

Task 2-3

Studenti:

Iole Morabito M63001448 i.morabito@studenti.unina.it

Marika Sasso M63001438 marik.sasso@studenti.unina.it

Francesca Terracciano M63001550 francesca.terracciano@studenti.unina.it

Doriana Traetto M63001416 d.traetto@studenti.unina.it

Indice

1. METODOLOGIA DI LAVORO	4
2. SPECIFICA DEI REQUISITI.....	5
2.1 TASK T2.....	5
2.2 TASK T3.....	5
GLOSSARIO	5
3. REQUISITI	6
3.1 REQUISITI INFORMALI.....	6
3.2 REQUISITI FUNZIONALI.....	6
3.3 REQUISITI SUI DATI.....	7
3.4 REQUISITI NON FUNZIONALI.....	7
4. MODELLI DEI CASI D'USO	8
4.1 USE CASE DIAGRAM.....	8
4.2 SCENARI	9
5. USER STORIES	12
5.1 CRITERI DI ACCETTAZIONE.....	12
6. SYSTEM DOMAIN MODEL.....	13
7. MODELLAZIONE DEI DATI.....	14
8. COMPONENT DIAGRAM.....	15
9. CONTEXT DIAGRAM	16
10. ACTIVITY DIAGRAM.....	17
10.1 DIAGRAMMA DI ATTIVITÀ REGISTER.....	17
10.2 DIAGRAMMA DI ATTIVITÀ LOGIN	17
10.3 DIAGRAMMA DI ATTIVITÀ LOGOUT.....	18
10.4 DIAGRAMMA DI ATTIVITÀ RESET_PASSWORD	18
10.5 DIAGRAMMA DI ATTIVITÀ CHANGE_PASSWORD	18
11. SEQUENCE DIAGRAM.....	19
11.1 DIAGRAMMA DI SEQUENZA REGISTER.....	19
11.2 DIAGRAMMA DI SEQUENZA LOGIN.....	20

11.3 DIAGRAMMA DI SEQUENZA LOGOUT	20
11.4 DIAGRAMMA DI SEQUENZA DI RESET_PASSWORD	21
11.5 DIAGRAMMA DI SEQUENZA DI CHANGE_PASSWORD	21
12. TECNICHE DI SVILUPPO	22
13. PACKAGE DIAGRAM	24
14. DEPLOYMENT	25
14.1 DEPLOYMENT DIAGRAM	25
15. API	26
15.1 REGISTRAZIONE	27
15.2 LOGIN.....	28
15.3 RESET PASSWORD	29
15.4 CHANGE PASSWORD.....	30
15.5 LOGOUT	31
15.6 GET_ID	31
16. TESTING.....	32

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.

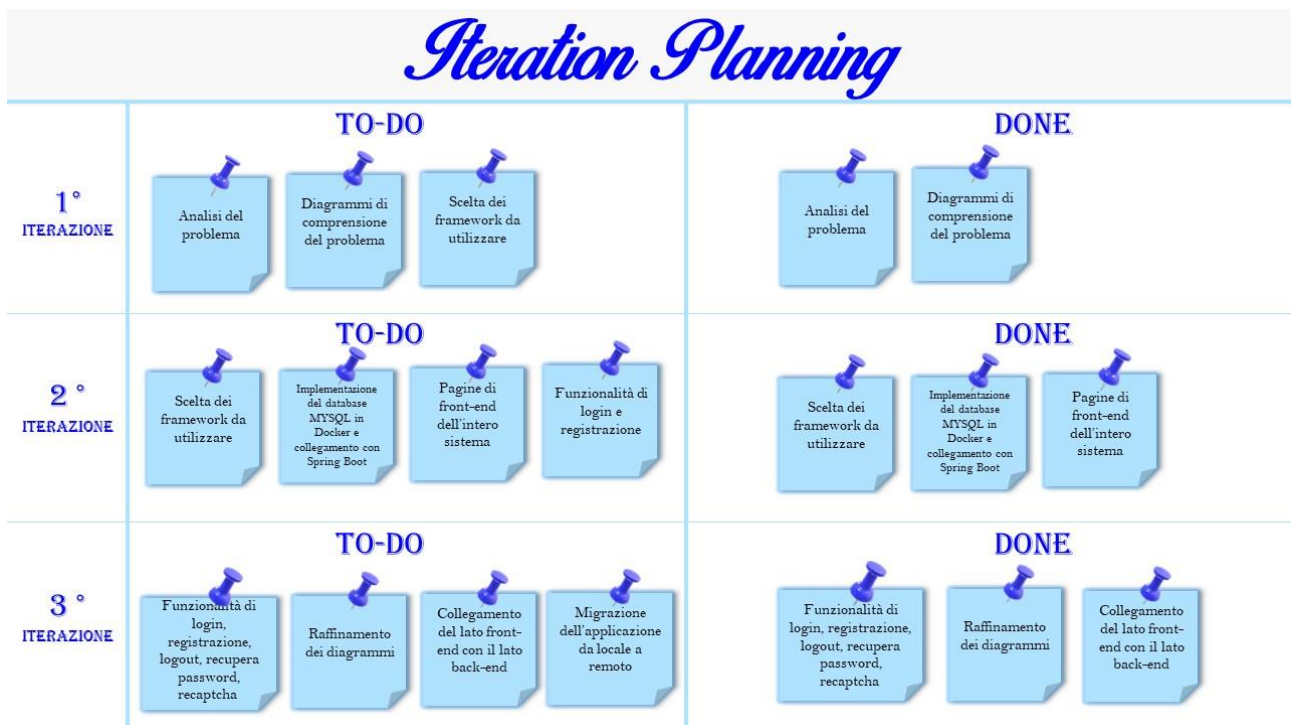


Figura 1: Iteration Planning.

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, **autenticcherà** 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

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

3. 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.

3.1 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.

3.2 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.

3.3 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.

3.4 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.

4. 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: changePassword

4.1 USE CASE DIAGRAM

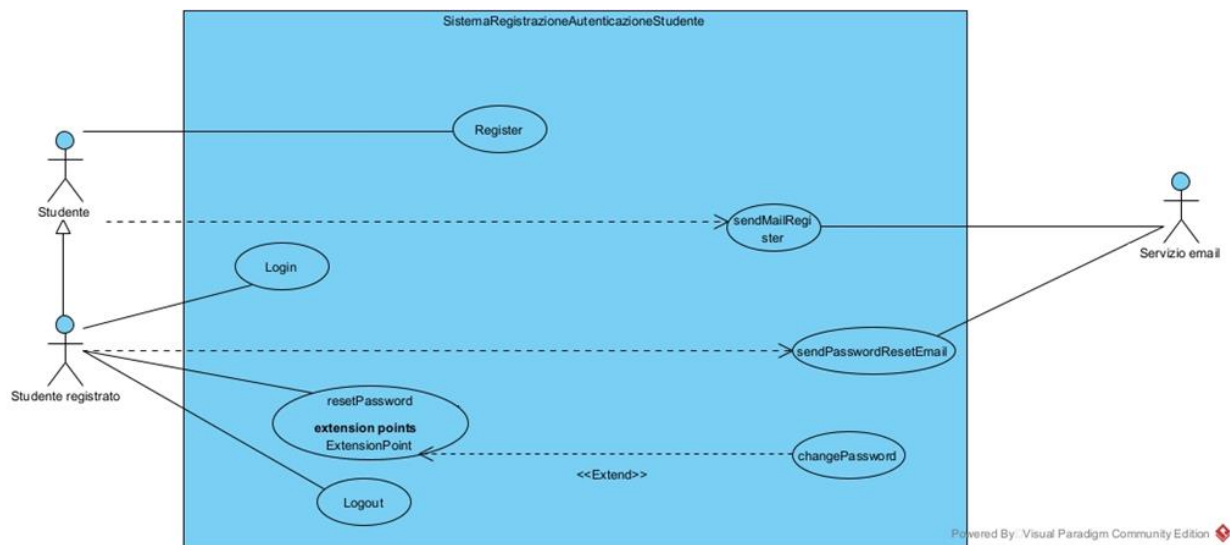


Figura 2: Diagramma dei casi d'uso

4.2 SCENARI

Caso d'uso	<i>Register</i>
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	<i>Login</i>
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	<i>SendMailRegister</i>
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	<i>SendPasswordResetEmail</i>
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	<i>Logout</i>
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	<i>Reset_password</i>
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 <i>Change_password</i>	
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	-

5. USER STORIES

User Stories

Una User Story è la definizione di un piccolo pezzo di funzionalità della soluzione dal punto di vista dell'utente. Ogni storia è espressa da una semplice frase. Dopo aver scritto le User Story, è importante classificarle in ordine di importanza per l'utente.

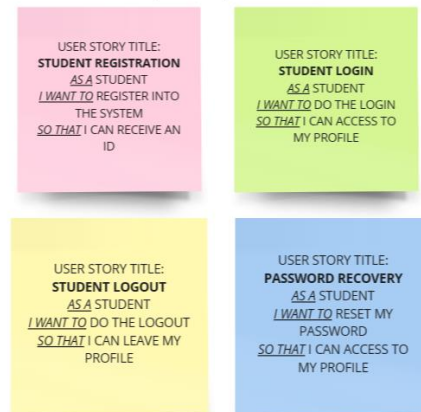


Figura 3 - User Stories

5.1 CRITERI DI ACCETTAZIONE



Figura 4 - Criteri di accettazione delle User Story precedentemente definite

6. SYSTEM DOMAIN MODEL

Il System Domain Model aiuta a comprendere e definire i concetti chiave, le entità, le relazioni e i comportamenti all'interno del dominio specifico del sistema.

