UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

CORSO DI SYSTEM ARCHITECTURE DESIGN

PROF.SSA FASOLINO ANNA RITA

A.A. 2022-23

PROGETTO ENACTEST

RELAZIONE GRUPPO #3

TASK DI RIFERIMENTO: T1

Studenti:

Costabile Luigia	M63001530	luig.costabile@studenti.unina.it
Gallucci Ciro	M63001446	ci.gallucci@studenti.unina.it
Negri Federico	M63001486	fe.negri@studenti.unina.it
Tornincasa Luca	M63001470	lu.tornincasa@studenti.unina.it

Sommario

Sommario

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 #3, composto dagli studenti Costabile Luigia, Gallucci Ciro, Negri Federico e Tornincasa Luca, frequentanti il primo anno magistrale di Ingegneria Informatica presso l'Università di Napoli Federico II, è relativo ai *requisiti sul mantenimento di un insieme di classi da testare* (T1) e recita come segue:

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.

1.3 Metodologia di lavoro

A partire dal requisito fornito, si è adoperato un approccio basato sullo Sviluppo Agile, il quale ha portato alla realizzazione del software in oggetto.
In particolare, ci si è basati su Scrum, framework (architettura logica di supporto sulla quale progettare e realizzare un software) Agile per la gestione del ciclo di sviluppo software iterativo ed incrementale, delineato da Schwaber e Sutherland nel 1995.
Secondo questi ultimi, i pilastri che devono sostenere ogni implementazione di un software sono trasparenza, ispezione e adattamento:

- *trasparenza* – rendere gli obiettivi chiari a tutti, rendere visibile l'informazione e accessibile all'intero team, adottando anche lavagne per delineare lo stato di avanzamento del progetto;
- *ispezione* – verifiche continue (ogni giorno, a fine dell'iterazione con feedback del cliente e analisi delle performance per individuare dove il team ha performato bene e dove un po' meno);
- *adattamento* – capacità continua del team di essere flessibile ai cambiamenti in corso d'opera.

Sono state seguite le fasi principali di Scrum:

- *Sprint Planning*: all'inizio di ogni Sprint il team si riunisce per pianificare il lavoro da svolgere. Durante il processo di Sprint Planning, il team seleziona un insieme di elementi prioritari dal Product Backlog che devono essere implementati.
- *Daily Scrum*: evento time-boxes dalla durata di 15 minuti, effettuato ogni giorno alla stessa ora e nello stesso luogo. Consente di ottimizzare la collaborazione tra i membri del team e massimizzare le prestazioni.
- *Sprint*: durante lo Sprint, il team lavora per implementare le funzionalità selezionate dal Product Backlog. Durante lo Sprint, il lavoro viene suddiviso in compiti gestiti dal team stesso.
- *Sprint Review*: alla fine di ogni Sprint, si tiene una Sprint Review per esaminare il lavoro completato e ottenere feedback del committente. Durante questa fase, vengono presentate le funzionalità implementate e il feedback viene utilizzato per migliorare il prodotto e aggiornare il Product Backlog.
- *Sprint Retrospective*: dopo la Sprint Review, il team di si riunisce per una Sprint Retrospective. Durante questa fase, il team riflette sulle prestazioni passate, identifica punti di forza e debolezza e propone miglioramenti per i futuri Sprint.

Gli artefatti prodotti sono:

- *Product Backlog*: è un elenco priorizzato di funzionalità, requisiti, correzioni di bug e miglioramenti che devono essere implementati nel prodotto finale.
- *Sprint Backlog*: è il piano dettagliato di sviluppo per ogni Spring ed è formato da un insieme di item (compiti da portare a termine per la realizzazione del prodotto).

In quanto framework di management, non vi sono ivi indicazioni su come lavorare nell'Ingegneria del Software. In tale ottica, si sono comunque scelte pratiche proprie dell'Agile, tra cui il pair programming.
Si sono considerati Sprint dalla durata di due settimane, durante le quali il gruppo ha esplorato ogni singolo requisito e realizzato quanto prefissato. Ci sono stati in totale quattro Sprint, di cui i primi tre durante l'erogazione del corso in oggetto.
Inoltre, al fine di tener traccia dei progressi parziali ottenuti giorno dopo giorno, si è redatto un diario di bordo, allegato nella pagina seguente.

1.3.1 Diario di Bordo

DIARIO DI BORDO - GRUPPO #3
COSTABILE LUIGIA, GALLUCCI CIRO, NEGRI FEDERICO, TORNINCASA LUCA

PRIMA ITERAZIONE

MAR 11/04/23	Effort: 1h	Analisi del requisito specifico assegnato e suddivisione in storie utente.
MER 12/04/23	Effort: 2h	Definizione del Product Backlog: aggiunta di descrizioni, tipologia e difficoltà per ciascuna storia utente.
GIO 13/04/23	Effort: 2h	Diagramma dei casi d'uso e descrizione informale degli scenari dei casi d'uso.
VEN 14/04/23	Effort: 2h	Glossario dei termini, tabella attori-obiettivi, diagramma delle classi.
SAB 15/04/23	Effort: 3h	Diagramma di contesto, primi due capitoli della relazione e ridefinizione del diario di bordo.
LUN 17/04/23	Effort: 2h	Mockup per il prospetto di realizzazione del task T1 al termine dello svolgimento del processo di sviluppo.

1

DIARIO DI BORDO - GRUPPO #3
COSTABILE LUIGIA, GALLUCCI CIRO, NEGRI FEDERICO, TORNINCASA LUCA

SECONDA ITERAZIONE

VEN 28/04/23	Effort: 2h	Progettazione dello Sprint Backlog per l'iterazione corrente. Analisi delle tecnologie necessarie per lo sviluppo del task.
SAB 29/04/23	Effort: 1h	Diagramma E-R per la definizione delle entità e delle relazioni presenti nel database del task T1.
LUN 01/05/23	Effort: 1,5h	Bozza provvisoria del diagramma MVC per la definizione dei package necessari alla stesura del codice per il mantenimento della repository.
MAR 02/05/23	Effort: 2,5h	Stesura del codice per il mantenimento degli attributi associati alle classi Java presenti nel database.
MER 03/05/23	Effort: 3h	Stesura del codice per il mantenimento delle classi Java presenti all'interno del database.
SAB 06/05/23	Effort: 2h	Testing e scelte progettuali per la realizzazione del controller nelle prossime iterazioni.
DOM 07/05/23	Effort: 2h	Stesura relazione circa l'iterazione appena conclusa, revisione del lavoro e ridefinizione del diario di bordo.

2

DIARIO DI BORDO - GRUPPO #3
COSTABILE LUIGIA, GALLUCCI CIRO, NEGRI FEDERICO, TORNINCASA LUCA

TERZA ITERAZIONE

LUN 15/05/23	Effort: 2h	Conclusione del codice relativo alla storia utente di mantenimento dei file nella repository. Revisione e aggiustamenti.
MAR 16/05/23	Effort: 2h	Analisi preliminare della storia utente relativa alla ricerca e stesura delle prime firme con relativo codice.
MER 17/05/23	Effort: 2,5h	Implementazione del codice per la ricerca applicata con utilizzo di filtri dinamicamente aggiunti.
GIO 18/05/23	Effort: 2 h	Refactoring delle API redatte per il backend del progetto e dei package del progetto.
SAB 20/05/23	Effort: 2h	Testing di tutte le API relativamente al backend del progetto fino a questo momento implementate.
DOM 21/05/23	Effort: 2h	Definizione del diagramma componenti e connettori e del diagramma MVC del sistema.
LUN 22/05/23	Effort: 1h	Stesura della relazione e della presentazione relativa alla terza iterazione.

3

DIARIO DI BORDO - GRUPPO #3
COSTABILE LUIGIA, GALLUCCI CIRO, NEGRI FEDERICO, TORNINCASA LUCA

ULTIMA ITERAZIONE

VEN 30/06/23	Effort: 2h	Documentazione approfondita sull'Analisi dei requisiti e sulle metodologie di lavoro effettutate.
SAB 01/07/23	Effort: 4h	Documentazione approfondita sul sistema, il suo contesto, i componenti, il deployment e le tecniche di sviluppo.
DOM 02/07/23	Effort: 4h	Realizzazione dei diagrammi di sequenza, diagramma di attività, diagramma di stato e documentazione sulle API effettuate.
LUN 03/07/23	Effort: 3h	Realizzazione del testing con relativa tabella di testing
MAR 04/07/23	Effort: 2h	Costruzione dell'immagine finale con operazione di "dockeraggio"
MER 05/07/23	Effort: 3h	Stesura della relazione finale.
GIO 06/07/23	Effort: 1h	Accorgimenti "Last Minute" & Stesura della relazione finale.

4

2. SPECIFICA DEI REQUISITI

2.1 Attori, funzionalità e attributi

Per prima cosa, risulta fondamentale analizzare attentamente il testo del requisito al fine di individuare i relativi attori, le funzionalità e gli attributi.

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 eseSmpio la complessità della classe, o altri attributi.

- Attori
- Funzionalità
- Attributi

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 `.java` che si vuole caricare nel database, contente la classe da testare	Specifiche file
File Java	File con estensione `.java` 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	Line di Codice

Complexity	Complessità del file Java. Le possibilità previste sono: - Simple - Moderate - Complex - Insane	Complessità nel testare la Classe
Recommended Opponent	Avversario consigliato. Le possibilità previste sono: - RANDOOP - EVOSUITE	Oppositore, 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
Amministratore	Utente che può aggiornare l'insieme delle classi disponibili mediante l'aggiunta di classi e relativo salvataggio del file di codice	Gestore
Giocatore	Utente che può visualizzare l'elenco delle classi e farne il relativo download	Competitore, tester
Repository	Archivio digitale di informazioni centralizzato	Archivio, database

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, una storia utente 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 3.1] vengono mostrate le cinque storie utente alla base del task.

Figura 3.1 Storie utente.

Inoltre, in [Figura 3.2], viene rappresentato uno schizzo di Product Backlog realizzato con FlyingDonut.

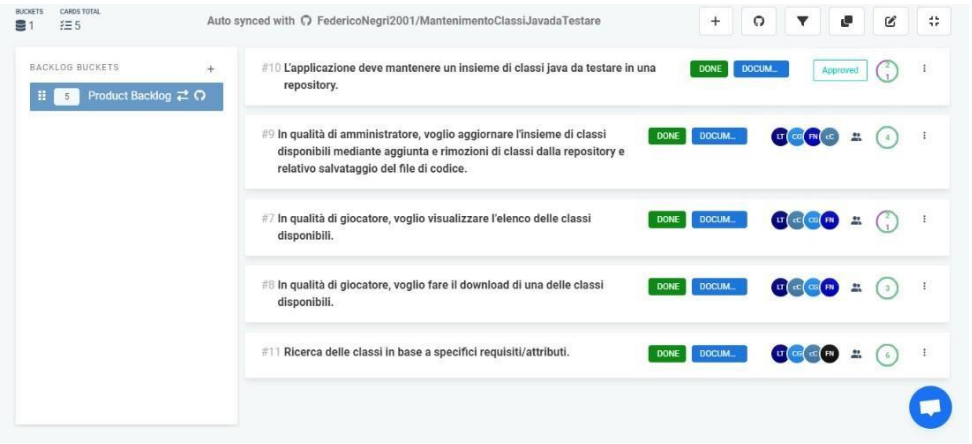


Figura 3.2 Product Backlog.

Nelle figure che seguono, [Figure 3.3 a 3.7], si riportano screenshots delle singole storie nel Backlog, ciascuna corredata da:

- **titolo** – un breve e descrittivo titolo che riassume il bisogno dell'utente o la funzionalità richiesta;
- **descrizione** – una descrizione più dettagliata della storia utente che spiega il contesto, il problema o la necessità dell'utente;
- **priorità** (data dall'ordine – che sarà di sviluppo – in [Figura 3.2]) – la storia utente viene generalmente assegnata a una determinata priorità nel Backlog, che indica l'importanza relativa rispetto ad altre storie utente;
- **dimensione** o **stima** -- la dimensione rappresenta una valutazione approssimativa del tempo o dell'effort necessario per completare la storia utente (*estimation* nelle figure).

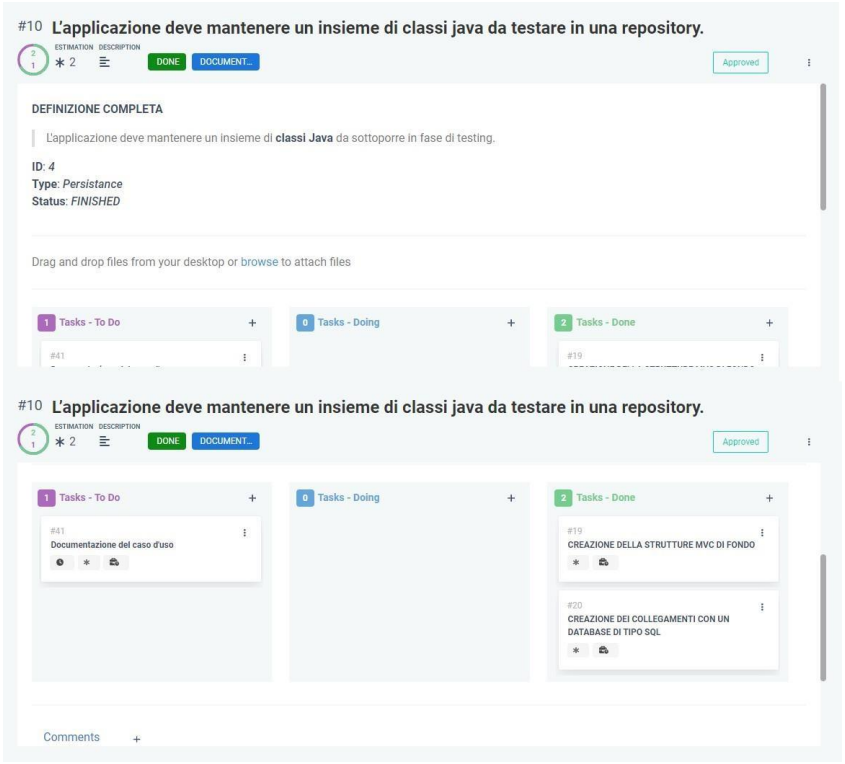


Figura 3.3 Storia Utente 1.

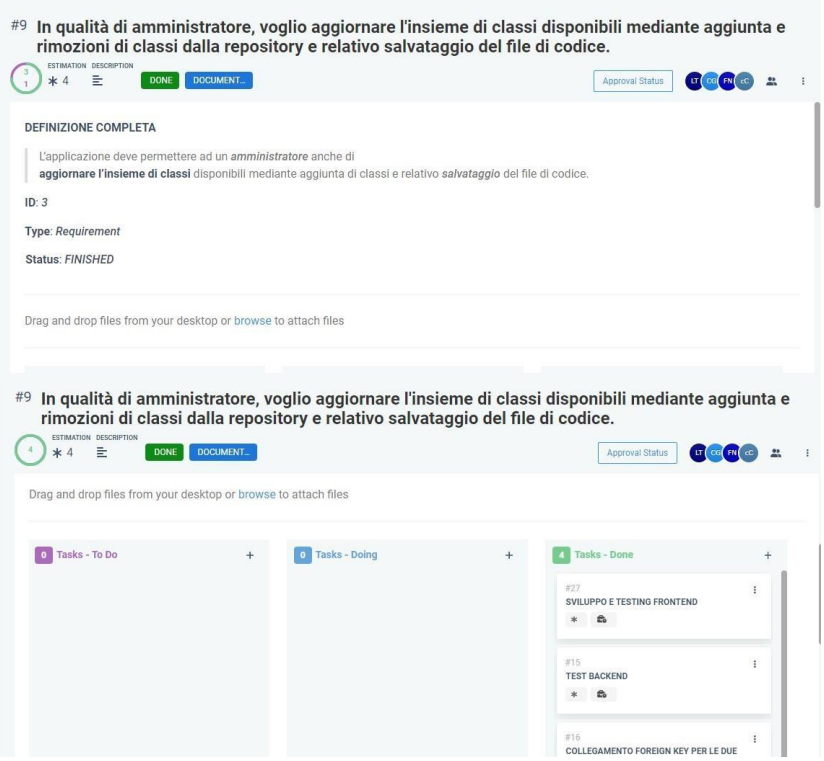


Figura 3.4 Storia Utente 2.

