

Università degli Studi di Napoli Federico II



Scuola Politecnica e delle Scienze di Base

Dipartimento di Ingegneria Elettrica e Tecnologie
dell'Informazione

Corso di Laurea Magistrale in Ingegneria Informatica

Documentazione A10

Software Architecture Design

Prof.ssa

Anna Rita Fasolino

Candidati

Vincenzo D'Angelo M63/1595

Giorgio Di Costanzo M63/1579

Aurelio Salvati M63/1619

*Il seguente documento fa riferimento al progetto presentato e sviluppato sulla
repo GitHub: [A10-2024](#)*

Indice

Introduzione	5
1.1 Caso di studio.....	5
1.1.1 Punto di partenza	6
1.2 Planning	6
1.3 Workflow diagram	9
Analisi dei Requisiti	10
2.1 User Stories.....	10
2.2 Use Case Diagram	11
2.3 Scenari	12
2.3.1 View Classes	12
2.3.2 View Players	13
2.4 Dipendenze tra i Task.....	14
2.4.1 Diagramma delle dipendenze.....	14
2.4.2 Tabella delle dipendenze	15
Progettazione	17
3.1 Architettura a Microservizi	17
3.2 Gateway Pattern.....	17
3.3 Component Diagram	25
3.3.1 Composite Diagram	26
3.4 Sequence Diagram	27
3.5 Communication Diagram	32
Deployment Diagram	33
4.1 Esposizione su Rete Pubblica	33
4.1.1 Soluzione 1: Servizi di Tunneling	34
4.1.2 Soluzione 2: Server esterno	35
Implementazione.....	38
5.1 Applicazione distribuita	38
5.2 Cruscotto amministratore.....	38
5.3 Ulteriori modifiche	39
Testing.....	44
6.1 Testing manuale	44
6.1.1 Login da più dispositivi per ogni provider.....	44
6.1.2 Giocare in contemporanea da dispositivi diversi.....	46

6.1.3 Login con lo stesso account da dispositivi diversi.....	47
6.1.4 Giocare con lo stesso account contemporaneamente da due dispositivi diversi ..	48
6.1.5 Registrazione in contemporanea.....	49
6.2 Testing automatico	52
6.2.1 Risultati Test Ngrok.....	54
6.2.2 Risultati Test Pinggy	56
Installazione	59
7.1 Installazione Windows/Unix	59
7.2 Installazione Ngrok	59
7.3 Installazione Pinggy.....	60
7.4 Installazione su server esterno	61
7.4.1 Problemi comuni	63
7.5 Uninstaller	63
Guida alle future integrazioni.....	64
8.1 Integrazione API	64
8.2 Integrazione Container	64
8.3 Integrazione con UI Gateway.....	64
8.4 Integrazione con API Gateway	65
8.5 Integrazione Installer	65
Note e sviluppi futuri	66
HOW TO: Come modificare l'applicazione	68
Strumenti Software e Tecnologie Utilizzate	71

Introduzione

Il progetto ENACTEST (European iNnovative AllianCe for TESTing) si impegna a evidenziare l'importanza del testing, spesso trascurato a causa delle conoscenze insufficienti da parte dello sviluppatore rispetto a quelle richieste dall'azienda. Da qui nasce il gioco educativo “Man vs Automated Testing Tools challenges”, che sfrutta la gamification per stimolare gli studenti alla progettazione di casi di test facendoli competere contro i robot Randoop o EvoSuite che sono in grado di generare automaticamente casi di test JUnit.

1.1 Caso di studio

Attualmente, il caso di studio prevede l'implementazione di un singolo scenario di gioco, nel quale un giocatore ha la possibilità di disputare una partita contro un avversario singolo (Randoop o EvoSuite), testando così una classe a scelta.

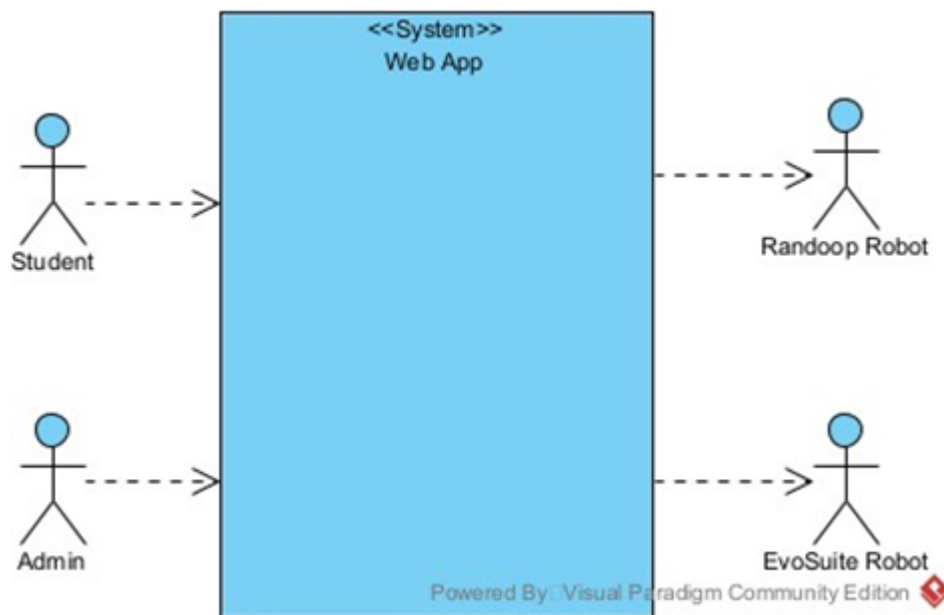


Figure 1.1: Diagramma di contesto ad alto livello

1.1.1 Punto di partenza

In questa documentazione è descritto il lavoro svolto partendo dal progetto A1-A8 2024, il quale si è occupato di sviluppare uno scenario alternativo durante il caricamento delle classi (prevedere di caricare test pre-calcolati) e predisporre l'app ad una versione in lingua inglese.