Il System Domain Model rappresentato comprende quattro entità principali: User, AuthenticatedUser, LoginInformation e ChangePasswordInformation.

La relazione tra ChangePasswordInformation e AuthenticatedUser è di tipo uno-a-uno. ChangePasswordInformation contiene le informazioni relative alla richiesta di cambio password, come un token di reset che è unico per ogni richiesta dell'utente autenticato.

La relazione tra AuthenticatedUser e User è una relazione di specializzazione, in cui AuthenticatedUser è un sottotipo di User che specializza le sue funzionalità per gestire l'autenticazione nel sistema. Questa relazione consente l'accesso alle informazioni di base dell'utente da parte dell'utente autenticato.

La relazione tra LoginInformation e AuthenticatedUser è una relazione uno-a-uno. LoginInformation è associata ad un unico AuthenticatedUser, e viceversa.

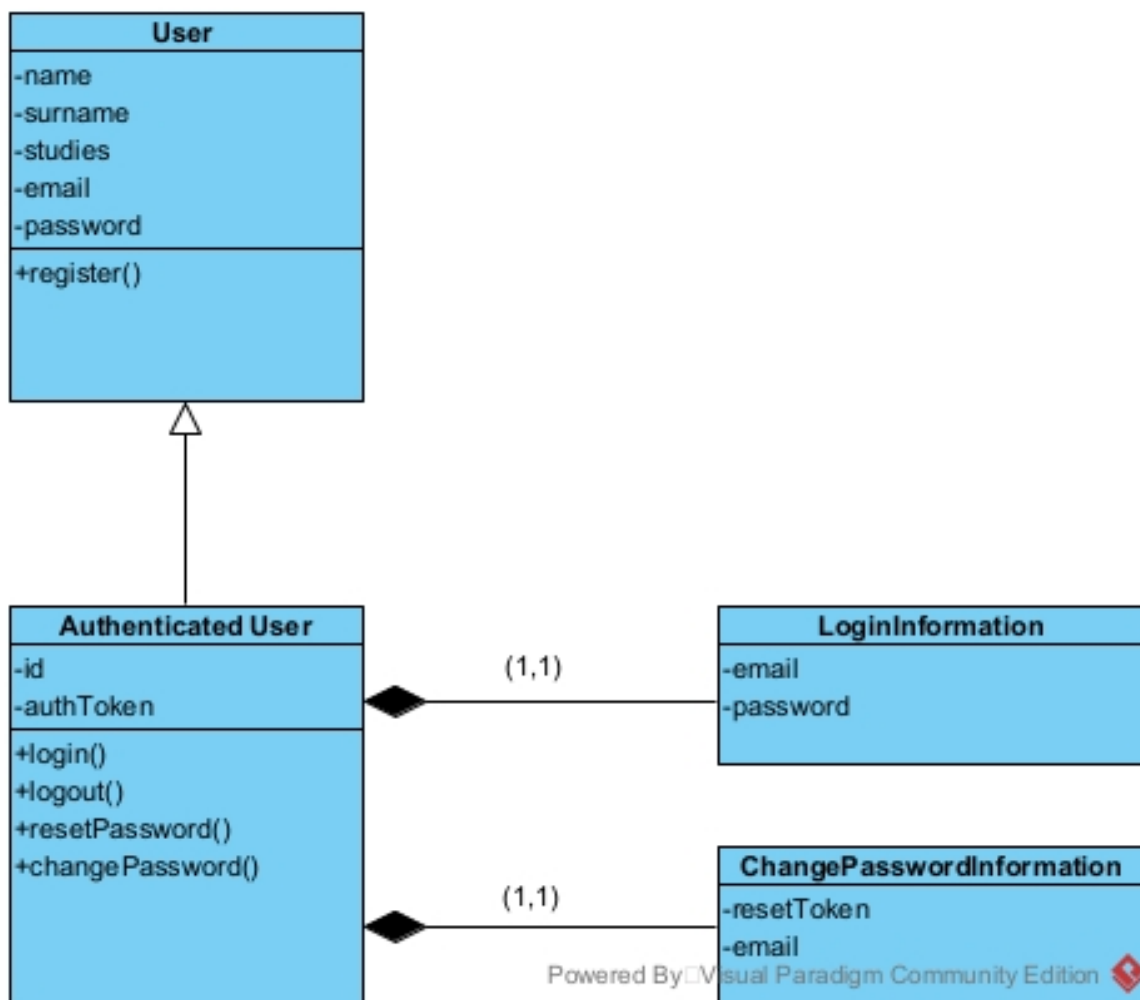


Figura 5: System domain model contenente le classi del software.

7. MODELLAZIONE DEI DATI

Il diagramma di modellazione dei dati è uno strumento visuale per rappresentare la struttura dei dati.

Il database risulta essere composto da tre tabelle, ossia collezioni di dati organizzata in righe e colonne possedenti una chiave primaria.

La chiave primaria è un attributo unico che identifica ogni riga in una tabella.

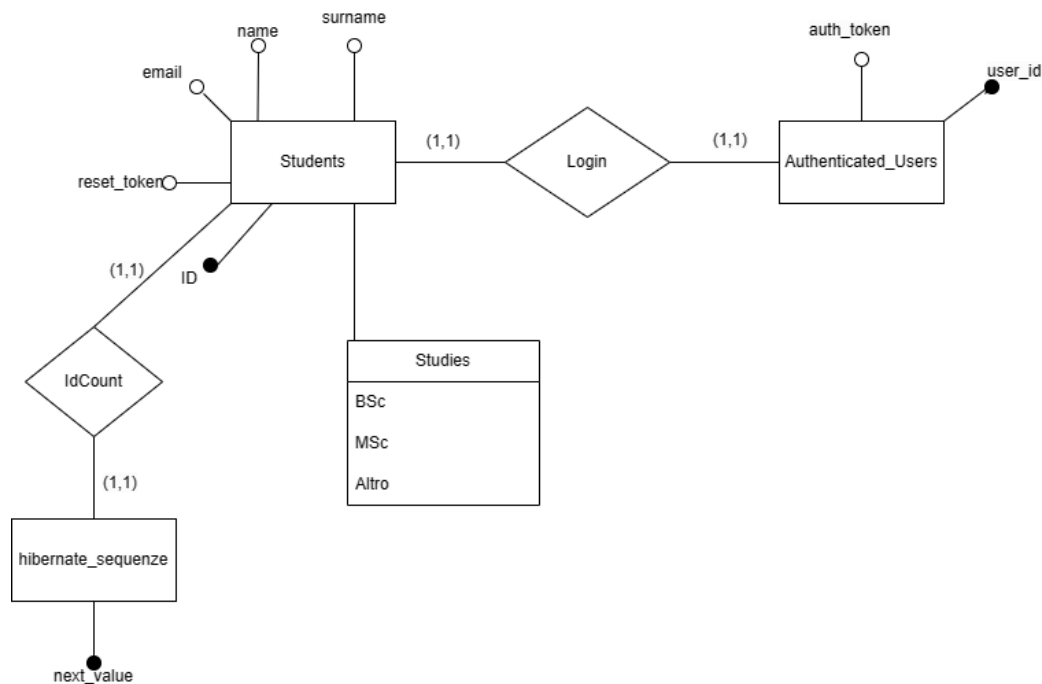


Figura 6 - Modello dei dati del sistema

8. COMPONENT DIAGRAM

Il diagramma dei componenti viene utilizzato per descrivere l'architettura software di un sistema, esso mostra i componenti del sistema e le relazioni tra di essi.

Nel diagramma dei componenti sottostante si ha un componente web front-end che richiede diverse interfacce di servizio, tra cui RegistrationService, AuthenticationService, ResetPasswordService e ChangePasswordService. Il web front-end comunica con un altro componente chiamato Docker, che offre queste interfacce di servizio, tutto è visibile tramite la notazione a lollipop.

Il componente Docker ha due sottocomponenti: docker-app, che gestisce la logica dell'applicazione e docker-db, che gestisce l'accesso ai dati.

