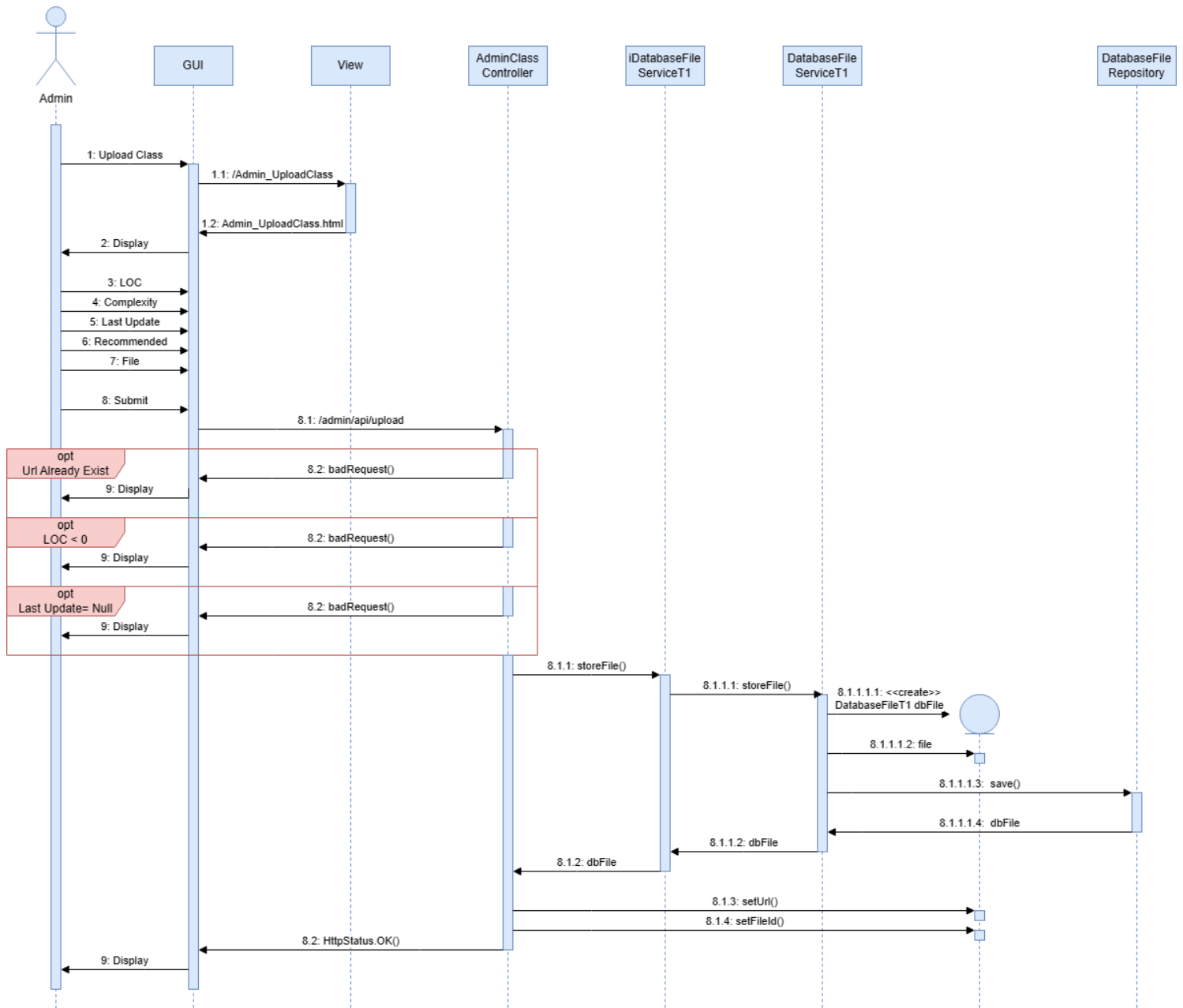
14.2. RICERCAAVANZATA



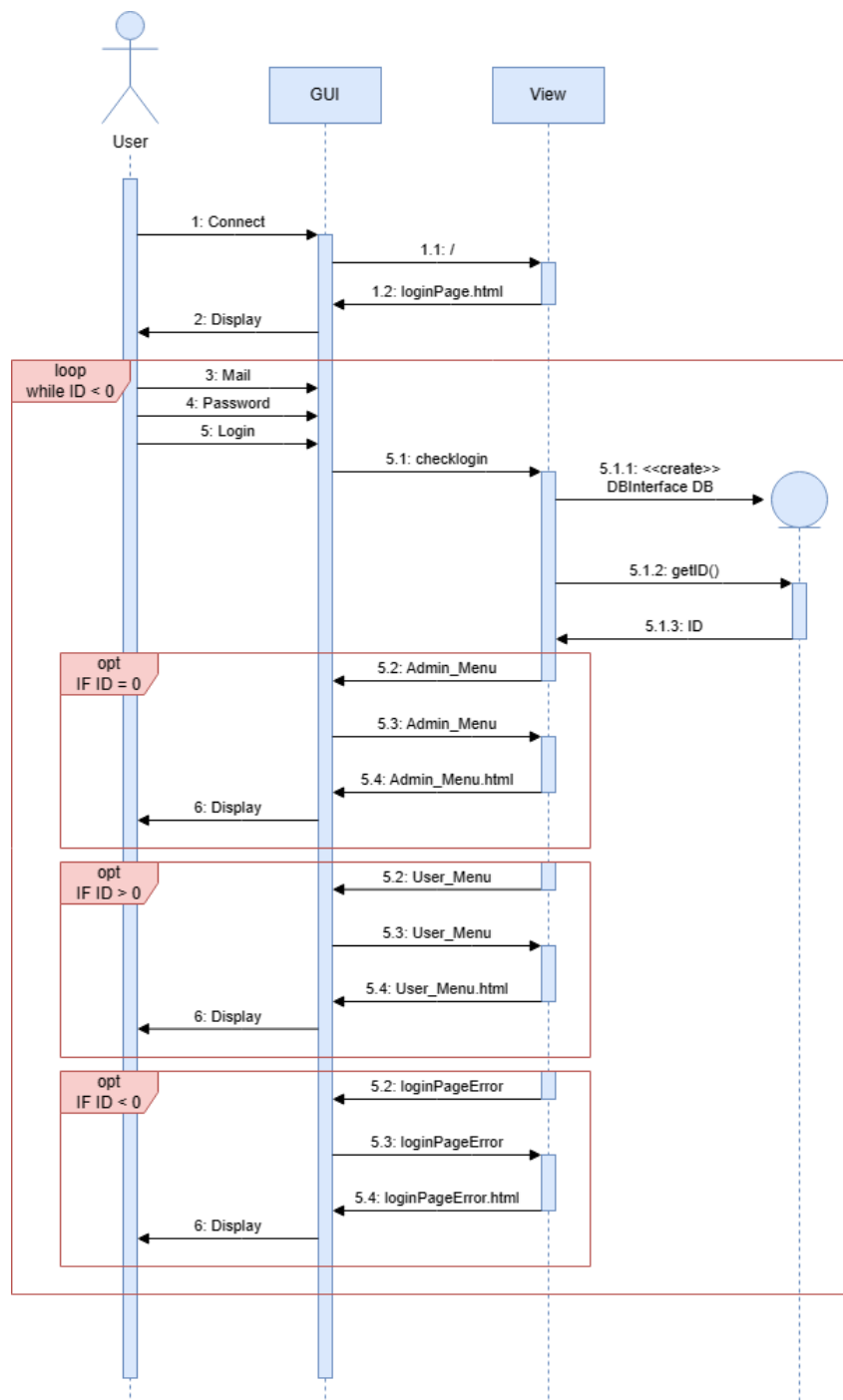
14.3. GESTIONECLASSE



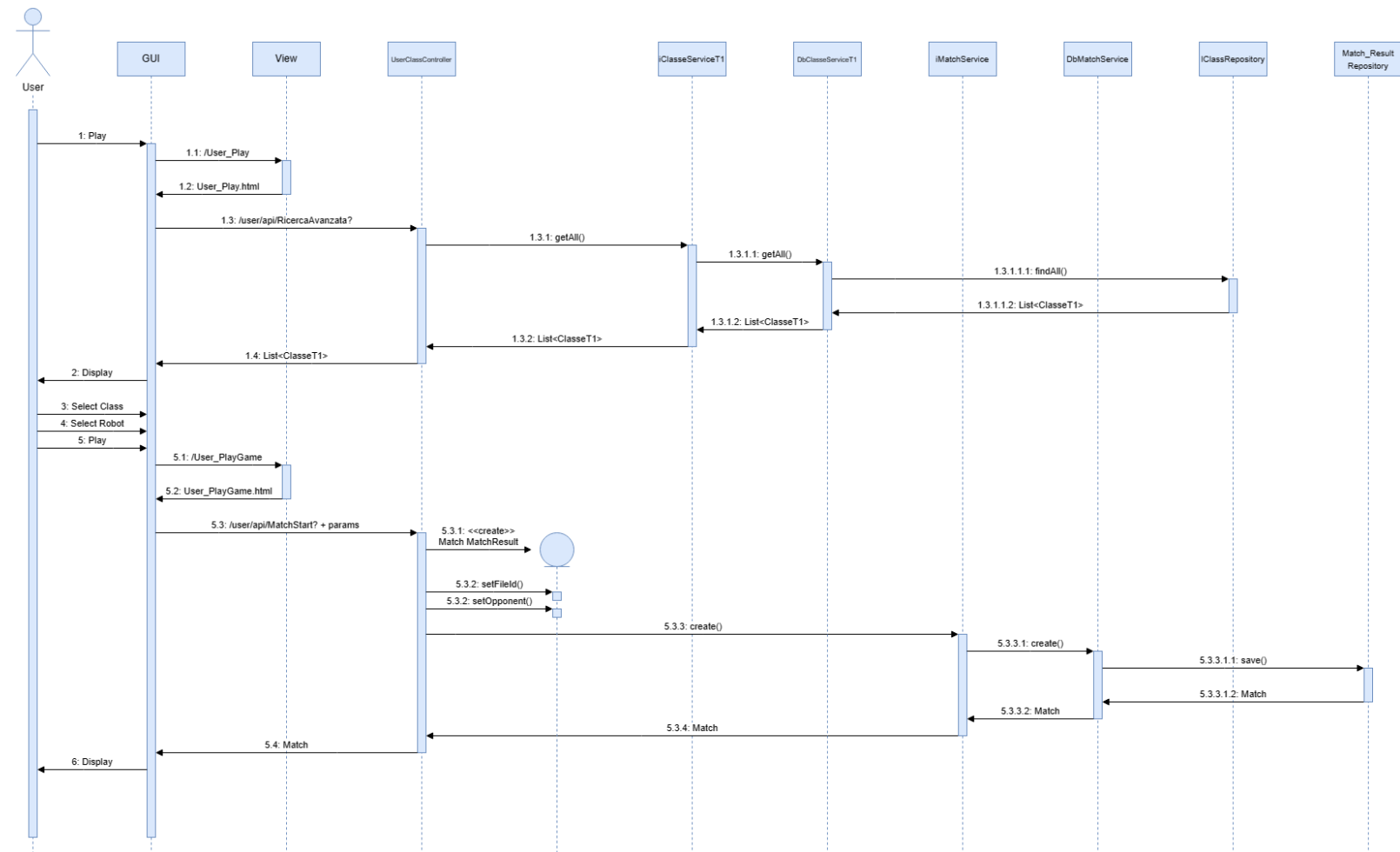
14.4. AGGIUNTA CLASSE



14.5. LOGIN



14.6. AVVIO PARTITA



15.STIMA DEI COSTI

Per stimare i costi e i tempi necessari per completare un progetto di sviluppo software, è possibile utilizzare la tecnica degli Use Case Points (UCP). Questo metodo si basa sui requisiti del sistema definiti tramite casi d'uso e si applica quando viene impiegato UML per la progettazione e sviluppo del software. L'UCP consente di stimare la dimensione del software necessaria per il progetto, includendo fattori tecnici ed ambientali. Utilizzando questo metodo, è possibile calcolare l'effort necessario per completare il