UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

CORSO DI LAUREA IN INGEGNERIA INFORMATICA
CORSO DI SYSTEM ARCHITECTURE DESIGN
PROF. FASOLINO- A.A. 2022 - 23

Task 2-3

Studenti:

Doriana Traetto M63001416 d.traetto@studenti.unina.it
Marika Sasso M63001438 marik.sasso@studenti.unina.it
Francesca Terracciano M63001550 francesca.terracciano@studenti.unina.it
Iole Morabito M63001448 i.morabito@studenti.unina.it

Indice

1. METODOLOGIA DI LAVORO

Per lo sviluppo del nostro software abbiamo adottato un *approccio agile* basato sulle iterazioni. La nostra metodologia di lavoro si è basata su sprint settimanali, durante i quali i componenti del gruppo si sono concentrati su un insieme di funzionalità specifiche. Ogni sprint è stato preceduto da una *pianificazione* in cui sono stati definiti gli *obiettivi* dell'iterazione e le attività necessarie per raggiungerli.

Inoltre, si è fatto uso della pratica del *pair programming*, che consiste nel lavorare a due su un singolo codice. Questa pratica ci ha consentito di migliorare la qualità del codice e di ridurre il numero di errori, in quanto ogni riga di codice è stata esaminata da almeno due persone. Inoltre, il pair programming ha consentito di diffondere la conoscenza e le competenze tra i membri del team, promuovendo la collaborazione e la condivisione delle conoscenze.

Fondamentale il *diario di bordo,* uno strumento di tracciamento utilizzato per registrare le attività, le decisioni e i problemi riscontrati durante il processo di sviluppo del software. Essere in grado di registrare queste informazioni ha aiutato il team a mantenere un alto livello di trasparenza, comunicazione e collaborazione durante tutto il processo di sviluppo del software.

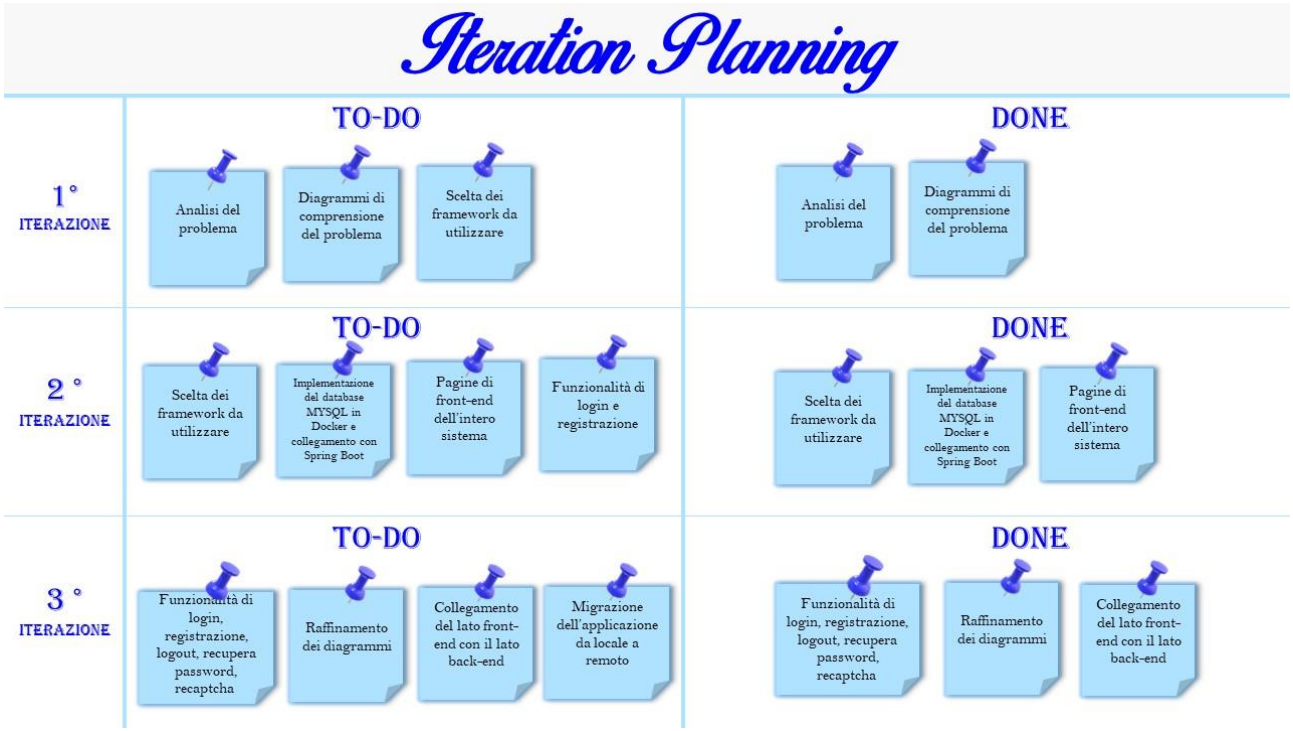

Figura 1: Iteration Planning.

2. SPECIFICA DEI REQUISITI

2.1 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).

2.2 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.

. = classi-attori
. = attributi
. = funzionalità

2.3 GLOSSARIO

Termine	Descrizione	Sinonimi
Studente	Colui che deve effettuare la registrazione.	Giocatore, Utente
Login	Procedura per l'autenticazione.	Autenticazione
Logout	Procedura per effettuare l'uscita dal profilo.	Disconnessione

2.4 REQUISITI

Per la definizione dei requisiti non funzionali ci si è basati sul modello già noto FURPS+, al quale sono stati aggiunti ulteriori requisiti oltre a quelli base definiti dal modello.

2.5 REQUISITI INFORMALI

Si vuole realizzare un'applicazione dove lo studente può effettuare due operazioni principali: registrazione e autenticazione.

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.

2.6 REQUISITI FUNZIONALI

RF-1 Il sistema deve consentire ad ogni utente di potersi registrare nel sistema
RF-2 Il sistema, in caso di avvenuta registrazione, deve inviare all'utente registrato un'e-mail contenente l'id.
RF-3 Il sistema deve permettere all'utente registrato di effettuare il login.
RF-4 In caso di login errato, il sistema deve restituire un errore.
RF-5 Il sistema deve permettere all'utente registrato di impostare una nuova password in caso l'utente abbia dimenticato la propria.

RF-6 Il sistema deve permettere all'utente registrato di accedere all'area riservata solo dopo aver effettuato il login.
RF-7 Il sistema deve permettere all'utente registrato di effettuare il logout.

2.7 REQUISITI SUI DATI

RD-1 La registrazione è caratterizzata da: nome, cognome, e-mail, password, corso di studi.
RD-2 A ciascun utente registrato è assegnato un identificativo univoco.
RD-3 A ciascun utente autenticato è assegnato un token di autenticazione.

2.8 REQUISITI NON FUNZIONALI

RNF-1 *Sicurezza*:
- È raccomandabile criptare la password e non salvarla in chiaro nel database.
- In fase di registrazione l'identità dello studente deve essere verificata anche tramite un recaptcha.
- Si deve effettuare una ricerca di tutti i caratteri pericolosi per la sicurezza del sistema: < > & ( ) , % ' ? + poiché possono essere utilizzati per eventuali attacchi, ad esempio l'*SQL injection*.

RNF-2 *Interoperabilità*:
- Il sistema deve essere in grado di interagire con altri sistemi per lo scambio di informazioni, infatti, l'ID associato al giocatore deve essere fornito agli altri sistemi che ne necessitano.

RNF-3 *Usabilità*:
- Il sistema deve fornire un funzionamento intuitivo, facilmente apprendibile ed esteticamente piacevole.

RNF-4 *Accuratezza*:
- Il sistema deve fornire i giusti o concordati risultati o effetti, con la precisione richiesta. Ad esempio, la restituzione dell'ID quando la registrazione avviene correttamente o la verifica del recaptcha.

RNF-5 *Efficienza*:
- Il sistema deve realizzare le funzioni richieste nel minor tempo possibile ed utilizzando nel miglior modo le risorse necessarie, quando opera in determinate condizioni.

