



Scuola Politecnica e delle Scienze di Base
Corso di Laurea Magistrale in Ingegneria Informatica

Software Architecture Design

Documentazione A13 New features - bug fix

Anno Accademico 2023/2024

Docente

prof.ssa Fasolino Anna Rita

Candidati

Andriy Korsun	M63001275
Giuseppe Laterza	M63001411
Luca D'Angelo	M63001453

Indice

1	Introduzione	1
1.1	Task: Google Login	3
1.2	Task : Bug fix cancellazione classi da FileSystem	4
1.3	Task : Bug fix Download delle classi da parte dell'amministratore	5
1.4	Task : Bug fix Logout da parte del player	7
1.5	Task : Bug fix Registrazione da parte del player	7
1.6	Task : Refactoring	9
1.7	Metodologia di sviluppo	16
1.7.1	Release	17
2	Processo di Sviluppo	26
2.1	Strumenti Software di Supporto	26
3	Specifiche di Progetto	30
3.1	Utenti del sistema	30
3.2	Glossario dei Dati	31
3.3	Architettura a Microservizi	32
3.4	Requisiti Funzionali	33
3.5	Storie Utente	36
3.6	Criteri di Accettazione	37

3.7 Requisiti non funzionali	39
4 Fase di Analisi	42
4.1 Class Diagram	42
4.2 Use Case Diagram	43
4.2.1 Casi d'uso relativi ai task T23	43
4.2.2 Casi d'uso relativi al task T1	48
4.3 Component diagram	50
4.4 Composite Diagram	51
4.5 Activity Diagram	52
4.5.1 Activity Diagram relativi al task T1	53
4.5.2 Activity Diagram relativi al task T23	54
4.6 Sequence Diagram	56
4.6.1 Sequence Diagram relativi al task T1	57
4.6.2 Sequence Diagram relativi ai task T23	57
4.7 Deployment Diagram	60
5 Implementazione	62
5.1 Login con Google	62
5.1.1 Configurazione dell'API Google	62
5.1.2 Configurazione del Progetto Spring Boot	63
5.1.3 Configurazione di Spring Security per l'autenticazione OAuth2	64
5.1.4 Implementazione OAuth User Google e OAuth Google Service	65
5.1.5 Implementazione del Success Handler	65
5.2 Cancellazione Classe	65
5.3 Download Classe	68
5.4 Logout	68
5.5 Registrazione	71
5.6 Refactoring	75

6 Testing	78
6.1 Login con Google	78
6.2 Cancellazione classe	78
6.3 Download Classe	83
6.4 Logout	84
6.5 Registrazione	86

Chapter 1

Introduzione

In questo capitolo vengono introdotti i task a noi assegnati con lo scopo di ampliare le funzionalità della web-app per quanto riguarda la Registrazione/Autenticazione e risolvere alcuni degli [issue](#) della versione di partenza dell'elaborato presi in carico. Questi mirano sia ad aggiungere nuove feature, che a risovare errori di componenti già esistenti. Infine, vengono forniti i dettagli sulla metodologia di sviluppo utilizzata.

AS-IS del sistema: La versione della web-app sulla quale è stata implementata sia la possibilità di accedere alla applicazione con il proprio account Google, sia le modifiche correttive di bug segnalati sotto forma di issue è la [A13](#). Al suo interno si può osservare che gli studenti al momento possono solo creare un account inserendo i propri dati nella form, compresi email e password, oppure hanno la possibilità di accedere utilizzando le informazioni associate al proprio account Facebook, che verranno memorizzate nel database dell'applicazione. Tali informazioni verranno utilizzate anche per permettere l'accesso all'app, confrontandole con quelle presenti nel database. Sebbene la registrazione dei nuovi utenti avvenisse correttamente, i processi responsabili presentavano degli errori che rendeva la sua gestione inefficiente e poco chiara.

Una volta all'interno della propria pagina dedicata, in caso l'utente abbia intenzione di

effettuare il Logout, egli visualizza a schermo un errore, associato alla sessione precedentemente instanziata non cancellata correttamente.

Da lato dell'admin, invece, all'interno del suo cruscotto tra le varie opzioni non è possibile risalire al codice sorgente delle classi caricate, né da lui stesso precedentemente, né da altri amministratori registrati, poiché viene scaricato per tutte le classi un file contenente un messaggio di errore.

TO-BE del sistema: Con la versione ultimata della web-app [A13 new features - bug fix](#), gli utenti avranno la possibilità di **autenticarsi anche utilizzando il loro account Google**. Essendo necessari per una vasta gamma di servizi e dispositivi, gli account Google sono più numerosi rispetto agli account attivi di Facebook, che principalmente è un social network. Questa integrazione, quindi, semplifica l'utilizzo del sistema a favore di un bacino più ampio di utenti.

Il processo, noto come "Single Sign-On" (SSO), velocizza l'esperienza di accesso, riducendo contemporaneamente la necessità di ricordare molteplici credenziali. Per implementare questa funzionalità, è necessario utilizzare l'API di Google per l'autenticazione e integrarla con il sistema esistente. I componenti utilizzati a tal fine sono: OAuthUserService e GoogleSuccessHandler.

Una volta autenticati con Google, gli utenti possono accedere alle stesse funzionalità e informazioni a cui possono accedere gli utenti che si autenticano tramite email e password, oltre alla possibilità di eseguire un **Logout corretto**, cancellando i token associati all'utente e reindirizzando alla pagina di Login.

Quest'ultima è stata oggetto di **refactoring**, così come tutte le pagine associate al componente T23, quindi quelle legate alla registrazione e accesso al sistema da parte dell'utente, compresi anche i casi di password dimenticata e cambio password, in aggiunta alla schermata di menu iniziale. Queste modifiche fanno parte di un processo più grande, che comprende la separazione delle pagine html dagli elementi di stile CSS e script JS, rior-

ganizzando gli elementi presenti nel componente T23, in aggiunta alla pagina index.html di T6, e migliorando le performance dell'applicazione in particolar modo riducendo i tempi di caricamento delle pagine sul browser.

Il processo di registrazione è stato migliorato per garantire un'esperienza utente più fluida e intuitiva e sono stati risolti vari errori che ne impedivano il corretto funzionamento, permettendo agli utenti di registrarsi senza incorrere in problemi tecnici. L'interazione tra il frontend e il backend è stata ottimizzata per assicurare che le informazioni inserite dagli utenti siano trasmesse e gestite correttamente. Inoltre, è stato implementato un sistema di notifiche che fornisce **feedback immediato e chiaro agli utenti in caso di errori** durante la registrazione, rendendo più semplice per gli utenti comprendere e correggere eventuali problemi.

Per quanto riguarda l'admin registrato, sono state ampliate le operazioni da lui eseguibili, potendo ora **scaricare il codice delle classi** usate dagli studenti mentre usano la web-app.

In dettaglio i problemi e come si è arrivati alla loro soluzione, con i conseguenti vantaggi ottenuti. Successivamente nel documento vengono dettagliate le operazioni svolte dal punto di vista implementativo nel [capitolo - Implementazione](#).

1.1 Task: Google Login

Lo scope di questo requisito è limitato al tipo di utente *studente*: integrare l'autenticazione Google, consentendo agli studenti di accedere utilizzando i propri account Google.

L'obiettivo principale di questo requisito è semplificare il processo di registrazione e accesso per i giocatori. Questo miglioramento ha diversi vantaggi:

- Esperienza utente migliorata: gli studenti non dovranno più creare e ricordare un nome utente e una password separati per la web-app. Ciò rende l'accesso più

comodo.

- Maggiore sicurezza: il sistema di autenticazione di Google è sicuro e affidabile grazie all'impiego dello standard OAuth2, riducendo il rischio di accesso non autorizzato agli account dei giocatori.
- Maggiore quantità di utenti: offrendo un metodo di accesso tramite account già esistenti, la web-app potrebbe spingere più studenti a utilizzare le sue funzionalità, aumentandone la base di utenti.

1.2 Task : Bug fix cancellazione classi da FileSystem

Correzione di un bug legato all'eliminazione delle classi da parte degli admin.

Attualmente, quando un admin elimina una classe, è stato segnalato che continua ad essere presente in una cartella (T9-G19/Progetto-SAD-G19-master/FolderTree per T9, e allo stesso modo in T8) all'interno del file system.

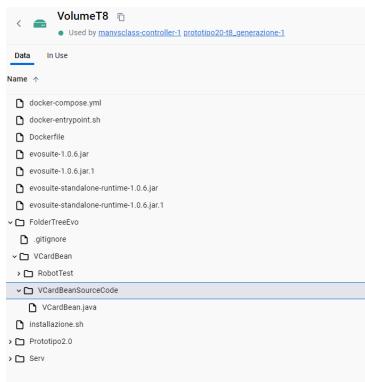


Figure 1.1: FolderTree T8

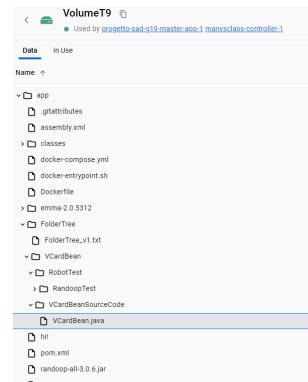


Figure 1.2: FolderTree T9

L'obiettivo è garantire che quando un admin elimina una classe, questa venga completamente rimossa dal file system.

Lo scopo della risoluzione del bug di eliminazione delle classi è garantire una corretta gestione delle classi per gli admin e mantenere l'integrità del file system dell'app. Risolvere

questo bug offre i seguenti vantaggi:

- Esperienza utente migliorata: garantendo che le classi eliminate vengano completamente rimosse dal file system, gli admin possono gestire più efficacemente le loro classi ed evitare disordine.
- Dati coerenti: risolvere il bug garantisce che l'interfaccia della web-app e il file system rimangano sincronizzati, prevenendo potenziali confusioni o incongruenze per studenti e amministratori.

Effettuando un'analisi dell'attività di cancellazione abbiamo constatato che il bug non sussisteva (Implementazione: la cancellazione avviene nella classe HomeController del container T1, nell'end point /deletefile/{filename}).

1.3 Task : Bug fix Download delle classi da parte dell'amministratore

Correzione di un bug legato al download delle classi da parte degli amministratori. Attualmente, quando un amministratore tenta di scaricare una qualsiasi delle classi presenti, viene scaricato un file .java contenente una pagina HTML con un errore "404 Not Found" generato dal server Nginx che risiede nel componente UI Gateway.

```
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx/1.23.1</center>
</body>
</html>
<!-- a padding to disable MSIE and Chrome friendly error page -->
<!-- a padding to disable MSIE and Chrome friendly error page -->
```

```
<!-- a padding to disable MSIE and Chrome friendly error page -->
<!-- a padding to disable MSIE and Chrome friendly error page -->
<!-- a padding to disable MSIE and Chrome friendly error page -->
<!-- a padding to disable MSIE and Chrome friendly error page -->
```

Questo problema è dovuto alla mancanza del path corretto nelle configurazioni del server.

Intervenendo come visto in [Download Classe](#), quindi aggiornando il file

"ui_gateway\default.conf", l'obiettivo è raggiunto, con il file .java della classe correttamente scaricato quando l'amministratore clicca sul pulsante di download, evitando la restituzione di un file di errore.

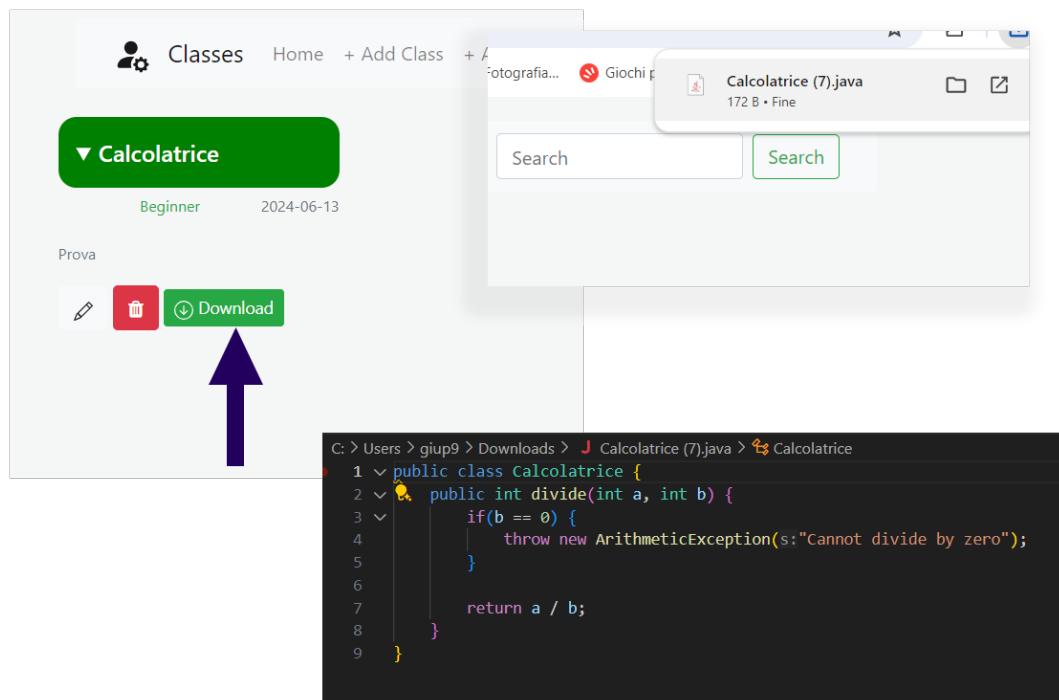
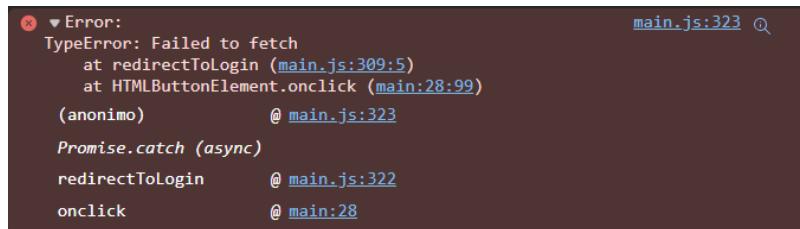


Figure 1.3: Procedimento di download della classe

1.4 Task : Bug fix Logout da parte del player

Correzione di un bug, e dell'[issue corrispondente](#), individuato quando un giocatore che ha effettuato l'accesso alla sua pagina principale tentava di eseguire il logout premendo l'apposito pulsante, ma il processo di logout non veniva completato correttamente.



```
✖ ▾ Error: main.js:323 🔎
  TypeError: Failed to fetch
    at redirectToLogin (main.js:309:5)
    at HTMLButtonElement.onclick (main:28:99)
  (anonimo)          @ main.js:323
  Promise.catch (async)
  redirectToLogin      @ main.js:322
  onclick              @ main:28
```

Figure 1.4: Errore - redirectToLogin()

In particolare nel componente T5 la chiamata alla funzione JS `redirectToLogin()`, in risposta alla pressione del bottone 'Logout', non funziona correttamente generando un messaggio d'errore e non effettuando l'uscita dall'account.

Apportando delle modifiche sia nel Controller del task T23, relativamente alla richiesta "/logout", utilizzando un metodo POST piuttosto dell'attuale GET, che nella funzione JS `redirectToLogin()` di "main.js" in T5, come è possibile osservare in dettaglio in [Implementazione - Logout](#), è stato risolto il problema ad esso legato.

1.5 Task : Bug fix Registrazione da parte del player

Durante il lavoro sul task T23, è emerso che, sebbene la registrazione dei nuovi utenti avvenisse correttamente, il file JavaScript contenuto nella pagina `register.js` conteneva degli errori e in pratica non veniva utilizzato in quanto sovrascritto dalla richiesta del form HTML.

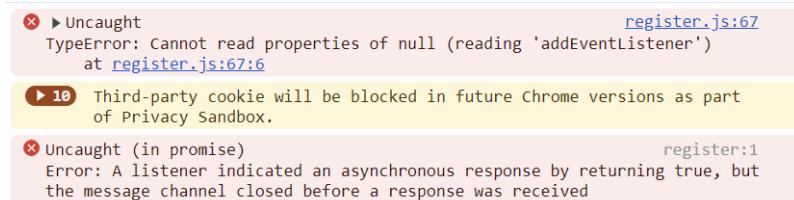


Figure 1.5: Errore - register.js

Questi errori sono stati risolti, garantendo così il corretto funzionamento del processo di registrazione.

Inoltre, il codice in javascript è stato integrato alla corretta chiamata POST al controller del container T23 /register , migliorando l'interazione tra il frontend e il backend.

Un ulteriore miglioramento implementato riguarda la gestione degli errori durante la registrazione. È stato sviluppato un sistema di notifiche popup che restituisce in maniera chiara e immediata gli errori all'utente. In precedenza, gli utenti ricevevano semplicemente una pagina contenente il body della response della richiesta POST, il che poteva risultare poco pratico e approssimativo. Ora, grazie a questo sistema, l'utente viene informato in modo tempestivo e dettagliato, migliorando significativamente l'esperienza utente.

Utente con questa email già registrato

Figure 1.6: Errore di registrazione precedente

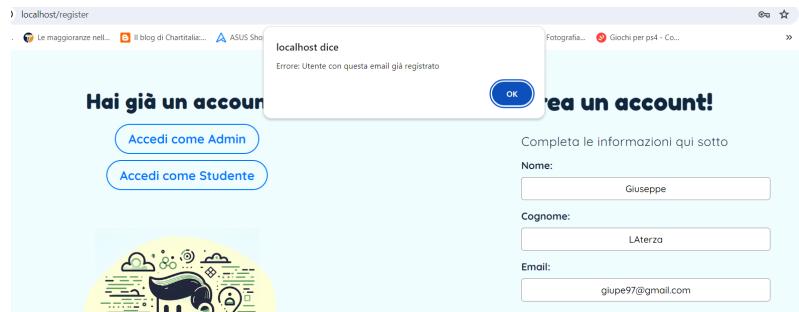


Figure 1.7: Errore di registrazione aggiornato

1.6 Task : Refactoring

Un altro task identificato e segnalato tramite [issue](#) GitHub riguarda il refactoring delle pagine HTML, CSS e JavaScript dell'applicazione, rispetto al task T23 (TODO1), in quanto già oggetto di studio per l'implementazione di nuovi casi d'uso e modifiche di correzione dei componenti necessari per i task di Bug Fix, e T6 (TODO2). La segnalazione sottolineava la necessità di rimuovere elementi di stile CSS e frammenti di script JS dalle pagine HTML, al fine di mantenere una netta separazione tra contenuto, stile e comportamento, garantendo anche una coerenza nella gerarchia del file system.

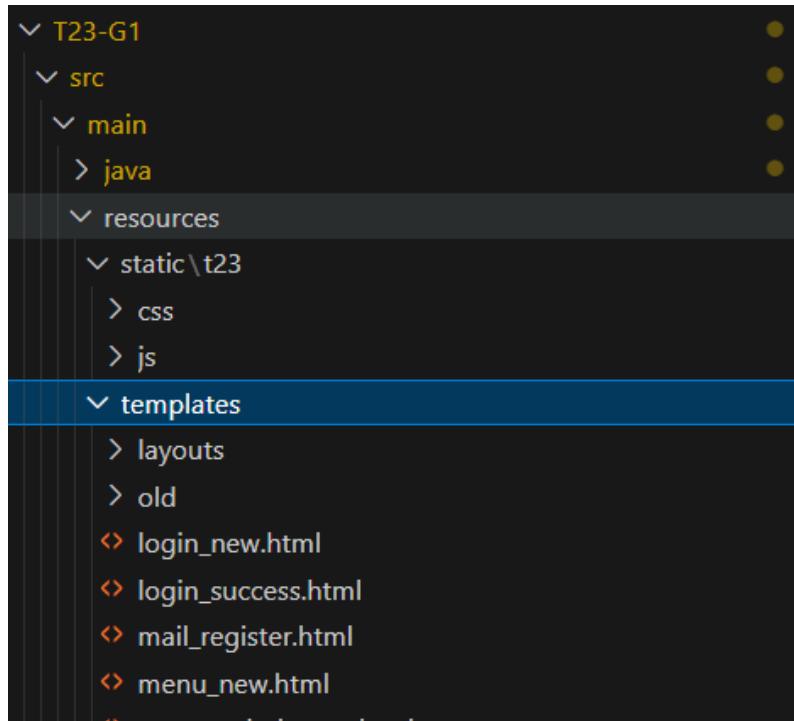


Figure 1.8: Refactoring - Pagine HTML

Nel dettaglio:

- Pagine HTML: Le pagine HTML devono essere memorizzate nella cartella `templates`.
- Pagine CSS: I file CSS devono essere memorizzati in una directory separata all'interno della cartella `static/css`.
- Pagine JS: I file JavaScript devono essere memorizzati in una directory separata all'interno della cartella `static/js`.
- Immagini: Le immagini devono essere memorizzate in una directory separata all'interno della cartella `css/images`.

La necessità di questo refactoring è emersa in quanto le pagine HTML attuali non rispondevano agli standard di pulizia e organizzazione richiesti. È stato utilizzato un approccio ottimizzato alla gestione del CSS, separando gli stili comuni da quelli specifici per pagina

in diversi file CSS. Questo approccio offre diversi vantaggi:

- Prestazioni Migliorate: Gli stili comuni sono centralizzati in un unico file CSS, riducendo la duplicazione e migliorando i tempi di caricamento. Inoltre, il file CSS comune viene memorizzato nella cache del browser, rendendo più veloci i caricamenti successivi.
- Manutenzione Semplificata: Le modifiche agli elementi comuni vengono effettuate in un solo posto, riducendo il rischio di incoerenze. Gli stili specifici per pagina sono in file separati, facilitando gli aggiornamenti senza influenzare altre pagine.
- Organizzazione e Scalabilità: Organizzare il CSS in file comuni e specifici rende il codice più pulito e comprensibile. È più facile aggiungere nuove pagine o funzionalità.