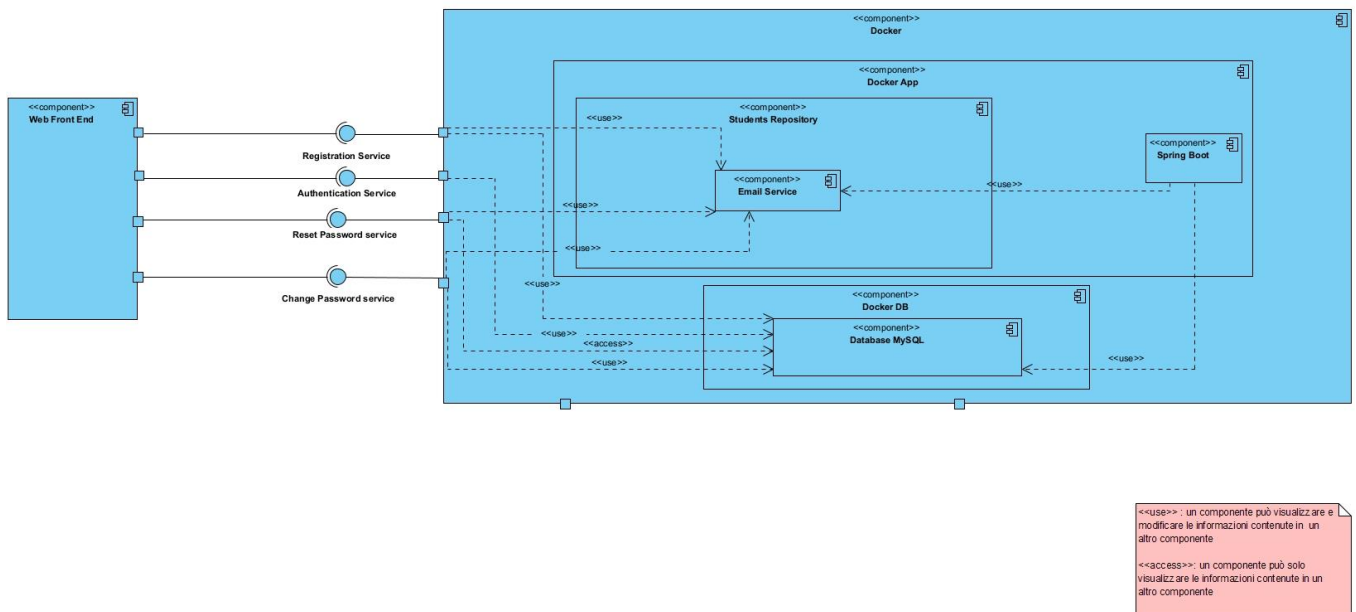


Figura 7: Diagramma dei componenti

9. CONTEXT DIAGRAM

Il context diagram è un ulteriore diagramma ad alto livello che rappresenta il contesto operativo del sistema e mette in evidenza le interazioni con gli attori esterni. Fornisce, dunque, una visione panoramica dell'architettura senza entrare nei dettagli interni del sistema.

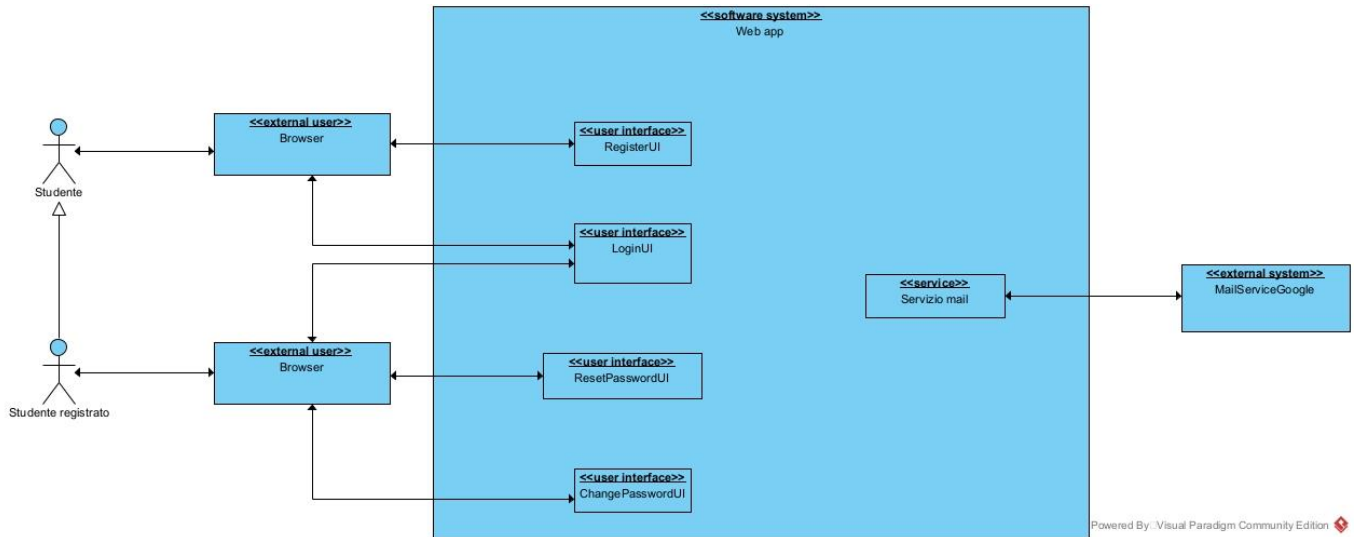


Figura 8: Diagramma di contesto

10. ACTIVITY DIAGRAM

Dopo aver definito in maniera univoca i requisiti, per comprendere meglio il flusso delle operazioni, si utilizzano i diagrammi di attività che consentono di rappresentare graficamente le attività, le azioni e le decisioni che compongono il processo.

10.1 DIAGRAMMA DI ATTIVITÀ REGISTER

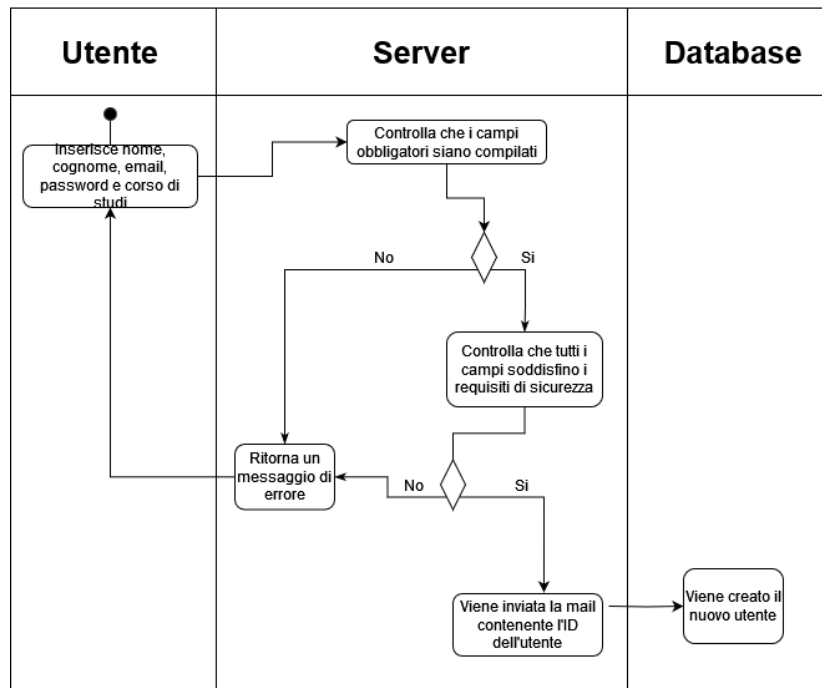


Figura 9: Diagramma di attività della REGISTRAZIONE.

10.2 DIAGRAMMA DI ATTIVITÀ LOGIN

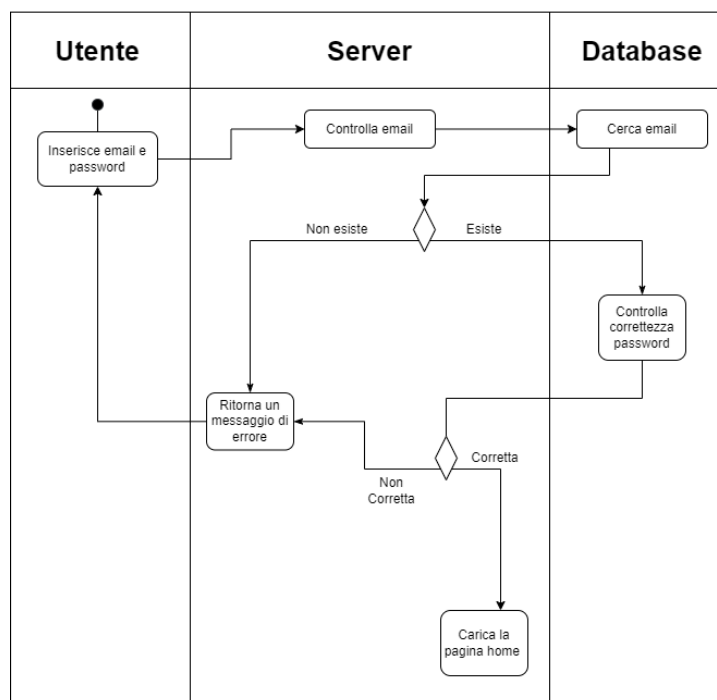


Figura 10: Diagramma di attività del LOGIN.

10.3 DIAGRAMMA DI ATTIVITÀ LOGOUT

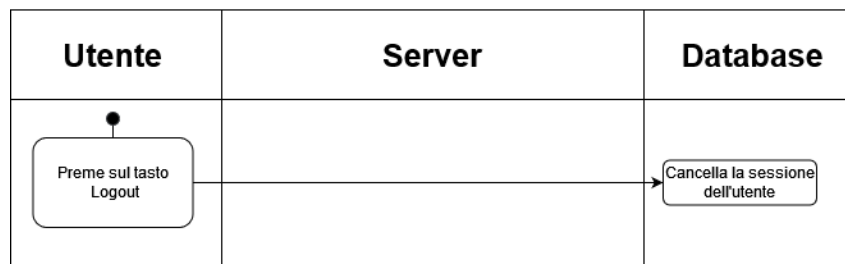


Figura 11: Diagramma di attività del LOGOUT.

