



UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
CORSO DI LAUREA IN INGEGNERIA INFORMATICA
CORSO DI SOFTWARE ARCHITECTURE DESIGN
PROF. FASOLINO - A.A. 2023 - 24

Task 2-3

Team A2

Studenti:

Alberto Arola	M63001266	al.arola@studenti.unina.it
Antonio Paolino	M63001394	antoni.paolino@studenti.unina.it
Alfonso Esposito	M63001406	alfonso.esposito17@studenti.unina.it

Repository: <https://github.com/AntoP96/A2-2024> (*forked*)

INDICE

1. Presentazione task T-23 e requisito R9.....	3
1.1 ENACTEST: EuropeanInnovationAllianCefor TEStingeducaTion.....	3
1.2 Inquadramento del task all'interno dell'applicazione.....	3
1.3 Presentazione della vista funzionale.....	4
1.3.1 Task T2.....	4
1.3.2 Task T3.....	4
1.3.3 Requisiti informali.....	4
1.3.4 Requisiti funzionali.....	5
1.3.5 Requisiti non funzionali.....	5
1.3.6 Casi d'uso.....	6
1.3.7 Scenari.....	7
1.3.8 Servizi offerti.....	10
1.3.9 Context Diagram.....	14
1.4 Vista logica.....	15
1.4.1 Pattern architettonico.....	15
1.4.2 Vista C&C.....	16
1.5 Vista dinamica.....	16
1.5.1 Register.....	17
1.5.2 Login.....	18
1.5.3 Logout.....	18
1.5.4 Reset Password.....	19
1.5.5 Change password.....	20
1.6 Tecnologie, framework e Piattaforme adottate.....	21
1.6.1 Front-end.....	21
1.6.2 Back end.....	21
1.6.3 Database.....	22
1.6.4 Deployment.....	23
1.6.5 Gestione delle dipendenze e del building.....	23
1.6.6 Mailing.....	23
1.7 Deployment.....	24
1.7.1 Deployment diagram.....	25
1.8 Criticità emerse.....	25
1.9 Presentazione nuove funzionalità.....	26
1.9.1 Requisito R9.....	26
2. Metodologia di lavoro.....	27
2.1 SCRUM.....	27
2.2 Applicazione del framework Agile SCRUM.....	28
2.3 Artefatti e pratiche SCRUM utilizzate.....	29
2.3.1 Product Backlog.....	29
2.3.2 User story e criteri di accettazione.....	30
2.3.3 Definition of Done.....	31
2.3.4 Planning Poker.....	32

2.4 Prima iterazione.....	33
2.5 Seconda iterazione.....	35
2.5.1 Sprint planning.....	35
2.5.2 Sprint review.....	38
2.5.3 Sprint retrospective.....	39
2.6 Terza iterazione.....	39
2.6.1 Sprint planning.....	39
2.6.2 Sprint review.....	42
2.6.3 Sprint retrospective.....	42
3. Requisito R9.....	43
3.1. Viste funzionali.....	43
3.1.1 Diagramma dei casi d'uso.....	43
3.1.2 Scenari.....	44
3.2. Package diagram.....	46
3.3. Viste dinamiche.....	48
3.3.1 Activity diagram.....	48
3.3.2 Sequence diagram.....	50
3.4. Testing.....	52
3.5. Mockup delle nuove pagine.....	53
3.6. Proposta modifica database T4.....	54
3.7. API.....	55
4. Installazione.....	56
4.1. Configurazione Docker File e Docker Compose.....	56
4.2. Maven Install e avvio del Docker File.....	59
5. Demo.....	60

1. Presentazione task T-23 e requisito R9

1.1 ENACTEST: EuropeanInnovationAllianCefor TESTingeducaTion

Lo scopo dell'applicazione è realizzare un gioco che consente agli studenti di Software Testing di addestrare i Task di Unit-Testing.

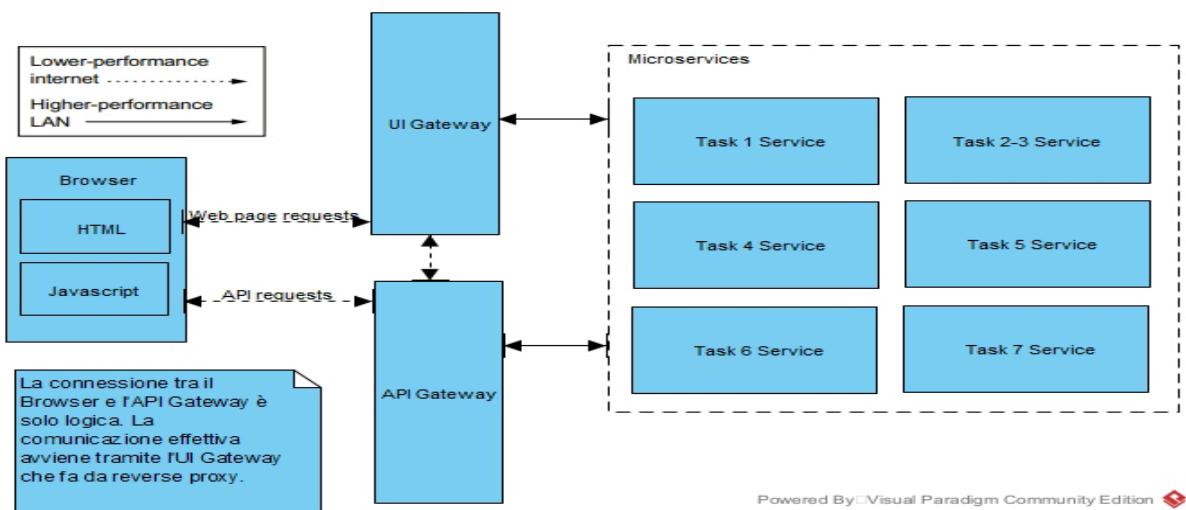
Il gioco verrà strutturato in modo che gli studenti (o le squadre) competono contro strumenti in grado di generare automaticamente casi di test JUnit (ad esempio Randoop o EvoSuite).

L'approccio basato su Gamification ha lo scopo di spronare gli studenti a:

- Utilizzare diversi tipi di tecniche di progettazione dei test.
- Utilizzare il framework di testing JUnit per scrivere i loro test.
- Comprendere il contributo degli strumenti di testing automatico nei processi di testing, esplorando i loro punti di forza e debolezza.

1.2 Inquadramento del task all'interno dell'applicazione

L'applicazione è stata implementata con un design architettonico a microservizi in modo tale che ogni gruppo potesse lavorare contemporaneamente e indipendentemente alle diverse funzionalità dell'applicazione.



I vari microservizi si interfacciano all' UI gateway il quale a sua volta si interfaccia con il browser in modo tale da fornire le diverse viste dell'applicazione agli utenti. Inoltre i microservizi si interfacciano anche con un API gateway in modo tale da gestire sia le richieste alle API dall'esterno sia per poter mettere a disposizione e utilizzare le API con gli altri microservizi.

In particolare il nostro team si è concentrato di analizzare e modificare il task T 2-3 sia per poterlo migliorare andando ad individuare eventuali criticità presenti, sia per aggiungere nuove funzionalità.

1.3 Presentazione della vista funzionale

1.3.1 Task T2

L'applicazione deve consentire agli studenti di registrarsi per poter conservare la storia delle attività svolte, oppure per accedere a requisiti di gioco più complessi. All'atto della registrazione, lo studente fornirà nome, cognome, un indirizzo e-mail valido ed una password, il sistema dopo aver controllato la validità dei dati forniti, aggiungerà il giocatore all'elenco dei giocatori registrati e gli assocerà un ID univoco.

Sarebbe desiderabile raccogliere anche altre informazioni sugli studenti, come il corso di studi a cui sono iscritti (Bachelor, Master Degree, o altro).

1.3.2 Task T3

All'atto della autenticazione, lo studente fornirà l'indirizzo e-mail fornito per la registrazione e la relativa password, il sistema dopo aver controllato la validità dei dati forniti, autenticherà il giocatore e gli fornirà una schermata per l'accesso alle funzionalità di gioco o di consultazione delle sessioni di gioco passate.

Inoltre, verrà fornita allo studente la possibilità di resettare la password. Lo studente dovrà fornire l'email, il sistema la ricercherà nell'elenco degli studenti registrati e se presente invierà un'email con un token univoco. Alla ricezione del token lo studente potrà reimpostare la password inserendo in una nuova schermata email, token e password (entro 24 ore dalla generazione del token).

1.3.3 Requisiti informali

All'atto della registrazione, lo studente fornisce nome, cognome, un indirizzo e-mail, una password e il corso di studi cui è iscritto. Viene, dunque, controllata la validità dei dati forniti: il nome e il cognome non possono contenere caratteri speciali (ovvero < > & () , % ' ? + ") o numeri e devono avere lunghezza da 2 a 30 caratteri; l'indirizzo e-mail deve contenere necessariamente il carattere '@' e almeno un punto; la password deve contenere da 8 a 16 caratteri, di cui almeno un carattere minuscolo, maiuscolo e un numero; viene reinserita nuovamente la password; il corso di studi viene scelto tra: Bachelor (BSc), Master Degree (MSc), o 'ALTRO'. Va controllato che nessun campo sia lasciato vuoto e che l'e-mail non sia già registrata. Viene effettuato un check tra le due password inserite (devono essere necessariamente uguali). Un'ultima verifica viene

effettuata inviando all'utente una e-mail contenente l'id univoco che gli è stato assegnato.

All'atto dell'autenticazione, lo studente registrato può accedere alla sua area riservata fornendo l'indirizzo e-mail inserito in fase di registrazione e la relativa password. Viene poi effettuato un doppio controllo: se l'utente non esiste oppure la password inserita è errata, il login fallisce. Se l'autenticazione fornisce un esito positivo, appare una schermata che permette l'accesso alle funzionalità di gioco o di consultazione delle sessioni di gioco passate.

Per ogni sessione di autenticazione correttamente effettuata all'id utente sarà associato un token. Nel caso in cui l'utente registrato abbia dimenticato la password, è possibile impostarne una nuova. L'applicazione invierà una e-mail all'utente contenente un token per il reset password. L'utente, nella apposita schermata, dovrà inserire nuovamente l'e-mail, il token che gli è stato inviato e la nuova password da impostare. Infine, viene implementata una funzionalità che permette all'utente registrato di effettuare il logout.

1.3.4 Requisiti funzionali

RF1. Gli studenti possono registrarsi fornendo nome, cognome, email, password e il corso di studi a cui sono iscritti.

RF2. Il sistema, in caso di avvenuta registrazione, deve inviare all'utente registrato un'e-mail contenente l'id.

RF3. Uno studente può accedere al sistema con le credenziali immesse durante la registrazione

RF4. In caso di login errato, il sistema deve restituire un errore.

RF5. Il sistema deve permettere all'utente registrato di impostare una nuova password in caso abbia dimenticato la propria.

RF6. Il sistema deve permettere all'utente registrato di accedere all'area riservata solo dopo aver effettuato il login.

RF7. Uno studente che ha effettuato il login può terminare la sua sessione di gioco effettuando il logout.

1.3.5 Requisiti non funzionali

RNF1. Sicurezza:

- È raccomandabile criptare la password e non salvarla in chiaro nel database.

- In fase di registrazione l'identità dello studente viene verificata anche tramite un reCAPTCHA.
- Si deve effettuare una ricerca di tutti i caratteri pericolosi: < > & () , % ' ? + poiché possono essere utilizzati per eventuali attacchi, ad esempio l'SQL injection.

RNF2. Interoperabilità: il sistema deve essere in grado di interagire con altri sistemi per lo scambio di informazioni, infatti, l'ID associato al giocatore viene fornito agli altri sistemi che ne necessitano.

RNF3. Usabilità: il sistema deve fornire un funzionamento intuitivo e facilmente apprendibile.

RNF4. Accuratezza: il sistema deve fornire i giusti o concordati risultati o effetti.

RNF5. Efficienza: il sistema deve realizzare le funzioni richieste nel minor tempo possibile ed utilizzando nel miglior modo le risorse necessarie.

1.3.6 Casi d'uso

Attore primario:

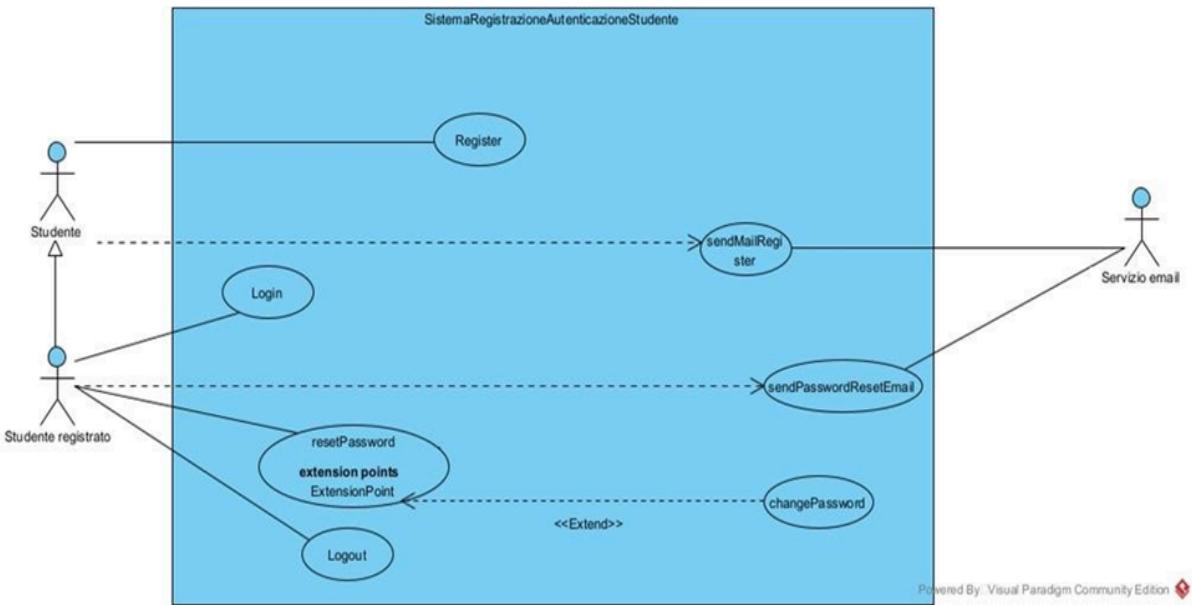
- Studente
- Studente registrato
- Servizio e-mail

Attore secondario:

- Studente registrato
- Servizio e-mail

I casi d'uso sono:

- UC1: Register
- UC2: Login
- UC3: sendMailRegister
- UC4: sendPasswordResetEmail
- UC4: Logout
- UC5: resetPassword
- UC6: changePassword



In particolare si nota come il caso d'uso `resetPassword` viene esteso da `changePassword` in quanto dopo aver resettato la password la nuova dovrà ovviamente essere diversa dalla precedente. Inoltre lo studente registrato sarà un attore secondario per lo use case `sendPasswordResetEmail` in quanto la funzionalità si attiverà nel momento in cui lo studente registrato cambi la propria password.

Infine lo studente è un attore secondario per lo use case `sendMailRegister` in quanto una e-mail di avvenuta registrazione sarà inviata nel momento in cui un utente non già registrato si registri.

1.3.7 Scenari

UC1: Register

Attore primario	Studente
Attore secondario	-
Descrizione	Permette ad uno studente di registrarsi
Pre-Condizioni	Lo studente non deve essere già registrato
Sequenza di eventi principale	<ol style="list-style-type: none"> 1) Lo studente apre la schermata di login; 2) Lo studente clicca su "Non sei ancora registrato? Registrati." 3) Lo studente, nella schermata di registrazione, inserisce i dati richiesti e clicca su "Invia"; 4) Lo studente riceve il suo id tramite posta.
Post-Condizioni	Lo studente può autenticarsi
Casi d'uso correlati	-
Sequenza di eventi alternativi	La registrazione fallisce se i dati inseriti non sono corretti. In tal caso, il sistema restituisce un messaggio di registrazione non effettuata.

UC2: Login

Attore primario	Studente registrato
Attore secondario	-
Descrizione	Permette ad uno studente di effettuare il login e iniziare a giocare.
Pre-Condizioni	Lo studente deve essersi registrato.
Sequenza di eventi principale	1) Lo studente registrato, nella schermata di login, inserisce e-mail e password nell'apposito spazio; 2) Clicca su "Accedi".
Post-Condizioni	Lo studente visualizza il token di accesso.
Casi d'uso correlati	-
Sequenza di eventi alternativi	1) Il login fallisce se e-mail e/o password sono errati, mostrando un messaggio di errore; 2) Se l'utente ha dimenticato la password può reimpostarla.

UC3: sendMailRegister

Attore primario	Servizio e-mail
Attore secondario	Studente
Descrizione	Si invia l'e-mail di conferma in caso di corretta registrazione.
Pre-Condizioni	Lo studente deve aver inserito i dati e cliccato su "Invia".
Sequenza di eventi principale	1) Il servizio e-mail invia un messaggio contenente l'id che è stato assegnato allo studente correttamente registrato.
Post-Condizioni	Lo studente può procedere con il login.
Casi d'uso correlati	-
Sequenza di eventi alternativi	-

UC4: sendPasswordResetEmail

Attore primario	Servizio e-mail
Attore secondario	Studente registrato
Descrizione	Il servizio e-mail invia il messaggio per resettare la password.
Pre-Condizioni	Lo studente registrato deve aver richiesto il reset della password.
Sequenza di eventi principale	1) Il servizio e-mail invia un messaggio contenente il token per il reset della password.
Post-Condizioni	Lo studente registrato può procedere con il cambio password.
Casi d'uso correlati	-
Sequenza di eventi alternativi	-

UC5: Logout

Attore primario	Studente registrato
Attore secondario	-
Descrizione	Permette ad uno studente registrato di effettuare il logout.
Pre-Condizioni	Lo studente deve essersi loggato.
Sequenza di eventi principale	<ul style="list-style-type: none"> 1) Clicca su "Logout".
Post-Condizioni	Lo studente torna alla pagina di login.
Casi d'uso correlati	-
Sequenza di eventi alternativi	-

UC6: resetPassword

Attore primario	Studente registrato
Attore secondario	Servizio e-mail
Descrizione	Permette ad uno studente registrato di resettare la password.
Pre-Condizioni	Lo studente deve essere registrato nel sistema
Sequenza di eventi principale	<ul style="list-style-type: none"> 1) Lo studente registrato, nella pagina di login, clicca su "Hai dimenticato la password?" 2) Lo studente registrato viene reindirizzato nella pagina "Recupero password", inserisce l'e-mail e clicca su "invia". 3) Il servizio e-mail invia un messaggio contenente un token per il reset password.
Post-Condizioni	La password viene resettata.
Casi d'uso correlati	È esteso da "Change_password".
Sequenza di eventi alternativi	-

UC7: changePassword

Attore primario	Studente registrato
Attore secondario	Servizio e-mail
Descrizione	Permette ad uno studente registrato di modificare la password.
Pre-Condizioni	Lo studente deve aver resettato la password
Sequenza di eventi principale	<ul style="list-style-type: none"> 1) Lo studente registrato visualizza l'e-mail, copia il token ricevuto, clicca su "Hai già ricevuto il token? Cambia la password". 2) Lo studente registrato viene reindirizzato alla schermata "Change Password" dove scrive e-mail, token ricevuto e la password nuova da impostare. 3) Lo studente registrato clicca su "Reimposta".
Post-Condizioni	La password viene modificata.
Casi d'uso correlati	Estende "Reset_password".
Sequenza di eventi alternativi	-

1.3.8 Servizi offerti

Le API (Application Programming Interface) sono un insieme di strumenti che consentono agli utenti del sistema di interagire con esso in modo programmato, tramite scambio di dati e richieste. Le API sono fondamentali per l'architettura software del sistema, poiché consentono una maggiore flessibilità e scalabilità del sistema.

Il sistema è composto da diverse API che consentono agli utenti di accedere e manipolare i dati. Le API sono progettate per essere modulari e consentono un facile accesso ai dati attraverso richieste HTTP.

L'API consente agli utenti di accedere ai dati relativi agli utenti del sistema. Gli utenti possono effettuare richieste GET per visualizzare i dati degli utenti, e richieste POST per aggiungere o modificare i dati degli utenti. Le richieste devono essere autenticate con le credenziali utente.

Postman è un'applicazione per il testing di API che consente agli sviluppatori di creare, testare e documentare le loro API in modo rapido ed efficiente. Una delle caratteristiche principali di Postman è la possibilità di inviare e ricevere dati in formato JSON.

JSON (JavaScript Object Notation) è un formato di scambio dati leggero e flessibile utilizzato per rappresentare dati strutturati. Postman consente di inviare richieste HTTP e ricevere risposte in formato JSON, semplificando il processo di test delle API che utilizzano questo formato.

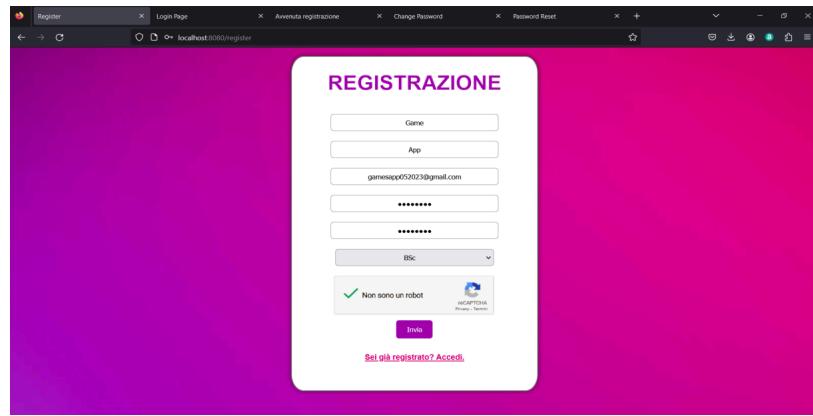
Postman consente agli sviluppatori di creare e modificare facilmente dati in formato JSON utilizzando il proprio editor integrato, che offre una sintassi intuitiva e una visualizzazione dei dati strutturati. Inoltre, Postman offre la possibilità di eseguire test automatizzati sulle API che utilizzano dati in formato JSON, garantendo una maggiore efficienza e accuratezza nel testing delle API.

Registrazione

L'API di registrazione è accessibile all'URL: <http://localhost:8080/register> e richiede i seguenti parametri: nome, cognome, e-mail, password, conferma della password, studi e una risposta al ReCAPTCHA di Google.

La funzione utilizza un meccanismo di sicurezza per evitare attacchi robotici effettuando una chiamata HTTP per verificare la risposta del ReCAPTCHA di Google, utilizzando un'API di terze parti. Se la verifica del ReCAPTCHA fallisce, viene restituito un errore.