Essendo l'applicazione un gioco, abbiamo deciso di cogliere l'occasione per migliorare anche il design delle pagine di accesso, rendendole più moderne, accattivanti e funzionali:

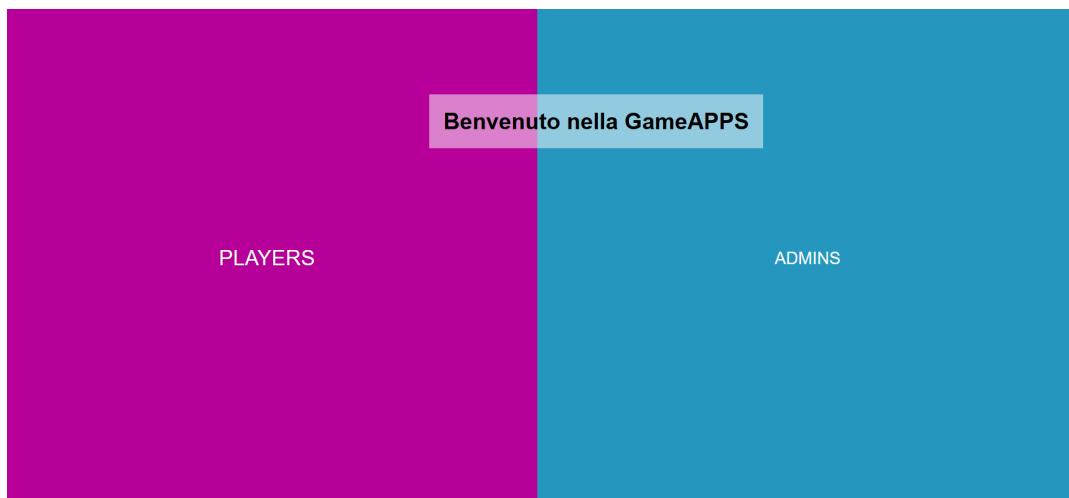


Figure 1.9: Pagina Menù precedente



Figure 1.10: Pagina Menù aggiornata

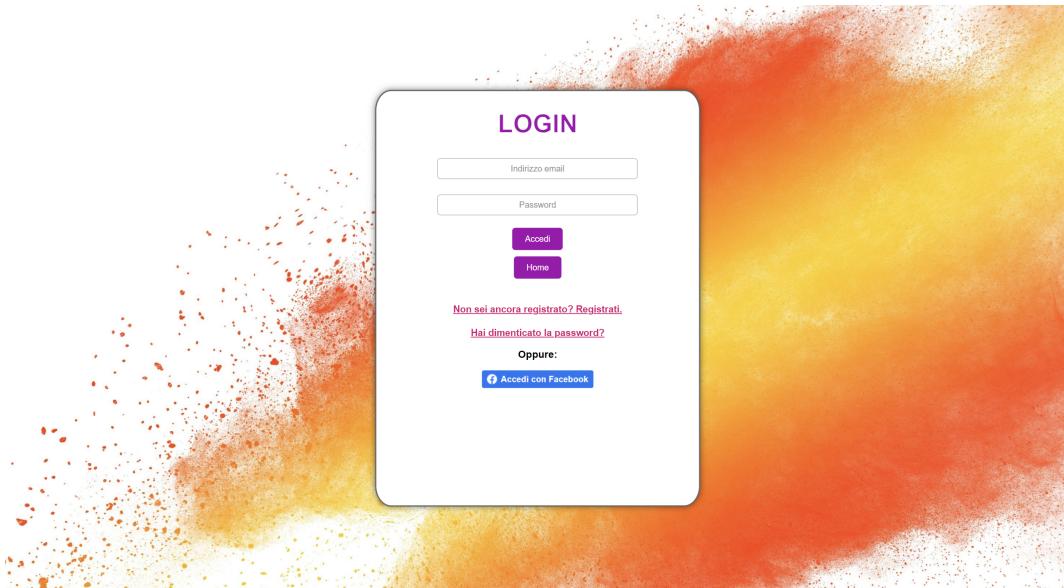


Figure 1.11: Pagina Login precedente

CHAPTER 1. INTRODUZIONE

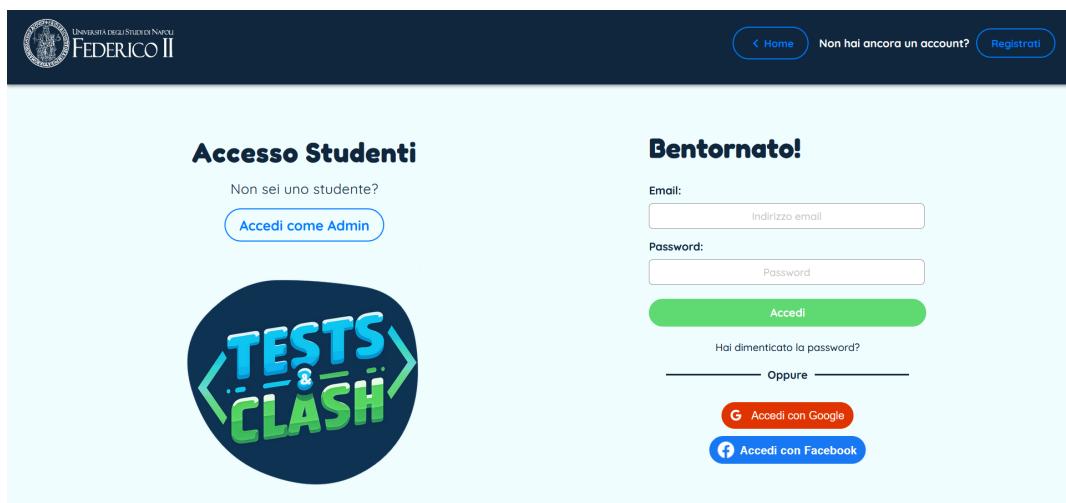


Figure 1.12: Pagina Login aggiornata

A screenshot of the previous user registration page. It features a purple header 'REGISTRAZIONE'. Below it is a form with six input fields: 'Nome', 'Cognome', 'Indirizzo e-mail', 'Password', 'Conferma Password', and a dropdown menu for 'BSc'. There is also a purple 'Invia' button at the bottom. A small note at the bottom right says 'Sei già registrato? Accedi.'

Figure 1.13: Pagina Registrazione nuovo utente precedente

CHAPTER 1. INTRODUZIONE

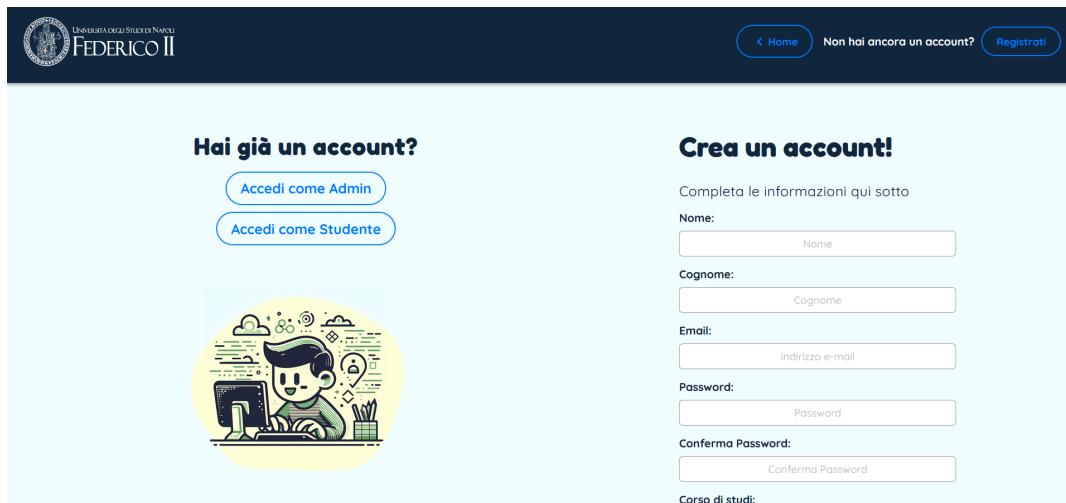


Figure 1.14: Pagina Registrazione nuovo utente aggiornata

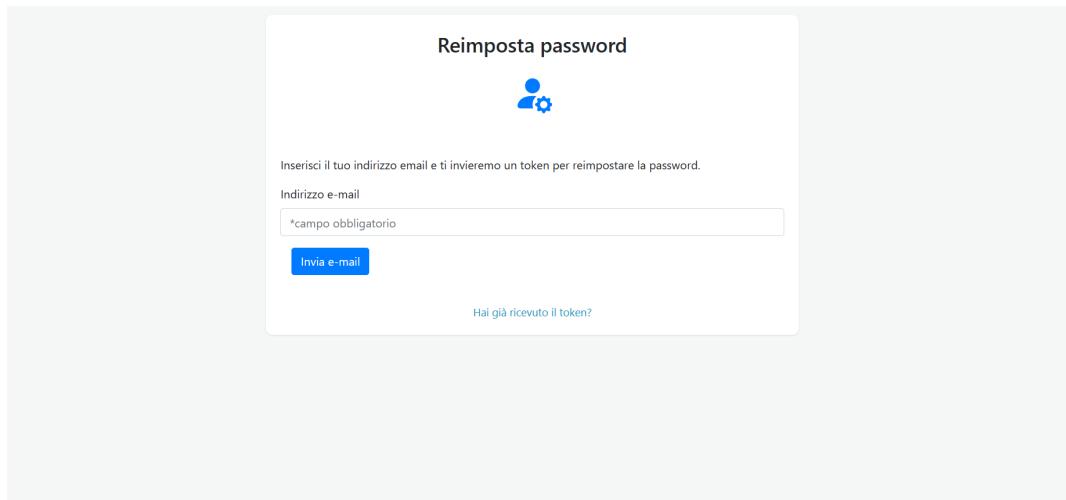


Figure 1.15: Pagina password dimenticata precedente

CHAPTER 1. INTRODUZIONE



Figure 1.16: Pagina password dimenticata aggiornata

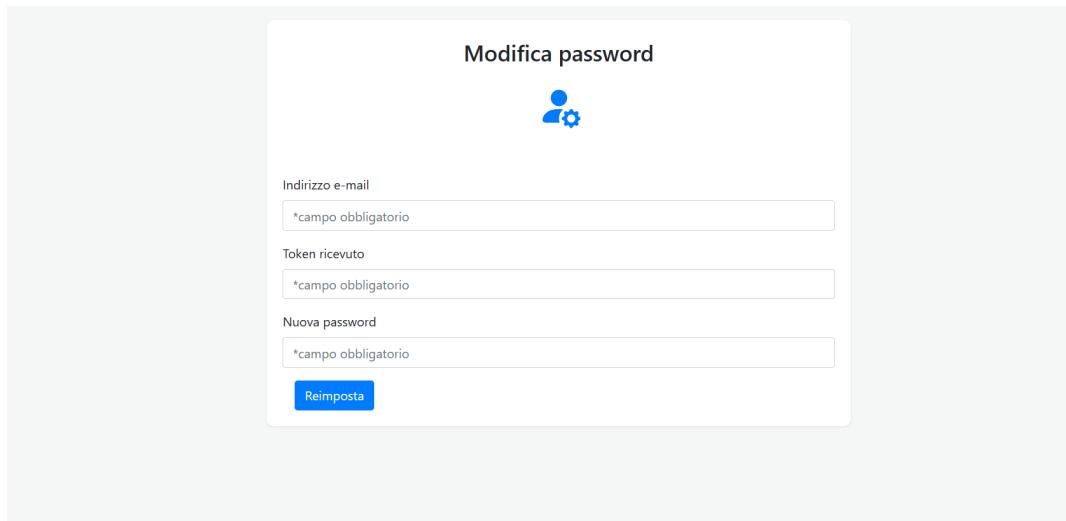


Figure 1.17: Pagina Modifica Password precedente

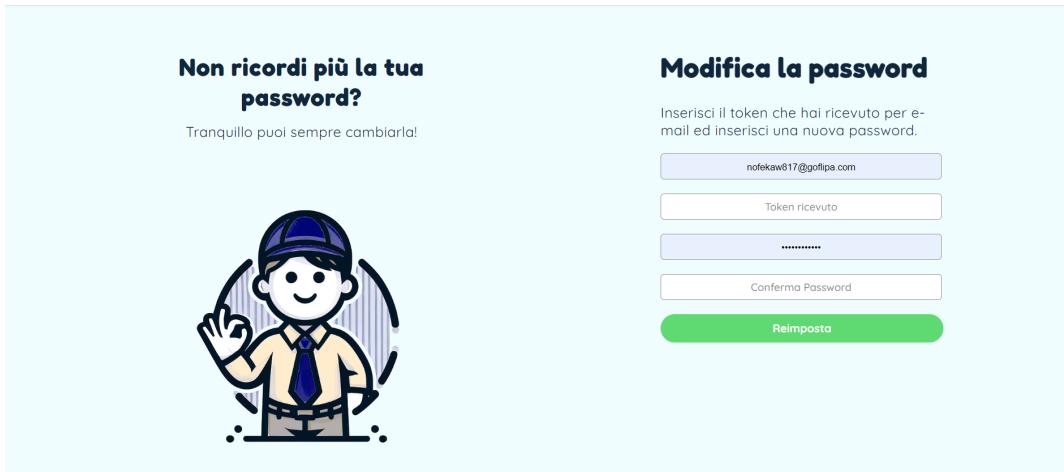


Figure 1.18: Pagina Modifica Password aggiornata

1.7 Metodologia di sviluppo

Il primo passo per la realizzazione dei task è stato quello di definire una metodologia di sviluppo.

Per la realizzazione di questo progetto abbiamo utilizzato una metodologia **AGILE** basato sul paradigma **SCRUM**, un framework per la gestione iterativa ed incrementale del ciclo di vita di uno sviluppo software.

Per la corretta gestione dei tasks assegnati, sono state effettuate quattro release: la prima è costituita da 3 sprint con cadenza settimanale, mentre la seconda, la terza e la quarta con due sprint di circa due settimane ciascuno.

Ogni release è stata organizzata al seguito di uno Sprint Planning in cui è stato deciso quali attività svolgere (Sprint Goal). Realizzato lo Sprint Backlog, per ogni attività viene specificata anche una stima della priorità.

Si noti che il team si è confrontato a cadenza regolare per sincronizzare le proprie attività e collaborare su particolari task.

Rispetto agli aspetti più tecnici dello sviluppo del software, un'altra metodologia AGILE presa in considerazione è l'**eXtreme Programming (XP)**, dal quale viene estratto un subset di pratiche che possono essere usate insieme a metodi di sviluppo agile più focalizzati sulla gestione del progetto come SCRUM:

- User Stories: i requisiti sono espressi come storie utente che il team dividerà in task da implementare.
- Pair Programming: permette un processo di revisione informale del codice, dal momento che ogni linea di codice viene controllata da più persone, e supporta il refactoring. Le coppie possono variare di continuo.
- Refactoring: svolgere una serie di cambiamenti che non modificano il comportamento, ma mantengono alta la qualità del codice.

1.7.1 Release

Release 1

Durante il primo Sprint sono stati stabiliti gli obiettivi della release, concentrandoci principalmente sullo studio dell'architettura del sistema di partenza. Una volta apprese le informazioni necessarie, la prima release è dedicata all'analisi dei requisiti, user stories e i criteri di accettazione.

CHAPTER 1. INTRODUZIONE

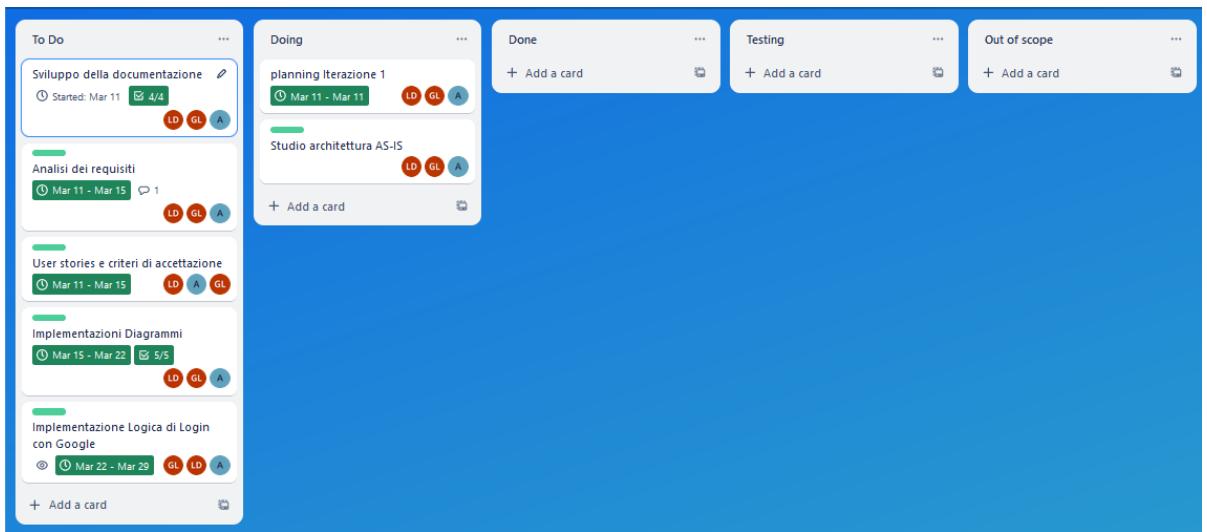


Figure 1.19: Release 1 - Sprint 1

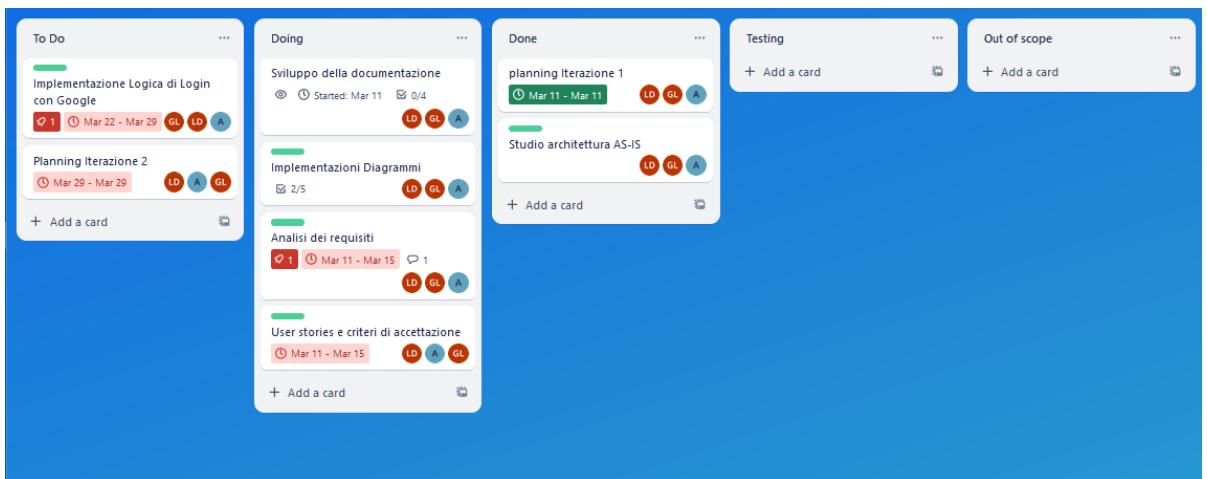


Figure 1.20: Release 1 - Sprint 2

CHAPTER 1. INTRODUZIONE

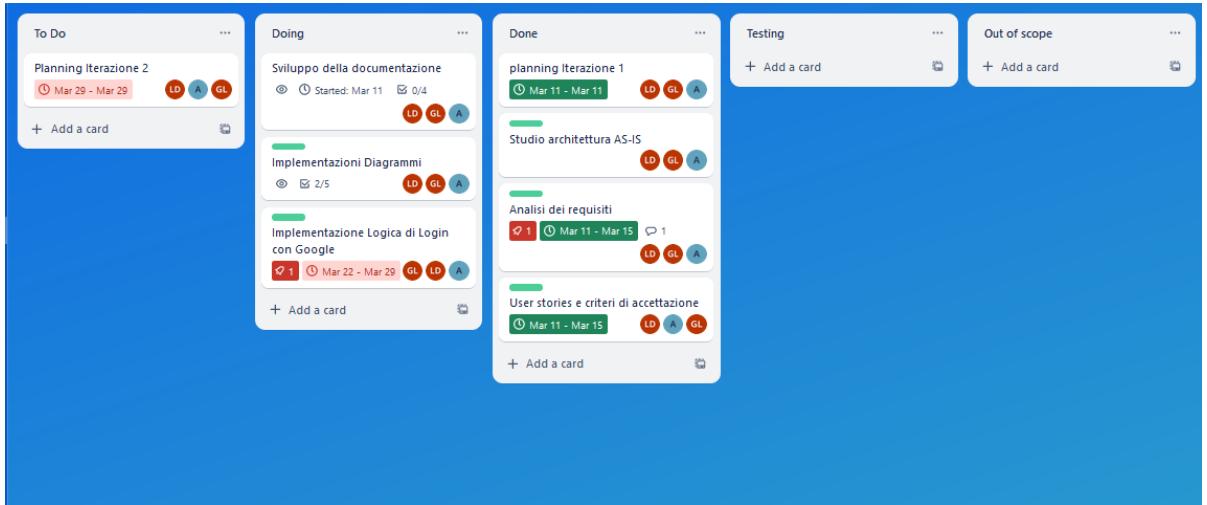


Figure 1.21: Release 1 - Sprint 3

Release 2

Nella seconda release viene portato a termine il task con maggiore priorità, il Login con Google, testando di conseguenza la funzionalità una volta implementata e di nuovo dopo aver gestito il logout.