1.2 Planning

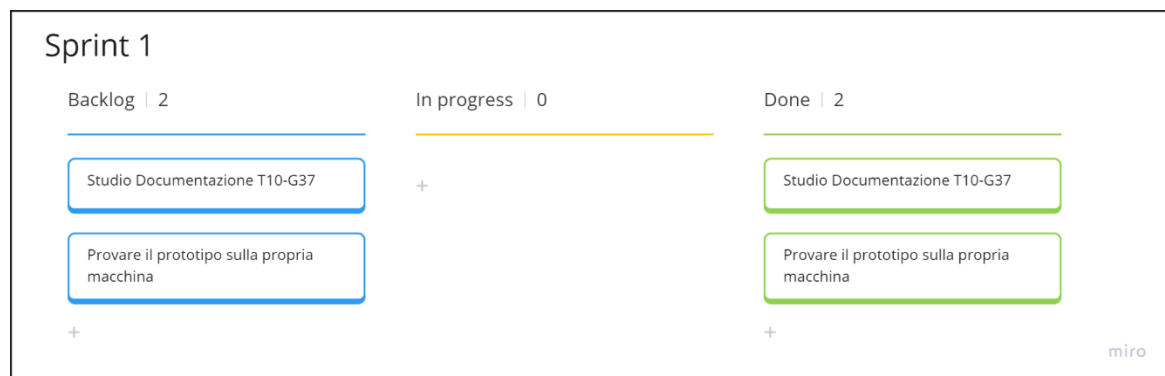
Il team ha seguito un approccio Agile basato sul paradigma SCRUM, un Framework caratterizzato dalla sua natura iterativa ed incrementale.

Per la realizzazione dei requisiti assegnati sono state svolte quattro iterazioni con un totale di cinque sprint. Ogni sprint è stato organizzato attraverso l'ausilio dello strumento Miro, che si è dimostrato ottimo per la collaborazione tra i membri del gruppo.

Si è adottata la seguente strategia per le modifiche al codice: le modifiche eseguite da un singolo componente del team sono state replicate anche dai restanti membri in modo da verificarne la correttezza.

All'inizio di ogni sprint si è creato un backlog con all'interno gli obiettivi prefissati per quel determinato sprint, man mano aggiornati con quelli ancora da completare (in progress) e quelli ultimati (done).

Di seguito sono riportate tutte le lavagne per ogni sprint.



