

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II



SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE
DELL'INFORMAZIONE

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

Documentazione Task T-1 Gruppo G11

Team:

Luigi Scotto Rosato
Leonardo Sole
Davide Vitale

M63001488
M63001487
M63001469

Docente:

Prof. Anna Rita Fasolino

Anno Accademico:

2022/2023

Indice

Capitolo 1: Introduzione	4
Capitolo 2: Specifiche di progetto	5
2.1 Utenti del sistema	5
2.2 Glossario dei termini	5
2.3 Requisiti Funzionali	6
2.4 Storie utente	6
2.5 Requisiti Non Funzionali	7
2.6 Visione ad alto livello del sistema	8
Capitolo 3: Processo di Sviluppo	13
3.1 Strumenti Software di Supporto	13
3.2 Analisi delle Tecnologie Software	14
Capitolo 4: Fase di analisi	14
4.1 Use Case Diagram	15
4.2 Sequence Diagram	15
4.2.1 Sequence Diagram: RicercaClassi	15
4.2.2 Sequence Diagram: elencaClassi	16
4.2.3 Sequence Diagram: ordinaClassi	17
4.2.4 Sequence Diagram: filtraClassi	18
4.2.5 Sequence Diagram: DownloadClasse	19
4.2.6 Sequence Diagram: ModificaClasse	20
4.2.7 Sequence Diagram: UploadClasse	21
4.2.8 Sequence Diagram: eliminaClasse	22
4.2.9 Sequence Diagram: elencaReport	22
4.2.10 Sequence Diagram: nuovoLike	23
4.2.11 Sequence Diagram: nuovoReport	23
4.3 Activity Diagram	24
4.3.1 Activity Diagram: RicercaClassi	24
4.3.2 Activity Diagram: elencaClassi	25
4.3.3 Activity Diagram: ordinaClassi	26
4.3.4 Activity Diagram: filtraClassi	27
4.3.5 Activity Diagram: DownloadClasse	28
4.3.6 Activity Diagram: ModificaClasse	29
4.3.7 Activity Diagram: UploadClasse	30
4.4 System Domain Model	31
Capitolo 5: Fase di Progettazione	32
5.1 Pattern Architetturale MVC	32
5.1.1 Spring MVC	32
5.2 Component Diagram	33
5.3 Package Diagram	34
5.4 Application Class Diagram	34
5.5 Deployment	35
5.6 Integrabilità	36

5.6.1 API Locali	36
5.6.2 Strutture Dati	36
5.6.2.1 ClassUT	36
5.6.2.2 Admin	37
5.6.2.3 Interaction	37
5.6.2.4 Operation	37
5.6.3 Services	38
5.6.3.1 Login admin	38
5.6.3.3 Registrazione admin	38
5.6.3.4 Modifica Classe	38
5.6.3.5 Download Classe	39
5.6.3.6 Ricerca Classe	39
5.6.3.7 Eliminazione Classe	40
5.6.3.8 Upload Classe	40
5.6.3.9 Elenca Classi	40
5.6.3.10 Filtra Classe	40
5.6.3.11 Ordinamento Classi	41
5.6.3.12 Nuovo Report	41
5.6.3.13 Nuovo Like	41
5.6.3.14 Upload Interaction	41
5.6.3.15 Elenca Report	42
5.6.3.16 Elenca Interaction	42
5.6.4 Utils	42
5.6.4.1 getLikes	42
5.6.4.2 findReport	43
5.6.4.3 findByText	43
5.6.4.4 findAdminByUsername	43
5.6.4.5 searchAndFilter	43
5.6.4.6 filterByCategory	44
5.6.4.7 orderByDate	44
5.6.4.10 searchAndFilter	44
5.6.4.11 downloadClassFile	45
5.6.4.12 saveClassFile	45
5.6.4.13 deleteDirectory	45
5.6.5 Rest API	46
5.6.5.1 Elenca classi "/home" (GET)	46
5.6.5.2 Ricerca Classe per nome "/home/{text}" (GET)	47
5.6.5.3 Ordina classi per nome "/orderbyname" (GET)	47
5.6.5.4 Ordina classi per data "/orderbydate" (GET)	48
5.6.5.5 Filtra classi per categoria "/Cfilterby/{category}" (GET)	48
5.6.5.6 Filtra classi per categoria "/Cfilterby/{category}" (GET)	49
5.6.5.7 Filtra classi per categoria "/Dfilterby/{difficulty}" (GET)	49
5.6.5.8 Filtra classi per categoria "/downloadFile/{name}" (GET)	49
5.6.5.9 Upload della descrittore classe+ file.java "/uploadFile (POST)	50

5.6.5.10	Modifica classe “/update/{name} (POST)	51
5.6.5.11	Nuovo like ad una classe “/newLike/{name} (POST)	51
5.6.5.12	Nuovo report ad una classe “/newReport/{name} (POST)	52
5.6.5.12	Registrazione Admin “/registraAdmin/ (POST)	52

Capitolo 1: Introduzione

Il compito assegnato al Team è quello di sviluppare un catalogo online di classi di programmazione, accessibile sia agli studenti che agli amministratori. Gli utenti avranno la possibilità di cercare, visualizzare, filtrare e scaricare classi di programmazione. Inoltre gli studenti, potranno sbloccare nuove classi man mano che avanzano di livello. Gli amministratori, invece, avranno il compito di inserire, modificare e cancellare classi all'interno del catalogo.

L'applicazione fornirà agli utenti un'interfaccia intuitiva per navigare e cercare tra le varie classi di programmazione disponibili. Saranno in grado di utilizzare filtri per restringere la ricerca in base a diversi criteri come il linguaggio di programmazione, il livello di difficoltà o la categoria di interesse.

Inoltre, gli utenti potranno visualizzare i dettagli di ogni classe, come la descrizione, gli obiettivi di apprendimento, i prerequisiti e il materiale di supporto. Saranno in grado di scaricare i file associati a ogni classe, come esempi di codice, presentazioni o documentazione.

Per motivare e coinvolgere gli studenti, il sistema permetterà loro di sbloccare nuove classi man mano che acquisiscono competenze e progrediscono di livello. Ciò li motiverà a continuare a imparare e ad avanzare nel percorso di apprendimento.

Gli amministratori avranno accesso a un'interfaccia dedicata per gestire il catalogo. Potranno aggiungere nuove classi, modificarne i dettagli o eliminarle se necessario. Questa funzionalità consentirà agli amministratori di mantenere il catalogo aggiornato e rilevante per gli utenti.

L'obiettivo principale dell'applicativo sarà fornire una piattaforma user-friendly e funzionale per studenti di diversi livelli di esperienza in programmazione, consentendo loro di accedere a una vasta gamma di classi di programmazione e supportando sia il loro apprendimento che il loro progresso nel campo della programmazione.

Capitolo 2: Specifiche di progetto

2.1 Utenti del sistema

- **Studenti:** Gli studenti sono gli utenti principali del sistema, possono accedere al catalogo di classi di programmazione, cercare, visualizzare, filtrare e scaricare le classi disponibili; possono inoltre interagire con le classi attraverso un like o un report.
- **Amministratori:** Gli amministratori sono responsabili della gestione del catalogo di classi di programmazione. Hanno accesso privilegiato per inserire nuove classi, modificarne i dettagli o eliminarle se necessario. Possono accedere a un'interfaccia dedicata per gestire le operazioni amministrative. Hanno il compito di mantenere il catalogo aggiornato, garantendo che le informazioni sulle classi siano accurate e pertinenti. Possono monitorare l'utilizzo del sistema visualizzando le interazioni degli studenti con le classi.

2.2 Glossario dei termini

Termine	Alias	Descrizione	Formato
Amministratore	Admin, Utente	Un utente con autorizzazioni speciali per gestire il software	Costituito da stringhe riguardanti l'username e la password

Studente	Utente, Fruitore del servizio	Studente che utilizza l'applicazione per accedere alle funzionalità disponibili, senza poter effettuare operazioni di amministrazione	Costituito da stringhe riguardanti i dati anagrafici, l'id, l'username e la password
Class List	Elenco delle classi	Lista di tutte le classi testabili disponibili nell'applicazione	Lista di classi
Scaricare	Download	Trasferire un file o un insieme di file da un server remoto al proprio dispositivo	Azione
Inserire	Aggiungere	Aggiungere una nuova classe al sistema	Azione
Cancellare	Rimuovere	Rimuovere una classe esistente dal sistema	Azione
Criteri di ordinamento	Ordinamento	Opzioni per ordinare le classi in base a diversi attributi	Azione
Etichette	Tag, Categorie	Le tag o le categorie assegnate alle classi per facilitare la ricerca e il filtraggio	Attributo della classe
Filtrare	Ridurre	Ridurre l'elenco di classi visualizzate in base alle etichette selezionate dall'utente	Azione
Modificare	Modifiche	Apportare modifiche a una classe esistente nel sistema	Azione
Interfaccia utente	UI	Il layout e l'aspetto visivo dell'applicazione che l'utente vede e con cui interagisce	Elemento grafico
Ricerca	Cercare	Cercare una classe specifica nel sistema	Azione
Barra di ricerca	Campo di ricerca	Area dell'interfaccia utente in cui l'utente può inserire testo per cercare un elemento specifico all'interno dell'applicazione	Elemento grafico dell'interfaccia utente
Difficoltà	Complessità	Misura della complessità del codice di una classe, valutata in base al numero di righe di codice, al numero di dipendenze e ad altre metriche	Numero, Valore numerico

2.3 Requisiti Funzionali

I requisiti funzionali previsti per la web app sono i seguenti:

- L'applicazione deve mostrare un elenco di tutte le classi.
- L'applicazione deve consentire all'amministratore e allo studente di scaricare il codice di una classe selezionata.
- L'applicazione deve consentire all'amministratore di inserire una nuova classe con nome, descrizione e codice.

- L'applicazione deve consentire all'amministratore di cancellare una classe esistente selezionata.
- L'applicazione deve consentire allo studente di cercare una classe per nome, attraverso un campo di ricerca.
- L'applicazione deve consentire allo studente di scegliere e cambiare i criteri di ordinamento delle classi visualizzate.
- L'applicazione deve consentire allo studente di selezionare una o più etichette per filtrare le classi visualizzate.
- L'applicazione deve consentire all'amministratore di modificare una classe esistente, inclusi nome, descrizione e codice.

2.4 Storie utente

STORIA	STIMA TEMPORALE	PRIORITÀ
Come utente voglio consultare le classi disponibili	1	1
Come utente voglio poter scaricare il codice di una classe	3	2
Come amministratore voglio inserire una nuova classe	3	3
Come amministratore voglio cancellare una classe esistente	2	4
Come utente voglio ricercare una classe per nome	2	5
Come utente voglio scegliere e cambiare i criteri di ordinamento delle classi.	2	6
Come utente voglio scegliere una o più etichette per filtrare le classi	3	7
Come amministratore voglio modificare una classe esistente	3	9
TOTALE	19	

***Parametri di ordinamento:** Complessità, data di ultima modifica, per livelli.

***Parametri di filtraggio:** Livello, etichetta, complessità.

2.5 Requisiti Non Funzionali

- **Usabilità:**
 - L'applicazione deve essere intuitiva e facile da usare, con un'interfaccia utente intuitiva e ben progettata.
 - Gli utenti devono poter navigare nel catalogo e utilizzare le funzionalità in modo semplice ed efficace.
- **Affidabilità:**
 - L'applicazione deve essere disponibile in modo continuo, garantendo una disponibilità del servizio pari almeno al 99,9% del tempo.
 - Gli utenti devono poter accedere al sistema senza interruzioni significative o tempi di inattività prolungati.
- **Scalabilità:**
 - L'applicazione deve essere scalabile in modo da supportare un aumento del numero di utenti e del traffico dati.
- **Sicurezza:**
 - L'applicazione deve essere sicura, proteggendo i dati degli utenti e delle classi.
- **Prestazioni:**
 - L'applicazione deve essere performante, con tempi di risposta rapidi e senza ritardi significativi.
 - Le richieste degli utenti, come la ricerca o il download di classi di programmazione, devono essere elaborate in modo efficiente e veloce.
- **Compatibilità:**
 - L'applicazione deve essere compatibile con diversi browser web e dispositivi, garantendo una buona esperienza utente su tutte le piattaforme.
- **Manutenibilità:**
 - L'applicazione deve essere facilmente manutenibile e aggiornabile, con un codice ben strutturato e documentato.
 - I futuri sviluppatori o amministratori devono essere in grado di comprendere e modificare il codice senza difficoltà, garantendo la scalabilità e la gestione continua dell'applicazione.

2.6 Visione ad alto livello del sistema

In figura (2.1) si presenta una vista semplice e chiara dall'esterno del sistema software, mostrando le User Interface con le quali le entità esterne interagiscono e i servizi esterni che vengono utilizzati. Nel sistema è previsto che tramite il browser si possa accedere alle seguenti interfacce utente:

- **Home_adminUI:** Questa interfaccia è dedicata agli amministratori del sistema, mostra un elenco delle classi disponibili. Possono cancellare e modificare le classi e fornisce un accesso ai report inviati dagli utenti.

- **HomeUI:** Questa interfaccia è dedicata agli studenti del sistema, mostra un elenco delle classi disponibili in una visualizzazione adatta all'utente. Hanno la possibilità di mettere "Mi piace" ad una classe e inviare un report.
- **ReportsUI:** Questa interfaccia è dedicata agli amministratori del sistema, mostra un elenco dei report inviati dagli utenti riguardanti le classi. Gli amministratori possono visualizzare i dettagli di ogni report, come la classe segnalata e il messaggio associato.
- **Report_classeUI:** Questa interfaccia è dedicata agli studenti e consente di inviare un report riguardante una classe specifica, gli utenti possono inserire un messaggio che descrive il problema o l'errore riscontrato nella classe.
- **Upload_classeUI:** Questa interfaccia è dedicata agli amministratori e consente di inserire una nuova classe e fare l'upload del file .java relativo.
- **Modifica_classeUI:** Questa interfaccia è dedicata agli amministratori e consente di modificare i dati di una classe esistente.