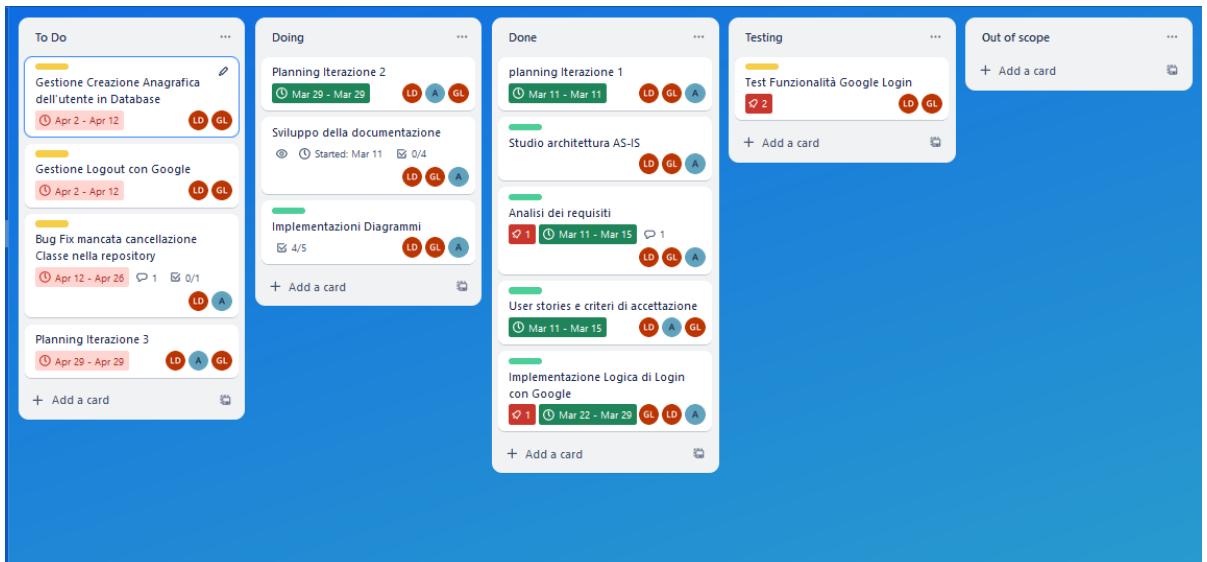


Figure 1.22: Release 2 - Sprint Planning

CHAPTER 1. INTRODUZIONE

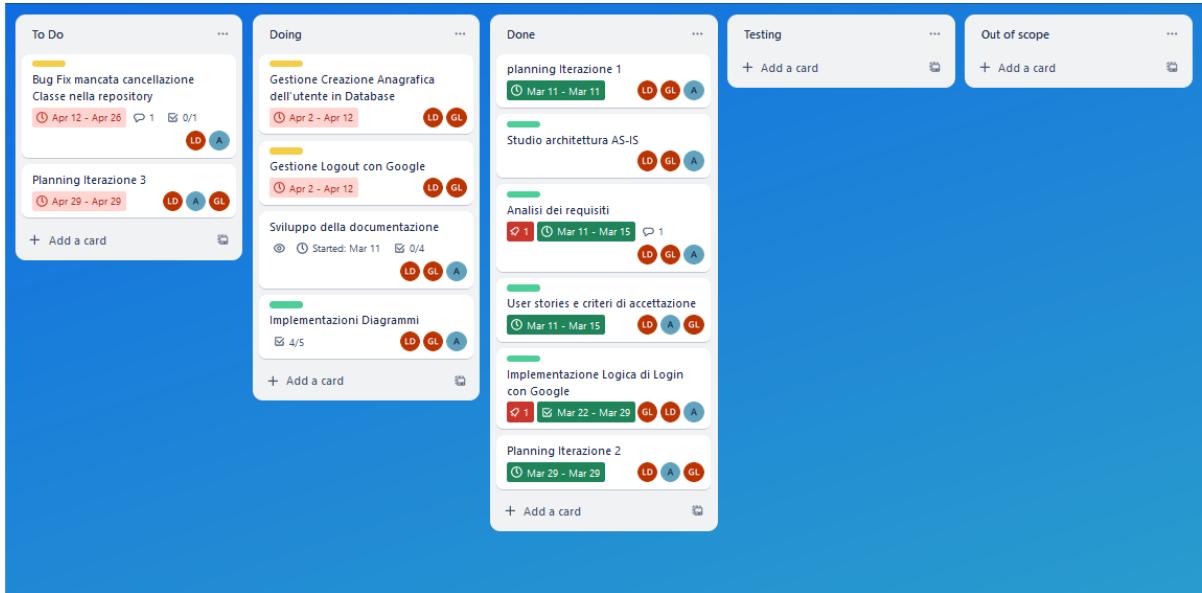


Figure 1.23: Release 2 - Sprint 1

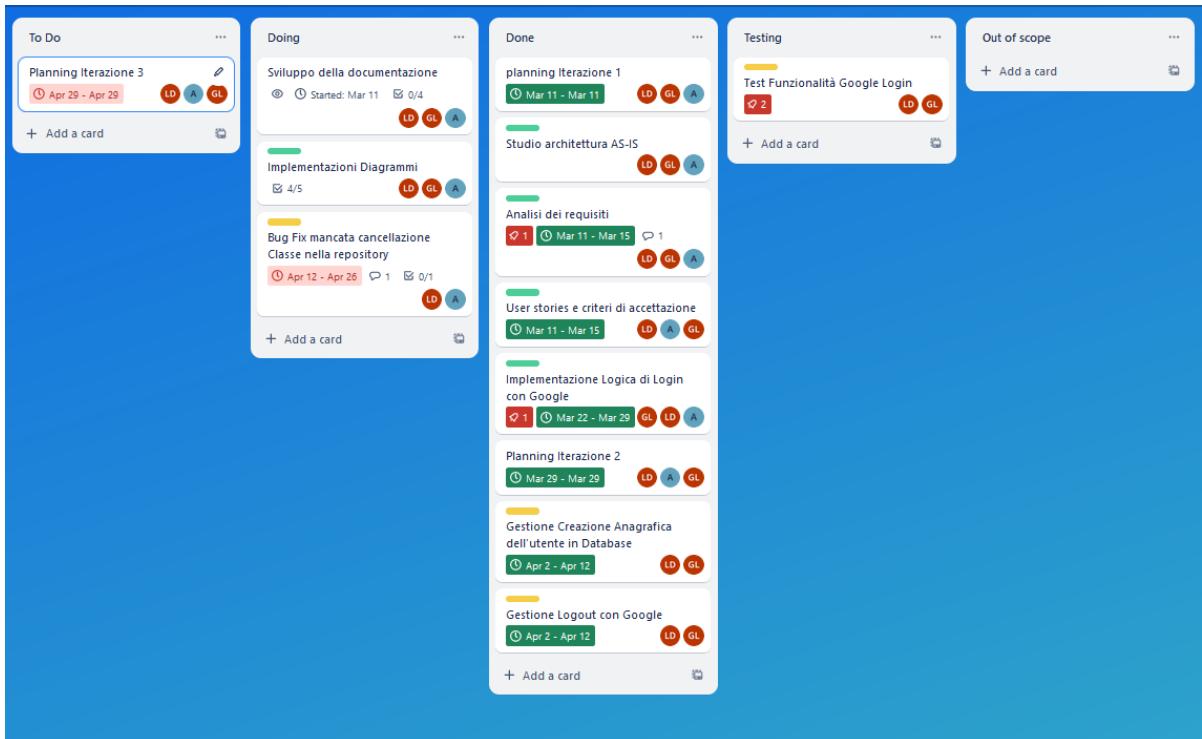


Figure 1.24: Release 2 - Sprint 2

Release 3

Durante la terza release il team ha posto l'attenzione alla risoluzione dei bug e issue identificati, così come alla realizzazione del refactoring, ossia dei task definiti a priorità media.

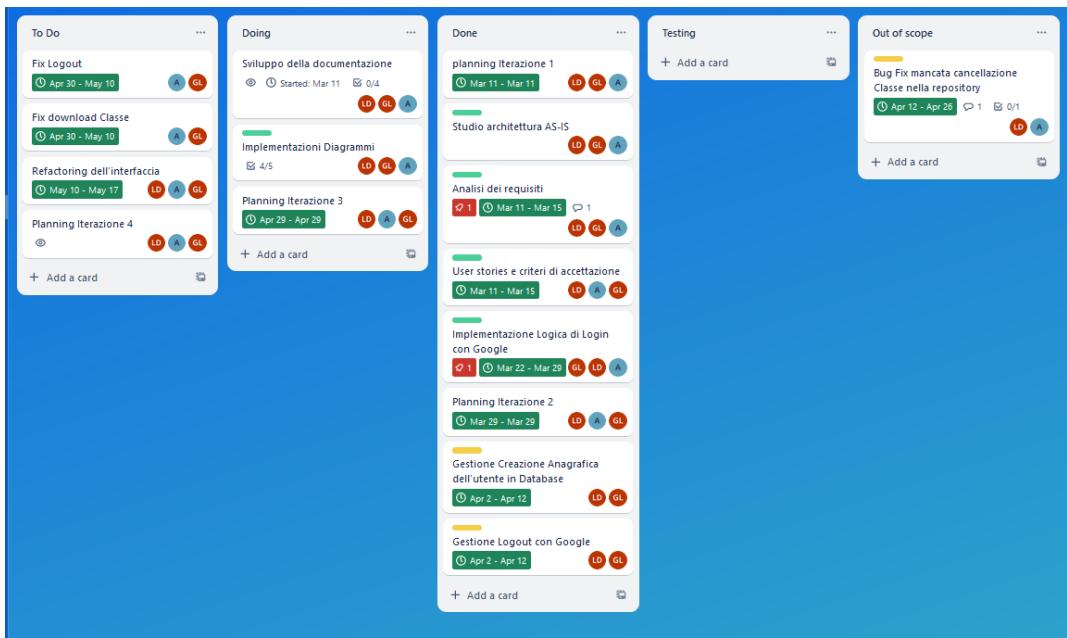


Figure 1.25: Release 3 - Sprint Planning

CHAPTER 1. INTRODUZIONE

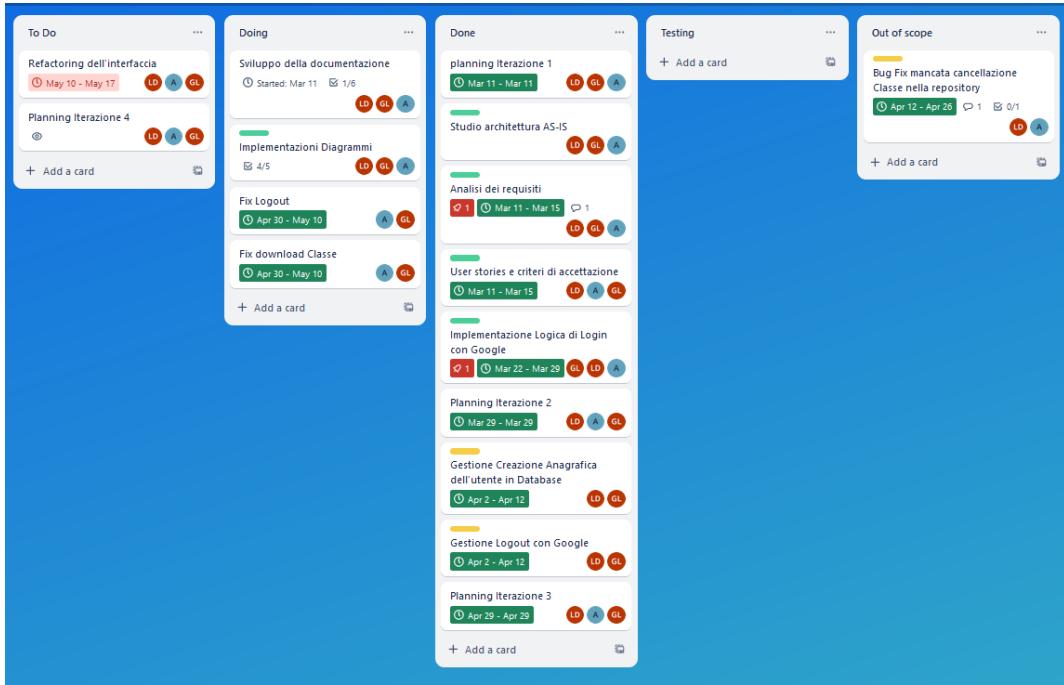


Figure 1.26: Release 3 - Sprint 1

CHAPTER 1. INTRODUZIONE

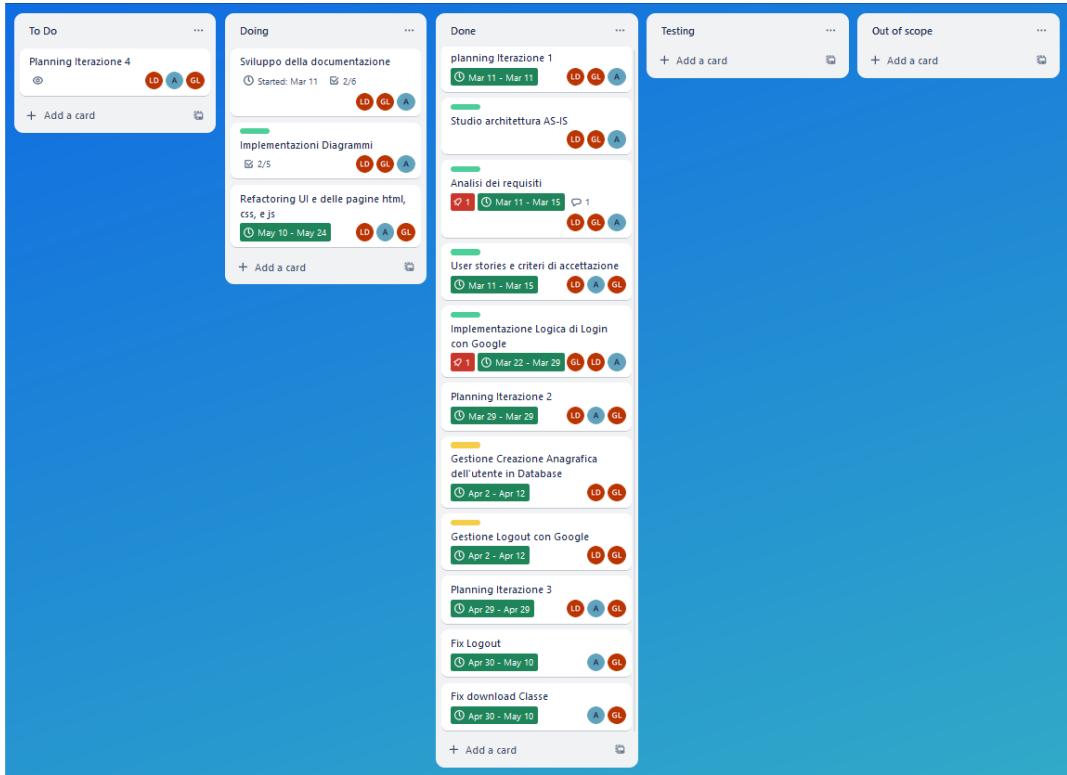


Figure 1.27: Release 3 - Sprint 2

Release 4

Nella quarta release il team si è riunito per la risoluzione del bug emerso durante i test delle funzionalità precedenti, relativo alla registrazione di un nuovo giocatore.

CHAPTER 1. INTRODUZIONE

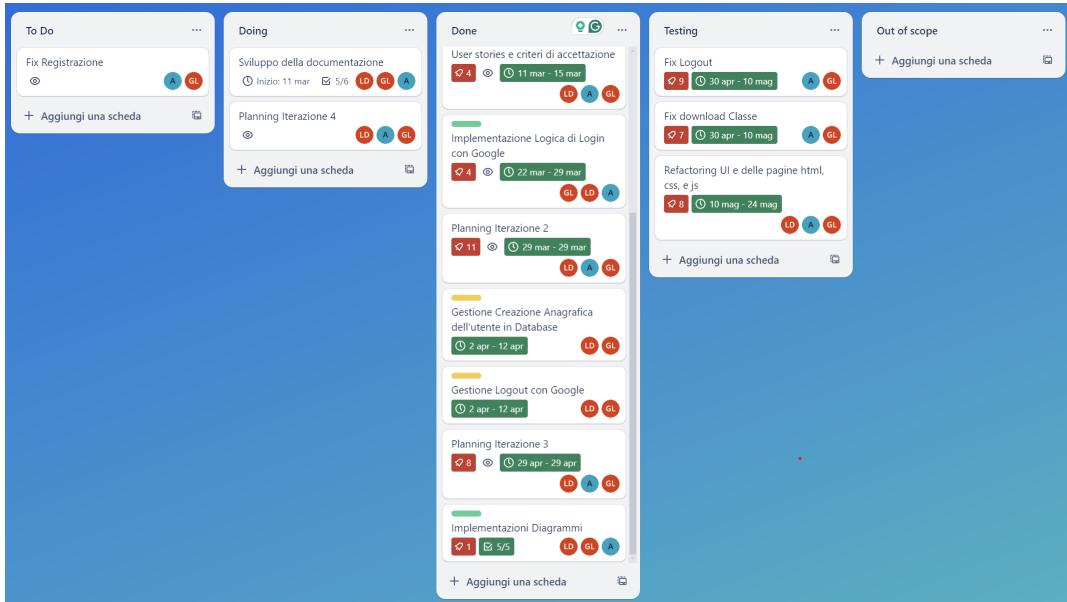


Figure 1.28: Release 4 - Sprint Planning

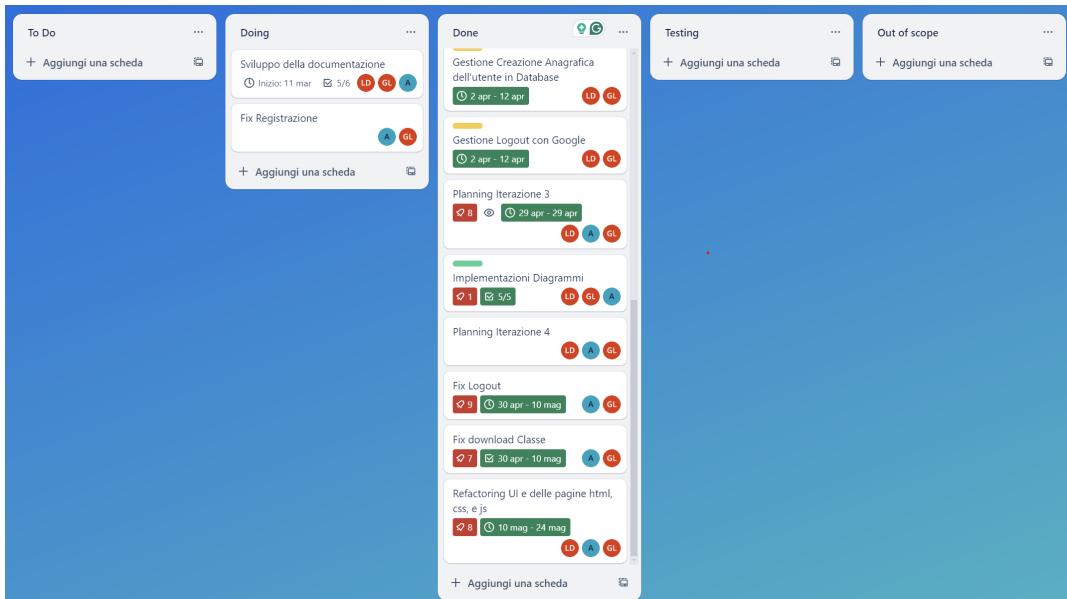


Figure 1.29: Release 4 - Sprint 1

CHAPTER 1. INTRODUZIONE

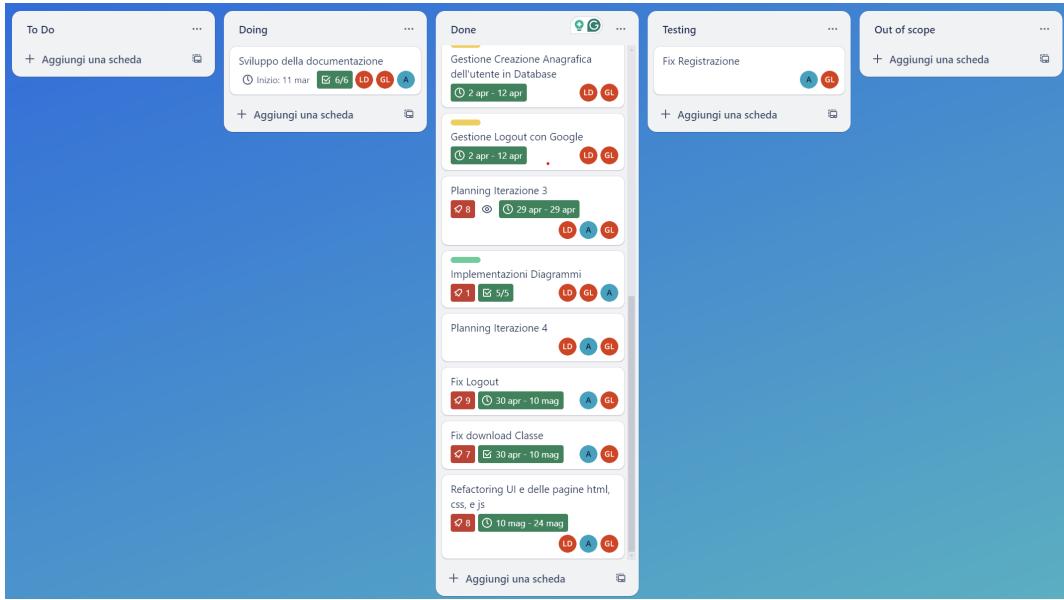


Figure 1.30: Release 4 - Sprint 2

Chapter 2

Processo di Sviluppo

2.1 Strumenti Software di Supporto

- **Strumenti per la comunicazione del team:** si è utilizzato Microsoft Teams per la comunicazione del team, il quale ci ha permesso di pianificare le riunioni, comunicare in tempo reale, scambiare rapidamente file ed informazioni e risolvere problemi in modo collaborativo.



Figure 2.1: Microsoft Teams

- **Strumenti per l'organizzazione di attività del progetto:** si è utilizzato Trello, il quale ci ha permesso di avere: una gestione visuale delle attività, flessibilità e adattabilità e collaborare in tempo reale.



Figure 2.2: Trello

- **Strumenti per la modellazione dei diagrammi:** si è utilizzato draw.io 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.



Figure 2.3: Draw.io



Figure 2.4: Visual Paradigm

- **Strumenti per il design dell'interfaccia:** è stato utilizzato DALL-E per la realizzazione dei loghi e immagini presenti nella nuova interfaccia attraverso l'utilizzo di prompt.



Figure 2.5: DALL-E

- **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. Per il deployment dell'applicazione, è stato utilizzato Docker, una piattaforma di containerizzazione che ha semplificato la distribuzione dell'applicazione su server.



Figure 2.6: Eclipse



Figure 2.7: VS Code



Figure 2.8: Docker

- **Strumenti di sviluppo di applicazioni web:** Il framework SpringBoot è un framework open-source basato su Java progettato per semplificare il processo di sviluppo di applicazioni web. Viene utilizzato per sfruttare le sue caratteristiche come configurazione automatica e gestione delle dipendenze.

È stato usato il modulo Spring Boot Security OAuth2 che fornisce funzionalità per l'integrazione del protocollo di autorizzazione OAuth2 e OpenID Connect nelle applicazioni Spring Boot. Questo modulo semplifica notevolmente il processo di connessione dell'applicazione con i provider OAuth2 come Google, Facebook e GitHub per la funzionalità di social login. Gestisce inoltre il complesso processo di autorizzazione e scambio di token OAuth2. Infine, si integra perfettamente con Spring Security, offrendo robuste funzionalità di sicurezza.



Figure 2.9: Spring Boot

- **Strumenti di gestione del progetto:** Git è stato utilizzato come sistema di controllo versione per il progetto. GitHub invece 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, al fine di contribuire al codice e gestire problemi (issues).



Figure 2.10: Github

- **Strumenti per il testing:** Postman è uno strumento software che permette di effettuare testing sulle applicazioni web e relative API effettuando richieste API ed ispezionando i dati di richiesta e risposta. Può gestire diversi tipi di formati di risposta come JSON, XML, HTML e testo semplice.



Figure 2.11: Postman

Questi strumenti hanno permesso di lavorare in modo collaborativo e di gestire il progetto in modo efficiente e produttivo.

Chapter 3

Specifiche di Progetto

3.1 Utenti del sistema

- **Studenti:** Gli studenti sono gli utenti principali del sistema, che possono accedere al catalogo di classi di programmazione, cercare, visualizzare, filtrare e scaricare le classi disponibili. Anche chiamato "giocatore" o "player", è l'utente registrato che sfida un Robot.
- **Amministratori:** Gli amministratori sono anch'essi utenti registrati, ma responsabili della gestione del catalogo di classi di programmazione. Hanno accesso privilegiato per inserire nuove classi e/o test, 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.