Sprint 2

Backlog | 4

Esporre l'applicazione su un indirizzo IP configurabile

Rimuovere i riferimenti a localhost dai vari componenti/ task

Leggere l'IP di partenza da un file di configurazione

Cruscotto per l'amministratore

+

In progress | 3

Lettura IP da file

Esporre l'applicazione su un indirizzo IP configurabile

Cruscotto per l'amministratore

+

Done | 1

Rimuovere i riferimenti a localhost dai vari componenti/ task

+

miro

Sprint 3

Backlog | 3

Esporre l'applicazione su un indirizzo IP configurabile

Leggere l'IP di partenza da un file di configurazione

Cruscotto per l'amministratore

+

In progress | 1

Cruscotto per l'amministratore

+

Done | 2

Esporre l'applicazione su un indirizzo IP configurabile

Leggere l'IP di partenza da un file di configurazione

+

miro

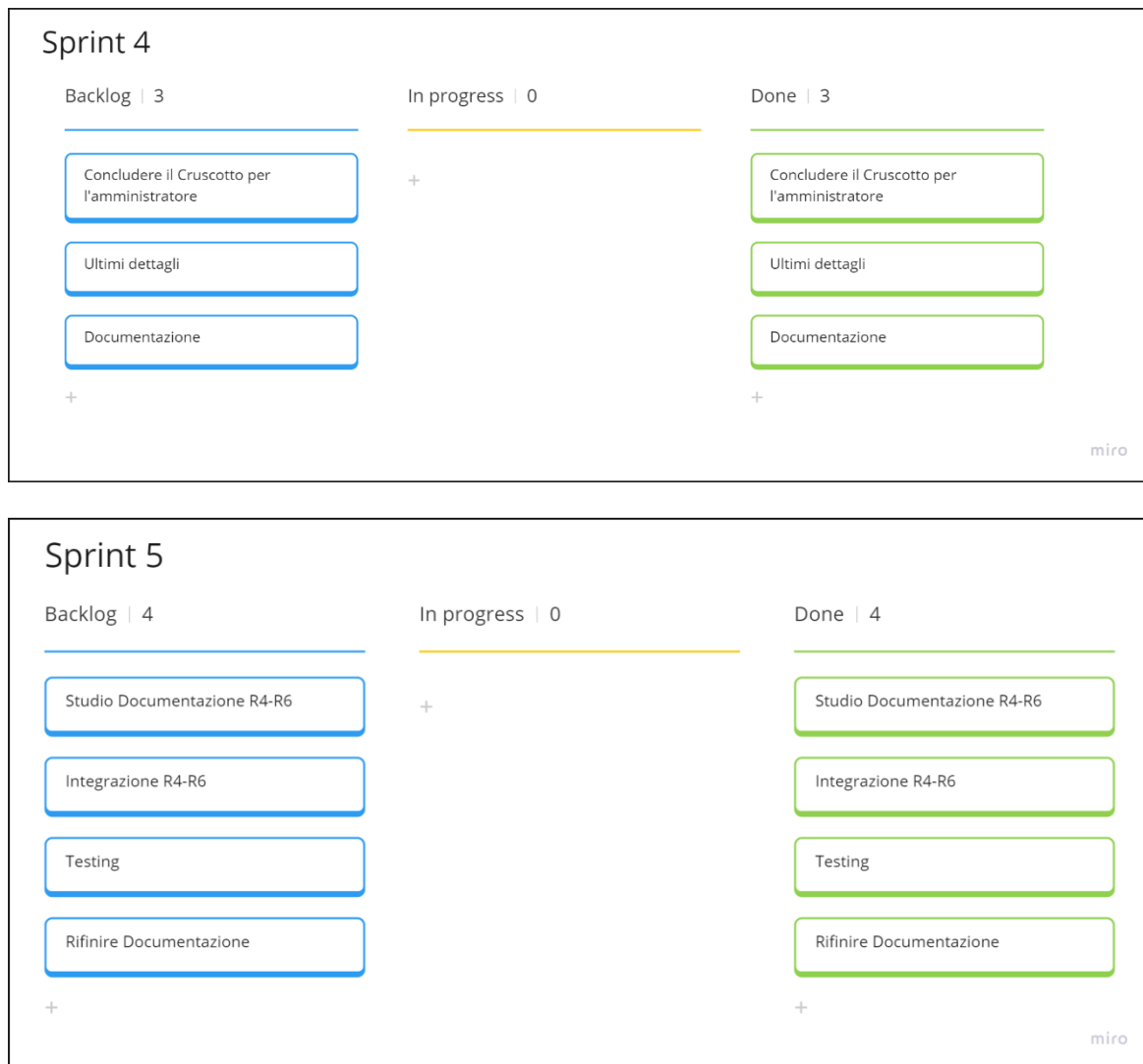


Figure 1.2: Iteration planning

1.3 Workflow diagram

L'unico scenario di gioco al momento disponibile per lo studente è descritto al meglio tramite il seguente workflow diagram.

Esso prevede prima il login dell'utente, quindi la selezione della classe da testare e il robot contro il quale si vuole competere. Dopo la scrittura del test personale e il relativo submit avviene la compilazione. Successivamente viene calcolato il punteggio dell'utente, confrontato con quello del robot, ed entrambi i risultati vengono visualizzati sulla console insieme all'esito della partita (vittoria o sconfitta).

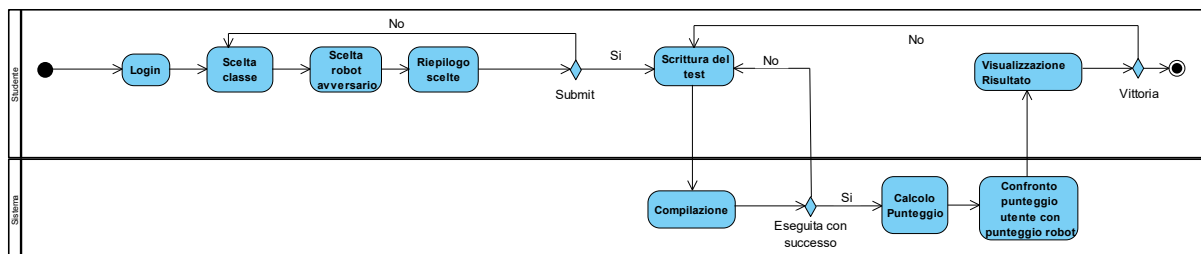


Figure 1.3: Primo scenario di gioco

Analisi dei Requisiti

Di seguito si riportano i requisiti assegnati.

ID	Descrizione
R5	Si vuole esporre l'applicazione su un indirizzo IP configurabile, in modo che non funzioni solo su localhost. Verificare che l'applicazione sia effettivamente raggiungibile da più giocatori e che questi possano giocare contemporaneamente
R11	Si vuole prevedere per l'amministratore un sistema che dia accesso diretto a diverse funzionalità di interrogazione che prevedono la visualizzazione dell'elenco dei giocatori iscritti e delle classi disponibili. Per ogni giocatore l'amministratore deve poter visualizzare Nome, Cognome, Corso, Numero di partite giocate e tempo totale di gioco. L'elenco dei giocatori deve prevedere la possibilità di ordinamento. L'elenco delle classi disponibili deve riportare tutti i dettagli delle singole classi.

Tabella 2-1: Requisiti assegnati

2.1 User Stories

Nei Metodi Agili, i requisiti vengono definiti attraverso frequenti interazioni con gli utenti. Le storie utente rappresentano funzionalità o requisiti di interesse per gli utenti del sistema software e seguono un formato standard. Vengono spesso scritte su piccole schede, come i Post-It, limitando i dettagli per favorire la chiarezza. Per la loro stesura si è scelto di adottare il classico template

As a < type of user >,
I want < to perform some task >
so that I can < achieve some goal/benefit/value>.

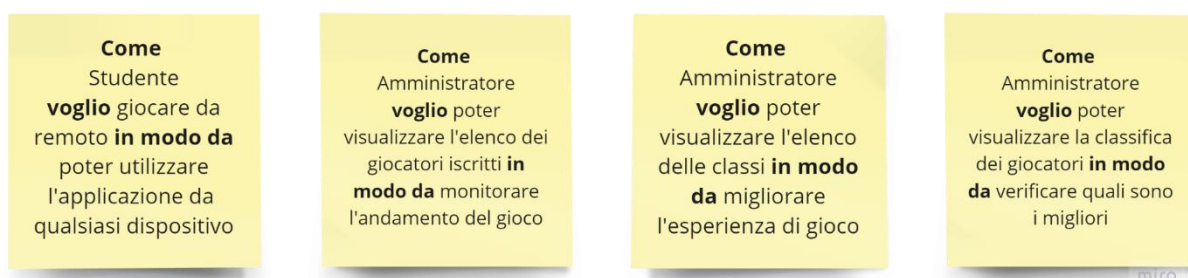


Figure 2.1: User Stories

2.2 Use Case Diagram

Si è deciso di realizzare un diagramma dei casi d'uso generale di tutte le funzionalità disponibili in questa integrazione. Per tenere separate le funzionalità relative alle classi e quelle relative ai giocatori, la home dell'amministratore è stata riprogettata con la realizzazione di un pratico Menu.