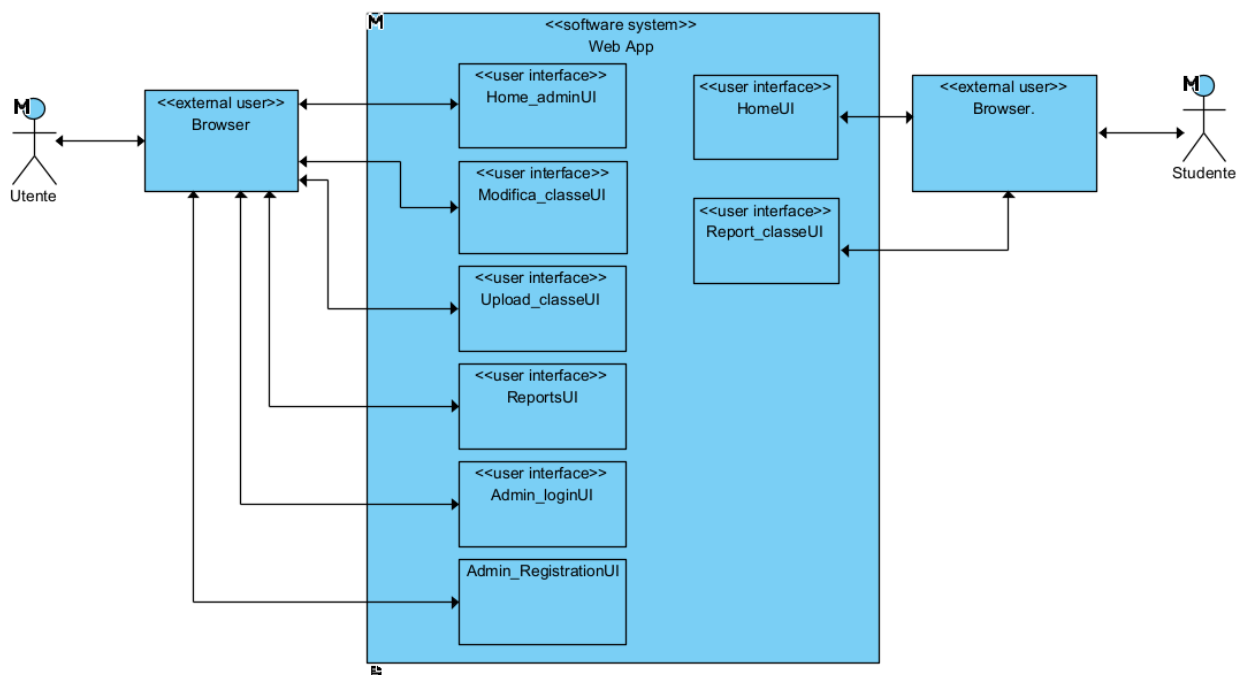


Figura 2.1: Context Diagram

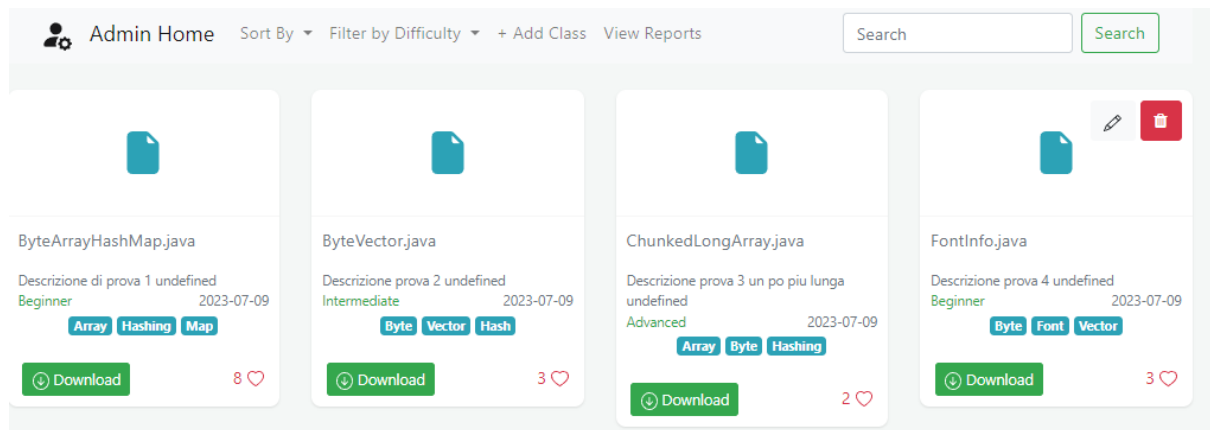


Figura 2.2: Home_adminUI

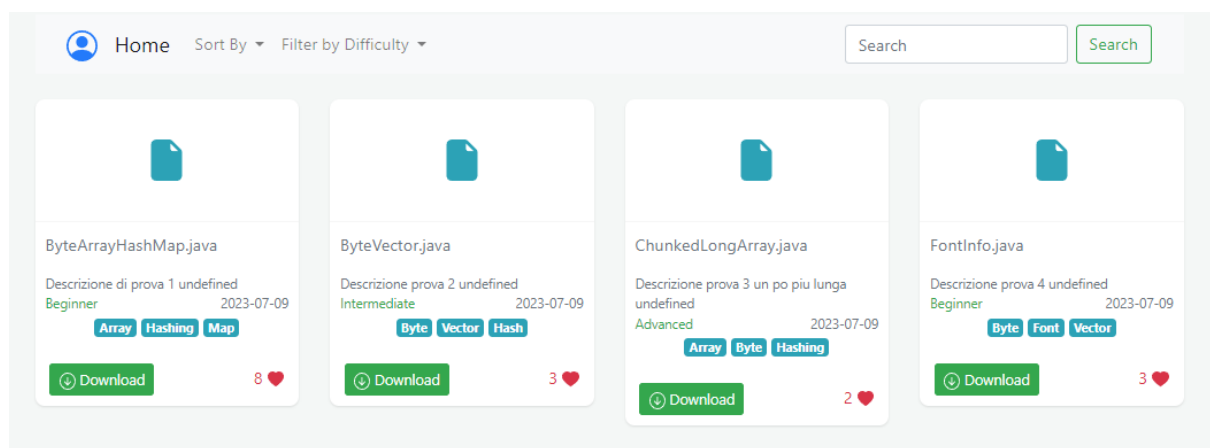


Figura 2.3: HomeUI

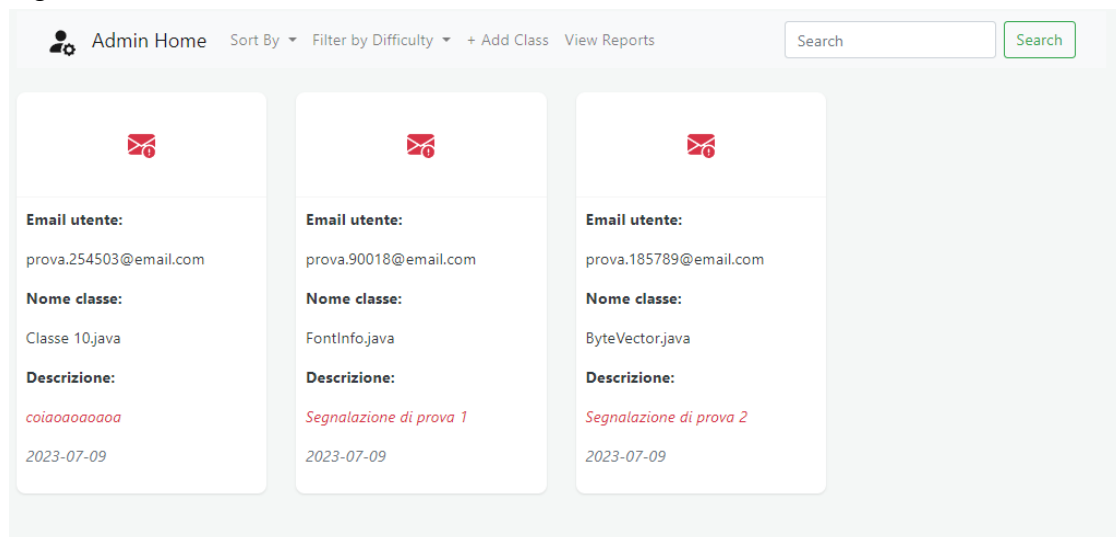



Figura 2.4: ReportsUI

Report ByteVector




Comment

Invia

Figura 2.5: Report_classeUI

Class upload



Class name

Date

gg/mm/aaaa

Difficulty

Beginner

Select the difficulty level of this item

Description

Category 1

Category 2

Category 3

Upload your file (.java) :


Scegli file

Nessun file selezionato

Upload

Figura 2.6: Upload_classeUI

Modifica Classe: ByteArrayHashMap




Class name

ByteArrayHashMap


Date

09/07/2023



Difficulty


Beginner



Select the difficulty level of this item

Description

Descrizione di prova modificata



Category 1

Array

Category 2

Hashing


Category 3

Map

Upload

Figura 2.7: Modifica_classeUI

Login Amministratore

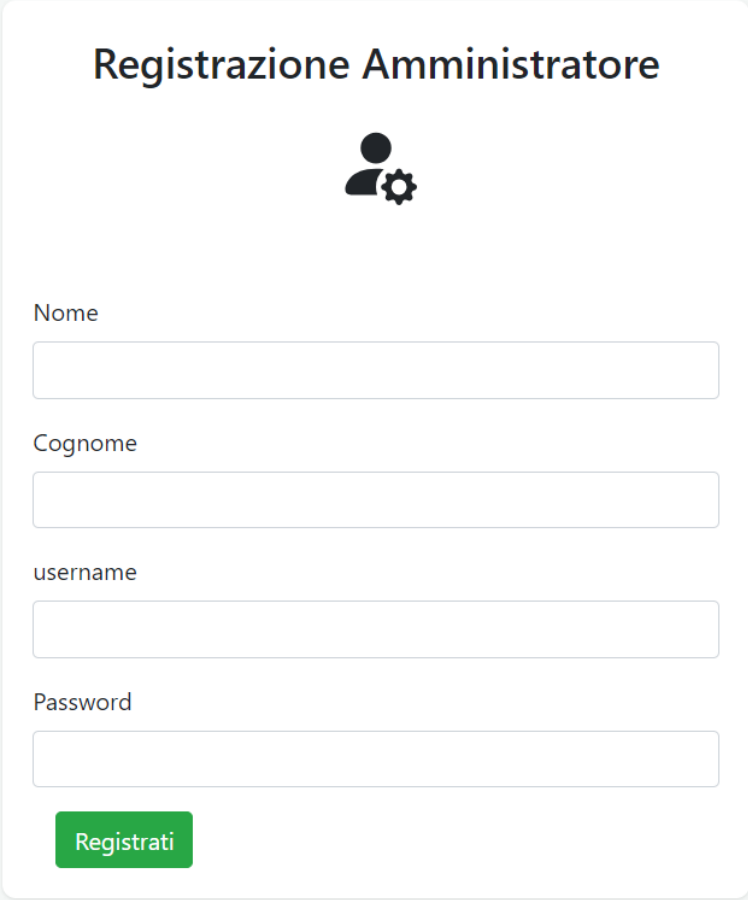


Username

Password

Login

Figura 2.8: Admin_loginUI



The image shows a registration form titled "Registrazione Amministratore". At the top, there is a black icon of a person with a gear. Below the icon, there are four input fields labeled "Nome", "Cognome", "username", and "Password". At the bottom of the form, there is a green button labeled "Registrati".

Figura 2.8: Registration_adminUI

Capitolo 3: Processo di Sviluppo

3.1 Strumenti Software di Supporto

Per il progetto di sviluppo del software, sono stati utilizzati i seguenti strumenti software di supporto:

- **Strumenti per la comunicazione del team:** si è utilizzato Discord e Microsoft Teams per la comunicazione del team. Questi strumenti hanno permesso di comunicare in tempo reale, scambiare informazioni e risolvere problemi in modo collaborativo. È stata utilizzata anche la funzione di messaggistica istantanea e la videoconferenza per facilitare la collaborazione a distanza.
- **Strumenti per la condivisione di file e documenti:** si è utilizzato Google Drive per la condivisione di file e documenti. Questo strumento ha permesso di accedere ai file in modo rapido e semplice da qualsiasi dispositivo e di lavorare contemporaneamente su di essi in tempo reale. Sono state utilizzate

le funzioni di condivisione e commento per facilitare la collaborazione tra i membri del team.

- **Strumenti per la modellazione dei diagrammi:** si è utilizzato diagrams.net e Visual Paradigm per la modellazione dei diagrammi. Questi strumenti hanno permesso di creare diagrammi UML e di altri tipi, di visualizzare la struttura del software e di identificare i requisiti del sistema. Sono state utilizzate anche le funzioni di condivisione e collaborazione per lavorare contemporaneamente sui diagrammi.
- **Strumenti per l'implementazione del software:** si è utilizzato Eclipse e Visual Studio Code per l'implementazione del software. Questi strumenti hanno permesso di scrivere e testare il codice, gestire le dipendenze e compilare l'applicazione. È stata utilizzata anche la funzione di debugging per identificare e risolvere i problemi di codice.
- **Strumenti di sviluppo e gestione del progetto:** Git è stato utilizzato come sistema di controllo versione per il progetto. GitHub è stato utilizzato come piattaforma di hosting del progetto, fornendo uno spazio di archiviazione per il codice e uno strumento per la collaborazione tra i membri del team. JIRA è stato utilizzato come strumento di gestione del progetto, fornendo una piattaforma per la pianificazione, il monitoraggio e la gestione delle attività.

In sintesi, sono stati utilizzati una combinazione di strumenti software di supporto per facilitare la comunicazione del team, la condivisione di file e documenti, la modellazione dei diagrammi e l'implementazione del software. Questi strumenti hanno permesso di lavorare in modo collaborativo e di gestire il progetto in modo efficiente e produttivo.

3.2 Analisi delle Tecnologie Software

- **Linguaggio di programmazione per il backend:** Java è stato scelto come linguaggio di programmazione per lo sviluppo del backend del software.
- **Framework per il backend:** Spring è stato scelto come framework per Java per lo sviluppo del backend del software. Spring è un framework flessibile e modulare che fornisce un'ampia gamma di funzionalità per lo sviluppo di applicazioni web. Grazie alla sua architettura modulare, Spring consente di selezionare solo i moduli necessari per l'applicazione e di integrarli facilmente con altri framework e librerie.
- **Database:** MongoDB è stato scelto come database per l'archiviazione dei dati del software. MongoDB è un database NoSQL orientato ai documenti, noto per la sua scalabilità, la flessibilità del modello dei dati e la facilità di utilizzo.

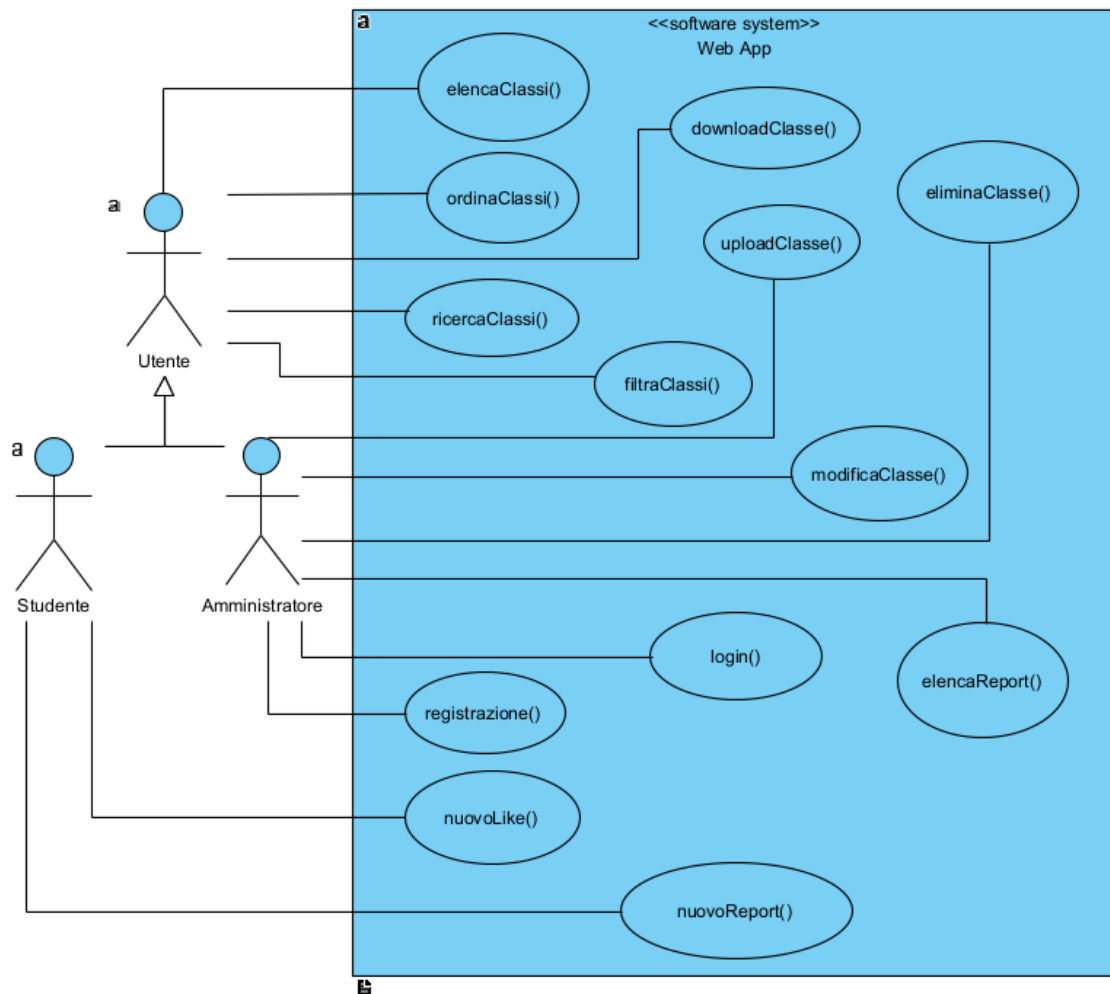
Grazie alla sua architettura distribuita, MongoDB è una scelta comune per lo sviluppo di applicazioni web scalabili e ad alte prestazioni.

- **Linguaggi di markup, stili e scripting per il front end:** HTML, CSS e JavaScript sono stati scelti come linguaggi di markup, stili e scripting per lo sviluppo del front end del software. HTML è il linguaggio di markup standard per la creazione di pagine web, mentre CSS è il linguaggio di stile utilizzato per la formattazione del contenuto HTML. JavaScript è il linguaggio di scripting utilizzato per interagire con gli elementi HTML e per aggiungere funzionalità dinamiche alle pagine web.
- **Framework per il front end:** Bootstrap è stato scelto come framework per il front end del software. Bootstrap è un framework CSS e JavaScript che fornisce un set di strumenti per lo sviluppo di interfacce utente reattive e mobile-friendly. Grazie alla sua architettura modulare e alla sua flessibilità, Bootstrap consente di creare rapidamente interfacce utente accattivanti e funzionali.

Capitolo 4: Fase di analisi

Il presente capitolo si concentra sulla documentazione di analisi prodotta dal team al fine di chiarire le funzionalità e i servizi all'interno del sistema. Verranno presentati i diagrammi realizzati utilizzando il tool Visual Paradigm, che adotta la notazione UML (Unified Modeling Language).