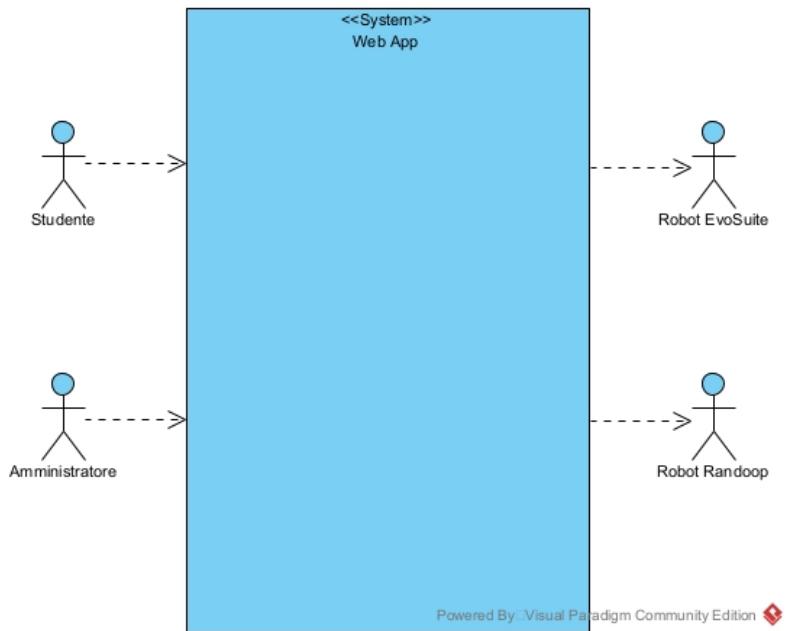


Figure 3.1: Top level Context Diagram

3.2 Glossario dei Dati

Oltre alle caratterizzazioni degli utenti del sistema, vengono definiti i seguenti termini significativi.

- **Robot**: I Robot dell'applicazione sono Randoop ed Evosuite, che rappresentano strumenti di generazione automatica di test JUnit, ed in particolare, gli avversari dei giocatori.
- **Game**: Entità del sistema composta da più Round. Quando uno studente avvia una partita, seleziona il Robot avversario e la classe da sottoporre a test.
- **Round**: Entità che contiene le informazioni di gioco. Ogni Game è composto da più Round, durante i quali ciascun giocatore sottopone un risultato.
- **Classe**: Artefatto software sottoposto a test, i quali verranno associati agli score del singolo game.

3.3 Architettura a Microservizi

Per lo sviluppo dell'intero sistema si è optato per un'architettura a **microservizi**, architettura importante soprattutto per il modo in cui essa influisce sui requisiti di qualità, infatti consente di: garantire una maggiore agilità ai vari team, che godono della possibilità di lavorare sulla progettazione, sviluppo, testing e distribuzione di un piccolo componente alla volta in autonomia e indipendentemente dai team che si occupano dello sviluppo degli altri servizi; aumentare la scalabilità del sistema ad un livello di granularità più fine, grazie alla possibilità di creare più istanze dei singoli servizi; migliorare la tolleranza agli errori poiché sono unità isolate a contenere gli errori.

L'architettura a microservizi utilizza i servizi come unità di modularità e ognuno di essi ha un'API che rappresenta un confine, in modo da non permettere l'accesso alle classi interne, e che permette la comunicazione tra servizi *loosely coupled*, che hanno una propria architettura esagonale.

Per quanto riguarda l'integrazione di tali microservizi viene utilizzato il **Gateway Pattern** che consiste nell'introdurre dei componenti che costituiscono gli unici punti di accesso ai servizi a disposizione tramite un'interfaccia (come il Design Patter Facade), la quale nasconde all'esterno i dettagli implementativi. Il Gateway gestisce le richieste provenienti dall'utente ed esegue tutte le operazioni necessarie per restituire i risultati.

Il vero punto di ingresso del sistema è l'UI Gateway (implementato tramite Nginx), infatti l'utente interagisce esclusivamente mediante questo componente, che gestisce le richieste di servizi, per poi dirigerle verso l'API Gateway, e anche di pagine web. Se il client richiede una pagina web, l'UI Gateway, dopo aver consultato la propria tabella di routing, effettua la richiesta al microservizio necessario e inoltra la risposta.

L'API Gateway si occupa invece del re-routing delle richieste verso lo specifico servizio e, inoltre, gestisce l'autenticazione dell'utente tramite token, verificando non solo che

l'utente sia correttamente registrato, ma abbia anche effettuato l'accesso, in modo da poter richiedere determinati servizi.

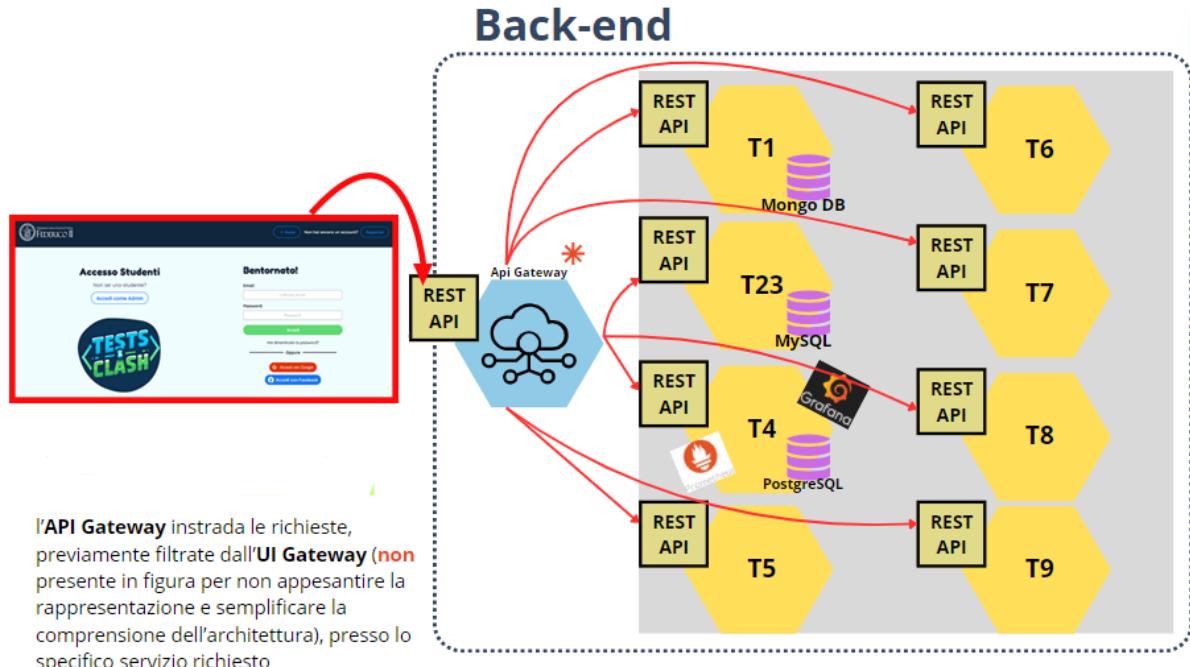


Figure 3.2: Architettura a microservizi

3.4 Requisiti Funzionali

I requisiti funzionali sono le specifiche delle funzionalità e dei comportamenti attesi che un sistema software deve essere in grado di fornire. In altre parole, descrivono ciò che il software deve fare per soddisfare le esigenze dell'utente.

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

1. Autenticazione Google per gli Studenti:

- Creare l'account dello studente nell'applicazione web utilizzando le informazioni del loro profilo Google
- Consentire agli studenti di accedere utilizzando il proprio account Google,

concedendo loro l'accesso alle funzionalità e alle risorse specifiche per gli studenti dell'applicazione web

2. Correzione Bug - Logout:

- Garantire che quando lo studente effettua il logout, il token associato alla sua sessione venga correttamente eliminato

3. Correzione Bug - Registrazione:

- Gestire eventuali errori di registrazione in modo più efficiente

Requisiti funzionali	Stato d'avanzamento
Il sistema deve consentire ad ogni giocatore di potersi registrare nel sistema inserendo: nome, cognome, e-mail, password e corso di studi	
Il sistema, in caso di avvenuta registrazione, deve inviare al giocatore registrato un'e-mail contenente un identificativo univoco (ID)	
Il sistema deve permettere al player registrato di effettuare il login	
In caso di login errato, il sistema deve restituire un errore	
Il sistema deve permettere al player registrato di impostare una nuova password nel caso in cui avesse dimenticato la propria	
Il sistema deve permettere al player registrato di accedere all'area riservata di selezione dei parametri di gioco solo dopo aver effettuato il login	
Il sistema deve permettere al player registrato di effettuare il logout	
Il sistema, per ogni sessione di autenticazione correttamente effettuata, deve assegnare al giocatore un token di autenticazione	
Il sistema deve consentire al player di inserire un nuovo "like" ad una classe	
Il sistema deve consentire al player di inserire un nuovo report ad una classe	

Table 3.1: Tabella dello stato d'avanzamento complessivo dei requisiti funzionali definiti per il task T23 - *In rosa i requisiti implementati; in giallo quelli non ancora integrati*

4. Correzione Bug - Eliminazione Classe:

- Assicurarsi che quando un amministratore elimina una classe, questa venga completamente rimossa dal file system

5. Correzione Bug - Dowload Classe:

- Consentire agli amministratori di poter scaricare correttamente i file associati alle classi

Requisiti funzionali	Stato d'avanzamento
L'applicazione deve mostrare agli admin un elenco di tutte le classi di programmazione	Verde
L'applicazione deve consentire all'admin di scaricare il codice di una classe selezionata	Rosso
L'applicazione deve consentire all'admin di inserire una nuova classe con nome, descrizione e codice	Verde
L'applicazione deve consentire all'admin di cancellare una classe esistente selezionata	Rosso
L'applicazione deve consentire all'admin di cercare una classe per nome, attraverso un campo di ricerca	
L'applicazione deve consentire all'admin di scegliere e cambiare i criteri di ordinamento delle classi visualizzate	Verde
L'applicazione deve consentire all'admin di selezionare una o più etichette per filtrare le classi visualizzate	Verde
L'applicazione deve consentire all'admin di modificare una classe esistente, inclusi nome, descrizione e codice ¹	Giallo
L'applicazione deve consentire all'admin di registrarsi inserendo le proprie informazioni personali (nome, cognome, username e password)	Verde
L'applicazione deve consentire all'admin di effettuare il login al sistema inserendo il proprio username e password	Verde
L'applicazione deve consentire agli admin di visualizzare il numero di "likes" per ogni classe disponibile	Giallo
L'applicazione deve consentire all'admin di visualizzare l'elenco dei report effettuati dai players sulle classi	
L'applicazione deve consentire all'admin di visualizzare la classifica di tutti i giocatori correttamente iscritti	Giallo

Table 3.2: Tabella dello stato d'avanzamento complessivo dei requisiti funzionali definiti per il task T1 - *In rosa i requisiti implementati; in giallo quelli non ancora integrati*

3.5 Storie Utente

Descrivono in modo semplice una funzionalità o un requisito che è di valore per un utente del sistema. Le storie utente, anche note come *User Story*, sono utilizzate tramite il template:

As a < type of user >

I want < to perform some task >

so that I can < achieve some goal/benefit/value>

Storia dell'Utente (Priorità: Alta) - Autenticazione Google per gli Studenti

Come studente, **voglio** poter accedere all'applicazione web utilizzando il mio account Google, **così da** poter accedere alle funzionalità dell'applicazione senza dover creare e ricordare un nome utente e una password separati.

Storia dell'Utente (Priorità: Media) - Correzione Bug - Logout

Come studente autenticato, **voglio** poter uscire dal mio account e dall'applicazione, **così da** garantire la sicurezza del mio account quando termine la sessione di utilizzo.

Storia dell'Utente (Priorità: Media) - Correzione Bug - Registrazione

Come studente, **voglio** correggere le informazioni di registrazione nella stessa pagina, **così da** non dover reinserire tutti i campi del form.

Storia dell'Utente (Priorità: Media) - Correzione Bug - Eliminazione Classe

Come admin, **voglio** poter eliminare le classi caricate sia dall'applicazione web sia dal file system, **così da** poter gestire efficientemente le mie classi e evitare confusione.

Storia dell'Utente (Priorità: Media) - Correzione Bug - Download Classe

Come admin, **voglio** poter scaricare i file associati alle classi, **così da** poter consultare il file sorgente.

¹Nella versione corrente dell'applicazione, non è possibile modificare il codice di una classe già caricata, ma è possibile aggiornare le seguenti informazioni: nome, data, difficoltà e descrizione.

Priorizzando le storie degli utenti, il team di sviluppo può concentrarsi prima sull'implementazione della funzionalità di Autenticazione Google, poiché ha un maggiore impatto sull'esperienza utente. Una volta completata, il team può affrontare la correzione dei bug.

3.6 Criteri di Accettazione

I criteri di accettazione sono dettagliati utilizzando il modello GIVEN-WHEN-THEN, per determinare quando una storia è completa e controllare che il sistema soddisfi i suoi requisiti.

1. Autenticazione Google per gli studenti:

GIVEN che uno studente si trova nella pagina di login della web-app

WHEN fa clic sul pulsante "Accedi con Google"

THEN dovrebbe essere reindirizzato alla pagina di autorizzazione OAuth 2.0 di Google.

GIVEN che uno studente ha concesso il permesso alla web-app di accedere alle informazioni del proprio account Google

WHEN Google restituisce un codice di autorizzazione alla web-app

THEN la web-app dovrebbe utilizzare il codice di autorizzazione che riceve per accedere alle informazioni tramite un token di accesso.

GIVEN che la web-app ha ricevuto un token di accesso valido

WHEN richiede le informazioni del profilo Google dello studente

THEN dovrebbe ricevere le informazioni necessarie, come nome e indirizzo email.

GIVEN che la web-app ha recuperato le informazioni del profilo Google dello studente

WHEN crea o aggiorna l'account dello studente

THEN lo studente dovrebbe essere autenticato e avere accesso alle funzioni e

risorse specifiche per gli studenti della web-app.

2. Risoluzione del bug - Eliminazione delle classi:

GIVEN che un admin seleziona una classe Java da eliminare

WHEN conferma l'eliminazione

THEN la classe dovrebbe essere rimossa dall'interfaccia della web-app e il file della classe dovrebbe essere eliminato dal file system.

GIVEN che un admin ha eliminato una classe Java

WHEN controlla il file system

THEN il file della classe non dovrebbe più essere presente nel file system.

3. Risoluzione del bug - Download delle classi:

GIVEN che un admin seleziona una classe Java da scaricare

WHEN conferma il download

THEN la classe dovrebbe essere scaricata nella repository di download dell'admin.

GIVEN che un admin ha effettuato il download di una classe

WHEN l'admin accede al contenuto del file

THEN dovrebbe poter visualizzare il file correttamente senza errori al suo interno.

4. Risoluzione del bug - Logout:

GIVEN che uno studente autenticato si trovi all'interno della web-app

WHEN fa clic sul pulsante "Logout"

THEN dovrebbe essere re-indirizzato alla pagina di Login e i token associati alla sua sessione dovrebbero essere eliminati correttamente.

5. Risoluzione del bug - Registrazione:

GIVEN che uno studente compila in modo non conforme uno o più campi della form

WHEN fa clic sul pulsante "Invia"

THEN dovrebbe visualizzare un pop up di errore nella stessa pagina di registrazione

3.7 Requisiti non funzionali

I requisiti non funzionali sono caratteristiche di un sistema che non riguardano le funzionalità specifiche, ma piuttosto le qualità del sistema stesso. In riferimento al sistema prodotto è possibile osservare i seguenti requisiti di qualità:

1. Sicurezza:

- Una volta stabilita la connessione, il server assegna all'utente un token criptato valido per tutta la durata della sessione, per rappresentare in modo sicuro l'identità dei players quando si comunica in rete senza la necessità di dover interrogare il database.
- Proteggere i dati degli utenti e garantire la privacy delle informazioni personali: viene utilizzata la classe *BCryptPasswordEncoder*, fornita dal framework Spring Security, per crittografare le password memorizzate inserite durante l'autenticazione dagli utenti.
- Implementare controlli di accesso e autorizzazione per garantire che gli utenti possano accedere solo alle risorse e alle funzionalità che gli sono consentite.

2. Scalabilità:

- Decomporre una architettura in più microservizi offre la possibilità di scalare l'architettura ad un livello di granularità più fine: è possibile scalare istanze di singoli microservizi.
- Documentando costantemente le modifiche al sistema si semplifica il coinvol-

gimento di nuovi sviluppatori all'interno del progetto.

3. Usabilità:

- Creare un'interfaccia utente intuitiva e facile da usare per gli studenti e gli amministratori.
- Garantire che l'applicazione sia utilizzabile dagli utenti con diversi livelli di competenza tecnica.
- Uso di un browser come client che implica l'assenza di dipendenze o programmi esterni da installare da parte dell'utente.
- Le notifiche di errore devono essere visibili e facilmente comprensibili dagli utenti.

4. Interoperabilità:

- La capacità dell'applicazione di integrarsi con altri sistemi o servizi, come servizi di autenticazione esterni.

5. Deployability:

- Componenti lasciamenti accoppiati, come quelli delle architetture a microservizi, supportano frequenti release, non dovendo ricompilare tutto il sistema.

6. Manutenibilità:

- L'utilizzo delle documentazioni fornite dai gruppi coinvolti su GitHub facilita l'aggiornamento sulle ultime versioni e dunque la manutenzione dell'applicazione.
- La modularità del codice facilita l'aggiunta di nuove funzionalità e la modifica di quelle esistenti.
- Il refactoring delle pagine HTML e CSS deve seguire le linee guida per garantire che siano facilmente aggiornabili.

7. Prestazioni:

- Ridotti tempi di caricamento delle pagine e dei contenuti grazie ad un'operazione di refactoring delle singole pagine HTML, implementata attraverso l'utilizzo di un file CSS centralizzato.
- Gli elementi comuni come header e footer vengono ora caricati da file individuali in modo da velocizzare il rendering della pagina da parte del browser.
- Gestione degli errori di registrazione eseguita in tempo reale e con un feedback adeguato.

Chapter 4

Fase di Analisi

4.1 Class Diagram

I Class Diagram sono un tipo di diagramma UML (Unified Modeling Language) che in fase di progetto consentono di rappresentare le classi e le relazioni fra esse, a prescindere da come saranno rappresentate in archivi.

Il diagramma mostra una vista statica del modello, mettendo in evidenza gli attributi e le operazione degli elementi, ma non mostra informazioni temporali.

Il diagramma delle classi sviluppato rappresenta un dettaglio del task T23, relativo al Login dell'utente ed in particolar modo di come sia ora possibile accedere tramite account Google ([1.1 Task: Google Login](#)).

È stato necessario modificare classi pre-esistenti come `User` e `UserRepository` in seguito all'aggiunta del flag `"isRegisteredWithGoogle"`. Analogamente, è stato ampliato il `Controller` per permettere il nuovo metodo di Login.

In colore diverso, invece, sono evidenziate le classi aggiunte, che comprendono le librerie, le informazioni associate a Google e il modulo Spring Boot Security OAuth2 che

semplifica il processo di connessione della web-app con Google, gestendo il processo di autorizzazione, lo scambio di token e funzionalità di sicurezza.

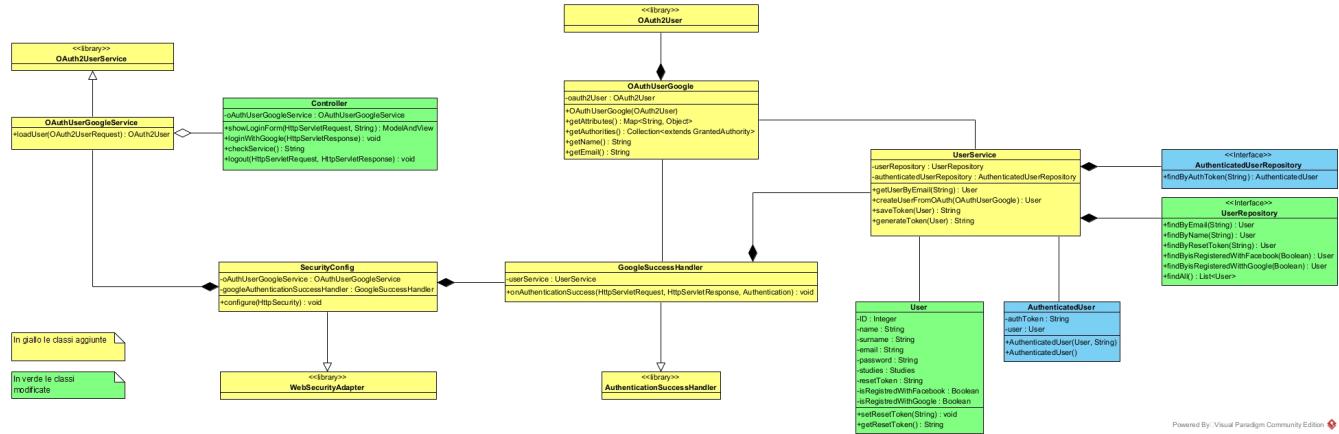


Figure 4.1: Class Diagram - dettaglio di T23

4.2 Use Case Diagram

Un diagramma dei casi d'uso è un diagramma UML che rappresenta visivamente le interazioni tra gli utenti (o "attori" in questo caso) e un sistema. Acquisisce i requisiti funzionali di un sistema e illustra il modo in cui i diversi utenti interagiscono con il sistema per raggiungere obiettivi specifici attraverso collezioni di scenari correlati, sia di successo che di fallimento.

Per chiarire la rappresentazione e mostrare meglio cosa fa effettivamente il sistema, compresi i servizi forniti e chi sono gli attori interagenti, realizziamo due diagrammi dei casi d'uso:

4.2.1 Casi d'uso relativi ai task T23

A partire dal diagramma dei casi d'uso che mostrava soltanto la possibilità di registrazione e accesso tramite compilazione del form o le credenziali del proprio account Facebook, è stata aggiunta la possibilità di **utilizzare il proprio account Google** per compiere tali

azioni ([1.1 Task: Google Login](#)).

Per una migliore descrizione dei casi d'uso, sono stati aggiunti come attori secondari: **GoogleAuth Provider**, per completare i casi d'uso appena aggiunti; **Facebook Provider** per perfezionare il caso d'uso preesistente.

Inoltre, così come da task assengato, è stato modificato il caso d'uso relativo al **Logout** ([1.4 Task: Bug fix Logout da parte del player](#)), il quale non cancellava correttamente la sessione, e **Registrazione** ([1.5 Task: Bug fix Registrazione da parte del player](#)).