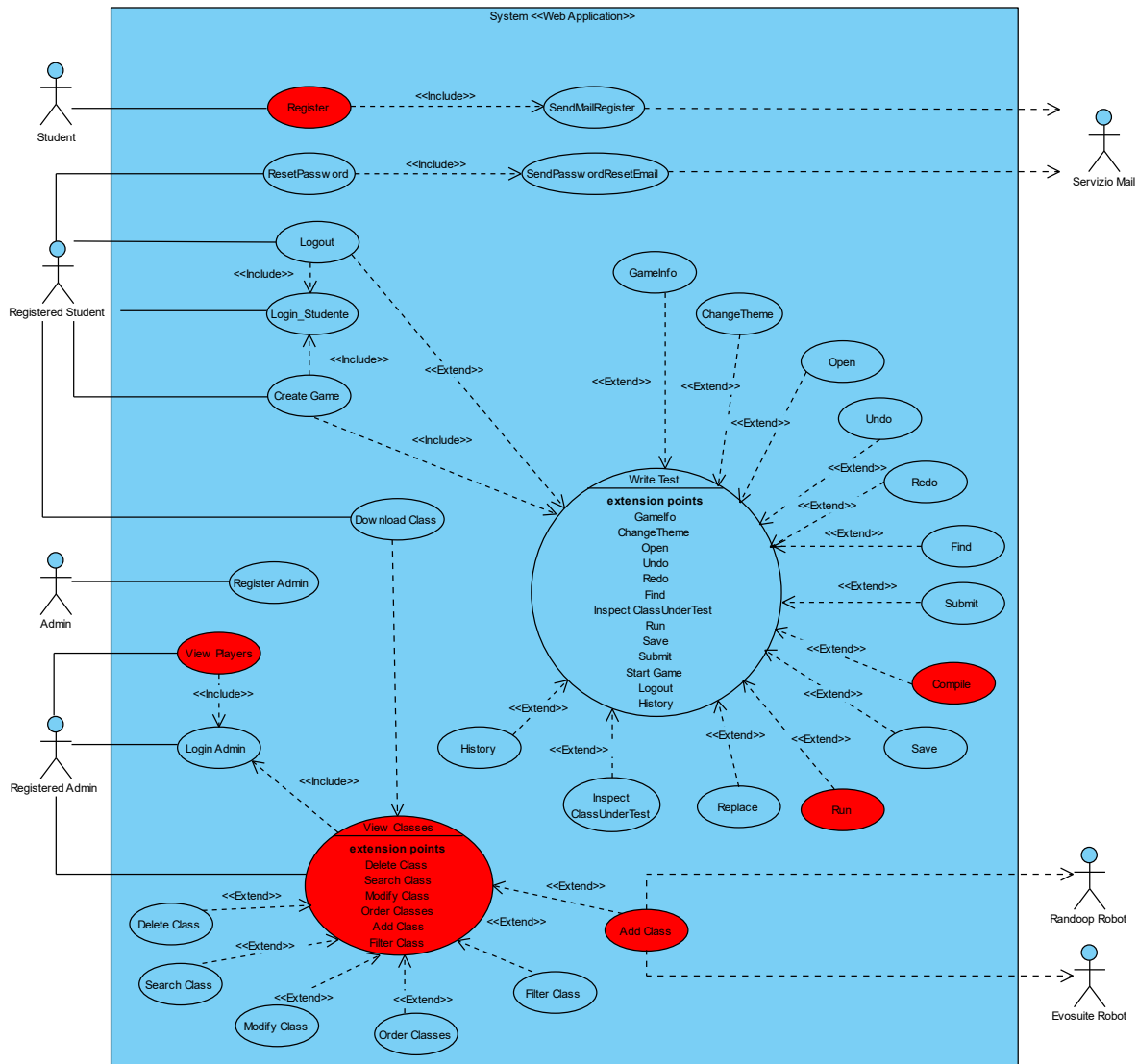


Figure 2.2: Diagramma dei casi d'uso. In rosso sono indicati i casi d'uso aggiunti/modificati

Si riporta una lista dei casi d'uso modificati o aggiunti:

- **“Register”** - In fase di registrazione dello studente è stato rimosso il CAPTCHA, poiché non funzionante. Inoltre, il sistema di sicurezza causava problemi una volta portata l'applicazione su rete pubblica;
- **“Submit”, “Compile”, “Run”, “Add Class”** - È stato aggiunto un feedback grafico durante le fasi di caricamento;
- **“View Classes”** - L'amministratore può accedere tramite un Menu alle funzionalità relative alle classi;
- **“View Players”** - L'amministratore può accedere tramite un Menu alle funzionalità relative ai giocatori.

2.3 Scenari

Si riportano gli scenari dei casi d'uso assegnati.

2.3.1 View Classes

Caso d'uso View Classes	
Attore primario	Registered Admin
Attore Secondario	-
Descrizione	Permette ad un amministratore registrato di visualizzare le informazioni relative alle classi
Pre-condizioni	L'amministratore è loggato ed è nella Home
Sequenza di eventi principale	L'amministratore clicca il tasto “Classes” del Menu nella Home
Post-Condizioni	L'amministratore visualizza l'elenco delle classi
Casi d'uso correlati	<i>Delete Class, Search Class, Modify Class, Order Classes, Filter Class, Add Class, Download Class</i>
Sequenza di eventi alternativi	-

Figure 2.3: View Classes

2.3.2 View Players

Caso d'uso View Players	
Attore primario	Registered Admin
Attore Secondario	-
Descrizione	Permette ad un amministratore registrato di visualizzare le informazioni relative ai giocatori
Pre-condizioni	L'amministratore è loggato ed è nella Home
Sequenza di eventi principale	L'amministratore clicca il tasto "Players" del Menu nella Home
Post-Condizioni	L'amministratore visualizza l'elenco dei giocatori
Casi d'uso correlati	Nessuno
Sequenza di eventi alternativi	-

Figure 2.4: View Players

2.4 Dipendenze tra i Task

Durante il lavoro di aggiunta e modifica precedentemente descritto è stato necessario porre l'attenzione su determinate operazioni che richiedono l'impiego di servizi forniti da altri task. Le informazioni ricavate sono riportate nei seguenti sottoparagrafi.