progetto. La dimensione del software (UCP) viene calcolata in base agli elementi dei casi d'uso, utilizzando un fattore di correzione per considerare le diverse variabili. In questo modo, l'UCP rappresenta uno strumento utile per la stima dei costi e dei tempi associati ad un progetto di sviluppo software basato su UML.

15.1. UNADJUSTED USE CASE WEIGHT (UUCW)

I valori di riferimento assegnati a ciascun caso d'uso, detti anche USE CASE CLASSIFICATION, sono:

- Simple (1-3 transiction) Weight 5.
- Average (4-7 transiction) Weight 10.
- Complex (8 or more transiction) Weight 15.

Use Case	Complessità
Login	Simple
Avvia Partita	Average
Choose Class	Simple
Choose Robot	Simple
View Classes	Simple
Search Classes	Simple
Download Class	Average
Manage Class	Simple
Upload Class	Average
Delete Class	Simple
Update Class	Simple

Valutazione dell'UUCW:

$$UCCW = (TotalN. of SimpleUC * 5) + (TotalN. AverageUC * 10) + (TotalN. ComplexUC * 15)$$

$$UCCW = 70$$

15.2. UNADJUSTED ACTOR WEIGHT (UAW)

Actor	Complessità
Utente Registrato	Simple
Utente Autenticato	Average
Amministratore	Average

Valutazione dell'UAW:

$$UAW = (N. of Simple Actors * 1) + (N. Average Actors * 2) + (N. Complex Actors * 3)$$

$$UAW = 5$$

15.3. TECHNICAL COMPLEXITY FACTOR (TCF)

Il TCF è uno dei fattori applicati alla dimensione stimata del software al fine di tenere conto delle considerazioni tecniche del sistema. Si determina assegnando un punteggio compreso tra 0 (il fattore è irrilevante) e 5 (il fattore è essenziale) a ciascuno dei 13 fattori tecnici elencati nella tabella seguente.