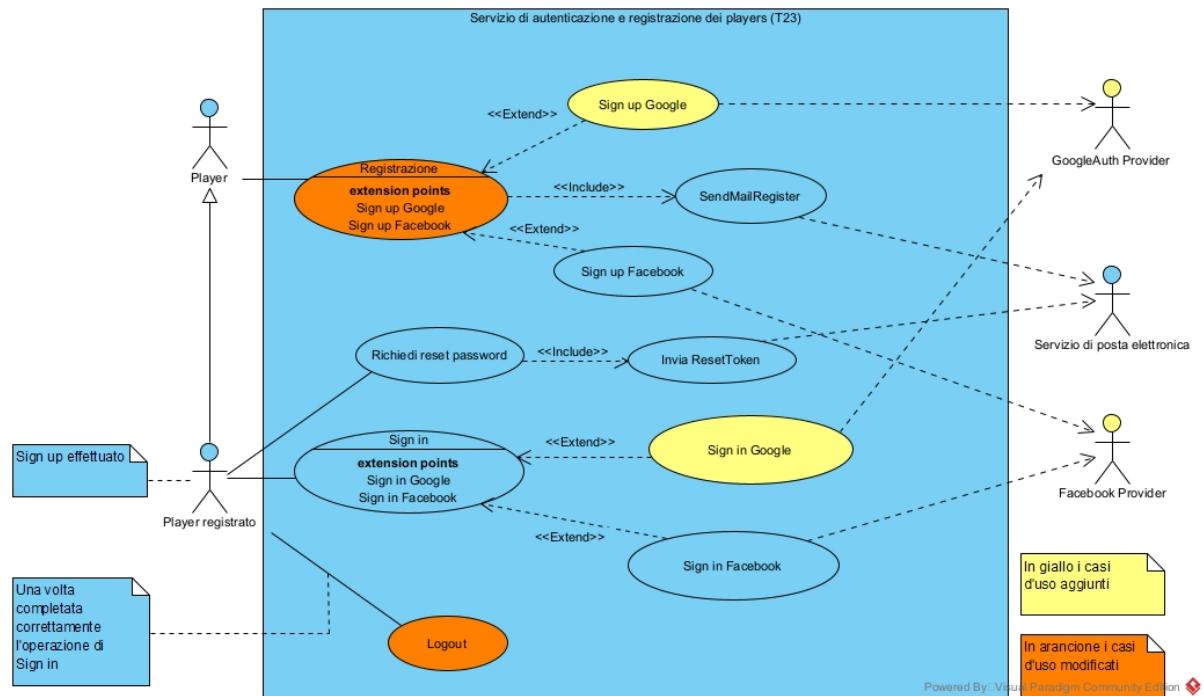


Figure 4.2: Use Case Diagram - Task T23

Sign up Google

- Nome: Sign up Google
- Descrizione: Permette ad un player di effettuare la registrazione all' applicazione tramite Google

- Pre-Condizioni: Il player deve possedere un account Google, non deve essere ancora registrato e si deve trovare nella pagina di registrazione
- Flusso principale:
 1. Il player clicca sul pulsante "Accedi con Google"
 2. Il player compila il form di autenticazione di Google inserendo: e-mail (o numero di telefono), password e clicca sul pulsante "Accedi"
 3. Il player viene re-indirizzato alla propria area di gioco
- Flussi alternativi:
 - 2.a Impossibile trovare il tuo Account Google
 1. Il player inserisce in modo errato la propria email nel form
 2. Il sistema mostra a video un messaggio di errore
 - 2.b Password scorretta
 1. Il player inserisce in modo errato la propria password nel form
 2. Il sistema mostra a video un messaggio di errore
- Post-Condizioni: Il player ha completato la registrazione al sistema e l'account è stato registrato
- Attore primario: Player
- Attore secondario: GoogleAuth Provider
- Casi d'uso estesi: Registrazione

Sign in Google

- Nome: Sign in Google

- Descrizione: Permette ad un player registrato di accedere all'applicazione tramite il proprio account Google
- Pre-Condizioni: Il player deve possedere un account Google, deve essere già registrato e si deve trovare nella pagina di login dedicata
- Flusso principale:
 1. Il player clicca sul pulsante "Accedi con Google"
 2. Il player compila il form di autenticazione di Google inserendo: e-mail (o numero di telefono), password e clicca sul pulsante "Accedi"
 3. Il player viene re-indirizzato alla propria area di gioco
- Flussi alternativi:
 - 2.a Impossibile trovare il tuo Account Google
 1. Il player inserisce in modo errato la propria email nel form
 2. Il sistema mostra a video un messaggio di errore
 - 2.b Password scorretta
 1. Il player inserisce in modo errato la propria password nel form
 2. Il sistema mostra a video un messaggio di errore
- Post-Condizioni: Il player ha completato l'autenticazione al sistema
- Attore primario: Player registrato
- Attore secondario: GoogleAuth Provider
- Casi d'uso estesi: Sign in

Registrazione

- Nome: Registrazione
- Descrizione: Permette ad un player di registrarsi all'applicazione
- Pre-Condizioni: Il player non deve essere registrato e deve trovarsi nella pagina di registrazione
- Flusso principale:
 1. Il player compila il form con i propri dati
 2. Il player viene re-indirizzato alla pagina di Login
- Flussi alternativi:
 - 2.a Errore di compilazione del form
 1. Il player inserisce in modo errato uno o più informazioni non conformi
 2. Il sistema mostra a video un messaggio di errore
 - 2.a Email già esistente
 1. La mail inserita dal player è già presente nel database
 2. Il sistema mostra a video un messaggio di errore
- Post-Condizioni: Il player ha completato la registrazione al sistema e riceve la mail di conferma: l'account è stato registrato
- Attore primario: Player
- Casi d'uso inclusi: SendMailRegister

Logout

- Nome: Logout

- Descrizione: Permette ad un player registrato di effettuare il logout
- Pre-Condizioni: Il player registrato deve aver effettuato precedentemente il Sign in
- Flusso principale:
 1. Il player registrato clicca sul pulsante "Logout"
 2. Il player registrato viene re-indirizzato alla pagina di Login
- Post-Condizioni: I token associati alla sessione vengono eliminati
- Attore primario: Player registrato

4.2.2 Casi d'uso relativi al task T1

Con riferimento al diagramma dei casi d'uso del task T1, specifico per la figura dell'amministratore, è stato necessario aggiungere il caso d'uso **Scarica Classe**, espandendo le azioni eseguibili dall'admin registrato, e con un token JWT valido, una volta entrato nella sua pagina personale.

Il nuovo caso d'uso gli permette ora di scaricare correttamente il file contenente il codice associato ad una classe, inserita da se stesso o anche colleghi.

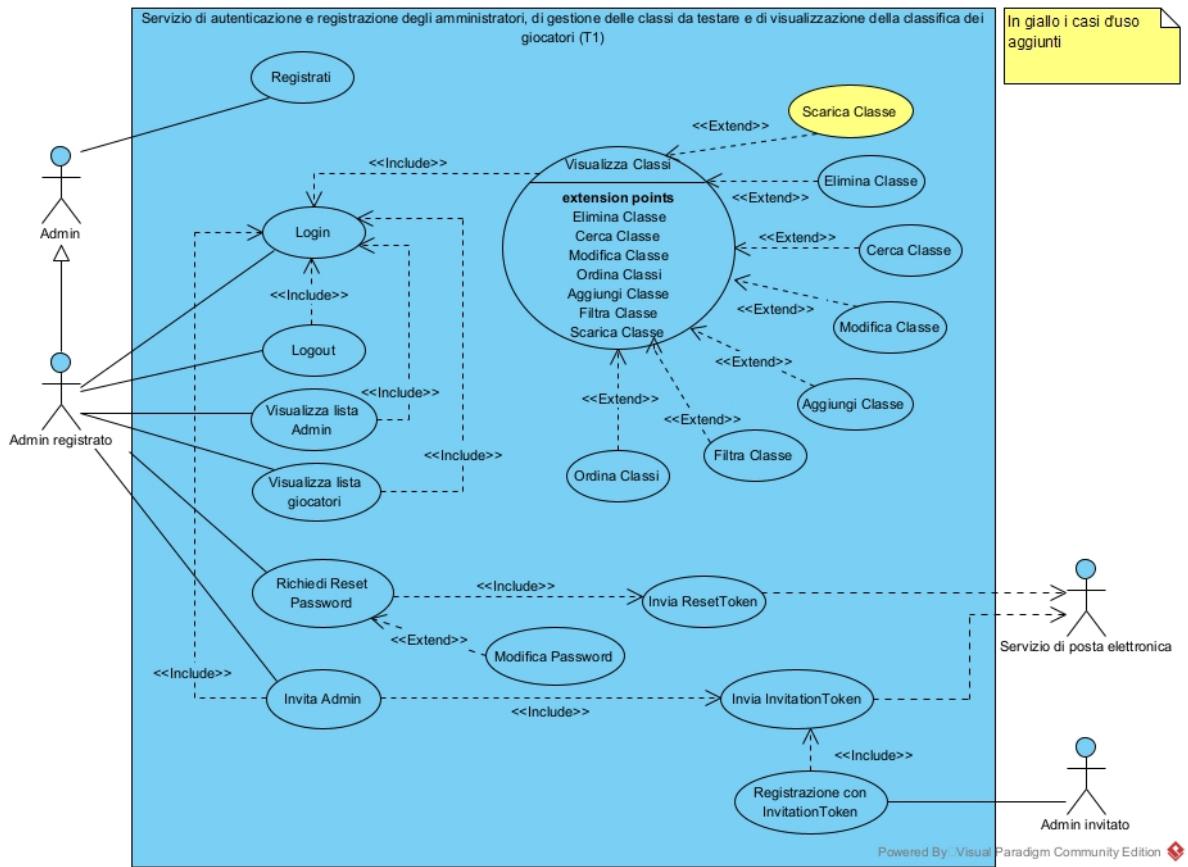


Figure 4.3: Use Case Diagram - Task T1

Scarica classe

- Nome: Scarica classe
- Descrizione: Permette all'admin registrato di scaricare il codice di una classe selezionata
- Pre-Condizioni: L'admin registrato deve aver precedentemente effettuato l'accesso (token JWT valido) e deve trovarsi nella sua area riservata
- Flusso principale:
 1. L'admin registrato clicca sul tasto "Classes"

2. L'admin seleziona la classe che vuole scaricare e clicca sul tasto "Download"

- Flussi alternativi:

1.a Sessione non valida

1. Il token JWT associato all'admin risulta invalido e l'utente viene re-indirizzato alla pagina di login

2.a Classe non presente sul DB

1. Viene mostrato un errore a video "Impossibile accedere al file" a causa di un errore nel path della classe nel DB

- Post-Condizioni: L'admin registrato ha scaricato il codice di una delle classi
- Attore primario: Admin registrato
- Casi d'uso estesi: Visualizza Classi

4.3 Component diagram

Il diagramma dei componenti è uno strumento utilizzato per visualizzare l'architettura di un sistema ad alto livello, identificando i principali componenti, evidenziando i ruoli e le dipendenze tra essi.

Inoltre, grazie a questo Component Diagram è possibile anche apprezzare la separazione delle funzionalità di autenticazione per il player e per l'admin.

Nel diagramma dei componenti fornитоci dalla documentazione precedente, sono presenti i componenti Student Front End e Student Repository, che sono stati modificati ed ora interconnessi con le nuove «*External Entity*» Google e Facebook, che rappresentano le interfacce esterne che offrono i rispettivi *Authentication Service* e *Registration Service* tramite una notazione a lollipop. Nel dettaglio, questi incapsulano la logica di autenti-

cazione, tra cui l’interazione con le API per l’autenticazione degli utenti e l’inserimento, o aggiornamento, dei dati degli utenti nello Student Repository.

Per quanto riguarda l’accesso degli studenti, si può notare che è stato garantito un controllo di sicurezza aggiuntivo grazie all’API Gateway. Tuttavia, quest’ultimo non offre lo stesso per le API dell’admin.

Oltre ai due componenti relativi allo studente, è stato necessario modificare UI Gateway per il corretto Download della classe scelta dall’admin registrato.

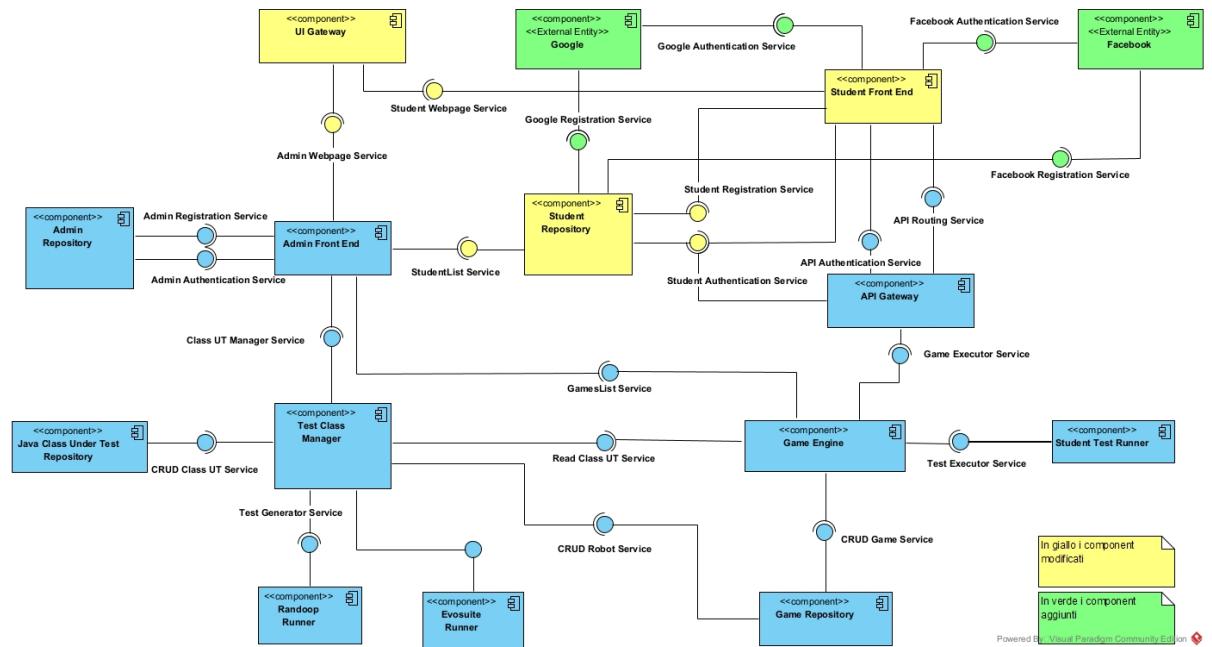


Figure 4.4: Component Diagram

4.4 Composite Diagram

In riferimento al Component Diagram, i componenti vengono inseriti all’interno di un package per specificare quale task sviluppano, al fine di realizzare il Composite Diagram del sistema. Si possono apprezzare in colore diverso i **task che hanno subito modifiche**, ed in particolare si nota che le operazioni svolte sul componente "*Student Front End*"

interessano sia il task T23, per quanto riguarda la registrazione dell’utente, ma anche il task T5 per permettere un corretto Logout.

Le modifiche riguardanti T6, invece, sono essenzialmente di refactoring.

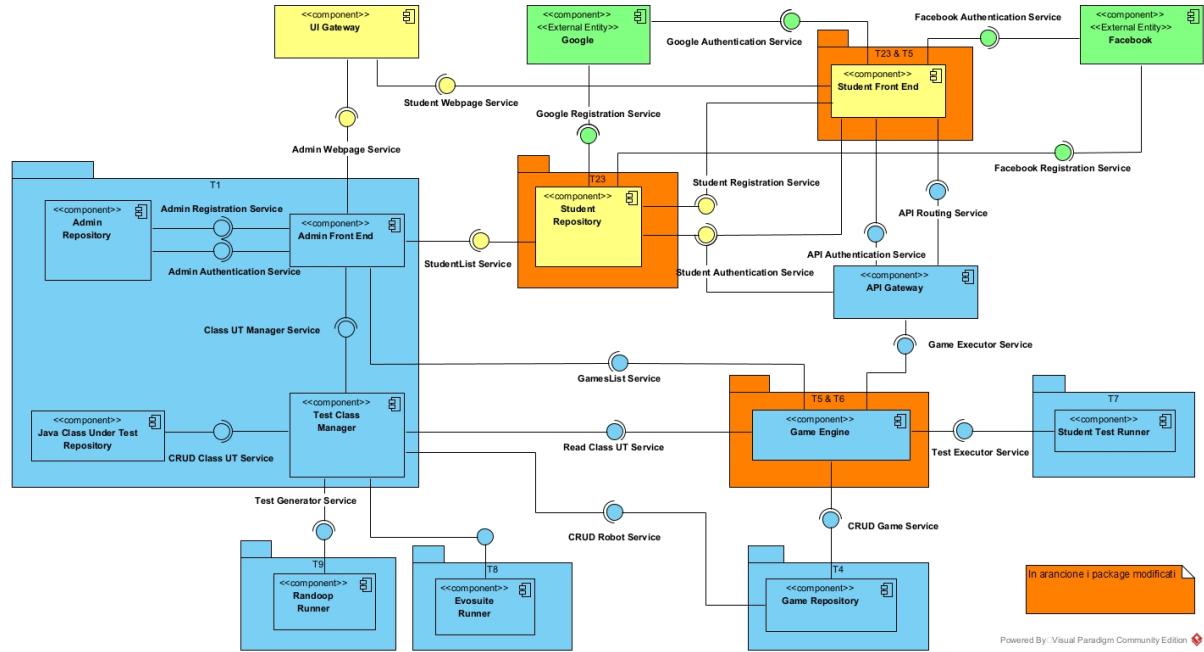


Figure 4.5: Composite Diagram

4.5 Activity Diagram

Un Activity Diagram è uno strumento di modellazione UML utilizzato per descrivere il flusso delle attività all’interno di un sistema software. Questo tipo di diagramma fornisce una descrizione chiara delle attività sequenziali e parallele coinvolte nell’esecuzione di una funzionalità del sistema. I [casi d’uso](#), che possono essere descritti come un insieme di scenari alternativi intrinsecamente sequenziali, sono rappresentabili graficamente in un tutt’uno.

Tutte le classi responsabili per le attività sono rappresentate mediante la propria swimlane.