4.1 Use Case Diagram



4.2 Sequence Diagram

Si riportano i sequence diagram applicati in fase di analisi utilizzati per descrivere il comportamento del sistema. Tali diagrammi si concentrano sulla descrizione del comportamento del sistema, mostrando le interazioni tra attori e sistema nel tempo.

4.2.1 Sequence Diagram: RicercaClassi

- **Nome del caso d'uso:** Ricerca delle classi
- **Portata:** File Manager Man vs automated Testing Tools challenges
- **Livello:** Obiettivo Utente
- **Attore Primario:** Utente
- **Parti interessate:**
 - Utente (vuole ricercare una classe per nome all'interno del sistema)
- **Pre-condizioni:** L'utente è autenticato ed identificato

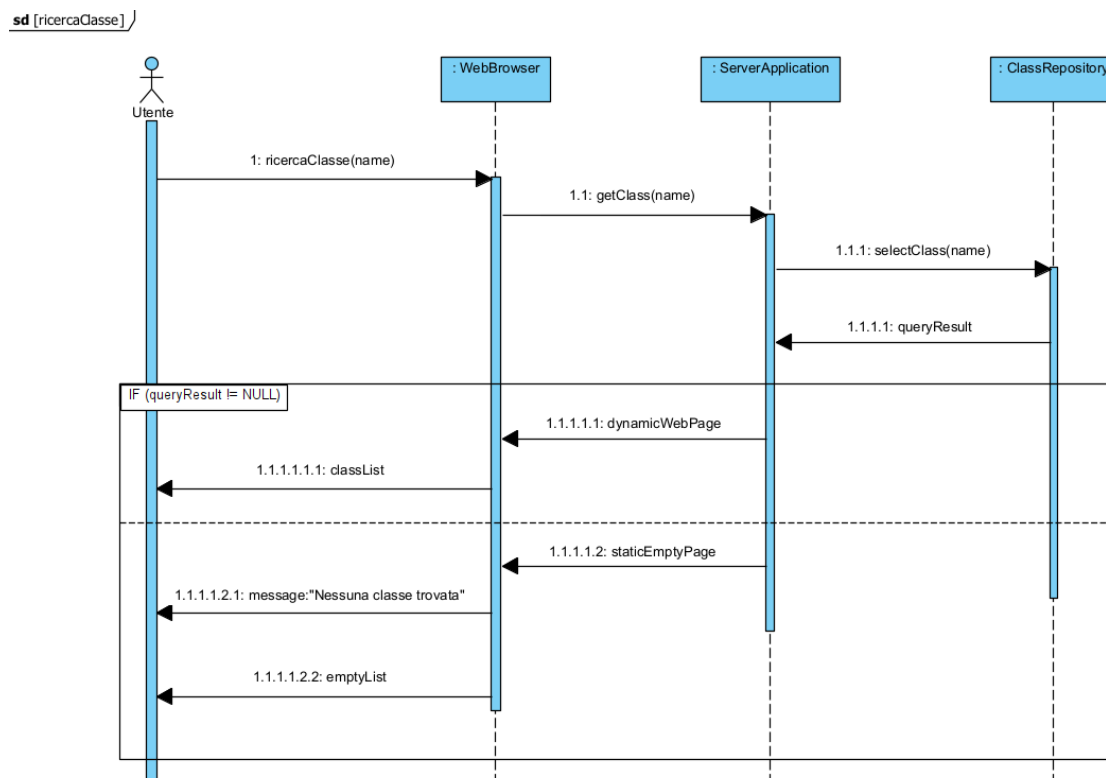
- **Garanzia di Successo:** Le classi presenti sono mostrate con successo e sono selezionabili

- **Scenario Principale di successo (Flusso Base)**

1. L'utente chiede di ricercare una classe
2. Il sistema richiede di inserire il nome della classe
3. L'utente inserisce il nome della classe e da la conferma
4. Il sistema mostra la/le classe/i corrispondenti

Flussi alternativi:

- 4.a: Non è presente nessuna classe corrispondente al nome ricercato
 1. Il sistema mostra un elenco vuoto



4.2.2 Sequence Diagram: elencaClassi

- **Nome del caso d'uso:** Elenco delle classi

- **Portata:** File Manager Man vs automated Testing Tools challenges

- **Livello:** Obiettivo Utente

- **Attore Primario:** Utente

- **Parti interessate:**

- Utente (vuole visualizzare tutte le classi all'interno del sistema)

- **Pre-condizioni:** L'utente è autenticato ed identificato

- **Garanzia di Successo:** Le classi presenti sono mostrate con successo e sono selezionabili

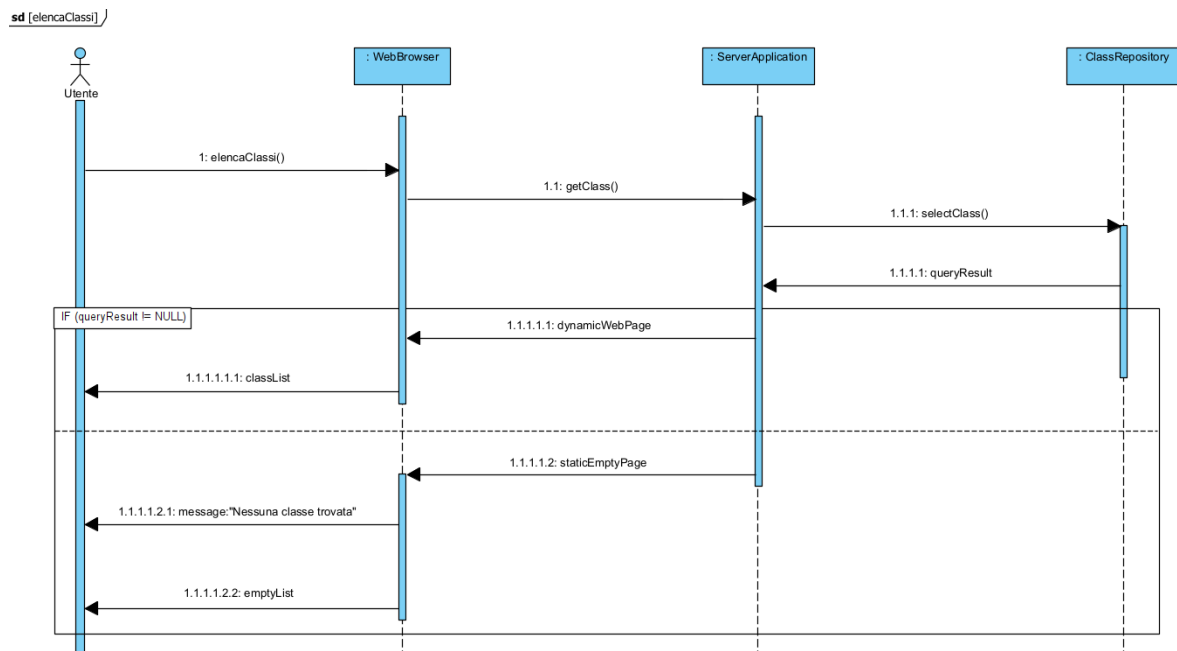
- **Scenario Principale di successo (Flusso Base)**

1. L'utente chiede di visualizzare le classi
2. Il sistema mostra la/le classe/i corrispondenti

Flussi alternativi:

2.a: Non è presente nessuna classe nel sistema

1. Il sistema mostra un elenco vuoto



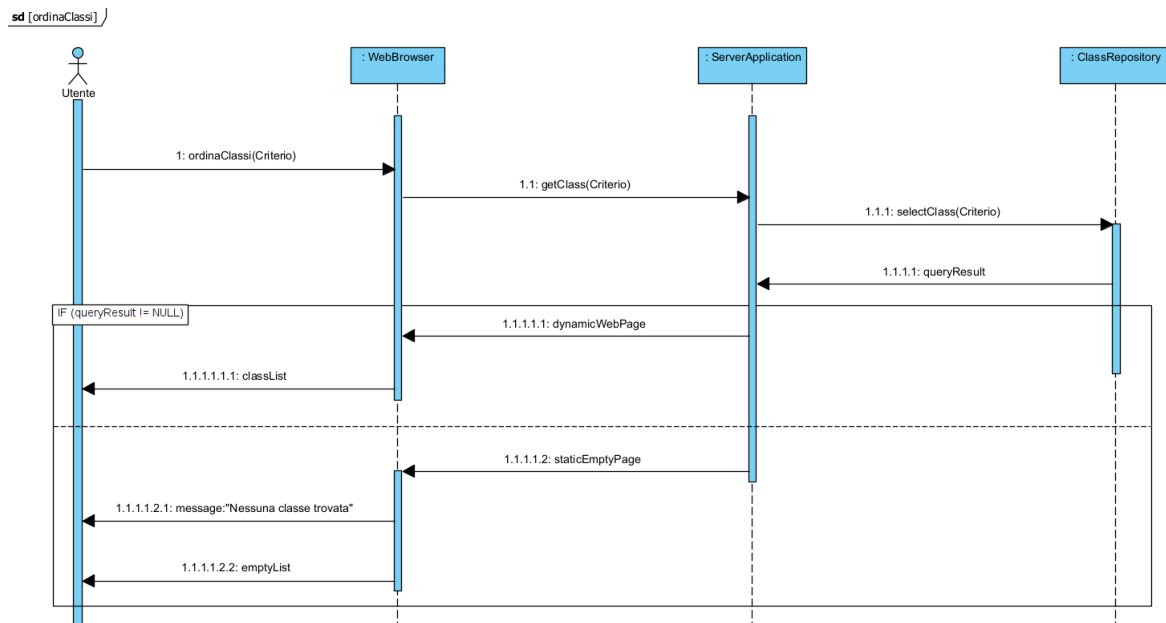
4.2.3 Sequence Diagram: ordinaClassi

- **Nome del caso d'uso:** Ordinamento delle classi
- **Portata:** File Manager Man vs automated Testing Tools challenges
- **Livello:** Obiettivo Utente
- **Attore Primario:** Utente
- **Parti interessate:**
 - Utente (vuole ordinare le classi all'interno del sistema in base ad un criterio scelto)
- **Pre-condizioni:** L'utente è autenticato ed identificato
- **Garanzia di Successo:** Le classi presenti sono mostrate con successo e sono selezionabili
- **Scenario Principale di successo (Flusso Base)**
 1. L'utente chiede di ordinare le classi del sistema
 2. Il sistema richiede di inserire il criterio di ordinamento delle classi
 3. L'utente inserisce il criterio di ordinamento delle classi e da la conferma
 4. Il sistema mostra la/le classe/i corrispondenti

Flussi alternativi:

2.a: Non è presente nessuna classe nel sistema

1. Il sistema mostra un elenco vuoto

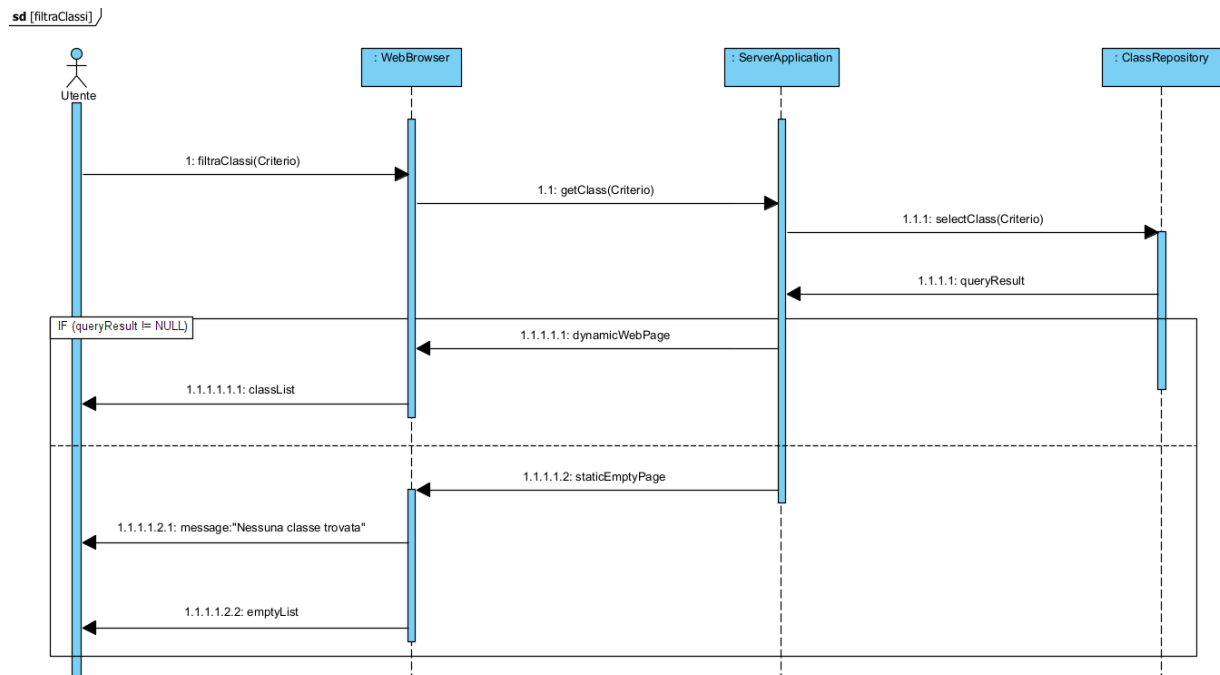


4.2.4 Sequence Diagram: filtraClassi

- **Nome del caso d'uso:** Filtraggio delle classi
- **Portata:** File Manager Man vs automated Testing Tools challenges
- **Livello:** Obiettivo Utente
- **Attore Primario:** Utente
- **Parti interessate:**
 - Utente (vuole filtrare le classi all'interno del sistema in base ad un criterio scelto)
- **Pre-condizioni:** L'utente è autenticato ed identificato
- **Garanzia di Successo:** Le classi presenti sono mostrate con successo e sono selezionabili
- **Scenario Principale di successo (Flusso Base)**
 1. L'utente chiede di filtrare le classi del sistema
 2. Il sistema richiede di inserire il criterio di filtraggio delle classi
 3. L'utente inserisce il criterio di filtraggio delle classi e dà la conferma
 4. Il sistema mostra la/le classe/i corrispondenti

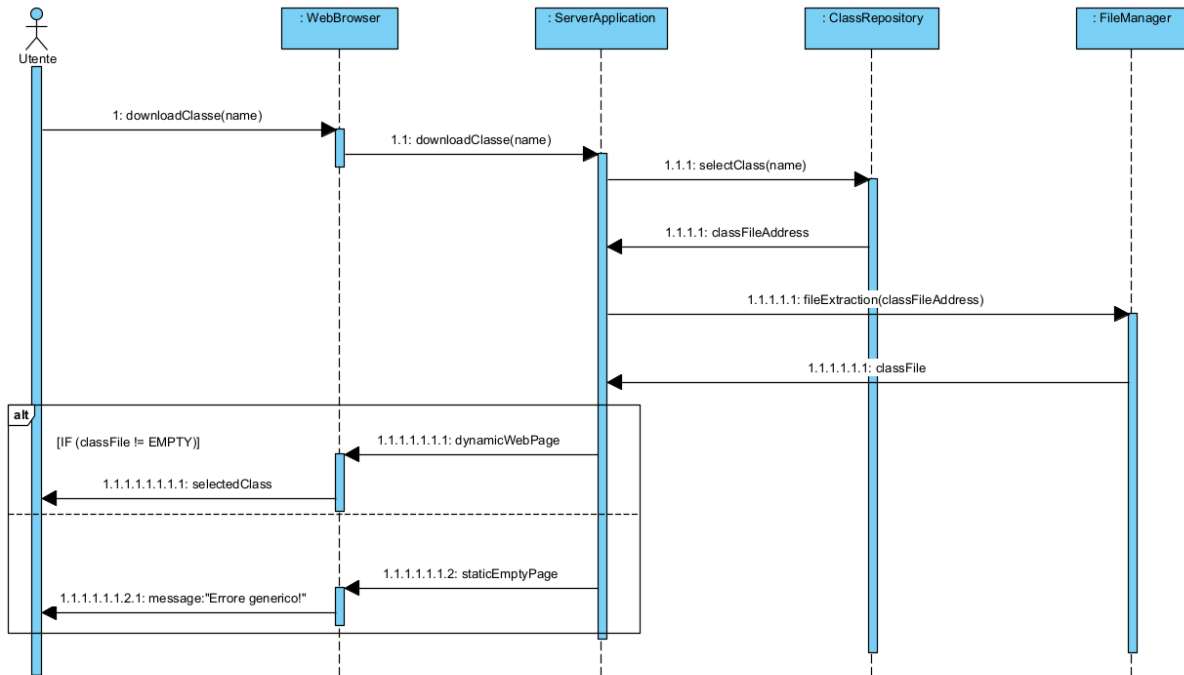
Flussi alternativi:

- 3.a: Non è presente nessuna classe corrispondente al nome ricercato
 1. Il sistema mostra un elenco vuoto