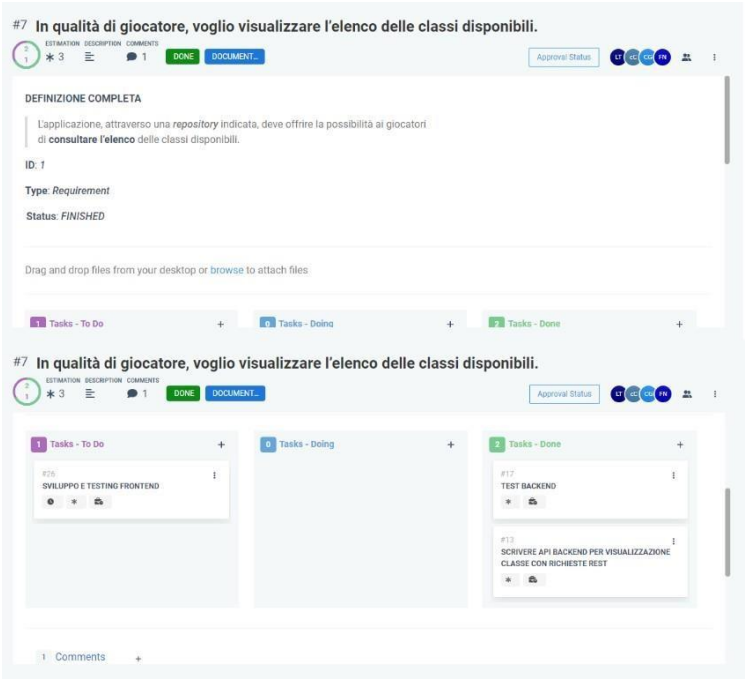


Figura 3.5 Storia Utente 3.

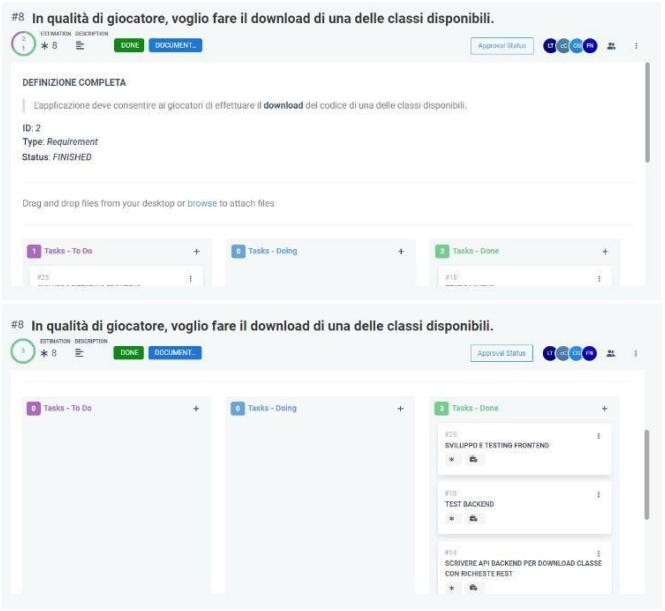


Figura 3.6 Storia Utente 4.

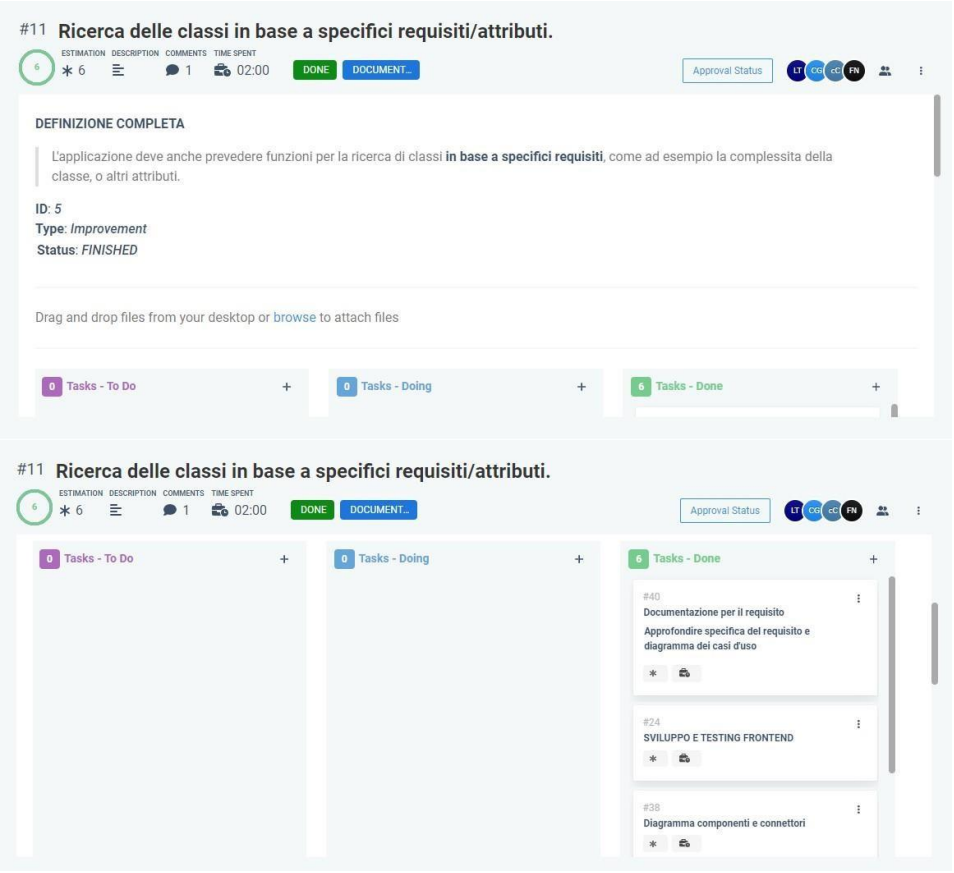


Figura 3.7 Storia Utente 5.

3.2 Criteri di Accettazione

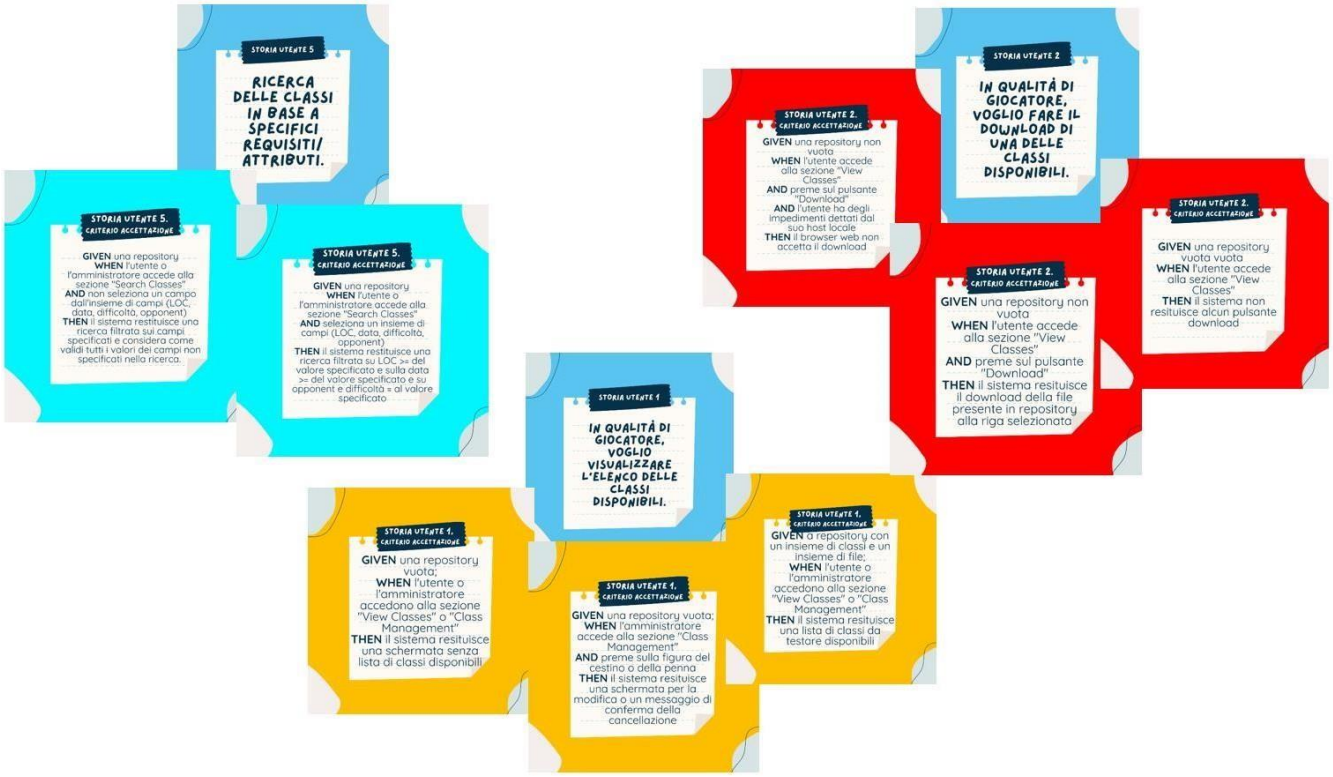
Un **criterio di accettazione** è una descrizione dettagliata delle condizioni che devono essere soddisfatte affinché una user story sia considerata completata e accettata. Questi criteri definiscono le aspettative precise dell'utente o del cliente rispetto alla funzionalità o al requisito specifico.

I criteri di accettazione sono scritti in modo **chiaro e misurabile**, fornendo una guida obiettiva per il team di sviluppo. Essi possono includere requisiti funzionali, prestazionali, di sicurezza, di usabilità o di compatibilità con altre componenti del sistema.

Dovrebbero inoltre essere definiti in collaborazione tra il cliente o il rappresentante degli utenti e il team di sviluppo, in modo da garantire una comprensione comune delle aspettative; vengono utilizzati come base per la **pianificazione**, l'esecuzione dei **test** e la valutazione della **conformità** del prodotto finale.

I criteri di accettazione possono anche essere utilizzati come base per l'automazione dei test, consentendo una valutazione rapida ed efficiente del prodotto.

Per ciascuna storia utente nell'ordine definito dal Product Backlog in [Figura 3.2], si riportano i relativi criteri di accettazione.



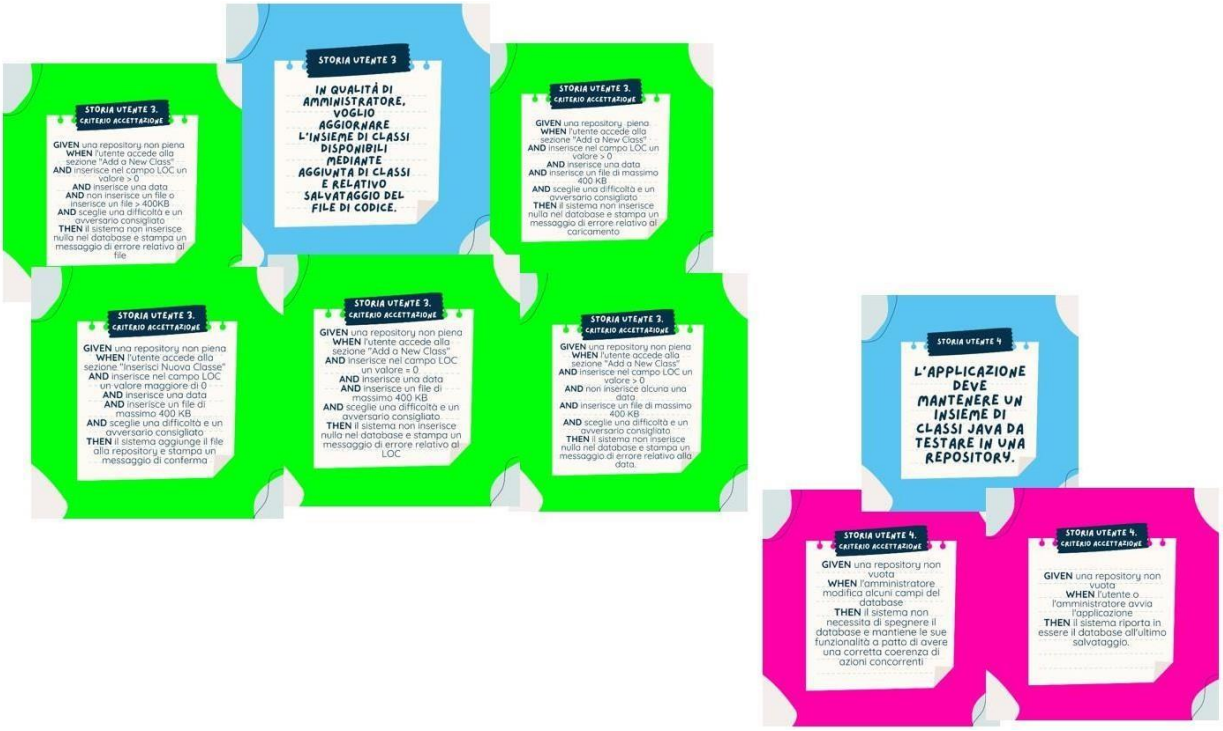


Figura 3.8 Criteri di accettazione.

4. ANALISI DEI REQUISITI

In tale capitolo vengono anzitutto descritti i requisiti al fine di classificarli in funzionali e non, nonché in requisiti sui dati.

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**: utile per la 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.
- 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.

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` che contiene 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 la 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.

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 e tipo di file
- **RD-2**. Per ogni classe di testing si deve specificare: complessità, avversario consigliato, data dell'ultima modifica degli attributi e numero di linee di codice.
- **RD-3.** Ad ogni classe di testing deve essere associato il corrispondente file `.java`.

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.
  - Deve verificare l'integrità degli input prima di effettuare inserimento nel database
- **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

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.

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*: rappresenta i dati dell'applicazione e la logica di gestione dati ad essi associata. Nel contesto di Spring Boot, il model può essere implementato rappresentando le classi di dominio dell'applicazione con le annotazioni `@Entity` per indicare la mappatura delle classi Java con le tabelle del database. Mentre si occupano delle operazioni sui dati i metodi contenuti nei package `repository` per la logica di data management.
  - *View*: si occupa della visualizzazione dei dati all'utente finale. Nel contesto di Spring Boot, la view può essere implementata nel package `view`, in combinazione con i template engine come Thymeleaf per generare le pagine web.
  - *Controller*: gestisce le richieste dell'utente e coordina le interazioni tra il model e la view. Nel contesto di Spring Boot, i controller sono implementati nei package `controller` e `service` e si occupano di accogliere le richieste HTTP, elaborare i dati, invocare i servizi appropriati e restituire le risposte corrispondenti.

  L'utilizzo di questo pattern architetturale aiuta a separare le responsabilità all'interno di un'applicazione, rendendola più **modulare**, **manutenibile** e **testabile**.

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 i seguenti:

- *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 (Representational State Transfer) 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. Quindi è lo strumento che connette il front-end della applicazione, ed 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. Consentono di realizzare il **principio di separazione tra client e server** che permette una maggiore modularità e manutenibilità dell'architettura.

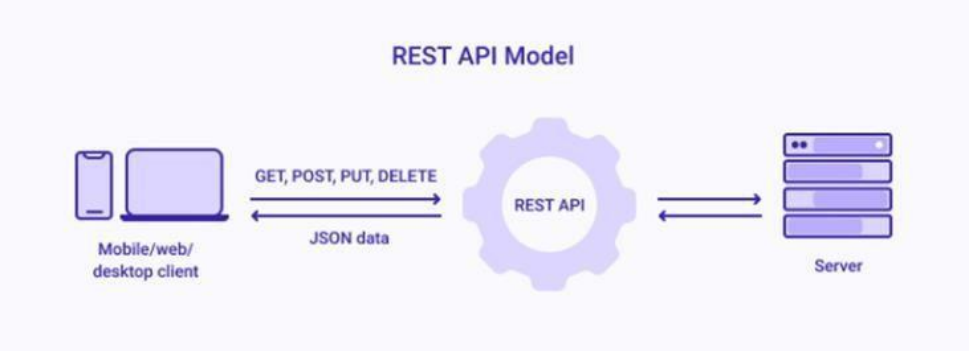


Figura 5.1 REST API Diagram

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, AJAX, JAVASCRIPT**

Figura 6.1 HTML Logo

HTML, AJAX e JavaScript sono fondamentali per lo sviluppo del frontend di un'applicazione web. HTML (HyperText Markup Language) viene utilizzato per creare la struttura e il layout della pagina, definendo gli elementi come titoli, paragrafi, tabelle e form. JavaScript, invece, permette di rendere la pagina interattiva, manipolando gli elementi HTML, gestendo gli eventi e implementando logiche di business. AJAX (Asynchronous JavaScript and XML), infine, consente di aggiornare il contenuto della pagina in modo asincrono senza doverla ricaricare completamente, migliorando l'esperienza utente. Grazie a queste tecnologie, è possibile creare un frontend dinamico e reattivo, in grado di interagire con il server in modo fluido e fornire agli utenti un'esperienza web più intuitiva e coinvolgente.

- **THYMELEAF**

Figura 6.2 Thymeleaf Logo

Thymeleaf è un motore di template per Java che consente di integrare la logica del backend con il frontend 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 frontend. Inoltre, Thymeleaf offre un'ottima integrazione con framework come Spring, rendendolo una scelta popolare per lo sviluppo web basato su Java.

• **MYSQL**

Figura 6.3 MySQL Logo

MySQL è un sistema di gestione di database relazionali ampiamente utilizzato nel backend delle applicazioni web. Per utilizzare MySQL nel backend, 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 backend delle applicazioni web.