Viene verificata la validità dei dati inseriti dall'utente, come la lunghezza del nome e del cognome, il formato dell'e-mail e la complessità della password. Se i dati inseriti non sono validi, viene restituito un messaggio di errore appropriato come risposta. Altrimenti, nel caso di registrazione corretta, viene restituito un messaggio e viene inviata un'e-mail di conferma all'indirizzo fornito dall'utente.

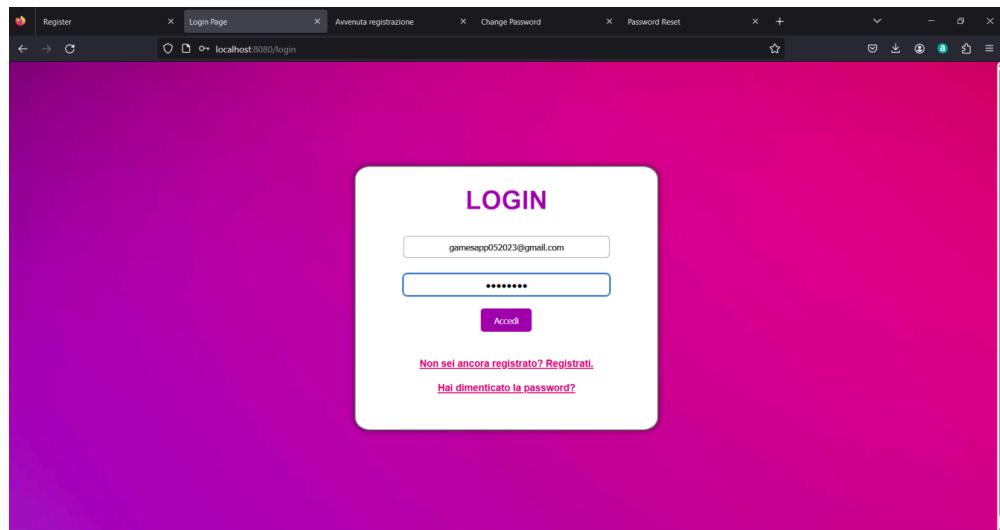


Login

L'API di login è accessibile all'URL: <http://localhost:8080/login> e accetta come parametri l'e-mail e la password dell'utente. Questa API è inclusa di richiesta GET per poter accedervi tramite interfaccia web.

La funzione verifica prima che l'e-mail sia registrata nel sistema e successivamente verifica che la password effettivamente coincida con quella inserita tramite un algoritmo di hashing. In caso di fallimento di una delle due condizioni viene restituito un messaggio di errore, altrimenti in caso corretto viene restituito il token di autenticazione.

L' API risulta avere un buon livello di sicurezza poiché utilizza un meccanismo di autenticazione basato su token e crittografia per proteggere le informazioni dell'utente.

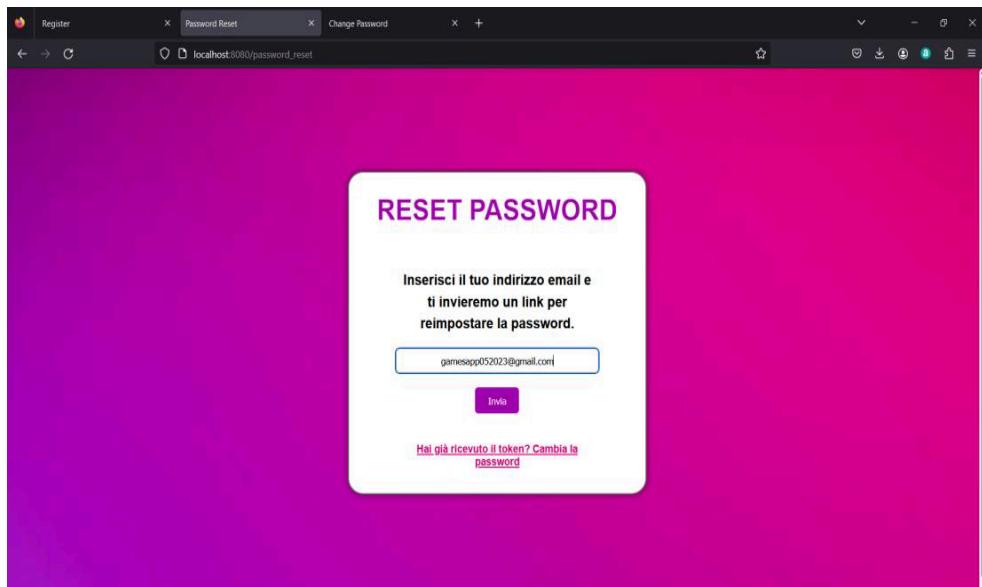


Reset Password

L'API di reset password è accessibile all'URL: http://localhost:8080/password_reset e accetta come parametri l'e-mail dell'utente che richiede il reset della password. In questa API è inclusa la richiesta GET per poter accedervi tramite interfaccia web.

La funzione, insieme all'API di password change, consente agli utenti di reimpostare la propria password in caso di smarrimento o dimenticanza della stessa. Questa funzione verifica che la e-mail inserita appartenga a un utente registrato e in caso di risposta negativa restituisce un messaggio di errore, mentre, in caso di risposta positiva, viene generato un token di reset della password e tramite un servizio e-mail viene inviato all'utente.

L'API prevede un ulteriore livello di sicurezza poiché in caso di errore del servizio mail viene restituito lo stato HTTP 500 Internal Server Error e il messaggio "Failed to send password reset email".



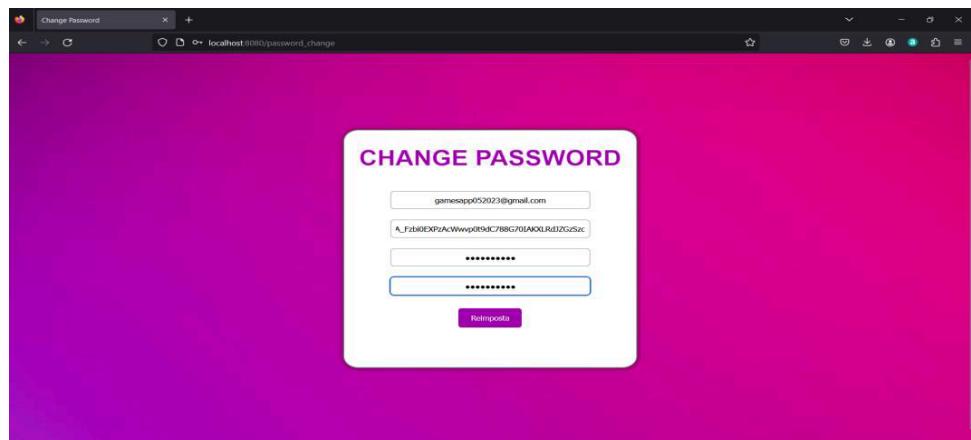
Change password

L'API di change password è accessibile all'URL:http://localhost:8080/password_change e accetta come parametri l'e-mail dell'utente, token, password, conferma della password. In questa API è inclusa la richiesta GET per poter accedervi tramite interfaccia web.

Il parametro "token" è il token di reset della password che l'utente ha ricevuto per e-mail tramite l'API "password_reset".

La funzione verifica che l'e-mail inserita esista e sia associata effettivamente al token inserito, altrimenti viene restituito un messaggio di errore. Successivamente, viene verificato che la nuova password scelta dall'utente sia valida e nel caso in cui non rispetta i criteri richiesti viene restituito un messaggio di errore.

Se tutte le verifiche precedenti sono state superate con successo viene restituito un messaggio e la nuova password viene criptata e salvata nel database per l'utente corrispondente. Inoltre, il token di reset della password viene impostato a null per indicare che è stato utilizzato con successo.



Logout

L'API di logout è accessibile all'URL: <http://localhost:8080/logout> ed è progettata per consentire agli utenti di eseguire il logout dal sistema, ovvero di invalidare un token di autenticazione attivo per un determinato utente. L'API non espone nessuna interfaccia, poiché si presuppone che venga richiamata da interfacce adibite ad altri team di sviluppo.

La richiesta contiene un parametro "authToken" che rappresenta il token di autenticazione attivo dell'utente che vuole eseguire il logout.

La funzione verifica che il authToken sia effettivamente associato ad un utente loggato e in caso di risposta negativa viene visualizzato un errore. In caso di risposta positiva, l'oggetto "AuthenticatedUser" viene eliminato dal database, invalidando così il token di autenticazione.

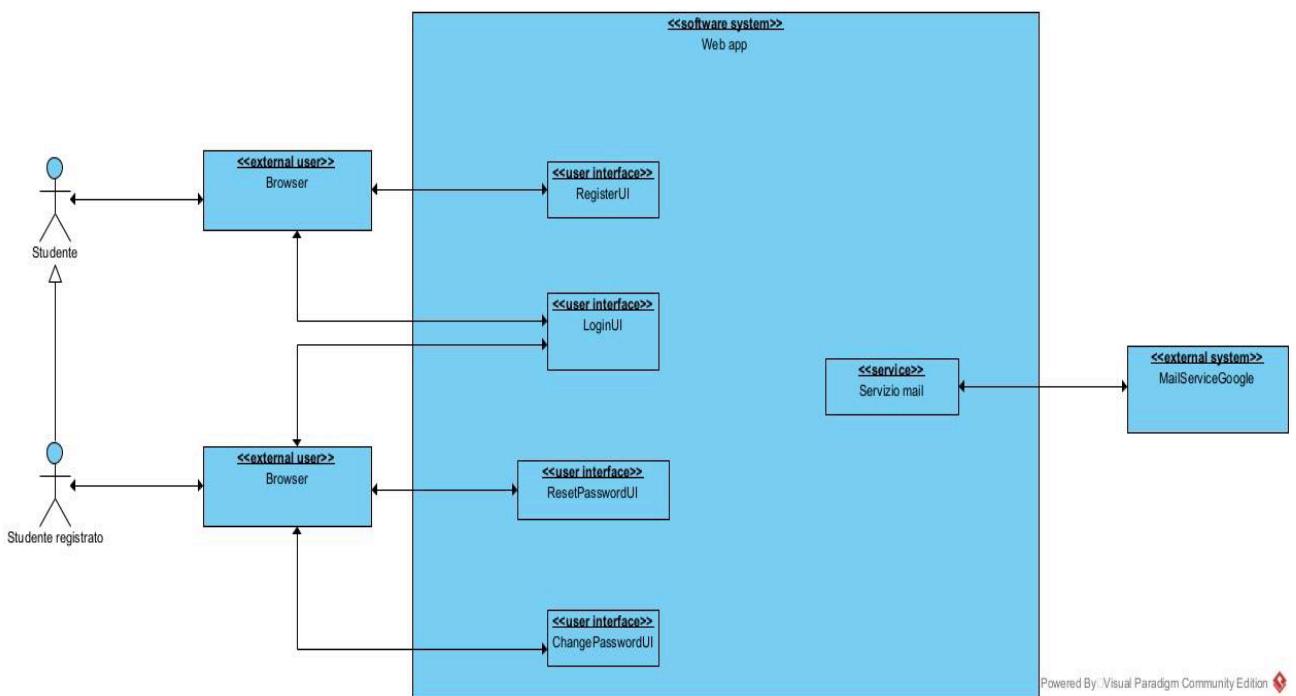
Get ID

L'API di `get_id` è accessibile all'URL: http://localhost:8080/get_ID e richiede come parametri l'username e la password dell'utente. L'API non espone nessuna interfaccia, poiché è stata esplicitamente richiesta da un altro team di sviluppo (G14).

La funzione cerca l'utente nel repository utilizzando l'e-mail fornita e verifica che la password corrisponda alla password dell'utente nel repository. In caso di risposta negativa, la funzione restituirà un valore -1 per indicare l'errore, altrimenti se l'autenticazione è riuscita, la funzione restituirà l'ID dell'utente.

1.3.9 Contex Diagram

Tramite il contrex diagramm è possibile mettere in evidenza le interazioni tra i servizi offerti dal microservizio e gli attori esterni al sistema.



Lo studente tramite il browser può accedere a delle interfacce utenti che mettono a disposizione la possibilità sia di registrarsi che di effettuare il login.

Lo studente registrato, sempre tramite browser, può accedere alle interfacce che gli permettono di resettare e di cambiare la propria password.

Il servizio mail, utilizzato sia nel cambio password che in fase di registrazione, interagisce soltanto con il mail service di Google senza utilizzare alcuna interfaccia.

Nota: nel diagramma non è presente la funzionalità di logout la quale non fornisce un'interfaccia specifica ma dovrà essere presente ad esempio un bottone in ogni interfaccia presente nel sistema che interagisce con un utente registrato.

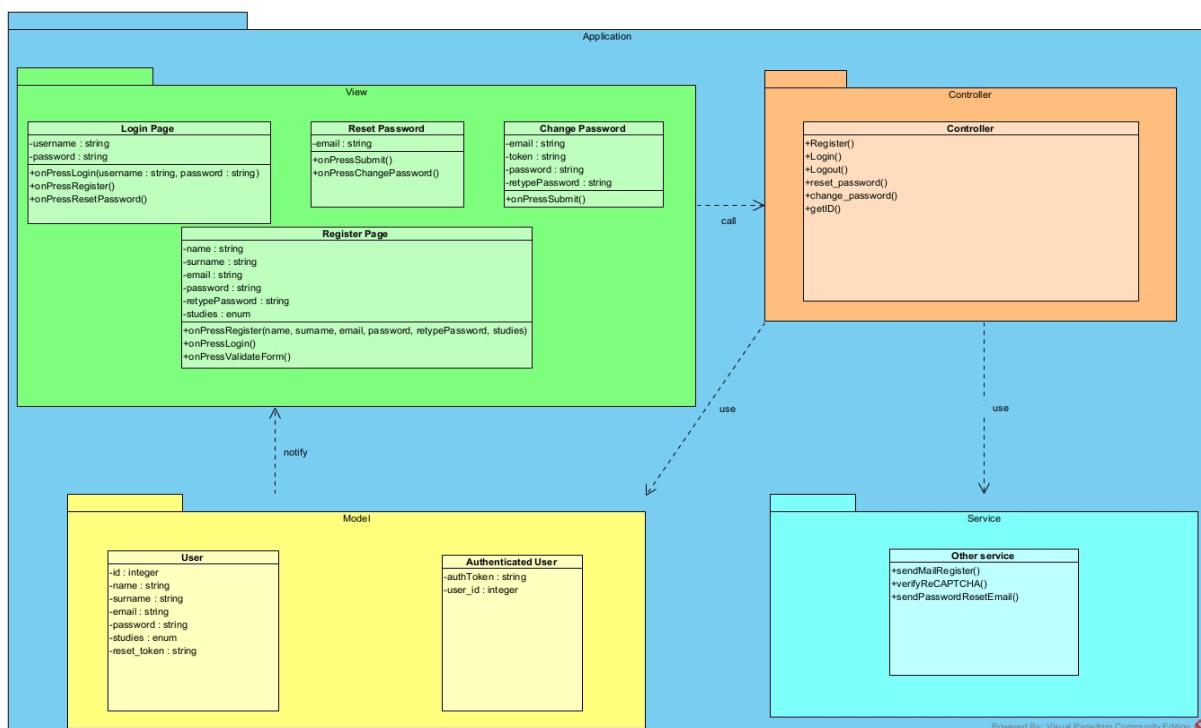
1.4 Vista logica

1.4.1 Pattern architettonale

Il pattern utilizzato per questo microservizio è l'MVC (Model-View-Controller) che si presta in maniera efficace alle specifiche richieste. Quest'ultimo, infatti, offre numerosi vantaggi:

- **Separazione delle responsabilità:** MVC aiuta a separare le diverse responsabilità all'interno di un'applicazione. Il modello (Model) rappresenta i dati e la logica di business dell'applicazione, la vista (View) si occupa della presentazione dell'interfaccia utente e il controller (Controller) gestisce le interazioni tra il modello e la vista. Questa separazione consente di gestire in modo indipendente le diverse componenti dell'applicazione e favorisce la modularità e la manutenibilità del codice.
- **Riutilizzo del codice:** Grazie alla separazione delle responsabilità, il pattern MVC facilita il riutilizzo del codice.
- **Testabilità:** MVC favorisce l'indipendenza dei componenti dell'applicazione, rendendo più facile il testing, infatti, abbiamo potuto testare separatamente l'interfaccia utente e il back-end della nostra applicazione.
- **Scalabilità:**

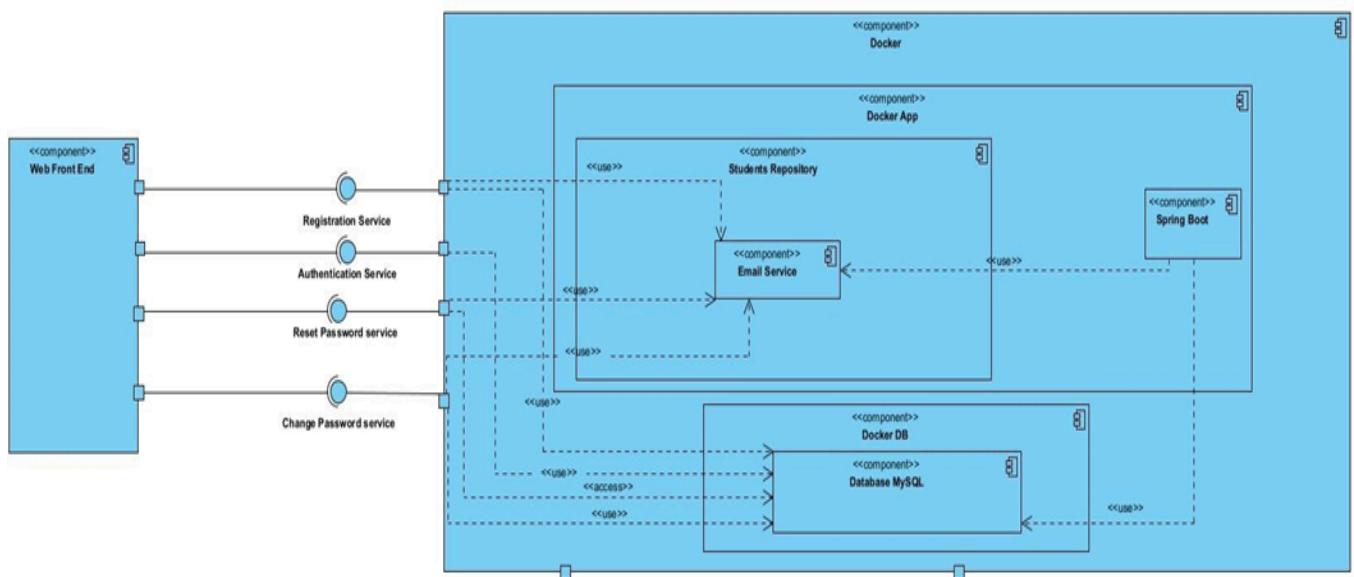
Di seguito, attraverso il Package diagram, vedremo come è stato adottato il pattern MVC per la suddivisione delle classi in package, ai quali sono associate le rispettive funzionalità.



Il Controller si occupa della logica di business per poter eseguire i servizi offerti dal microservizio. Il package View contiene le diverse pagine messe a disposizione per gli utenti tramite browser. Il package model contiene le classi i cui dati saranno salvati su DB. Infine il package Service contiene i servizi esterni di cui il microservizio si serve come il RECaptcha di Google e il servizio mail dello stesso.

1.4.2 Vista C&C

Nel diagramma dei componenti sottostante, si trova il componente front-end web che può richiedere diverse interfacce di servizio, tra cui RegistrationService, AuthenticationService, ResetPasswordService e ChangePasswordService. Alcune di queste interfacce visualizzano e utilizzano le informazioni contenute in altri componenti (indicati con "<<use>>"), mentre altre interfacce accedono solamente alle informazioni contenute negli altri componenti (indicati con "<<access>>"). Queste interfacce sono rese disponibili da un componente principale chiamato "Docker," il quale è collegato tramite la notazione a lollipop, e consentono al front-end web di comunicare con i due sottocomponenti di Docker: docker-app, responsabile della gestione della logica dell'applicazione, e docker-db, che gestisce l'accesso ai dati.



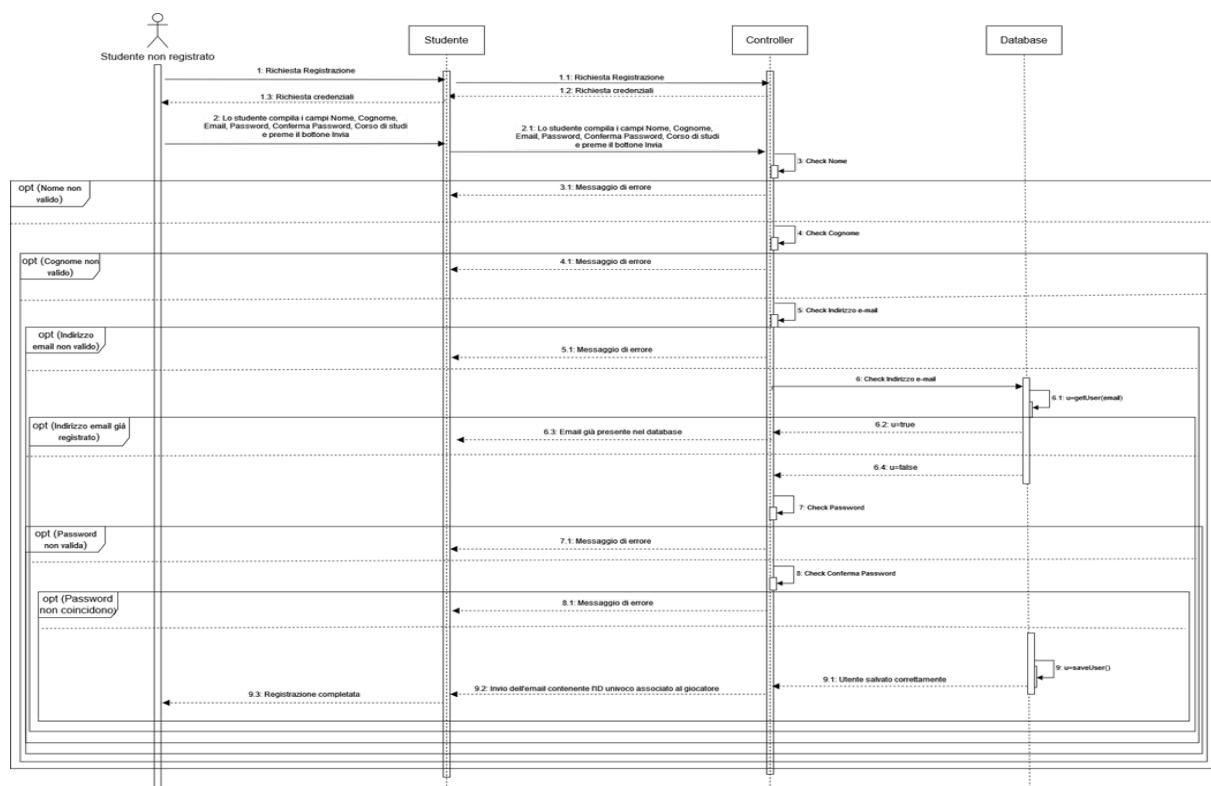
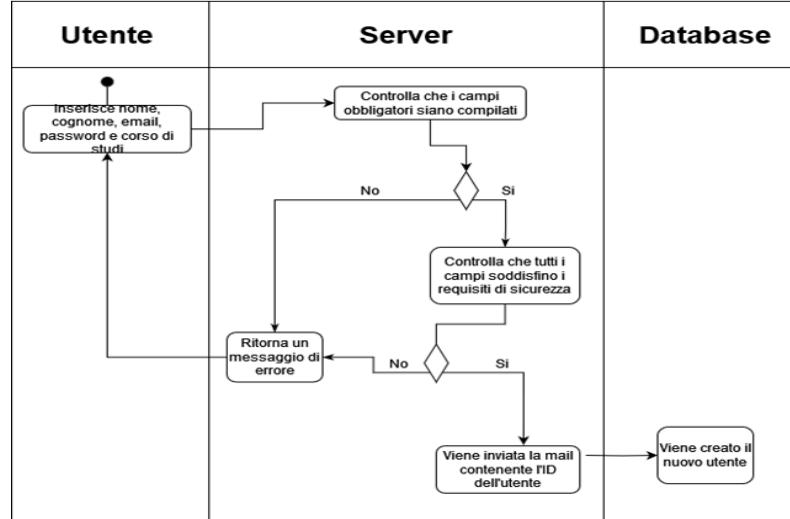
1.5 Vista dinamica

Per comprendere meglio il flusso delle operazioni, utilizzeremo, per ciascuno scenario, sia il diagramma di attività che il diagramma di sequenza.