10.4 DIAGRAMMA DI ATTIVITÀ RESET_PASSWORD

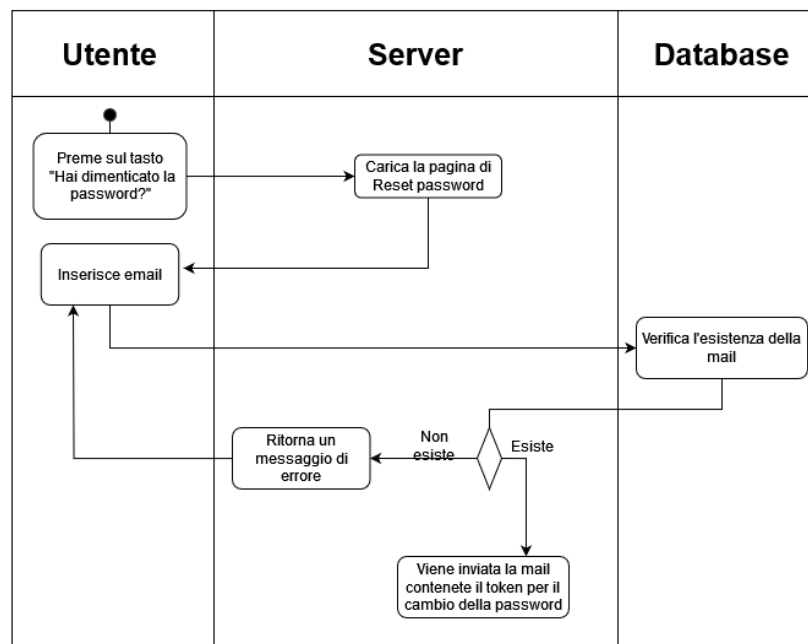


Figura 12: Diagramma di attività di RESET PASSWORD.

10.5 DIAGRAMMA DI ATTIVITÀ CHANGE_PASSWORD

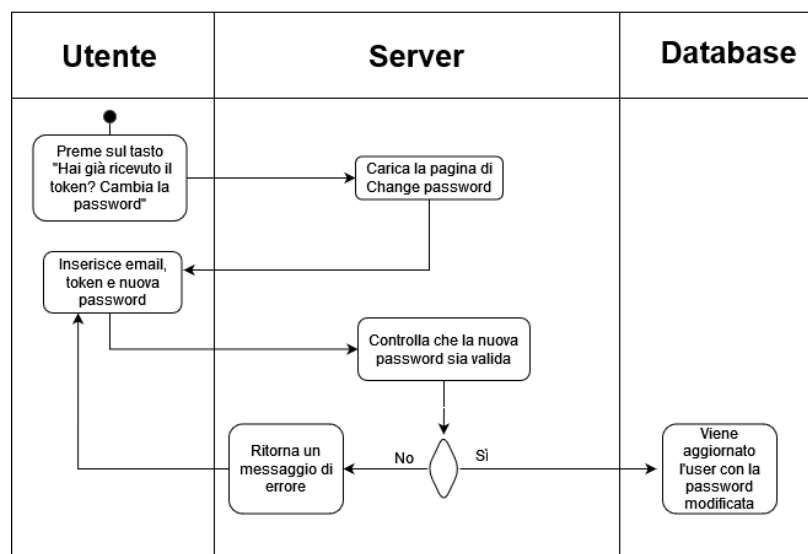


Figura 13: Diagramma di attività di CHANGE PASSWORD.

11. SEQUENCE DIAGRAM

Il diagramma di sequenza è un tipo di diagramma che rappresenta l'interazione tra gli oggetti o le entità di un sistema software in un ordine cronologico.

In un diagramma di sequenza di tipo MVC, il controller riceve la richiesta dall'utente e invia i dati al model (Database), che li elabora e li restituisce al controller. Il controller, quindi, utilizza i dati per aggiornare la view (Studente) e restituisce la view aggiornata all'utente.