2.9 MODELLI DEI 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: generateToken
- UC7: changePassword

2.10 DIAGRAMMA DEI CASI D'USO

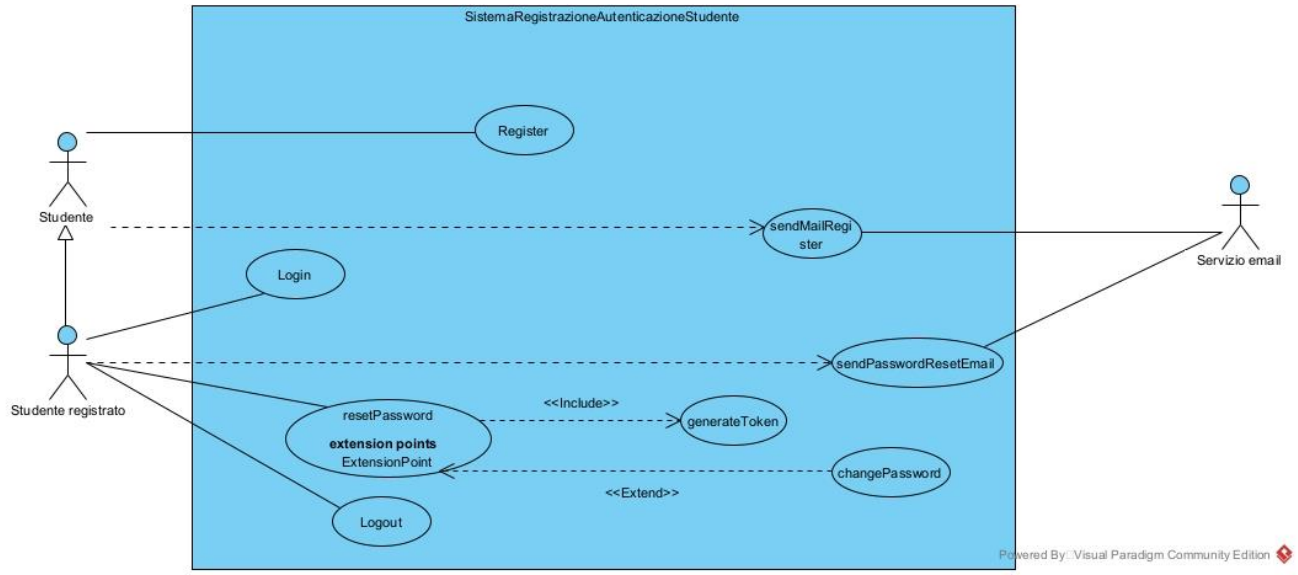

Figura 2: Diagramma dei casi d'uso

2.11 SCENARI

Caso d'uso	*Register*
Attore primario	Studente
Attore secondario	-
Descrizione	Permette ad uno studente di registrarsi per accedere alle funzionalità di gioco.
Pre-Condizioni	Lo studente non deve essere già registrato.
Sequenza di eventi principale	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.

Caso d'uso	*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.

Caso d'uso	*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	-

Caso d'uso	*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	-

Caso d'uso	*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	1) Clicca su "Logout".
Post-Condizioni	Lo studente torna alla pagina di login.
Casi d'uso correlati	-
Sequenza di eventi alternativi	-

Caso d'uso	*Reset_password*
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	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	-

Caso d'uso	*Change_password*
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	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	-

3. USER STORIES

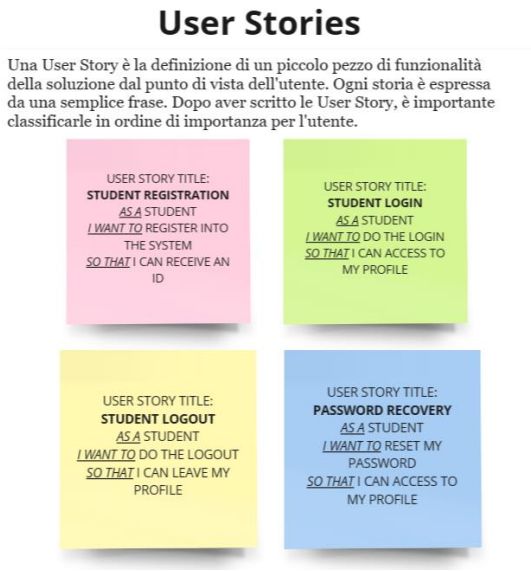

Figura 3 - User Stories

3.1 Criteri di accettazione

Figura 4 - Criteri di accettazione delle User Story precedentemente definite

3.2 MODELLAZIONE DEI DATI

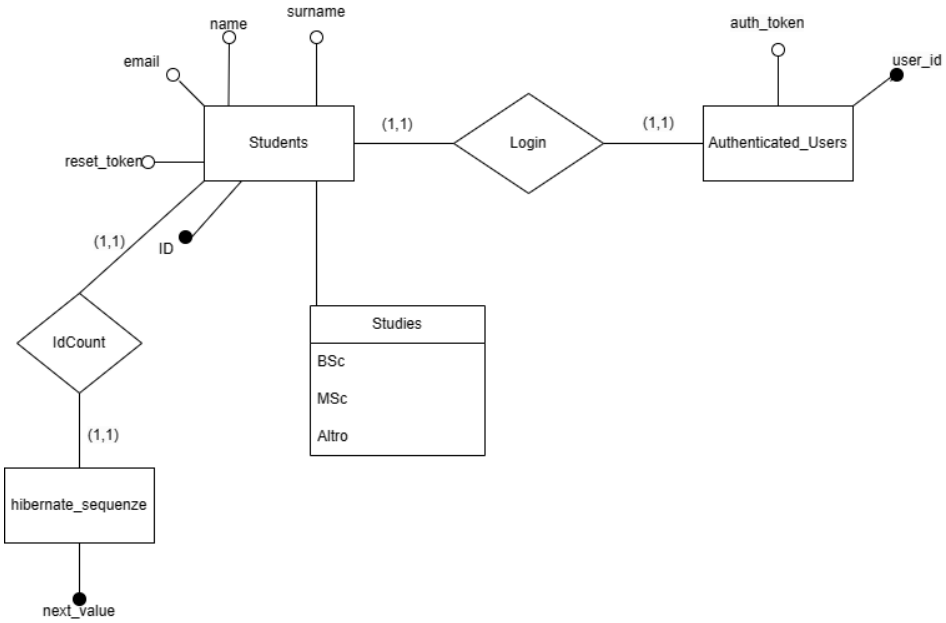

Figura 5 - Modello dei dati del sistema

3.3 DIAGRAMMI DI ATTIVITÀ

3.3.1 Diagramma di attività REGISTER

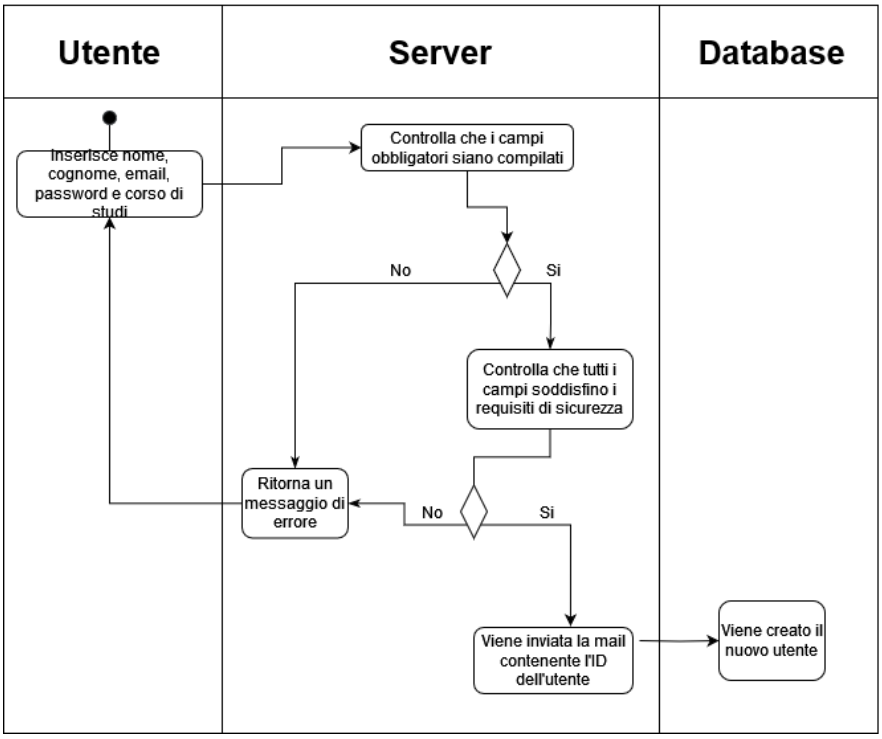

Figura 6: Diagramma di attività della REGISTRAZIONE.