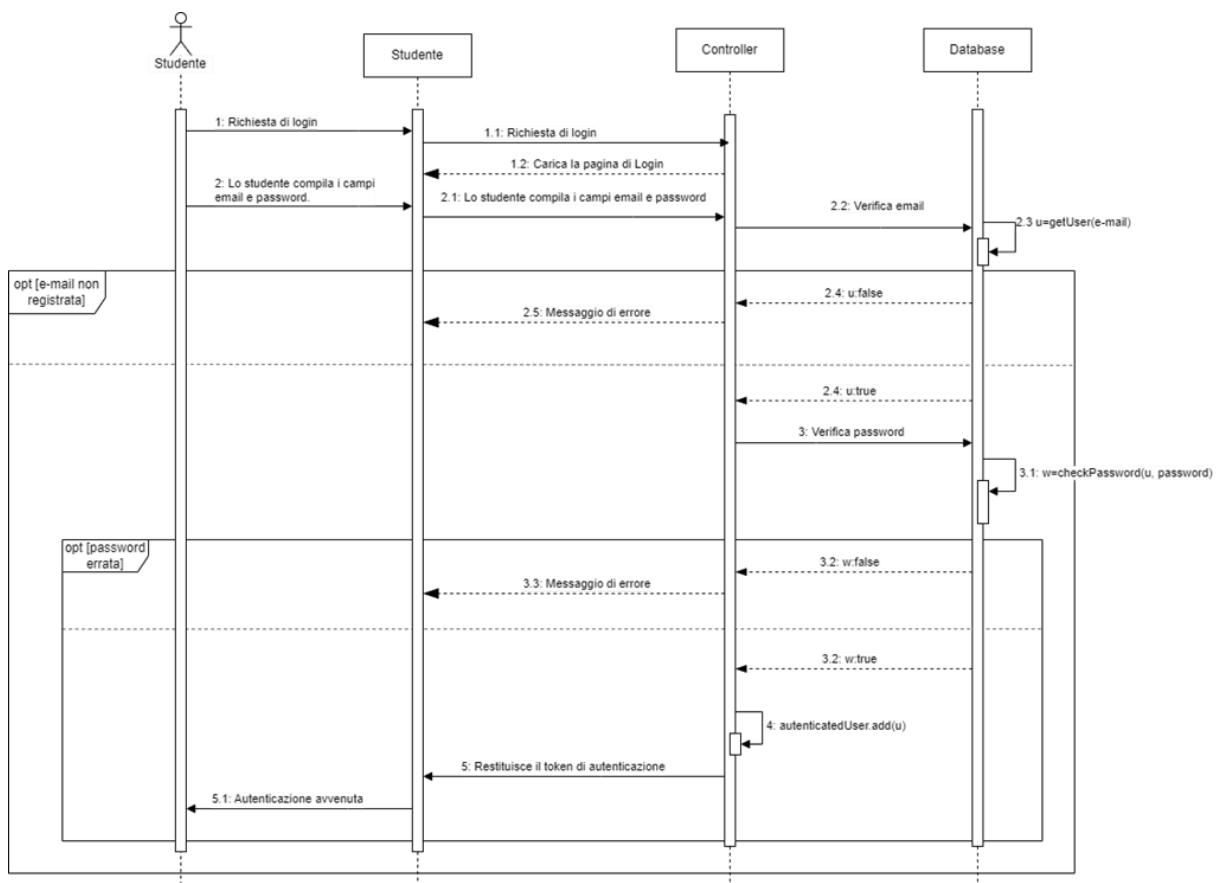
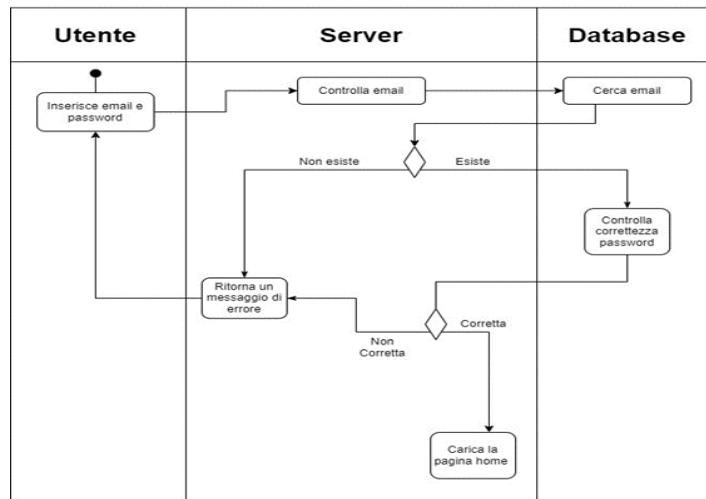
Il diagramma di attività consente di rappresentare in modo grafico le attività, le azioni e le decisioni che compongono il processo, mentre il diagramma di sequenza viene impiegato per rappresentare l'interazione tra gli oggetti o le entità all'interno di un sistema software in un ordine cronologico. In un diagramma di sequenza di tipo MVC, il

controller riceve la richiesta dall'utente e la inoltra ai dati del model (Database), che li elabora e li restituisce al controller. Quindi, il controller utilizza tali dati per aggiornare la view (Studente) e restituisce la view aggiornata all'utente.

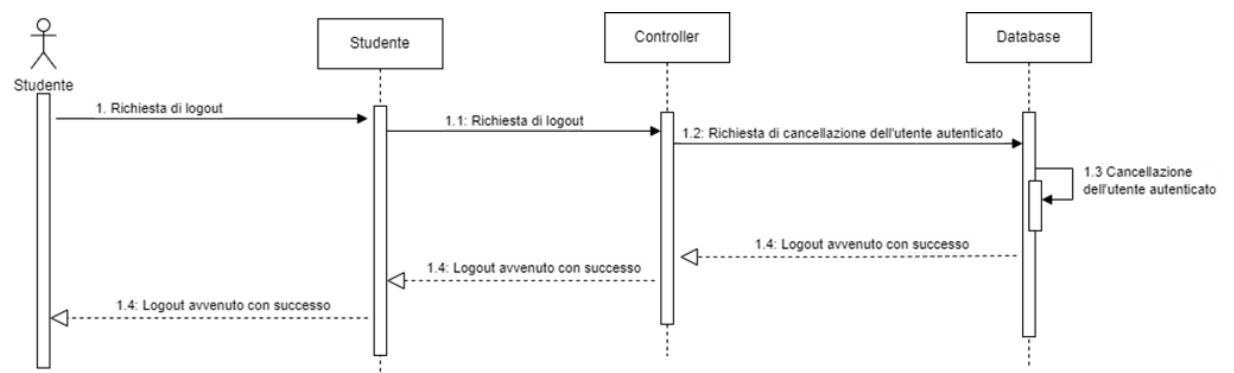
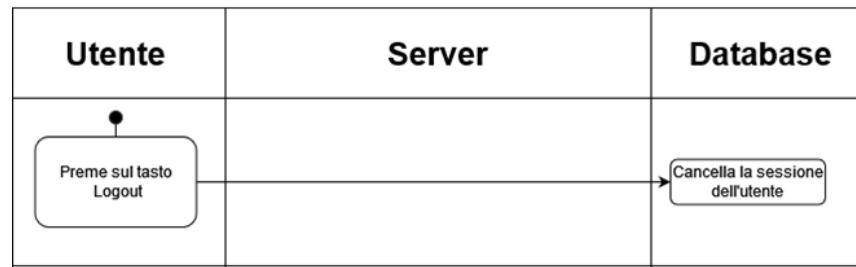
1.5.1 Register



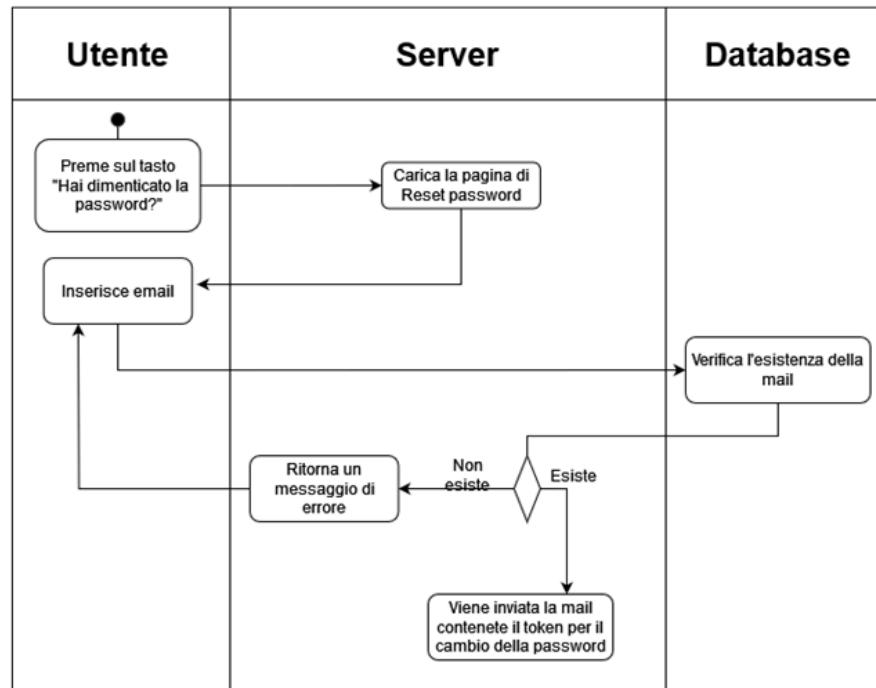
1.5.2 Login

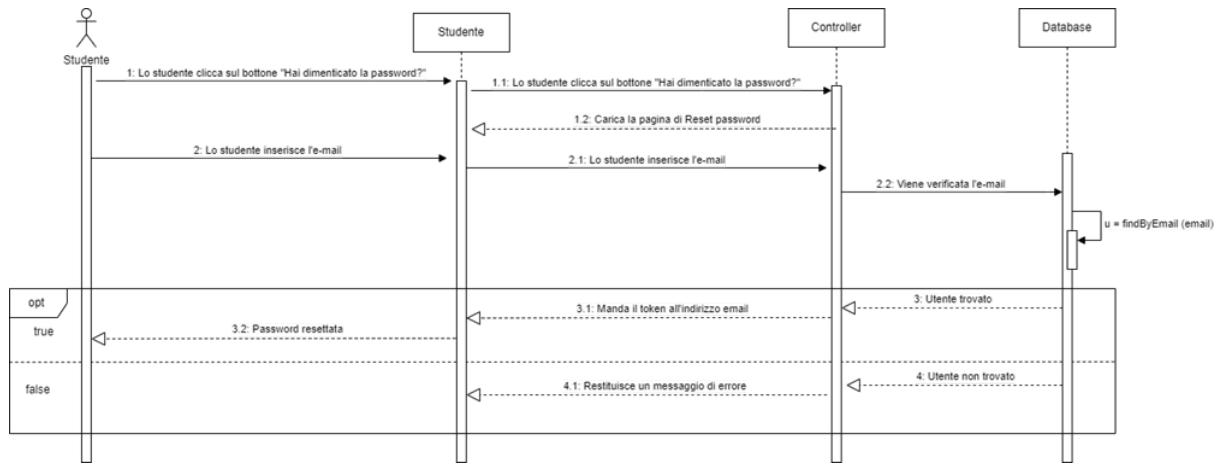


1.5.3 Logout

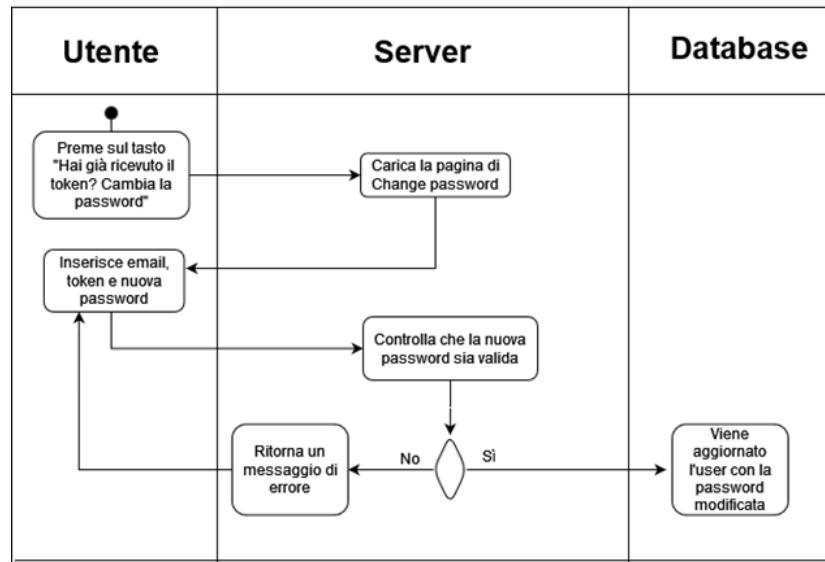


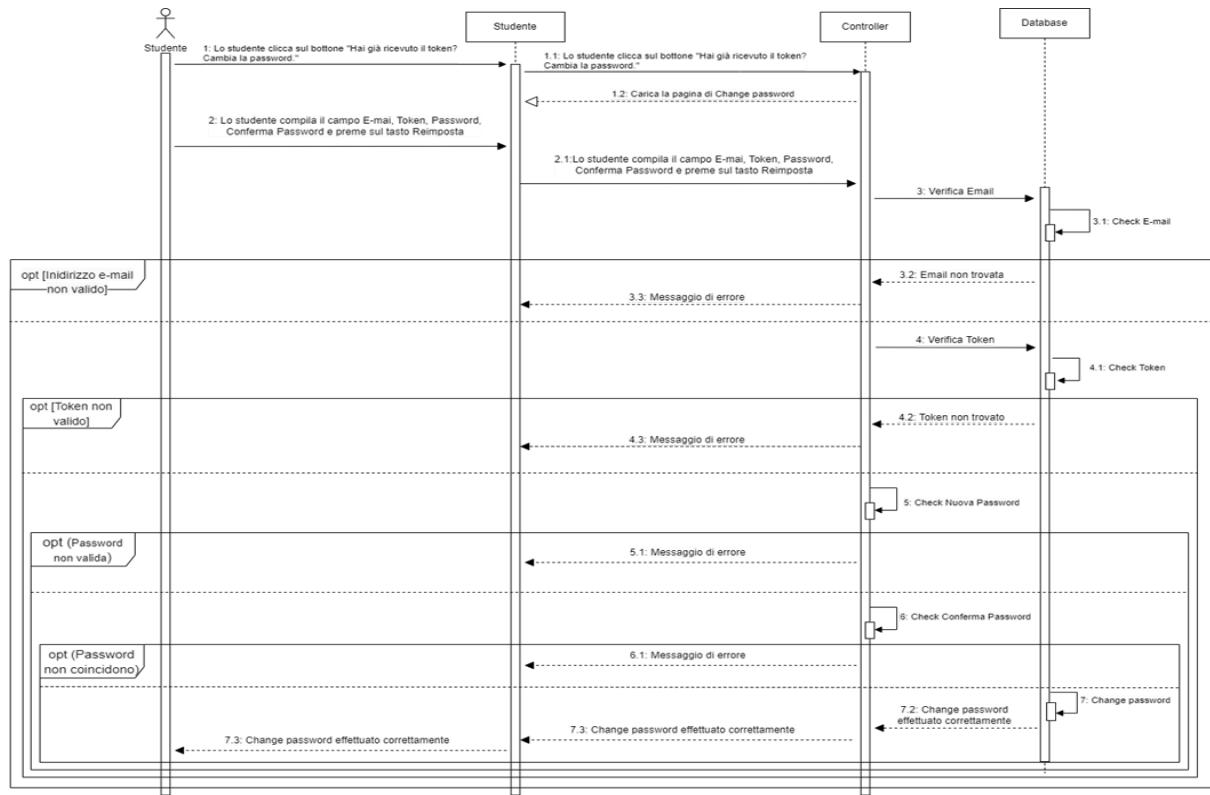
1.5.4 Reset Password





1.5.5 Change password





1.6 Tecnologie, framework e Piattaforme adottate

1.6.1 Front-end

HTML, CSS, JAVASCRIPT

Per il front end dell'applicazione, sono stati utilizzati i linguaggi HTML, CSS e JavaScript per definire l'aspetto e il comportamento dell'interfaccia utente. Questi linguaggi sono stati scelti per la loro ampia diffusione e per la loro compatibilità con tutti i principali browser web.

1.6.2 Back end

THYMELEAF

È stato utilizzato Thymeleaf, un motore di template XML/HTML per applicazioni web Java che consente di creare template visuali per le pagine web. Thymeleaf è altamente configurabile e supporta molte funzionalità avanzate, tra cui la gestione dei layout, la

validazione dei modelli, l'internazionalizzazione, l'elaborazione condizionale, la manipolazione degli attributi HTML e la gestione degli eventi JavaScript. Inoltre, Thymeleaf è altamente integrato con Spring Framework, uno dei framework di sviluppo web più popolari per Java, il che lo rende una scelta popolare per lo sviluppo di applicazioni web in ambiente Java.

SPRING BOOT

Per il back end dell'applicazione, è stato utilizzato il framework Spring Boot per lo sviluppo del server e la gestione delle richieste HTTP. Spring Boot semplifica lo sviluppo di applicazioni web complesse e scalabili, fornendo un'ampia gamma di funzionalità e librerie integrate, nel nostro caso per garantire la sicurezza delle informazioni gestite dall'applicazione. Spring Boot è stato scelto per la sua versatilità e compatibilità con molte altre librerie e framework Java, semplificando lo sviluppo di applicazioni web robuste e scalabili.

1.6.3 Database

MYSQL

Inoltre, per la gestione del database dell'applicazione, è stato utilizzato il linguaggio MySQL e, per facilitare la distribuzione e la gestione del database, è stato creato un container Docker appositamente configurato per ospitare il server MySQL. Docker è una piattaforma che consente di creare, distribuire e gestire container di applicazioni con un'ottimizzazione delle risorse e un isolamento dei processi, consentendo di eseguire il database in un ambiente controllato e isolato dal sistema operativo sottostante.

HIBERNATE

È stato utilizzato Hibernate per semplificare la gestione dei dati nel database e offrire un'interfaccia Object-Relational Mapping (ORM) per le entità JPA (Java Persistence API) dell'applicazione. Nel file di configurazione dell'applicazione è stata impostata la proprietà "spring.jpa.hibernate.ddl-auto" su "update", che consente a Hibernate di generare automaticamente il DDL (Data Definition Language) necessario per creare o aggiornare le tabelle del database in base alle entità JPA definite nell'applicazione. Tuttavia, è importante prestare attenzione alla sicurezza dell'applicazione, poiché l'utilizzo di questa impostazione può portare a problemi di sicurezza, come ad esempio l'iniezione di codice SQL da parte di utenti malintenzionati. Si consiglia quindi di utilizzare questa impostazione solo durante lo sviluppo dell'applicazione, e di disabilitarla in ambiente di produzione, utilizzando invece un approccio controllato per la migrazione dei dati.

1.6.4 Deployment

DOCKER

La scelta di utilizzare Docker consente di semplificare la configurazione e la distribuzione dell'applicazione, oltre a garantire una maggiore sicurezza e controllo sull'ambiente di esecuzione del database. Inoltre, la creazione di un container appositamente configurato per ospitare il server MySQL consente di minimizzare le dipendenze del database da altri componenti del sistema e di garantire una maggiore portabilità dell'applicazione, facilitando la migrazione tra differenti ambienti di sviluppo o di produzione.

Inizialmente, l'intero processo era diviso in una fase in cui veniva avviato docker e in un'altra in cui era necessario avviare Spring manualmente, con altri comandi tramite IDE. Successivamente, invece, si è scelto di "dockerizzare" l'applicazione, rendendo possibile l'avvio dell'applicazione e l'esecuzione di essa tramite il semplice comando "docker-compose up".

1.6.5 Gestione delle dipendenze e del building

MAVEN

Per la gestione delle dipendenze del progetto e l'automazione del processo di build, è stato utilizzato Maven. Maven è un potente strumento di gestione dei progetti Java, che semplifica la configurazione e la gestione delle dipendenze del progetto e automatizza l'intero processo di build, dalla compilazione del codice sorgente fino alla generazione dell'artefatto finale. Grazie a Maven, la gestione delle dipendenze del progetto è stata semplificata e automatizzata, garantendo la coerenza e l'affidabilità dell'intero processo di sviluppo. Inoltre, Maven ha permesso di integrare facilmente il progetto con altri strumenti di sviluppo, come ad esempio IDE e strumenti di Continuous Integration, semplificando ulteriormente il processo di sviluppo e distribuzione dell'applicazione.

1.6.6 Mailing

SERVIZIO EMAIL

Il software utilizza un servizio di posta per inviare e-mail agli utenti dell'applicazione. La configurazione del servizio di posta è definita nel file di configurazione

dell'applicazione, attraverso un insieme di proprietà che includono l'host del server di posta, la porta utilizzata per la connessione, il nome utente e la password per l'autenticazione, e altre opzioni di configurazione. La corretta configurazione del servizio di posta è importante per garantire la corretta gestione delle comunicazioni tra l'applicazione e gli utenti.