• **XAMPP**

Figura 6.4 XAMPP Logo

XAMPP è una distribuzione di software open-source che semplifica l'installazione e la configurazione di un ambiente di sviluppo per il backend. 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 backend 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 multi-piattaforma e supporta sistemi operativi come Windows, macOS e Linux, rendendolo uno strumento versatile per lo sviluppo back-end su macchine diverse.

• **POSTMAN**

Figura 6.5 Postman Logo

Postman è uno strumento di sviluppo API che consente di testare, documentare e collaborare con le API nel processo di sviluppo backend. Utilizzando Postman, è possibile inviare richieste http personalizzate agli endpoint delle API, visualizzarne le risposte e analizzare i dati restituiti. Inoltre, Postman 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 backend.

• **DOCKER**

Figura 6.6 Docker Logo

Docker è una piattaforma di containerizzazione che consente di creare e gestire facilmente ambienti di sviluppo e produzione per il backend delle applicazioni. Utilizzando Docker per il backend, 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 i 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**

Figura 6.7 Hibernate Logo

Hibernate è un framework di Object-Relational Mapping (ORM) che semplifica l'interazione con il database nel backend 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**

Figura 6.8 Maven Logo

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**

Figura 6.9 Visual Paradigm Logo

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.

- **FLYINGDONUT**

Figura 6.10 FlyingDonut

FlyingDonut è un sito web molto utile per la gestione di progetti che si basano su collaborazione e pianificazione flessibile, come il framework agile Scrum. In particolare, lavorando sul concetto di sprint, consente di descrivere Product Backlog in modo semplice e pratico, accompagnando il team di sviluppo in tutte le fasi del progetto agile, tenendo traccia dei requisiti e dello stato di avanzamento della relativa realizzazione.

7 MODELLO DEI CASI D'USO

7.1 Componenti principali

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

- *Attori primari*:
  - Amministratore;
  - Giocatore;
- *Attori secondari*:
  - Browser Web;
  - Repository.
- *Casi d'uso*:
  - **UC1**: VisualizzaElenco;
  - **UC2**: DownloadCodiceClasse;
  - **UC3**: ModificaClasse;
  - **UC4**: EliminaClasse;
  - **UC5**: GestioneClasse;
  - **UC6**: AggiuntaClasse;
  - **UC7**: RicercaAvanzata.

7.2 Diagramma dei Casi d'Uso

Si riporta in [Figura 7.1] il diagramma dei casi d'uso. Come ivi osservabile, vi sono quattro relazioni di `extends` tra i casi d'uso.

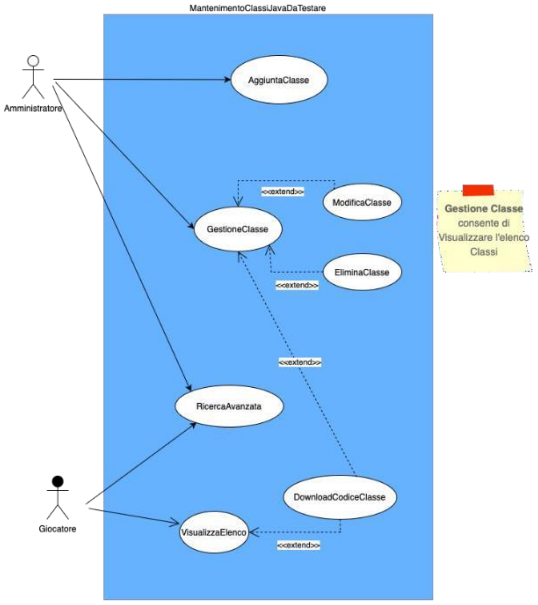


Figura 7.1 Diagramma dei casi d'uso.

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, pre- e post-condizioni, sequenze di eventi principali e alternative.

UC1: VisualizzaElenco

Caso d'uso:	**VisualizzaElenco**
Attore primario	Giocatore
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 "Visualizza Elenco" 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.

UC2: DownloadCodiceClasse

Caso d'uso:	**DownloadCodiceClasse**
Attore primario	Giocatore, Amministratore
Attore secondario	Browser Web, Repository
Descrizione	Il giocatore effettua il download di una delle classi Java disponibili
Pre-Condizioni	L'elenco delle classi da testare non è vuoto Il giocatore ha visualizzato l'elenco delle classi disponibili (VisualizzaElenco) L'amministratore ha visualizzato l'elenco delle classi disponibili (GestioneClasse)
Sequenza di eventi Principale	1. Il giocatore seleziona la classe Java da scaricare. 2. Il giocatore 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.

UC3: ModificaClasse

Caso d'uso:	Modifica Classe
Attore primario	Amministratore
Attore secondario	Browser Web, Repository
Descrizione	L'amministratore aggiorna gli attributi di una Classe già inserita L'amministratore deve aver selezionato la classe da modificare dopo aver visualizzato la lista (GestioneClasse)
Pre-Condizioni	La classe da modificare deve essere già presente
Sequenza di eventi principale	1. L'amministratore seleziona la classe Java da modificare 2. L'amministratore seleziona i campi: - Complexity - Opponent Recommended 3. L'amministratore inserisce i campi: - LOC (Lines of Code) - Last Update 4. L'amministratore conferma le modifiche
Sequenza di eventi alternativa	Se alcuni campi non vengono modificati, allora per quel campo verrà lasciato il valore precedente
Post-Condizioni	Il file .java viene correttamente scaricato, dal repository, nella cartella Download.

UC4: EliminaClasse

Caso d'uso:	EliminaClasse
Attore primario	Amministratore
Attore secondario	Browser Web, Repository
Descrizione	L'amministratore elimina una classe e il file associato
Pre-Condizioni	La classe da eliminare deve essere già presente. L'amministratore deve aver selezionato la classe da cancellare dopo aver visualizzato la lista (GestioneClasse)
Sequenza di eventi Principale	1. L'amministratore seleziona la classe Java da eliminare 2. L'amministratore avvia la cancellazione
Post-Condizioni	Il file .java selezionato viene correttamente eliminato dal database, e anche tutte le informazioni ad esso associate.

UC5: GestioneClasse

Caso d'uso:	Gestione Classi
Attore primario	Amministratore
Attore secondario	Browser Web, Repository
Descrizione	L'amministratore visualizza l'elenco delle classi disponibili, e può selezionarle per eseguire le operazioni di:

	- Download - Eliminazione - Update
Pre-Condizioni	L'elenco delle classi non deve essere vuoto
Sequenza di eventi principale	1. L'amministratore visualizza l'elenco delle classi 2. L'amministratore seleziona una classe, e decide se effettuare: o Cancellazione (EliminaClasse) o Update (ModificaClasse) o Download (DownloadCodiceClasse)
Sequenza di eventi principale	Se l'amministratore seleziona l'opzione Cancellazione: - Viene richiamato il metodo per l'Eliminazione della Classe Se l'amministratore seleziona l'opzione Update: - Viene richiamato il metodo per la Modifica della Classe Se l'amministratore seleziona l'opzione Download: - Viene richiamato il metodo per il Download delle Classe
Post-Condizioni	Il file .java selezionato viene correttamente eliminato dal database, e anche tutte le informazioni ad esso associate.

UC6: AggiuntaClasse

Caso d'uso:	**Aggiunta Classe**
Attore primario	Amministratore
Attore secondario	Browser Web, Repository
Descrizione	L'amministratore aggiorna l'elenco delle classi disponibili aggiungendo una nuova Classe Java
Pre-Condizioni	Il file da caricare non deve essere già presente nel database, e la dimensione deve essere minore uguale ai 450kB
Sequenza di eventi principale	1. L'amministratore sceglie l'opzione "Aggiungi una Nuova Classe". 2. L'amministratore seleziona i seguenti campi: – Complexity – Opponent Recomended 3. L'amministratore compila i seguenti campi: – Lines of Code – Last Update 4. L'amministratore sceglie il file da caricare, selezionandone il percorso 5. L'amministratore conferma l'operazione
Sequenza di eventi alternativa	3.1 L'amministratore conferma senza compilare il campo Lines of Code: - Viene mostrato a video un errore che segnala che il campo Line of Code non deve essere vuoto

	- Non consente la creazione della nuova classe 3.2 L'amministratore conferma senza compilare il campo Last Update: - Viene mostrato a video un errore che segnala che il campo Last Update non deve essere vuoto - Non consente la creazione della nuova classe 4.1 L'amministratore seleziona un file da caricare, che ha lo stesso nome di un file già contenuto nel database: - Viene mostrato a video un errore che segnala che il è gia esistente un file così nominato - Non consente la creazione della nuova classe Se la dimensione del file è superiore ai 450kB, compare un errore che indica che non è possibile caricare i file
Post-Condizioni	La classe è stata correttamente caricata nell'elenco, visualizzabile sull'interfaccia web dell'applicazione. Il file selezionato è correttamente caricato nel database, in corrispondenza della classe

UC7: RicercaAvanzata

Caso d'uso:	**RicercaAvanzata**
Attore primario	Utente
Attore secondario	Browser Web, Repository
Descrizione	L'utente ricerca delle Classi in base a specifici requisiti/attributi
Pre-Condizioni	L'elenco delle classi da testare non è vuoto
Sequenza di eventi principale	1. L'utente sceglie l'opzione "Ricerca Classe" sulla home page dell'applicazione. 2. L'utente seleziona il valore dei seguenti filtri: – Complexity – Opponent Recommended 3. Inserisce i seguenti campi: – Lines of Code – Last Update 4. L'interfaccia web mostra l'elenco delle classi che rispondono agli attributi specificati dall'utente.
Sequenza di eventi alternativa	Se l'utente non inserisce alcuni dei filtri, la lista restituita risponderà solo ai filtri specificati, mentre per i filtri non specificati sarà considerato qualsiasi valore
Post-Condizioni	Vengono mostrate nell'elenco solo le classi che rispettano gli attributi specificati.

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 per il sistema. In particolare, il System Domain Model rappresenta il **modello di dominio di un sistema software**.
Il System Domain Model rappresentato comprende tre entità principali: `Classe`, `File` e `Response`.

La relazione fra `Classe` e `File` è di tipo 1:1 ed è collegata tramite la chiave esterna `fileid` in `Classe` e il campo id presente in `File`.
Gli attributi di tipo enumerazione dell'entità `Classe` sono due: `Level` e `Opponent`. Entrambi si compongono di una stringa selezionabile fra quattro alternative per il tipo `Level` e due alternative per il tipo `Opponent`. L'entità `Response` invece non corrisponde a nessuna tabella del database me è un oggetto utilizzato dal software come tipo di ritorno di alcuni metodi.

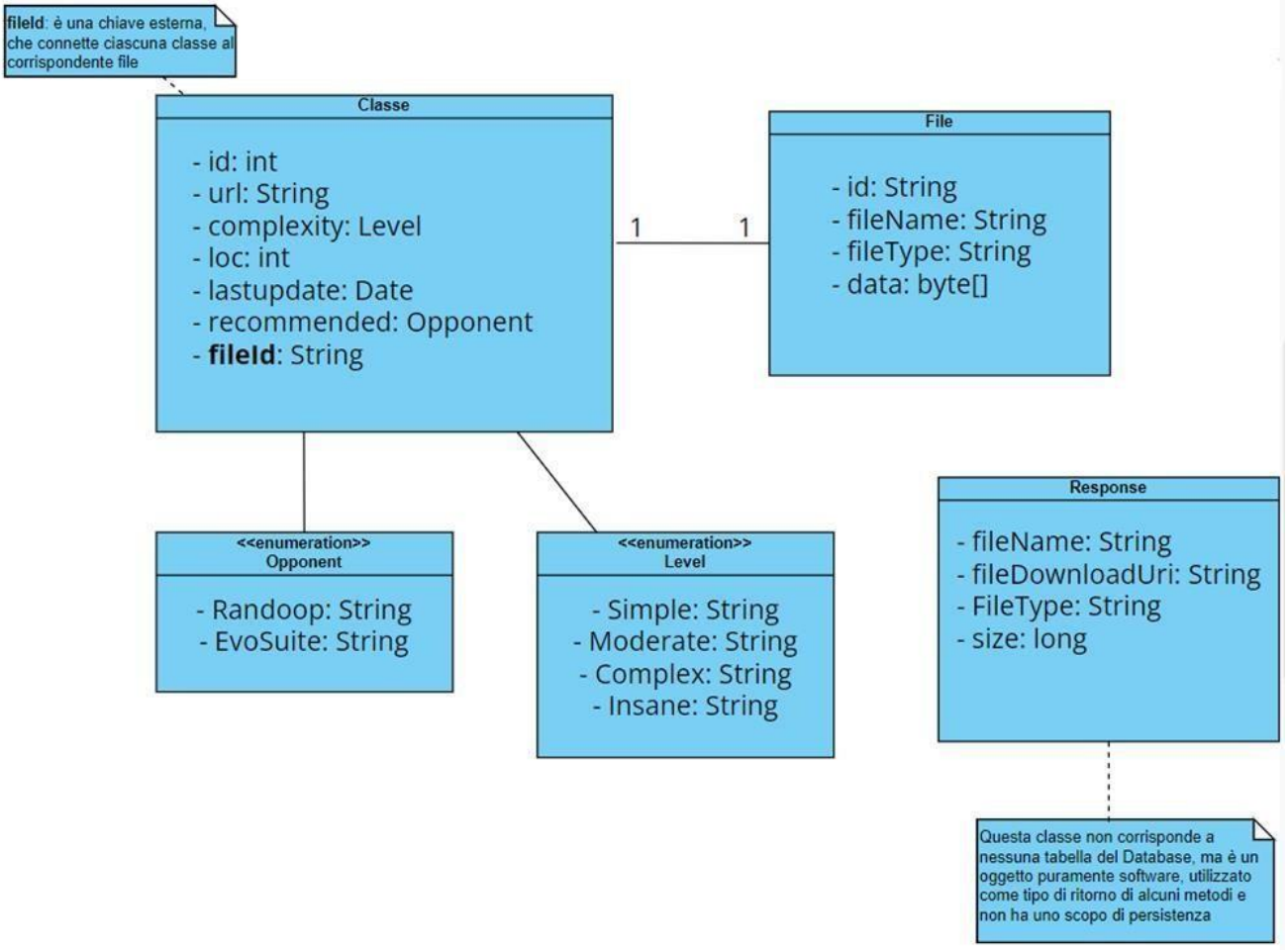


Figura 8.1 System Domain Model.

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 visuale e 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 diagramma di modellazione dei dati può essere utilizzato anche per definire la struttura di altri tipi di database come i database ad oggetti.

Il database risulta essere composto da due tabelle, ossia collezioni di dati organizzata in righe e colonne possedenti **una chiave primaria.** La chiave primaria è un attributo unico che identifica ogni riga in una tabella attraverso la dicitura `PK` .

La tabella `Classe` inoltre contiene al suo intero una **Foreign Key** indicata dall'indice `FK` collegato con una relazione 1:1 con la `PK` della tabella files.

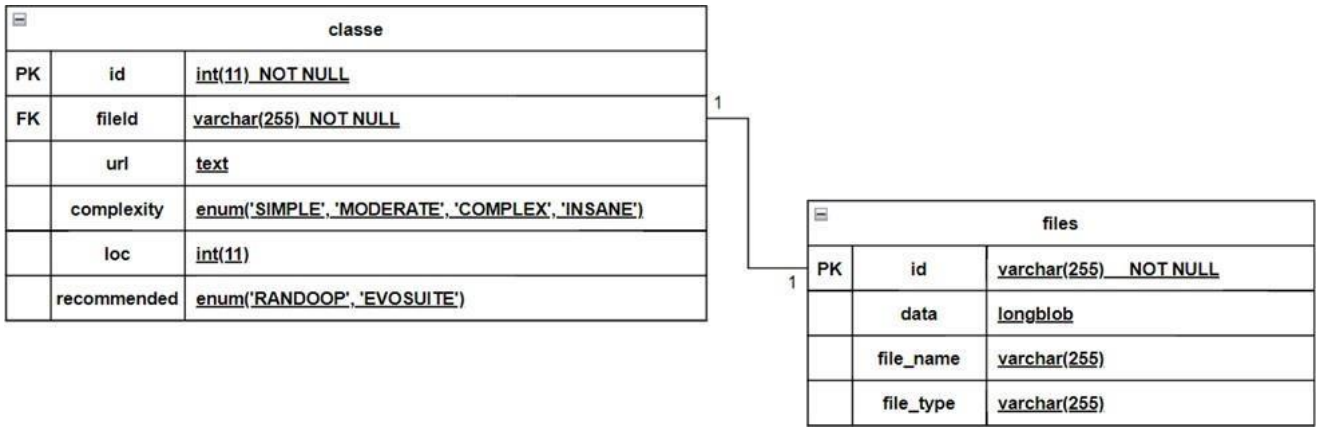


Figura 9.1 Diagramma Modellazione dei Dati.

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**.

Nel diagramma di contesto che segue si comprende come i due attori `Admin` e `User` si interfacciano col sistema attraverso il proprio browser web. Successivamente, scegliendo l'opportuna interfaccia, il sistema richiama il servizio di Gestione del database per l'operazione richiesta.