3.3.2 Diagramma di attività LOGIN

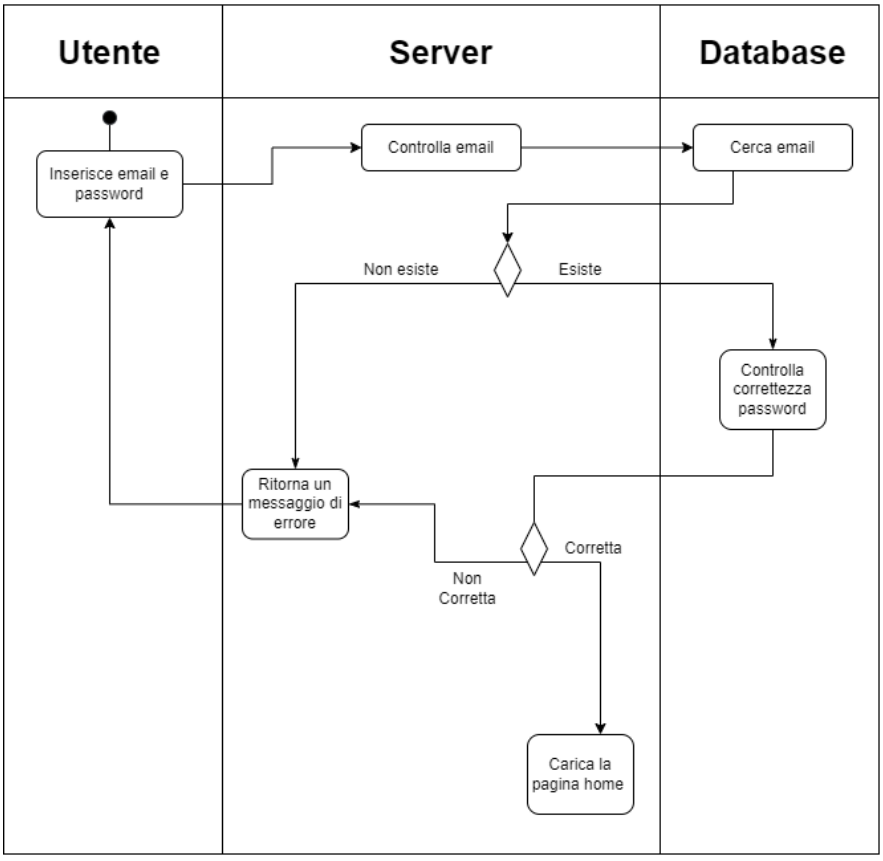

Figura 7: Diagramma di attività del LOGIN.

3.3.3 Diagramma di attività LOGOUT

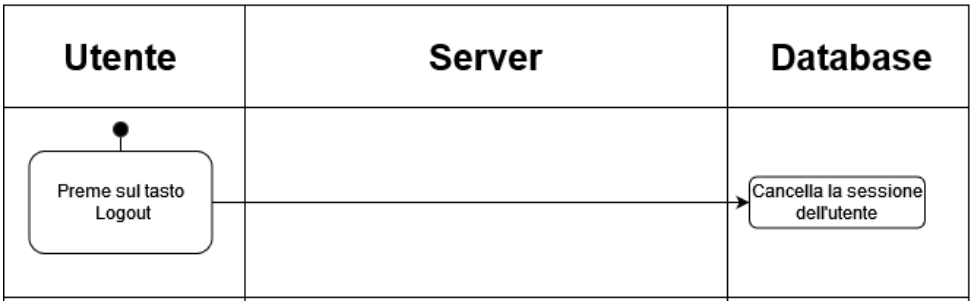

Figura 8: Diagramma di attività del LOGOUT.

3.3.4 Diagramma di attività RESET_PASSWORD

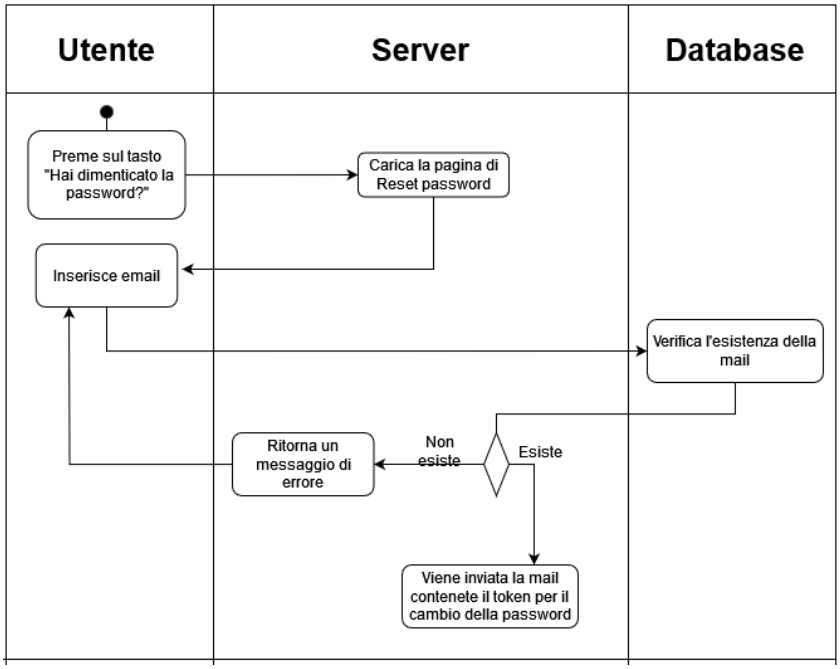

Figura 9: Diagramma di attività di RESET PASSWORD.

3.3.5 Diagramma di attività CHANGE_PASSWORD

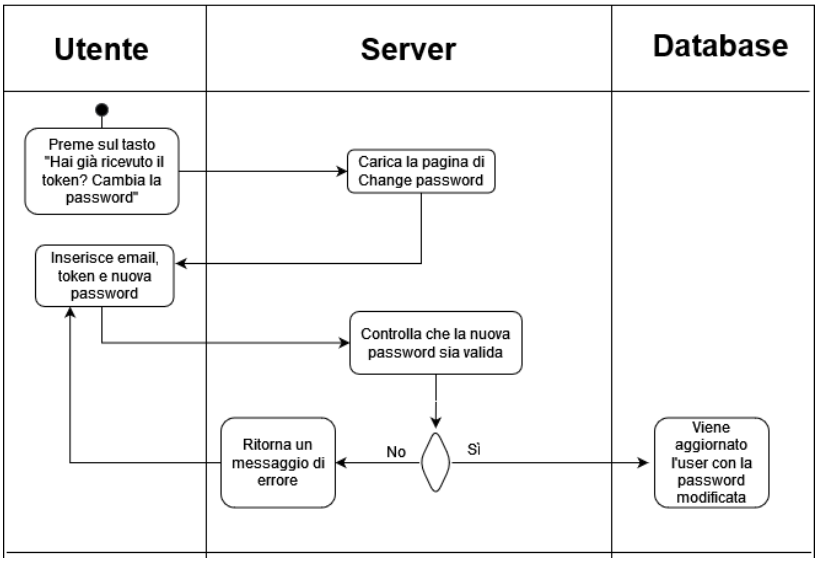

Figura 10: Diagramma di attività di CHANGE PASSWORD.