1.7 Deployment

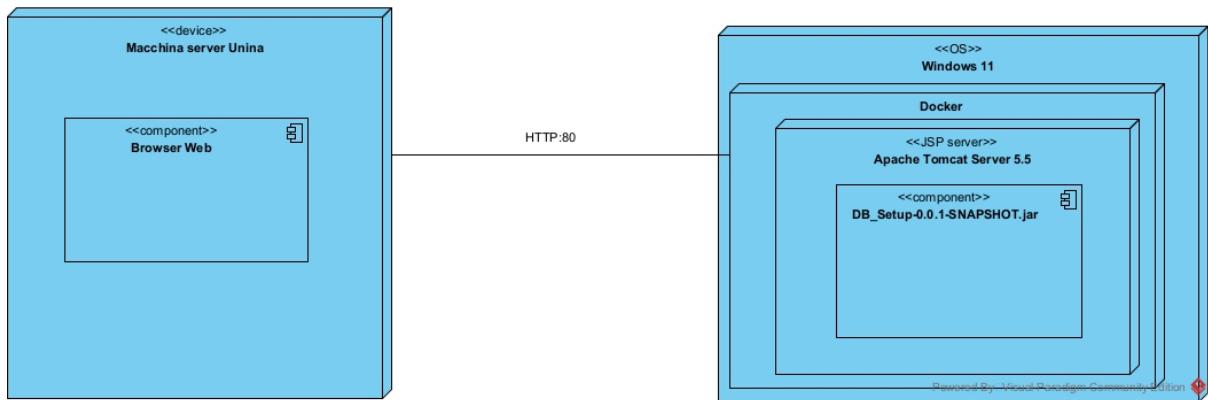
Il deployment, o rilascio del software, è una fase cruciale del processo di sviluppo del software in cui il prodotto viene distribuito e reso disponibile per l'uso da parte degli utenti finali. Il deployment richiede una pianificazione accurata e una procedura ben definita per garantire che il software sia distribuito in modo sicuro, affidabile e senza interruzioni per gli utenti.

Nella seguente sezione, descriveremo la procedura di deployment utilizzata per distribuire il nostro software:

1. Aprire **Docker Desktop**.
2. Aprire un terminale da amministratore e posizionarsi sul percorso dove è contenuto il progetto
 - - **docker-compose up**: viene utilizzato per avviare i servizi definiti in un file di configurazione "docker-compose.yml". Viene creata l'immagine del container e viene eseguito il running.
3. Per effettuare le richieste, aprire da browser le pagine tramite i path in locale (<http://localhost:8080/register>, <http://localhost:8080/login>, ecc.).
4. Per verificare la correttezza del popolamento delle tabelle del database, aprire un terminale ed eseguire i seguenti comandi:
 - - **docker exec -it g1-t2t3-app-1 bash**: viene utilizzato per entrare all'interno di un container Docker in esecuzione e avviare una shell interattiva al suo interno.
 - - **mysql -u root -p STUDENTSREPO**: viene utilizzato per accedere all'interfaccia della riga di comando di MySQL e connettersi al database "**STUDENTSREPO**" utilizzando l'utente "**root**" e richiedendo la password "**password**".
5. Utilizzare i comandi SQL per la gestione delle tabelle (**SELECT, DROP**, ecc.).

1.7.1 Deployment diagram

Il diagramma di deployment è una rappresentazione visiva ad alto livello che illustra la distribuzione fisica dei componenti del sistema software su hardware o ambienti di esecuzione, senza entrare nei dettagli interni del funzionamento dei singoli componenti.



Si evidenzia come un dispositivo possa connettersi al sistema tramite la porta 80 e come il sistema sia stato deployato sulla medesima macchina creando due container: uno per quanto riguarda l'applicazione l'altro per quanto riguarda il DB.

1.8 Criticità emerse

Tra le varie criticità emerse una è stata già segnalata nel context diagram in quanto non conteneva il servizio di logout. In particolare l'applicazione non mette a disposizione un bottone per effettuare il logout ma bisogna immettere nel browser il link: <http://localhost/logout>.

Il servizio di mailing non è stato implementato, rendendo di fatto indisponibili i servizi di changePassword e resetPassword inoltre non vi è alcuna mail di corretta registrazione, con relativo ID, mandata ad uno studente che si registra.

Dopo aver effettuato la registrazione il sistema reindirizza ad una pagina bianca con il solo messaggio di corretta registrazione. L'utente dovrà successivamente effettuare il login per entrare nel sistema invece di accedervi automaticamente dopo aver effettuato la registrazione.

1.9 Presentazione nuove funzionalità

1.9.1 Requisito R9

Si vuole modificare l'applicazione in modo tale che dopo aver effettuato il login l'utente sia indirizzato ad una nuova pagina nella quale potrà scegliere tra diverse opzioni:

1. accedere allo storico delle partite
2. iniziare una nuova partita
3. visualizzare la classifica

In particolare nello storico partite l'utente potrà visualizzare tutte le partite giocate e ogni partita sarà accompagnata da diverse informazioni: data, durata, classe testata, robot sfidato, vincitore (utente o robot), LOC coverage raggiunta dal giocatore e dal robot.

L'utente avrà, inoltre, la possibilità di iniziare una nuova partita contro un singolo robot, già presente come funzionalità del sistema, oppure una nuova partita in cui sfida tutti i robot.

Il giocatore, infine, avrà la possibilità di visionare la classifica in cui sono presenti tutti i giocatori che hanno giocato con il sistema nella quale verrà riportata il numero di partite giocate e di quelle vinte da ogni giocatore.

2. Metodologia di lavoro

2.1 SCRUM

Scrum è un framework di gestione dei progetti Agile che assiste i team nell'organizzazione e gestione del lavoro mediante l'adozione di valori, principi e pratiche specifiche. Analogamente a una squadra di rugby (da cui deriva il termine "Scrum", riferendosi alla mischia), che si allena in vista di un importante evento sportivo, il framework Scrum incentiva i team a imparare attraverso l'esperienza, a

organizzarsi autonomamente durante la risoluzione di un problema e a riflettere sui risultati ottenuti e sugli insuccessi per perseguire un miglioramento continuo.

Principi Fondamentali di Scrum:

- Iterazione e Incremento:

Scrum suddivide il lavoro in intervalli di tempo definiti chiamati "Sprint", di solito di durata fissa tra una settimana e un mese. Ogni Sprint produce un incremento del prodotto, un'aggiunta di valore potenzialmente consegnabile.

- Ruoli Definiti:

Scrum definisce chiaramente i ruoli all'interno del team:

- Il Product Owner, responsabile della definizione delle caratteristiche e della priorizzazione.
- Lo Scrum Master, che facilita il processo Scrum e rimuove gli ostacoli.
- Il Team di Sviluppo, composto da professionisti auto-organizzati che consegnano il lavoro.

- Artefatti chiave

- Il Product Backlog è un elenco prioritizzato di tutte le richieste di lavoro, di solito è formato dalle user story prodotte dal team di sviluppo in relazione ai requisiti specificati dal Product Owner.
- Lo Sprint Backlog contiene gli elementi selezionati per essere completati durante lo Sprint.
- L'Incremento è il risultato tangibile alla fine di ogni Sprint.
- Il Project Burndown Chart è uno strumento visuale utilizzato nel contesto di Scrum e altri approcci agili per monitorare e visualizzare il progresso del lavoro durante un progetto. Questo grafico mostra la quantità di lavoro rimanente rispetto al tempo previsto per completare il progetto. L'asse orizzontale rappresenta il tempo, mentre l'asse verticale rappresenta il lavoro rimanente.

Eventi Scrum:

- Sprint Planning:

Alla fine di ogni Sprint, il team e il Product Owner si incontrano per pianificare il lavoro da svolgere nel prossimo Sprint, andando a selezionare dal Product Backlog le user stories che dovranno essere completate durante questa iterazione.

- Daily Scrum:

Riunioni giornaliere brevi (circa 15 minuti) in cui il team si coordina, discute i progressi e identifica eventuali ostacoli.

- **Sprint Review:**

Alla fine di ogni Sprint, il Team di Sviluppo presenta il lavoro completato agli stakeholder per ottenere feedback.

- **Sprint Retrospective:**

Un'opportunità alla fine di ogni Sprint per riflettere sulle performance e identificare miglioramenti.

Valori Scrum:

- **Comprensione:** Tutti i membri del team devono condividere una comprensione comune degli obiettivi e delle priorità del progetto.
- **Autonomia:** Il Team di Sviluppo è auto-organizzato e prende le decisioni necessarie per raggiungere gli obiettivi dello Sprint.
- **Collaborazione:** La collaborazione tra tutti i membri del team, inclusi il Product Owner e lo Scrum Master, è essenziale per il successo.
- **Adattamento:** Scrum promuove un ambiente in cui il team può adattarsi rapidamente ai cambiamenti nelle esigenze del progetto.

2.2 Applicazione del framework Agile SCRUM

Per quanto riguarda il nostro progetto, abbiamo deciso di utilizzare il framework SCRUM per migliorare lo sviluppo e l'organizzazione del team. Nel nostro caso, trattandosi di un esame e non di un vero e proprio incarico, ogni singolo membro del team ha interpretato i ruoli di Product Owner, per quanto riguarda la prioritizzazione delle funzionalità da implementare, Scrum Master, per quanto riguarda la direzione del team di sviluppo e la creazione dei vari artefatti SCRUM che abbiamo utilizzato, e team di sviluppo per quanto riguarda l'implementazione vera e propria delle funzionalità richieste.

2.3 Artefatti e pratiche SCRUM utilizzate

Di seguito sono elencati gli artefatti creati e le pratiche utilizzate durante lo sviluppo delle funzionalità da implementare.

2.3.1 Product Backlog

Il "**Product Backlog**" è uno degli artefatti chiave nel framework Scrum e nei metodi agili in generale. Si tratta di un elenco dinamico e prioritizzato di tutte le caratteristiche, funzionalità, miglioramenti e correzioni di bug desiderati per un prodotto. Il Product

Backlog rappresenta la lista completa delle attività che il team di sviluppo deve affrontare per migliorare il prodotto nel corso del tempo.

Caratteristiche principali del Product Backlog:

Prioritizzazione:

- Gli elementi nel Product Backlog sono ordinati in base alla loro priorità, con gli elementi più importanti o urgenti posizionati in cima alla lista. La prioritizzazione è responsabilità del Product Owner.

Flessibilità:

- Il Product Backlog è dinamico e può essere adattato e aggiornato in qualsiasi momento. Nuove richieste, feedback degli utenti o cambiamenti nelle esigenze del mercato possono influenzare la sua composizione.

Elementi Vari:

- Gli elementi del Product Backlog possono assumere diverse forme, inclusi nuovi sviluppi, correzioni di bug, attività di manutenzione, miglioramenti delle prestazioni e richieste di nuove funzionalità.

Dettagli Evolutivi:

- Gli elementi in cima al Product Backlog tendono ad essere più dettagliati, mentre quelli in fondo possono essere meno specifici. Man mano che ci si avvicina all'implementazione, gli elementi vengono affinati e dettagliati.

Responsabilità del Product Owner:

- Il Product Owner è il responsabile della gestione del Product Backlog. Deve garantire che gli elementi siano chiari, ben compresi e che rappresentino il massimo valore possibile per gli stakeholder.

Comunicazione con lo Sprint Planning:

- Durante lo Sprint Planning, il team di sviluppo seleziona gli elementi più prioritari dal Product Backlog e li porta nello Sprint Backlog per essere lavorati durante lo Sprint corrente.

Riflessione della Visione del Prodotto:

- Il Product Backlog riflette la visione del prodotto e le esigenze del cliente. Gli elementi più prioritari rispecchiano le caratteristiche chiave o gli obiettivi principali del prodotto.

Strumento per le Priorità a Lungo Termine:

- Il Product Backlog è uno strumento che consente al team di gestire le priorità a lungo termine, fornendo una visione continua delle attività future che devono essere svolte.

Il **Product Backlog** è uno strumento fondamentale per mantenere l'orientamento al valore del prodotto e per garantire che il team stia costantemente lavorando sulle attività più importanti per raggiungere gli obiettivi complessivi del progetto.

2.3.2 User story e criteri di accettazione

Una **User Story** è una tecnica utilizzata nei metodi agili, in particolare nel framework Scrum, per definire i requisiti del software dal punto di vista dell'utente finale. Una User Story è una breve descrizione di una funzionalità del prodotto, scritta dal punto di vista dell'utente, e fornisce informazioni essenziali su chi, cosa e perché. Le User Stories sono spesso utilizzate per guidare lo sviluppo iterativo e incrementale all'interno di uno Sprint. Segue una struttura semplice:

Come [tipo di utente], voglio [un'azione], in modo che [beneficio o obiettivo desiderato].

- Come: Rappresenta il tipo di utente coinvolto.
- Voglio: Indica l'azione o la funzionalità desiderata.
- In modo che: Fornisce il contesto o l'obiettivo che giustifica la richiesta.

I criteri di accettazione per le User Stories rappresentano condizioni specifiche che devono essere soddisfatte affinché una particolare User Story possa essere considerata completata. Questi criteri forniscono una base oggettiva per valutare se la funzionalità implementata soddisfa effettivamente le aspettative del cliente e del team di sviluppo. I criteri di accettazione sono spesso definiti durante la fase di pianificazione di uno Sprint o prima dell'inizio dell'implementazione di una specifica User Story.

Ecco alcuni aspetti chiave dei criteri di accettazione per le User Stories:

Chiarezza e Specificità:

- I criteri devono essere chiari e specifici, evitando ambiguità. Devono essere comprensibili a tutti i membri del team e agli stakeholder coinvolti.

Centrati sull'Utente:

- I criteri di accettazione dovrebbero riflettere gli aspetti più importanti dal punto di vista dell'utente finale. Cosa ci si aspetta che l'utente possa fare o ottenere con questa funzionalità?

Misurabili e Verificabili:

- I criteri devono essere formulati in modo da consentire una verifica oggettiva. Ad esempio, "il sistema deve rispondere in meno di 2 secondi" è misurabile, mentre "il sistema deve rispondere rapidamente" è vago.

Rilevanti e Prioritari:

- I criteri dovrebbero riflettere i requisiti più importanti e prioritari. Il team e il Product Owner devono concentrarsi sui criteri che contribuiscono maggiormente al valore del prodotto.

Di solito vengono rappresentati nella forma:

Dato che [scenario] , quando [effettuo un'azione], allora [succede questo]

- Dato che: rappresenta lo scenario in cui l'utente sta per eseguire un'azione
- Quando: rappresenta l'azione effettuata dall'utente
- Allora: rappresenta la risposta del sistema all'azione dell'utente specificata nella user story

2.3.3 Definition of Done

La DoD (Definition of Done) va di pari passo con le user stories e rappresenta un concetto fondamentale nell'ambito del framework Scrum e degli approcci agili. La Definition of Done rappresenta un insieme di criteri e standard che devono essere soddisfatti affinché un elemento di lavoro, come una User Story o un Task, possa essere considerato completato e potenzialmente consegnato al cliente o utilizzato in un ambiente di produzione. In altre parole, stabilisce quando un lavoro è considerato finito e soddisfa gli standard di qualità del team. La Definition of Done è cruciale per garantire che ci sia un chiaro consenso all'interno del team su cosa significhi "finito". Questo contribuisce a evitare ambiguità, migliorare la trasparenza, e assicurare che tutti i membri del team comprendano e rispettino gli standard di qualità stabiliti. La Definition of Done è spesso mantenuta e aggiornata di Sprint in Sprint, riflettendo l'apprendimento continuo e l'evoluzione del team. La Definition of Done varia da team a team e può includere una combinazione di criteri tecnici, di qualità, di test e di approvazione. Ecco alcuni esempi di elementi che potrebbero essere inclusi nella Definition of Done:

Verifica del Codice:

- Il codice deve essere stato sottoposto a revisione tra pari (peer review).
- Il codice deve rispettare gli standard di stile del team.
- Devono essere risolti tutti gli avvisi e gli errori del compilatore.

Test Unitari:

- Tutti i test unitari devono essere scritti e superati.
- L'implementazione deve essere conforme ai requisiti specificati nella User Story.

Test di Integrazione e Funzionali:

- Tutti i test di integrazione e funzionali devono essere eseguiti con successo.
- Tutte le funzionalità della User Story devono essere testate e approvate.

Documentazione:

- La documentazione del codice deve essere aggiornata.
- La documentazione degli utenti deve essere creata o aggiornata, se applicabile.

Revisone del Product Owner:

- La User Story deve essere dimostrata al Product Owner e approvata.
- Eventuali feedback del Product Owner devono essere affrontati.

Accettazione da Parte degli Stakeholder:

- Se necessario, la User Story deve essere accettata dagli stakeholder o dagli utenti finali.

Nel nostro caso abbiamo adottato una Definition of Done che arriva fino al testing locale del progetto, di seguito sono indicati gli elementi che la compongono:

- Il codice per implementare le nuove funzionalità è stato prodotto
- Il codice rispetta le assunzione delle user stories
- Il progetto viene buildato senza errori
- La nuova feature è stata testata rispetto ai criteri di accettazione
- La documentazione è stata aggiornata
- Codice revisionato da più sviluppatori (peer review)

2.3.4 Planning Poker

Il Planning Poker è una tecnica utilizzata per stimare il lavoro necessario per completare gli elementi del Product Backlog. Coinvolge il Team di Sviluppo, il Product Owner e, talvolta, lo Scrum Master. Ecco come funziona:

- **Selezione degli elementi:** Il Product Owner presenta gli elementi del Product Backlog da discutere.
- **Stima:** Ogni membro del Team di Sviluppo riceve una serie di carte con valori numerici (ad esempio, Fibonacci: 1, 2, 3, 5, 8, 13).
- **Votazione Anonima:** Per ogni elemento, i membri del Team di Sviluppo selezionano una carta corrispondente alla loro stima in modo anonimo.
- **Discussione e rivotazione:** Se ci sono stime molto diverse, il team discute i motivi e procede con una nuova votazione finché non si raggiunge un consenso.
- **Registro delle stime:** Le stime finali vengono registrate e utilizzate per pianificare la quantità di lavoro da includere nello Sprint.

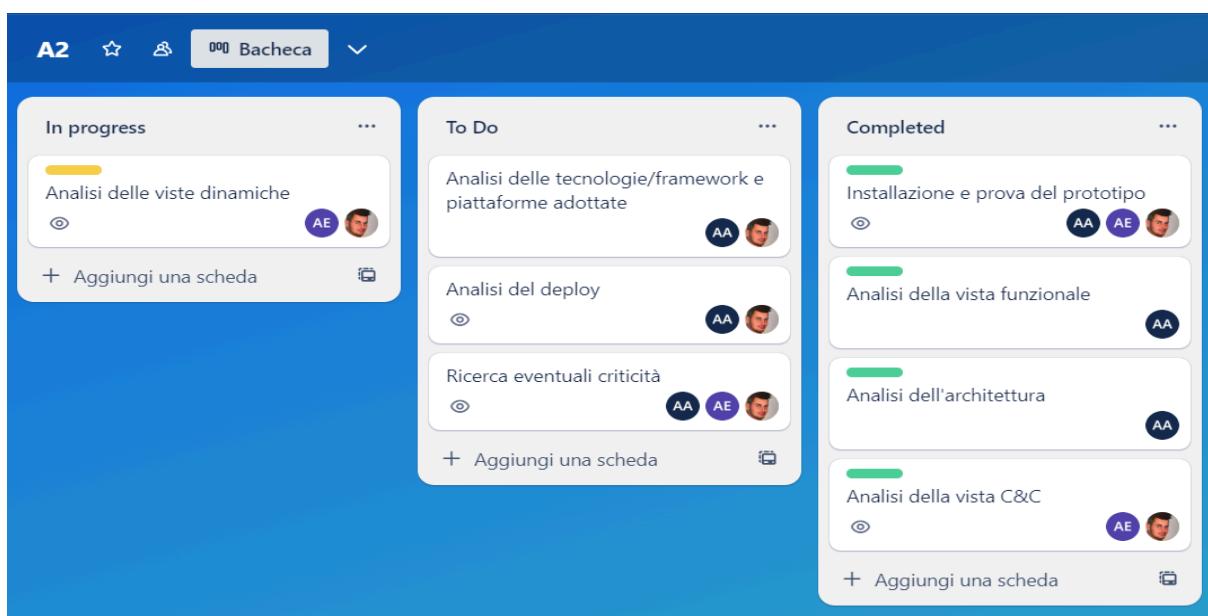
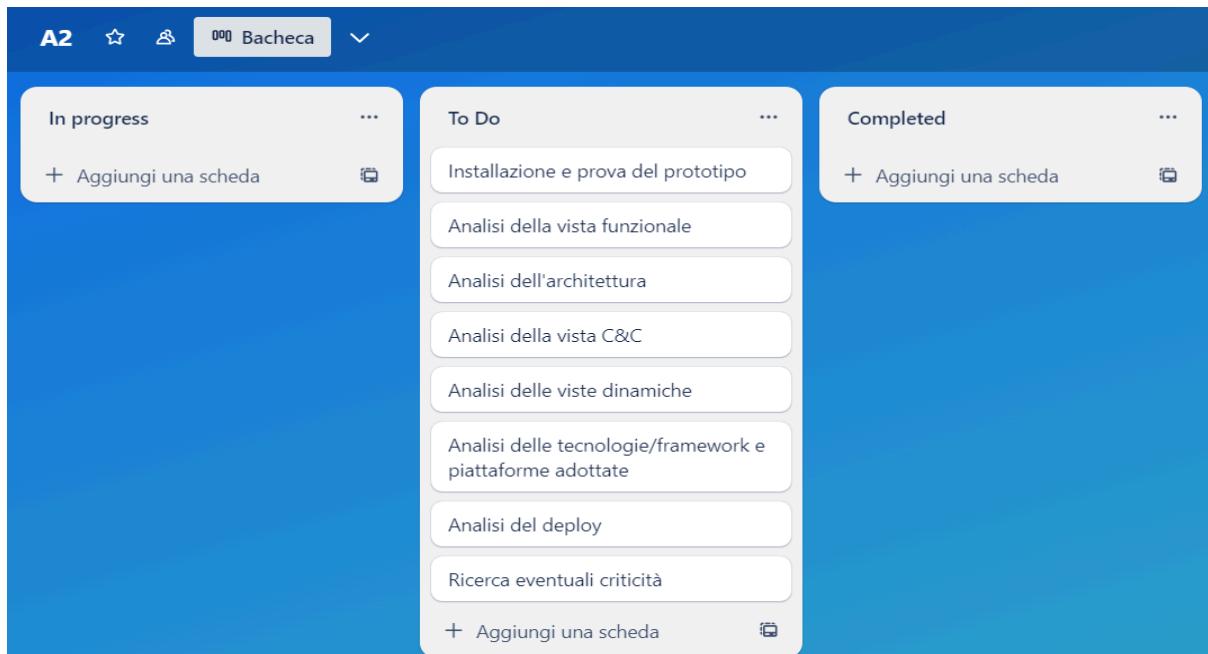
2.4 Prima iterazione

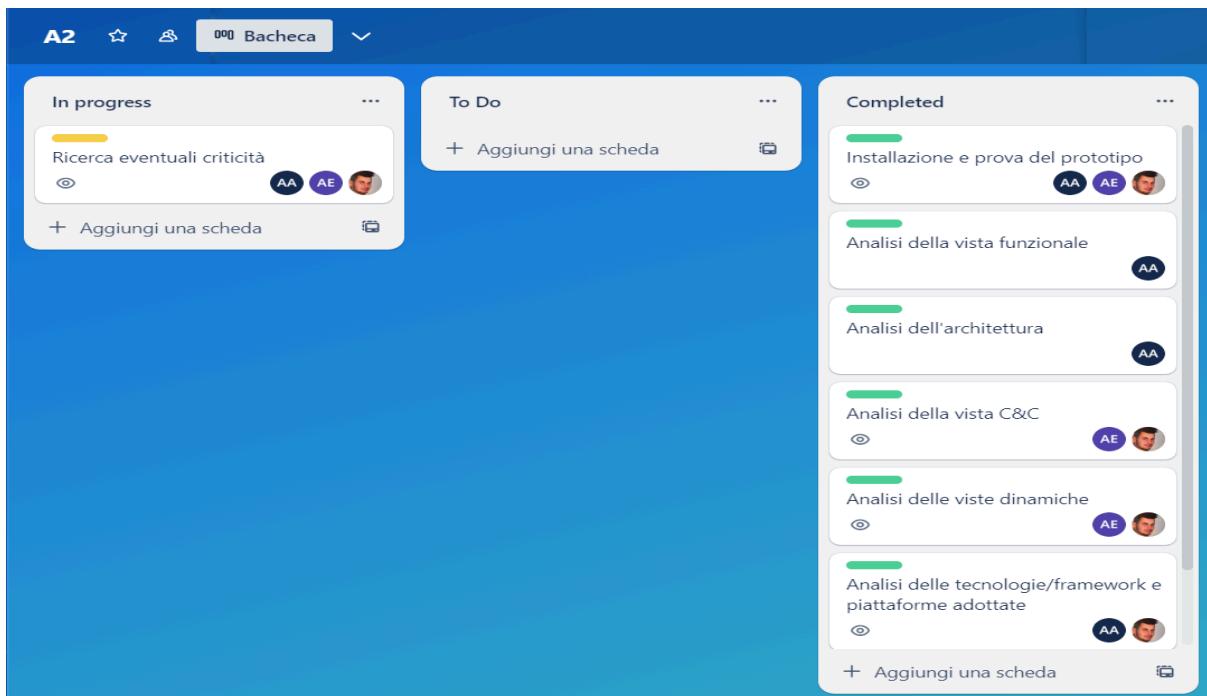
I task relativi alla prima iterazione riguardano l'analisi del progetto esistente per comprendere la struttura del progetto e le problematiche presenti.