4.5.1 Activity Diagram relativi al task T1

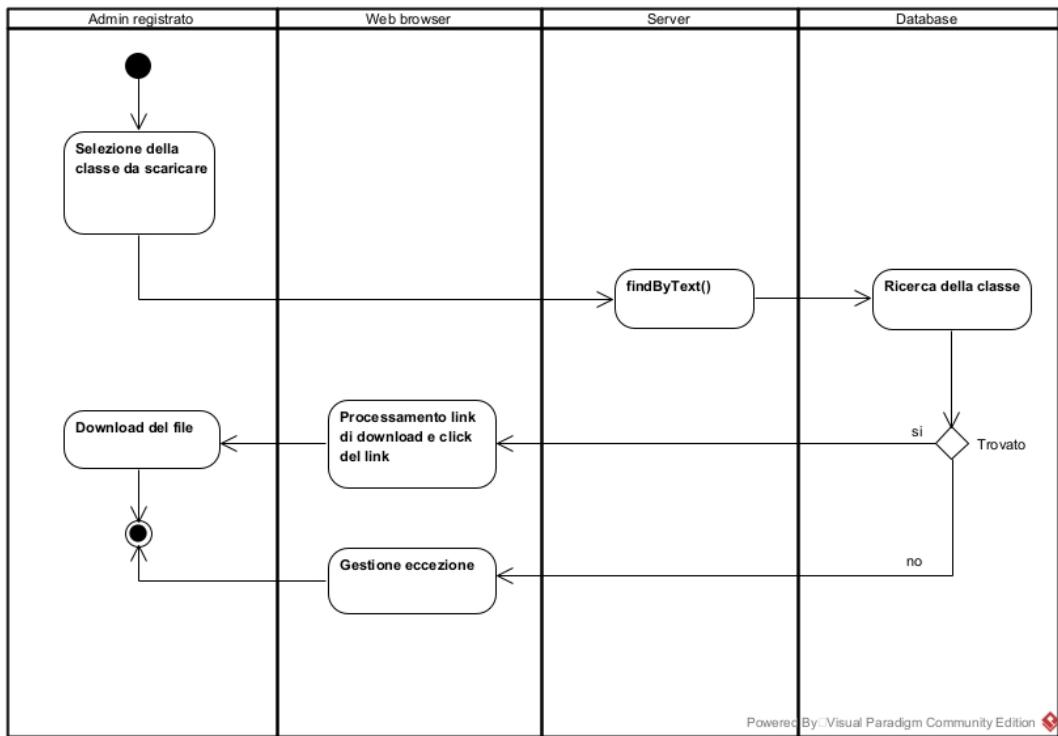


Figure 4.6: Activity Diagram - Download Classe

4.5.2 Activity Diagram relativi al task T23

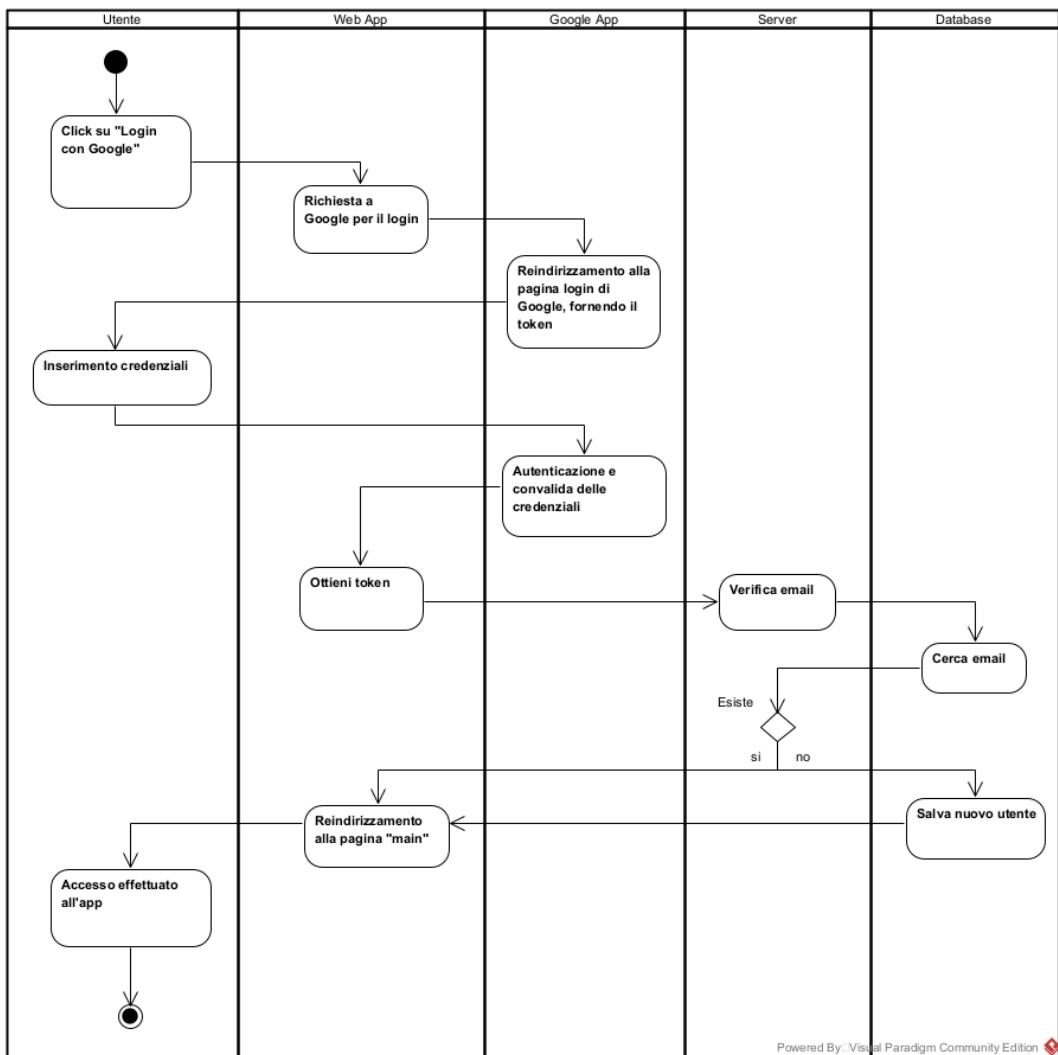


Figure 4.7: Activity Diagram - Sign in con Google & Sign up con Google

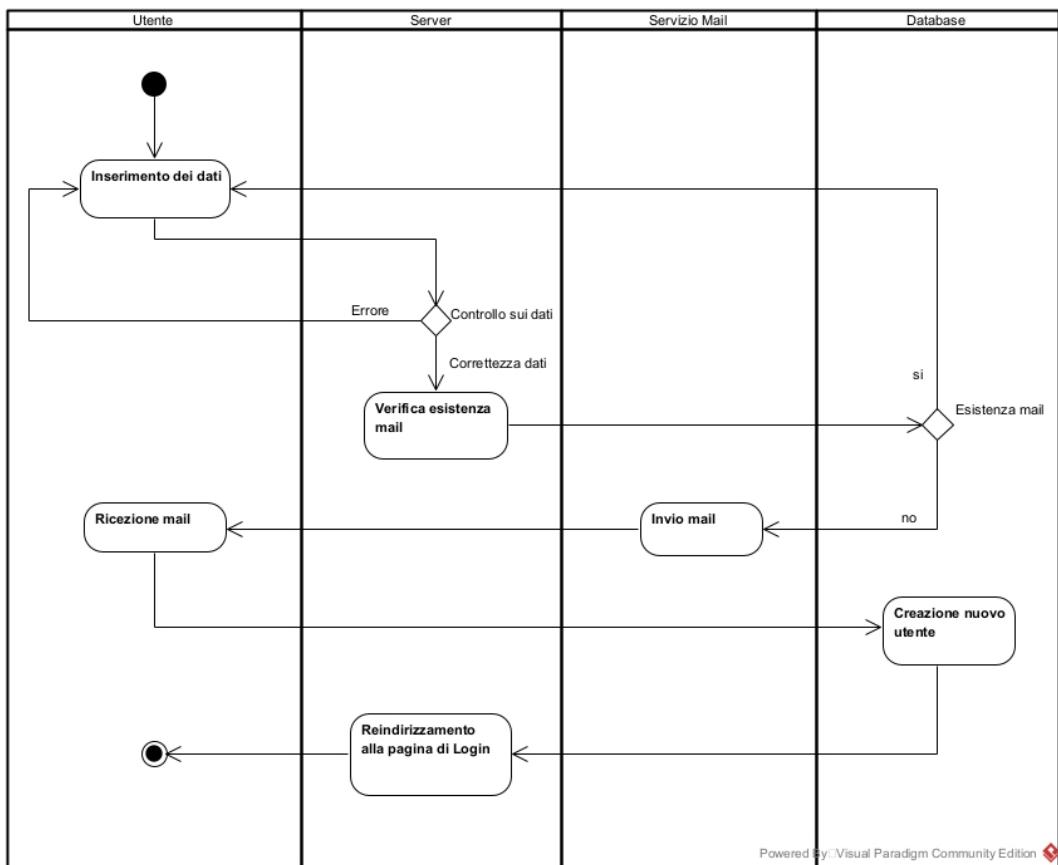


Figure 4.8: Activity Diagram - Registroazione

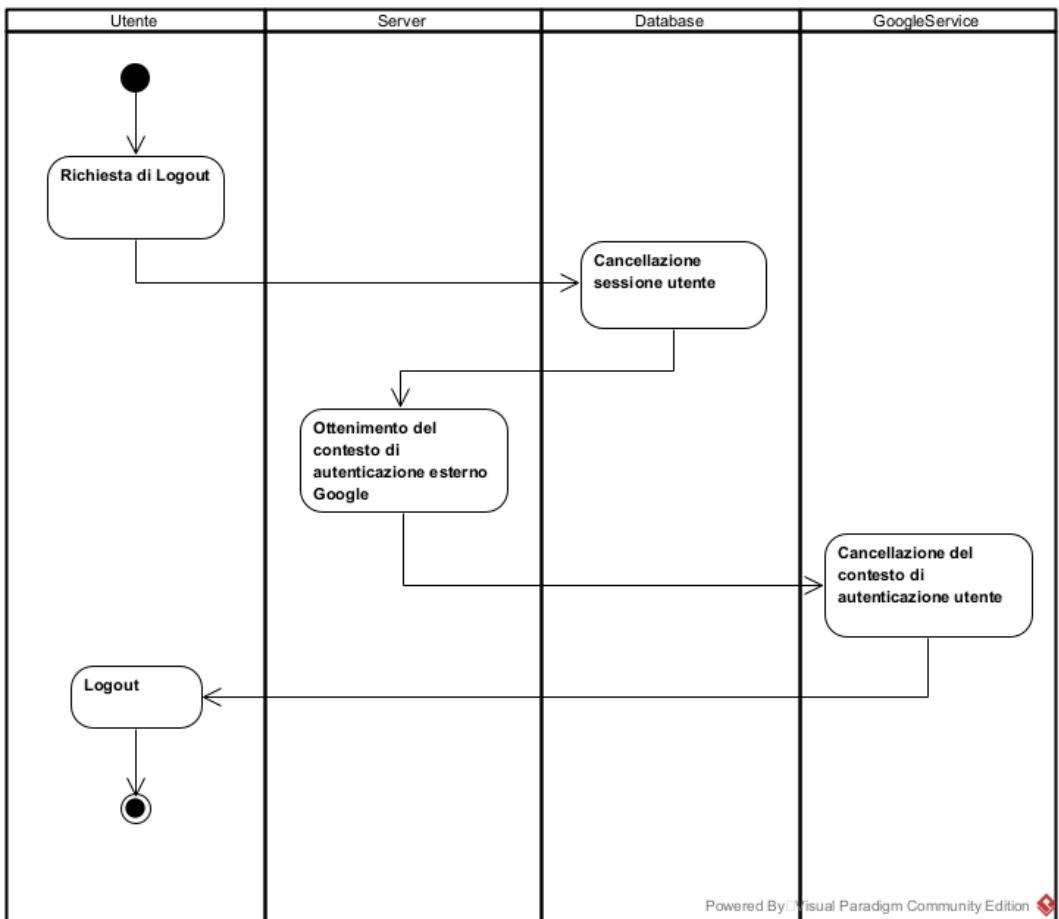


Figure 4.9: Activity Diagram - Logout

4.6 Sequence Diagram

Un diagramma di sequenza è un tipo di diagramma di interazione che mostra come gli oggetti operano tra loro e in che ordine all'interno di un sistema, definendo una timeline degli eventi. Questo tipo di diagramma è particolarmente utile per visualizzare il trasferimento del flusso di controllo tramite messaggi o chiamate tra oggetti o componenti all'interno di un sistema software.

4.6.1 Sequence Diagram relativi al task T1

È mostrato in figura il sequence diagram del caso d'uso aggiunto: [Scarica Classe](#).

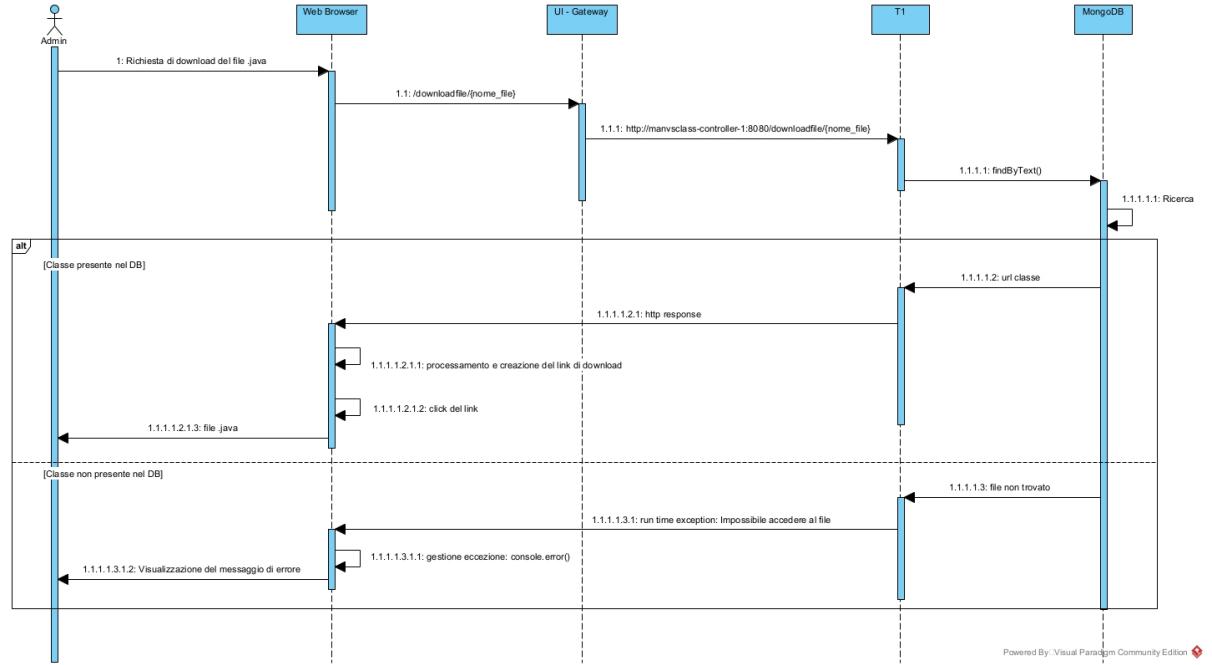


Figure 4.10: Sequence Diagram - Download Classe

4.6.2 Sequence Diagram relativi ai task T23

Possiamo di seguito osservare i sequence diagram inerenti alla Registrazione tramite compilazione del form, ma anche al Sign in/Sign up ed al Logout con Google osservati nei [casi d'uso T23](#), e come i vari componenti, della Web-App e quelli di Google, comunicano.

CHAPTER 4. FASE DI ANALISI

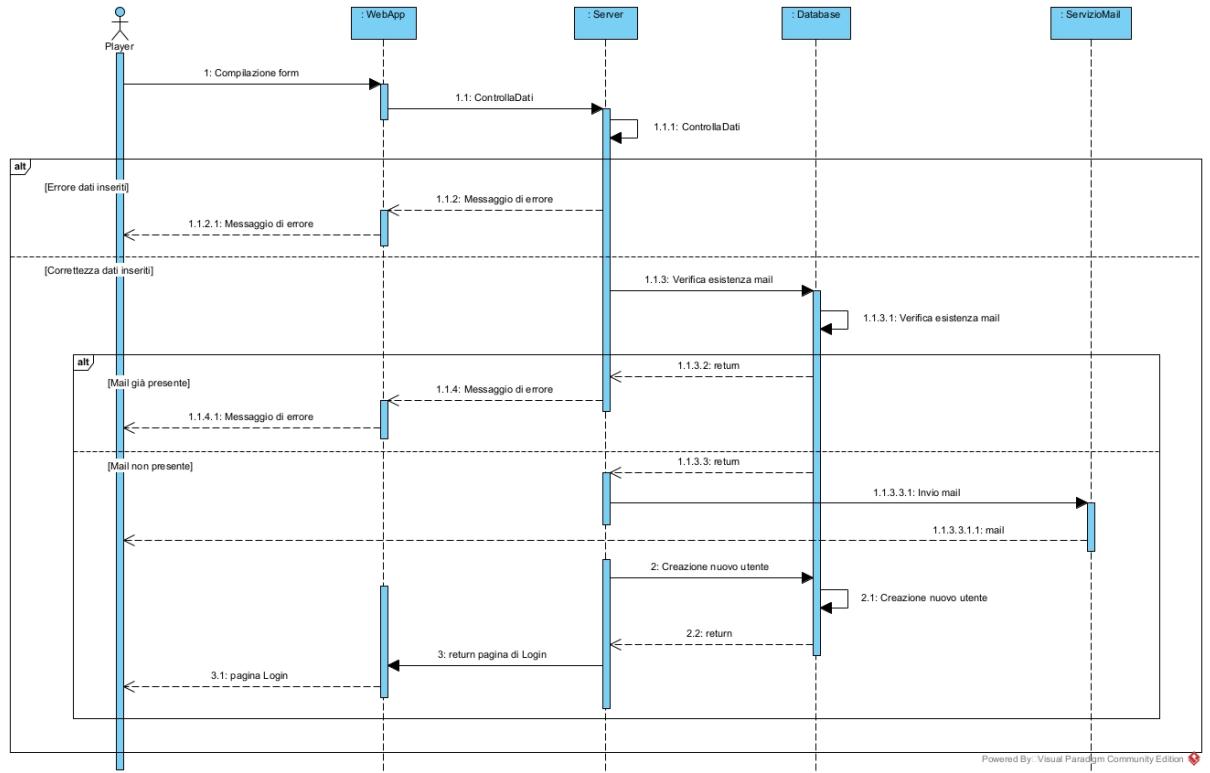


Figure 4.11: Sequence Diagram - Registrazione

CHAPTER 4. FASE DI ANALISI

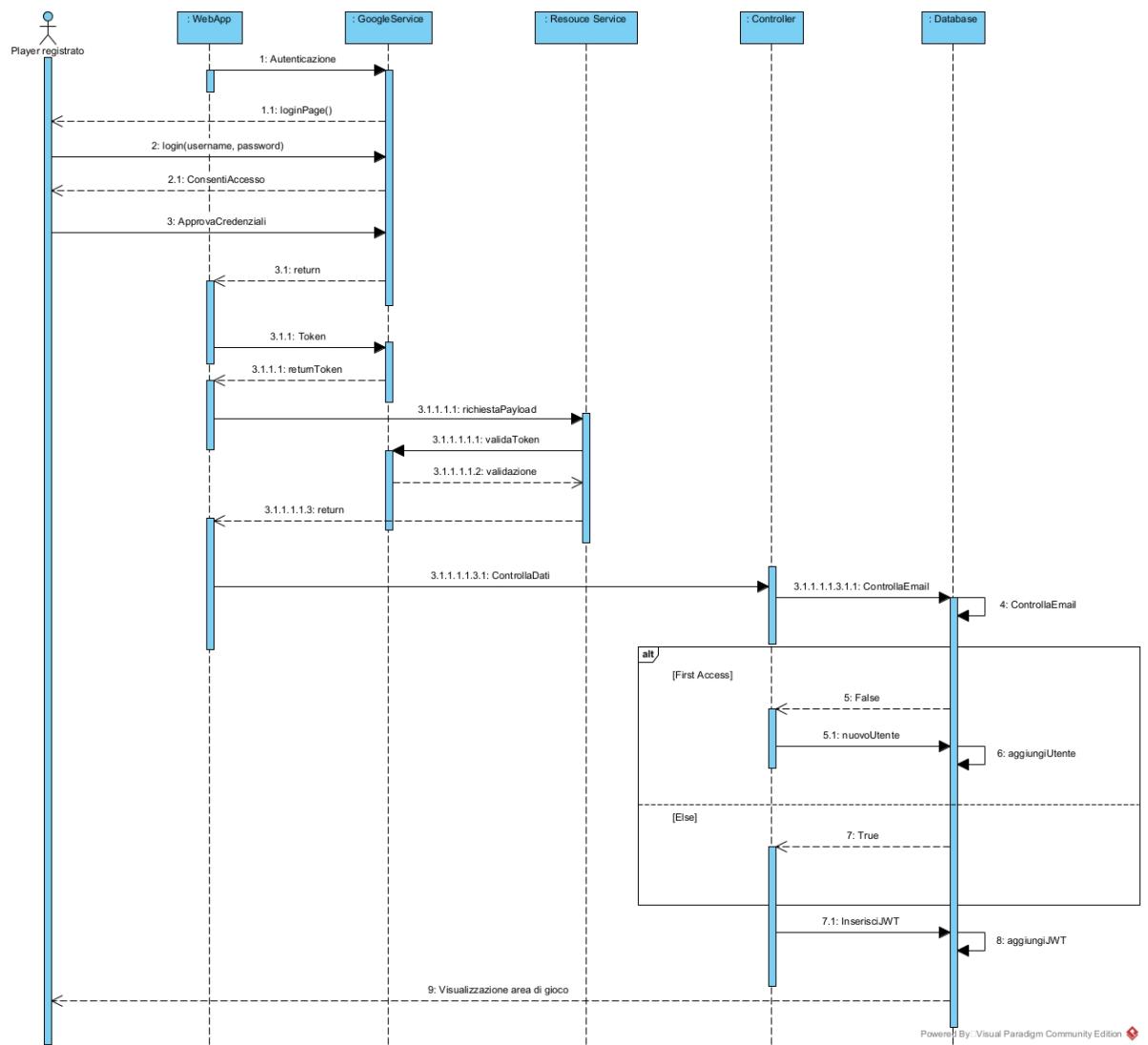


Figure 4.12: Sequence Diagram - Sign up Google & Sing in Google

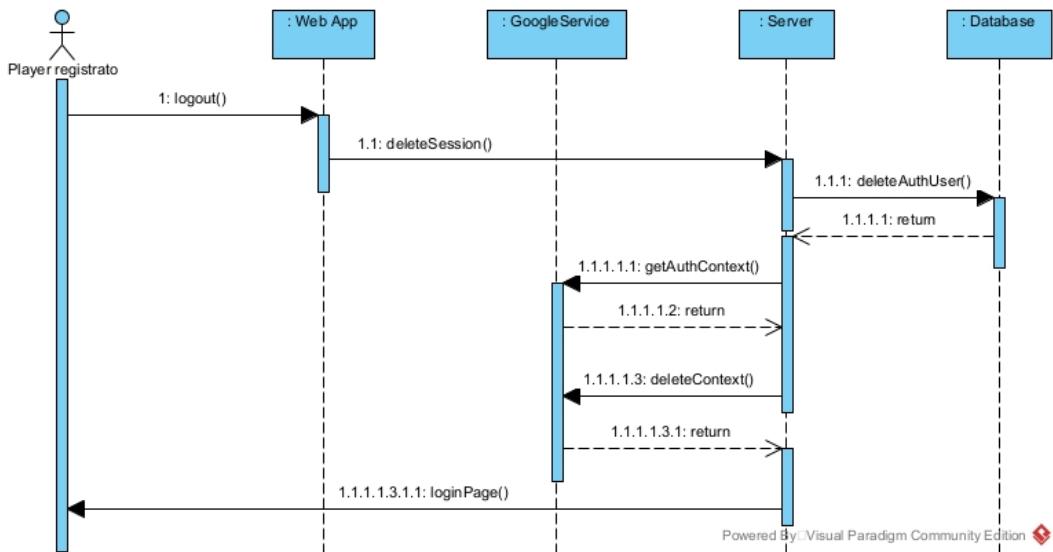


Figure 4.13: Sequence Diagram - Logout

4.7 Deployment Diagram

Tramite un UML Deployment Diagram vogliamo rappresentare l'assegnazione degli artefatti software, quindi in questo caso i container associati ai diversi task, ai corrispettivi nodi, che a loro volta possono assumere lo steriotipo «device», e quindi indicare il client o il server, o anche steriotipi non standard come «OS» quando si tratta di ambienti di esecuzione, ossia nodi software. Nello specifico, è riportato il diagramma della versione precedente del sistema, mettendo però in evidenza i **container che sono stati modificati**.

CHAPTER 4. FASE DI ANALISI

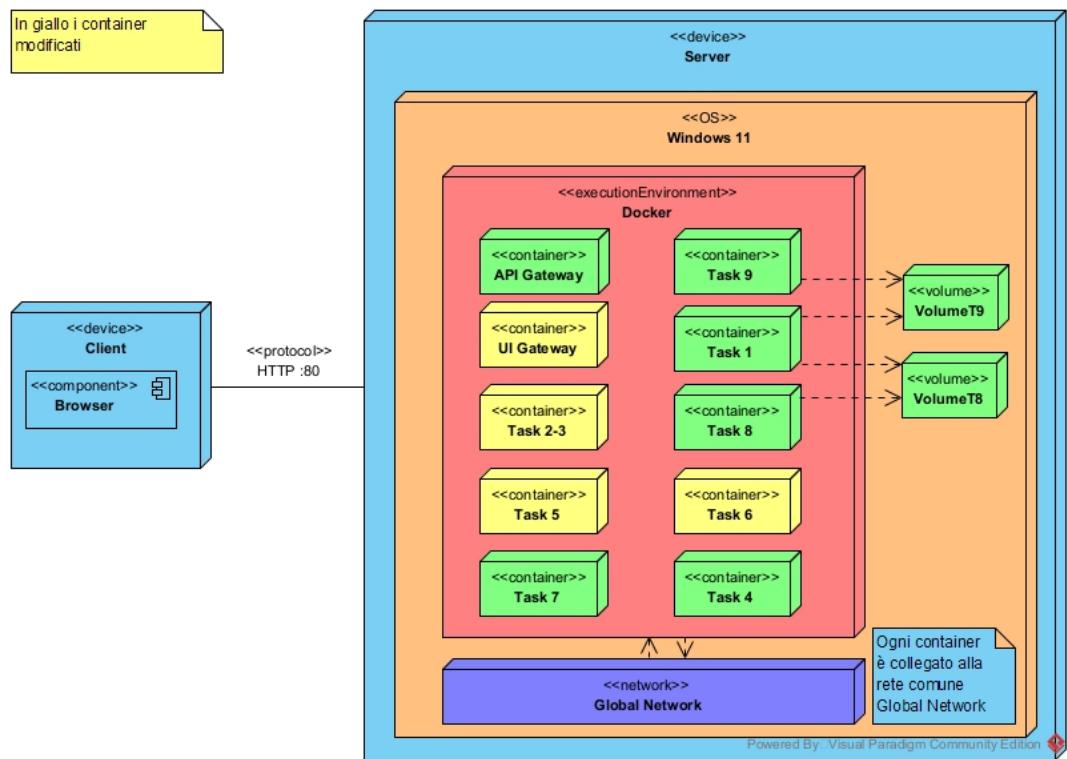


Figure 4.14: Deployment Diagram

Chapter 5

Implementazione

5.1 Login con Google

L'integrazione di un sistema di social login tramite Google rappresenta un miglioramento significativo per le applicazioni web moderne, consentendo agli utenti di autenticarsi utilizzando le credenziali del loro account Google. In seguito è descritto in dettaglio il processo di implementazione del social login con Google utilizzando il framework Spring Boot e le sue librerie.