3.4 DIAGRAMMI DI SEQUENZA

3.4.1 Diagramma di sequenza REGISTER

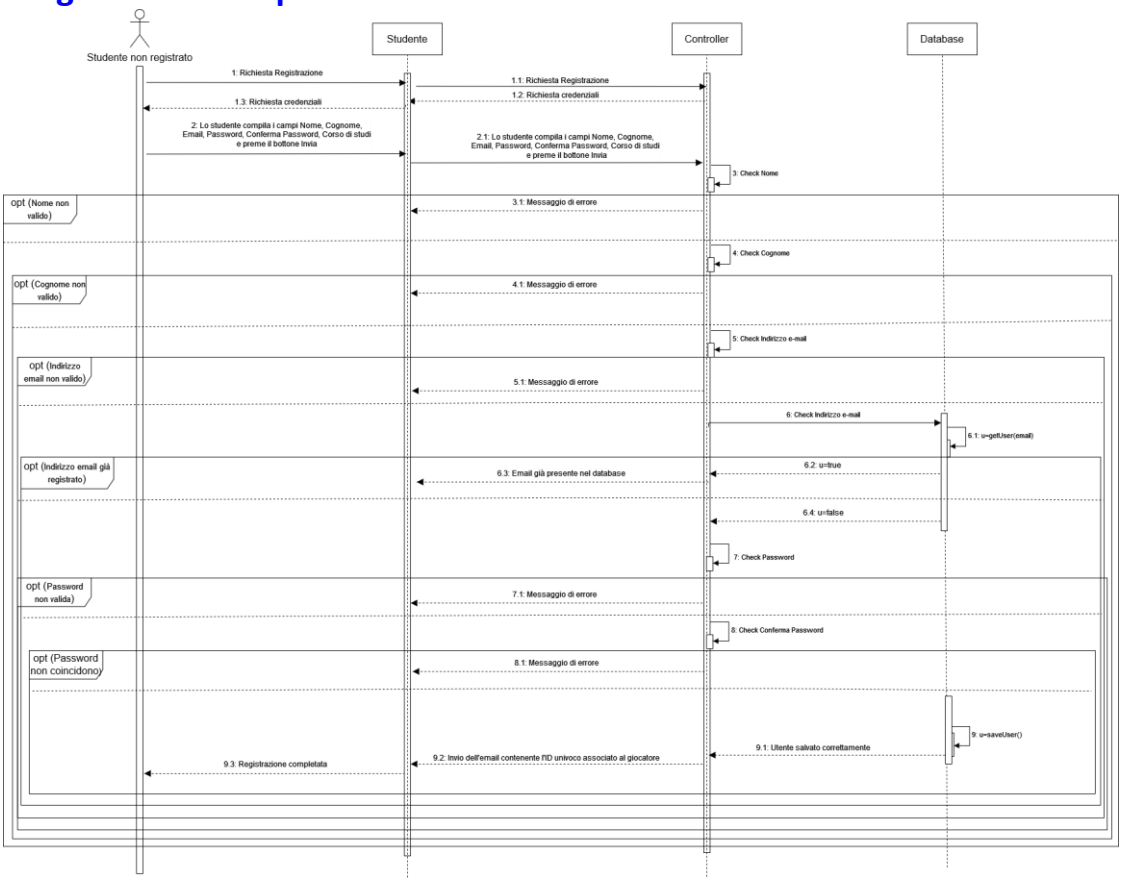

Figura 11: Diagramma di sequenza dell'operazione di REGISTRAZIONE

3.4.2 Diagramma di sequenza LOGIN

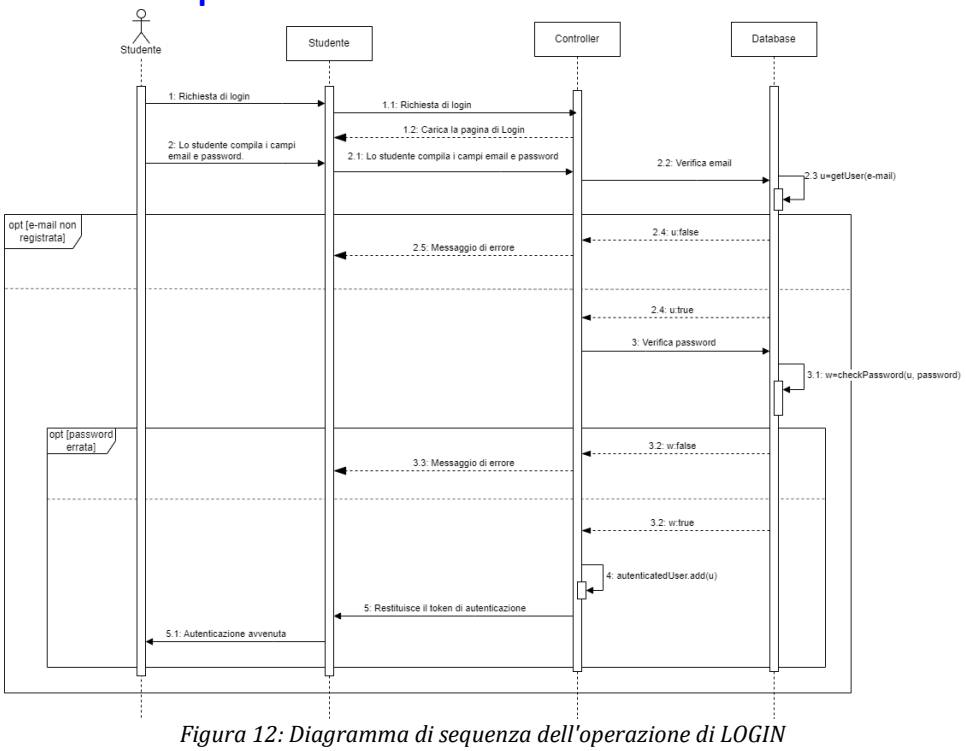

Figura 12: Diagramma di sequenza dell'operazione di LOGIN

3.4.3 Diagramma di sequenza LOGOUT

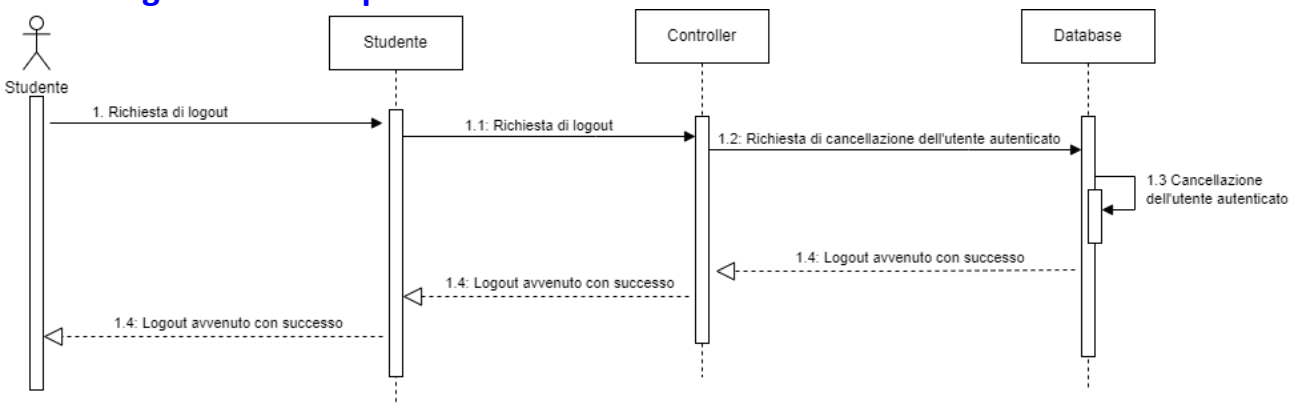

Figura 13: Diagramma di sequenza dell'operazione di LOGOUT

3.4.4 Diagramma di sequenza di RESET_PASSWORD

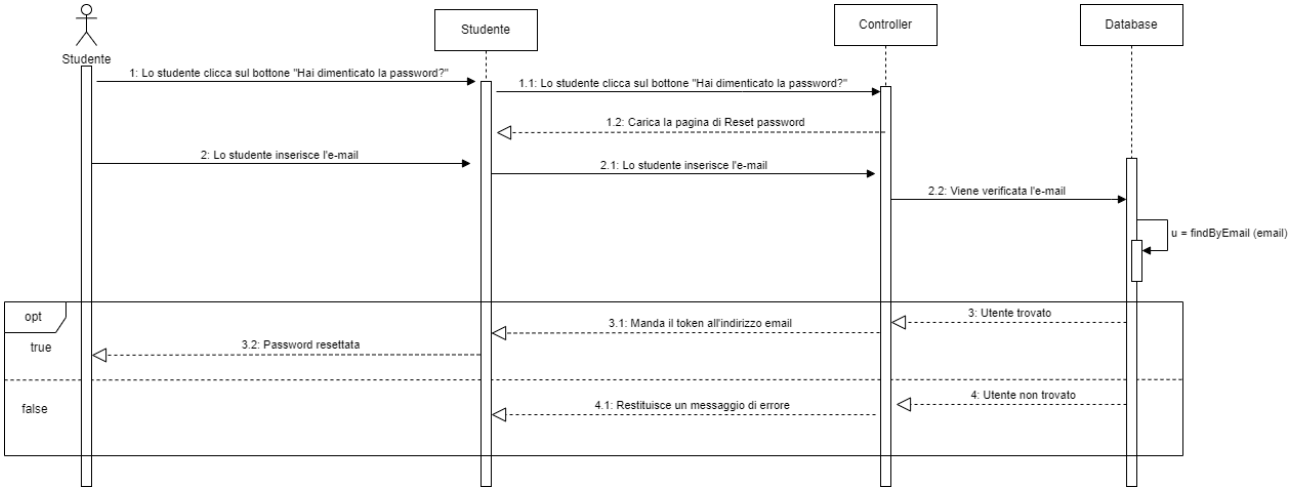

Figura 14: Diagramma di sequenza dell'operazione di RESET PASSWORD

3.4.5 Diagramma di sequenza di CHANGE_PASSWORD

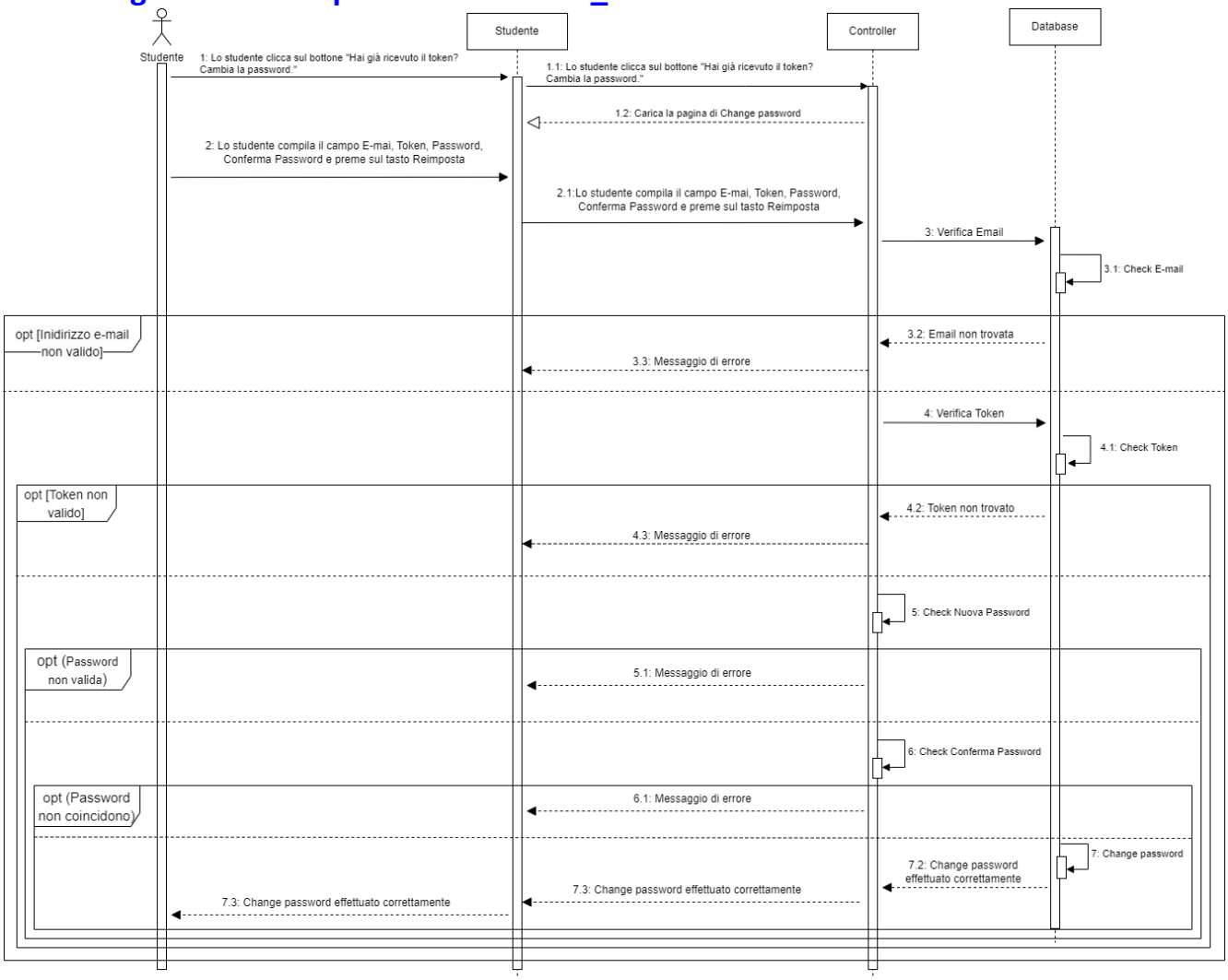

Figura 15: Diagramma di sequenza dell'operazione di CHANGE PASSWORD

4. TECNICHE DI SVILUPPO

Dopo una prima fase di progettazione, è stato impiegato del tempo per approfondire e decidere le tecnologie di sviluppo da adottare per passare alla fase di implementazione.

Il pattern architetturale scelto per l'applicazione web è MVC (Model-View-Controller), che separa le diverse funzionalità dell'applicazione in tre componenti principali: il Model, che gestisce i dati e la logica di business, la View, che visualizza i dati all'utente, e il Controller, che gestisce l'input dell'utente e coordina le interazioni tra il Model e la View. Questo approccio aiuta a mantenere il codice pulito e ben organizzato, facilitando la manutenzione e l'evoluzione dell'applicazione nel tempo.