Questo punteggio viene quindi moltiplicato per il valore ponderato definito per ciascun fattore. I valori di riferimento:

- T1 - Sistema distribuito, 2.0
- T2 - Tempo di risposta/obiettivi prestazionali, 1.0
- T3 - Efficienza dell'utente finale, 1.0
- T4 - Complessità di elaborazione interna, 1.0
- T5 – Riutilizzabilità del codice, 1.0
- T6 - Facile da installare, 0.5
- T7 - Facile da usare, 1.0
- T8 - Portabilità su altre piattaforme, 2.0
- T9 - Manutenzione del sistema, 1.0
- T10 - Elaborazione simultanea/parallela, 1.0
- T11 - Funzioni di sicurezza, 0.5
- T12 - Accesso per terze parti, 1.0.
- T13 - Formazione dell'utente finale, 1.0.

Per il sistema software in esame si usano i seguenti valori di fattori: T1 (3), T2 (4), T3 (5), T4 (2), T5 (5), T6 (4), T7 (5), T8 (4), T9 (3), T10 (3), T11 (2), T12 (5), T13 (1). Il totale di tutti i valori calcolati e il fattore tecnico (TF), calcolato come segue: $TF = 3 * 2 + 4 * 1 + 5 * 1 + 2 * 1 + 5 * 1 + 4 * 0.5 + 5 * 1.0 + 4 * 2 + 3 * 1 + 3 * 1 + 2 * 0.5 + 5 * 1 + 1 * 1 = 50$ Il TF viene quindi utilizzato per calcolare il TCF con la seguente formula:

$$TCF = 0.6 + (TF/100) = 0.6 + (50/100) = 0.6 + 0.5 = 1.1$$

15.4. ENVIRONMENTAL COMPLEXITY FACTOR (ECF)

L'ECF è un fattore applicato alla dimensione stimata del software che tiene conto delle considerazioni ambientali del sistema. Viene determinato assegnando un punteggio compreso tra 0 (nessuna esperienza) e 5 (esperto) a ciascuno degli 8 fattori ambientali elencati nella tabella seguente. Questo punteggio viene quindi moltiplicato per il valore ponderato definito per ciascun fattore. Valori di riferimento:

- E1 - Familiarità con il processo di sviluppo utilizzato, 1.5
- E2 - Esperienza applicativa, 0.5
- E3 - Esperienza di team orientata agli oggetti, 1.0
- E4 - Capacità di lead analyst, 0.5
- E5 - Motivazione del team, 3.0
- E6 - Stabilità dei requisiti, 2.0
- E7 - Personale part-time, -1.0
- E8 - Linguaggio di programmazione difficile, 1.0

Per il sistema software in esame si usano i fattori: E1 (2) + E2 (2) + E3 (2) + E4 (3) + E5 (5) + E6 (4) + E7 (0) + E8 (2). Il totale di tutti i valori calcolati e il fattore ambiente (EF), calcolato come segue: $EF = 2 * 1.5 + 2 * 0.5 + 2 * 1 + 3 * 0.5 + 5 * 1 + 4 * 2 + 0 * (-1) + 2 * (1) = 22.5$ L'EF viene quindi utilizzato per calcolare l'ECF con la seguente formula:

$$ECF = 1.4 + (-0.03 * 22.5) = 0.725$$

15.5. TOTAL USE CASE POINTS (UCP)

Infine, una volta determinate le dimensioni del progetto Unadjusted (UUCW e UAW), e calcolati il fattore tecnico (TCF) e il fattore ambientale (ECF) si può calcolare l'UCP. L'UCP viene calcolato in base alla seguente formula: $UCP = (UUCW + UAW) * TCF * ECF = (70 + 5) * 1.1 * 0.725 = 59.8125$. Da

quest'ultimo parametro e possibile valutare il numero di ore di lavoro totali considerando che un singolo UC venga sviluppato in 8 ore mediamente:

$$Th = UCP * 8 = 59.8125 * 8 = 478.5h$$

In definitiva, se si suppone che un membro del progetto lavori per circa 30 ore a settimana (Wh), ed il team sia composto da 4 membri, essi lavoreranno per un totale di 120 ore a settimana (TWh). Th è il numero di ore di lavoro totali necessarie a terminare il progetto e TWh rappresenta il numero di ore di lavoro a settimana del gruppo di lavoro (team), dunque facendone il rapporto si ottiene il numero di settimane necessarie per terminare il lavoro:

$$Th / TWh = 478.5 / 120 = 4 \text{ settimane}$$

16.DEPLOYMENT

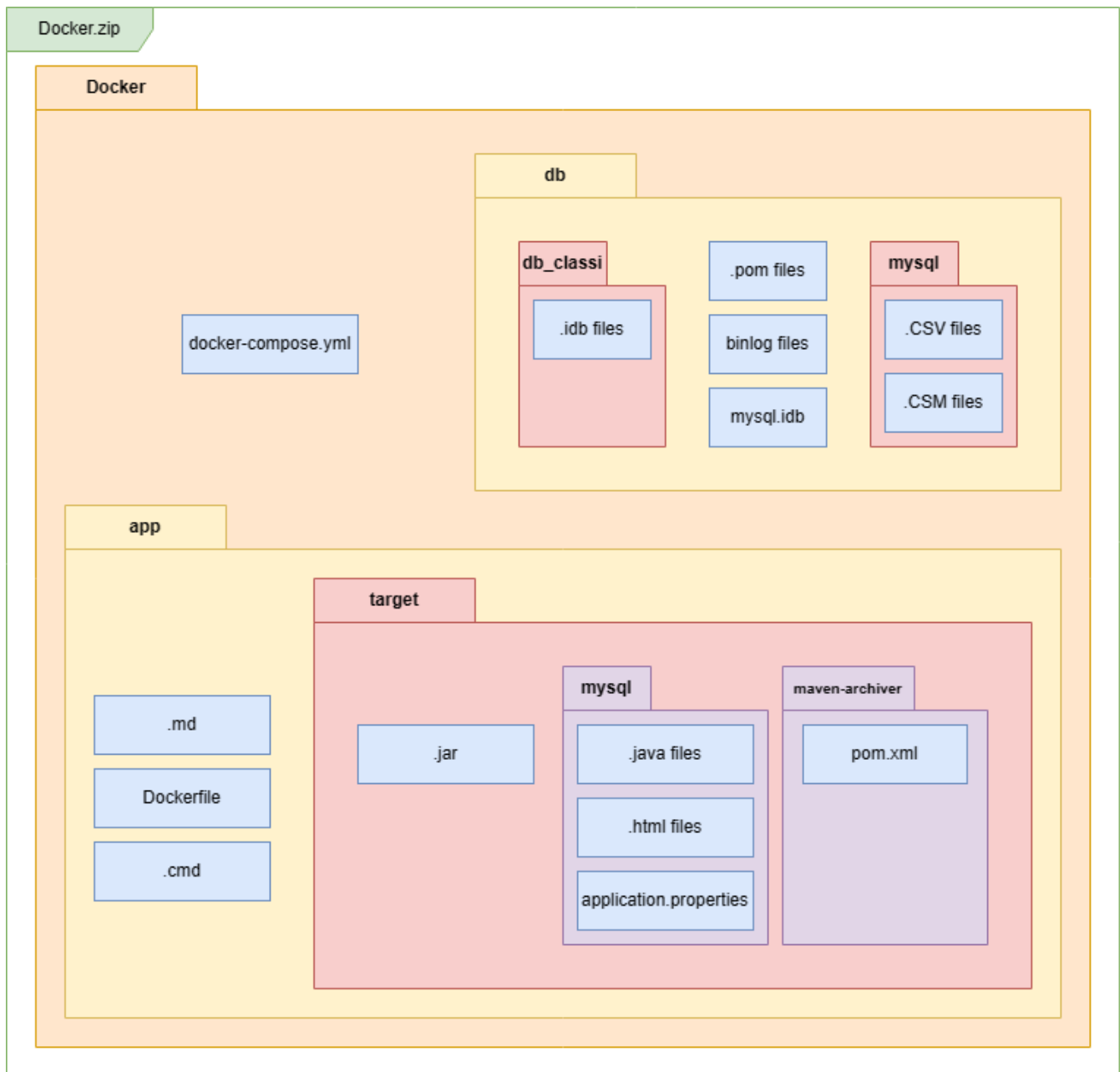
Il Deployment, nell'ambito del software e delle tecnologie dell'informazione, si riferisce al processo di distribuzione e messa in esercizio di un'applicazione o di un sistema informatico in un ambiente operativo. È una fase cruciale dello sviluppo software in cui il prodotto finale viene reso disponibile agli utenti finali. Durante il deployment, l'applicazione o il sistema viene installato, configurato e attivato su uno o più server o dispositivi di destinazione. Questo può includere sia l'hardware fisico sia le risorse virtuali, come i server cloud. L'obiettivo principale del deployment è consentire agli utenti di interagire con l'applicazione o il sistema in modo efficiente e affidabile. Il processo di deployment può coinvolgere molteplici attività, come la preparazione dell'ambiente di produzione, la gestione delle dipendenze e delle configurazioni, il controllo di versione, il testing delle funzionalità, la configurazione dei parametri di sicurezza e molto altro. È importante garantire che l'applicazione o il sistema siano pronti per l'uso in termini di prestazioni, affidabilità e sicurezza. Il deployment può essere automatizzato utilizzando strumenti e pratiche di Continuous Integration e Continuous Deployment (CI/CD), che consentono di ridurre gli errori umani e di semplificare il processo complessivo. Inoltre, le moderne architetture di deployment, come i contenitori e le tecnologie di orchestrazione, come Docker-Compose, offrono una maggiore scalabilità e flessibilità per gestire ambienti complessi e distribuiti.