Visto il task proposto, abbiamo deciso di non utilizzare gli artefatti SCRUM riguardanti l'implementazione di nuove funzionalità, come ad esempio il Product Backlog, ma ci siamo serviti esclusivamente del Daily Scrum per discutere del lavoro svolto e pianificare i prossimi passi riguardanti l'analisi del progetto e di una bacheca condivisa dal team su Trello per elencare i diversi task da svolgere nel corso dello sprint.

<https://trello.com/b/N0szbfVw/a2>

Di seguito viene presentata la suddetta bacheca e la sua evoluzione nel corso dell'iterazione.





Alla fine della prima iterazione è stata prodotta la documentazione relativa al progetto e sono stati individuati diversi bug presenti nel progetto che verranno aggiunti poi al carico di lavoro della prossima iterazione.

2.5 Seconda iterazione

La seconda iterazione aveva come obiettivo quello di implementare una nuova pagina in cui l'utente potesse scegliere tra visualizzare lo storico delle partite giocate, visualizzare la classifica totale dei giocatori e iniziare una nuova partita accedendo alla pagina principale.

2.5.1 Sprint planning

Durante la prima fase dell'iterazione, ovvero quella di **Sprint Planning**, abbiamo svolto i diversi ruoli SCRUM in modo da creare il **Product Backlog** formato dalle user stories, alle quali abbiamo assegnato una priorità e, attraverso la fase di **Planning Poker** abbiamo assegnato ad esse un valore in story point, che indica l'effort necessario per essere completate dal team di sviluppo. Di seguito viene presentato il product backlog prodotto per questa iterazione.

User Stories

US1
As an authenticated user, i want to visualize my match history, in order to check the statistics of my matches.

US2
As an authenticated user, i want to start a new base game, in order to challenge a single robot on a class

US3
As an authenticated user, i want to start a 1 vs All game, in order to challenge all the available robots on a class

US4
As an authenticated user, i want to visualize the total ranking of all players, in order to check the results of all players

Acceptance Criteria

US1
Given that i've logged in and accessed the options page, when i click on the 'match history' button, then i can visualize all the statistics of my previous matches

US2
Given that i've logged in and accessed the options page, when i click on the 'new base game' button, then i can select the class, robot and level to start a new base game

US3
Given that i've logged in and accessed the options page, when i click on the '1 vs All' button, then i can select the class and the available robots to start a new 1 vs All game

US4
Given that i've logged in and accessed the options page, when i click on the 'ranking' button, then i can visualize the total ranking of all players.

Bugs

- 1.0: Mail service keeps throwing an error due to his properties setting
- 2.0: Logout button missing, the disconnection is possible only accessing the URL with the logout API
- 3.0: Redirection to some HTML pages needs to be fixed.

Definition of Done for user stories

- Produced code for presumed functionalities
- Assumptions of User Story met
- Project builds without errors
- Feature is tested against acceptance criteria
- Documentation updated
- Peer Code Review performed

Product Backlog		
ID	STORY POINTS	PRIORITY
User story 1 : Storico Match	5	1
User story 2: Nuova partita base	5	1
User story 3: Nuova partita '1 vs All'	5	1
User story 4: Ranking	5	1
Bug 1.0: Mail service	1	2
Bug 2.0: Logout button	3	2
Bug 3.0: Redirect to some html pages	1	2

Total story points: 25

Team velocity estimation: 30

Di seguito viene riportata l'evoluzione della bacheca su Trello nel corso di questo sprint.

A2 ⚡ Visibile allo Spazio di lavoro Bacheca

Product Backlog

- US1: As an authenticated user, i want to visualize my match history, in order to check the statistics of my matches.
- US2: As an authenticated user, i want to start a new base game, in order to challenge a single robot on a class
- US3: As an authenticated user, i want to start a 1 vs All game, in order to challenge all the available robots on a class
- US4: As an authenticated user, i want to visualize the total ranking of all players, in order to check the results of all players
- Bug 1.0: Errore nella configurazione delle properties del mail service
- Bug 2.0: Aggiunta del bottone di logout
- Bug 3.0: Fix dei redirect alle varie pagine HTML.

Features for user stories (To Do)

- Task 0.1: Creare pagina HTML 'options'
- Task 0.2: Aggiungere la pagina 'options' nel file di configurazione del webserver nginx
- Task 0.3: Creare file css per settare lo stile della pagina 'options'
- Task 0.4: Creare script in JS per gestire le funzionalità della pagina 'options'
- Task 0.5: Modifica della pagina a cui si viene reindirizzati quando si effettua il login da 'main' a 'options'

User Story 1-2 (To Do)

- Task 1.1: Creare button 'match history' nella pagina HTML 'options'
- Task 1.2: Aggiunta stile per il bottone 'match history' nel file css
- Task 1.3: Aggiunta evento sul bottone 'match history' nello script JS per visualizzare lo storico dei match giocati
- Task 2.1: Creare button 'new base game' nella pagina HTML 'options'
- Task 2.2: Aggiunta stile per il bottone 'new base game' nel file css
- Task 2.3: Aggiunta evento sul bottone 'new base game' nello script JS per iniziare una nuova partita base

User Story 3-4 (To Do)

- Task 3.1: Creare button 1 vs All' nella pagina HTML 'options'
- Task 3.2: Aggiunta stile per il bottone '1 vs All' nel file css
- Task 3.3: Aggiunta evento sul bottone '1 vs All' nello script JS per iniziare una nuova partita '1 vs All'
- Task 4.1: Creare button 'ranking' nella pagina HTML 'options'
- Task 4.2: Aggiunta stile per il bottone 'ranking' nel file css
- Task 4.3: Aggiunta evento sul bottone 'ranking' nello script JS per visualizzare la classifica totale dei giocatori

Bugs (To Do)

- Task Bug 1.1: Modificare il protocollo utilizzato in SSL nel file properties del mail server
- Task Bug 2.1: Aggiungere un bottone nella pagina html 'options' per permettere il logout all'utente
- Task Bug 2.2: Aggiungere stile per il bottone nel file css
- Task Bug 2.3: Aggiungere nel controller la logica per effettuare la disconnessione alla pressione del bottone 'logout'
- Task Bug 3.0: Aggiungere nel controller la logica per reintridare l'utente alla pagina di password_change dopo aver effettuato il reset della password

A2 ⚡ Visibile allo Spazio di lavoro Bacheca

Definition of Done

- User story 1: B/ 0/5
- User story 2: B/ 0/5
- User story 3: B/ 0/5
- User story 4: B/ 0/5

Features for user stories (In Progress)

- Task 0.1: Creare pagina HTML 'options'
- Task 0.2: Aggiungere la pagina 'options' nel file di configurazione del webserver nginx
- Task 0.3: Creare file css per settare lo stile della pagina 'options'
- Task 0.4: Creare script in JS per gestire le funzionalità della pagina 'options'
- Task 0.5: Modifica della pagina a cui si viene reindirizzati quando si effettua il login da 'main' a 'options'

User Story 1-2 (In Progress)

- Task 1.1: Creare button 'match history' nella pagina HTML 'options'
- Task 1.2: Aggiunta stile per il bottone 'match history' nel file css
- Task 1.3: Aggiunta evento sul bottone 'match history' nello script JS per visualizzare lo storico dei match giocati
- Task 2.1: Creare button 'new base game' nella pagina HTML 'options'
- Task 2.2: Aggiunta stile per il bottone 'new base game' nel file css
- Task 2.3: Aggiunta evento sul bottone 'new base game' nello script JS per iniziare una nuova partita base

User Story 3-4 (In Progress)

- Task 3.1: Creare button 1 vs All' nella pagina HTML 'options'
- Task 3.2: Aggiunta stile per il bottone '1 vs All' nel file css
- Task 3.3: Aggiunta evento sul bottone '1 vs All' nello script JS per iniziare una nuova partita '1 vs All'
- Task 4.1: Creare button 'ranking' nella pagina HTML 'options'
- Task 4.2: Aggiunta stile per il bottone 'ranking' nel file css
- Task 4.3: Aggiunta evento sul bottone 'ranking' nello script JS per visualizzare la classifica totale dei giocatori

Bugs (In Progress)

- Task Bug 1.1: Modificare il protocollo utilizzato in SSL nel file properties del mail server
- Task Bug 2.1: Aggiungere un bottone nella pagina html 'options' per permettere il logout all'utente
- Task Bug 2.2: Aggiungere stile per il bottone nel file css
- Task Bug 2.3: Aggiungere nel controller la logica per effettuare la disconnessione alla pressione del bottone 'logout'
- Task Bug 3.0: Aggiungere nel controller la logica per reintridare l'utente alla pagina di password_change dopo aver effettuato il reset della password

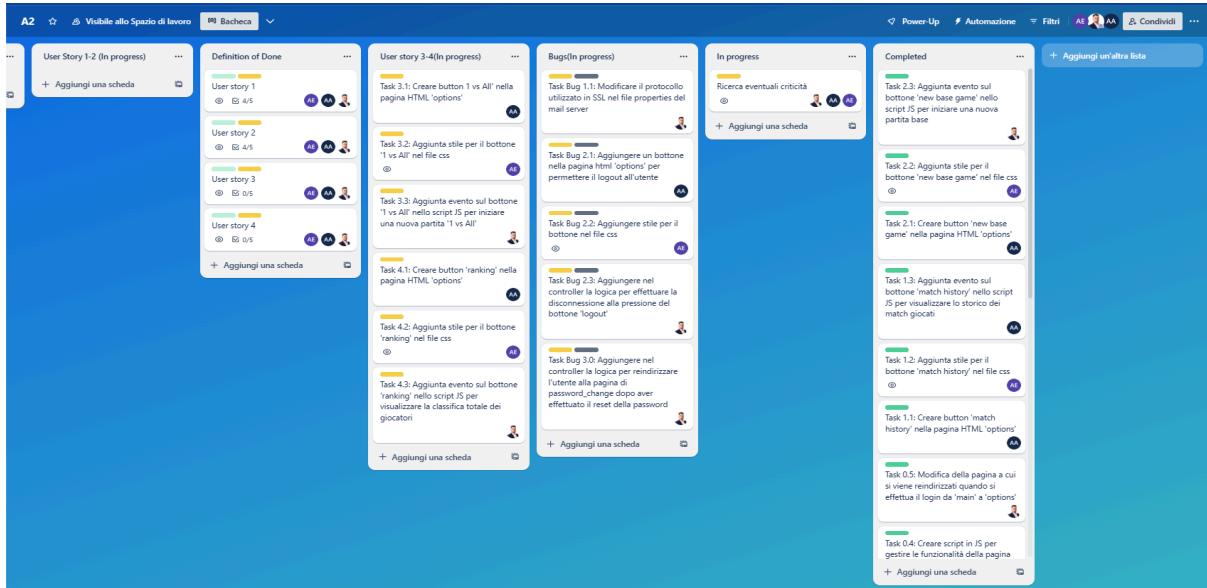
In progress

- Ricerca eventuali criticità

Completed

- Installazione e prova del prototipo
- Analisi della vista funzionale
- Analisi dell'architettura
- Analisi della vista C&C
- Analisi delle viste dinamiche
- Analisi delle tecnologie/framework e piattaforme adottate
- Analisi del deploy

Aspetto sul bottone nuovo base come nelle coloni in alto

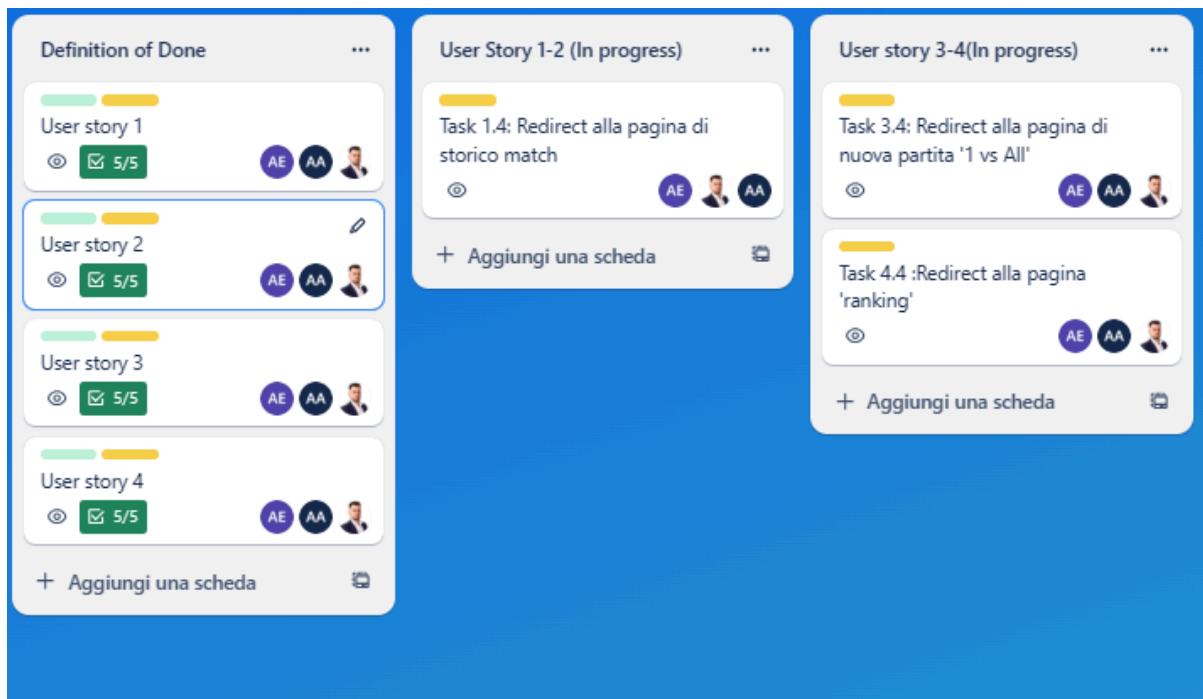


Legenda:

- **Verde** : completato
- **Giallo** : in corso
- **Arancione** : pending
- Nero : bug
- **Viola** : to do
- **Celeste** : user story

2.5.2 Sprint review

Nella fase di sprint review il team revisiona il lavoro svolto durante l'iterazione e calcola la velocità effettiva in termini di story points completati, la quale verrà usata come stima nella successiva iterazione, inoltre si rimettono in cima al product Backlog i task che non sono stati completati durante lo sprint.



- **Team Velocity estimation:** 30
- **Completed story points:** 22
- **Current team velocity:** 22
- **Completed tasks:** User story 2, Bug 1.0, 2.0 e 3.0
- **Pending tasks:** Redirect to 'match history', 'ranking', "1 vs All game" to complete user stories 1, 3, 4

2.5.3 Sprint retrospective

Durante la fase di sprint retrospective il team di sviluppo riflette su quello che è andato bene durante lo sprint e su quello che invece non è andato come pianificato. In questa fase il team cerca di trovare dei rimedi ai problemi che si sono verificati, riflettendo su cosa bisogna fare diversamente per migliorare le cose.

What went well? 😊	What could have been better? 😐	What will we do differently? 🚀
<p>Incontro giornaliero di 15 minuti su Teams per parlare dei progressi raggiunti e di come continuare lo sviluppo</p>	<p>Prestare attenzione a eventuali conflitti quando si effettua una push</p>	
<p>Avvisare il team su Whatsapp ogni volta che si effettua una push</p>	<p>Aggiornamento della documentazione in relazione alle modifiche al codice e non dopo aver implementato già la funzionalità</p>	<p>Utilizzo di Discord per comunicazioni e incontri giornalieri</p>

2.6 Terza iterazione

I task relativi alla terza iterazione riguardavano la creazione di nuove pagine per visualizzare lo storico partite e la classifica totale, la creazione dei mock-up relativi a queste interfacce e un esempio di ri-progettazione del database esistente in modo da facilitare la creazione delle chiamate API necessarie per ottenere i dati relativi a queste nuove funzionalità.

2.6.1 Sprint planning

Durante la fase iniziale di Sprint Planning, è stato rifinito prima di tutto il Product Backlog, quindi sono stati aggiunti i nuovi task relativi a questa iterazione, oltre a questi sono stati aggiunti a questa iterazione tutti i task non completati durante la scorsa e infine è stato aggiunto il bug relativo al database del T4. Di seguito viene riportato il Product Backlog aggiornato per questa iterazione.

PRODUCT BACKLOG

User Stories

US1 As an authenticated user, i want to visualize my match history, in order to check the statistics of my matches.	US2 As an authenticated user, i want to start a new game, in order to challenge myself against the robots	US3 As an authenticated user, i want to visualize the total ranking of all players, in order to check the results of all players
---	---	--

Acceptance Criteria

US1 Given that i've logged in and accessed the options page, when i click on the 'match history' button, then i can visualize all the statistics of my previous matches	US2 Given that i've logged in and accessed the options page, when i click on the 'new game' button, then i can select the class and the available robots to start a new game	US3 Given that i've logged in and accessed the options page, when i click on the 'ranking' button, then i can visualize the total ranking of all players.
---	--	---

Bugs

- T4_Database does not provide any information about which robots have played in a certain game.