11.1 DIAGRAMMA DI SEQUENZA REGISTER

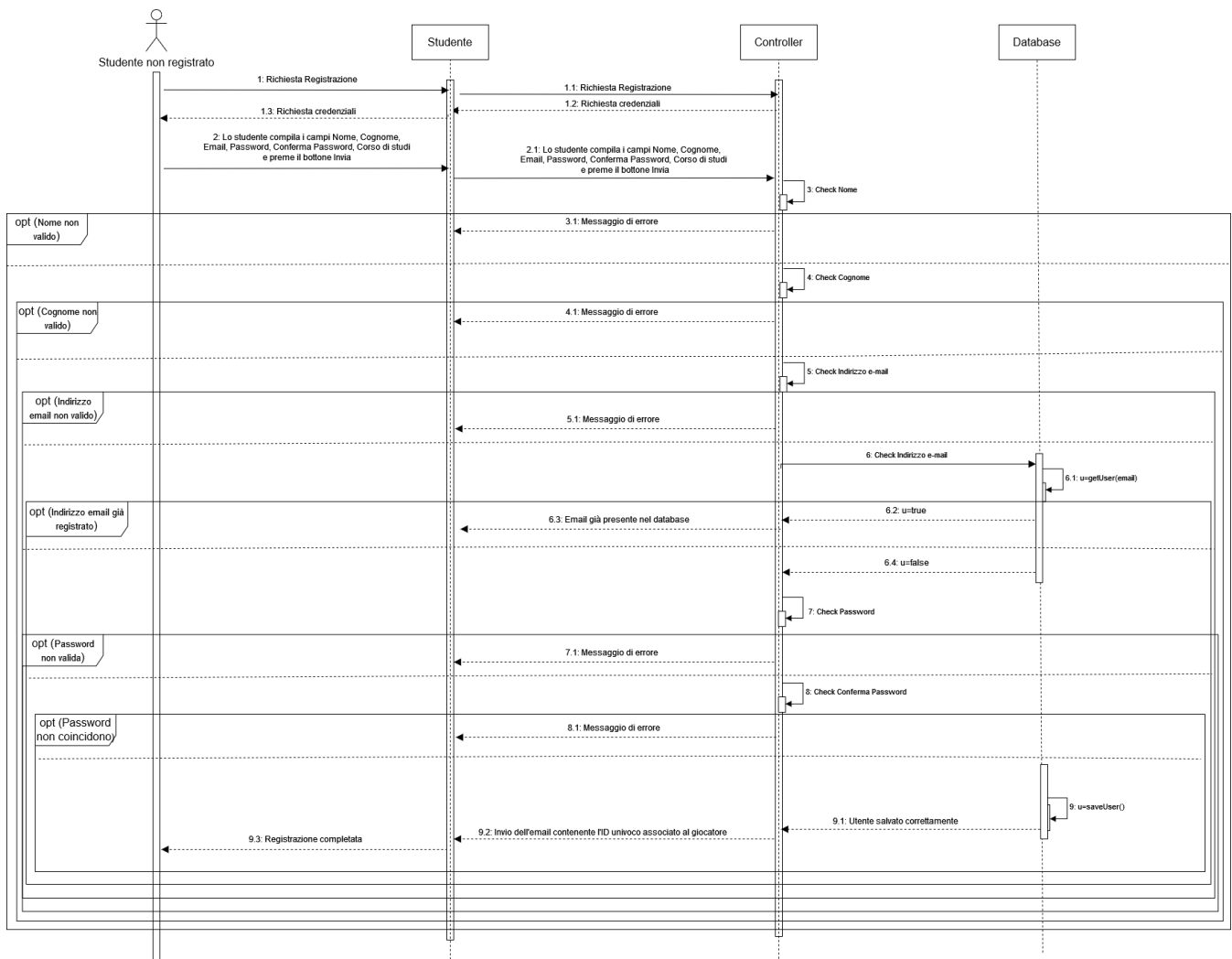


Figura 14: Diagramma di sequenza dell'operazione di REGISTRAZIONE

11.2 DIAGRAMMA DI SEQUENZA LOGIN

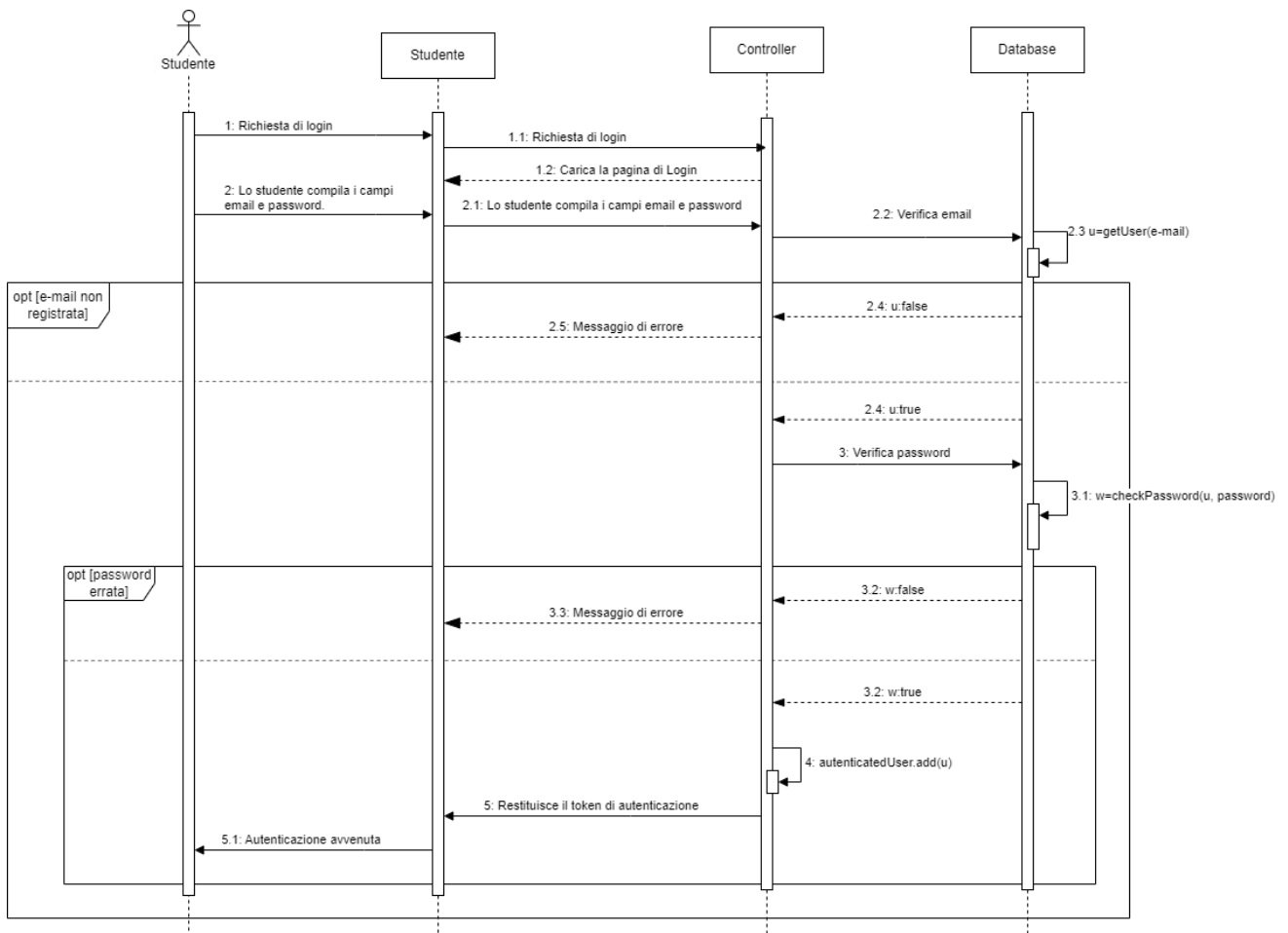


Figura 15: Diagramma di sequenza dell'operazione di LOGIN

11.3 DIAGRAMMA DI SEQUENZA LOGOUT

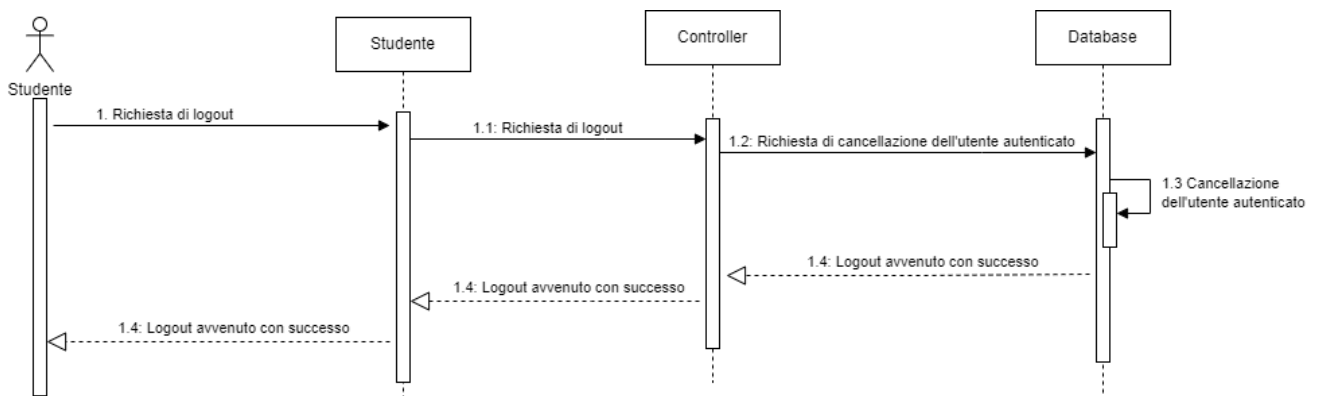


Figura 16: Diagramma di sequenza dell'operazione di LOGOUT

11.4 DIAGRAMMA DI SEQUENZA DI RESET_PASSWORD

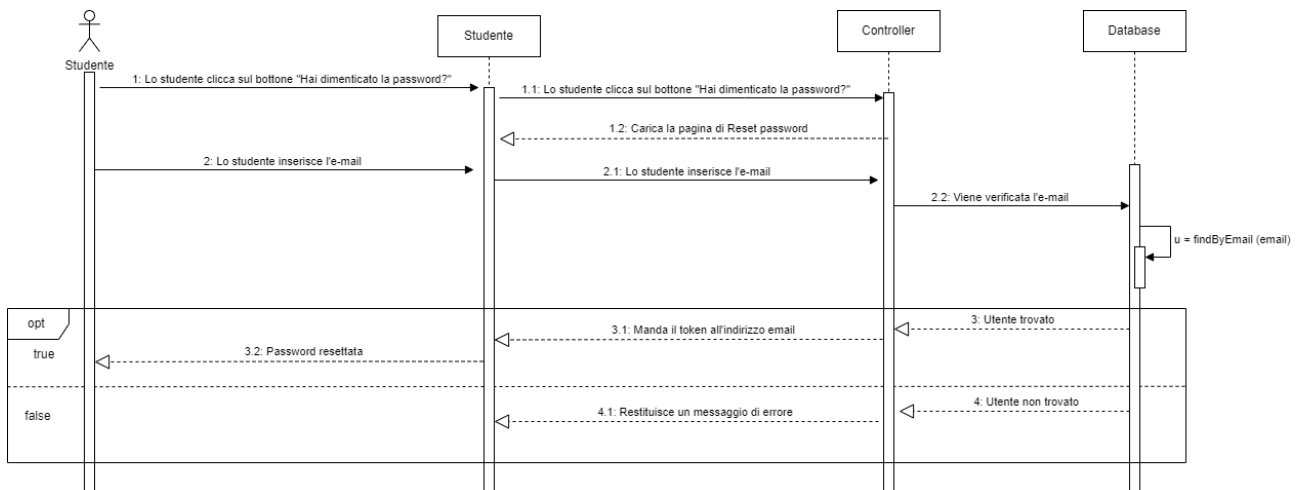


Figura 17: Diagramma di sequenza dell'operazione di RESET PASSWORD

11.5 DIAGRAMMA DI SEQUENZA DI CHANGE_PASSWORD

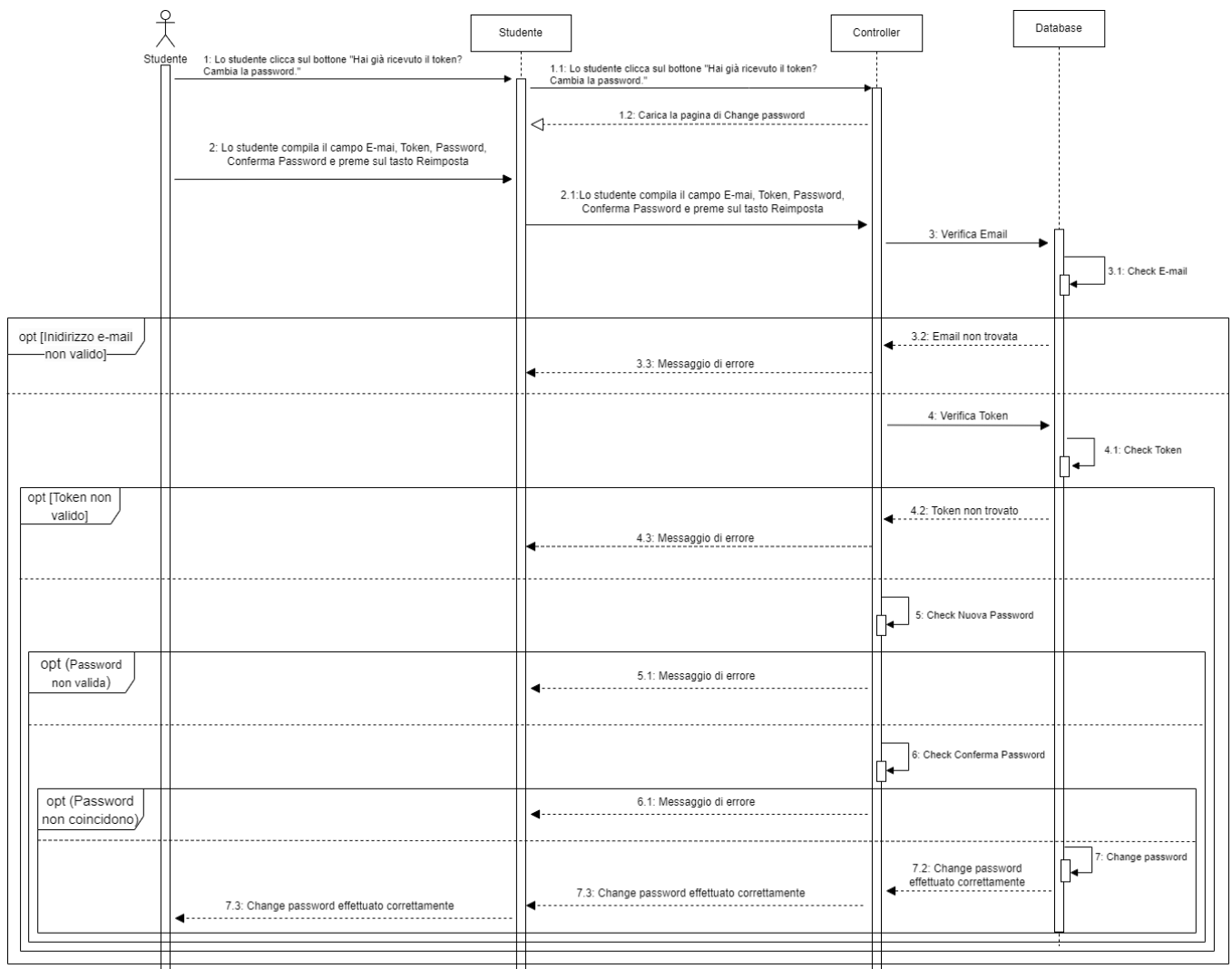


Figura 18: Diagramma di sequenza dell'operazione di CHANGE PASSWORD

12. 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.