4.2.5 Sequence Diagram: DownloadClasse

- **Nome del Caso d'uso** : Download di una classe
- **Portata**: File Manager Man vs automated Testing Tools challenges
- **Livello**: Obiettivo Utente
- **Attore Primario**: Utente
- **Parti interessate**:
 - Utente (vuole scaricare il codice di una classe)
- **Pre-condizioni**: L'utente è autenticato ed identificato
- **Garanzia di Successo**: Il file viene scaricato con successo , il download è registrato.
- **Scenario Principale di successo (Flusso Base)**
 1. L'utente seleziona la classe da scaricare
 2. L'utente conferma di voler effettuare il download
 3. Il sistema registra il download
 4. L'utente riceve il file
 5. L'utente continua la sua navigazione
- **Flussi alternativi**:
 - 3.a: Il sistema segnala un errore nel download.
 1. Il sistema non registra il download.
 2. Il sistema chiede all'utente di riprovare

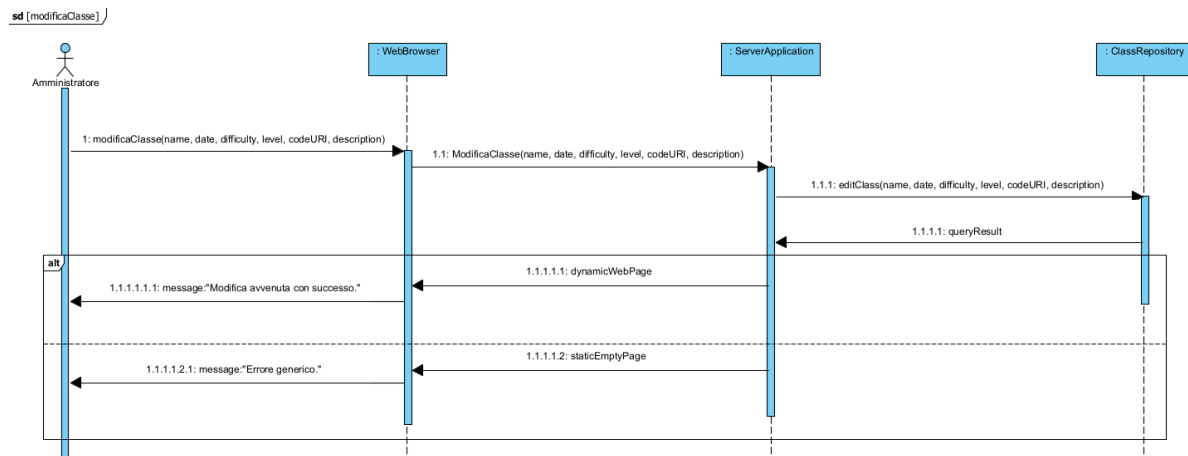


4.2.6 Sequence Diagram: ModificaClasse

- **Nome del Caso d'uso** : Modifica di una classe
- **Portata**: File Manager Man vs automated Testing Tools challenges
- **Livello**: Obiettivo Amministratore
- **Attore Primario**: Amministratore
- **Parti interessate**:
 - Amministratore (vuole modificare le informazioni di una classe)
- **Pre-condizioni**: L'amministratore è autenticato ed identificato
- **Garanzia di Successo**: La classe viene modificata con successo.
- **Scenario Principale di successo (Flusso Base)**
 1. L'amministratore seleziona la classe da modificare
 2. L'amministratore inserisce le informazioni da modificare
 3. Il sistema controlla la validità dei campi
 4. Il sistema registra le modifiche
 5. L'amministratore continua la sua navigazione

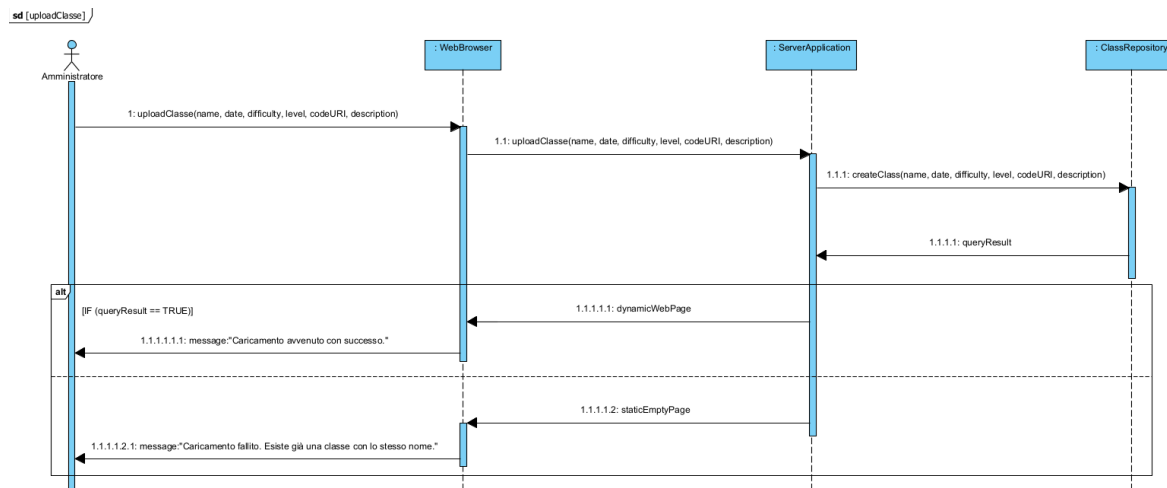
Flussi alternativi:

- 3.a: Il sistema segnala un errore nell'inserimento dei campi.
- b. Il sistema non registra la modifica..
- c. Il sistema chiede all'amministratore di riprovare



4.2.7 Sequence Diagram: UploadClasse

- **Nome del Caso d'uso** : Upload di una classe
- **Portata**: File Manager Man vs automated Testing Tools challenges
- **Livello**: Obiettivo Amministratore
- **Attore Primario**: Amministratore
- **Parti interessate**:
 - Amministratore (vuole caricare una nuova classe nel sistema)
- **Pre-condizioni**: L'amministratore è autenticato ed identificato
- **Garanzia di Successo**: La classe viene caricata con successo.
- **Scenario Principale di successo (Flusso Base)**
 1. L'amministratore chiede di caricare una classe
 2. Il sistema richiede di inserire il nome, la data, la difficoltà, la descrizione, le categorie e il file della classe
 3. L'amministratore inserisce le informazioni richieste e il file della classe e da la conferma
 4. Il sistema mostra la/le classe/i corrispondenti
- **Flussi alternativi**:
 - 3.a: Il sistema segnala un errore nell'inserimento dei campi.
 - b. Il sistema non registra l'inserimento
 - c. Il sistema chiede all'amministratore di riprovare



4.2.8 Sequence Diagram: eliminaClasse

- **Nome del Caso d'uso** : Eliminazione di una classe
- **Portata**: File Manager Man vs automated Testing Tools challenges
- **Livello**: Obiettivo Amministratore
- **Attore Primario**: Amministratore
- **Parti interessate**:
 - Amministratore (vuole eliminare una classe nel sistema)
- **Pre-condizioni**: L'amministratore è autenticato ed identificato
- **Garanzia di Successo**: La classe viene eliminata con successo.
- **Scenario Principale di successo (Flusso Base)**
 1. L'amministratore chiede di eliminare una classe
 2. Il sistema richiede la conferma di eliminazione della classe
 3. L'amministratore conferma di voler effettuare l'eliminazione
 4. Il sistema elimina la classe corrispondente

Flussi alternativi:

- 4.a: Il sistema segnala un errore generico nell'eliminazione di una classe.
- b. Il sistema non registra l'eliminazione

4.2.9 Sequence Diagram: elencaReport

- **Nome del caso d'uso**: Elenco dei report
- **Portata**: File Manager Man vs automated Testing Tools challenges
- **Livello**: Obiettivo Amministratore
- **Attore Primario**: Amministratore
- **Parti interessate**:
 - Amministratore (vuole visualizzare tutti i report all'interno del sistema)
- **Pre-condizioni**: L'amministratore è autenticato ed identificato
- **Garanzia di Successo**: I report presenti sono mostrati con successo e sono selezionabili
- **Scenario Principale di successo (Flusso Base)**
 1. L'amministratore chiede di visualizzare i report
 2. Il sistema mostra i report disponibili

Flussi alternativi:

- 2.a: Non è presente nessun report nel sistema
- b. Il sistema mostra un elenco vuoto

4.2.10 Sequence Diagram: nuovoLike

- **Nome del Caso d'uso** : Inserimento di un like ad una classe
- **Portata**: File Manager Man vs automated Testing Tools challenges
- **Livello**: Obiettivo Studente
- **Attore Primario**: Studente
- **Parti interessate**:
 - Studente (vuole esprimere un gradimento per una classe nel sistema)
- **Pre-condizioni**: Lo studente è autenticato ed identificato
- **Garanzia di Successo**: Il "Mi piace" viene salvato e incrementato per la classe scelta con successo.
- **Scenario Principale di successo (Flusso Base)**
 1. Lo studente chiede di inserire il "Mi piace" ad una classe
 2. Il sistema richiede la conferma di inserimento "Mi piace" per la classe
 3. Lo studente conferma di voler effettuare l'inserimento
 4. Il sistema incrementa il numero di "Mi piace" per la classe corrispondente

Flussi alternativi:

- 4.a: Il sistema segnala un errore generico nell'inserimento di un "Mi piace".
- b. Il sistema non registra l'inserimento

4.2.11 Sequence Diagram: nuovoReport

- **Nome del Caso d'uso** : Nuovo Report
- **Portata**: File Manager Man vs automated Testing Tools challenges
- **Livello**: Obiettivo Studente
- **Attore Primario**: Studente
- **Parti interessate**:
 - Studente (vuole inserire un nuovo report nel sistema)
- **Pre-condizioni**: Lo studente autenticato ed identificato
- **Garanzia di Successo**: Il report viene caricato con successo.
- **Scenario Principale di successo (Flusso Base)**
 1. Lo studente chiede di caricare un nuovo report
 2. Il sistema richiede di inserire la motivazione del report
 3. Lo studente inserisce la motivazione e dà la conferma
 4. Lo studente continua la propria navigazione

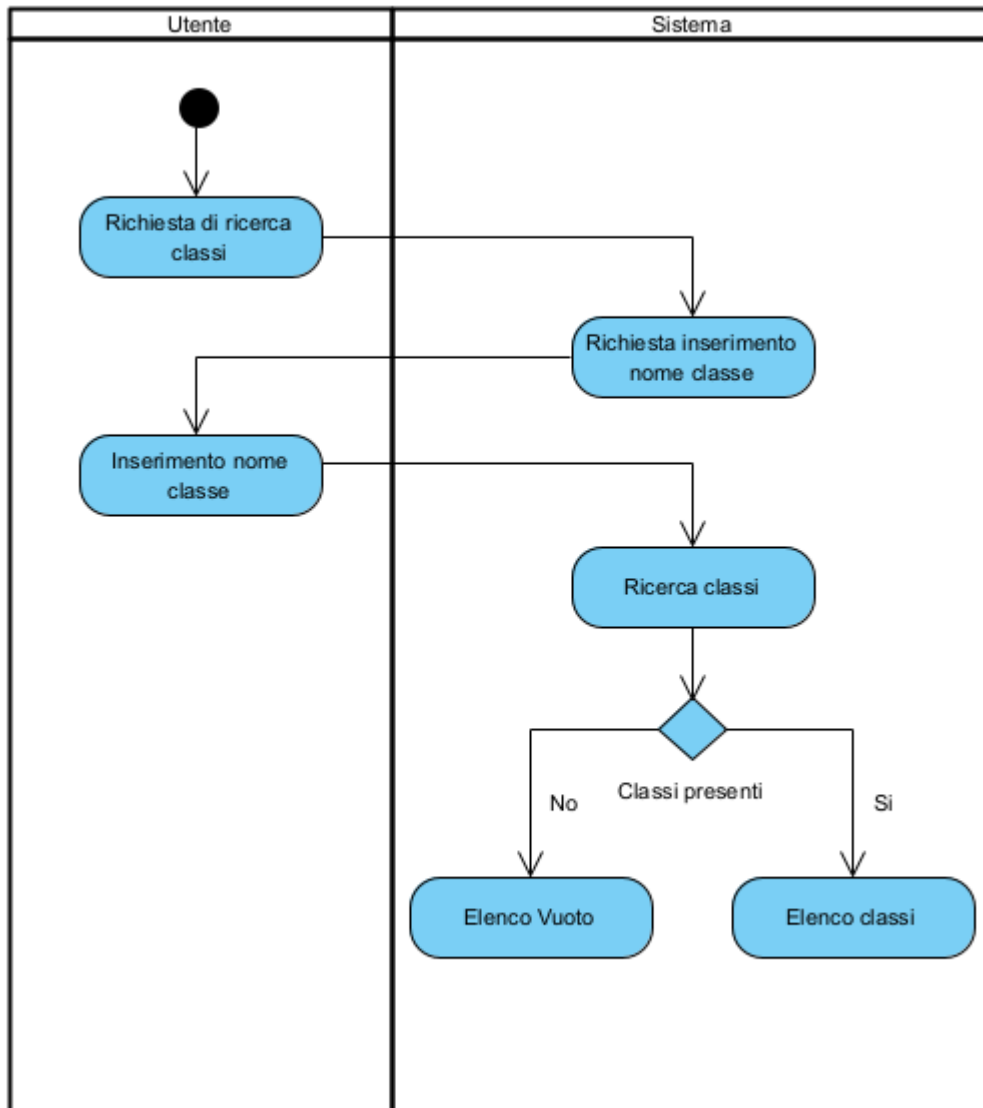
Flussi alternativi:

- 3.a: Il sistema segnala un errore generico nell'inserimento.
- b. Il sistema non registra il report
- c. Il sistema chiede allo studente di riprovare

4.3 Activity Diagram

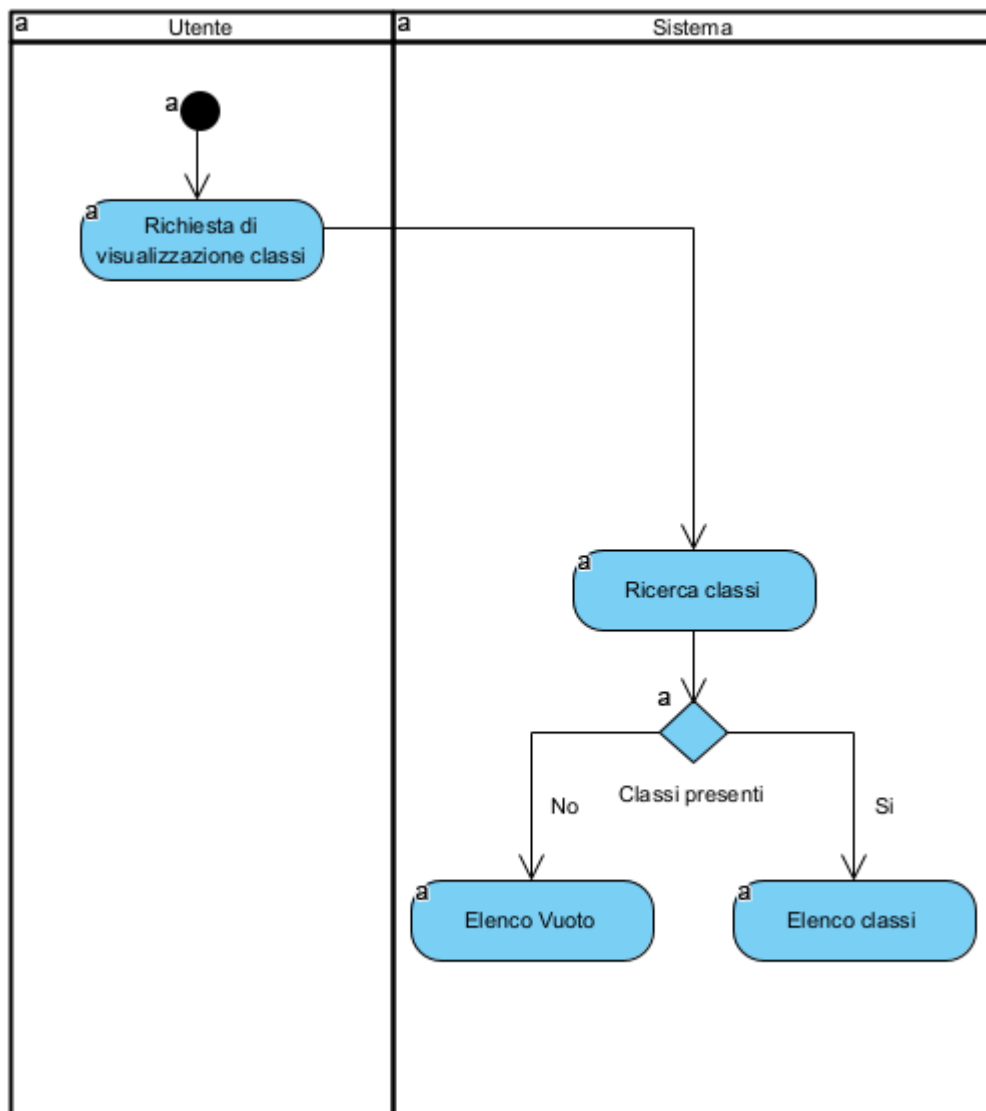
Per offrire una panoramica del flusso di attività di ciascun caso d'uso e dell'interazione tra l'attore e il sistema, sono presentati di seguito i seguenti diagrammi di attività:

4.3.1 Activity Diagram: RicercaClassi



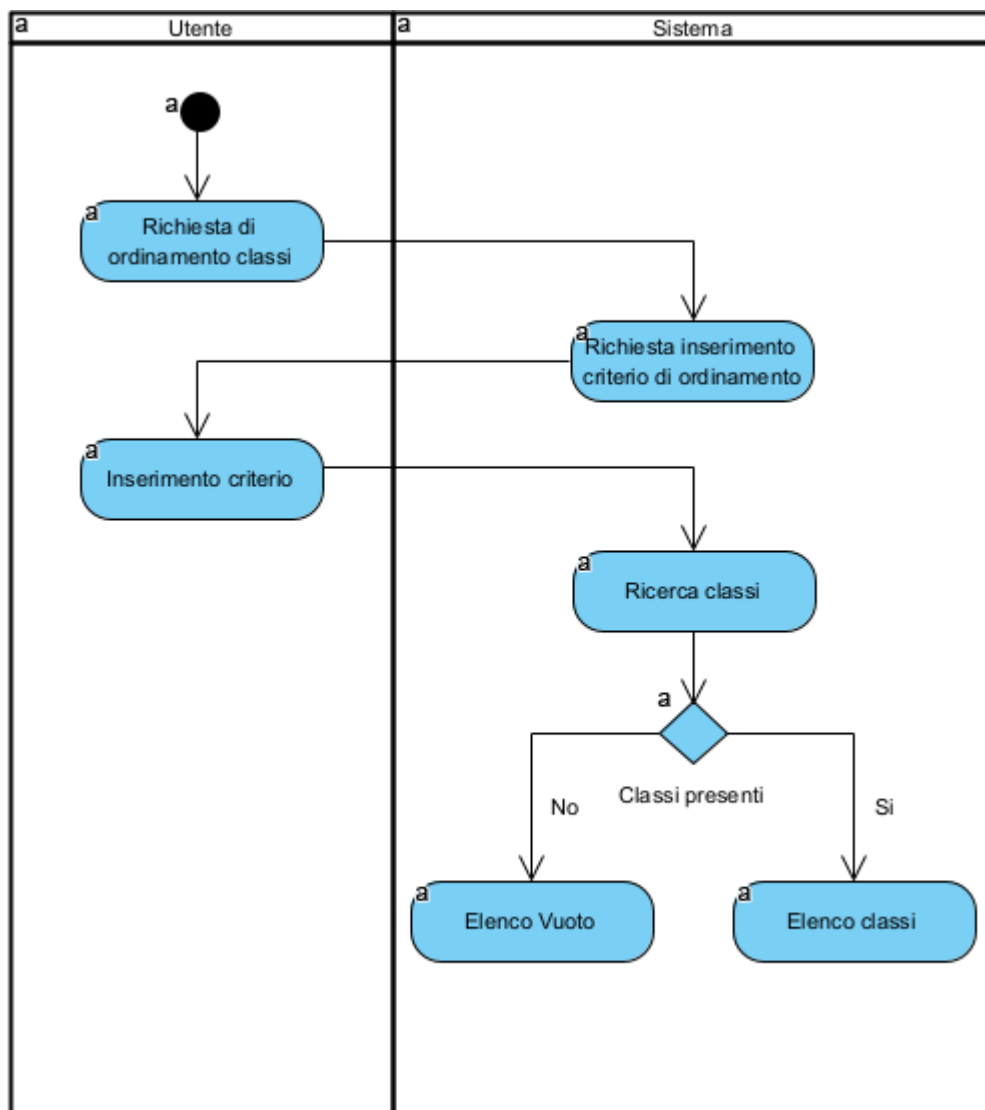
Il diagramma di attività rappresenta il flusso delle azioni coinvolte nel caso d'uso "Ricerca delle classi". Sono state definite due partizioni: la prima rappresenta l'utente che vuole effettuare la ricerca di una classe, la seconda il sistema che richiede l'inserimento del nome della classe da ricercare. Se la ricerca non riscontra risultati nel database viene mostrato un elenco vuoto altrimenti un elenco pieno.

4.3.2 Activity Diagram: elencaClassi



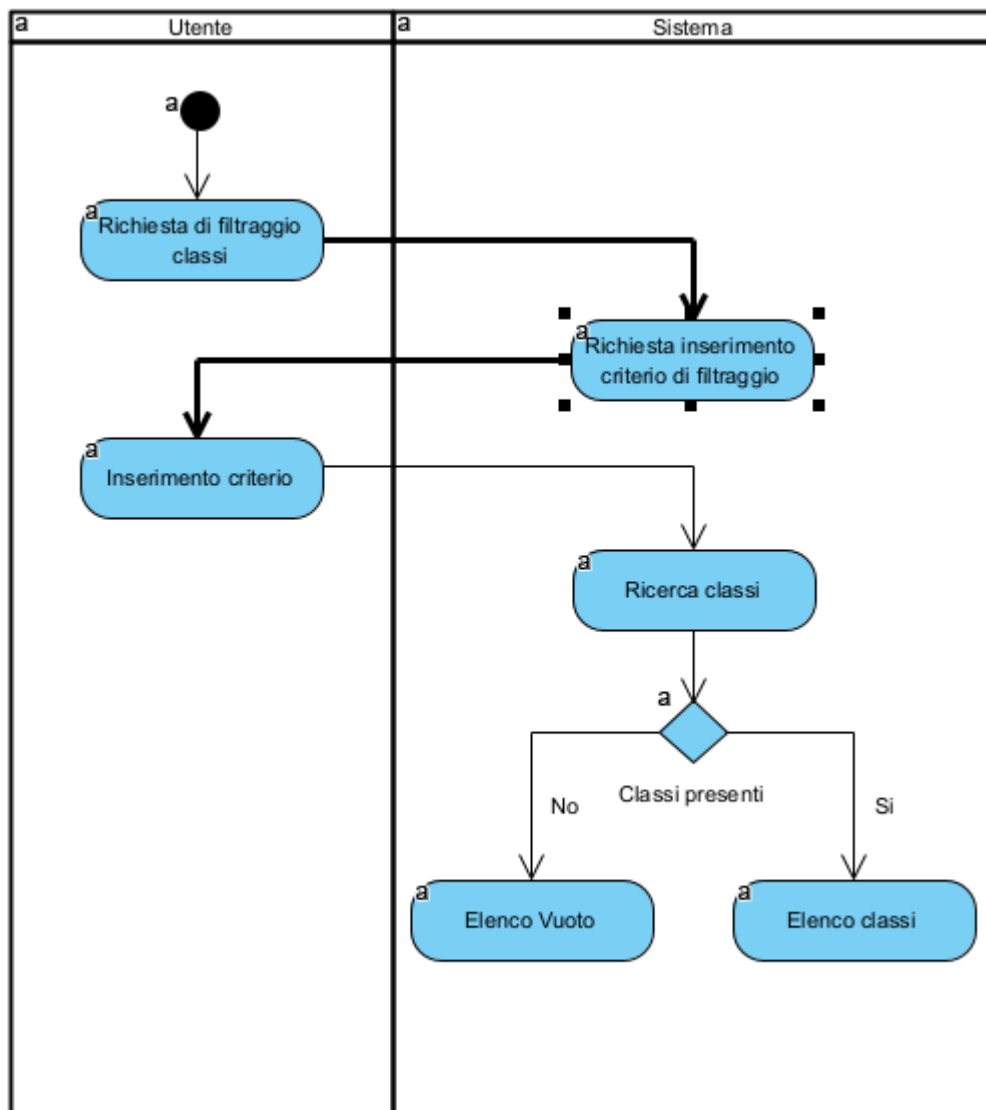
Il diagramma di attività rappresenta il flusso delle azioni coinvolte nel caso d'uso "Elenca classi". Sono state definite due partizioni: la prima rappresenta l'utente che vuole visualizzare le classi presenti nel sistema, la seconda il sistema che effettua la ricerca. Se la ricerca non riscontra risultati nel database viene mostrato un elenco vuoto altrimenti un elenco pieno.

4.3.3 Activity Diagram: ordinaClassi



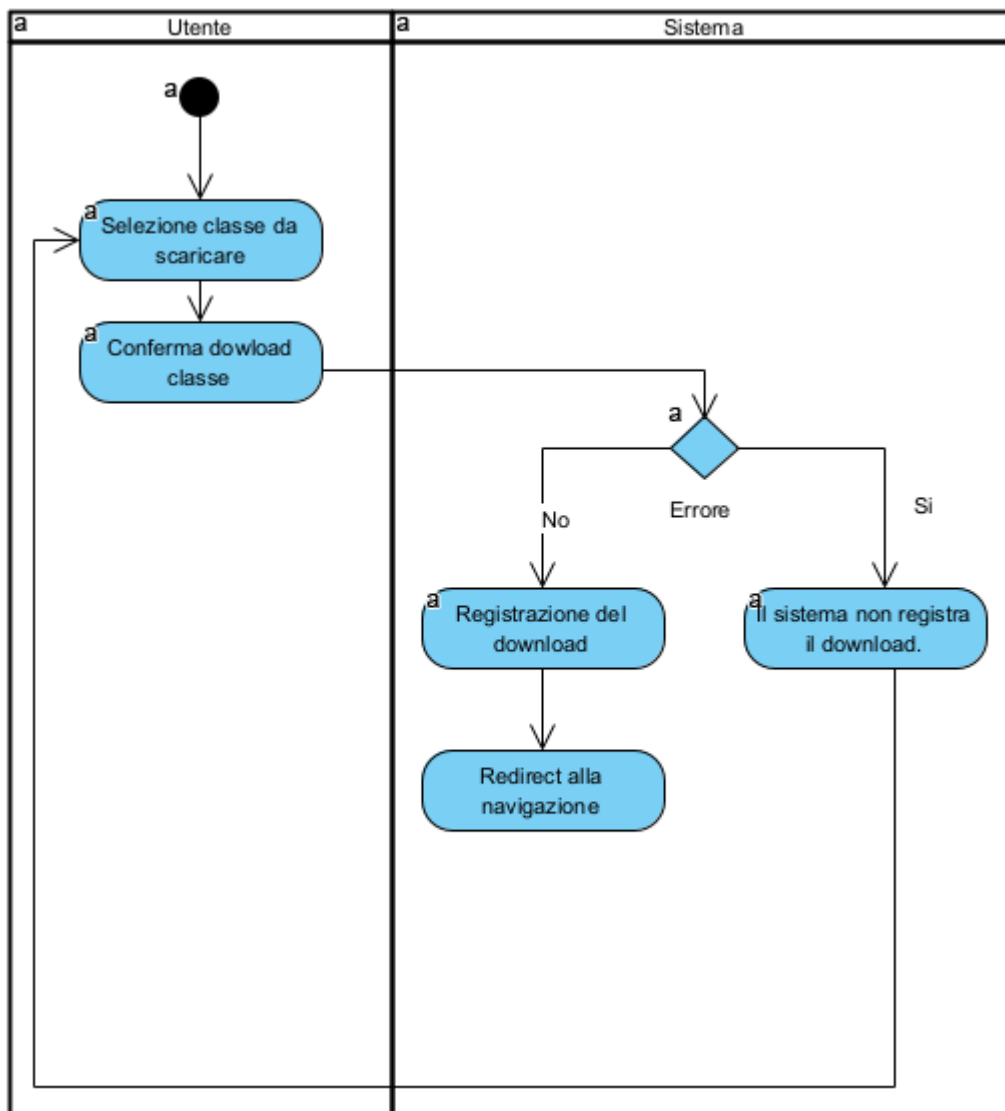
Il diagramma di attività rappresenta il flusso delle azioni coinvolte nel caso d'uso "Ordinamento delle classi". Sono state definite due partizioni: la prima rappresenta l'utente che vuole ordinare le classi da visualizzare, la seconda il sistema che richiede l'inserimento del criterio di ordinamento della classe da applicare. Se la ricerca non riscontra risultati nel database viene mostrato un elenco vuoto altrimenti un elenco pieno.

4.3.4 Activity Diagram: filtraClassi



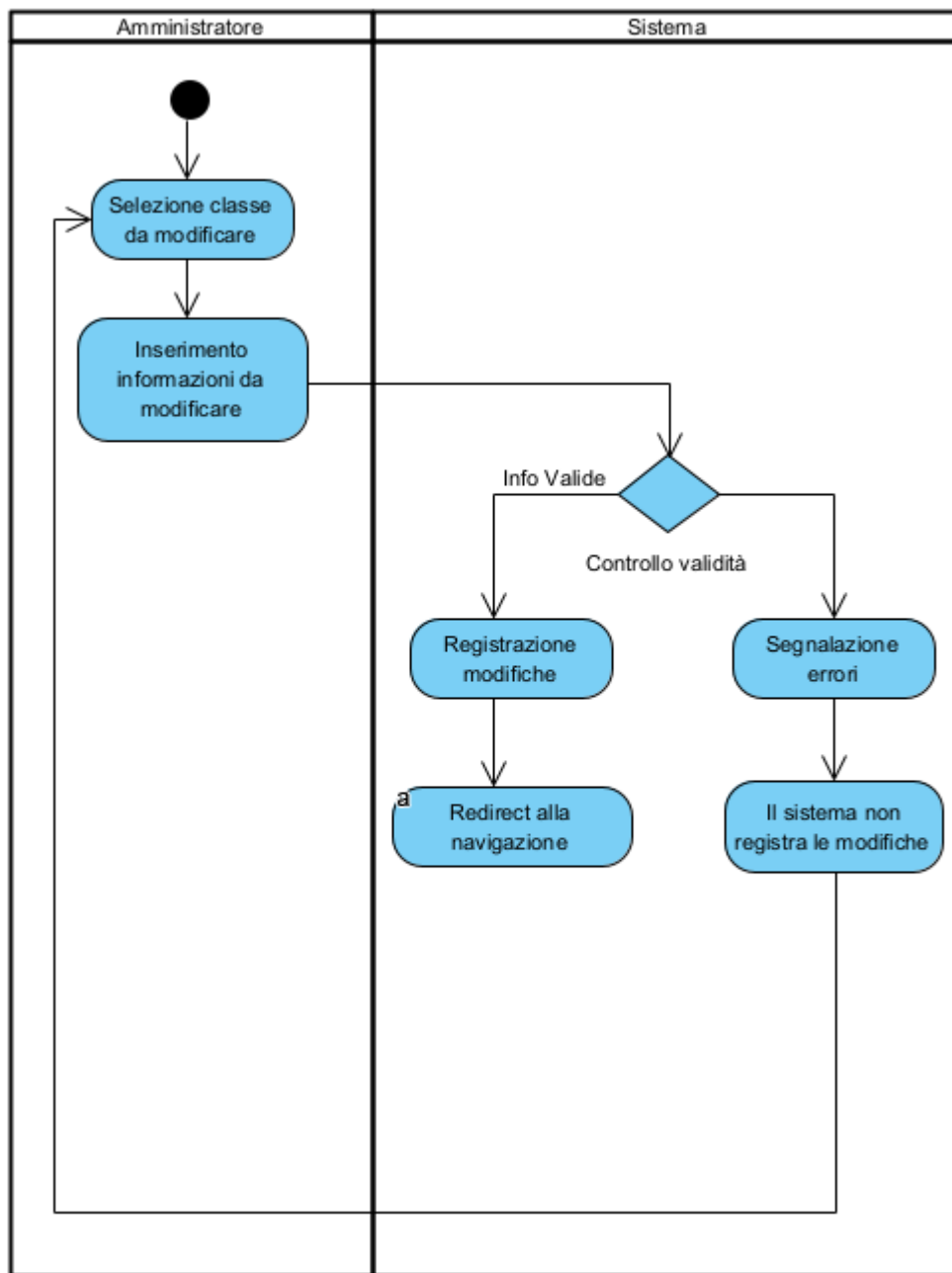
Il diagramma di attività rappresenta il flusso delle azioni coinvolte nel caso d'uso "Filtraggio delle classi". Sono state definite due partizioni: la prima rappresenta l'utente che vuole effettuare il filtraggio delle classi, la seconda il sistema che richiede l'inserimento del criterio di filtraggio della classe da applicare. Se la ricerca non riscontra risultati nel database viene mostrato un elenco vuoto altrimenti un elenco pieno.