In conclusione, il deployment è un processo essenziale per portare un'applicazione o un sistema informatico dallo sviluppo all'ambiente di produzione, garantendo che sia configurato correttamente, funzioni in modo affidabile e soddisfi le esigenze degli utenti finali. Un deployment efficace richiede pianificazione, automazione, monitoraggio e gestione continua per garantire il successo e il corretto funzionamento dell'applicazione o del sistema.

Nella seguente sezione, si descrive la procedura di deployment utilizzata per distribuire il software in oggetto:

1. Aprire Docker Desktop.
2. Dal seguente link GitHub, scaricare la versione aggiornata del progetto.
<https://github.com/Testing-Game-SAD-2023/T10-G36>
3. Estrarre il file zip scaricato
4. Aprire un terminale da amministratore e posizionarsi sul percorso dove è stato estratto il progetto, successivamente digitare:
> docker-compose up
Il comando viene utilizzato per avviare i servizi definiti in un file di configurazione docker-compose.yml già presente all'interno della cartella. Viene creata l'immagine del container e viene eseguito il running all'interno di docker.
5. A questo punto per utilizzare l'applicazione basterà recarsi sul proprio browser web e digitare il seguente link: <http://localhost:8080/>.
6. Dopo aver effettuato l'accesso, sarà possibile utilizzare il menu per avere accesso alle diverse funzionalità richieste al task T1-T5.