Per il *frontend* 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.

Inoltre, è stato utilizzato Thymeleaf, un motore di template XML/HTML per applicazioni web Java, che consente di creare template visuali per le pagine web utilizzando un linguaggio di espressioni simile a quello di JSP, ma con una sintassi più semplice e intuitiva. 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.

Per il backend 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.

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.

La scelta di utilizzare Docker per la gestione del database 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. 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".

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.

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.

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.

5. DIAGRAMMA DEI COMPONENTI

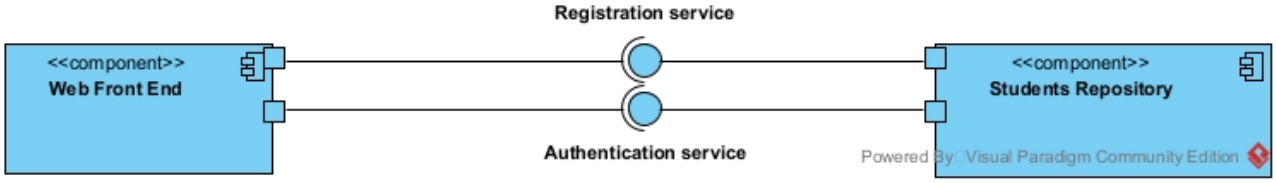

Figura 16: Diagramma dei componenti

6. 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, inclusi gli strumenti e le tecniche utilizzati per garantire un deployment efficiente ed efficace.