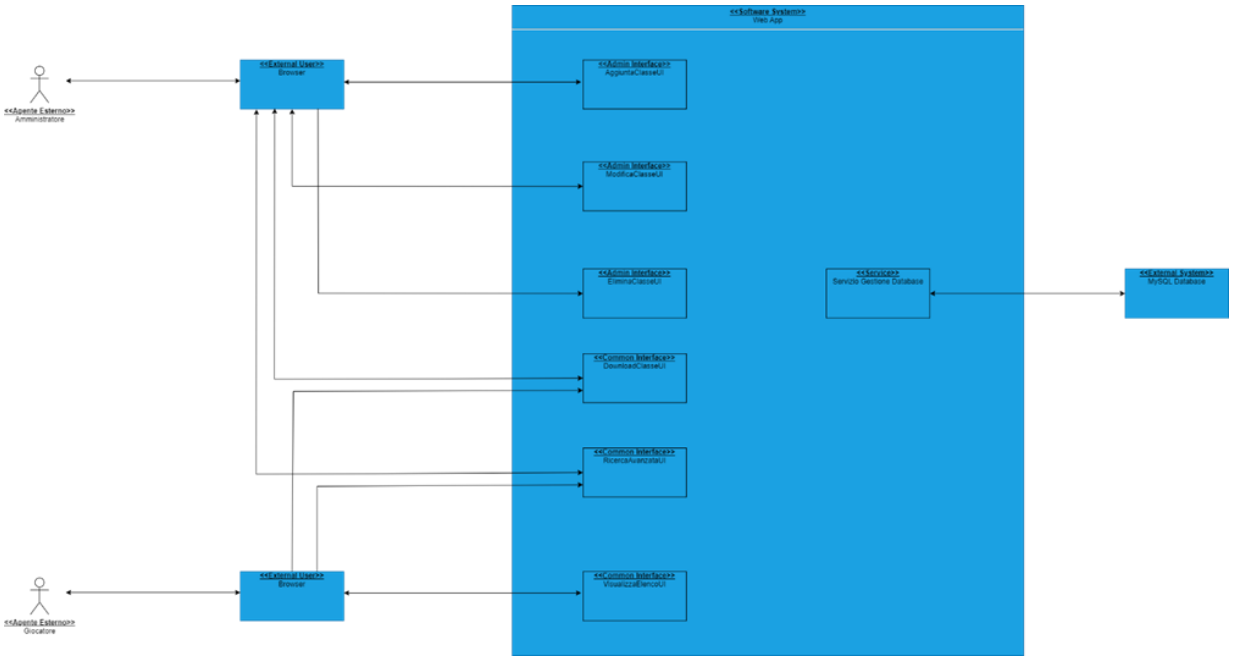


Figura 10.1 Context Diagram.

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.
Nel diagramma dei componenti sottostante si ha un componente "Front End View" che si interfaccia con il browser web per mezzo di **REST API**. Quando l'utente seleziona una delle possibili voci presenti sull'interfaccia web dell'applicazione (es. Search Classes, Add a new Class …) richiama tramite API un metodo della classe `front_end` del package `View`, la quale seleziona il corrispondente file `.html` dalla cartella `templates` e lo restituisce al browser. La classe `front_end` utilizza diversi componenti: `Admin Class Controller`, `UserClassController` e i templates per la gestione delle interfacce web. `AdminClassController` e `UserClassController` **richiedono le interfacce fornite** da `IDatabaseFileService` e `IClassService`, i quali vengono implementati rispettivamente da `DatabaseFileService` e `DBClasseService`.
`DatabaseFileService` e `DBClasseService` richiedono rispettivamente le interfacce offerte da `DBFileRepository` e `IclassRepository`, che comunicano col database attraverso la tecnica di interazione **ORM** ed estendono a loro volta `JpaRepository` e `CrudRepository`.

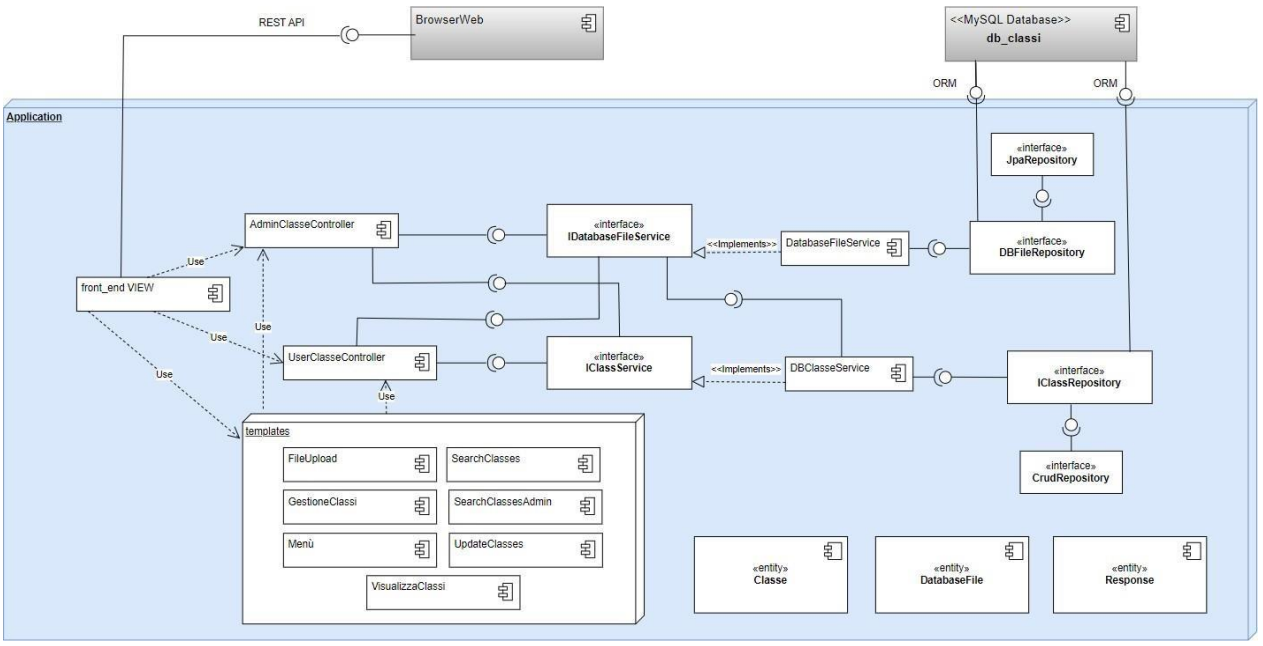


Figura 11.1 Component Diagram.

Di seguito inoltre vengono mostrati attraverso un'analisi approfondita i due elementi presenti nel diagramma dei componenti: "Browser" e "MySQL Database".

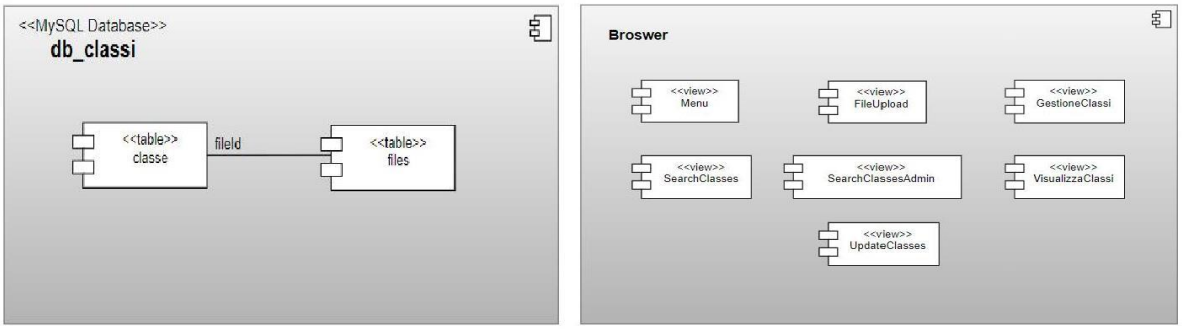


Figura 11.2 Component Diagram – Analisi approfondita.

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.

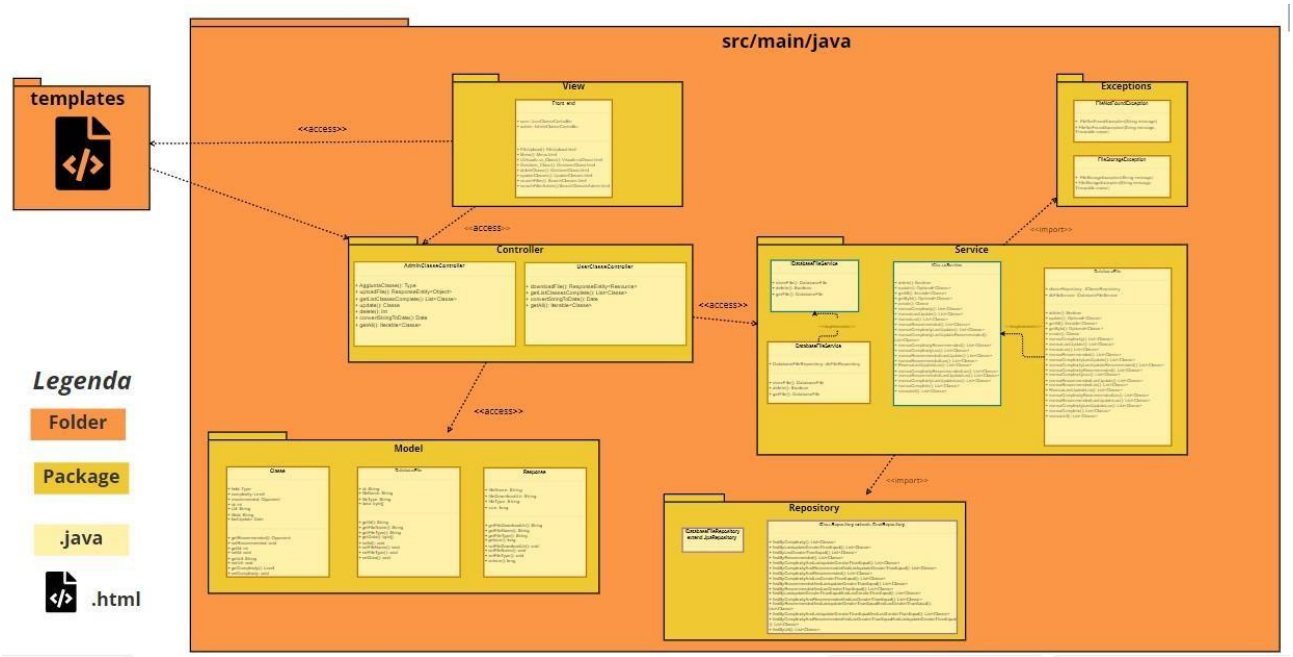


Figura 12.1 Package Diagram.

Di seguito si riporta una rappresentazione estesa del contenuto dei package `View` e `Controller` e della cartella contente 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 richiamto viene selezionato l'apposito file HTML.

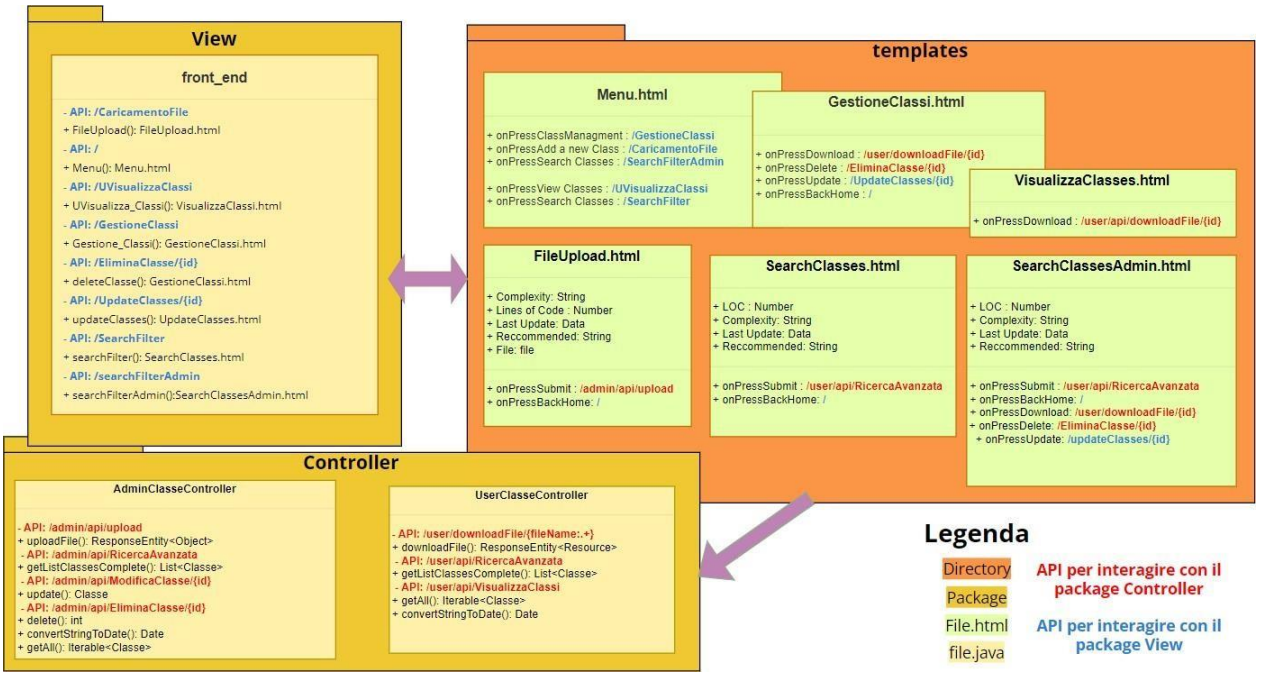


Figura 12.2 Package Diagram (front-end side).

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 surisorse 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 allocarele 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 APIper 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 terzeparti.
- **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 funzionalitàpossono 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 REST API in AdminClasseController

Di seguito si riporta il file `.json` contente la descrizione delle API utilizzate nel modulo software `AdminClasseController.java`, utilizzate per testare ed invocare le funzionalità riservare all'admin.

Figura 13.1 Descrizione API `AdminClasseController.java`.

• *Test delle API dell'AdminClasseController.*

- **/admin/api/upload**

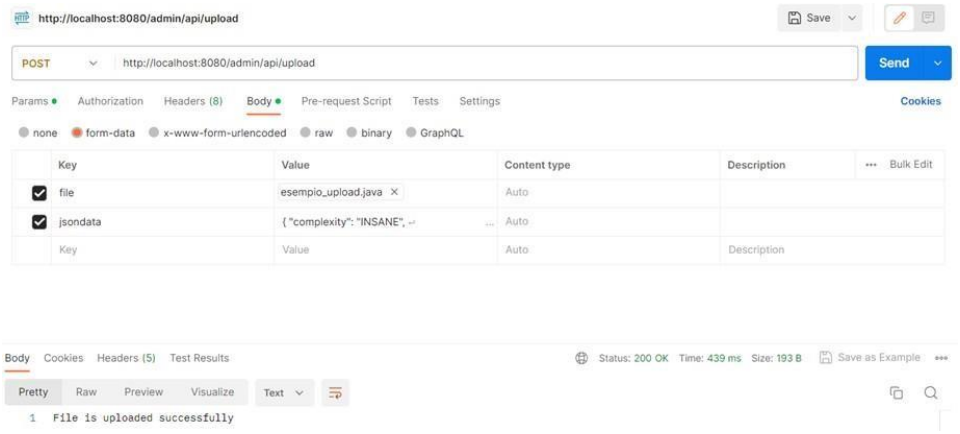


Figura 13.2 AdminClasseController – `/admin/api/upload`.

- **/admin/api/EliminaClasse/{id}**

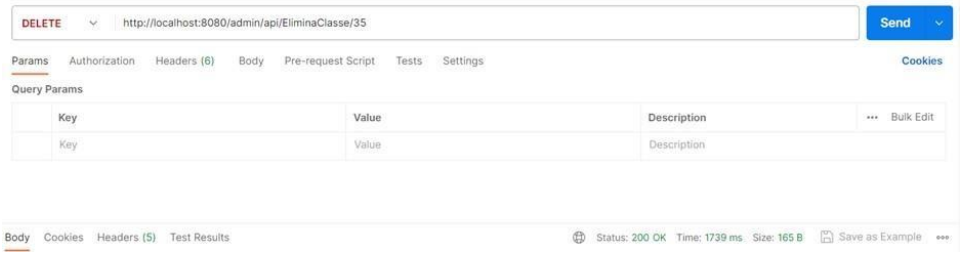


Figura 13.3 AdminClasseController – `/admin/api/EliminaClasse/{id}`.