16.1. INSTALL VIEW

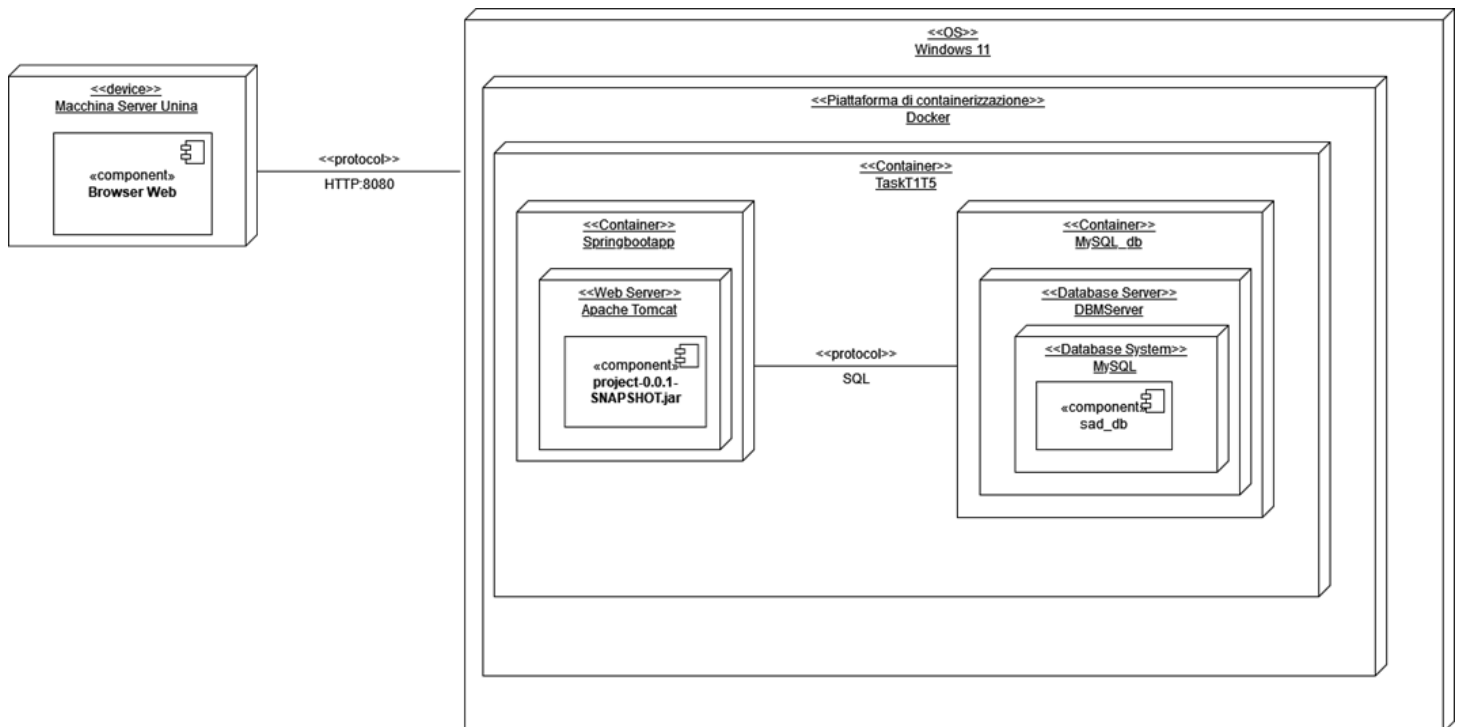


16.2. DIAGRAMMA DI DEPLOYMENT

Il diagramma di deployment è una rappresentazione grafica dell'architettura di un sistema software che illustra come i suoi componenti hardware e software sono distribuiti e interconnessi. Mostra la disposizione fisica dei nodi di elaborazione, come server, client, dispositivi mobili, e le connessioni tra di essi. Il diagramma evidenzia la distribuzione delle applicazioni, dei servizi e delle risorse di rete all'interno del sistema, consentendo una visione chiara delle relazioni tra i componenti e dei flussi di dati.

Come illustrato sul diagramma, la piattaforma di containerizzazione è Docker che al suo interno possiede due container: Il primo contenente il Server MySQL, il secondo con l'immagine dell'applicazione web.

I Container lavorano su un sistema operativo Windows 11 e comunicano con il client per mezzo del porto 80 attraverso il protocollo HTTP. Il client si collega su tale porto attraverso il browser Web a disposizione delle macchine UniNA utilizzate.



17.TESTING

Il testing è un processo essenziale nello sviluppo del software che mira a valutare la qualità, l'affidabilità e il corretto funzionamento di un sistema o di un'applicazione. Durante il testing, vengono eseguite una serie di attività per identificare difetti, errori e problemi nel software. Ciò viene fatto attraverso l'esecuzione di input specifici, la comparazione dei risultati attesi con quelli effettivi e l'analisi del comportamento del sistema in diversi scenari. I test possono essere automatizzati o eseguiti manualmente e includono test di unità, test di integrazione, test funzionali, test di regressione e molti altri. L'obiettivo finale del testing è garantire che il software soddisfi i requisiti stabiliti e funzioni in modo affidabile per gli utenti finali.

Per il testing dell'interfaccia, vanno eseguiti test funzionali per verificare che tutte le funzionalità della GUI

siano implementate correttamente e che rispondano in modo adeguato agli input.

Le classi che popolano il database sono state prelevate dal repository del Team Software Architecture Design. [Path: Documenti/General/EsempiDiClassiDaTestare/generated_tests_2016.zip]

- [1] http://localhost:8080/Admin_ManageClass
- [2] http://localhost:8080/Admin_UploadClass
- [3] http://localhost:8080/Admin_AdvancedSearchClasses
- [4] http://localhost:8080/User_ViewClass
- [5] http://localhost:8080/User_SearchClasses
- [6] http://localhost:8080/Admin_ViewClasses
- [7] http://localhost:8080/Admin_UpdateClasses/1
- [8] http://localhost:8080/User_Play

Test Case Id	Descrizione	Precondizioni	Input	Output	Post condizioni	Esito
1 CM	L'amministratore visualizza l'elenco delle classi disponibili	C'è almeno una classe nel Database	L'amministratore e accede al link [1]	Restituzione a video Elenco Classi	L'amministratore riesce a visualizzare le classi disponibili	PASS
2 CM	L'amministratore visualizza il Database vuoto	Non ci sono classi nel Database	L'amministratore e accede al link [1]	Restituzione a video di Elenco Vuoto	/	PASS
1 D	L'amministratore fa il Download di una delle classi disponibili	C'è almeno una classe nel Database	L'amministratore e nella pagina [1]/[3]/[6] preme sul tasto Download	File java scaricato nel Download del browser web	L'amministratore riesce ad accedere al download del file	PASS
1 U	Si modificano i parametri di una classe disponibile	C'è almeno una classe nel Database	L'amministratore e nella pagina [1] preme sull'icona della matita blu L'amministratore e accede alla pagina [7] e riconfigura i parametri della classe selezionata Preme poi sul pulsante Submit	Viene mostrato un Popup con su scritto "Class Update Successfully"	La classe è modificata	PASS
1 DEL	Si vuole eliminare una classe tra quelle disponibili	C'è almeno una classe nel Database	L'amministratore e nella pagina [1] preme sull'icona del cestino rosso	Viene mostrato un messaggio con su scritto "La cancellazione è andata a buon fine"	La classe viene rimossa dal database. Viene mostrato un elenco delle classi rimanenti.	PASS
1 ADD	Si vuole aggiungere una nuova classe	Si accede al link [2] cliccando sulla sezione	L'amministratore e accede alla pagina [2] e aggiunge: ➤ LOC ➤ Comple	Viene mostrato un messaggio "File uploaded Successfully"	La classe è presente fra quelle disponibili nel database	PASS