5.1.1 Configurazione dell'API Google

Prima di procedere con l'implementazione è necessario ottenere le credenziali sviluppatore OAuth2 (Client ID and Client Secret) su Google Cloud Console per permettere all'applicazione di comunicare con l'API Google.

CHAPTER 5. IMPLEMENTAZIONE

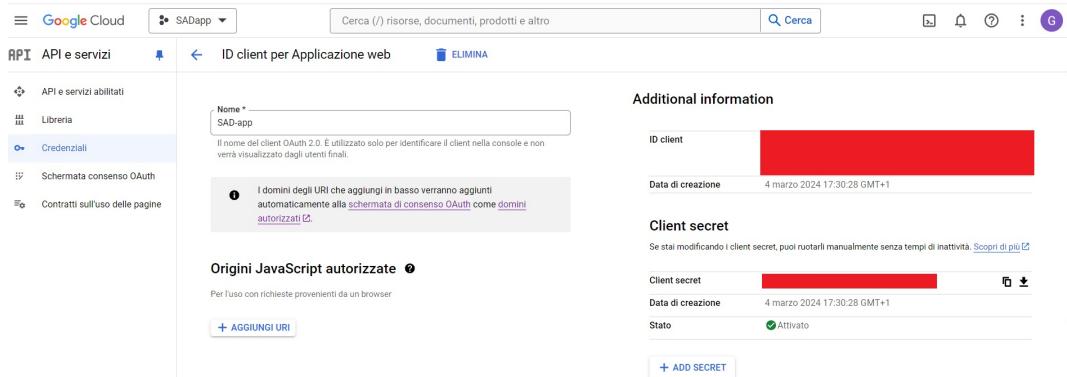


Figure 5.1: Pagina Sviluppatore Google API

Inoltre è necessario aggiungere gli URI specifici di reindirizzamento necessari per gestire il flusso di autenticazione, nel nostro caso `http://localhost/login/oauth2/code/google`.

5.1.2 Configurazione del Progetto Spring Boot

Una volta creato l'account su Google Cloud Console, bisogna configurare il progetto Spring Boot:

- Aggiungere le dipendenze necessarie per gestire l'autenticazione OAuth2 e Spring Security nel `pom.xml`.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-oauth2-client</artifactId>
</dependency>
```

- Configurare le proprietà OAuth2 nel file `application.properties` includendo i valori delle credenziali OAuth2 ottenuti dalla console di Google Cloud.

5.1.3 Configurazione di Spring Security per l'autenticazione OAuth2

Spring Security gestisce l'autenticazione OAuth2 all'interno dell'applicazione Spring Boot. Sono state scelte le configurazioni includendo le regole di autorizzazione e autenticazione necessarie per proteggere le risorse dell'applicazione. Per fare ciò viene inserita una classe `SecurityConfig`, che estende `WebSecurityConfigurerAdapter`. Qui vengono specificate che alcune rotte come `"/login"` e `"/oauth/**"` sono accessibili a tutti, senza richiedere autenticazione. Questo significa che chiunque può visitare queste pagine, indipendentemente dal fatto che sia autenticato o meno. È importante avere queste rotte pubbliche per consentire agli utenti di accedere alla pagina di login e avviare il processo di autenticazione con Google, per tutte le altre richieste, richiediamo che l'utente sia autenticato.

Per il login OAuth2, la configurazione specifica la pagina di login personalizzata con `oauth2Login().LoginPage("/login")`. Questa pagina sarà utilizzata per avviare il processo di autenticazione con Google. Inoltre, viene configurato l'endpoint delle informazioni utente con `userInfoEndpoint().userService(oAuthUserService)`, che utilizza un **servizio utente personalizzato** per gestire le informazioni degli utenti ottenute da Google.

Un'altra configurazione importante è la gestione dell'autenticazione che avviene correttamente. Utilizzando `successHandler(googleAuthenticationSuccessHandler)`, viene specificato un gestore personalizzato che definisce cosa accade dopo che l'utente è stato autenticato con successo tramite Google ed includere tutte le logiche personalizzate.

5.1.4 Implementazione OAuth User Google e OAuth Google Service

Una volta che la web-application si può interfacciare con l'API di Google sono state definite due classi per riuscire a raccogliere e contenere i dati provenienti dagli account Google.

La classe `OAuthUserGoogle` è progettata per gestire le informazioni di un utente autenticato tramite Google. In pratica, questa classe agisce come un **contenitore per le informazioni** dell'utente fornite da Google, come nome utente ed e-mail, consentendo di accedere facilmente a tali informazioni.

Il caricamento dei dati dell'utente viene effettuato attraverso il servizio `OAuthUserService` che contatta Google e ottiene i dettagli dell'utente e li resittuisce sottoforma di `OAuthUserGoogle`.

5.1.5 Implementazione del Success Handler

Una volta che l'utente accede con successo attraverso Google, l'evento viene gestito attraverso il componente personalizzato `GoogleSuccessHandler`.

Si ottengono le informazioni dell'utente autenticato, come l'indirizzo email, utilizzando l'oggetto `Authentication` fornito da Spring Security. Se l'utente non esiste già nel sistema, viene creato utilizzando i dettagli forniti da Google e salvato nel DB. Successivamente, viene generato un token JWT per l'utente e salvato all'interno di un cookie HTTP per effettuare l'accesso all'app. Infine, l'utente viene reindirizzato alla pagina principale dell'applicazione.

5.2 Cancellazione Classe

L'obiettivo principale di questo task è garantire che quando un amministratore elimina una classe, questa venga completamente rimossa dal file system, evitando eventuali incongruenze o confusione. Durante l'analisi dell'attività di cancellazione, abbiamo constatato che il **bug non sussisteva**. La cancellazione avviene correttamente nella classe

HomeController del container T1, nell'endpoint /deletefile/{filename}.

Per verificare che il bug non fosse presente e che la cancellazione avvenisse correttamente, sono stati eseguiti i seguenti passaggi:

- **Controllo nel File System su Docker:** Verificare che la cancellazione avvenga effettivamente nel file system. Abbiamo controllato i path effettivi (" /VolumeT9/app/FolderTree/" e " /VolumeT8/FolderTreeEvo/") delle directory per assicurare che venissero eliminati correttamente.
- **Controllo dell'implementazione:** Il processo di cancellazione viene gestito dal metodo /deleteFile/fileName e /delete/name della classe HomeController. /delete/name si occupa di verificare la validità del token JWT dell'utente amministratore, costruire una query per trovare la classe specificata nel database MongoDB utilizzando il nome della classe e la rimuove. /deleteFile/fileName invece individua la cartella corrispondente alla classe da eliminare, e la rimuove dal file system. Inoltre, si assicura che le directory associate, directoryRandoop e directoryEvo, vengano anch'esse eliminate.
- **Verifica delle Richieste con Postman:** Si è utilizzato Postman per inviare richieste all'endpoint /deleteFile/{fileName} e verificare le risposte ricevute. Si è visto poi che gli status code HTTP restituiti fossero corretti (200 OK per successo, 404 Not Found per directory non esistente, 401 Unauthorized per token non valido, ecc.).
- **Controllo su MongoDB:** Si è esplorato anche MongoDB per essere sicuri che i riferimenti alle classi fossero stati rimossi.

L'indagine ha garantito che la cancellazione delle classi avvenisse correttamente e completamente, eliminando ogni possibile incongruenza tra il file system e i dati del sistema. I controlli effettuati hanno confermato che il bug non sussisteva, assicurando la coerenza

e l'affidabilità della funzionalità di cancellazione.

/deleteFile/{fileName}

```
@PostMapping("/deleteFile/{fileName}")
@ResponseBody
public ResponseEntity<String> eliminaFile(@PathVariable String
    fileName, @CookieValue(name = "jwt", required = false) String
    jwt) {
    if (isJwtValid(jwt)) {

        System.out.println("Token valido, pu rimuovere il file
            selezionato (/deleteFile/{fileName})");
        String folderPath = "Files-Upload/" + fileName;
        File directoryRandoop = new File("/VolumeT9/app/FolderTree/"
            + fileName);
        File directoryEvo = new File("/VolumeT8/FolderTreeEvo/" +
            fileName);

        File folderToDelete = new File(folderPath);
        if (folderToDelete.exists() && folderToDelete.isDirectory()) {
            try {
                FileUploadUtil.deleteDirectory(folderToDelete);
                FileUploadUtil.deleteDirectory(directoryRandoop);
                FileUploadUtil.deleteDirectory(directoryEvo);
                return new ResponseEntity<>("Cartella eliminata con
                    successo (/deleteFile/{fileName})", HttpStatus.OK);

            } catch (IOException e) {
                return new ResponseEntity<>("Impossibile eliminare la
                    cartella.", HttpStatus.INTERNAL_SERVER_ERROR);
            }
        }
    }
}
```

```
        }

    } else {

        return new ResponseEntity<>("Cartella non trovata.",
            HttpStatus.NOT_FOUND);
    }

} else {

    return new ResponseEntity<>("Token JWT non valido.",
        HttpStatus.UNAUTHORIZED);
}

}
```

5.3 Download Classe

L'implementazione della funzionalità di download della classe da parte dell'admin registrato è stata possibile aggiornando il file di configurazione "default.conf" dell'UI Gateway contenente le root degli API endpoints dei corrispondenti task. In particolare è stato necessario modificare il *proxy_pass* del task T1, aggiungendo l'endpoint corrispondente al download della classe.

```
location ~^/(scalata|login_with_invitation|...|downloadFile|test) {
    include /etc/nginx/includes/proxy.conf;
    proxy_pass http://manvsclass-controller-1:8080;
}
```

5.4 Logout

L'esecuzione del logout viene aggiornata intervenendo sia nel Controller del task T23, relativamente alla richiesta POST "/logout", che nella funzione JS redirectToLogin() presente in "main.js" di T5, risolvendo l'[issue](#) segnalato su GitHub.

La nuova funzionalità di logout nel controller è effettuata con il metodo POST e a partire dal token jwt, cerca nel DB l'utente autenticato corrispondente, lo elimina e contestualmente cancella i relativi cookie, invalida la sessione http e elimina il contesto di sicurezza gestito da Spring Boot. Di conseguenza è stata adeguata `redirectToLogin()` in maniera opportuna.

redirectToLogin()

```
function redirectToLogin() {
    if (confirm("Sei sicuro di voler effettuare il logout?")) {
        const jwt = getCookie('jwt');

        fetch(`/logout?authToken=${encodeURIComponent(jwt)}`, {
            method: 'POST',
        })
            .then(response => {
                if (!response.ok) {
                    throw new Error('Richiesta logout non andata a buon fine');
                }
                else {
                    console.log("stai per essere reindirizzato alla pagina di
                                login");
                    alert("Logout avvenuto con successo, a breve verrai
                            re-indirizzato alla pagina di login");
                    window.location.href = "/login";
                }
            })
            .catch((error) => {
                console.error('Error:', error);
            });
    }
}
```

```
}
```

@PostMapping("/logout")

```
@PostMapping("/logout")
public ResponseEntity<String> logout(@RequestParam("authToken")
    String authToken, HttpServletResponse response,
    HttpServletRequest request) {
    AuthenticatedUser authenticatedUser =
        authenticatedUserRepository.findByAuthToken(authToken);
    System.out.println("POST logout called, token removed");

    if (authenticatedUser == null) {
        return
            ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("User
                not authenticated");
    }

    Cookie jwtTokenCookie = new Cookie("jwt", null);
    jwtTokenCookie.setMaxAge(0);
    response.addCookie(jwtTokenCookie);

    // Delete JSESSIONID cookie
    Cookie jsessionIdCookie = new Cookie("JSESSIONID", null);
    jsessionIdCookie.setMaxAge(0);
    response.addCookie(jsessionidCookie);

    authenticatedUserRepository.delete(authenticatedUser);
    //Modifica 18/05/2024: Cancellazione dei cookie e del contesto
        di autenticazione di spring
```

```
SecurityContextHolder.clearContext();

HttpSession session= request.getSession(false);

if(session != null) {

    session.invalidate();

}

return ResponseEntity.ok("Logout successful");
}
```

Osservando le similitudini con quanto indicato in un [altro issue](#) relativo al Logout all'interno dell'allenamento, applicando le stesse modifiche correttive all'interno del task T5 anche questo caso di Logout dovrebbe tornare a eseguire correttamente.

5.5 Registrazione

Le implementazioni chiave derivanti dal task [1.5 Task: Bug fix Registrazione da parte del player](#) sono focalizzate sul miglioramento del processo di registrazione degli utenti all'interno dell'applicazione. Durante l'analisi iniziale, è stato visto che nonostante il processo di registrazione funzioni correttamente, erano presenti degli errori in "register.js", richiamato all'interno della pagina HTML. Allo stato attuale l'utente una volta compilato il form, quest'ultimo inoltra direttamente una richiesta POST al Controller di T23 "/register", che effettua tutti i controlli di correttezza dei campi inseriti. Nel caso in cui ci fosse un errore, come ad esempio l'email già presente nel DB, all'utente viene mostrato il body della response con il relativo status code. In questo procedimento, essendoci dei problemi in "register.js", il codice JavaScript non sarà proprio utilizzato.

Sono state introdotte delle modifiche mirate a correggere il file:

- Il problema iniziale con *register.js* era che il codice JavaScript si caricava immediatamente, prima che l'intero documento HTML fosse completamente caricato.

Questo causava problemi nel momento in cui veniva eseguita l'istruzione `const form = document.querySelector("form")`, poiché il DOM non era ancora pronto e quindi il selettore non trovava l'elemento `<form>` correttamente. Utilizzando `document.addEventListener('DOMContentLoaded', ...)` come wrapper principale, tutte le operazioni avvengono quando il DOM è completamente accessibile e pronto per l'interazione.

- Si è identificato un errore nel formato di invio dei dati del form tramite JSON, che risultava in una richiesta *Bad Request*. Per risolvere questo problema, i dati sono stati invece inviati tramite un oggetto `FormData`, che è più adatto per l'invio di dati di form.
- Per consentire un corretto reindirizzamento dell'utente dopo una registrazione riuscita, è stata modificata la risposta del `Controller.java` aggiungendo l'header `Location` con il percorso `"login_success"` e restituendo un codice di stato **MOVED PERMANENTLY (301)**. Questo permette al client di essere reindirizzato automaticamente alla pagina di login avvenuto correttamente.
- È stato implementato un sistema di gestione degli errori che utilizza popup di alert per notificare chiaramente all'utente eventuali problemi durante la registrazione.

register.js

```
document.addEventListener('DOMContentLoaded', (event) => {
  const form = document.querySelector("form");
  console.log(form);
  form.addEventListener("submit", async (event) => {
    event.preventDefault();

    //Costruzione del form per l'invio dei dati tramite la POST
    request.
```

```
const nome = document.getElementById("name").value.trim();
const cognome = document.getElementById("surname").value.trim();
const email = document.getElementById("email").value.trim();
const password = document.getElementById("password").value.trim();
const confermaPassword =
    document.getElementById("check_password").value.trim();
const corsoDiStudi = document.getElementById("studies").value;

const formData = new FormData();
formData.append("name", nome);
formData.append("surname", cognome);
formData.append("email", email);
formData.append("password", password);
formData.append("check_password", confermaPassword);
formData.append("studies", corsoDiStudi);

try {
    if (nome === '') {
        alert("Compila il campo Nome!");
        return;
    }
}

[...]

if (password !== confermaPassword) {
    alert("Le password non corrispondono!");
    return;
}
```

```
// fetch della richiesta

const response = await fetch('/register', {
  method: 'POST',
  body: formData,
}) ;

// Controllo dello status code di reindirizzamento (301)
if (!response.redirected) {
  // Se non e' 301 lancia un errore
  const errorBody = await response.text();
  throw new Error(`Errore: ${errorBody}`);
}

// Se la response e' di reindirizzo cerca l'url nell'header
// della response
console.log('Received data:', response.body);
window.location.href = response.url;
} catch (error) {
  // Pop-up del messaggio di errore da parte della pagina,
  // gestisce anche gli errori del controller
  console.error('Error:', error.message);
  alert(error.message);
}
});

});
```

Controller.java (/register)

```
try {
  emailService.sendMailRegister(email, ID);
```

```
//Modifica: Cambiato il codice per consentire il  
reindirizzamento  
  
HttpHeaders headers = new HttpHeaders();  
headers.add("Location", "/login_success");  
return new  
    ResponseEntity<String>(headers, HttpStatus.MOVED_PERMANENTLY);  
//FINE MODIFICA  
  
} catch (MessagingException e) {  
    return  
        ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Failed  
to confirm your registration");  
}
```

5.6 Refactoring

L'implementazione del refactoring e redesign delle interfacce si è concentrata su due principali obiettivi: garantire una separazione chiara tra HTML, CSS e JavaScript per mantenere il codice di T23 pulito e facilmente manutenibile, e migliorare l'aspetto grafico delle pagine per offrire un'esperienza utente più moderna e user-friendly. Per quanto riguarda T6, è stata rispettata la separazione gerarchica come detto precedentemente. Tuttavia, analizzando la pagina `index.html`, è emerso che si tratta solamente di una pagina di testing dell'editor di codice, quindi non rilevante a livello di funzionalità. L'editor, infatti, viene caricato dal microservizio T5 che gestisce la GUI dell'applicazione dello studente e, pertanto, il file potrebbe essere rimosso. Nonostante ciò, il nostro team ha scelto di mantenere `index.html` poiché potrebbe essere utile per testare nuove funzionalità aggiunte nell'editor.

I principali cambiamenti implementati sono:

- **Separazione del Contenuto:**

- **HTML:** Tutti i file HTML sono stati spostati nella cartella `templates` per mantenere il contenuto separato dagli stili e dal comportamento.
- **CSS:** Gli stili CSS sono stati spostati in una directory separata all'interno della cartella `static/css`.
- **JavaScript:** Gli script JavaScript sono stati spostati in una directory separata all'interno della cartella `static/js`.

- **Design Modernizzato:**

- Le pagine di login e dashboard sono state ridisegnate per includere layout responsive, palette di colori moderni e una navigazione intuitiva.
- Abbiamo cercato di garantire una coerenza stilistica tra le varie pagine, tenendo conto che l'applicazione web deve essere presentata sottoforma di gioco educativo.

Un'altra scelta importante è stata di implementare header e footer come elementi ricorrenti in un singolo file di layout per migliorare l'efficienza e la coerenza visiva dell'applicazione. Includere l'header e il footer in un file separato ha consentito di ridurre la duplicazione del codice, evitando la necessità di replicare gli stessi elementi in ogni singola pagina HTML. In questo modo, qualsiasi modifica apportata all'header o al footer può essere aggiornata in un unico punto, assicurando coerenza in tutto il sito.

Utilizzando Thymeleaf per l'inclusione dinamica:

```
<!DOCTYPE html><html lang="en">
  xmlns:th="http://www.thymeleaf.org">
```

CHAPTER 5. IMPLEMENTAZIONE

```
<head>...</head>
<body>
    <div th:replace="layouts/header :: header"></div>
    <!-- Contenuto della pagina -->
    <div th:replace="layouts/footer :: footer"></div>
</body>
</html>
```

Chapter 6

Testing

Il testing è una fase cruciale nel ciclo di vita del software, in quanto permette di individuare e correggere errori, garantendo che il prodotto finale risponda ai requisiti specificati e funzioni correttamente in tutte le condizioni previste. Nel nostro caso il testing è stato orientato sia alla verifica delle nuove funzionalità aggiunte, come il login tramite Google, sia alla risoluzione di bug presenti nelle versioni precedenti dell'applicazione.

6.1 Login con Google

La correttezza del Login con Google è stata testata dal punto di vista di un giocatore che vuole accedere alla schermata di gioco iniziale.

È possibile apprezzare la sua prospettiva nel seguente [video dimostrativo](#).

6.2 Cancellazione classe

Per quanto riguarda il testing di questa sezione, vogliamo verificare che una classe venga cancellata correttamente in tutte le directory del file system come visto nell'implementazione.

- L'amministratore ha caricato correttamente la classe `Calcolatrice.java`, quindi

CHAPTER 6. TESTING

i file sorgente ed i test generati saranno conservati rispettivamente in "app/Files-Upload" all'interno di T1 ed in "FolderTree/Calcolatrice" nei volumi T8 e T9. In più è stata aggiunta una riga nella tabella ClassUT all'interno del database MongoDB.

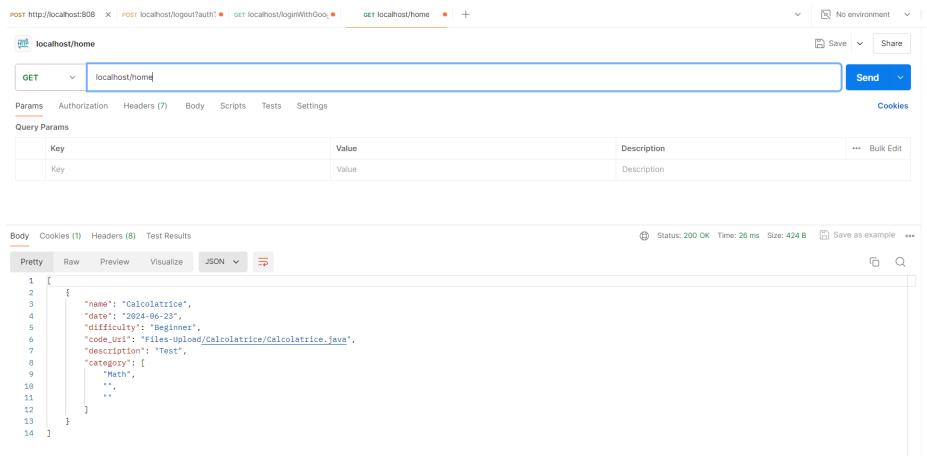


Figure 6.1: Lista di tutte le classi disponibili



Figure 6.2: Codice sorgente della classe in Files-Upload/Calcolatrice

CHAPTER 6. TESTING

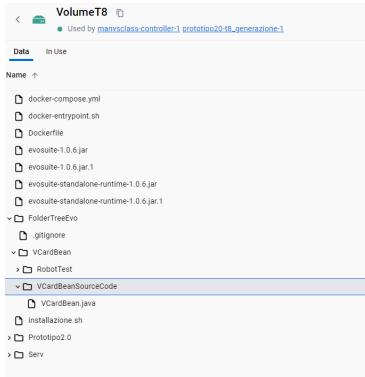


Figure 6.3: FolderTree T8

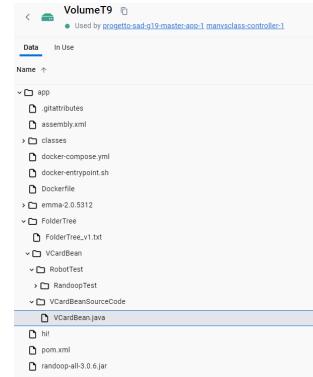


Figure 6.4: FolderTree T9

```
manvsclass> db.ClassUT.find().pretty()
[
  {
    _id: ObjectId("6677db789b92311f11d4f5f7"),
    name: 'Calcolatrice',
    date: '2024-06-23',
    difficulty: 'Beginner',
    code_Uri: 'Files-Upload/Calcolatrice/Calcolatrice.java',
    description: 'Test',
    category: [ 'Math', '', '' ],
    _class: 'com.groom.manvsclass.model.ClassUT'
  }
]
manvsclass> 
```

Figure 6.5: Contenuto della tabella ClassUT di MongoDB

- Viene fatta poi una richiesta di cancellazione attraverso Postman utilizzando il token jwt di accesso dell'amministratore autenticato: la classe scelta è stata eliminata effettuando una richiesta POST all'endpoint `localhost/delete/Calcolatrice`.

CHAPTER 6. TESTING

The screenshot shows a Postman interface with the following details:

- URL:** http://localhost/delete/Calcolatrice
- Method:** POST
- Headers:** Content-Type: application/json
- Body (JSON):**

```
1 {
2     "name": "Calcolatrice",
3     "date": "2024-06-23",
4     "difficulty": "Beginner",
5     "code_Url": "files-Upload/Calcolatrice/Calcolatrice.java",
6     "description": "",
7     "category": [
8         {
9             "name": "Math"
10        }
11    ]
12 }
```
- Status:** 200 OK
- Time:** 70 ms
- Size:** 414 B

Figure 6.6: POST request per la cancellazione della classe

- Andando a cercare nelle directory del filesystem, in MongoDB e nella lista delle classi si può notare che la classe è cancellata correttamente.