Data la cartella con il progetto:

1. Aprire Docker
2. Aprire in Visual Studio Code il progetto.
3. Aprire un terminale (ctrl+j).
   - -*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.
4. Per effettuare le richieste, aprire da browser la pagine tramite i path in locale (http://localhost:8080/register, http://localhost:8080/login, ecc.)
5. Per verificare la correttezza del popolamento delle tabelle del database, aprire un terminale ed eseguire i seguenti comandi:
   - -*docker exec –it nome_docker 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".
6. Utilizzare i comandi SQL per la gestione delle tabelle (SELECT, DROP, SHOW TABLES, ecc.)

7. API

Le API (Application Programming Interface) sono un insieme di strumenti che consentono agli utenti del nostro sistema di interagire con esso in modo programmato, tramite scambio di dati e richieste. Le API sono fondamentali per la

nostra architettura software, poiché consentono una maggiore flessibilità e scalabilità del sistema.
Il nostro 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'API1 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.

Di seguito sono riportati alcuni screenshot delle richieste API effettuate con Postman.

7.1 REGISTRAZIONE

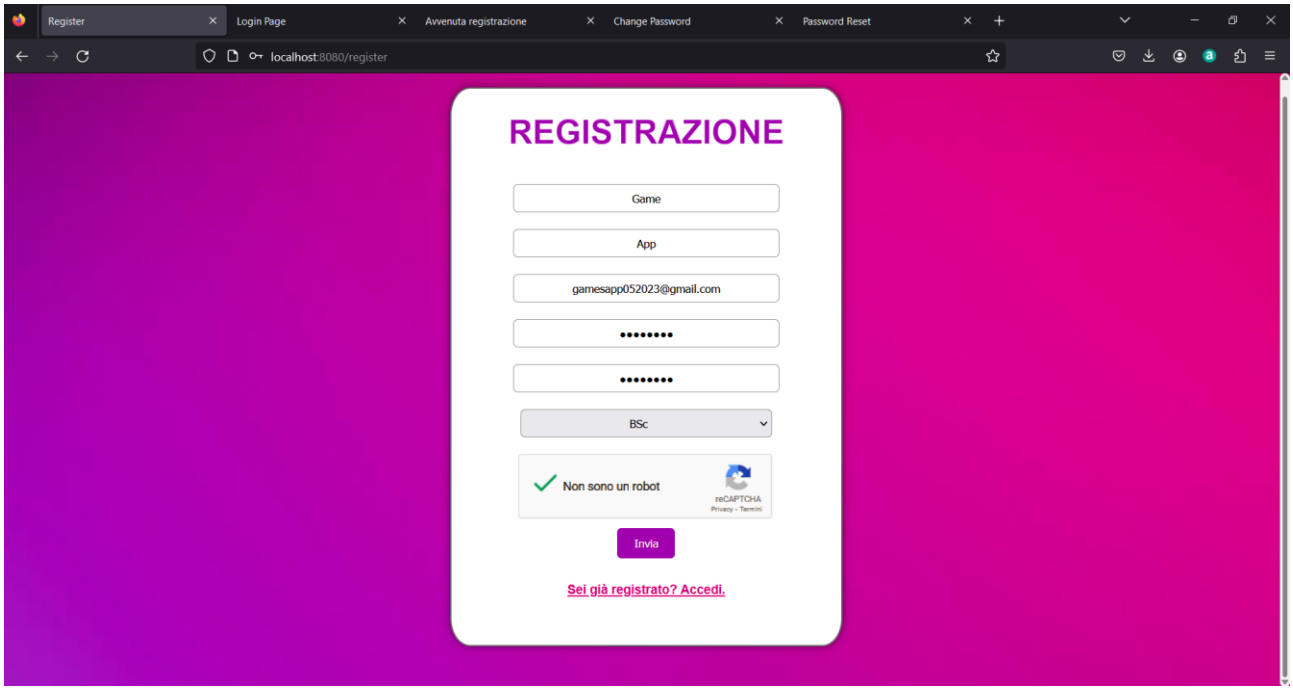

Figura 17: Front-end dell'operazione di REGISTRAZIONE.

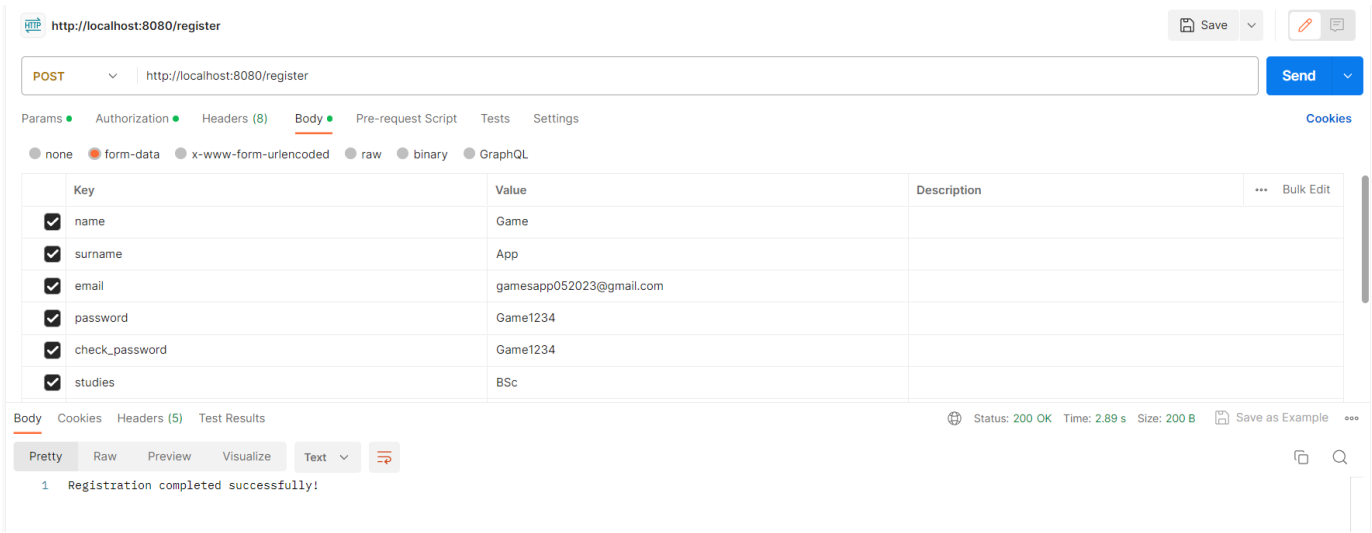

Figura 18: Esecuzione della post REGISTER.