Definition of Done for user stories

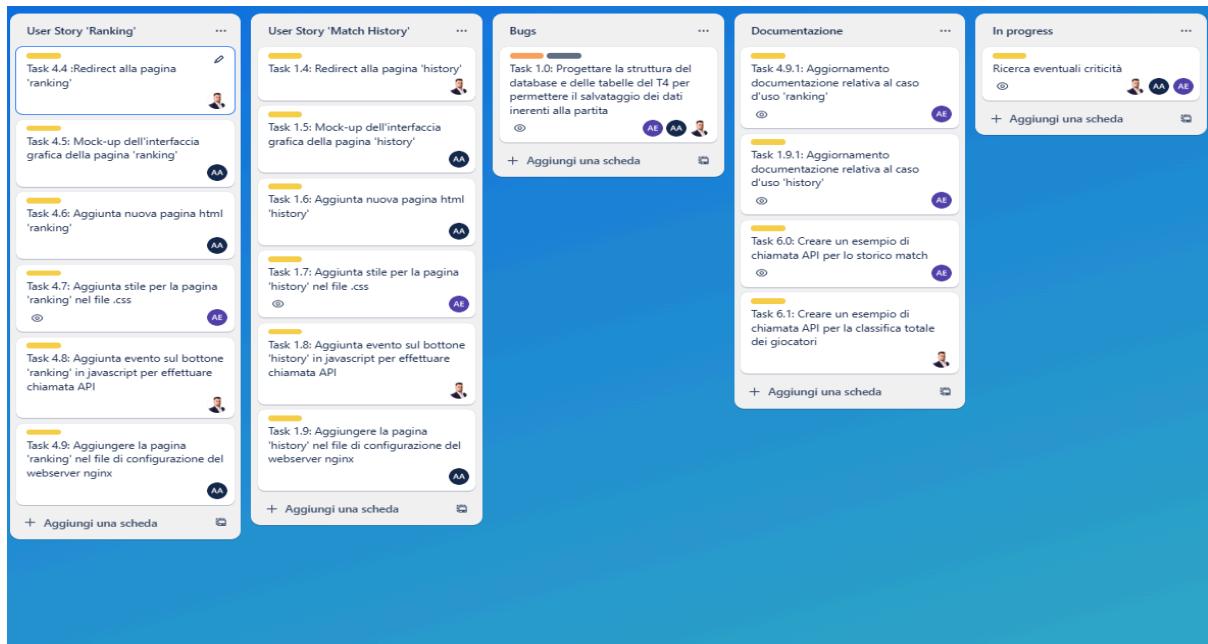
- Produced code for presumed functionalities
- Assumptions of User Story met
- Project builds without errors
- Feature is tested against acceptance criteria
- Documentation updated
- Peer Code Review performed

Product Backlog item	ID	Story Points	Tasks (TO DO)	Tasks completed from previous iteration
User story 'Match History'	1	5	Task 1.4: Redirect alla pagina 'history'	Task 1.1: Creare button 'match history' nella pagina HTML 'options'
User story 'Match History'	1	5	Task 1.5: Mock-up dell'interfaccia grafica della pagina 'history'	Task 1.2: Aggiunta stile per il bottone 'match history' nel file css
User story 'Match History'	1	5	Task 1.6: Aggiunta nuova pagina html 'history'	Task 1.3: Aggiunta evento sul bottone 'match history' nello script JS per visualizzare lo storico dei match giocati
User story 'Match History'	1	5	Task 1.7: Aggiunta stile per la pagina 'history' nel file .css	-
User story 'Match History'	1	5	Task 1.8: Aggiunta evento sul bottone 'history' in javascript per effettuare chiamata API	-
User story 'Match History'	1	5	Task 1.9: Aggiungere la pagina 'history' nel file di configurazione del webserver nginx	-
User story 'New game'	2	5	-	Task 2.1: Creare button 'new game' nella pagina HTML 'options'
User story 'New game'	2	5	-	Task 2.2: Aggiunta stile per il bottone 'new game' nel file css
User story 'New game'	2	5	-	Task 2.3: Aggiunta evento sul bottone 'new game' nello script JS per iniziare una nuova partita
User story 'Ranking'	3	5	Task 4.4 :Redirect alla pagina 'ranking'	Task 4.1: Creare button 'ranking' nella pagina HTML 'options'
User story 'Ranking'	3	5	Task 4.5: Mock-up dell'interfaccia grafica della pagina 'ranking'	Task 4.2: Aggiunta stile per il bottone 'ranking' nel file css
User story 'Ranking'	3	5	Task 4.6: Aggiunta nuova pagina html 'ranking'	Task 4.3: Aggiunta evento sul bottone 'ranking' nello script JS per visualizzare la classifica totale dei giocatori
User story 'Ranking'	3	5	Task 4.7: Aggiunta stile per la pagina 'ranking' nel file .css	-
User story 'Ranking'	3	5	Task 4.8: Aggiunta evento sul bottone 'ranking' in javascript per effettuare chiamata API	-
User story 'Ranking'	3	5	Task 4.9: Aggiungere la pagina 'ranking' nel file di configurazione del webserver nginx	-
Bug 'T4_Database'	4	3	Task 1.0: Progettare la struttura del database e delle tabelle del T4 per permettere il salvataggio dei dati inerenti alla partita	

Total story points: 13

Estimated team velocity from previous iteration: 22

Di seguito viene riportata la bacheca Trello aggiornata per la terza iterazione:



Durante questa iterazione per effettuare la procedura di peer code review ci siamo serviti dello strumento di pull request messo a disposizione sulla piattaforma Github. In questo modo risulta molto semplice osservare tutte le modifiche apportate al codice per gli sviluppatori che devono revisionare il lavoro effettuato da un altro membro del team.

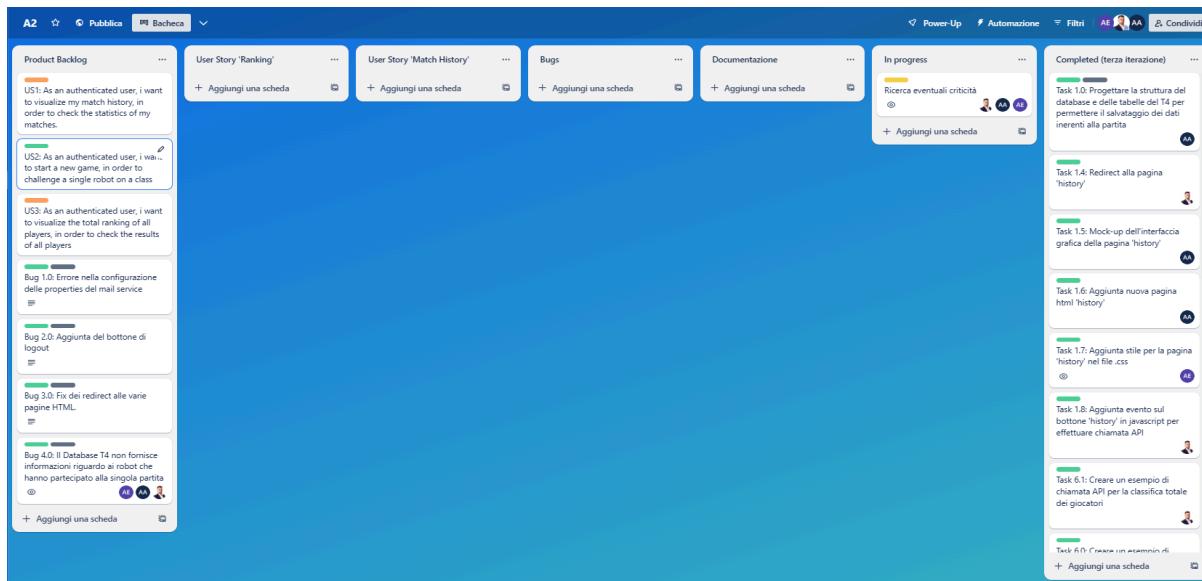
Fix service fast api #1

A screenshot of a GitHub pull request titled "fix--service-fastAPI". The pull request has 5 commits from "Alfoesp97" into the "master" branch. The diff view shows changes in the "history.css" file. The left sidebar shows the project structure with folders like T23-G1, src/main/resources, static/t23, js, templates, target, and T4-G18. The right pane displays the diff with red highlights for deleted lines and green highlights for added lines. The status bar at the top indicates "+162 -51" changes across 11 files.

Questo è un esempio di pull request effettuata su GitHub. È possibile notare in rosso le righe cancellate da quel file, in verde quelle aggiunte o modificate. Dopo la revisione del codice, il membro del team incaricato può fare clic sul pulsante '*Complete Merge*' per accettare le modifiche revisionate e caricarle direttamente sul branch principale (main).

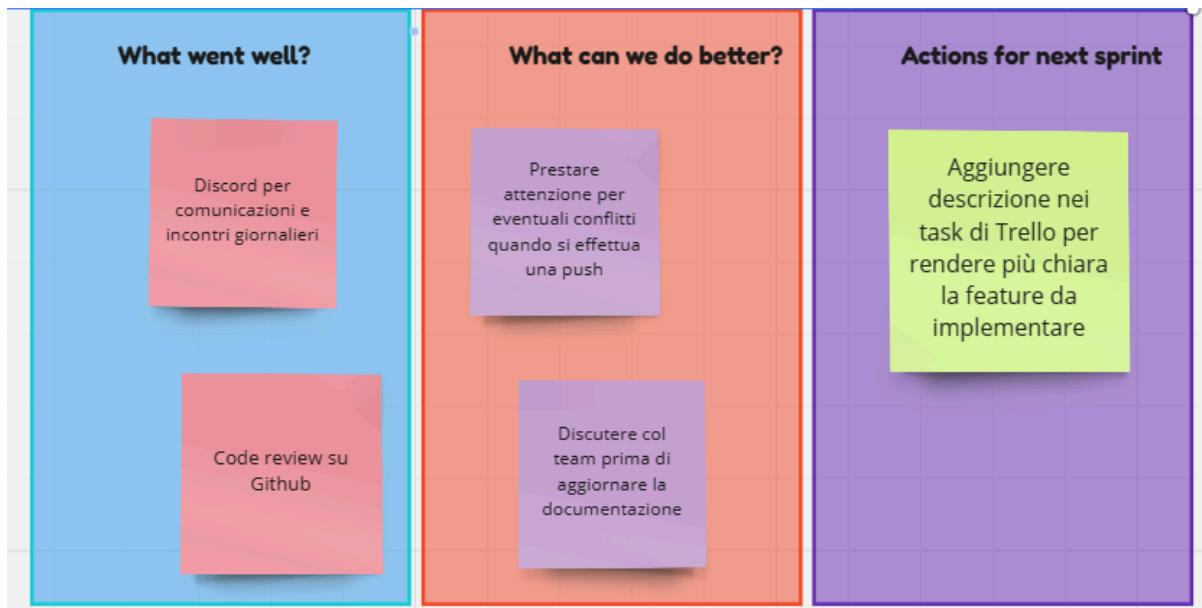
2.6.2 Sprint review

Nella fase di sprint review, il team rivede il lavoro svolto durante l'iterazione e calcola la velocità effettiva in termini di story points completati. Questa velocità viene utilizzata come stima per la successiva iterazione. Inoltre, vengono rimessi in cima al Product Backlog i task che non sono stati completati durante lo sprint.



2.6.3 Sprint retrospective

Durante la fase di sprint retrospective, il team di sviluppo riflette su ciò che è andato bene durante lo sprint e su ciò che invece non ha seguito la pianificazione. In questa fase, il team cerca di individuare rimedi ai problemi verificatisi, riflettendo su cosa può essere fatto in modo diverso per migliorare la situazione.



3. Requisito R9

Nel prossimo capitolo, esamineremo l'implementazione del requisito R9 attraverso nuove viste funzionali, analizzando il processo di sviluppo tramite il diagramma dei casi d'uso e esplorando i nuovi scenari emergenti. Approfondiremo l'architettura del sistema, concentrandoci sulle viste dinamiche e illustrando i diagrammi di attività e sequenza.

Successivamente, affronteremo l'approccio al testing, evidenziando le strategie adottate per garantire la corretta funzionalità del sistema. Presenteremo un mockup dell'interfaccia utente per anticipare la nuova esperienza derivante dal requisito R9.

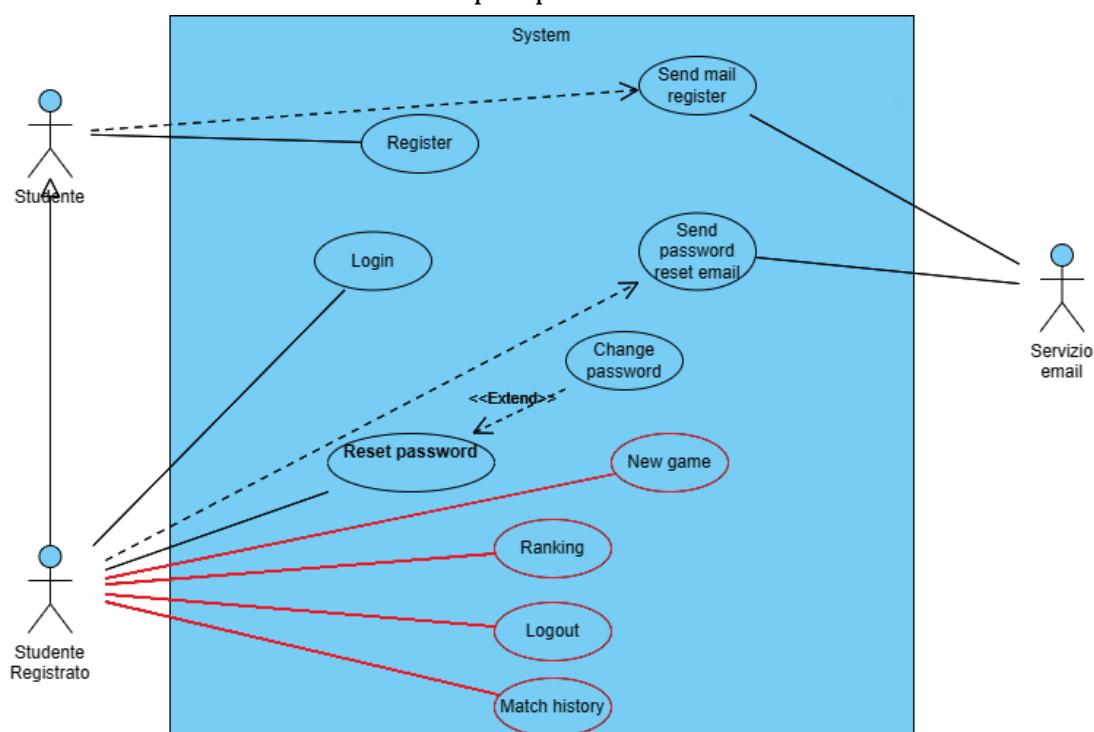
Infine, discuteremo la proposta di modifica del database T4, esplorando le nuove esigenze di struttura dati, e illustreremo un esempio di API, focalizzandoci sull'impatto di questa implementazione sulla comunicazione e sull'accesso ai dati del sistema.

3.1. Viste funzionali

Le **viste funzionali** si concentrano sugli aspetti "cosa fa" del sistema. Esse mirano a rappresentare le funzionalità offerte dal sistema senza entrare nei dettagli implementativi. Questo tipo di vista è utile per gli stakeholder non tecnici o per coloro che desiderano ottenere una panoramica ad alto livello delle capacità del sistema.

3.1.1 Diagramma dei casi d'uso

I casi d'uso rappresentano situazioni o scenari in cui un utente interagisce con un sistema per raggiungere un obiettivo specifico. Essi sono spesso utilizzati per catturare i requisiti funzionali del sistema da una prospettiva orientata all'utente.



Per svolgere il nostro task, sono stati introdotti i casi d'uso "New game", "Ranking", "Match History" e "Logout":

- **New Game (Nuova Partita):** Questo caso d'uso riguarda l'iniziazione di una nuova partita nel contesto della web application. Gli utenti potrebbero avviare questa funzionalità per iniziare una nuova sessione di gioco, avviare una sfida o selezionare nuove impostazioni di gioco.
- **Ranking (Classifica):** Il caso d'uso "Ranking" coinvolge l'accesso o la visualizzazione della classifica all'interno del sistema. Gli utenti potrebbero utilizzare questa funzionalità per monitorare la loro posizione relativa agli altri utenti, visualizzare i punteggi più alti o confrontare le prestazioni.
- **Match History (Storico delle Partite):** Il caso d'uso "Match History" coinvolge l'accesso e la visualizzazione dello storico delle partite precedenti. Gli utenti potrebbero utilizzare questa funzionalità per esaminare dettagli e risultati delle partite passate, analizzare le proprie prestazioni o valutare la progressione nel tempo.
- **Logout (Disconnessione):** Questo caso d'uso riguarda il processo di disconnessione o logout dall'applicazione. Gli utenti possono utilizzare questa funzionalità quando desiderano terminare la loro sessione corrente, garantendo la sicurezza e la privacy del loro account.

3.1.2 Scenari

Gli scenari sono spesso utilizzati per illustrare e dettagliare i casi d'uso, mettendo in rilievo gli aspetti pratici delle interazioni tra utenti e sistema. Essi contribuiscono a catturare il contesto, i requisiti funzionali e le aspettative degli utenti, fornendo uno strumento prezioso per la comunicazione tra sviluppatori, progettisti e stakeholder. Andremo ora a illustrare i vari scenari che abbiamo introdotto.

UC1: New game

Attore primario	Studente registrato
Attore secondario	-
Descrizione	Permette ad uno studente di iniziare una nuova partita base.
Pre-Condizioni	Lo studente deve aver effettuato il login con le sue credenziali..
Sequenza di eventi principale	<ul style="list-style-type: none"> 1) Lo studente viene reindirizzato alla pagina Options. 2) Lo studente clicca sul bottone 'new game'. 3) Lo studente viene reindirizzato alla pagina main dove può scegliere i settaggi della nuova partita.
Post-Condizioni	-
Casi d'uso correlati	-
Sequenza di eventi alternativi	-

UC2: Match history

Attore primario	Studente registrato
Attore secondario	-
Descrizione	Permette ad uno studente di visualizzare lo storico delle partite giocate con le statistiche associate ad ogni partita.
Pre-Condizioni	Lo studente deve aver effettuato il login con le sue credenziali.
Sequenza di eventi principale	<ul style="list-style-type: none"> 1) Lo studente viene reindirizzato alla pagina Options 2) Lo studente clicca sul bottone 'match history'. 3) Lo studente viene reindirizzato alla pagina 'history' dove può visualizzare lo storico delle partite giocate e le statistiche associate ad ogni partita giocata.
Post-Condizioni	-
Casi d'uso correlati	-
Sequenza di eventi alternativi	-

UC3: Ranking

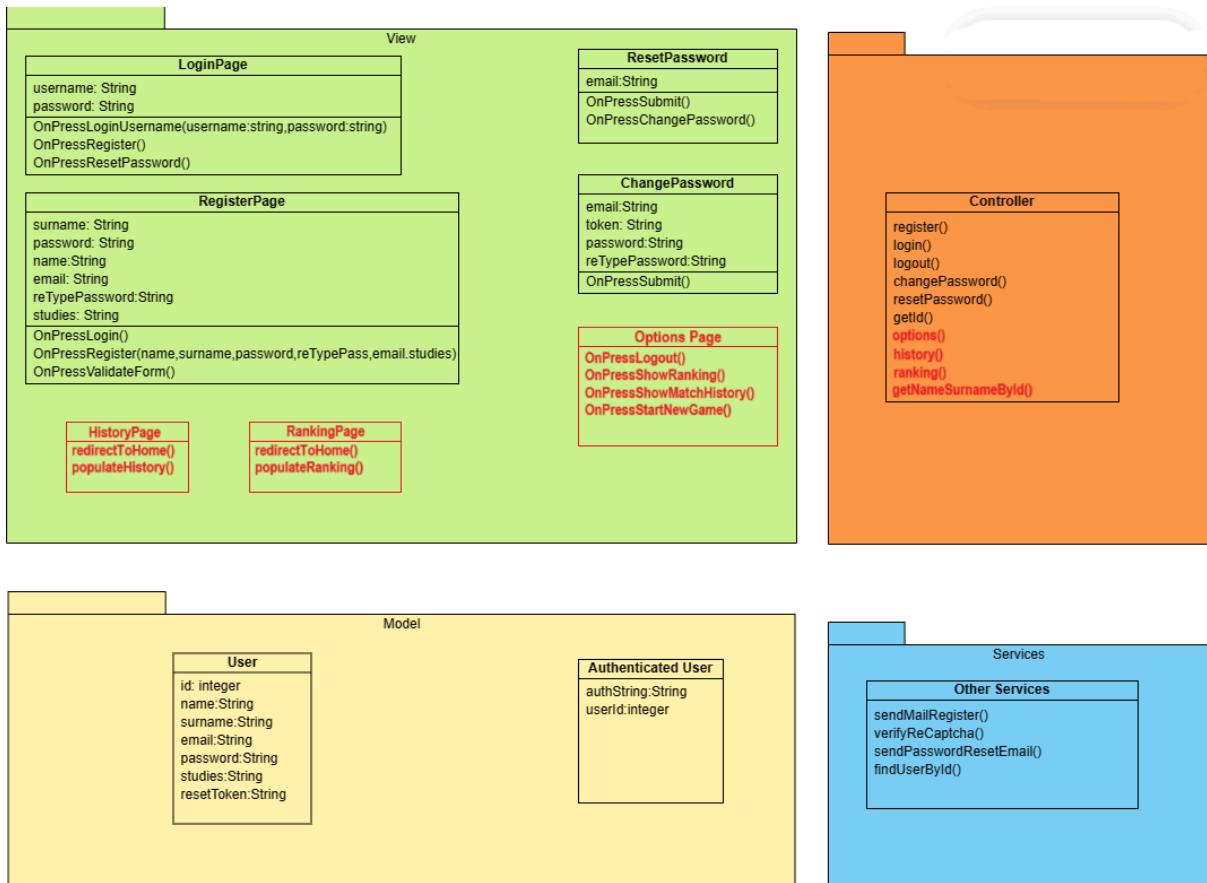
Attore primario	Studente registrato
Attore secondario	-
Descrizione	Permette ad uno studente di visualizzare la classifica totale di tutti i giocatori iscritti.
Pre-Condizioni	Lo studente deve aver effettuato il login con le sue credenziali.
Sequenza di eventi principale	<ol style="list-style-type: none">1) Lo studente viene reindirizzato alla pagina Options2) Lo studente clicca sul bottone 'ranking'.3) Lo studente viene reindirizzato alla pagina 'ranking' dove può visualizzare la classifica totale di tutti i giocatori.
Post-Condizioni	-
Casi d'uso correlati	-
Sequenza di eventi alternativi	-

UC4: Logout

Attore primario	Studente registrato
Attore secondario	-
Descrizione	Permette ad uno studente registrato di effettuare il logout.
Pre-Condizioni	Lo studente deve essersi loggato.
Sequenza di eventi principale	<ol style="list-style-type: none">1) Clicca su "Logout".
Post-Condizioni	Lo studente torna alla pagina di login.
Casi d'uso correlati	-
Sequenza di eventi alternativi	-

3.2. Package diagram

Per affrontare il requisito R9, abbiamo introdotto diverse pagine all'interno delle View, ognuna dotata di funzionalità specifiche. Questa espansione delle funzionalità è riflessa nel package diagram, il quale rappresenta in modo chiaro e organizzato le relazioni e le interconnessioni tra i diversi componenti che consentono l'esecuzione del requisito R9. In questa sezione, esploreremo le nuove aggiunte al diagramma di package, evidenziando come tali modifiche contribuiscano all'ampliamento delle capacità del sistema per soddisfare le richieste del requisito in questione.



- **HistoryPage**: Pagina dello Storico Partite con i metodi `redirectToHome()` per tornare alla pagina Home (options) e `populateHistory()` che effettua una chiamata API per popolare dinamicamente la tabella dello storico.
- **RankingPage**: Pagina della Classifica con i metodi `redirectToHome()` e `populateRanking()`.
- **OptionsPage**: Pagina Home con tre buttoni: uno per accedere alla pagina History, uno per la pagina Ranking, e uno per accedere alla pagina NewGame. Sono stati introdotti i metodi `OnPressLogout()` per consentire all'utente di effettuare il Logout, `OnPressShowRanking()`, `OnPressShowMatchHistory()` e `OnPressStartNewGame()` per accedere alle rispettive pagine cliccando sui rispettivi buttoni.

Inoltre, sono stati aggiunti nuovi metodi nel Controller:

- **options()**, **history()**, **ranking()**: Metodi utilizzati per effettuare il redirect alle rispettive pagine `options.html`, `history.html` e `ranking.html`, verificando la validità del JWT estratto dal cookie.
- **getNameSurnameById()**: Metodo utilizzato per effettuare una richiesta al Database al fine di ottenere le informazioni di Nome e Cognome dell'utente registrato tramite l'ID passato come input.