- **/admin/api/ModificaClasse/{id}**

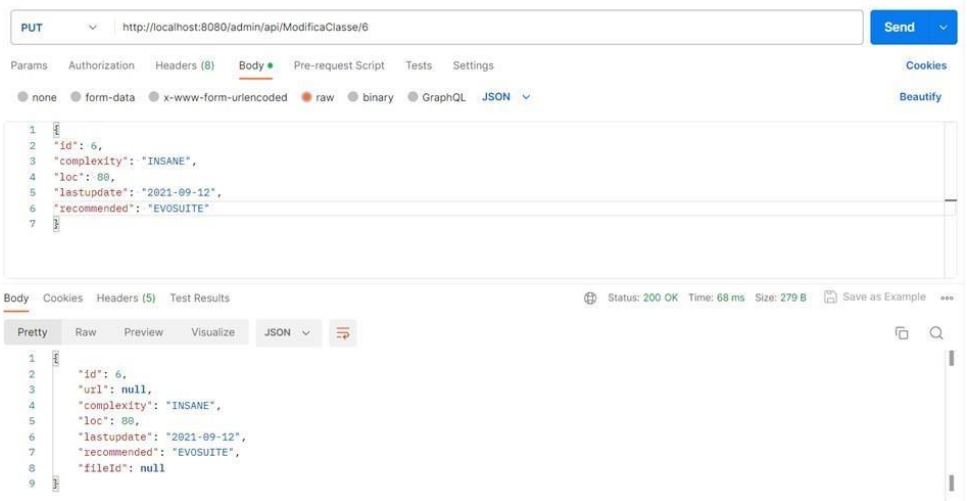


Figura 13.4 AdminClasseController – `/admin/api/ModificaClasse/{id}`.

- **`/admin/api/RicercaAvanzata`**

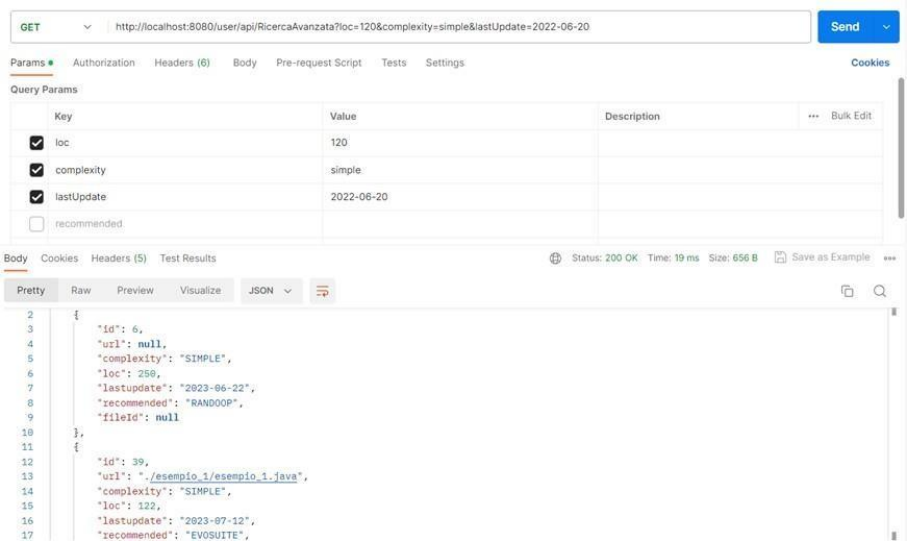


Figura 13.5 AdminClasseController – `/admin/api/RicercaAvanzata`.

13.4 REST API in UserClasseController

Di seguito si riporta il file `.json` contente la descrizione delle Api utilizzate nel modulo software `UserClasseController.java`, utilizzate per testare ed invocare le funzionalità riservare al giocatore.

Figura 13.6 Descrizione API `UserClasseController.java`.

• *Test delle API dell'UserClasseController.*

Si riporta il testing dell'API relativa alla funzionalità di Download di cui gode il giocatore.

- **`/user/downloadFile/{fileName:.+}`**

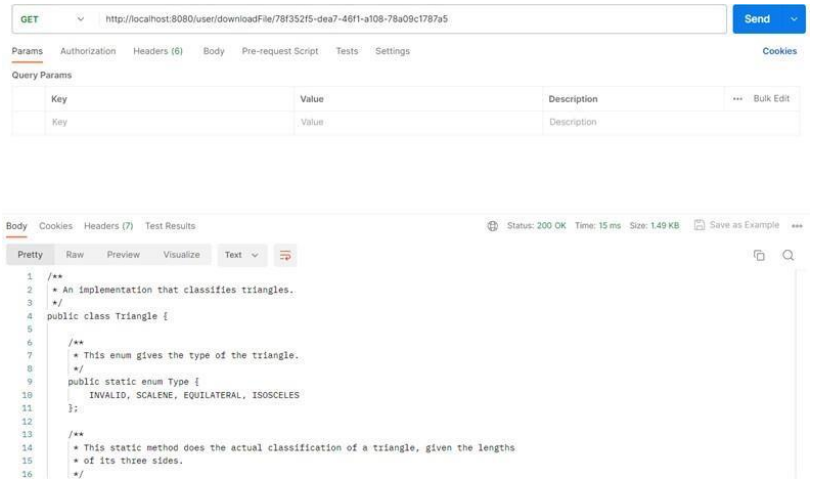


Figura 13.7 UserClasseController – `/user/downloadFile/{fileName:.+}`.

14 ACTIVITY DIAGRAM

Un diagramma di attività è un tipo di diagramma che permette di **descrivere un processo attraverso dei grafi** in cui i nodi rappresentano le **attività** e gli archi l'**ordine** con cui vengono eseguite. Questo tipo di diagramma viene spesso utilizzato per **pianificare, implementare e ottimizzare i processi** in ambito informatico e gestionale. Il diagramma di attività è un tipo di diagramma all'interno dell'Unified Modeling Language (UML). Può essere utilizzato per **modellare il flusso di lavoro**, i requisiti aziendali e le funzionalità del sistema.

14.1 EliminaClasse

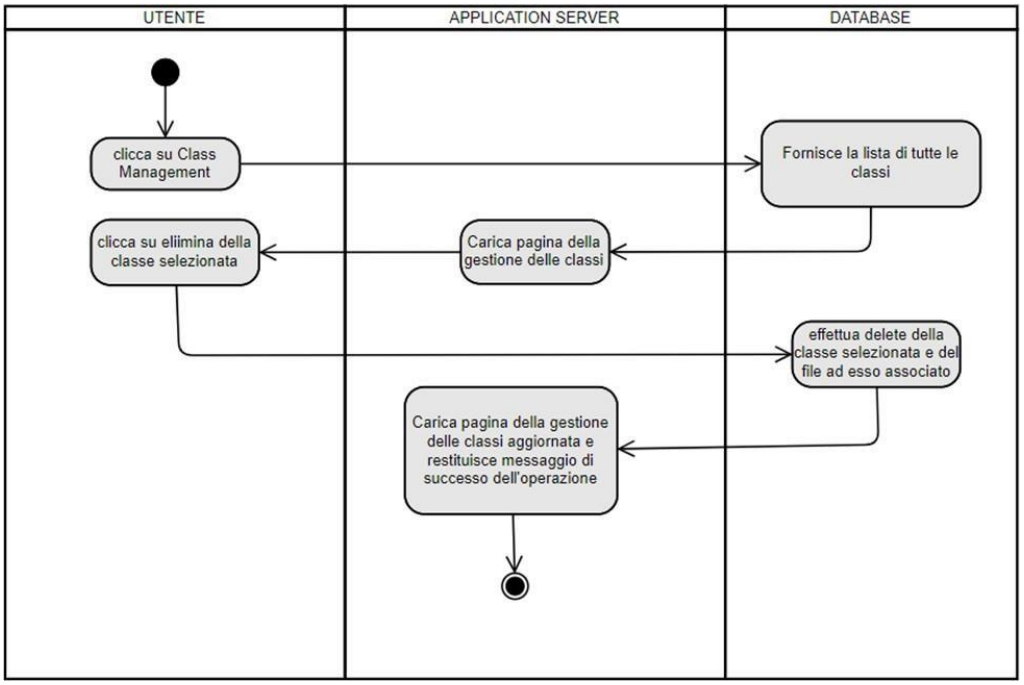


Figura 14.1 Activity Diagram – *EliminaClasse*.

14.2 DownloadCodiceClasse

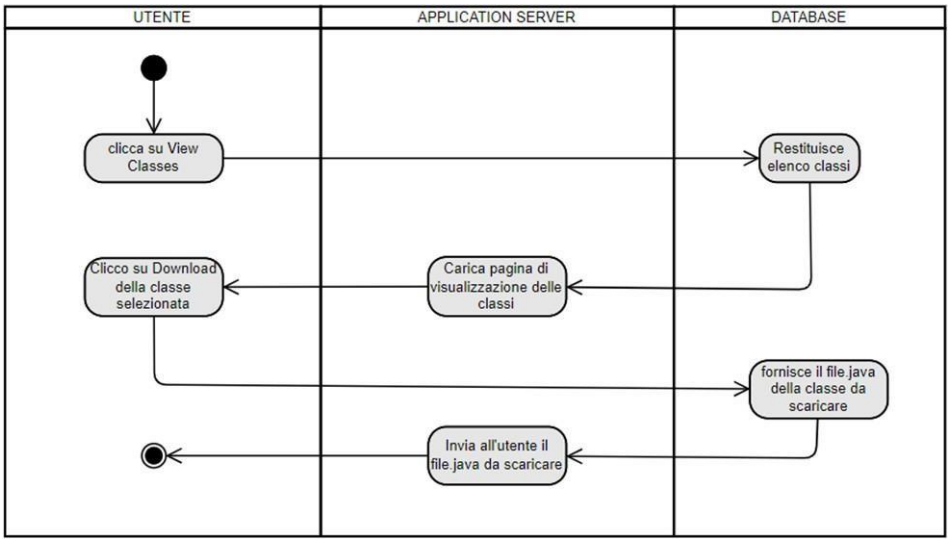


Figura 14.2 Activity Diagram – *DownloadCodiceClasse*.

14.3 AggiungiClasse

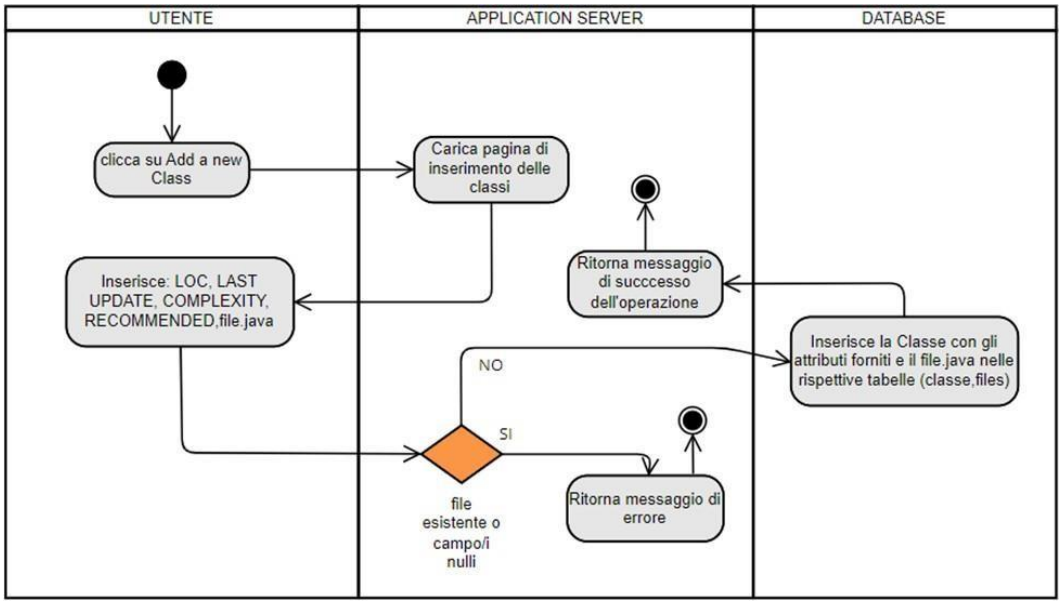


Figura 14.3 Activity Diagram – AggiungiClasse.

14.4 RicercaAvanzata

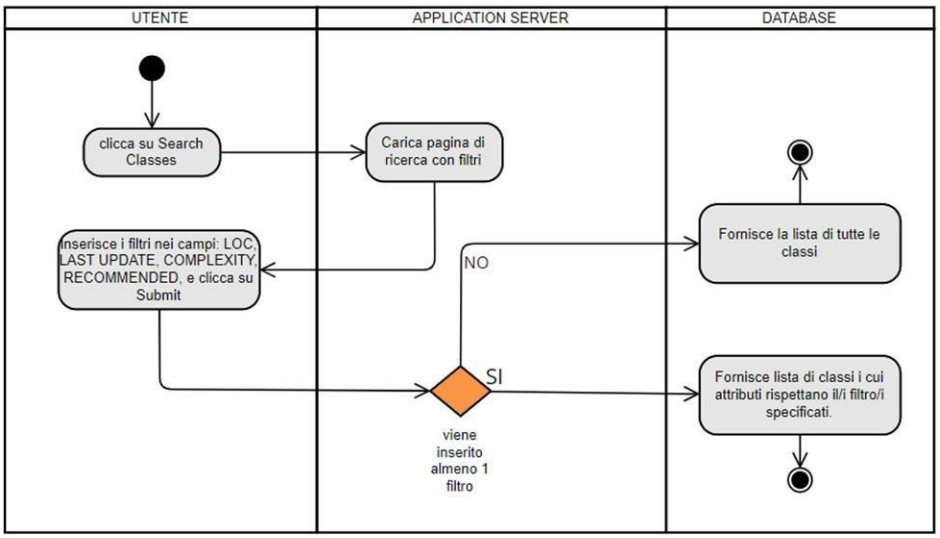


Figura 14.4 Activity Diagram – *RicercaAvanzata*.

14.5 ModificaClasse

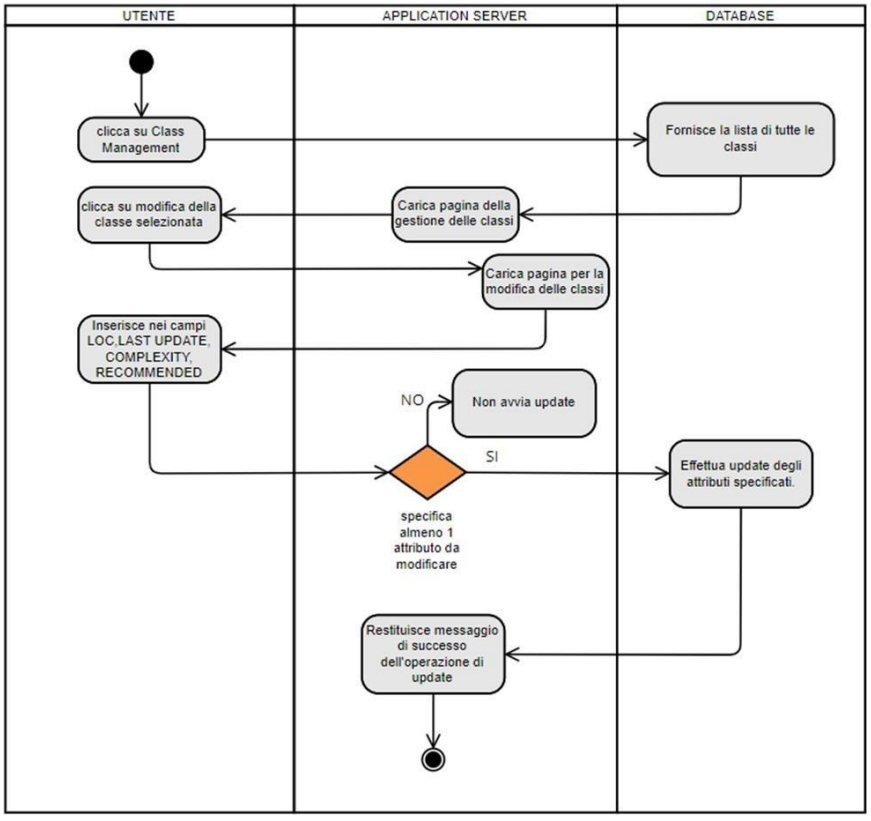


Figura 14.5 Activity Diagram – *ModificaClasse*.

14.6 VisualizzaElenco

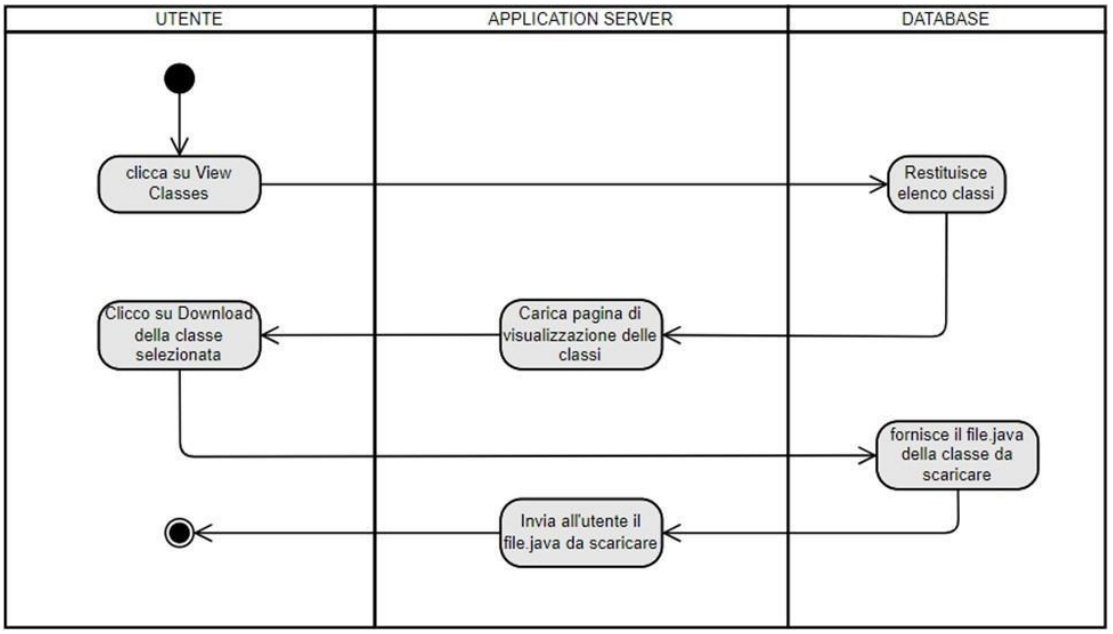


Figura 14.6 Activity Diagram – *VisualizzaElenco*.