HTML, CSS, JAVASCRIPT

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.

THYMELEAF

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. 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 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.

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.

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".

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.

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.

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.

13. PACKAGE DIAGRAM

Il diagramma dei package è una rappresentazione visiva della struttura organizzativa dei moduli o dei servizi all'interno di un sistema software.

Nel contesto dell'architettura MVC, il diagramma dei package mostra come i componenti sono organizzati nei tre strati: Model, View e Controller.

I servizi esterni sono componenti o sistemi software di terze parti che vengono integrati nel sistema principale.

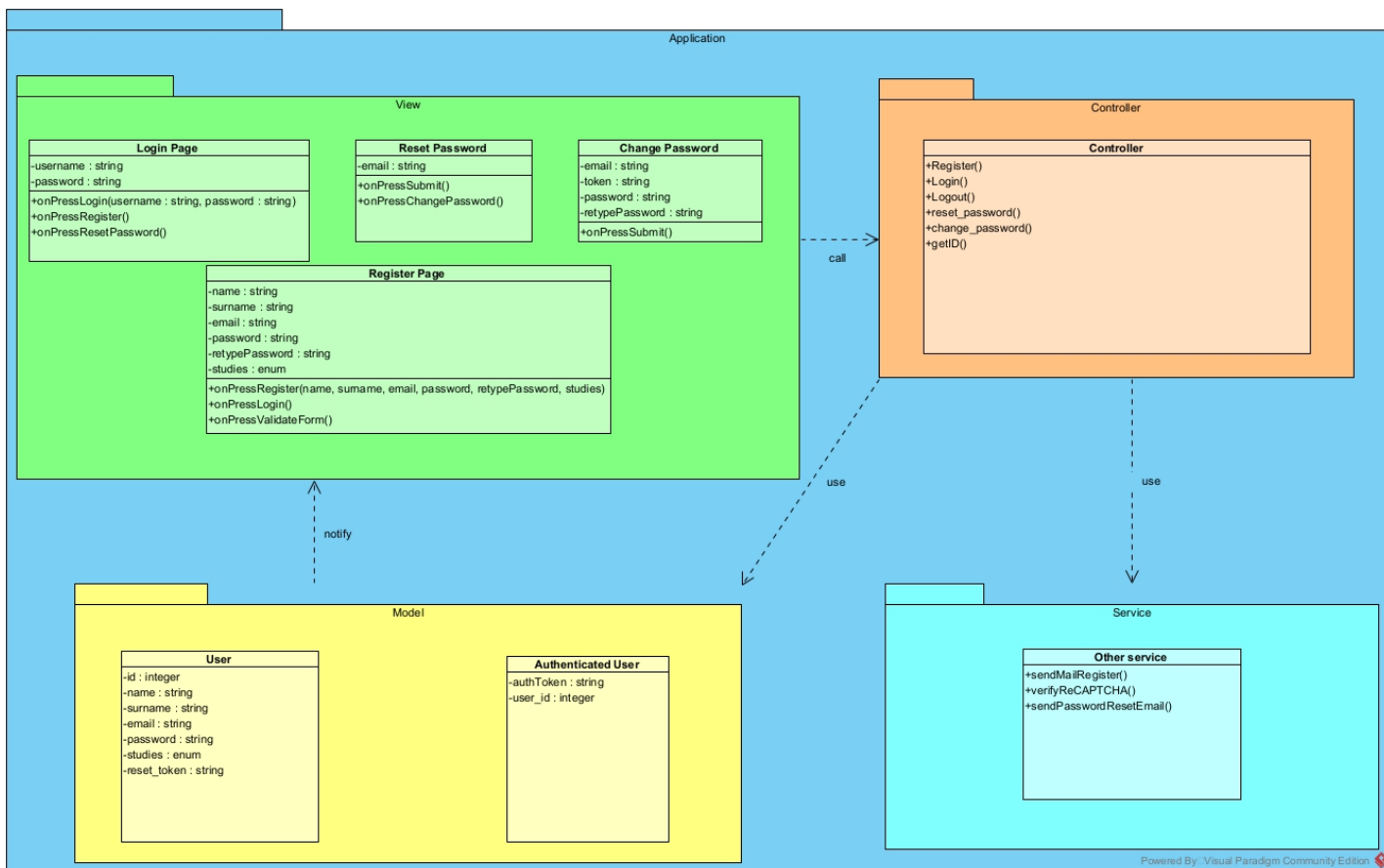


Figura 19: Package diagram basato su MVC.

14. 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.).

14.1 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.

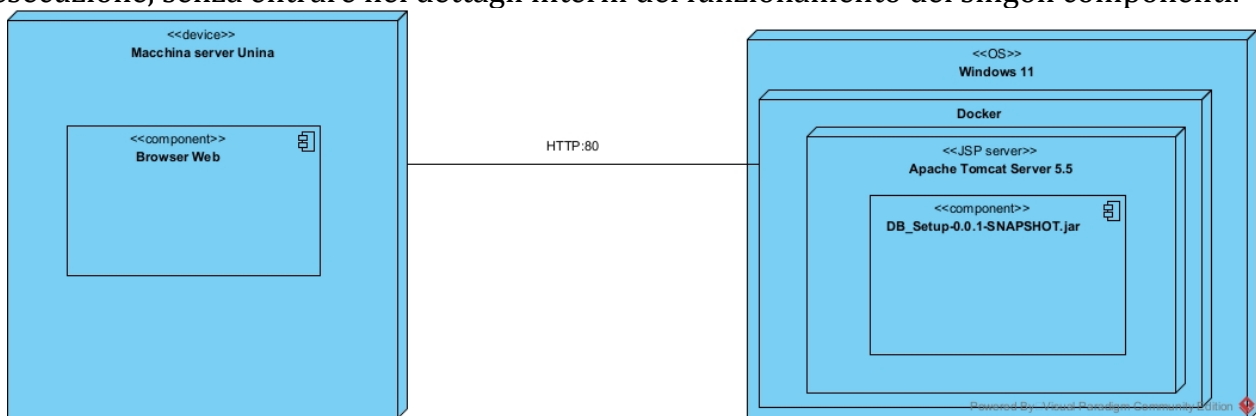


Figura 20: Deployment diagram inerente all'installazione dell'applicazione.

15. 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.

15.1 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.

Figura 21: Front-end dell'operazione di REGISTRAZIONE.

Key	Value	Description
<input checked="" type="checkbox"/> name	Game	
<input checked="" type="checkbox"/> surname	App	
<input checked="" type="checkbox"/> email	gamesapp052023@gmail.com	
<input checked="" type="checkbox"/> password	Game1234	
<input checked="" type="checkbox"/> check_password	Game1234	
<input checked="" type="checkbox"/> studies	BSc	

Figura 22: Esecuzione della post REGISTER con risposta affermativa.

15.2 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.