Test Case Id	Descrizione	Precondizioni	Input	Output	Post condizioni	Esito
		"Add a new Class"	<ul style="list-style-type: none"> ➤ xity ➤ Last Update ➤ Opponent ➤ Recommended ➤ File 			
2 ADD	Si sbaglia a inserire LOC: float, null, nonvalid	Si accede al link [2] cliccando sulla sezione "Upload Class"	L'amministratore accede alla pagina [2] e aggiunge i LOC della classe di tipo float, null o nonvalid	Viene mostrato un messaggio "Insert LOC greater than 0"	La classe non viene aggiunta al Database	PASS
3 ADD	Si sbaglia a inserire Last Update: float, null, nonvalid	Si accede al link [2] cliccando sulla sezione "Upload Class"	L'amministratore accede alla pagina [2] e aggiunge Last Update della classe di tipo float, null o nonvalid	Viene mostrato un messaggio "Insert a valid Last Update"	La classe non viene aggiunta al Database	PASS
4 ADD	Si sbaglia ad inserire il file Java: non si inserisce	Si accede al link [2] cliccando sulla sezione "Upload Class"	L'amministratore accede alla pagina [2] e non carica un file	Viene mostrato un messaggio "File is not uploaded. Error occurred"	La classe non viene aggiunta al Database	PASS
5 ADD	Si sbaglia ad inserire il file Java: si inserisce un file duplicato	Si accede al link [2] cliccando sulla sezione "Upload Class" Il file da inserire è già presente nel database con	L'amministratore accede alla pagina [2] e carica il file	Viene mostrato un messaggio "Url already Exists"	La classe non viene aggiunta al Database	PASS

Test Case Id	Descrizione	Precondizioni	Input	Output	Post condizioni	Esito
		lo stesso nome				
1 SE	Si ricerca attraverso appositi filtri, una classe specifica fra quelle disponibili	Si accede al link [3] cliccando sulla sezione "Search Class" La classe esiste fra quelle disponibili nel database	L'amministratore inserisce i features della classe da cercare	Restituzione a video della classe specificata	La classe può essere scaricata, eliminata o modificata	PASS
2 SE	Si ricerca attraverso appositi filtri, una classe non presente fra quelle disponibili	Si accede al link [3] cliccando sulla sezione "Search Class" La classe non esiste fra quelle disponibili nel database	L'amministratore inserisce i features della classe da cercare	Restituzione a video dell'elenco di classi vuoto	/	PASS
3 SE	Si ricerca attraverso appositi filtri, una serie di classi fra quelle disponibili con parametri maggiori o uguali di quelli specificati	Si accede al link [3] cliccando sulla sezione "Search Class" Esiste almeno una classe fra quelle disponibili con parametri maggiori o uguali di	L'amministratore inserisce i features della classe da cercare	Restituzione a video delle classi che rispettano i vincoli	Le classi possono essere scaricate, eliminate o modificate	PASS

Test Case Id	Descrizione	Precondizioni	Input	Output	Post condizioni	Esito
		quelli specificati				
1 V	L'utente visualizza l'elenco delle classi disponibili	C'è almeno una classe nel Database	L'utente accede al link [4]	Restituzione a video elenco classi	L'utente riesce a visualizzare le classi disponibili	PASS
2 V	L'utente non visualizza l'elenco delle classi disponibili	Non c'è alcuna classe nel Database	L'utente accede al link [4]	Restituzione a video elenco classi vuoto	L'utente non riesce a visualizzare le classi disponibili	PASS
1 D	L'utente fa il Download di una delle classi disponibili	C'è almeno una classe nel Database	L'utente nella pagina [4]/[5] preme sul tasto Download	File Java scaricato nei Download del Browser Web	L'utente riesce ad accedere al download del file	PASS
1 P	L'utente seleziona la classe da testare e il robot tramite due menu, salvando le sue scelte cliccando sul pulsante Play	C'è almeno una classe da testare e un robot nel Database	L'utente nella pagina [8] inserisce la classe da testare e il robot avversario, poi clicca sul pulsante Play	Si avvia l'editor per testare la classe	L'utente riesce a giocare	PASS