15 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.

15.1 VisualizzaClassi e DownloadCodiceClasse

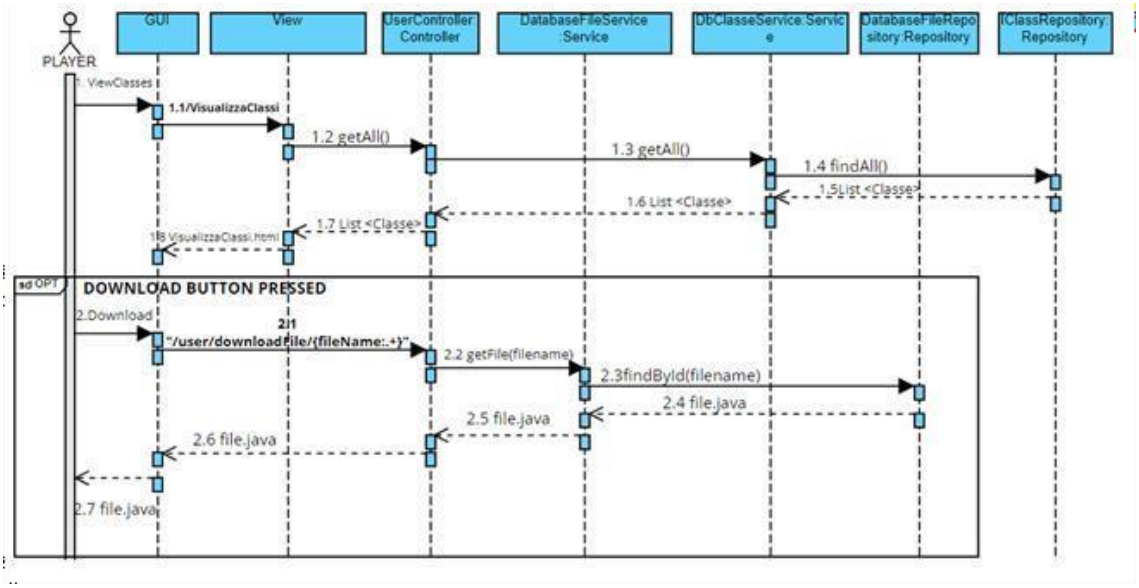


Figura 15.1 Sequence Diagram – *VisualizzaClassi* e *DownloadCodiceClassi*.

15.2 RicercaAvanzata

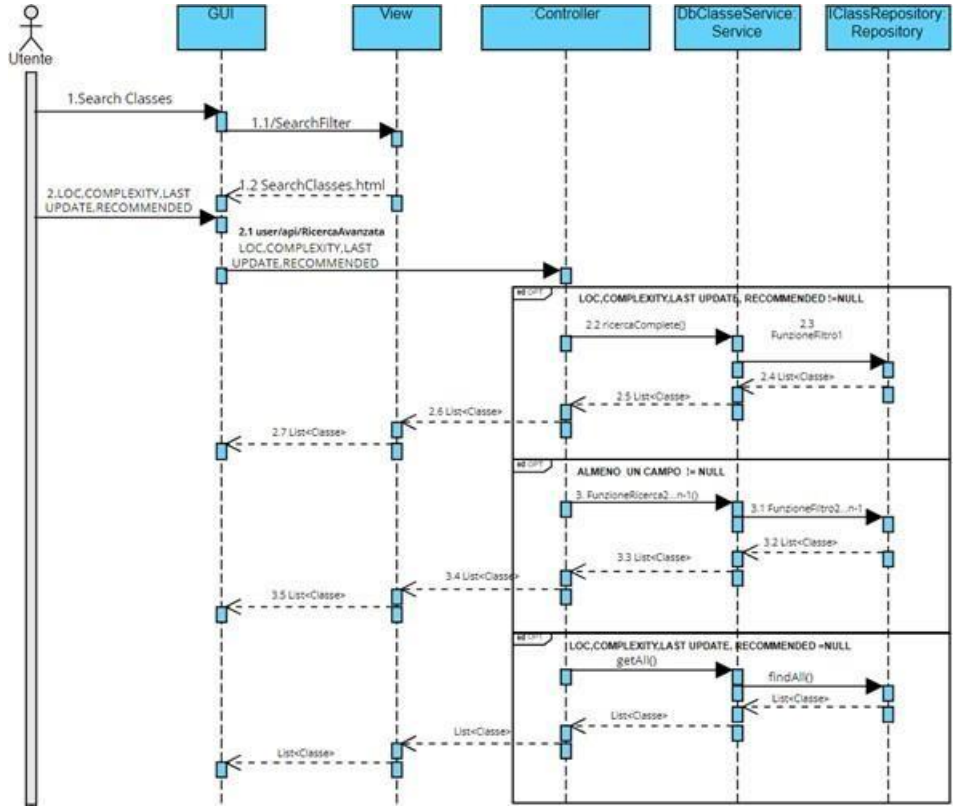


Figura 15.2 Sequence Diagram – *RicercaAvanzata*.

15.2.1 Analisi di tutte le possibili ricerche implementabili

i	**Funzione Ricerca *i*-esima**	**Funzione Filtro *i*-esimo**
1	`ricercaComplete(Level complexity, Opponent recommended,Integer loc, Date lastUpdate);`	`findByComplexityAndRecommendedAndLocGreaterThanEqualAndLastupdateGreaterThanEqual( Level key1,Opponent key2, Integer key3, Date key4);`
2	`ricercaRecommendedLastUpdateLoc (Opponent recommended, Date lastUpdate, Integer loc);`	`findByRecommendedAndLastupdateGreaterThanEqualAndLocGreaterThanEqual (Opponent key1, Date key2, Integer key3);`
3	`ricercaComplexityLastUpdateLoc (Level complexity, Date lastupdate, Integer loc);`	`findByComplexityAndLastupdateGreaterThanEqual AndLocGreaterThanEqual (Level key1, Date key2, Integer key3);`

4	ricercaComplexityRecommendedLoc (Level complexity, Opponent recommended, Integer loc)	findByComplexityAndRecommendedAnd LocGreaterThanEqual(Level key1, Opponent key2, Integer key3);
5	ricercaComplexityLastUpdateReco mmended(Level complexity, Date lastUpdate,Opponent recommended);	findByComplexityAndRecommendedAnd LastupdateGreaterThanEqual (Level key1, Opponent key3, Date key2);
6	ricercaLastUpdateLoc(Date lastupdate,Integer loc);	findByLastupdateGreaterThanEqualA ndLocGreaterThanEqual(Date key1,Integer key2);
7	ricercaRecommendedLoc(Opponent recommended,Integer loc)	findByRecommendedAndLoc GreaterThanEqual(Opponent key1,Integer key2);
8	ricercaRecommendedLastUpdate(Op ponent recommended,Date lastUpdate);	findByRecommendedAndLastupdate GreaterThanEqual(Opponent key1,Date key2);
9	ricercaComplexityLoc(Level complexity,Integer loc);	findByComplexityAndLocGreaterThan Equal(Level key1, Integer key2);
10	ricercaComplexityLastUpdate(Lev el complexity,Date lastUpdate);	findByComplexityAndLastupdate GreaterThanEqual(Level key1,Date key2);
11	ricercaComplexityRecommended (Level complexity,Opponent recommended);	findByComplexityAndRecommended (Level key1, Opponent key2);
12	ricercaRecommended(Opponent recomended);	findByRecommended(Opponent key);
13	ricercaLoc(Integer loc);	findByLocGreaterThanEqual(Integer key);
14	ricercaLastUpdate(Date lastUpdate);	findByLastupdateGreaterThanEqual (Date key);
15	ricercaComplexity(Level complexity);	findByComplexity(Level key1);

15.3 GestioneClasse

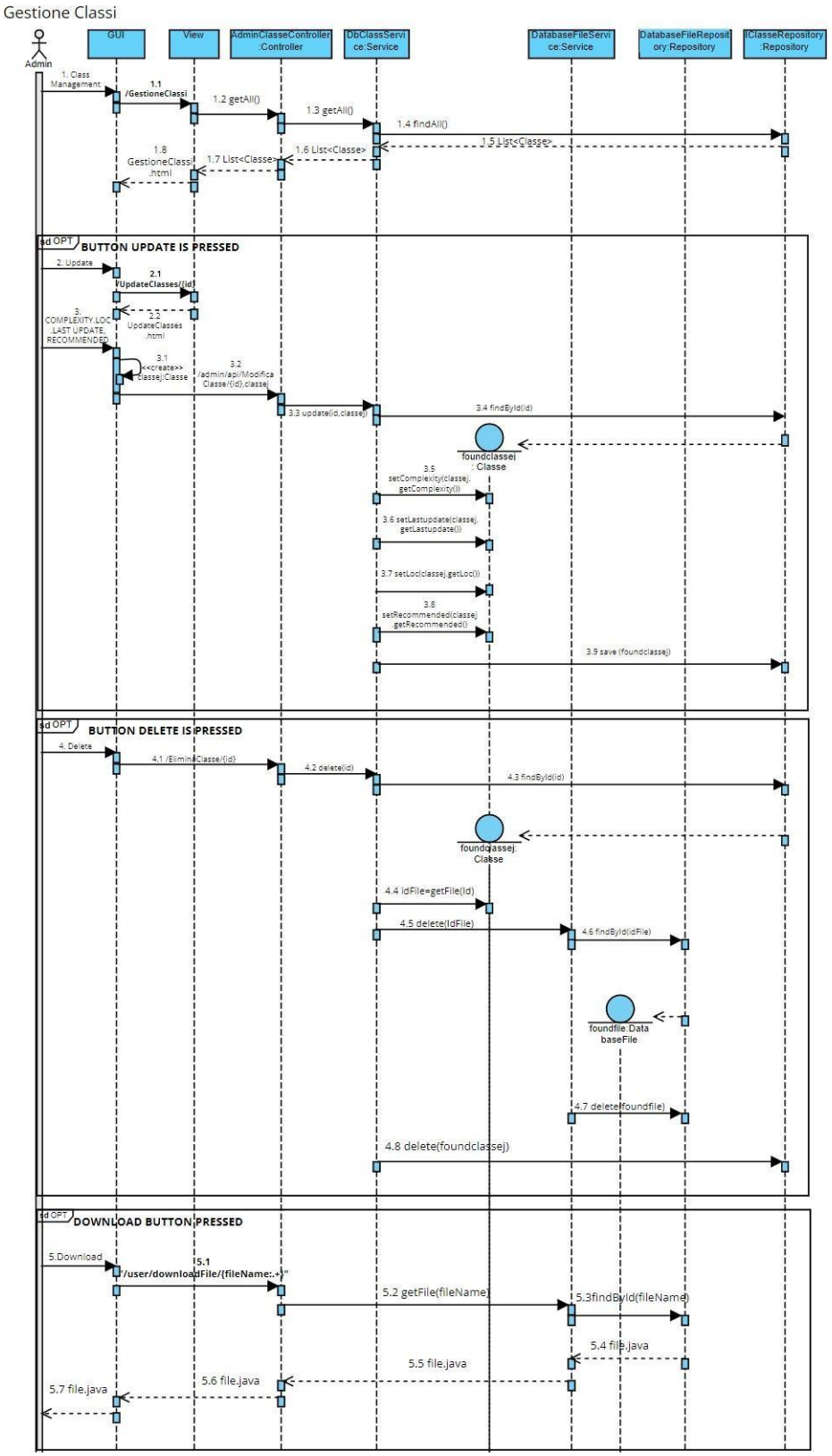


Figura 15.3 Sequence Diagram – *GestioneClasse*

15.4 AggiuntaClasse

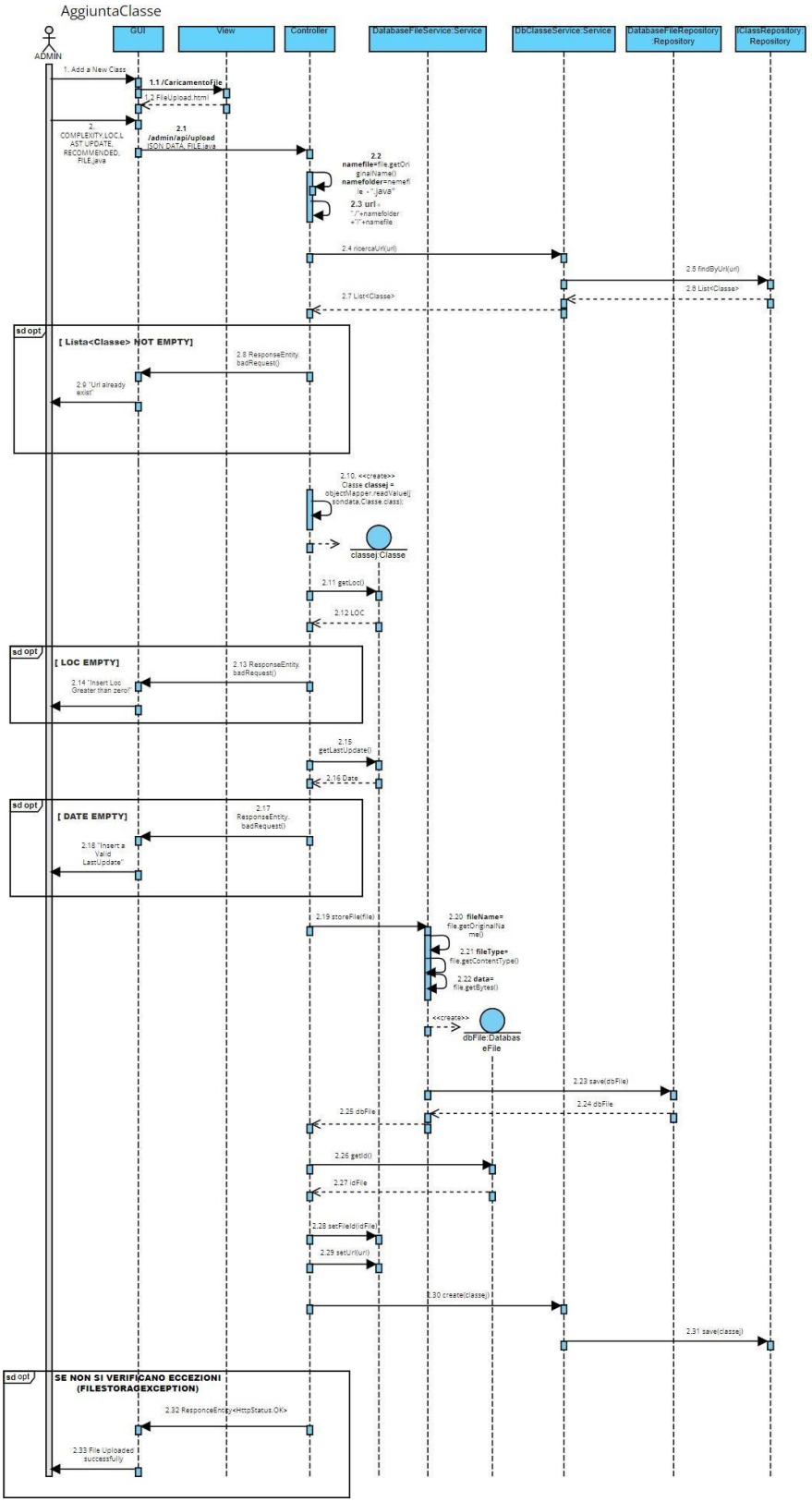


Figura 15.4 Sequence Diagram – *AggiuntaClasse*.