2.4.1 Diagramma delle dipendenze

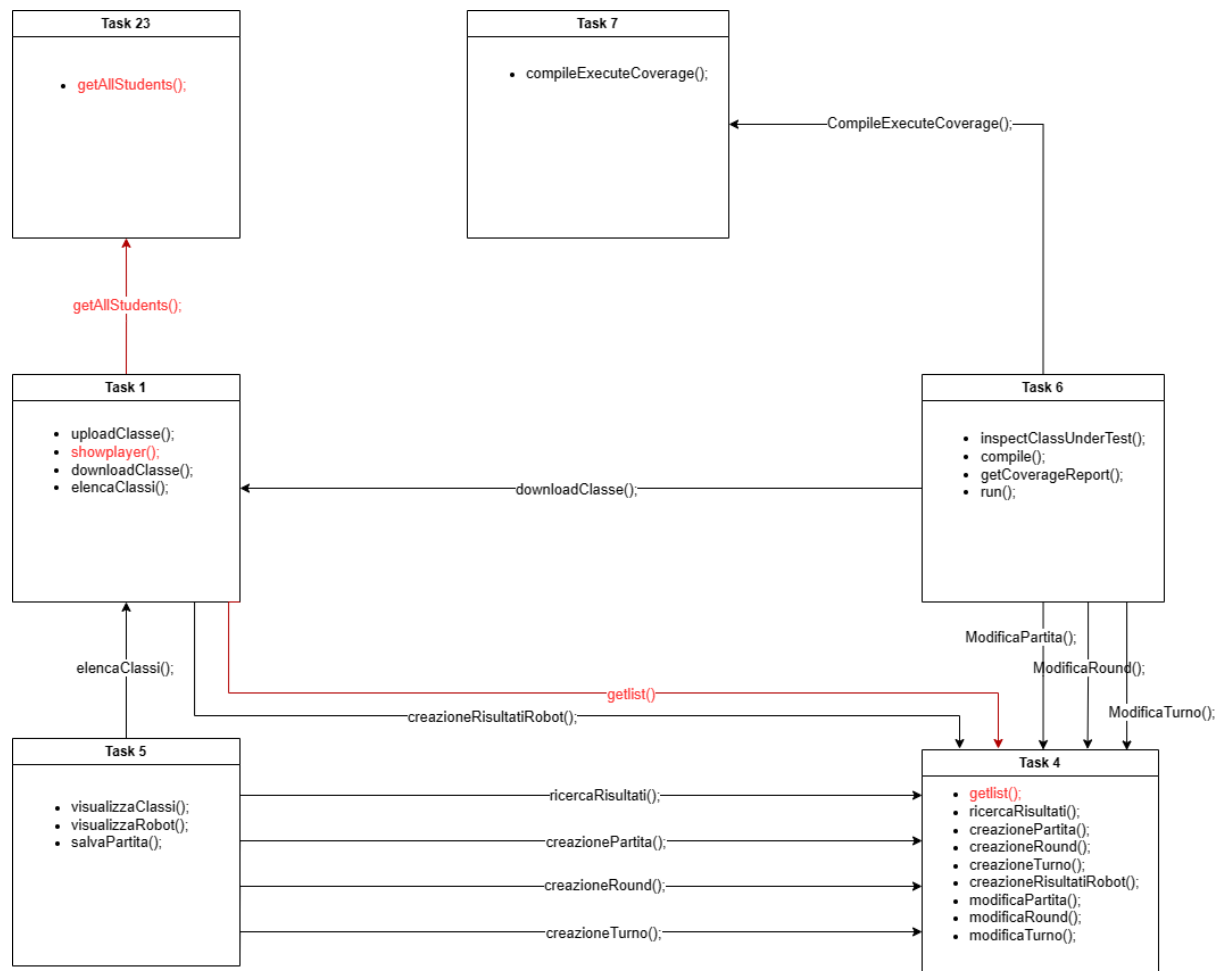


Figure 2.5: Diagramma delle dipendenze. In rosso sono indicate le operazioni aggiunte/modificate

2.4.2 Tabella delle dipendenze

Service	Operation	Collaborators
Task 1	uploadClasse()	Task 4: creazioneRisultatiRobot(): /robots POST
	showplayer()	Task 23: getAllStudents(): /students_list GET Task 4: getlist(): /games GET
Task 5	visualizzaClassi()	Task 1: elencaClassi(): /home GET
	visualizzaRobot()	Task 4: ricercaRisultatiRobot(): /robots GET
	salvaPartita()	Task 4: creazionePartita(): /games POST creazioneRound(): /rounds POST creazioneTurno(): /turns POST
Task 6	inspectClassUnderTest()	Task 1: downloadClasse(): /downloadFile GET

	compile()	Task 7: compileExecuteCoverage(): /compile-and-codecoverage POST
	getCoverageReport()	Task 7: compileExecuteCoverage(): /compile-and-codecoverage POST
	run()	Task 7: compileExecuteCoverage(): /compile-and-codecoverage POST Task 4: modificaPartita(): /games PUT modificaRound(): /rounds PUT modificaTurno(): /turns PUT

Tabella 2-2: Tabella delle dipendenze. In rosso sono indicate le operazioni aggiunte/modificate

Progettazione

3.1 Architettura a Microservizi

I microservizi costituiscono un paradigma architetturale nell'ambito dello sviluppo software, caratterizzato dalla suddivisione di un'applicazione complessa in singoli servizi modulari, ciascuno responsabile di una specifica funzionalità autonoma. Questi microservizi operano in modo indipendente l'uno dall'altro, consentendo lo sviluppo, l'implementazione e la gestione separati di ciascun componente. L'architettura favorisce la modularità, la scalabilità e la flessibilità nell'evoluzione del sistema. La comunicazione tra i microservizi avviene attraverso interfacce chiaramente definite, spesso basate su protocolli leggeri come HTTP.