```
-----  
Token valido, puoi rimuovere la classe selezionata (/delete/{name})  
Token valido, puoi rimuovere il file selezionato (/deleteFile/{fileName})  
Rimozione avvenuta con successo (/delete/{name})  
(/home) visualizzazione delle classi di gioco  
(/home) visualizzazione delle classi di gioco
```

Figure 6.7: Log di cancellazione della classe nella console di Docker del container T1



Figure 6.8: Directory di T1 dopo la cancellazione

CHAPTER 6. TESTING

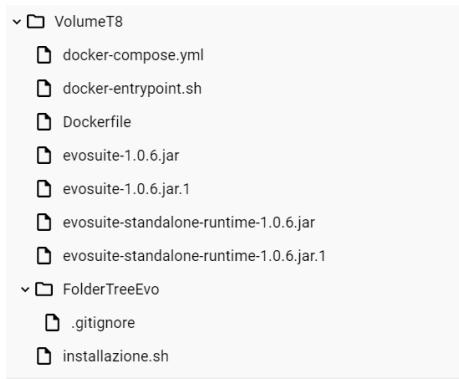


Figure 6.9: FolderTree T8 dopo la cancellazione

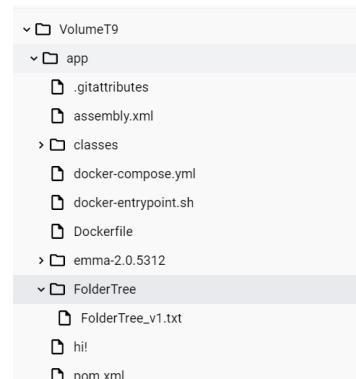
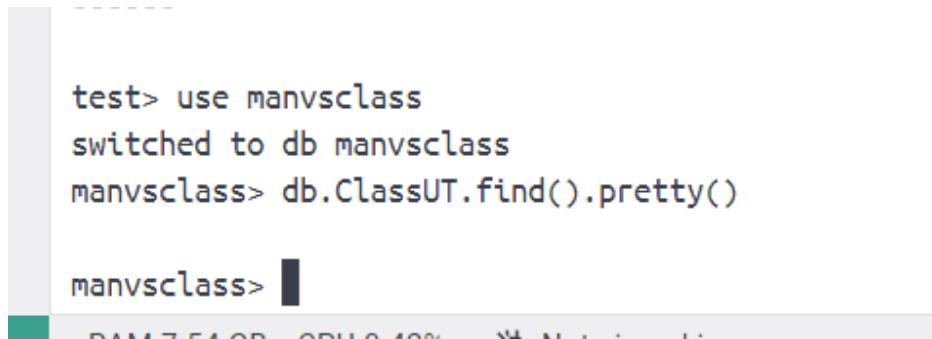


Figure 6.10: FolderTree T9 dopo la cancellazione

The screenshot shows a Postman request for 'http://localhost/home'. The method is 'GET'. Under 'Headers', there are four checked fields: User-Agent (Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.110 Safari/537.36'), sec-ch-ua ("Not(A)Brand";v="8", "Chromium";v="116", "Google Chrome";v="116"), sec-ch-ua-mobile (?0), and sec-ch-ua-platform ("Windows"). The response status is 200 OK, Time: 17 ms. The response body is a JSON array: [{}].

Figure 6.11: Lista delle classi dopo la cancellazione



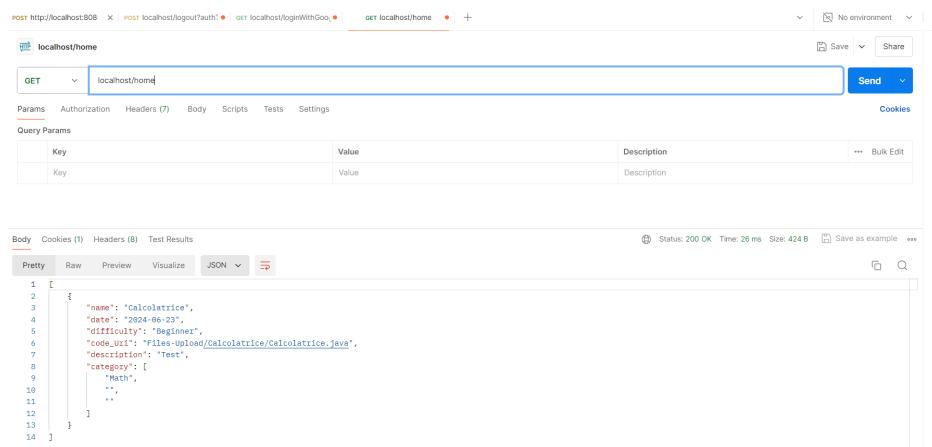
```
test> use manvsclass
switched to db manvsclass
manvsclass> db.ClassUT.find().pretty()

manvsclass>
```

Figure 6.12: Contenuto della tabella su MongoDB dopo la cancellazione

6.3 Download Classe

Vogliamo verificare che la funzionalità di download delle classi funzioni correttamente restituendo un file contenente il codice della classe scelta. Utilizzando Postman sono state effettuate le richieste della lista completa delle classi e successivamente del download della classe presente `Calcolatrice.java` e di una non presente per confrontare i risultati.



The screenshot shows a Postman interface with the following details:

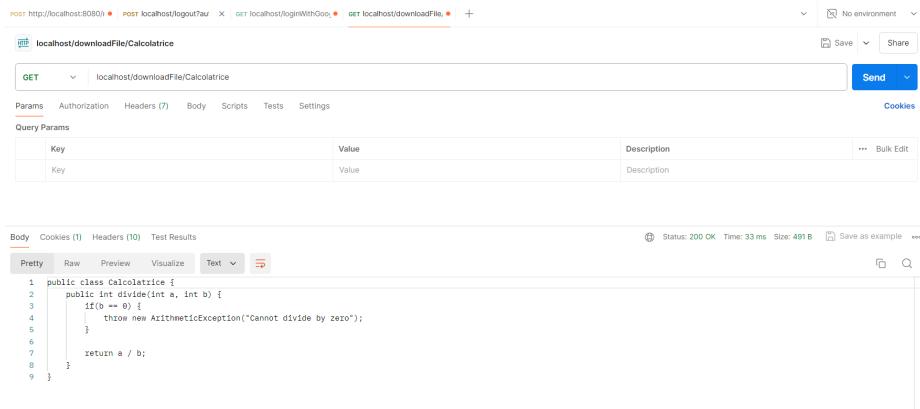
- Request URL:** `http://localhost:8080`
- Method:** GET
- URL:** `localhost/home`
- Headers:** (7 items listed)
- Body:** (Pretty) JSON response:

```

1 [ 
2   {
3     "name": "Calcolatrice",
4     "date": "2024-06-23T00:00:00Z",
5     "difficulty": "Beginner",
6     "code_URL": "Files-Upload/Calcolatrice/Calcolatrice.java",
7     "description": "Test",
8     "category": [
9       "Math",
10      ...
11    ]
12  }
13 ]
14 ]
```
- Headers:** (8 items listed)
- Test Results:** Status: 200 OK, Time: 26 ms, Size: 424 B

Figure 6.13: Lista di tutte le classi disponibili

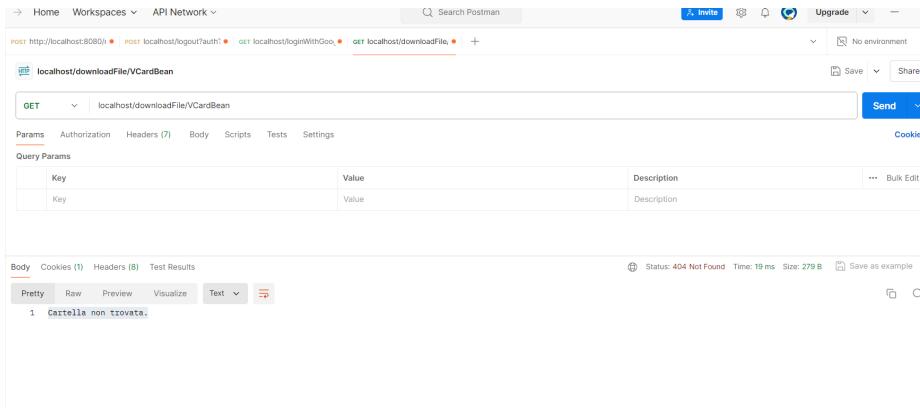
CHAPTER 6. TESTING



The screenshot shows the Postman interface with a successful API call. The URL is `localhost/downloadFile/Calcolatrice`. The response status is 200 OK, time 33 ms, size 491 B. The response body contains the following Java code:

```
1 public class Calcolatrice {
2     public int divide(int a, int b) {
3         if(b == 0) {
4             throw new ArithmeticException("Cannot divide by zero");
5         }
6         return a / b;
7     }
8 }
```

Figure 6.14: Download di una classe presente nella lista



The screenshot shows the Postman interface with an unsuccessful API call. The URL is `localhost/downloadFile/VCardBean`. The response status is 404 Not Found, time 19 ms, size 279 B. The response body contains the message: "Cartella non trovata."

Figure 6.15: Eccezione classe non trovata

6.4 Logout

Per la funzionalità del logout vogliamo testare l'effettiva cancellazione di tutti i cookie associati all'utente autenticato che ha effettuato la richiesta, sono perciò stati visti dal browser i cookie prima e dopo l'operazione di logout.

CHAPTER 6. TESTING

The screenshot shows a POST request to `localhost/logout?authToken=eyJhbGciOiJIUzI1NiJ9eyJzdWIiOiJnaXVwZTk3QGdtYWlsLmNvbSlsImhdCI6MTcxOTE4MTMzMjwiZXhwIjoxNzE5`. The 'Params' tab is selected, showing two entries: 'Key' and 'authToken'. The 'authToken' entry has a value of `eyJhbGciOiJIUzI1NiJ9eyJzdWIiOiJnaXVwZTk3QGdtYWlsLmNvbSlsImhdCI6MTcxOTE4MTMzMjwiZXhwIjoxNzE5MTg...`. The 'Body' tab shows the response: 'Logout successful'. The status bar at the bottom indicates 'Status: 200 OK'.

Figure 6.16: Richiesta di logout di postman

JSESSIONID	87FC91C4...	localh...	/	Session	42	✓		
_stripe_mid	7561c7fa...	.fonta...	/	2025-05-2...	54	✓	St...	
_utma	22653991...	.fonta...	/	2025-07-0...	61			
_utmz	22653991...	.fonta...	/	2024-11-2...	1...			
_fbp	fb.1.1716...	.fonta...	/	2024-08-2...	32		Lax	
_ga	GA1.1.19...	.fonta...	/	2025-07-0...	29			
_ga_BPMS41FJD2	GS1.1.171...	.fonta...	/	2025-07-0...	52			
fblo_689086720...	y	localh...	/	2025-06-2...	21			
jwt	eyJhbGci...	localh...	/	2024-06-2...	1...			

Figure 6.17: Cookie prima della richiesta di logout

Name	value	Domai...	P...	Expires / ...	Si...	H...	Se...	S...	P...
_stripe_mid	7561c7fa...	.fonta...	/	2025-05-2...	54		✓	St...	
_utma	22653991...	.fonta...	/	2025-07-0...	61				
_utmz	22653991...	.fonta...	/	2024-11-2...	1...				
_fbp	fb.1.1716...	.fonta...	/	2024-08-2...	32			Lax	
_ga	GA1.1.19...	.fonta...	/	2025-07-0...	29				
_ga_BPMS41FJD2	GS1.1.171...	.fonta...	/	2025-07-0...	52				
fblo_689086720...	y	localh...	/	2025-06-2...	21				

Figure 6.18: Cookie dopo la richiesta di logout

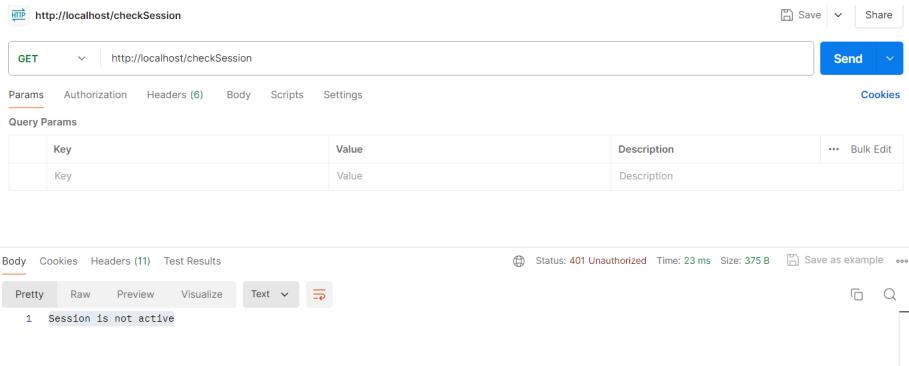


Figure 6.19: Controllo della sessione attiva attraverso Postman

6.5 Registrazione

Il testing della funzionalità di registrazione è fondamentale per assicurare che tutte le modifiche apportate funzionino correttamente. Gli aspetti principali del testing sono:

- Verifica del Caricamento del JavaScript:** Ci siamo assicurati che il `register.js` venga caricato ed eseguito solo dopo il completo caricamento del DOM così da riuscire a rilevare correttamente il `<form>`.

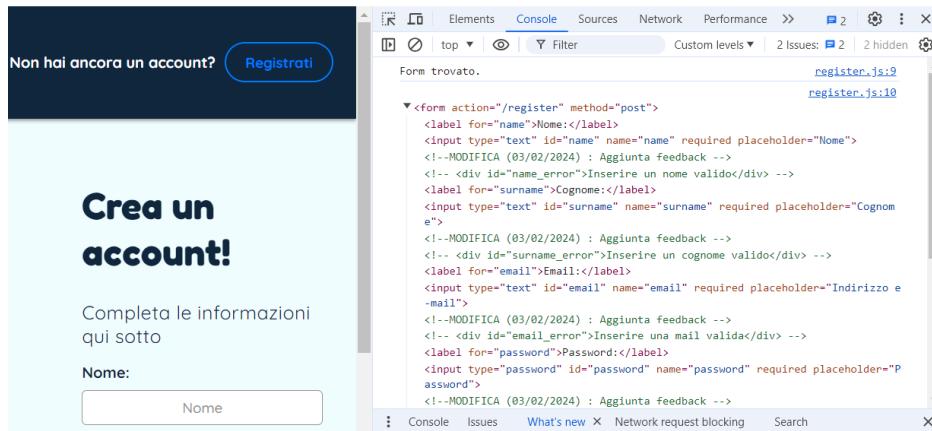


Figure 6.20: Il form viene correttamente identificato

- Test di Invio Dati del Form:** È stato controllato che i dati del form siano inviati

CHAPTER 6. TESTING

correttamente e che il server li elabori senza errori utilizzando il tool Postman.

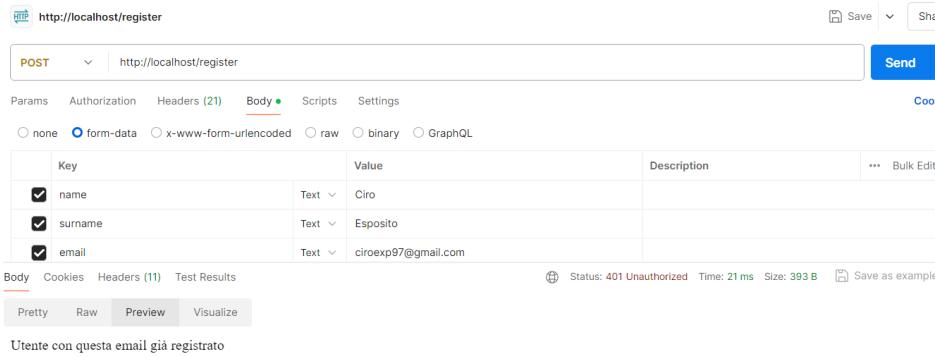
The screenshot shows a POST request to `http://localhost/register`. The request body is set to `form-data` and contains three fields: `name` (value: Ciro), `surname` (value: Esposito), and `email` (value: ciroexp97@gmail.com). The response status is 200 OK, time 12 ms, size 861 B. A message at the bottom says "Registrazione avvenuta con successo, tra poco verrai reindirizzato alla pagina di login."

Figure 6.21: Richiesta corretta di registrazione

The screenshot shows a POST request to `http://localhost/register`. The request body is set to `form-data` and contains three fields: `name` (value: 44), `surname` (value: Esposito), and `email` (value: ciroexp97@gmail.com). The response status is 400 Bad Request, time 14 ms, size 367 B. An error message at the bottom says "Name not valid".

Figure 6.22: Esempio di errore della request, nome non valido

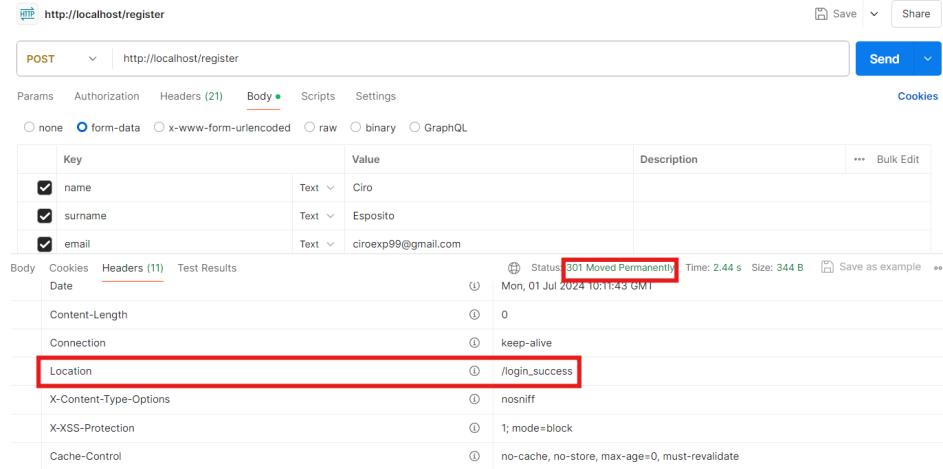
CHAPTER 6. TESTING



The screenshot shows a POST request to `http://localhost/register`. The request body contains three fields: `name` (Ciro), `surname` (Esposito), and `email` (`ciroexp97@gmail.com`). The response status is 401 Unauthorized, indicating that the email address is already registered.

Figure 6.23: Esempio di errore della request, email già registrata

- **Verifica del Reindirizzamento:** Garantire che l'utente venga reindirizzato alla pagina di login dopo una registrazione riuscita, grazie all'header Location.



The screenshot shows a POST request to `http://localhost/register`. The request body contains three fields: `name` (Ciro), `surname` (Esposito), and `email` (`ciroexp99@gmail.com`). The response status is 301 Moved Permanently, with the Location header set to `/login_success`.

Figure 6.24: Headers della richiesta per il reindirizzamento

- **Test del Sistema di Notifiche:** È stato verificato che gli errori venissero gestiti correttamente e venissero notificati in maniera opportuna all'utente.

CHAPTER 6. TESTING

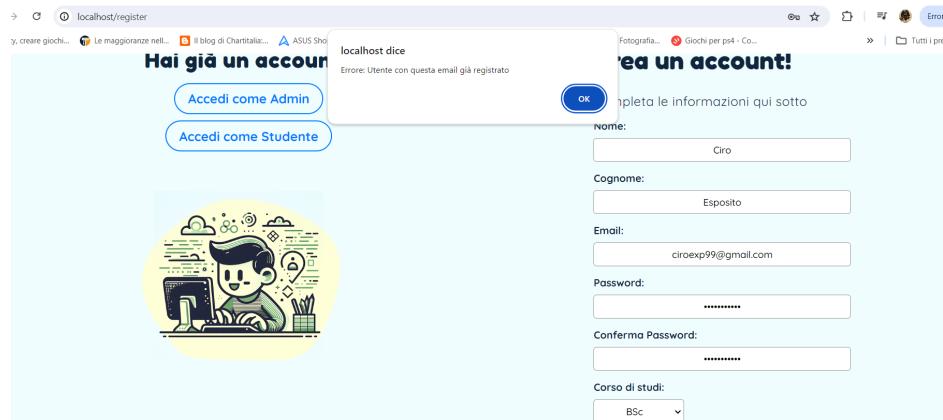


Figure 6.25: Pop-up errore di email già registrata

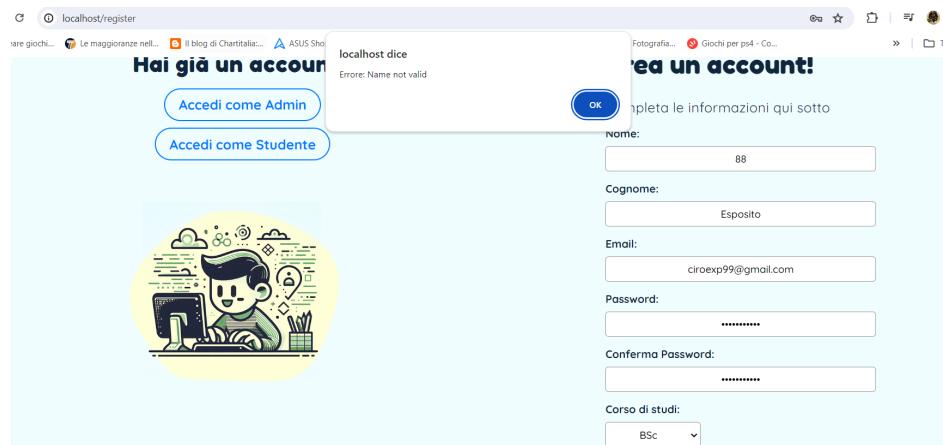


Figure 6.26: Pop-up errore nel campo del nome