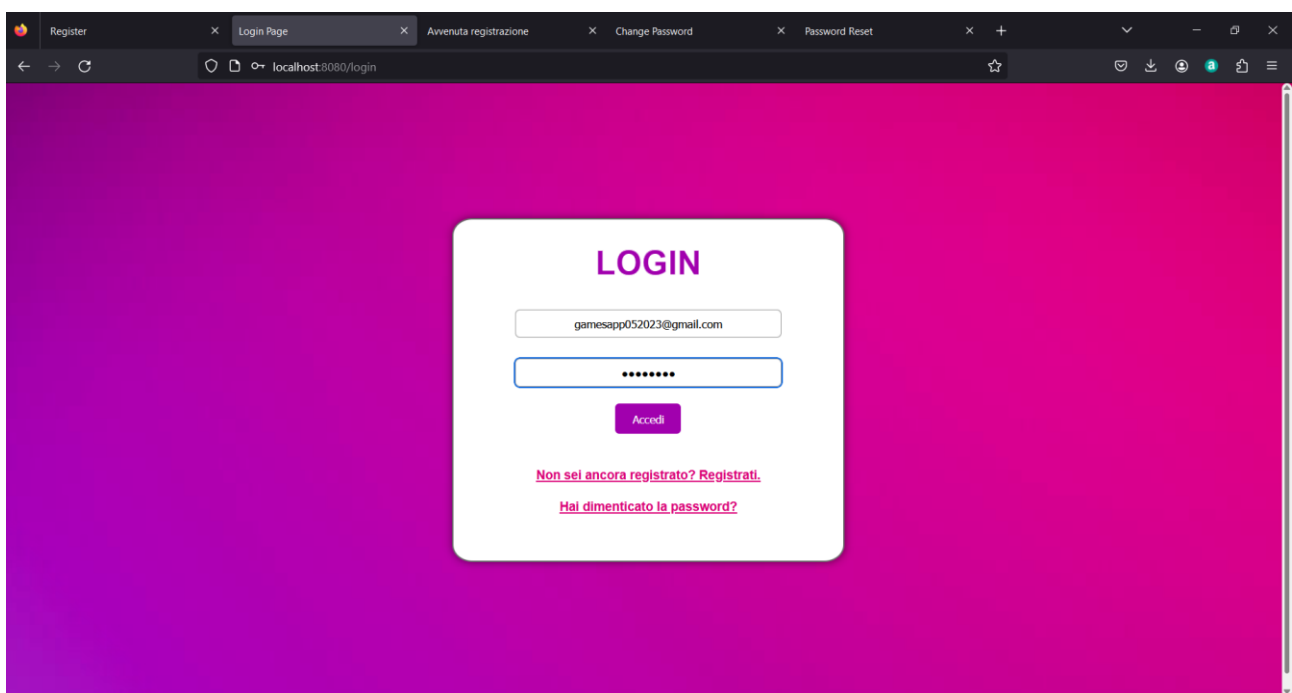


Figura 23: Front-end dell'operazione di LOGIN.

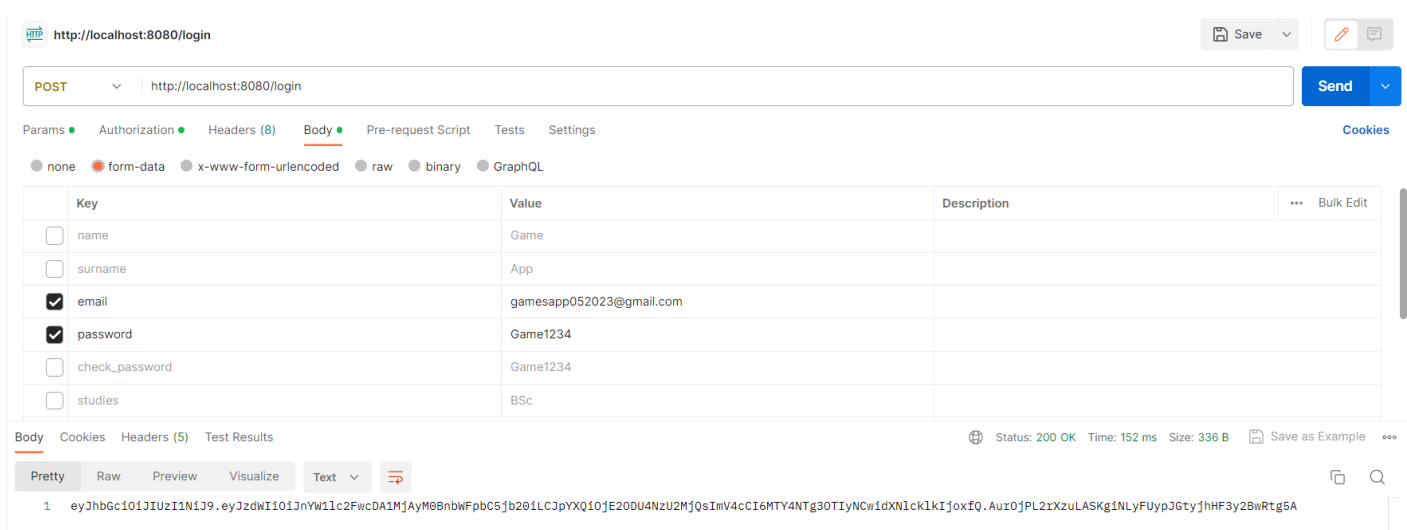


Figura 24: Esecuzione della post LOGIN con esito corretto.

15.3 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".

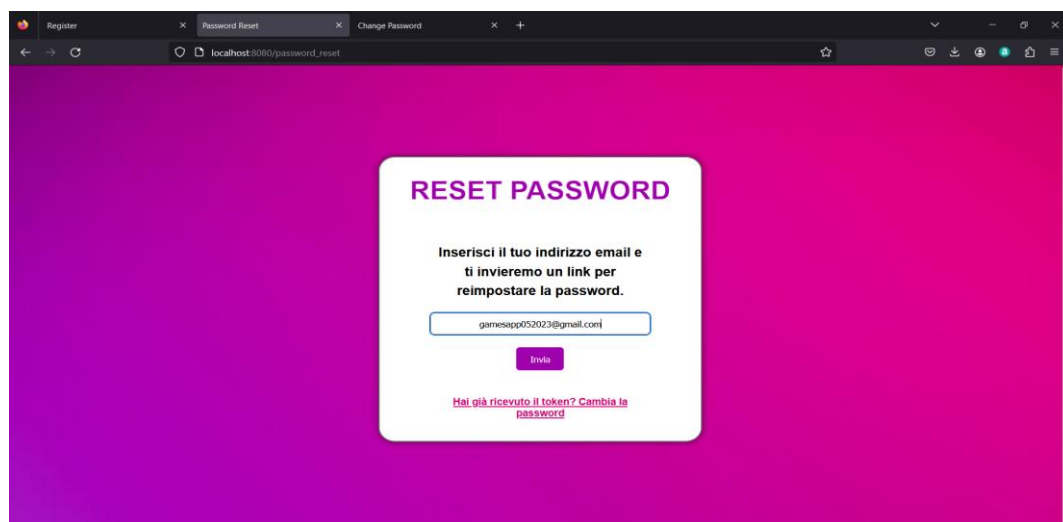


Figura 25: Front-end dell'operazione di RESET PASSWORD.

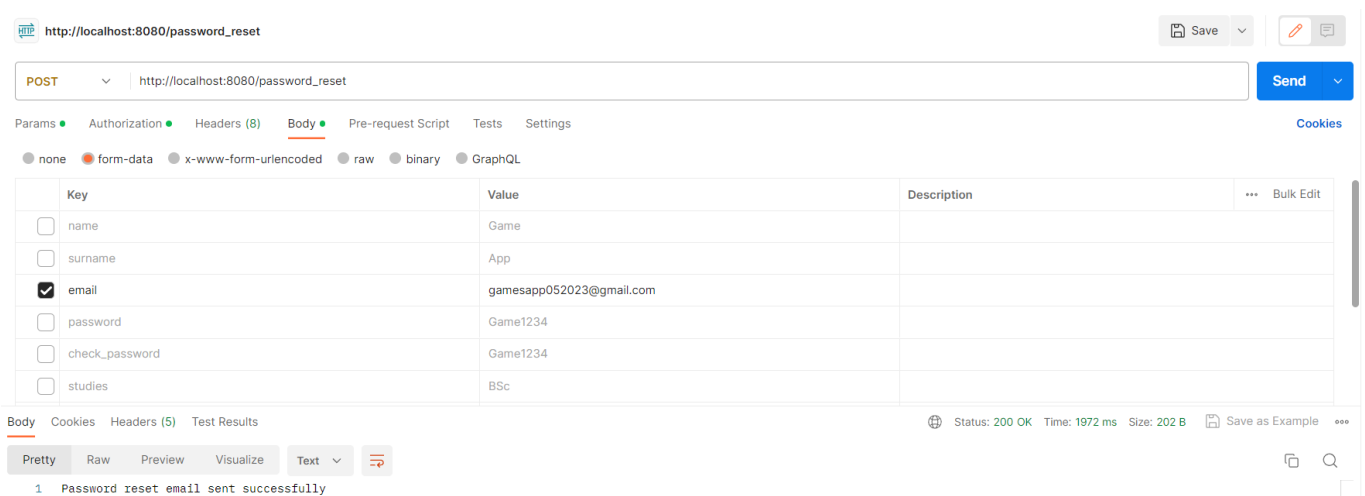


Figura 26: Esecuzione della post RESET PASSWORD.

15.4 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.