3.2 Gateway Pattern

Sebbene l'architettura a microservizi scelta si adatti perfettamente all'esigenza di dividere i task per gruppo, l'integrazione di tali microservizi risulta essenziale. L'approccio scelto nell'architettura utilizza il Gateway Pattern. Esso consiste nell'introdurre dei componenti che costituiscono gli unici punti di accesso ai microservizi: ciò permette di implementare meccanismi di autenticazione, autorizzazione, routing, load-balancing e composizione delle API (una sorta di facade) delle richieste molto più versatili ed efficienti.

API Gateway

Il componente è stato realizzato utilizzando Spring Boot insieme a Netflix Zuul, realizzando un reverse proxy: si occupa di effettuare il routing, di gestire autenticazione e autorizzazione delle richieste relative alle REST API. Ciò ha permesso di rendere disponibili tutte le API al di sotto del percorso URL `"/api/"`, ottenendo così una certa uniformità.

UI Gateway

Il componente è stato realizzato utilizzando Nginx, di fatto implementando un reverse proxy. La presenza di questo gateway gioca un ruolo fondamentale per l'intera architettura del sistema. Infatti, non solo uniforma tutte le richieste sul porto 80 (predefinito per il protocollo HTTP) sul quale è in ascolto, ma offre anche una grande possibilità di scalabilità e distribuzione.

Il gateway gestisce tutte le richieste in arrivo sulla macchina in esecuzione e smista tali richieste ai vari microservizi, offrendo così un'unica interfaccia all'esterno e aumentando la sicurezza e la scalabilità del sistema.

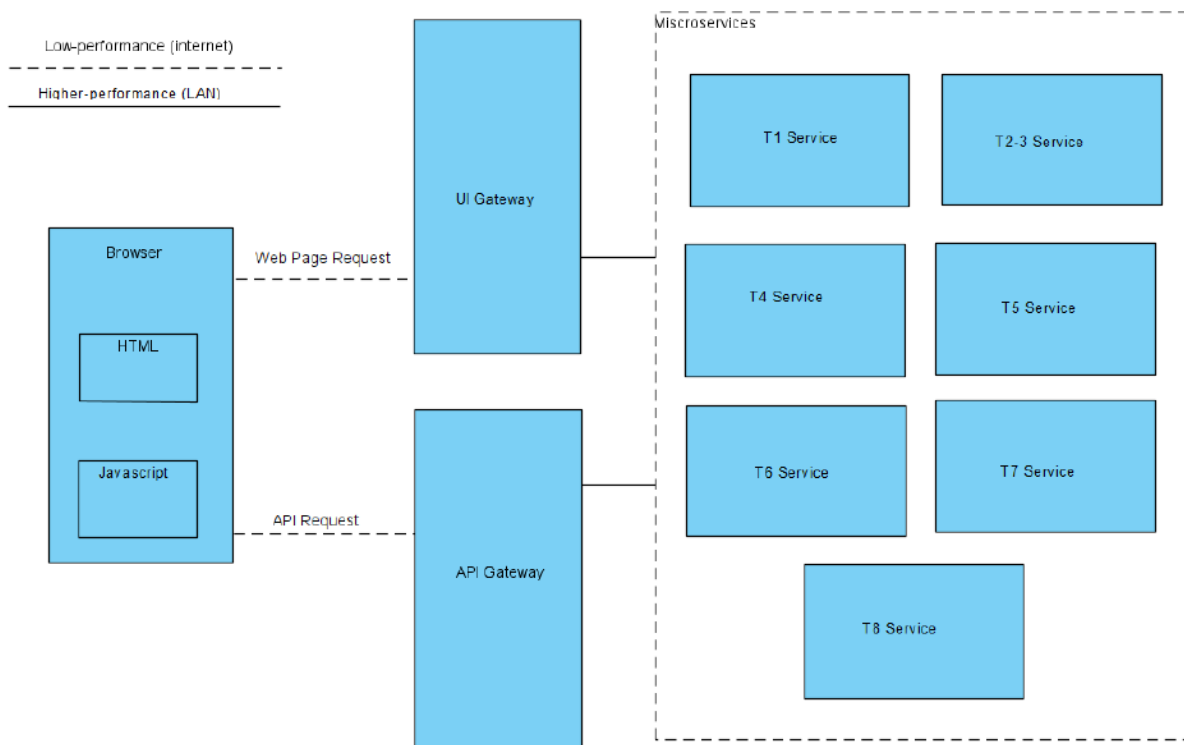


Figure 3.1: Architettura a microservizi

Il vero punto di ingresso del sistema è l'UI Gateway, l'utente interagisce esclusivamente con questo componente. L'UI Gateway gestisce le richieste e dirige l'API Gateway (la connessione con l'utente è solo logica).

L'API Gateway si occupa del re-routing, fungendo da punto di accesso del sistema che reindirizza alle risorse richieste. Inoltre, gestisce l'autenticazione dell'utente tramite token, garantendo non solo che l'utente sia registrato, ma abbia anche effettuato l'accesso.