4.3.5 Activity Diagram: DownloadClasse



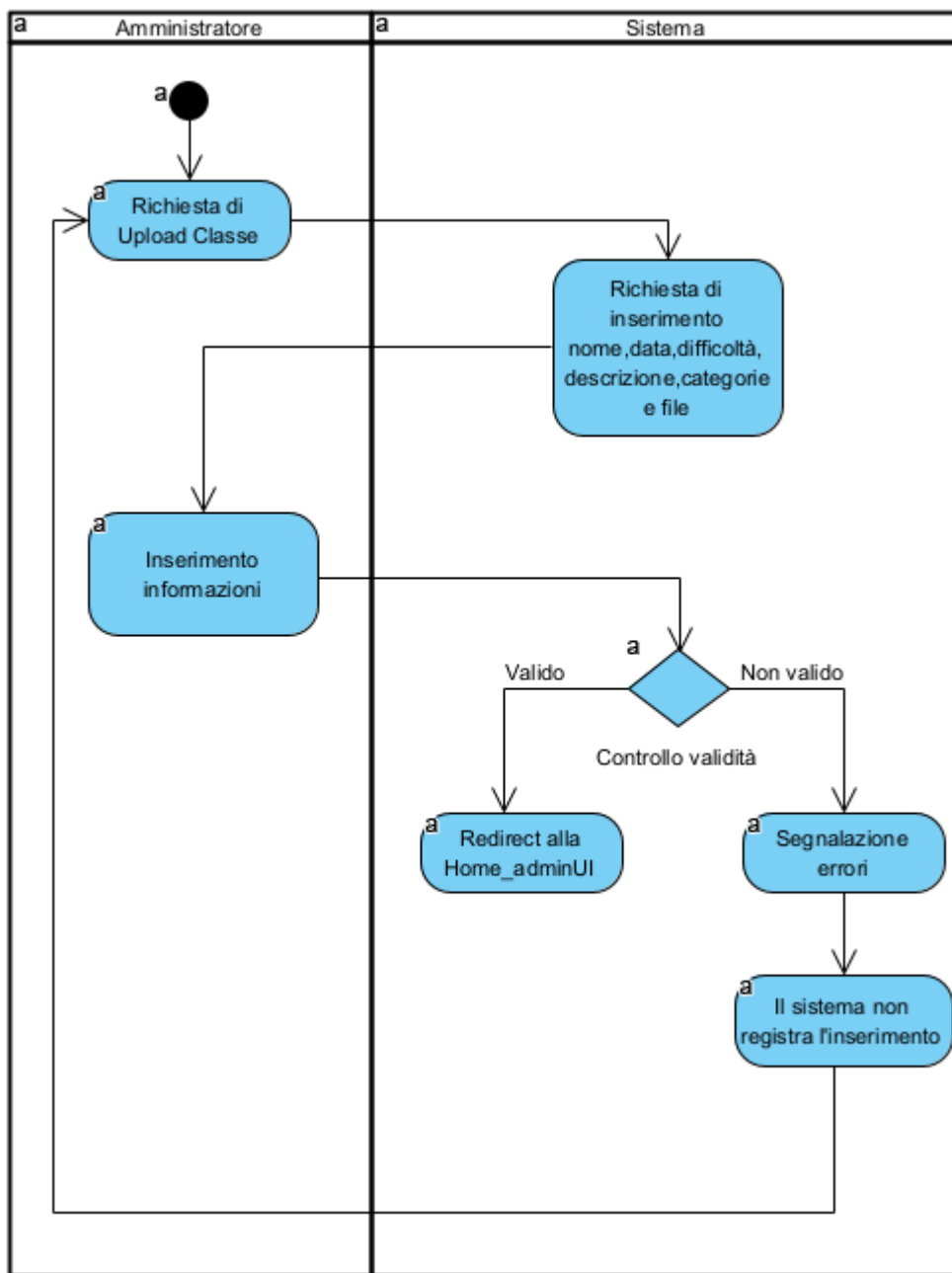
Il diagramma di attività rappresenta il flusso delle azioni coinvolte nel caso d'uso "Download di una classe". Sono state definite due partizioni: la prima rappresenta l'utente che vuole effettuare il download, la seconda il sistema che controlla se il download va a buon fine senza errori. Se dovessero esserci errori il sistema chiede all'utente di riprovare.

4.3.6 Activity Diagram: ModificaClasse



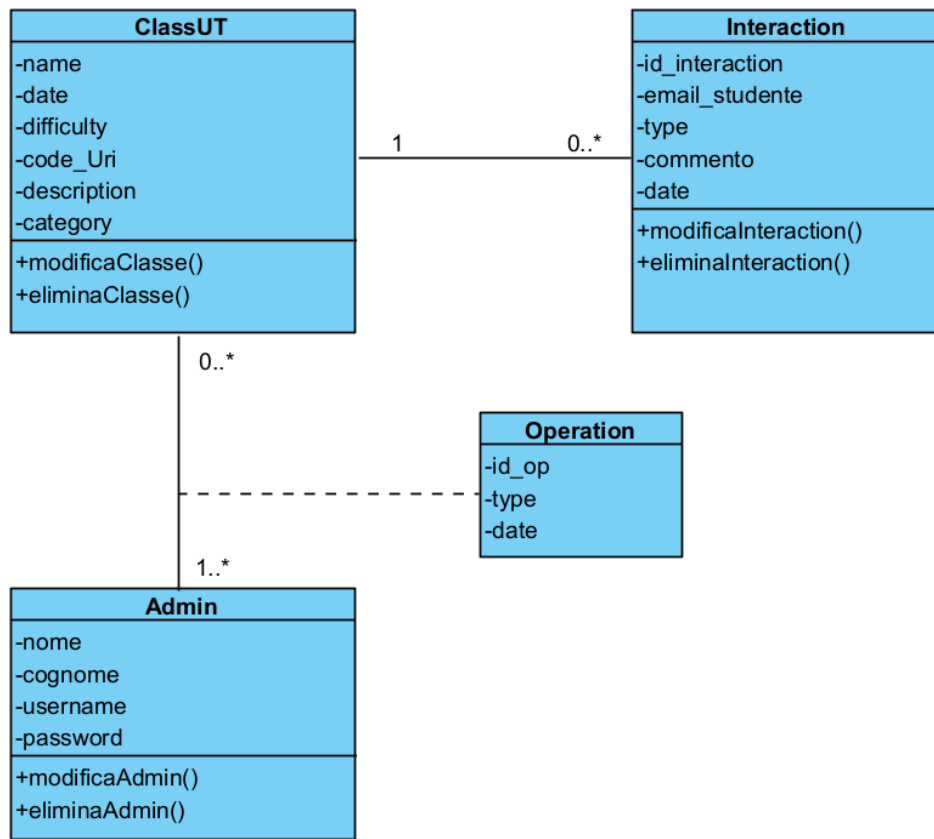
Il diagramma di attività rappresenta il flusso delle azioni coinvolte nel caso d'uso "Modifica di una classe". Sono state definite due partizioni: la prima rappresenta l'amministratore che vuole effettuare la modifica, la seconda il sistema che controlla se l'inserimento delle informazioni sono corrette. Se dovessero esserci errori il sistema chiede all'amministratore di riprovare.

4.3.7 Activity Diagram: UploadClasse



Il diagramma di attività rappresenta il flusso delle azioni coinvolte nel caso d'uso "Upload di una classe". Sono state definite due partizioni: la prima rappresenta l'amministratore che vuole effettuare l'upload, la seconda il sistema che controlla se l'inserimento delle informazioni sono corrette. Se dovessero esserci errori il sistema chiede all'amministratore di riprovare.

4.4 System Domain Model



Il System Domain Model (SDM) è un modello di dominio che rappresenta gli oggetti, i concetti e le interazioni all'interno del dominio del sistema, fornendo una visione ad alto livello del sistema software. Il SDM descrive i concetti del sistema e le relazioni tra di essi, senza specificare ancora come questi saranno implementati come classi e oggetti nel sistema software.

Nel sistema in questione, il System Domain Model (SDM) rappresenta le entità principali del sistema attraverso due classi: **ClassUT**, **Interaction**, **Operation** e **Admin**.

La classe **ClassUT** modella le classi testabili disponibili sul sistema e contiene attributi che descrivono le loro proprietà. Tra gli attributi, ci sono il nome della classe, la data di creazione, l'indirizzo del file associato, la descrizione e la categoria. Questi attributi consentono di identificare e descrivere le classi disponibili nel sistema.

La classe **ClassUT** contiene anche le responsabilità che definiscono le azioni che possono essere eseguite su di essa. Tra le responsabilità sono presenti la modifica e l'eliminazione di un'istanza di una classe.

La classe **Interaction** modella le interazioni tra studenti e classi e contiene attributi che descrivono le sue proprietà. Tra gli attributi, c'è l'attributo `type`, che si riferisce al tipo di interazione che può essere `like` o `report`. Questo attributo consente di identificare il tipo di interazione che un utente del sistema ha con una classe specifica.

La classe **Admin** modella gli amministratori del sistema e contiene attributi che descrivono le loro proprietà. Tra gli attributi, ci sono il nome, il cognome, lo username e la password. Questi attributi consentono di identificare e autenticare gli amministratori del sistema.

La classe Admin ha le responsabilità di gestire e monitorare il sistema. Tra le responsabilità ci sono la creazione, l'aggiornamento e la cancellazione di classi testabili.

Admin e ClassUT sono collegati dalla classe associativa **Operation** che tiene traccia di tutte le operazioni di inserimento, modifica e cancellazione che un admin può fare sulle classi del sistema.

Capitolo 5: Fase di Progettazione

Il seguente paragrafo illustra la fase di progettazione eseguita dal team di sviluppo, partendo dai diagrammi di analisi. La documentazione di progettazione dettagliata fornirà una visione chiara degli obiettivi e dei requisiti del sistema, delle scelte architetturali, della struttura dei componenti, dei flussi di dati e delle interazioni tra le parti del sistema.

La documentazione di progettazione dettagliata mostrata in seguito fornisce una descrizione completa del sistema, compresi i requisiti funzionali e non funzionali, le scelte architetturali, la struttura dei componenti, i flussi di dati e le interazioni tra le parti del sistema.

5.1 Pattern Architetturale MVC

Il pattern architetturale MVC (Model-View-Controller) è un approccio strutturale per progettare e sviluppare applicazioni software. Questo pattern separa logicamente le responsabilità all'interno di un'applicazione in tre componenti principali: il Modello (Model), la Vista (View) e il Controllore (Controller). Ognuno di questi componenti svolge un ruolo specifico e collabora insieme per creare un'architettura ben strutturata e scalabile.

5.1.1 Spring MVC

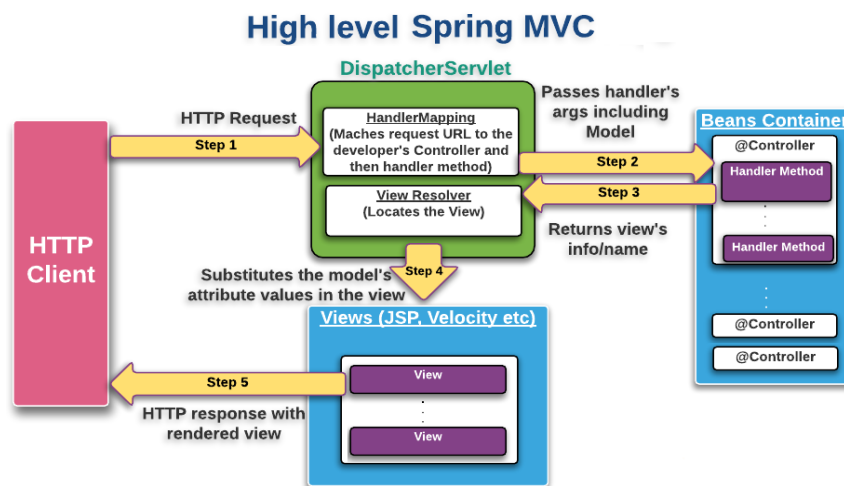
Spring MVC è un framework basato su Spring Framework che offre un'implementazione del pattern architetturale MVC per lo sviluppo di applicazioni web Java. Spring MVC semplifica la creazione di applicazioni web scalabili, modulari e mantenibili, fornendo una struttura solida per organizzare il codice e separare le responsabilità.

I vantaggi principali di Spring MVC sono:

- **Separazione delle responsabilità:** Spring MVC adotta il pattern architetturale MVC, che consente una chiara separazione delle responsabilità tra il Modello, la Vista e il Controllore. Questo favorisce la modularità e la manutenibilità del codice, facilitando la comprensione e l'evoluzione dell'applicazione nel tempo.
- **Configurazione dichiarativa:** Spring MVC utilizza un approccio basato su annotazioni o file di configurazione XML per definire il comportamento dell'applicazione. Questo consente di gestire in modo dichiarativo le

interazioni tra i componenti, riducendo la complessità e la quantità di codice da scrivere

- **Scalabilità e performance:** Spring MVC è progettato per fornire prestazioni elevate e scalabilità. Offre meccanismi per la memorizzazione nella cache, la gestione delle richieste concorrenti e l'ottimizzazione delle risorse, consentendo di costruire applicazioni web efficienti e scalabili.
- **Gestione delle richieste e delle risposte:** fornisce un sistema di gestione delle richieste e delle risposte che consente di mappare richieste HTTP a metodi specifici del controller utilizzando annotazioni come `@RequestMapping`.



5.2 Component Diagram

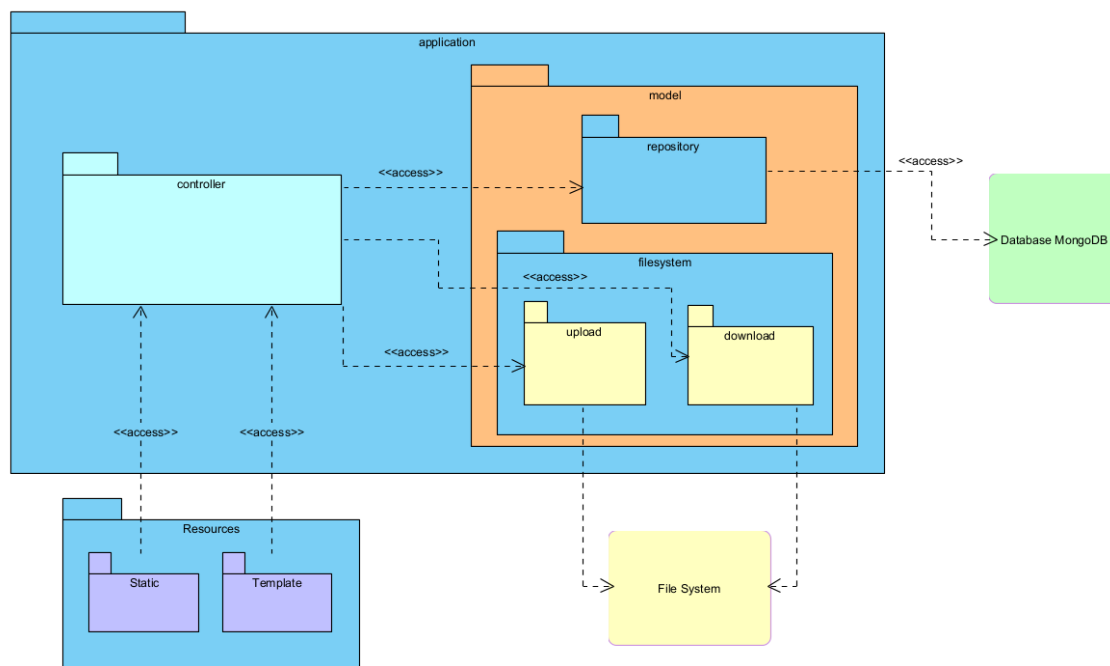
—IMMAGINE DA INSERIRE (VA MODIFICATA) —

Questo componente diagram descrive l'interazione tra i vari componenti nel sistema, mostrando le relazioni e le dipendenze tra di loro per supportare la gestione dei test, l'amministrazione del sistema e l'accesso alle informazioni correlate. I componenti di cui il sistema si compone sono:

- **UI Admin:** Questo componente rappresenta l'interfaccia utente (UI) per l'amministratore del sistema. L'UI Admin fornisce funzionalità specifiche per l'amministratore, come la gestione delle classi di programmazione, la visualizzazione dei report e la modifica o cancellazione di classi.
- **Test Manager:** Il Test Manager è responsabile della gestione dei test nel sistema. Questo componente si occupa della creazione, esecuzione e monitoraggio dei test, inclusi i test generati automaticamente. Collabora con altri componenti come l'Evo Suite Generator e il Test Repository per ottenere informazioni sulle classi di programmazione e archiviare i risultati dei test.
- **Student:** Questo componente rappresenta lo studente o l'utente del sistema. Lo studente utilizza l'interfaccia utente per accedere alle classi di programmazione, eseguire i test e visualizzare i risultati. Interagisce con il Test Manager

- **Evo Suite Generator:** L'Evo Suite Generator è responsabile della generazione automatica di suite di test. Utilizza tecniche per generare automaticamente test case e suite di test per le classi di programmazione. Fornisce i test generati al Test Manager per l'esecuzione e il monitoraggio.
- **Mongo Database:** Questo componente rappresenta un database MongoDB utilizzato per la memorizzazione dei dati nel sistema. Il Mongo Database archivia informazioni come le classi di programmazione. È accessibile da altri componenti come il Test Manager per la memorizzazione e il recupero dei dati necessari.
- **Test Repository:** Il Test Repository gestisce la memorizzazione e l'accesso ai test e ai risultati dei test nel sistema. Interagisce con il Test Manager per recuperare i test, archiviare i risultati e fornire le informazioni necessarie per l'esecuzione dei test.
- **Randoop:** Randoop è un componente che supporta la generazione automatica di test case.