7.2 LOGIN

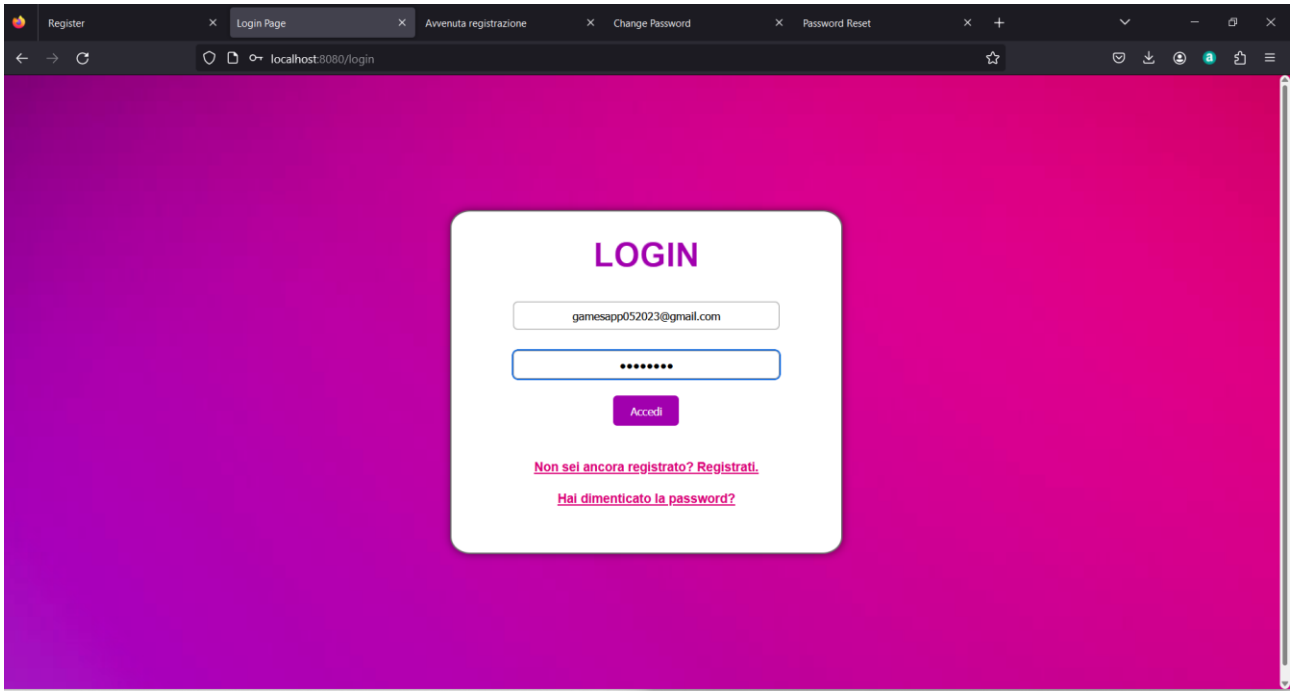

Figura 19: Front-end dell'operazione di LOGIN.

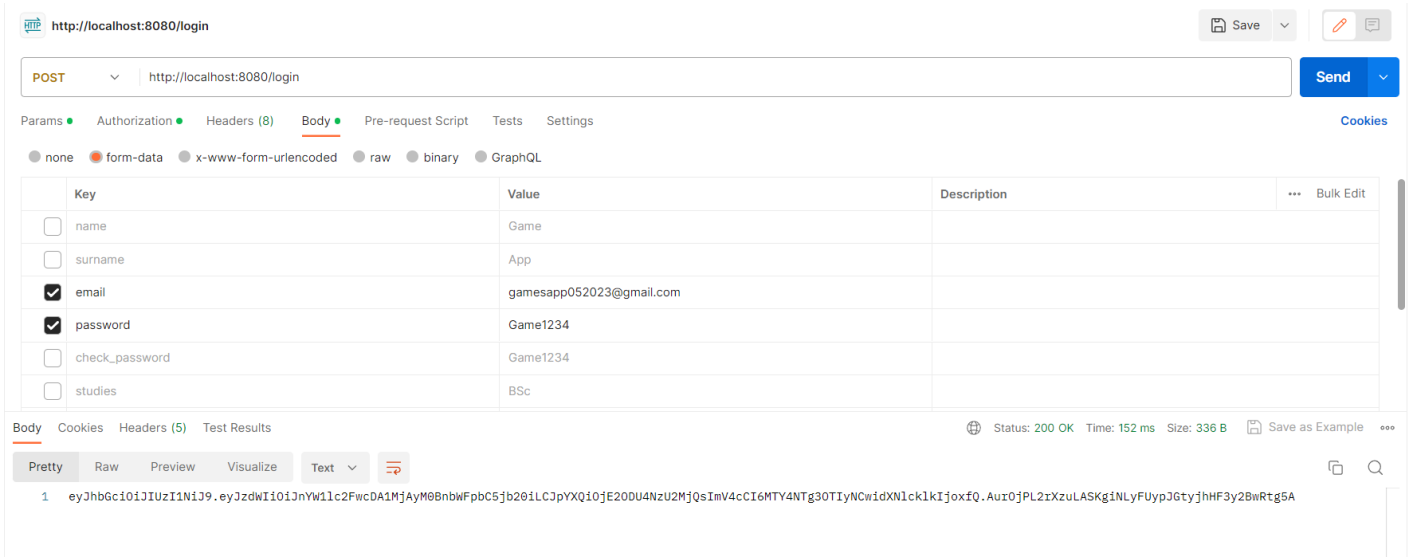

Figura 20: Esecuzione della post LOGIN.

7.3 RESET PASSWORD

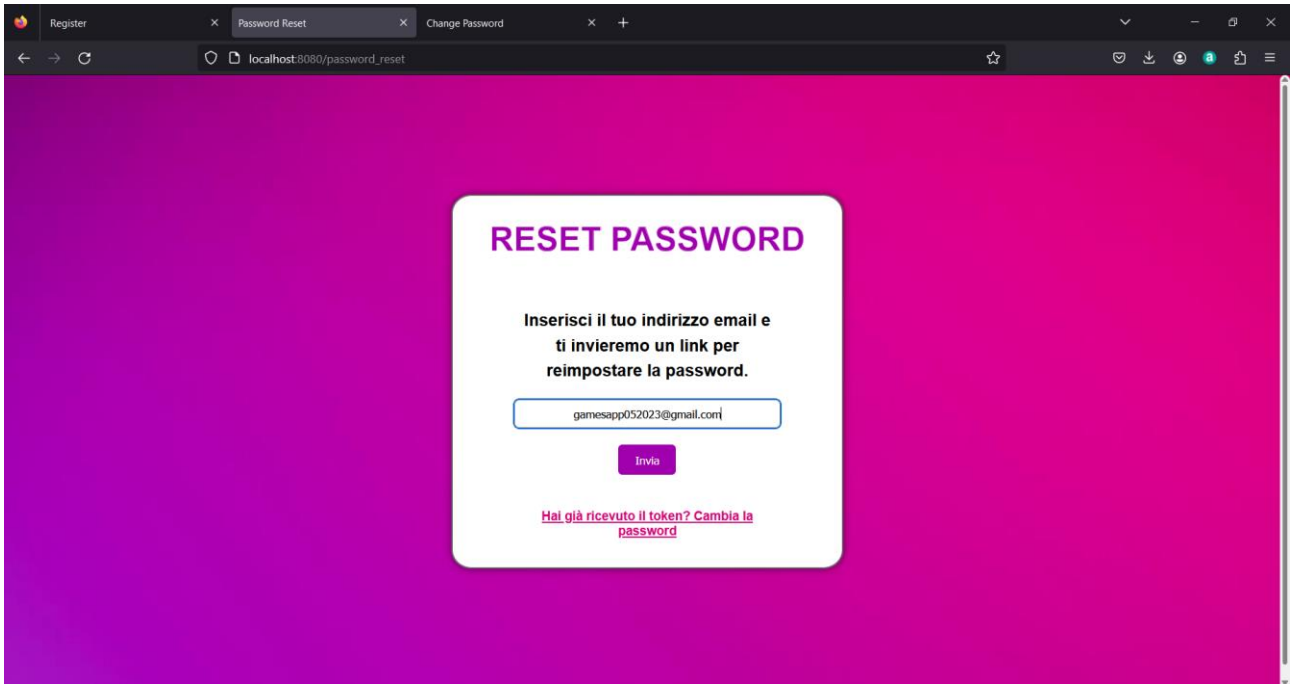

Figura 21: Front-end dell'operazione di RESET PASSWORD.