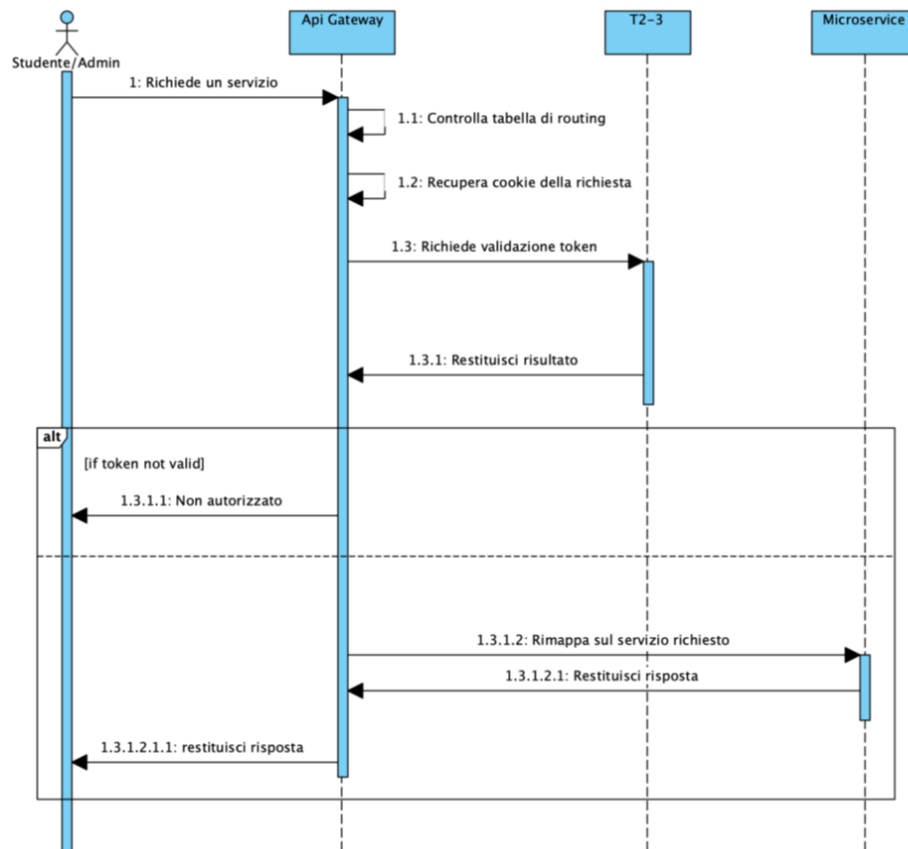


Figure 3.2: Sequence dimostrativo dell'API Gateway

A differenza dell'API Gateway, l'UI Gateway dispone di una lista di pagine escluse dal controllo del token, come ad esempio le pagine di accesso come il login o la registrazione. Se la pagina non è presente in tale lista, significa che richiede autenticazione e, di conseguenza, il token deve essere verificato. Il flusso di controllo del token è del tutto analogo a quello dell'API Gateway.

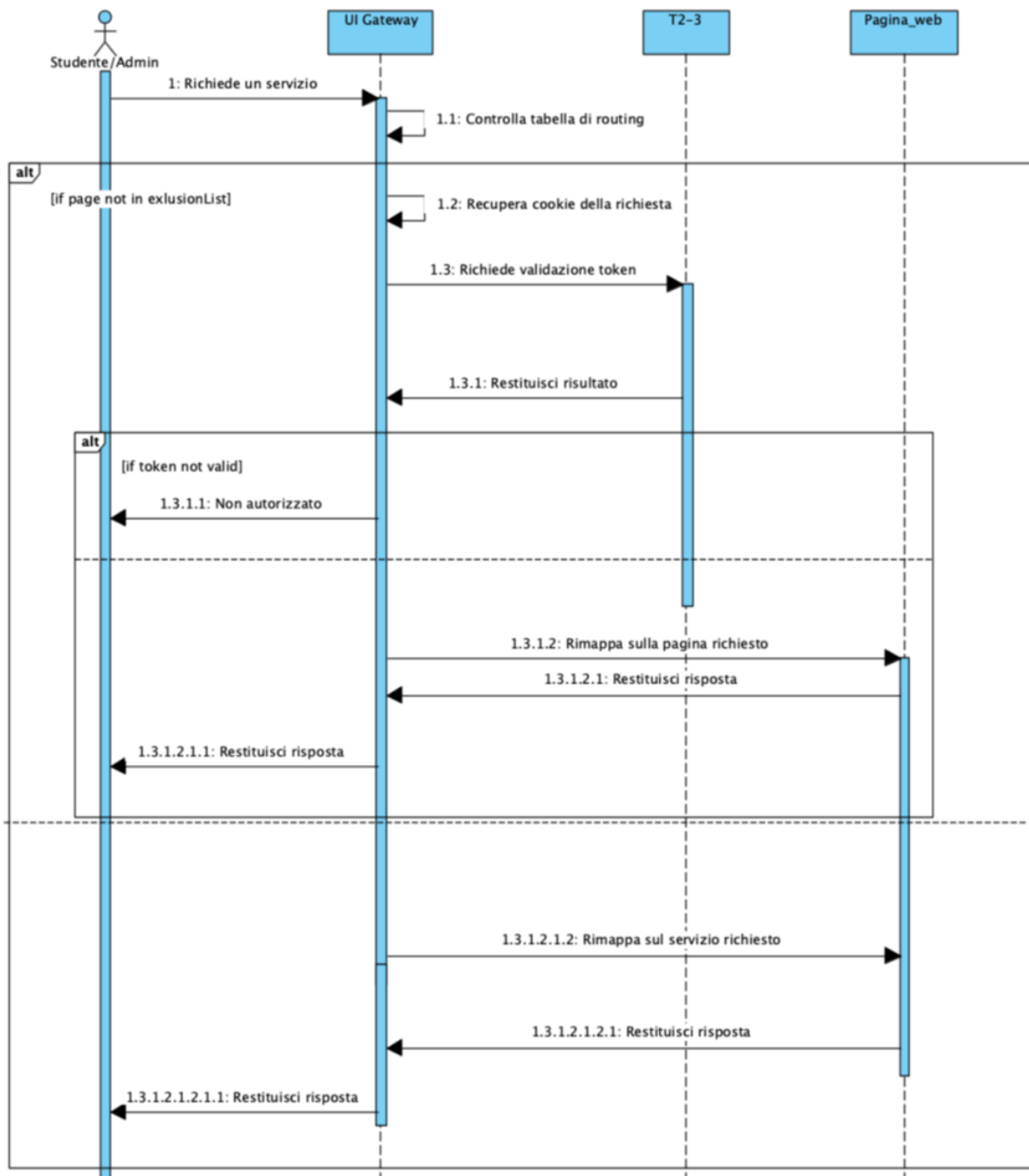


Figure 3.3: Sequence dimostrativo dell'UI Gateway

Di seguito però riportiamo quella che è l'implementazione effettiva dei gateway e di come avviene l'interazione con essi.