16 COMMUNICATION DIAGRAMS

I diagrammi di comunicazione sono strumenti visivi utilizzati per rappresentare le **interazioni** e le **comunicazioni** tra gli attori o i componenti di un sistema software. Questi diagrammi sono parte della Unified Modeling Language (UML) e forniscono una visione chiara delle connessioni tra gli oggetti coinvolti.
Nei diagrammi di comunicazione, gli oggetti sono rappresentati da rettangoli con il nome dell'oggetto all'interno. Le frecce tra gli oggetti indicano le comunicazioni e le interazioni tra di essi. Inoltre, consentono di visualizzare il flusso delle informazioni, i messaggi scambiati tra gli oggetti e l'ordine temporale delle comunicazioni. Possono essere utilizzati per analizzare e comprendere i processi di comunicazione all'interno di un sistema, identificare le dipendenze e migliorare la progettazione e la comprensione complessiva del sistema.
Questi diagrammi sono utili per gli sviluppatori, gli analisti e gli stakeholder del progetto per comunicare in modo efficace e rappresentare visivamente le interazioni tra i componenti del sistema software.

16.1 Aggiunta Classe Communication Diagram

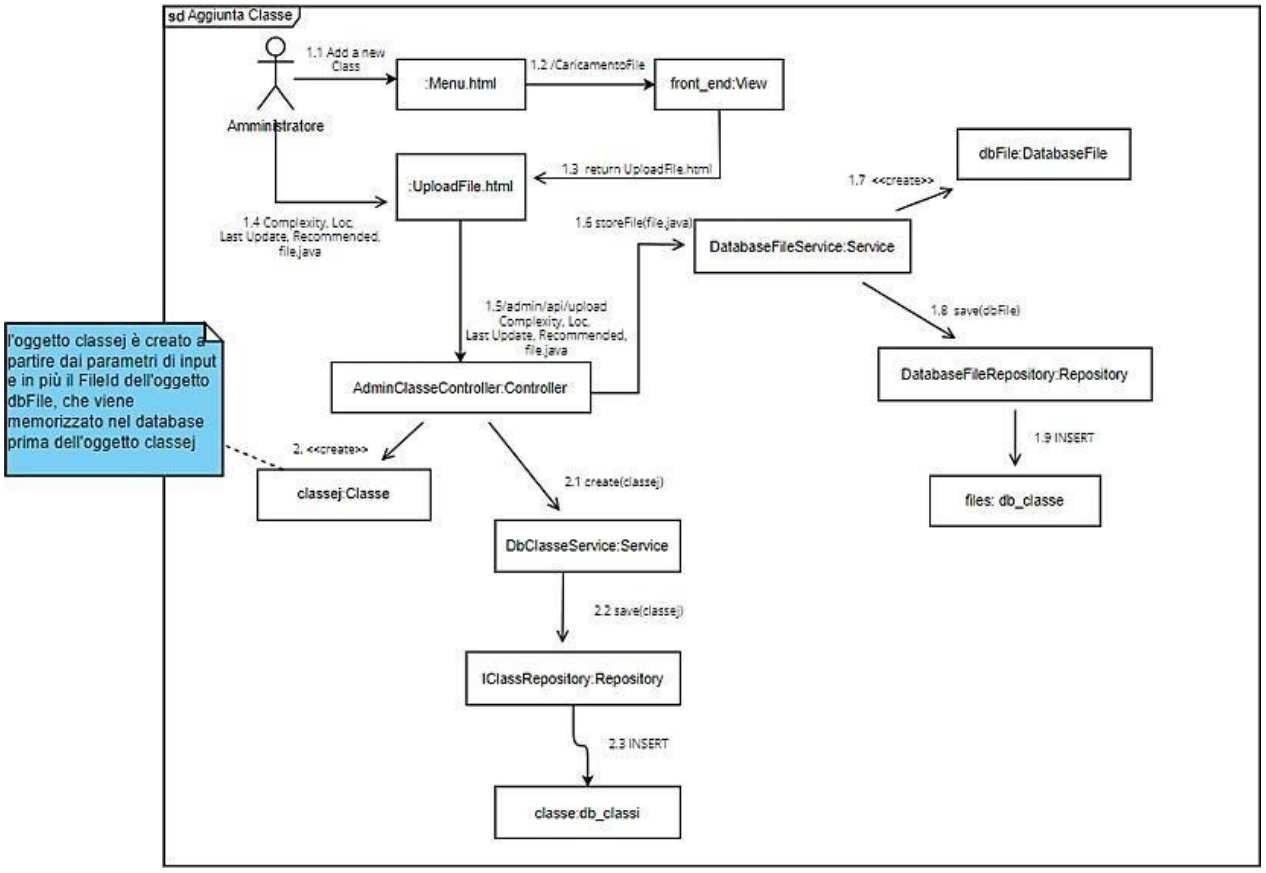


Figura 16.1 Aggiunta Classe Communication Diagram

16.2 Gestione Classi Communication Diagram

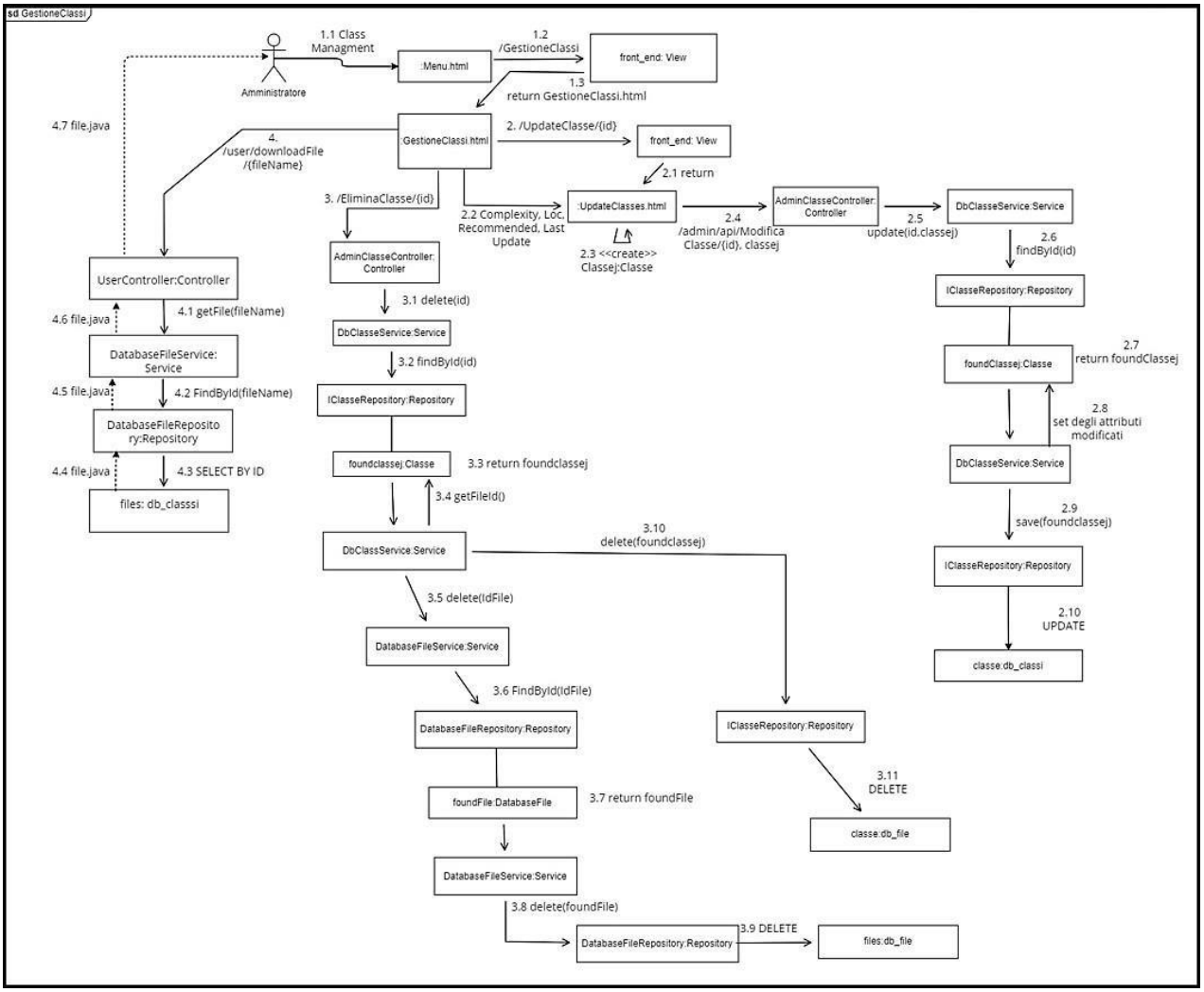


Figura 16.2 Gestione Classi Communication Diagram

16.3 Ricerca Avanzata Communication Diagram

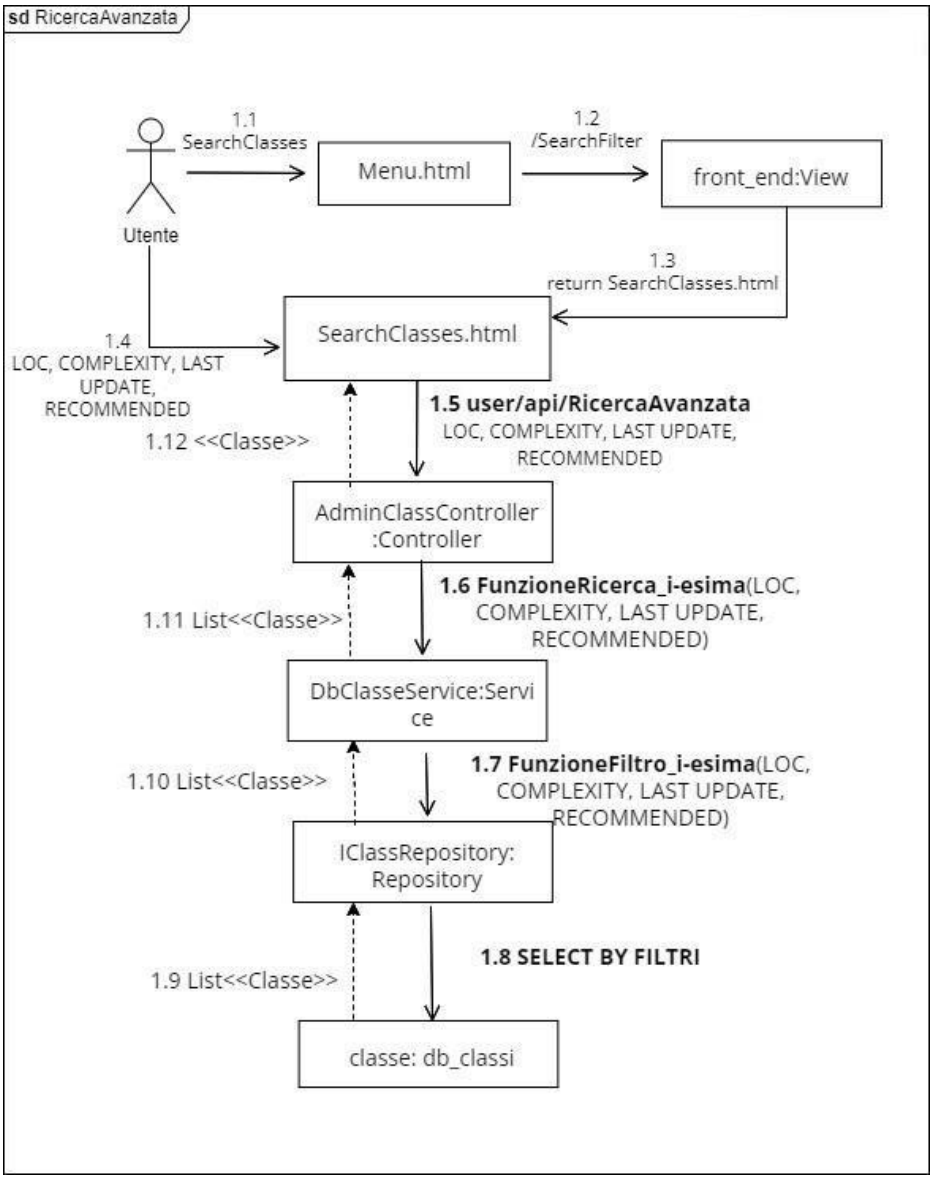


Figura 16.3 Ricerca Avanzata Communication Diagram

16.4 Visualizza e Download Communication Diagram

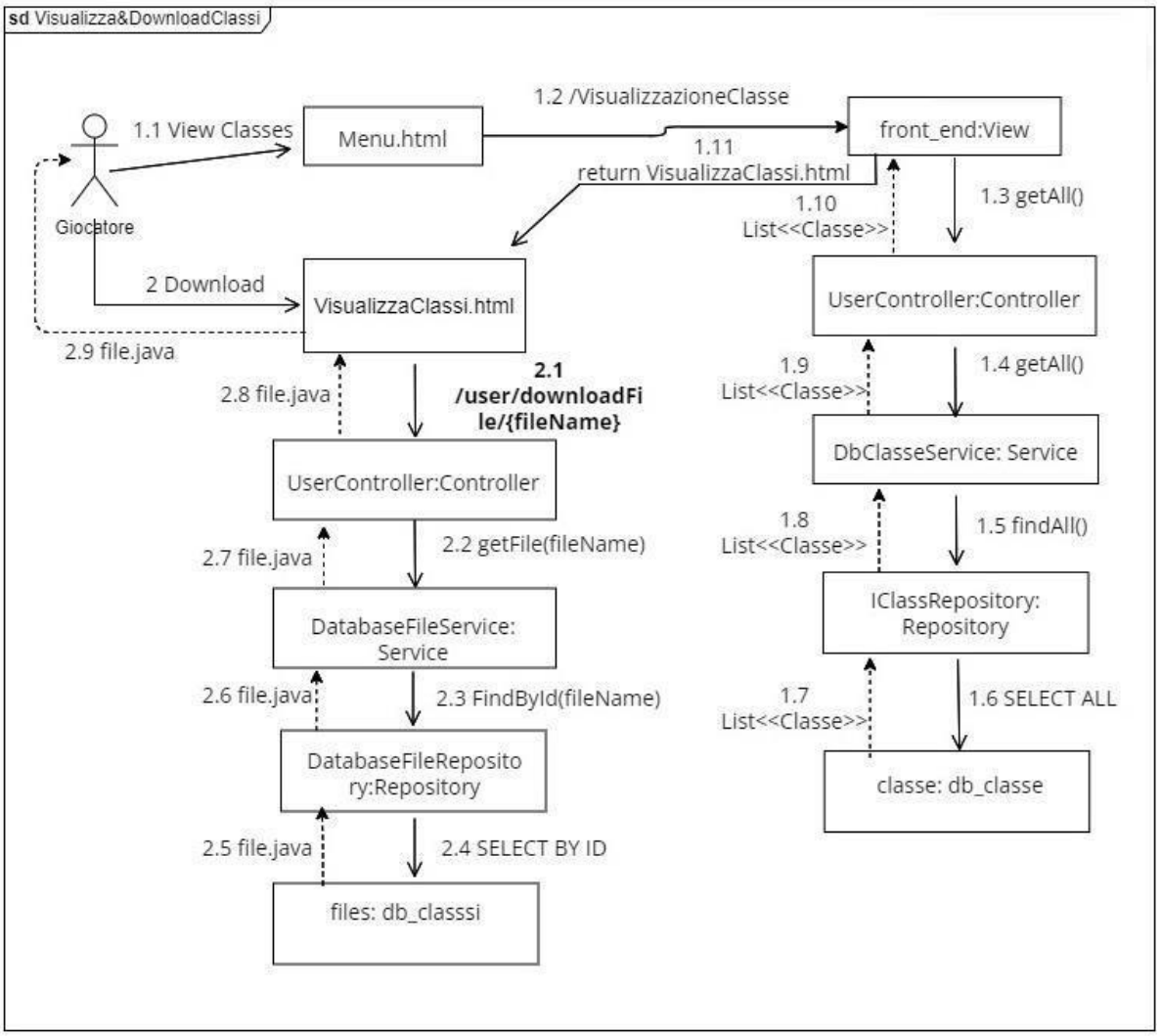


Figura 16.4 Visualizza e Download Communication Diagram

17 STATE DIAGRAM

Questo tipo di diagramma di progettazione è stato utilizzato per gli oggetti `Classe` e `File`. In particolare, bisogna, per ciascuno di essi, rappresentare le sequenze di stati, le risposte e le azioni che l'oggetto attraversa durante la sua vita in risposta agli stimoli.
Si parta con l'oggetto `Classe`. Vengono descritti tutti gli stati che può attraversare a partire dal caso d'uso Gestione Classi, o Ricerca Avanzata. Ne consegue che, considerando la vista dell'Admin, le operazioni possibili sono quelle di ricerca, di cancellazione e di modifica delle classi stesse: 1) selezionando il menu di ricerca e specificati i parametri, se i filtri sono validi verranno visualizzate le classi corrispondenti ai parametri di ricerca; 2) selezionando invece l'icona del cestino, la classe verrà cancellata e non più mostrata nel relativo elenco; 3) infine, alla selezione dell'icona della matita, i relativi parametri potranno essere modificati se validi.
Per il `File` si seguono ragionamenti analoghi. Un `File` può essere ricercato similmente ad una `Classe`, visualizzato nel menu e successivamente cancellato in egual modo. Per di più, è possibile selezionare l'icona di download al fine di scaricare lo stesso in locale.