5.3 Package Diagram



5.4 Application Class Diagram

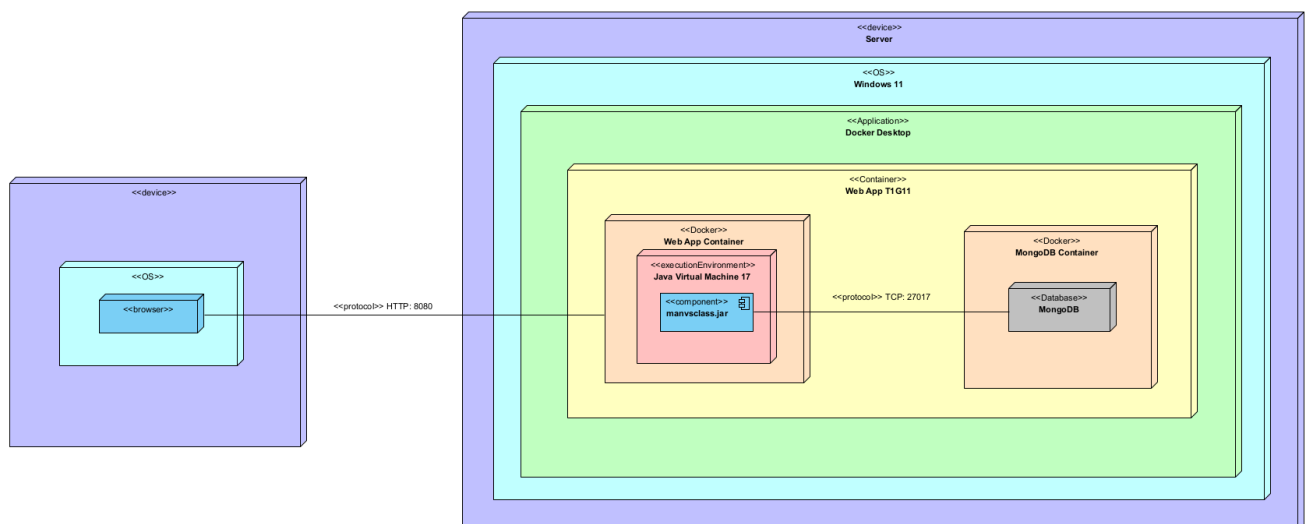
5.5 Deployment

La fase di deployment, o distribuzione, è un passaggio essenziale nel ciclo di vita di un'applicazione, in cui l'applicazione viene resa disponibile agli utenti finali attraverso un ambiente di produzione online. Questo processo implica una serie di attività complesse, tra cui la preparazione dell'ambiente di hosting, la configurazione dei server, l'installazione dei software necessari e la gestione delle dipendenze. Un deployment ben eseguito è fondamentale per garantire l'affidabilità, la scalabilità e le prestazioni ottimali dell'applicazione, assicurando un'esperienza utente positiva.

Il diagramma di deployment specifica come l'applicazione viene distribuita e allocata all'interno dei componenti hardware e software. In questo caso specifico, l'applicazione sviluppata viene eseguita su un computer che funge da server, su cui è installato il sistema operativo Windows 11. All'interno del sistema operativo viene eseguita l'applicazione Docker Desktop, che si occupa della gestione delle istanze di tipo Docker. Docker Desktop crea un container in cui vengono istanziati due Docker: uno per l'applicazione e uno per il Database.

Per consentire l'esecuzione dell'applicazione nel Docker, viene creato un ambiente di esecuzione basato su JVM17. I due Docker vengono connessi tramite una rete privata creata appositamente da Docker Desktop. In particolare, il Database viene connesso all'applicazione tramite il protocollo TCP alla porta 27017. Inoltre, viene esposto all'esterno del Docker la possibilità di connettersi all'applicazione tramite un qualsiasi browser, da qualsiasi sistema operativo, utilizzando il protocollo HTTP alla porta 8080.

Questo setup di deployment garantisce che l'applicazione sia eseguita in modo isolato all'interno del Docker e che i componenti necessari siano connessi correttamente per consentire la comunicazione e l'accesso all'applicazione tramite il protocollo HTTP.



5.6 Integrabilità

5.6.1 API Locali

Si riportano le API locali che sono utilizzate da altri moduli nello stesso ambiente durante lo sviluppo. Le API offrono i seguenti servizi:

- **Login admin**
- **Registrazione admin**
- **Modifica Classe**
- **Download Classe**
- **Ricerca Classe**
- **Eliminazione Classe**
- **Upload Classe**
- **Elenca Classi**
- **Filtra Classi**
- **Ordinamento Classi**
- **Nuovo Report**
- **Nuovo Like**
- **Upload Interaction**
- **Elenca Report**
- **Elenca Interaction**

5.6.2 Strutture Dati

5.6.2.1 ClassUT

E' la struttura dati utilizzata per memorizzare le informazioni sulle classi ed è strutturata nel seguente modo:

```
//Nome Classe
private String name;
//Data di inserimento
private String date;
//Difficoltà / Complessità
private String difficulty;
//Path nel file system
private String code_Uri;
//Breve descrizione della classe
private String description;
//Categorie di appartenenza della classe
private List<String> category;
```

5.6.2.2 Admin

E' la struttura dati utilizzata per memorizzare le informazioni personali e di accesso per gli amministratori ed è strutturata nel seguente modo:

```
//Informazioni personali
private String nome;
private String cognome;

//Informazioni profilo amministratore
private String username; //Username fase di registrazione
private String password; //Password di accesso al profilo
```

5.6.2.3 Interaction

E' la struttura dati utilizzata per memorizzare le informazioni sulle interazioni dell'utente con le classi, come i "Mi piace" o i report, ed è strutturata nel seguente modo:

```
//Id interazione
private int id_i;
//Email utente
private String email;
//Nome della classe
private String name;
//Id utente
private long id;
//Tipo di interazione:
//0 -> Report
//1 -> Like
private int type;
//Commento nel caso del report, null altrimenti
private String commento;
//Data di inserimento
private String date;
```

5.6.2.4 Operation

E' la struttura dati utilizzata per memorizzare le informazioni sulle operazioni dell'amministratore con le classi, come le eliminazioni o le modifiche, ed è strutturata nel seguente modo:

```
//Id Operazione
private int id_op;
//Username dell'admin che ha effettuato l'operazione
private String username_admin;
//Nome della classe
private String nome_classe;
//Tipo dell'operazione
private int type;
//Data dell'operazione
private String date;
```

5.6.3 Services

Di seguito sono riportati i metodi esposti dai Servizi in base al loro package di appartenenza

5.6.3.1 Login admin

```
public String loginAdmin(Admin admin1)
```

Metodo che permette di autenticare l'admin al fine di fornirgli l'accesso al sistema. Come parametro di ingresso deve essere passato un oggetto di tipo Admin con i campi username e password popolati. L'oggetto può essere popolato da un apposito form.

Input:

- Admin1 oggetto di tipo Admin che deve contenere i campi username e password popolati, al fine di autenticare l'amministratore.

Return: ritorna il messaggio "Ok" se l'autenticazione va a buon fine, "utente non loggato" altrimenti.

5.6.3.3 Registrazione admin

```
public Admin registraAdmin(Admin admin1)
```

Metodo che permette di salvare l'admin sulla base dati. Come parametro di ingresso deve essere passato un oggetto di tipo Admin con tutti i campi popolati. L'oggetto può essere popolato da un apposito form.

Input:

- Admin1 oggetto di tipo Admin che deve contenere i campi pieni, al fine di salvare l'amministratore sulla base dati.

Return: ritorna l'entità salvata nel database dopo l'operazione di salvataggio.

5.6.3.4 Modifica Classe

```
public ResponseEntity<String> modificaClasse( String name, ClassUT newContent)
```

Metodo che permette di modificare una classe già esistente nel sistema. Come parametri di ingresso devono essere passati il nome della classe da modificare e un oggetto di tipo ClassUT con tutti i nuovi campi popolati. L'oggetto può essere popolato da un apposito form.

Input:

- Name oggetto di tipo String che contiene il nome della classe
- newContent oggetto di tipo ClassUT che deve contenere tutti i campi popolati al fine di sostituire i valori vecchi della classe già esistente

Return: ritorna un oggetto di tipo ResponseEntity con il messaggio "Aggiornamento eseguito correttamente" e lo stato HTTP "OK" se non si verificano errori nella modifica.

Oppure il messaggio "Nessuna classe trovata o nessuna modifica effettuata" e lo stato NOT FOUND nel caso in cui ci dovessero essere errori.

5.6.3.5 Download Classe

```
public ResponseEntity<?> downloadClasse(("name") String name) throws Exception
```

Metodo che permette il download di un file contenente il codice sorgente di una classe dal sistema. Come parametro di ingresso deve essere passato il nome della classe da scaricare. La funzione cerca l'oggetto di tipo ClassUT corrispondente nel database utilizzando il parametro name e scarica il file contenente il codice della classe tramite il metodo FileDownloadUtil.downloadClassFile(). Il file scaricato viene poi restituito all'utente nel corpo della risposta HTTP.

Input:

- Name oggetto di tipo String che contiene il nome della classe

Return:

Restituisce un oggetto ResponseEntity contenente il file scaricato, utilizzando l'URI dell'oggetto ClassUT trovato.

5.6.3.6 Ricerca Classe

```
public List<ClassUT> ricercaClasse( String text)
```

Metodo che permette di cercare una classe all'interno del sistema. Come parametro di ingresso deve essere passato il nome della classe da ricercare. Il metodo usufruisce della funzione findByText() che si occupa di creare una lista di oggetti di tipo ClassUT corrispondenti alla ricerca del testo all'interno del campo nome della classe.

Input:

- Text oggetto di tipo String che contiene il nome della classe

Return: Restituisce una lista di oggetti di tipo ClassUT che corrispondono alla ricerca effettuata.

5.6.3.7 Eliminazione Classe

```
public ClassUT eliminaClasse( String name)
```

Metodo che permette di eliminare una classe all'interno del sistema. Come parametro di ingresso deve essere passato il nome della classe da eliminare.

Input:

- Name oggetto di tipo String che contiene il nome della classe

Return:

Restituisce l'oggetto ClassUT che è stato rimosso dalla collezione. In caso di mancata corrispondenza della query con alcun documento della collezione, il valore di ritorno sarà null.

5.6.3.8 Upload Classe

```
public ClassUT uploadClasse( ClassUT classe)
```

Metodo che permette di caricare una classe all'interno del sistema. Come parametro di ingresso deve essere passato un oggetto di tipo ClassUT con tutti i campi popolati. L'oggetto può essere popolato da un apposito form.

Input:

- Classe oggetto di tipo ClassUT che deve contenere tutti i campi popolati al fine di caricare la nuova classe nel sistema

Return: ritorna l'entità salvata nel database dopo l'operazione di salvataggio.

5.6.3.9 Elenca Classi

```
public List<ClassUT> elencaClassi()
```

Metodo che permette di visualizzare tutte le classi all'interno del sistema.

Return: Restituisce una lista di oggetti di tipo ClassUT che sono presenti all'interno del sistema

5.6.3.10 Filtra Classe

```
public List<ClassUT> filtraClassiD(String difficulty)
```

Metodo che permette di filtrare le classi all'interno del sistema. Come parametro di ingresso deve essere passato un oggetto di tipo String contenente la difficoltà secondo cui filtrare.

Input:

- Difficulty oggetto di tipo String che contiene la categoria secondo cui filtrare

Return: Restituisce una lista di oggetti di tipo ClassUT che corrispondono alla ricerca effettuata.

5.6.3.11 Ordinamento Classi

```
public List<ClassUT> ordinaClassi()
```

Metodo che permette di ordinare le classi all'interno del sistema per data. Si distingue dall'ordinamento per nomi che possiede un nome diverso:

```
public List<ClassUT> ordinaClassiNomi()
```

Le funzioni utilizzano i metodi orderByDate e orderByNome rispettivamente.

Return: Restituisce una lista di oggetti di tipo ClassUT che corrispondono alla ricerca effettuata.

5.6.3.12 Nuovo Report

```
public String newReport( String name, String commento )
```

Metodo che permette di caricare un nuovo report all'interno del sistema. Come parametro di ingresso deve essere passato un oggetto di tipo String contenente il nome della classe di cui fare il report e un oggetto di tipo String contenente una breve descrizione del report effettuato. Gli oggetti possono essere popolati con un apposito form.

Input:

- Name oggetto di tipo String che contiene il nome della classe da segnalare.
- Commento oggetto di tipo String che contiene una breve descrizione della segnalazione effettuata.

Return: ritorna il messaggio "Nuova interazione di tipo report inserita"

5.6.3.13 Nuovo Like

```
public String newLike( String name)
```

Metodo che permette di caricare un nuovo like all'interno del sistema. Come parametro di ingresso deve essere passato un oggetto di tipo String contenente il nome della classe a cui applicare il like

Input:

- Name oggetto di tipo String che contiene il nome della classe a cui applicare il like

Return: ritorna il messaggio "Nuova interazione di tipo like inserita"

5.6.3.14 Upload Interaction

```
public interaction UploadInteraction( interaction interazione)
```

Metodo che permette di caricare una nuova interazione all'interno del sistema. Come parametro di ingresso deve essere passato un oggetto di tipo interaction contenente tutti i campi popolati.

Input:

- Interazione oggetto di tipo Interaction che contiene tutti i campi popolati al fine di inserire una nuova interazione nel sistema

Return: ritorna l'entità salvata nel database dopo l'operazione di salvataggio.

5.6.3.15 Elenca Report

```
public List<interaction> elencaReport()
```

Metodo che permette di visualizzare tutti i report all'interno del sistema.

Return: Restituisce una lista di oggetti di tipo Interaction con il campo type=0 che sono presenti all'interno del sistema

5.6.3.16 Elenca Interaction

```
public List<interaction> elencaInt()
```

Metodo che permette di visualizzare tutti i report all'interno del sistema.

Return: Restituisce una lista di oggetti di tipo Interaction che sono presenti all'interno del sistema

5.6.4 Utils

Di seguito vengono riportate le API dei servizi di Utilità.

5.6.4.1 getLikes

```
public long getLikes(String name)
```

Restituisce il numero di "Likes" (valore intero) per un'interazione nella collezione "interaction" di MongoDB.

Input:

name (String): il nome dell'interazione di cui si vuole conoscere il numero di "Likes".

Return:

count (long): il numero di "Likes" dell'interazione.

5.6.4.2 findReport

```
public List<interaction> findReport()
```

Restituisce una lista di oggetti di tipo "interaction" e type=0 dalla collezione "interaction" di MongoDB, ovvero i report presenti.

Input:

Nessun parametro di input.

Return:

posts (List<interaction>): una lista di oggetti di tipo "interaction".

5.6.4.3 findByText

```
public List<ClassUT> findByText(String text)
```

Restituisce una lista di oggetti di tipo "ClassUT" dalla collezione "ClassUT" di MongoDB, filtrati in base al nome della classe.

Input:

text (String): il testo da cercare all'interno del nome della classe.

Return:

posts (List<ClassUT>): una lista di oggetti di tipo "ClassUT".

5.6.4.4 findAdminByUsername

```
public Admin findAdminByUsername(String username)
```

Restituisce un oggetto di tipo "Admin" dalla collezione "Admin" di MongoDB, filtrato in base allo username.

Input:

username (String): lo username dell'amministratore da cercare.

Return:

admin (Admin): un oggetto di tipo "Admin".

5.6.4.5 searchAndFilter

```
public List<ClassUT> searchAndFilter(String text, String category)
```

Restituisce una lista di oggetti di tipo "ClassUT" dalla collezione "ClassUT" di MongoDB, filtrati in base al testo da cercare all'interno del nome e alla categoria della classe.