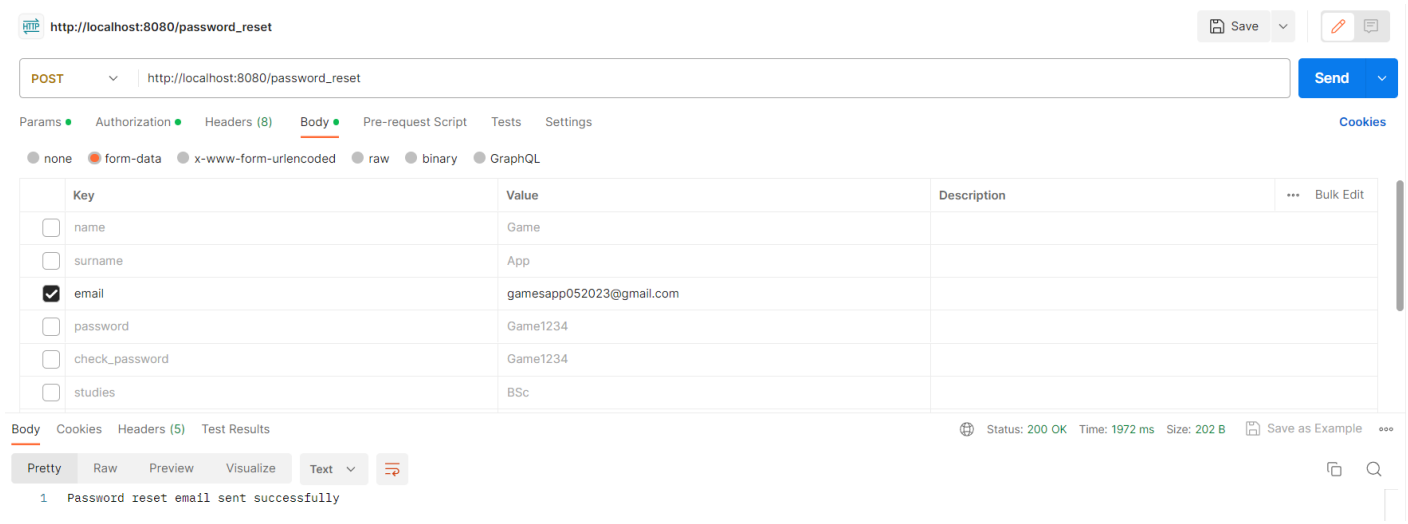

Figura 22: Esecuzione della post RESET PASSWORD.

7.4 CHANGE PASSWORD

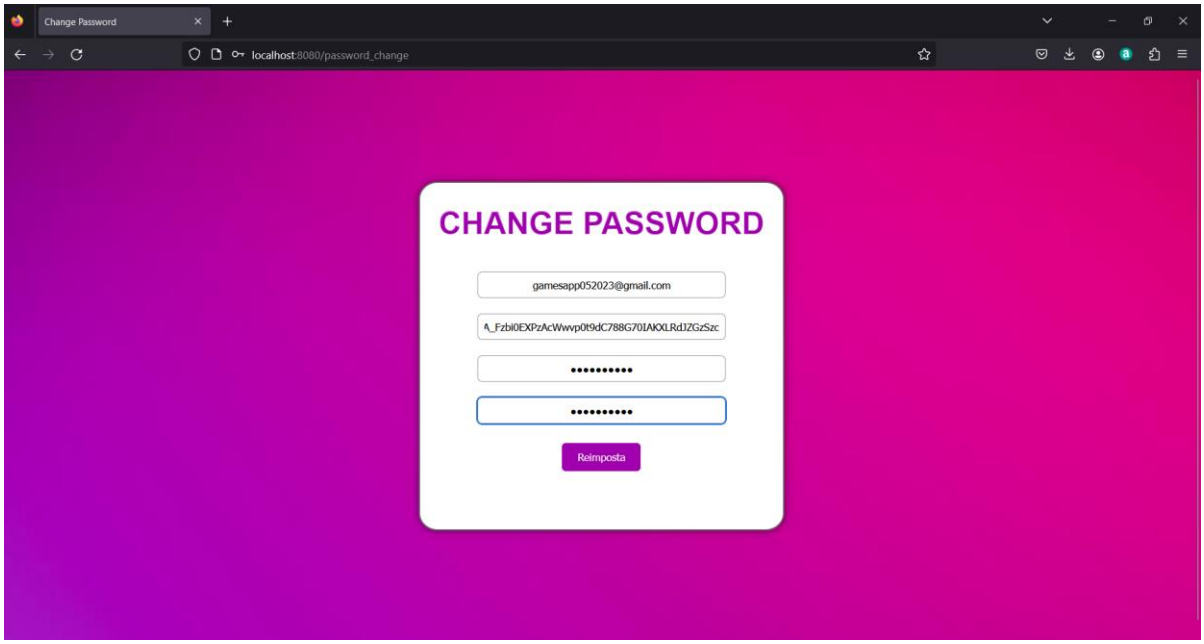

Figura 23: Front-end dell'operazione di CHANGE PASSWORD.

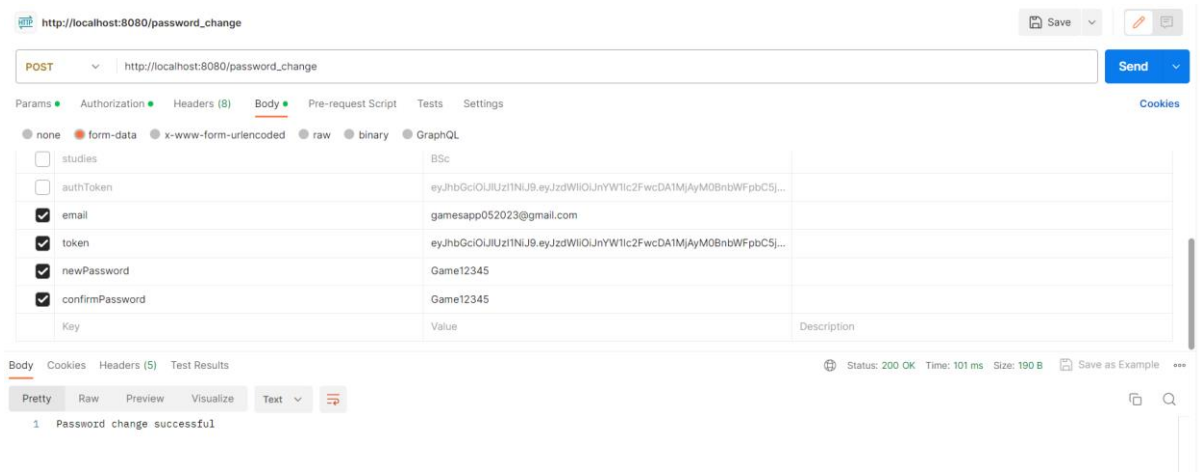

Figura 24: Esecuzione della post di CHANGE PASSWORD.

7.5 LOGOUT

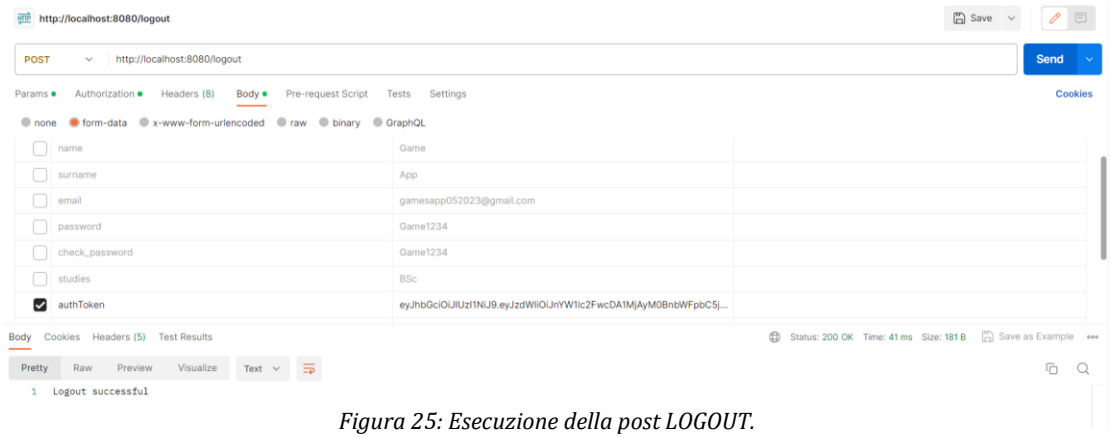

Figura 25: Esecuzione della post LOGOUT.