17.1 Class Object State Diagram

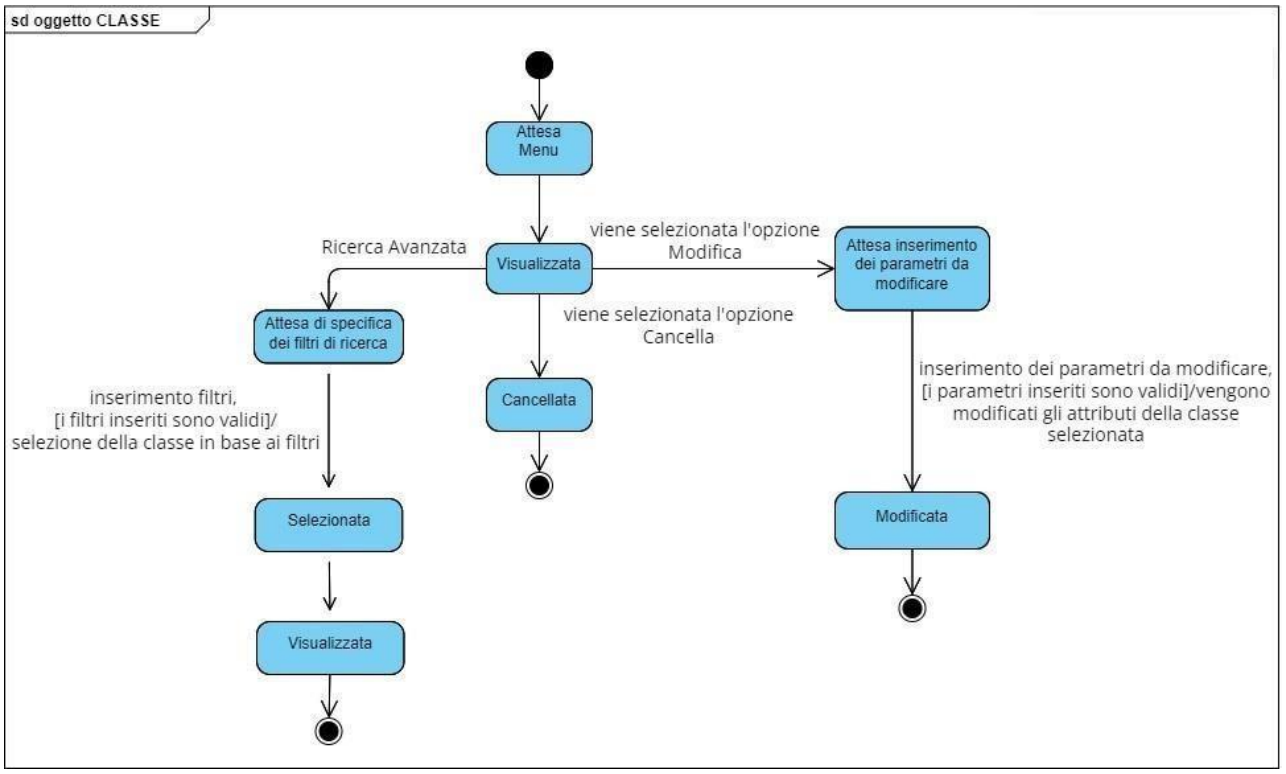


Figura 17.1 Class State Diagram.

17.2 File Object State Diagram

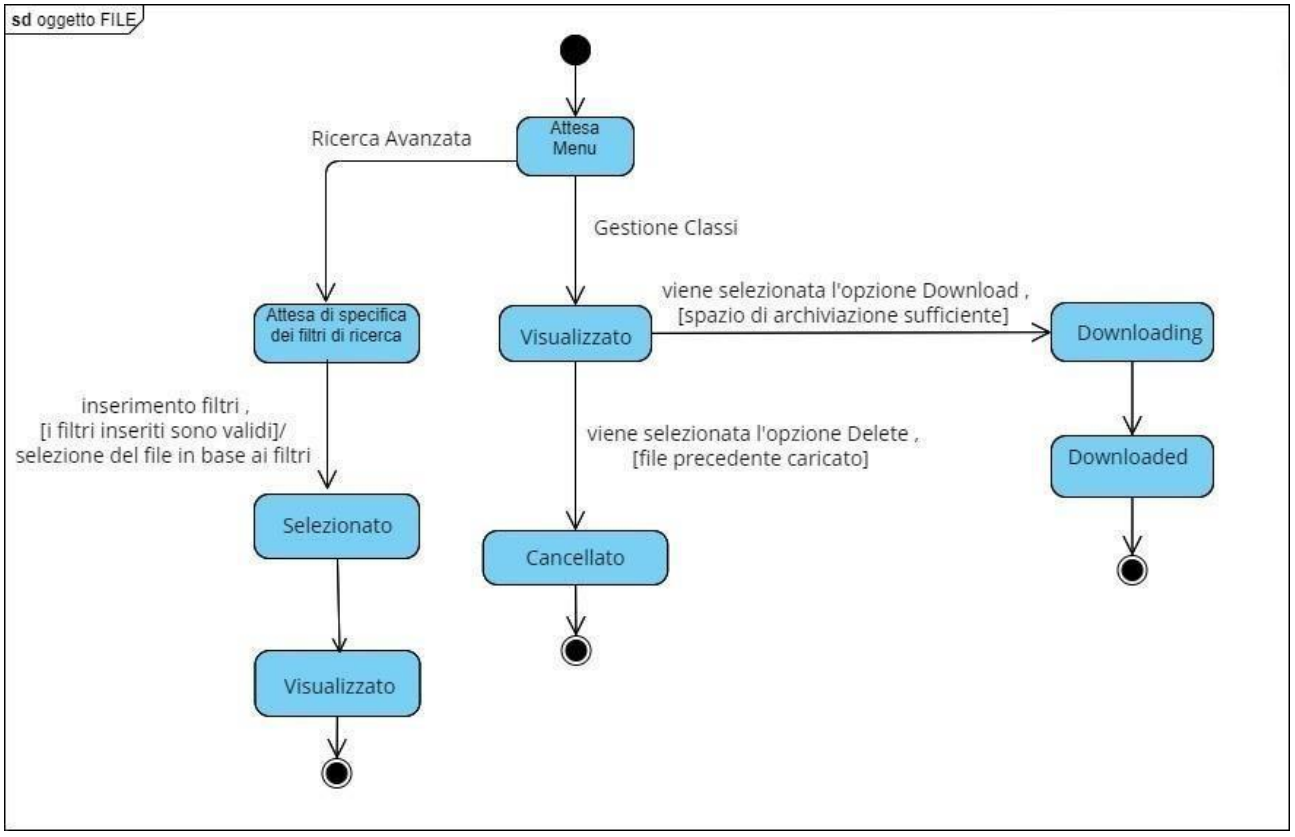


Figura 17.2 File State Diagram.

18 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/T1-G3

3. Estrarre il file **zip** scaricato (seguire la seguente guida in caso di difficoltà).

https://support.microsoft.com/it-it/windows/comprimere-e-decomprimere-file-f6dde0a7-0fec-8294-e1d3-703ed85e7ebc#:~:text=Apri%20Esplora%20file%20e%20trova,sulla%20cartella%20compressa%20per%20aprirla

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'intero di docker.
Per visualizzare la corretta esecuzione del comando digitare `docker-compose ps` che fornisce lo stato dei container definiti nel file `docker-compose.yml`.

5. Arrivati a questo punto l'applicazione è utilizzabile senza ulteriori sforzi, basterà recarsi sul proprio **browser web** e digitare il seguente comando: http://localhost:8080/.
Attraverso il menù laterale sarà possibile aprire schermate diverse con le diverse funzionalità richieste al task T1.

6. Per la gestione e configurazione del database è possibile scegliere fra due strade alternative. La prima è rappresentata da *Database Client* e rappresenta l'alternativa consigliata per semplicità e usabilità. Per scaricare questa estensione, aprire VSCode (se non si è in possesso recarsi al seguente sito per scaricare la versione aggiornata: https://www.dz-techs.com/it/windows-vs-code-set-up-guide), e dal menù laterale premere sull'icona "Extension", ricercare e scaricare dalla barra di ricerca l'estensione *Database Client*.
Una volta scaricata l'estensione, chiudere la finestra di VSCode e riavviare il PC, al termine del riavvio aprire una nuova finestra di VSCode; sul menù laterale dovrebbe essere apparsa una sezione `database`, cliccare quella sezione e accedere al database chiamato `db_classi`.
Tramite i diversi comandi MySQL (`alter table`, `create table`, …) è possibile modificare le tabelle `classe` e `file` o crearne di nuove. Cliccare “Execute” per renderle attive.
La seconda alternativa invece, prevede di agire direttamente dall'app Docker. Digitare `docker exec -it <nome_container o ID_container> bash` per entrare all'interno di un container Docker in esecuzione e avviare una shell interattiva al suo interno.
Successivamente digitare `mysql -u root -p` per accedere all'interfaccia della riga di comando di MySQL e connettersi al database "`db_classi`" utilizzando l'utente `root`.
Alternativamente digitare interamente `docker exec -it my-mysql-container mysql -u root -p db_classi`.
Questo comando si connetterà direttamente all'interfaccia della riga di comando di MySQL all'interno del container `my-mysql-container` e selezionerà il database `db_classi`. Verrà richiesto di inserire la password per l'utente `root`. Nel caso specifico con la password vuota, si può semplicemente premere nuovamente Invio senza inserire alcuna password.
ATTENZIONE: è necessario lasciare il database attivo durante tutta la durata delle operazioni di gestione del database

18.1 INSTALL VIEW

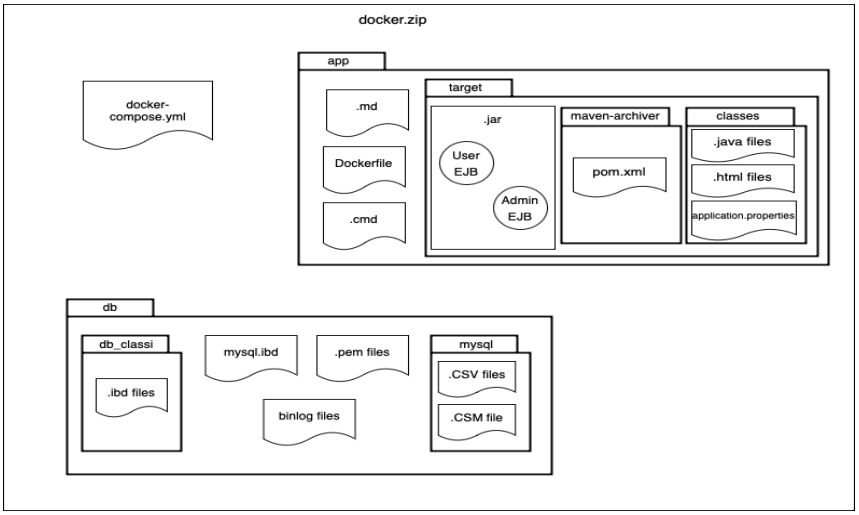


Figura 18.1 Install View.

18.2 DEPLOYMENT DIAGRAM

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.

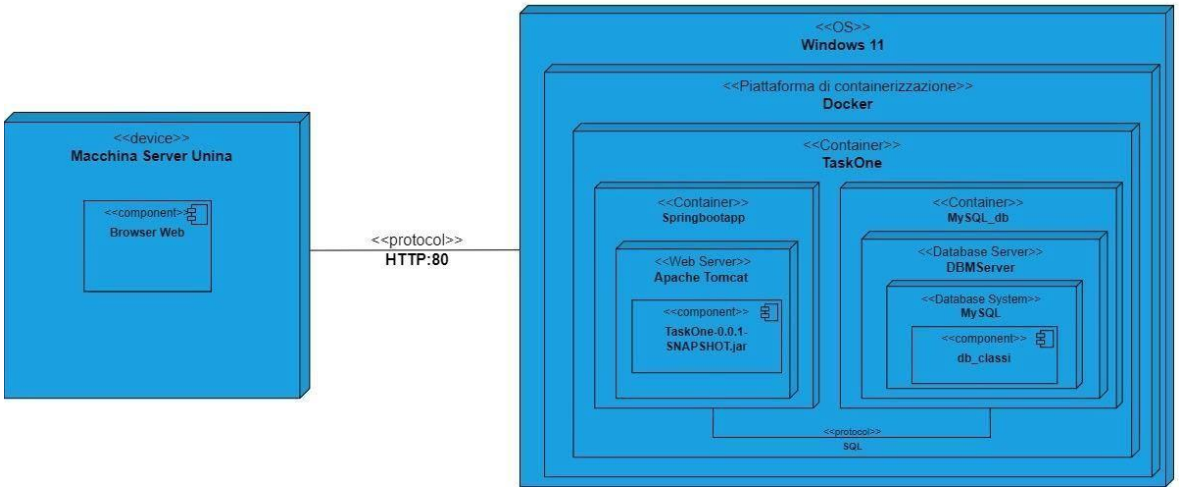


Figura 18.2 Deployment Diagram.

19. 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 dalla repository del Team Software Architecture Design. [Path: Documenti/General/EsempiDiClassiDaTestare/generated_tests_2016.zip]

Di seguito vengono riportati i link a cui si fa riferimento nella colonna input:

[1] http://localhost:8080/GestioneClassi

[2] http://localhost:8080/CaricamentoFile

[3] http://localhost:8080/SearchFilterAdmin

[4] http://localhost:8080/UVisualizzaClassi

[5] http://localhost:8080/SearchFilter

Test Case Id	*Descrizione*	*Precondizioni*	*Input*	*Output Ottenuti*	*Postcondizioni Ottenute*	*Esito (FAIL; PASS)*
1 CM	L'amministratore visualizza l'elenco delle classi disponibili	C'è almeno una classe nel database.	L'utente 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 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'amministatore nella pagina [1] preme sul tasto "Download".	File java scaricato nei 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'amministatore nella pagina [1] preme sull'icona della matita blu. L'amministratore riconfigura i parametri della classe selezionata.	Viene mostrato un pop-up con su scritto "Class Update Successfully"	La classe è modificata.	PASS

			Preme poi sul pulsante “SUBMIT”			
1 DEL	Si vuole eliminare una classe tra quelle disponibili	C’è almeno una classe nel database.	L’amministatore 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 l’elenco delle classi rimanenti.	PASS
1ADD	Si vuole aggiungere una nuova classe.	Si accede al link [2] cliccando sulla sezione “Add a new class”.	Complexity: Moderate Lines of code: 67 Last Update 08/06/2023 Opponent: EvoSuite File: TestHslColor.java	Viene mostrato un messaggio: “File uploaded Successfully”	La classe è presente fra quelle disponibili nel database	PASS
2ADD	Si sbaglia ad inserire lines of codes: float, null, nonvalid.	Si accede al link [2] cliccando sulla sezione “Add a new class”.	Complexity: Moderate Lines of code: nonvalid Last Update 08/06/2023 Opponent: EvoSuite File: TestByArrayHashMap.java	Viene mostrato un messaggio: “Insert Loc Greater than 0”	La classe non viene aggiunta al database.	PASS
3ADD	Si sbaglia ad inserire lines of codes: float, null, nonvalid.	Si accede al link [2] cliccando sulla sezione “Add a new class”.	Complexity: Moderate Lines of code: 40 Last Update / Opponent: EvoSuite File: TestByArrayHashMap.java	Viene mostrato un messaggio: “Insert a valid last update”	La classe non viene aggiunta al database.	PASS

4ADD	Si sbaglia ad inserire il file java: non si inserisce.	Si accede al link [2] cliccando sulla sezione “Add a new class”.	Complexity: Moderate Lines of code: 40 Last Update 10/07/2023 Opponent: EvoSuite File: /	Viene mostrato un messaggio: “file is not uploaded. Error Occurred ”	La classe non viene aggiunta al database.	PASS
5ADD	Si sbaglia ad inserire il file java: si inserisce un file duplicato.	Si accede al link [2] cliccando sulla sezione “Add a new class”. Il file da inserire sia già presente nel database con lo stesso nome.	Complexity: Moderate Lines of code: 40 Last Update 10/07/2023 Opponent: EvoSuite File: TestHSLColor.java	Viene mostrato un messaggio: “Url already exist ”	La classe non viene aggiunta al database.	PASS
1SE	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	Complexity:Any LOC: 67	Restituzione a video della classe specificata.	La classe può essere scaricata, eliminata o modificata	PASS
2SE	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	Complexity:Any LOC: 125	Restituzione a video dell’elenco di classi vuoto.		PASS

3SE	Si ricerca attraverso appositi filtri una serie di classi fra quelle disponibili con paremetri maggiori o uguali di quelli specificati.	Si accede al link [3] cliccando sulla sezione “Search Class”. Esiste almeno una classe fra quelle disponibili con paremetri maggiori o uguali di quelli specificati.	LOC: 66 Last Update: 01/01/2023	Restituzione a video delle classi che rispettano i vincoli.	La classi possono essere scaricate, eliminate o modificate.	PASS
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 alucuna 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 nell pagina [4] preme sul tasto “Download”.	File java scaricato nei Download del Browser Web.	L’utente riesce ad accedere al download del file.	PASS