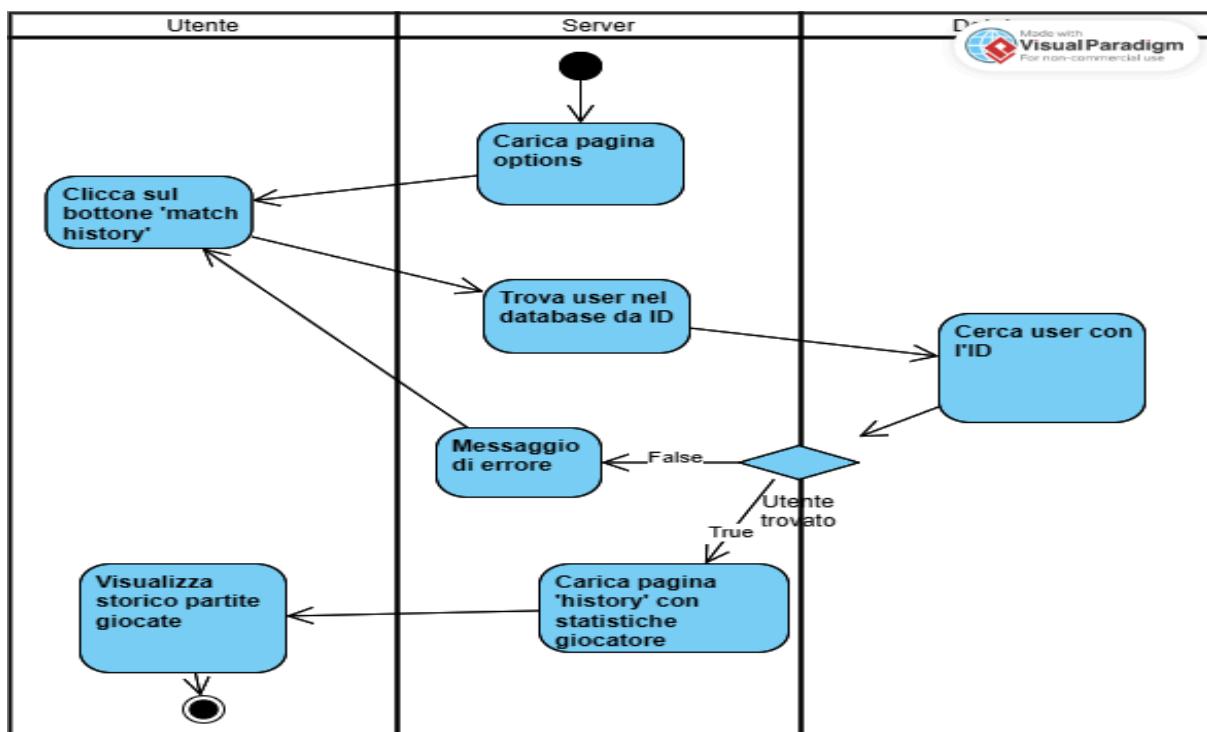
3.3. Viste dinamiche

Per comprendere meglio il flusso delle operazioni, utilizzeremo, per ciascuno scenario, sia il diagramma di attività che il diagramma di sequenza.

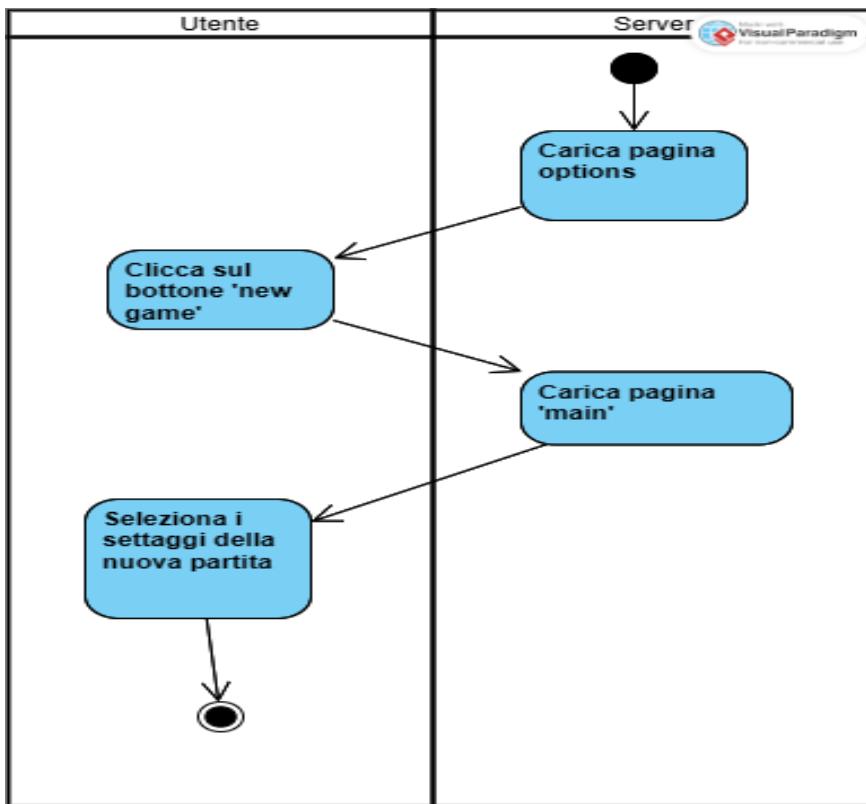
Il diagramma di attività consente di rappresentare in modo grafico le attività, le azioni e le decisioni che compongono il processo, mentre il diagramma di sequenza viene impiegato per rappresentare l'interazione tra gli oggetti o le entità all'interno di un sistema software in un ordine cronologico. In un diagramma di sequenza di tipo MVC, il controller riceve la richiesta dall'utente e la inoltra ai dati del model (Database), che li elabora e li restituisce al controller. Quindi, il controller utilizza tali dati per aggiornare la view (Studente) e restituisce la view aggiornata all'utente.

3.3.1 Activity diagram

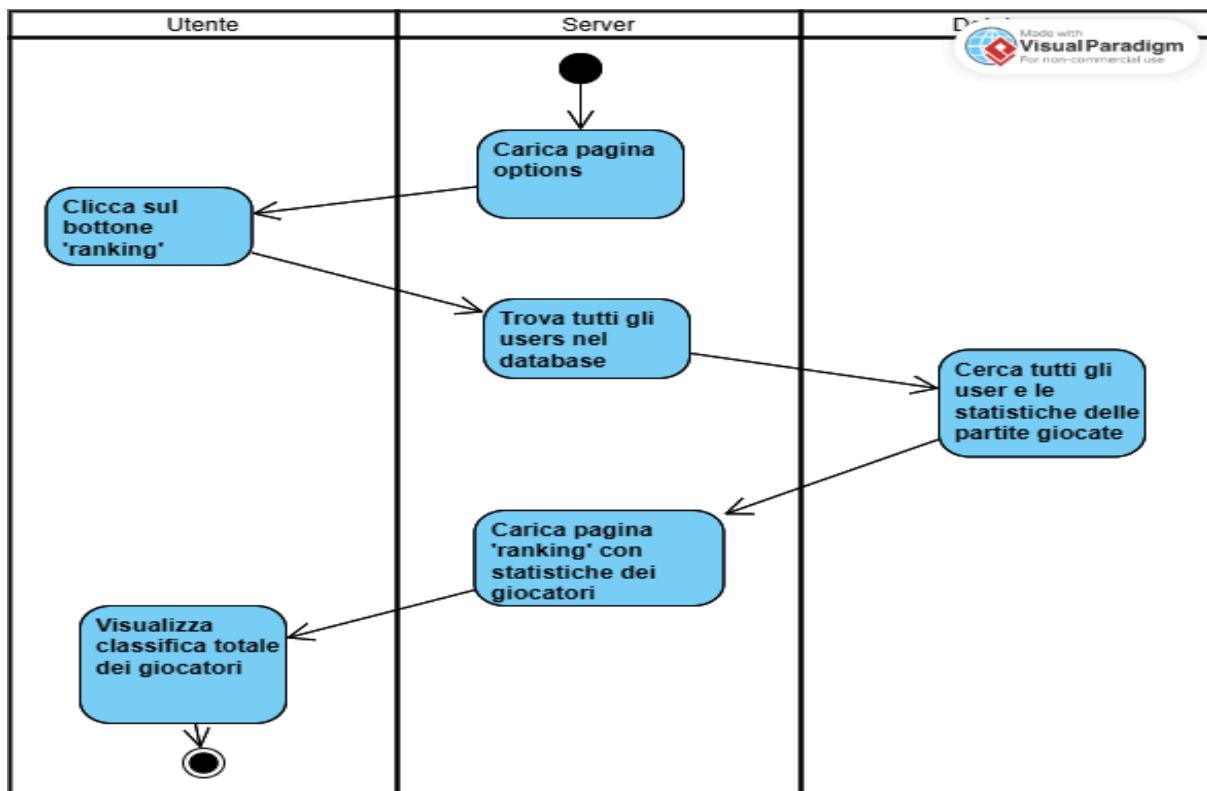
Match history



New game

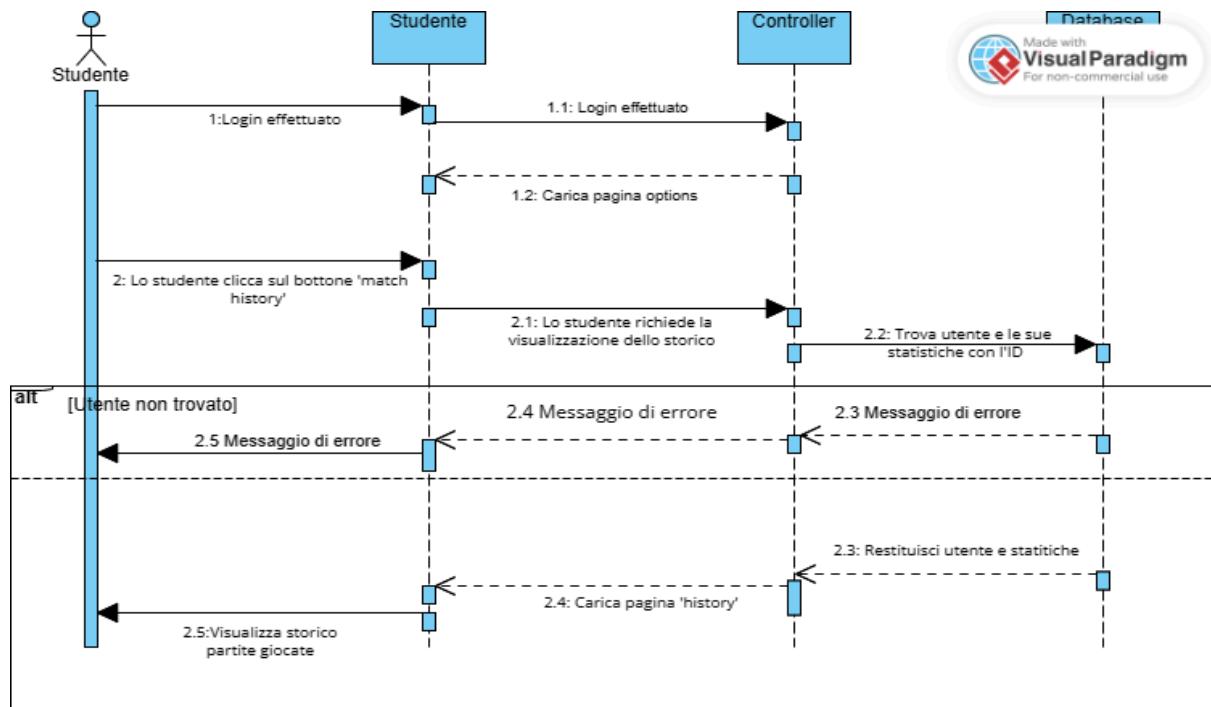


Ranking

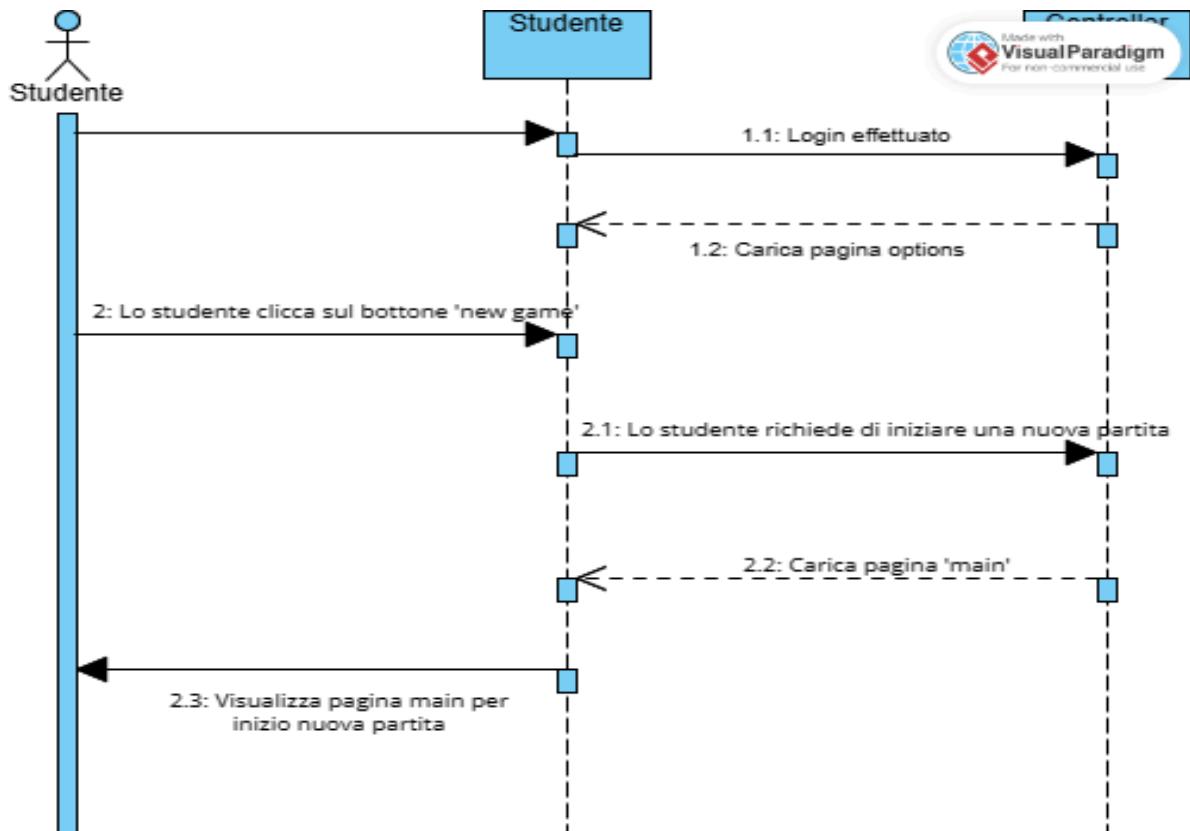


3.3.2 Sequence diagram

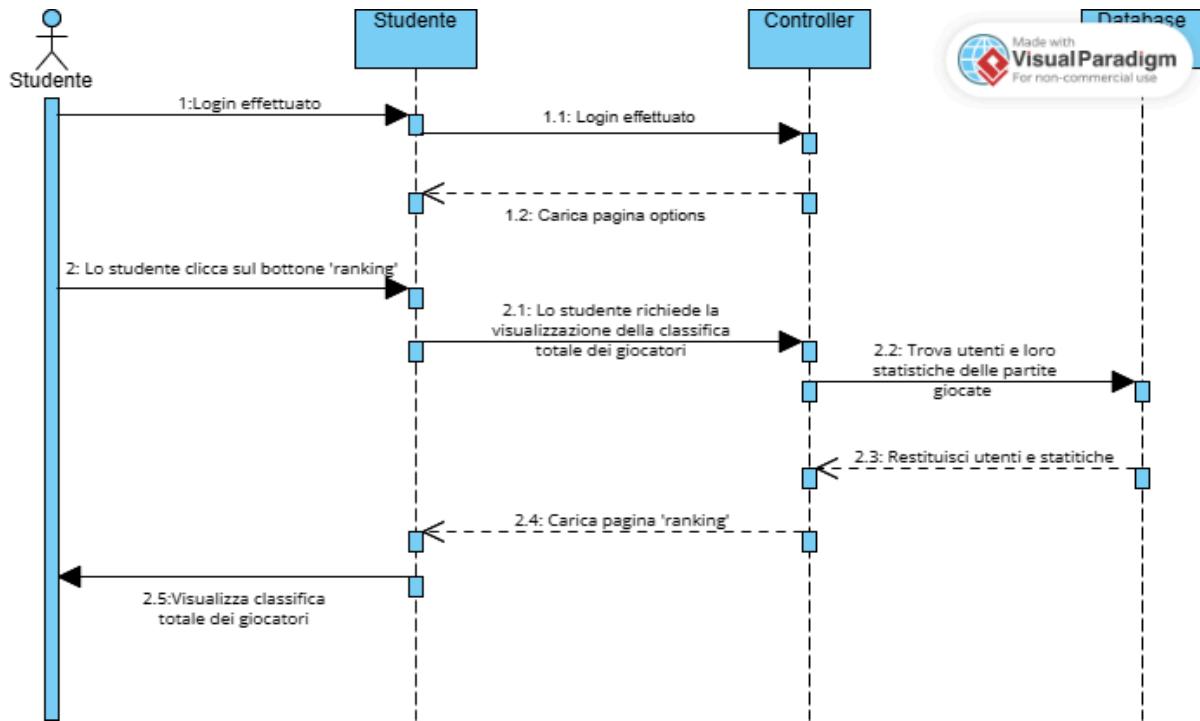
Match History



New Game



Ranking



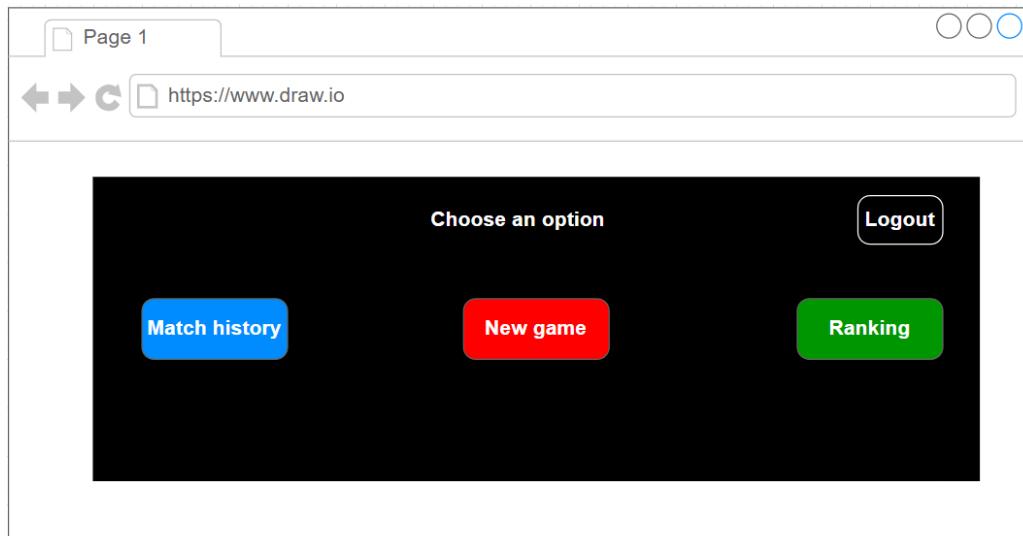
3.4. Testing

Il testing è un processo fondamentale finalizzato a valutare la correttezza, l'affidabilità e le prestazioni di un software. Attraverso il rilevamento e la correzione di difetti, il

testing assicura che l'applicazione o il sistema soddisfi le esigenze degli utenti. Questo processo, che può coprire diversi livelli come unità, integrazione, sistema e accettazione utente, contribuisce significativamente al miglioramento complessivo della qualità del software. Implementando strategie di testing adeguate, gli sviluppatori riducono i rischi di malfunzionamenti, garantendo la fornitura di un prodotto affidabile e di alta qualità.