Input:

text (String): il testo da cercare all'interno del nome e della descrizione della classe.

category (String): la categoria della classe da cercare.

Return:

posts (List<ClassUT>): una lista di oggetti di tipo "ClassUT".

5.6.4.6 filterByCategory

```
public List<ClassUT> filterByCategory(String category)
```

Restituisce una lista di oggetti di tipo "ClassUT" dalla collezione "ClassUT" di MongoDB, filtrati in base alla categoria della classe.

Input:

category (String): la categoria della classe da cercare.

Return:

posts (List<ClassUT>): una lista di oggetti di tipo "ClassUT".

5.6.4.7 orderByDate

```
public List<ClassUT> orderByDate()
```

Restituisce una lista di oggetti di tipo "ClassUT" dalla collezione "ClassUT" di MongoDB, ordinati in base alla data di creazione della classe.

Input:

Nessun parametro di input.

Return:

posts (List<ClassUT>): una lista di oggetti di tipo "ClassUT".

5.6.4.8 orderByName

```
public List<ClassUT> orderByName()
```

Restituisce una lista di oggetti di tipo "ClassUT" dalla collezione "ClassUT" di MongoDB, ordinati in base al nome della classe.

Input:

Nessun parametro di input.

Return:

posts (List<ClassUT>): una lista di oggetti di tipo "ClassUT".

5.6.4.9 filterByDifficulty

```
public List<ClassUT> filterByDifficulty(String difficulty)
```

Restituisce una lista di oggetti di tipo "ClassUT" dalla collezione "ClassUT" di MongoDB, filtrati in base alla difficoltà della classe.

Input:

difficulty (String): la difficoltà della classe da cercare.

Return:

posts (List<ClassUT>): una lista di oggetti di tipo "ClassUT".

5.6.4.10 searchAndFilter

```
public List<ClassUT> searchAndFilter(String text, String difficulty)
```

Restituisce una lista di oggetti di tipo "ClassUT" dalla collezione "ClassUT" di MongoDB, filtrati in base al testo da cercare all'interno del nome e alla difficoltà della classe.

Input:

text (String): il testo da cercare all'interno del nome e della descrizione della classe.

difficulty (String): la difficoltà della classe da cercare.

Return:

posts (List<ClassUT>): una lista di oggetti di tipo "ClassUT".

5.6.4.11 downloadClassFile

```
public static ResponseEntity<Resource> downloadClassFile(String downloadpath) throws Exception
```

Restituisce un'entità di risposta HTTP contenente il file Java situato nel percorso specificato.

Input:

downloadpath (String): il percorso del file Java da scaricare.

Return:

ResponseEntity<Resource>: un'entità di risposta HTTP contenente il file Java.

Exception:

Exception se si verifica un errore durante la creazione dell'entità di risposta HTTP o durante l'accesso al file Java.

5.6.4.12 saveClassFile

```
public static void saveClassFile(String fileName,String cname,MultipartFile multipartFile) throws IOException
```

Salva un file all'interno di una specifica directory "Files-Upload/nomeclasse". Se la directory non esiste, viene creata. Il file viene salvato con il nome specificato.

Input:

fileName (String): il nome del file da salvare.

cname (String): il nome della classe a cui appartiene il file.

multipartFile (MultipartFile): il file da salvare.

Return:

Nessun valore di ritorno.

Exception:

IOException: se si verifica un errore durante il salvataggio del file.

5.6.4.13 deleteDirectory

```
public static void deleteDirectory(File directory) throws IOException
```

Elimina una directory e tutti i file al suo interno.

Input:

directory (File): la directory da eliminare.

Return:

Nessun valore di ritorno.

Exception:

IOException: se si verifica un errore durante l'eliminazione della directory o dei file al suo interno.

5.6.5 Rest API

La REST API (Representational State Transfer) è un'architettura software che definisce un insieme di principi per la creazione di servizi web. Questi servizi consentono ai sistemi di comunicare e scambiare dati in modo efficiente grazie all'utilizzo di interfacce comuni.

Le API REST sono stateless, il che significa che ogni richiesta del client deve contenere tutte le informazioni necessarie per elaborare quella richiesta. Utilizzano i metodi HTTP, come GET, POST, PUT e DELETE, per definire le operazioni che possono essere eseguite sui dati. Ogni risorsa in un'API REST è identificata da un URI univoco. I dati vengono rappresentati in formati come JSON o XML.

Nel progetto, sono state implementate le REST API relative a ogni caso d'uso. Queste API consentono di accedere ai dati e di eseguire operazioni come la creazione, l'aggiornamento e la cancellazione di informazioni specifiche.

5.6.5.1 Elenca classi “/home” (GET)

Questa API permette di inoltrare una richiesta HTTP di tipo GET che restituisce l'elenco di tutte le classi disponibili.

Restituisce Oggetti Json del seguente tipo:

```
{
  "name": "sdds",
  "date": "2023-07-10",
  "difficulty": "Beginner",
  "code_uri": "Files-Upload/sdds/ciao.java",
  "description": "sddsrr",
  "category": [
    "sdsdyyyy",
    "sdsd",
    "sdd"
  ]
},
```

Tipi di Risposta :

- **200 OK:** viene restituito un elenco di classi o un elenco vuoto [].

- **400 Bad Request:** Il payload di input è incompleto o non valido.
- **500 Internal Server Error:** Si è verificato un errore interno al server.

5.6.5.2 Ricerca Classe per nome “/home/{text}” (GET)

Questa API permette di inoltrare una richiesta HTTP di tipo GET che restituisce una classe con un nome.

{text} rappresenta il parametro di ricerca per nome

Restituisce Oggetti Json del seguente tipo:

```
{
  "name": "sdds",
  "date": "2023-07-10",
  "difficulty": "Beginner",
  "code_uri": "Files-Upload/sdds/ciao.java",
  "description": "sddsrr",
  "category": [
    "sdsdyyyy",
    "sdsd",
    "sdd"
  ]
},
```

Tipi di Risposta :

- **200 OK:** viene restituito una classe o un elenco vuoto [].
- **400 Bad Request:** Il payload di input è incompleto o non valido.
- **500 Internal Server Error:** Si è verificato un errore interno al server.

5.6.5.3 Ordina classi per nome “/orderbyname” (GET)

Questa API permette di inoltrare una richiesta HTTP di tipo GET che restituisce un elenco di classi ordinate per nome.

Restituisce Oggetti Json del seguente tipo:

```
{
  "name": "sdds",
  "date": "2023-07-10",
  "difficulty": "Beginner",
  "code_uri": "Files-Upload/sdds/ciao.java",
  "description": "sddsrr",
  "category": [
    "sdsdyyyy",
    "sdsd",
    "sdd"
  ]
},
```

Tipi di Risposta :

- **200 OK:** viene restituito un elenco di classi o un elenco vuoto [].

- **400 Bad Request:** Il payload di input è incompleto o non valido.
- **500 Internal Server Error:** Si è verificato un errore interno al server.

5.6.5.4 Ordina classi per data `"/orderbydate"` (GET)

Questa API permette di inoltrare una richiesta HTTP di tipo GET che restituisce un elenco di classi ordinate per data.

Restituisce Oggetti Json del seguente tipo:

```
{
  "name": "sdds",
  "date": "2023-07-10",
  "difficulty": "Beginner",
  "code Uri": "Files-Upload/sdds/ciao.java",
  "description": "sddsrr",
  "category": [
    "sdsdyyyy",
    "sdsd",
    "sdd"
  ]
},
```

Tipi di Risposta :

- **200 OK:** viene restituito un elenco di classi o un elenco vuoto [].
- **400 Bad Request:** Il payload di input è incompleto o non valido.
- **500 Internal Server Error:** Si è verificato un errore interno al server.

5.6.5.5 Filtra classi per categoria `"/Cfilterby/{category}"` (GET)

Questa API permette di inoltrare una richiesta HTTP di tipo GET che restituisce un elenco di classi filtrate per categoria.

Restituisce Oggetti Json del seguente tipo:

```
{
  "name": "sdds",
  "date": "2023-07-10",
  "difficulty": "Beginner",
  "code Uri": "Files-Upload/sdds/ciao.java",
  "description": "sddsrr",
  "category": [
    "sdsdyyyy",
    "sdsd",
    "sdd"
  ]
},
```

Tipi di Risposta :

- **200 OK:** viene restituito un elenco di classi o un elenco vuoto [].
- **400 Bad Request:** Il payload di input è incompleto o non valido.
- **500 Internal Server Error:** Si è verificato un errore interno al server.

5.6.5.6 Filtra classi per categoria “/Cfilterby/{category}” (GET)

Questa API permette di inoltrare una richiesta HTTP di tipo GET che restituisce un elenco di classi filtrate per categoria.

{category} rappresenta il parametro di filtraggio della categoria

Restituisce Oggetti Json del seguente tipo:

```
{
  "name": "sdds",
  "date": "2023-07-10",
  "difficulty": "Beginner",
  "code Uri": "Files-Upload/sdds/ciao.java",
  "description": "sddsrr",
  "category": [
    "sdsdyyyy",
    "sdsd",
    "sdd"
  ]
},
```

Tipi di Risposta :

- **200 OK:** viene restituito un elenco di classi o un elenco vuoto [].
- **400 Bad Request:** Il payload di input è incompleto o non valido.
- **500 Internal Server Error:** Si è verificato un errore interno al server.

5.6.5.7 Filtra classi per categoria “/Dfilterby/{difficulty}” (GET)

Questa API permette di inoltrare una richiesta HTTP di tipo GET che restituisce un elenco di classi filtrate per categoria.

{difficulty} rappresenta il parametro di filtraggio per difficoltà

Restituisce Oggetti Json del seguente tipo:

```
{
  "name": "sdds",
  "date": "2023-07-10",
  "difficulty": "Beginner",
  "code Uri": "Files-Upload/sdds/ciao.java",
  "description": "sddsrr",
  "category": [
    "sdsdyyyy",
    "sdsd",
    "sdd"
  ]
},
```

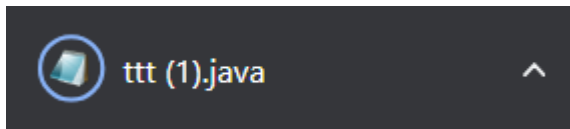
Tipi di Risposta :

- **200 OK:** viene restituito un elenco di classi o un elenco vuoto [].
- **400 Bad Request:** Il payload di input è incompleto o non valido.
- **500 Internal Server Error:** Si è verificato un errore interno al server.

5.6.5.8 Filtra classi per categoria “/downloadFile/{name}” (GET)

Questa API permette di inoltrare una richiesta HTTP di tipo GET che restituisce il file java corrispondente alla classe richiesta.

{name} rappresenta il parametro per scaricare il file :



Tipi di Risposta :

- **200 OK:** viene restituito il contenuto del file .java .
- **400 Bad Request:** Il payload di input è incompleto o non valido.
- **500 Internal Server Error:** Si è verificato un errore interno al server.

5.6.5.9 Upload della descrittore classe+ file.java `"/uploadFile (POST)`

Questa API permette di inoltrare una richiesta HTTP di tipo POST che inserisce il descrittore della classe e il file .java corrispondente.

Parametri richiesti:

File in formato .java

model contenente la stringa JSON corrispondente alla classe da inserire

	Key	Value	Content type
<input checked="" type="checkbox"/>	file	FTPFile.java ×	Auto
<input checked="" type="checkbox"/>	model	<pre>{ "name": "rafft2", "date": "2022-05-13", "difficulty": "Intermediate", "code_uri": "/code/123", "description": "This is a test class", "category": ["Java", "Programming", "Testing"] }</pre>	Auto
	Key		Auto

Tipi di Risposta :

- **200 OK:** viene restituito un messaggio con il path di salvataggio del file e la dimensione del file appena caricato.



- **400 Bad Request:** Il payload di input è incompleto o non valido.
- **500 Internal Server Error:** Si è verificato un errore interno al server.

5.6.5.10 Modifica classe `"/update/{name}"` (POST)

Questa API permette di inoltrare una richiesta HTTP di tipo POST che aggiorna i campi della classe specificata dal parametro di ricerca name.

Parametri richiesti:

{name} parametro di ricerca della classe

body contenente la stringa JSON del report con i parametri modificati

```
{
  "name": "sdds",
  "date": "2023-07-10",
  "difficulty": "Beginner",
  "code Uri": "Files-Upload/sdds/ciao.java",
  "description": "sddsrr",
  "category": [
    "sdsdyyyy",
    "sdsd",
    "sdd"
  ]
},
```

Tipi di Risposta :

- **200 OK:** viene restituito un messaggio che conferma il successo dell'operazione di modifica.
- **400 Bad Request:** Il payload di input è incompleto o non valido.
- **500 Internal Server Error:** Si è verificato un errore interno al server.

5.6.5.11 Nuovo like ad una classe `"/newLike/{name}"` (POST)

Questa API permette di inoltrare una richiesta HTTP di tipo POST che aggiorna i campi della classe specificata dal parametro di ricerca name.

Parametri richiesti:

{name} parametro di ricerca della classe

Tipi di Risposta :

- **200 OK:** viene restituito un messaggio che conferma il successo dell'operazione di inserimento del nuovo like.
- **400 Bad Request:** Il payload di input è incompleto o non valido.
- **500 Internal Server Error:** Si è verificato un errore interno al server.

5.6.5.12 Nuovo report ad una classe *"/newReport/{name}"* (POST)

Questa API permette di inoltrare una richiesta HTTP di tipo POST che aggiorna i campi della classe specificata dal parametro di ricerca name.

Parametri richiesti:

{name} parametro di ricerca della classe

stringa JSON con i parametri del report da inserire

```
{
  "id_i": 0,
  "email": "prova.449660@email.com",
  "name": "ttt",
  "id": 449660,
  "type": 0,
  "commento": "sdsd",
  "date": "2023-07-11"
}
```

Tipi di Risposta :

- **200 OK:** viene restituito un messaggio che conferma il successo dell'operazione di inserimento del nuovo report.
- **400 Bad Request:** Il payload di input è incompleto o non valido.
- **500 Internal Server Error:** Si è verificato un errore interno al server.

5.6.5.12 Registrazione Admin *"/registraAdmin/"* (POST)

Questa API permette di inoltrare una richiesta HTTP di tipo POST che inserisce un nuovo Admin nel sistema

Parametri richiesti:

stringa JSON con i parametri dell'admin da inserire

```
{
  "nome": "Mario",
  "cognome": "Rossi",
  "username": "mario.rossi",
  "password": "password123"
}
```

Tipi di Risposta :

- **200 OK:** viene restituito un messaggio che conferma il successo dell'operazione di inserimento del nuovo admin.
- **400 Bad Request:** Il payload di input è incompleto o non valido.
- **500 Internal Server Error:** Si è verificato un errore interno al server.

Capitolo 6: Implementazione

6.1 Introduzione

In questo capitolo, viene presentata l'implementazione del nostro progetto introducendo i relativi diagrammi di sequenza di dettaglio.

6.2 Sequence Diagram di Dettaglio

I Sequence Diagram di dettaglio sono un tipo di diagramma dinamico che forniscono una visione dettagliata dell'interazione tra gli oggetti all'interno di un sistema software. Questi diagrammi mostrano le chiamate di metodo tra gli oggetti nel corso del tempo e illustrano l'ordine e il flusso delle operazioni. Nei prossimi paragrafi verranno presentati i Sequence Diagram di dettaglio per illustrare l'interazione tra gli oggetti nel nostro sistema.

6.2.1 Sequence Diagram: RicercaClassi

6.2.2 Sequence Diagram: elencaClassi

6.2.3 Sequence Diagram: ordinaClassi

6.2.4 Sequence Diagram: filtraClassi

6.2.5 Sequence Diagram: DownloadClasse

6.2.6 Sequence Diagram: ModificaClasse

6.2.7 Sequence Diagram: UploadClasse

Capitolo 7: Testing

Il Testing ha l'obiettivo di trovare difetti e dimostrare che un sistema soddisfa i suoi requisiti. Questa fase, è stata eseguita con continuità durante il ciclo di sviluppo.

Sono stati scelte due tipologie di test effettuati:

- Testing GUI
- Testing basato su API REST

7.1 Testing GUI

Per testare le interfacce utente è stato utilizzato un approccio basato sul criterio di copertura dei cammini indipendenti. In particolare, sono stati definiti cinque CFG (Control Flow Graph) che rappresentano alcuni degli scenari implementati. Questo approccio permette di garantire una copertura completa dei possibili flussi di esecuzione all'interno dell'interfaccia utente, andando a testare tutti i percorsi possibili attraverso il grafo di controllo di flusso. In questo modo, è possibile identificare eventuali problemi o bug all'interno dell'interfaccia utente in modo completo e sistematico

7.1.1 Testing GUI:

7.2 Testing basato su API REST