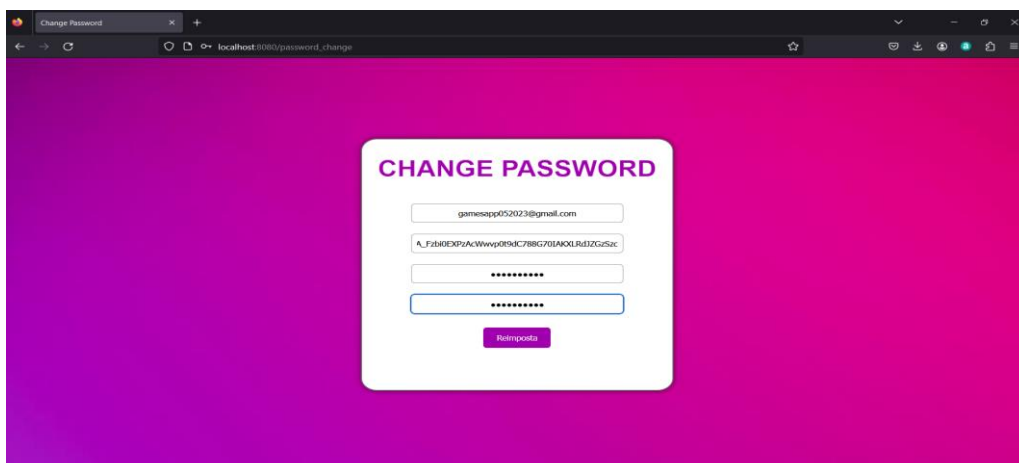
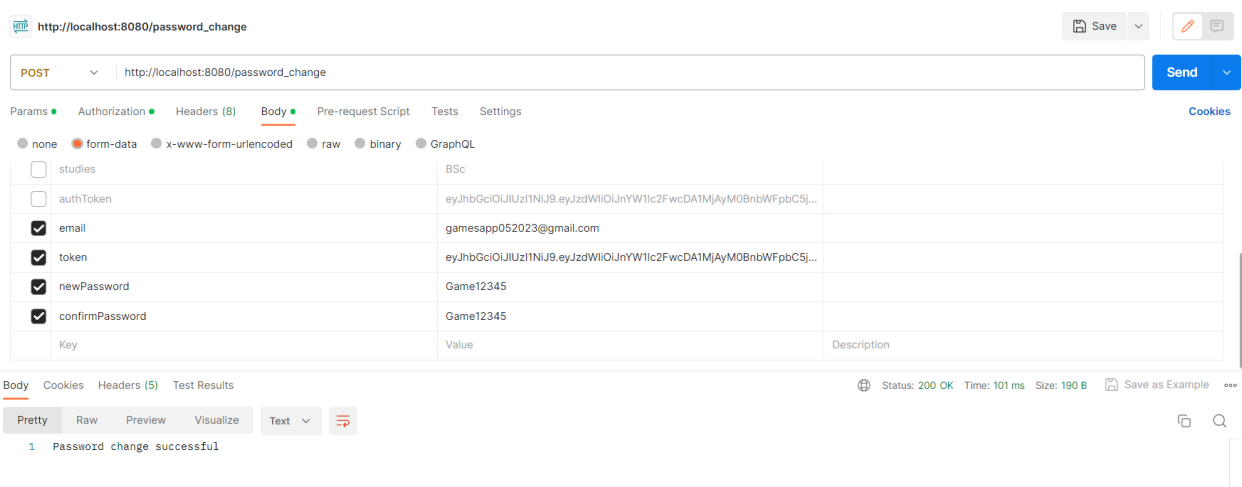


Figura 27: Front-end dell'operazione di CHANGE PASSWORD.



Key	Value	Description
email	gamesapp052023@gmail.com	
token	eyJhbGciOiJIUzI1NiU9.eyJzdWIiOiJnYW1lc2FwcDA1MjAyM0BnbWVpbC5j...	
newPassword	Game12345	
confirmPassword	Game12345	

1 Password change successful

Figura 28: Esecuzione della post di CHANGE PASSWORD.

15.5 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.

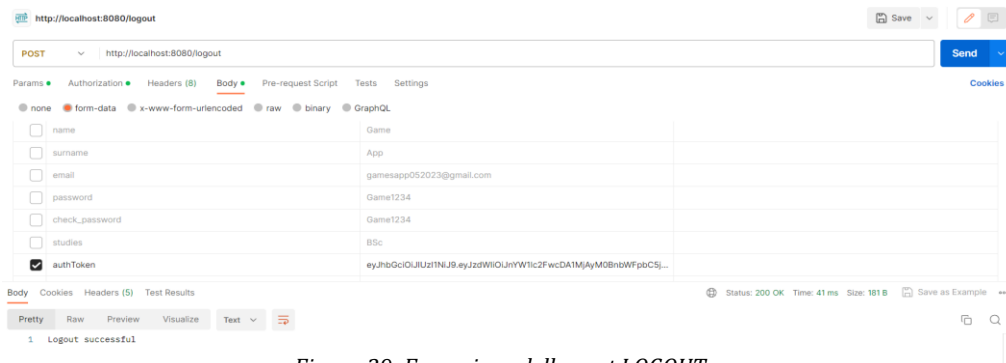


Figura 29: Esecuzione della post LOGOUT.

15.6 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 corrisponde 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.

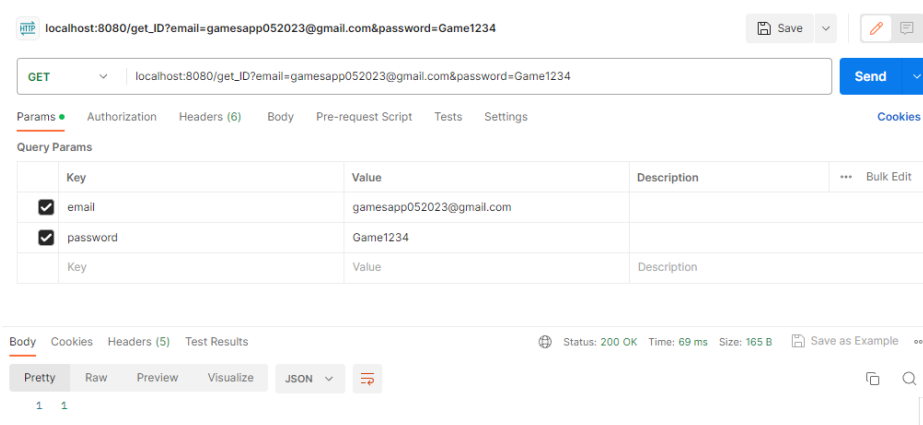


Figura 30: Esecuzione della GET_ID

16. TESTING

Test Case ID	Descrizione	Pre-condizioni	Input	Output Ottenuti	Post-condizioni Ottenute	Esito (FAIL, PASS)
1 R	Inserimento dati di input corretti.	N/A	Doriana Traetto dorianatraetto@gmail.com Dori1234 Dori1234 BSc	"Registration completed successfully!"	Utente registrato	PASS
2 R	Inserimento Nome formato non valido.	N/A	D Traetto dorianatraetto@gmail.com Dori1234 Dori1234 BSc	"Name not valid"	Utente non registrato	PASS
3 R	Inserimento Cognome formato non valido.	N/A	Doriana T dorianatraetto@gmail.com Dori1234 Dori1234 BSc	"Surname not valid"	Utente non registrato	PASS
4 R	Inserimento E-mail formato non valido.	N/A	Doriana Traetto dorianatraettogmail.com Dori1234 Dori1234 BSc	"E-mail not valid"	Utente non registrato	PASS
5 R	Inserimento Password formato non valido.	N/A	Doriana Traetto dorianatraetto@gmail.com Dori Dori BSc	"Password not valid"	Utente non registrato	PASS
6 R	Inserimento di Password e Conferma Password differenti.	N/A	Doriana Traetto dorianatraetto@gmail.com Iole1234 Dori1234 BSc	"Check_Password not valid"	Utente non registrato	PASS
7 R	Inserimento E-mail già registrata.	e-mail = dorianatraetto@gmail.com	Doriana Traetto dorianatraetto@gmail.com Dori1234 Dori1234 BSc	"E-mail already in use"	Utente non registrato	PASS
1 LI	Inserimento dati di input corretti.	User registrato	dorianatraetto@gmail.com Dori1234	Viene restituito il token di autenticazione.	Utente autenticato	PASS

2 LI	Inserimento E-mail errata, non associata ad un utente registrato nel database.	User registrato	marikasasso@gmail.com Dori1234	"E-mail not found"	Utente non autenticato	PASS
3 LI	Inserimento Password errata.	User registrato	dorianatraetto@gmail.com Marika1234	"Incorrect password"	Utente non autenticato	PASS
1 LO	Premere sul tasto Logout.	User autenticato	N/A	"Logout successful"	Utente non più autenticato	PASS
2 LO	L'utente non è autenticato	User autenticato	N/A	"User not authenticated"	Utente ancora autenticato	PASS
1 PR	Inserimento dati di input corretti.	User registrato	dorianatraetto@gmail.com	"Password reset e-mail sent successfully"	Viene inviata l'e-mail con il token di reset password.	PASS
2 PR	Inserimento E-mail errata, non associata ad un utente registrato nel database.	User registrato	marikasasso@gmail.com	"E-mail not found"	N/A	PASS
1 PC	Inserimento dati di input corretti.	User registrato, Token di reset password.	dorianatraetto@gmail.com sidnbs9839...28dhian Marika1234 Marika1234	"Password change successful"	Password modificata nell'Utente registrato.	PASS
2 PC	Inserimento E-mail errata, non associata ad un utente registrato nel database.	User registrato, Token di reset password.	marikasasso@gmail.com sidnbs9839...28dhian Marika1234 Marika1234	"Email not found"	Password non modificata nell'Utente registrato.	PASS
3 PC	Inserimento token errato.	User registrato, Token di reset password.	dorianatraetto@gmail.com aod1234 Marika1234 Marika1234	"Invalid reset token"	Password non modificata nell'Utente registrato.	PASS
4 PC	Inserimento Password formato non valido.	User registrato, Token di reset password.	dorianatraetto@gmail.com sidnbs9839...28dhian Marika Marika	"Password not valid"	Password non modificata nell'Utente registrato.	PASS
5 PC	Inserimento di Nuova e Conferma Password differenti.	User registrato, Token di reset password.	dorianatraetto@gmail.com sidnbs9839...28dhian Marika1234 Marika58	"Check_Password not valid"	Password non modificata nell'Utente registrato.	PASS