Test Case ID	Descrizione		Pre-condizioni	Input	Output	Post-condizioni	Esito
1 R9	Accesso alla pagina options		Utente autenticato	Dati di autenticazione	N/A	L'utente viene reindirizzato alla pagina options	Pass
2 R9	Visualizzare storico partite		Utente autenticato	Click sul relativo bottone	"View the history of the games you have played"	N/A	Pass
3 R9	Nuova partita base		Utente autenticato	Click sul relativo bottone	N/A	L'utente è reindirizzato alla pagina per creare una partita contro un robot a scelta	Pass
4 R9	Nuova partita contro tutti i robot		Utente autenticato	Click sul relativo bottone	N/A	N/A	Pass
5 R9	Visualizza classifica generale		Utente autenticato	Click sul relativo bottone	"View the overall players' ranking"	N/A	Pass
6 R9	Logout		Utente autenticato	Click sul relativo bottone	N/A	Utente non più autenticato	Pass
1 PC	Cambio password	User registrato, Token di reset password.	albertoarola@gmail.com ey...zl Federico3! Federico3!		N/A	Password modificata dall'Utente registrato e redirezione alla pagina options	PASS

3.5. Mockup delle nuove pagine



Match history								
Giocatore	Robots	Durata	Difficoltà	Vincitore	Score Giocatore	Score Robot	Classe testata	Data
Antonio Paolino	Randoop Evosuite	00:25:34	Intermediate	Antonio Paolino	92	82	Calcolatrice	2023-12-06 18:45:00
Alfonso Esposito	Evosuite	00:20:30	Intermediate	Evosuite	62	80	Calcolatrice	2023-12-06 20:45:00

Ranking				
Giocatore	Partite totali	Partite Vinte	Partite perse	Score
Antonio Paolino	10	8	2	82
Alfonso Esposito	12	6	6	70

3.6. Proposta modifica database T4

Diagramma ER attuale:

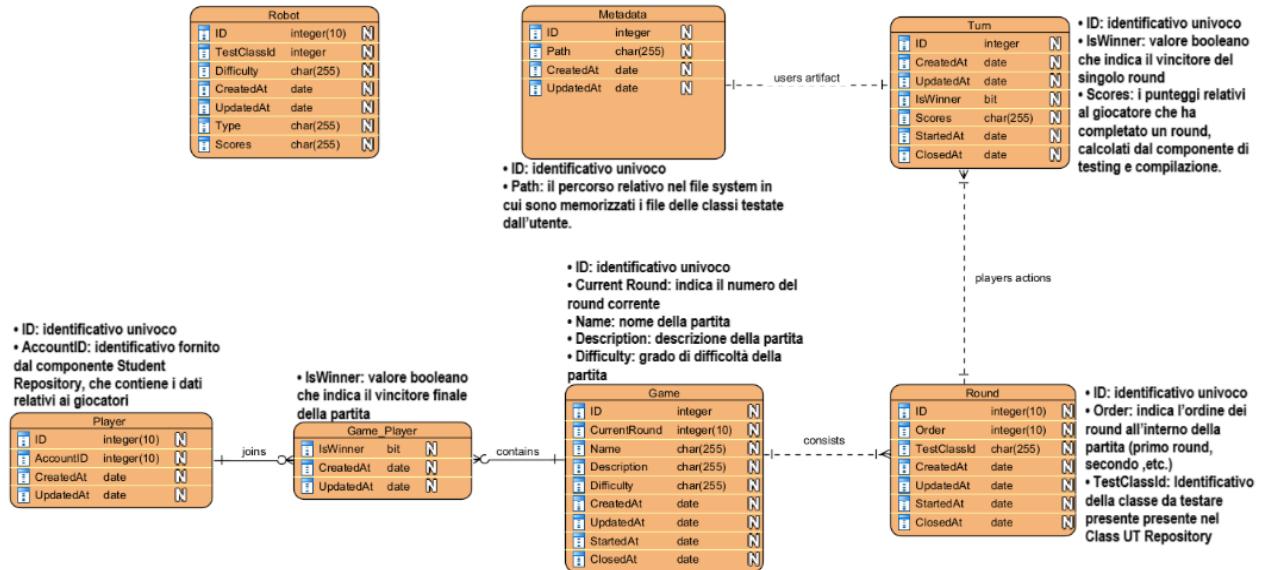
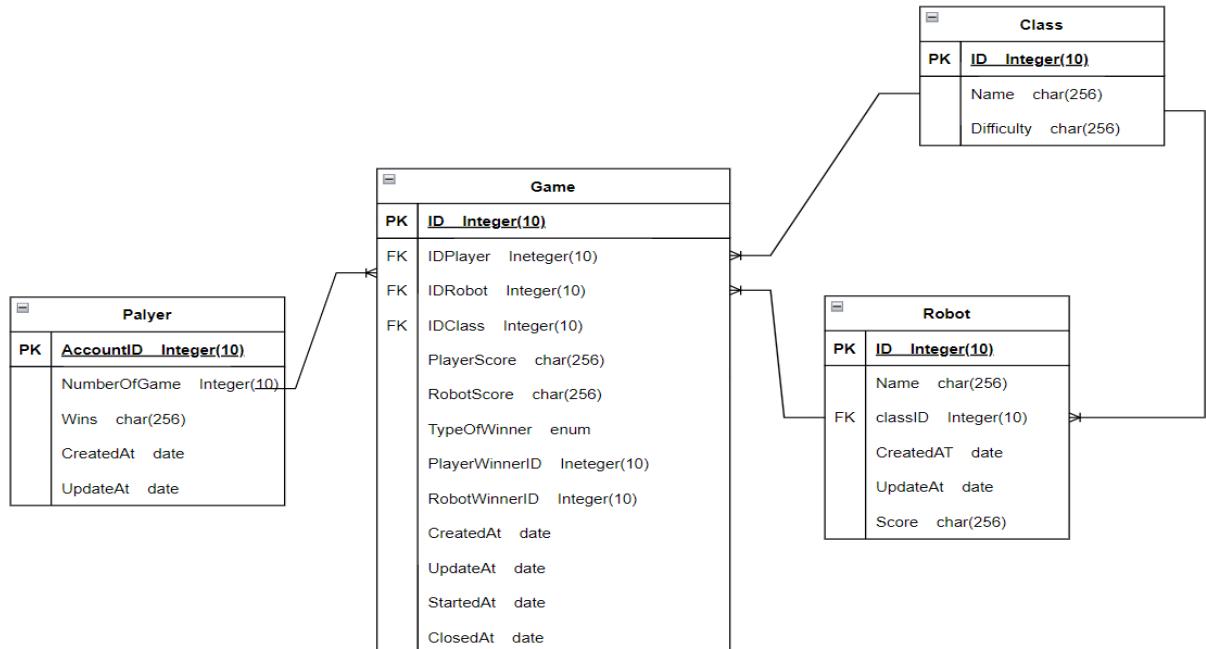


Diagramma ER modificato:



3.7. API

Al seguente link è presente una documentazione sulle API da implementare sul nuovo database proposto:

https://app.swaggerhub.com/apis/ALBERTOAROLA/api-db_t_4/1.0.0

Games

GET	/games Recupera l'elenco completo dei giochi	📋 ← ⏪ ⏴
POST	/games Crea un nuovo gioco	📋 ← ⏪ ⏴
GET	/games/{id} Recupera un gioco specifico	📋 ← ⏪ ⏴
PUT	/games/{id} Aggiorna un gioco esistente	📋 ← ⏪ ⏴
DELETE	/games/{id} Elimina un gioco	📋 ← ⏪ ⏴

Players

GET	/players Recupera l'elenco completo dei giocatori	📋 ← ⏪ ⏴
POST	/players Crea un nuovo giocatore	📋 ← ⏪ ⏴
GET	/players/{accountID} Recupera un giocatore specifico	📋 ← ⏪ ⏴
PUT	/players/{accountID} Aggiorna un giocatore esistente	📋 ← ⏪ ⏴
DELETE	/players/{accountID} Elimina un giocatore	📋 ← ⏪ ⏴

Robots

GET	/robots Recupera l'elenco completo dei robot	📋 ← ⏪ ⏴
POST	/robots Crea un nuovo robot	📋 ← ⏪ ⏴
GET	/robots/{id} Recupera un robot specifico	📋 ← ⏪ ⏴
PUT	/robots/{id} Aggiorna un robot esistente	📋 ← ⏪ ⏴
DELETE	/robots/{id} Elimina un robot	📋 ← ⏪ ⏴

Classes

GET	/classes Recupera l'elenco completo delle classi	📋 ← ⏪ ⏴
POST	/classes Crea una nuova classe	📋 ← ⏪ ⏴
GET	/classes/{id} Recupera una classe specifica	📋 ← ⏪ ⏴
PUT	/classes/{id} Aggiorna una classe esistente	📋 ← ⏪ ⏴
DELETE	/classes/{id} Elimina una classe	📋 ← ⏪ ⏴

4. Installazione

Il seguente capitolo si focalizza sul processo di installazione del software.

Si elencano di seguito in maniera sistematica i passi necessari per la corretta installazione ed esecuzione del software.

Il progetto è stato forkato dal progetto originale, ed è disponibile al seguente repository GitHub: <https://github.com/AntoP96/A2-2024>.

<NOTA>: Nel progetto complessivo, è stato integrato un nuovo servizio (FastAPI) dedicato al popolamento delle tabelle di storico delle partite e della classifica dei giocatori. Questo servizio è puramente illustrativo, poiché le API originariamente disponibili non consentivano di recuperare tutti i dati necessari per popolare le suddette tabelle. Al momento della modifica del database T4 (*consultare il capitolo sulla proposta di modifica del database T4*) e la creazione delle nuove API secondo lo schema Swagger API (*consultare il capitolo sulle API*), sarà possibile eliminare il servizio FastAPI. Le nuove API potranno quindi sostituire il servizio nelle pagine *ranking.js* (API classifica giocatori) e *history.js* (API storico partite).

4.1. Configurazione Docker File e Docker Compose

L'applicazione è dotata di un Docker File e un Docker Compose, in cui sono contenute le informazioni necessarie per la configurazione del container Docker e dei relativi porti di comunicazione (configurabili). Analizziamo in seguito i due file dettagliatamente.

DOCKER FILE

```
FROM eclipse-temurin:17-jdk-alpine
ARG JAR_FILE=target/DB_Setup-0.0.1-SNAPSHOT.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

Tramite il Docker File è stata creata un'immagine Docker basata su Alpine Linux con Java 17.

Tale file definisce le seguenti istruzioni:

- **FROM eclipse-temurin:17-jdk-alpine**: Specifica l'immagine di base da cui derivare, che è basata su Alpine Linux e include OpenJDK 17 fornito da Eclipse Temurin.
- **ARG JAR_FILE=target/DB_Setup-0.0.1-SNAPSHOT.jar**: Definisce un argomento denominato JAR FILE con il percorso predefinito del file JAR. Questo argomento può essere sovrascritto al momento della build dell'immagine Docker.
- **COPY \${JAR_FILE} app.jar**: Copia il file JAR specificato (definito dall'argomento JAR_FILE) nell'immagine Docker e lo rinomina come "app.jar". Questo file JAR sarà l'eseguibile principale dell'applicazione quando il container viene avviato.

- **ENTRYPOINT ["java","-jar","/app.jar"]**: Definisce il punto di ingresso dell'applicazione quando il container viene avviato. In questo caso, l'applicazione Java viene eseguita con il comando `"java -jar /app.jar"`.

DOCKER COMPOSE

```

version: '3.1'

services:
  app:
    build: .
    expose:
      - 8080
    ports:
      - "8081:8080"
    depends_on:
      - db
    environment:
      DB_URL: jdbc:mysql://db:3306/STUDENTSREPO
    networks:
      - global-network

  db:
    image: mysql:latest
    environment:
      MYSQL_ROOT_PASSWORD: 'password'
      MYSQL_DATABASE: STUDENTSREPO
    ports:
      - "3306:3306"
    healthcheck:
      test: ["CMD", "mysqladmin", "ping", "-h", "localhost"]
      timeout: 20s
      retries: 10
    networks:
      - global-network

networks:
  global-network:
    external: true

```

Il Docker Compose descrive l'architettura Docker multi-container necessaria per il funzionamento della nostra applicazione, include un servizio per il database MySQL e uno per l'applicazione.

- ***version***: '3.1': Specifica la versione della sintassi di Docker Compose utilizzata nel file.
- ***services***: Definisce i servizi Docker che saranno orchestrati insieme.
 - ***app***: Configura il servizio dell'applicazione.
 - ***build***: Specifica la directory di build per l'immagine del servizio.
 - ***expose***: - **8080**: Espone la porta 8080 del container.
 - ***ports***: - "**8081:8080
 - ***depends_on***: - **db**: Indica che il servizio "app" dipende dal servizio "db" e deve essere avviato dopo di esso.
 - ***environment***: **DB_URL: jdbc:mysql://db:3306/STUDENTSREPO**: Configura la variabile d'ambiente DB_URL con l'URL del database.
 - ***networks***: - **global-network**: Collega il servizio "app" alla rete denominata "global-network".**
 - ***db***: Configura il servizio del database MySQL.
 - ***image***: **mysql:latest**: Specifica l'immagine MySQL più recente da utilizzare.
 - ***environment***:
 - MYSQL_ROOT_PASSWORD: 'password'**
 - MYSQL_DATABASE: STUDENTSREPO**
Configura le variabili d'ambiente per la password di root e il nome del database.
 - ***ports***: - "**3306:3306
 - ***healthcheck***:
 - test: ["CMD", "mysqladmin", "ping", "-h", "localhost"]**
 - timeout: 20s**
 - retries: 10**
Configura il health check per il servizio "db" usando mysqladmin per pingare il database.
 - ***networks***: - **global-network**: Collega il servizio "db" alla rete denominata "global-network".**
- ***networks: global-network***: Definisce la rete denominata "global-network", alla quale sono collegati entrambi i servizi.

4.2. Maven Install e avvio del Docker File

Per poter eseguire l'applicazione tramite Docker occorre preliminarmente effettuare l'installazione del Maven.

Di seguito, sono riportate le due metodologie per poter effettuare tale operazione:

1. Per la prima procedura bisogna assicurarsi di avere Maven installato sul proprio sistema. Tramite il terminale o il prompt dei comandi, posizionandosi nella directory del progetto in cui è presente il file **pom.xml**, è necessario eseguire il comando **mvn clean install**. Questo comando esegue un'operazione di "pulizia" (clean) che rimuove i file generati dalle build precedenti e quindi esegue la build effettiva (install) del progetto. Durante la build, Maven scaricherà le dipendenze necessarie, compilerà il codice sorgente, eseguirà i test e genererà il pacchetto dell'applicazione. Una volta completata con successo la build, verrà generato un file JAR denominato "**DB_Setup-0.0.1-SNAPSHOT.jar**" nella cartella "**target**".
2. Per la seconda procedura bisogna effettuare la clean e la maven install tramite SpringTool. In particolare, premendo col tasto destro sul pom.xml, si deve cliccare sulla voce "**Run as -> maven clean**" e successivamente sulla voce "**Run as -> maven install**". Una volta completata con successo la build, verrà generato un file JAR denominato "**DB_Setup-0.0.1-SNAPSHOT.jar**" nella cartella "**target**".

Dopo aver eseguito l'install del maven è necessario, dopo aver aperto *Docker desktop*, eseguire il docker compose con il comando "**docker-compose up -d --build**" in una finestra di terminale, posizionandosi nella directory del progetto in cui è presente il file "**docker-compose.yml**".

In questo modo è possibile visualizzare la web application e testare il suo corretto funzionamento accedendo alla pagina <http://localhost:8081/login>.

5. Demo

Un video dimostrativo delle funzionalità della web application è disponibile su YouTube: https://youtu.be/S0m-hs1gaKo?si=Y3m5x03FgC_OVKBP.

LOGIN

Username
Password

Accedi

[Non sei ancora registrato? Registrati.](#)
[Hai dimenticato la password?](#)

REGISTRAZIONE

Nome
Cognome
antonio.paolino@studenti.unina.it
Conferma Password

BSc

Non sono un robot reCAPTCHA
Privacy - Terms

Invia

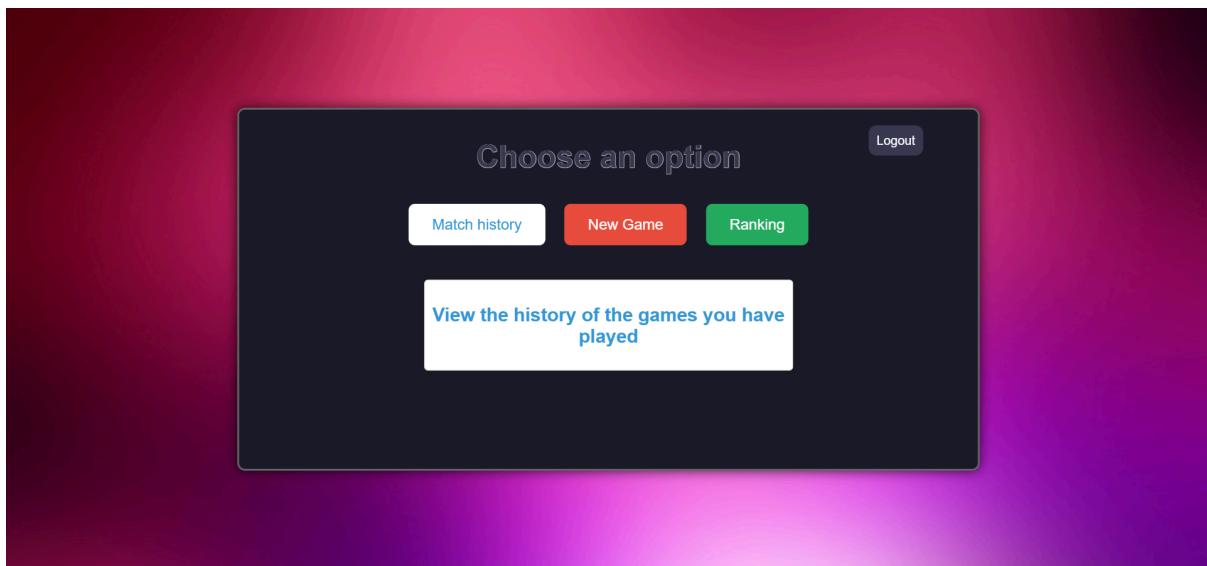
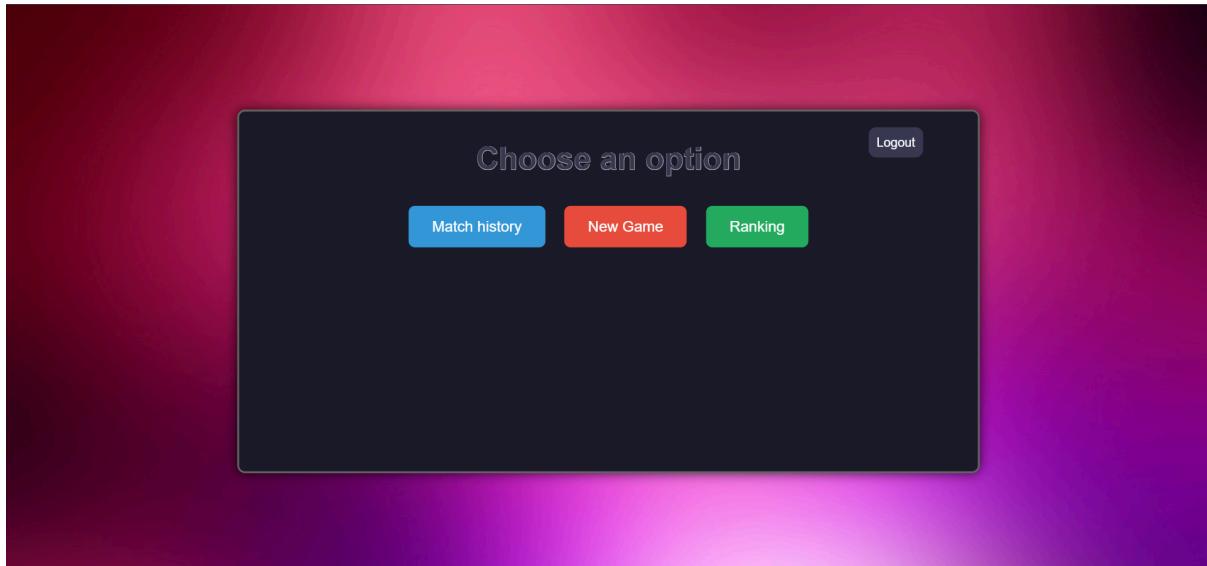
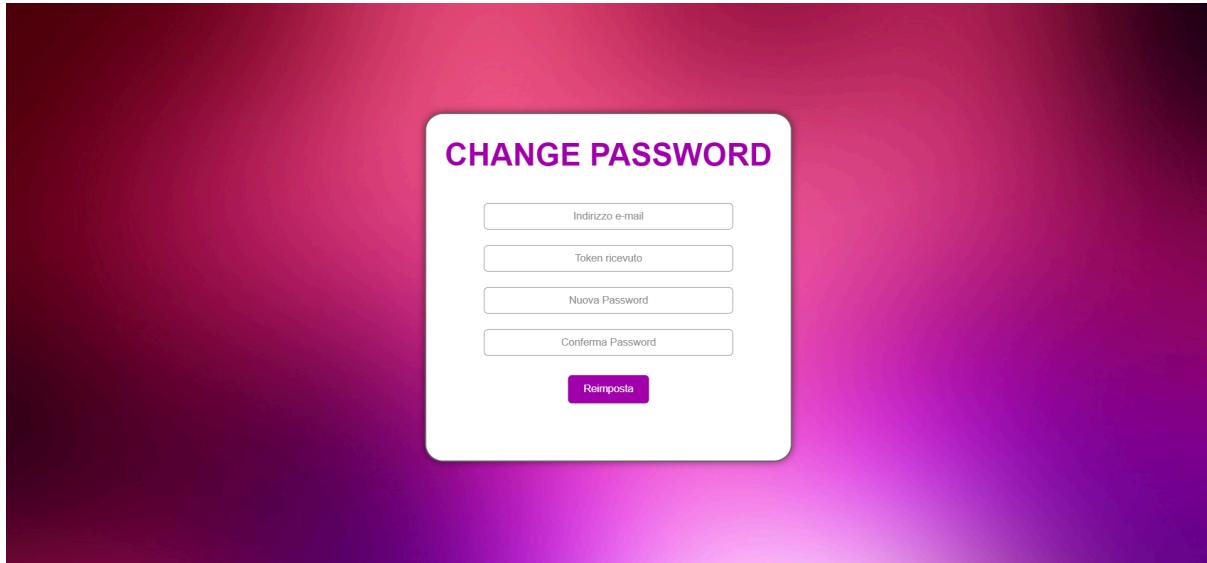
[Sei già registrato? Accedi.](#)

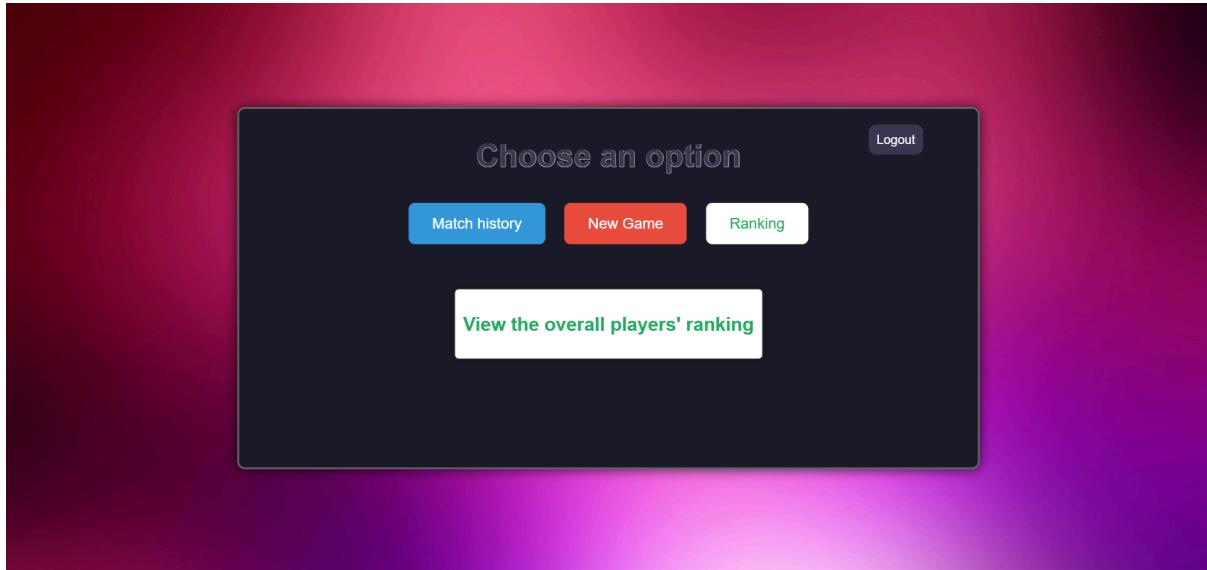
RESET PASSWORD

Inserisci il tuo indirizzo email e
ti invieremo un token per
reimpostare la password.

Indirizzo e-mail
Invia

[Hai già ricevuto il token? Cambia la
password](#)





ID	Robots	Durata	Classe Testata	Difficoltà	Vincitore	Score Giocatore	Score Robot	Data Inizio	Data Fine
1	Randoop, Evosuite	01:00:00	Calcolatrice	medium	Antonio Paolino	92	82	2023-12-06 18:45:00	2023-12-06 19:45:00
2	Randoop, Evosuite	00:45:00	Calcolatrice	easy	Antonio Paolino	88	70	2023-12-07 14:30:00	2023-12-07 15:15:00

Giocatore	Partite Totali	Partite Vinte	Partite Perse	Score Medio
👑 Antonio Paolino	10	8	2	82
Alfonso Esposito	12	6	6	70
Alberto Arola	10	8	7	68
Gianfranco Amadio	12	6	10	60