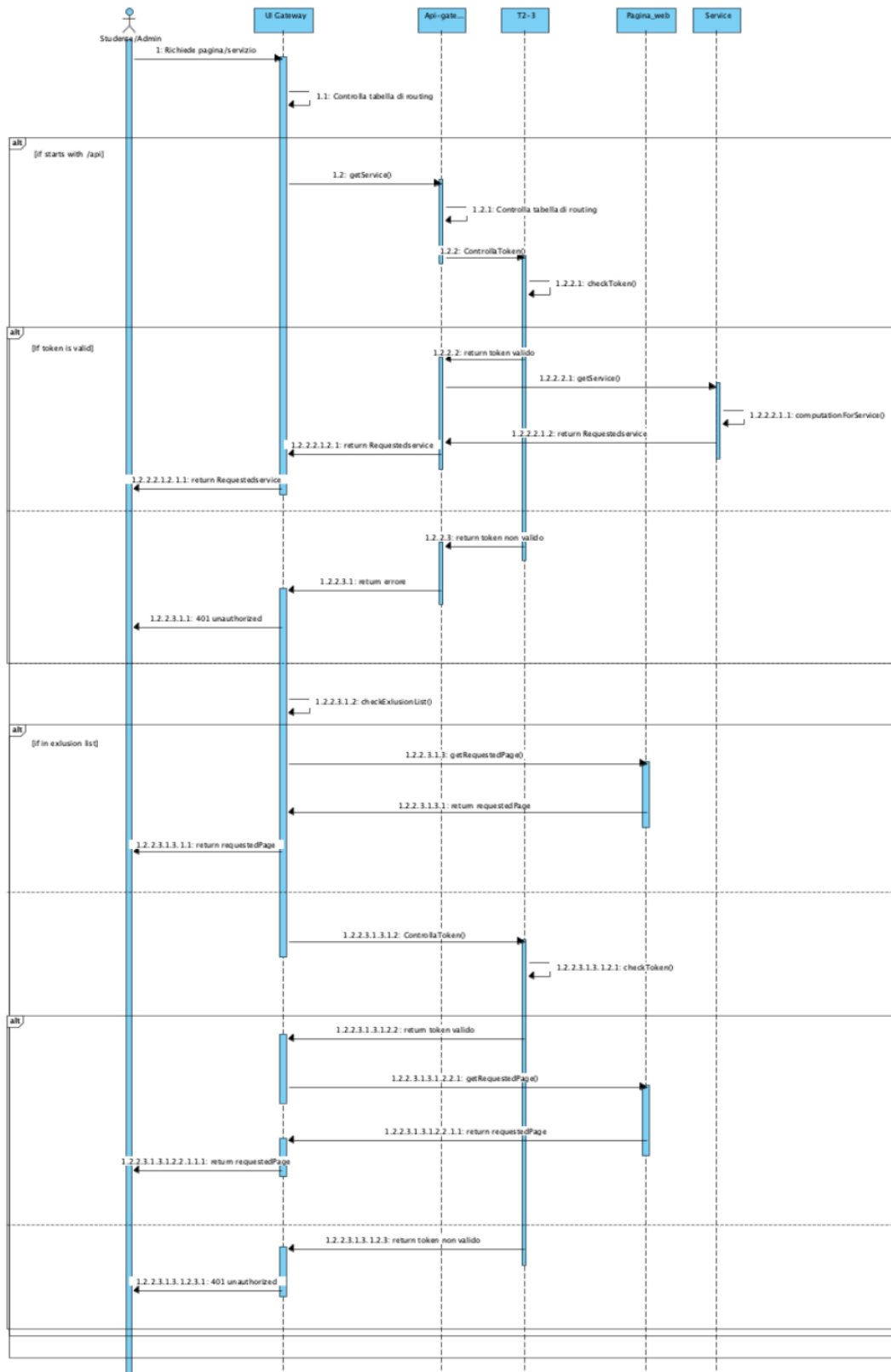


Figure 3.4: Sequence complessivo UI-API Gateway

Infine, una tabella riassuntiva esprime come tutte le location del sistema sono gestite:

UI-GATEWAY:

LOCATION	PROXY-PASS
/	/login
/class	http://manvsclass-controller-1:8080
/player	http://manvsclass-controller-1:8080
/loginAdmin	http://manvsclass-controller-1:8080
/registraAdmin	http://manvsclass-controller-1:8080
/home_adm	http://manvsclass-controller-1:8080
/modificaClasse	http://manvsclass-controller-1:8080
/orderbydate	http://manvsclass-controller-1:8080
/Dfilterby.+	http://manvsclass-controller-1:8080
/orderbyname	http://manvsclass-controller-1:8080
/Reports	http://manvsclass-controller-1:8080
/uploadClasse	http://manvsclass-controller-1:8080
/reportClasse	http://manvsclass-controller-1:8080
/delete	http://manvsclass-controller-1:8080
/getLikes	http://manvsclass-controller-1:8080
/uploadFile	http://manvsclass-controller-1:8080
/home	http://manvsclass-controller-1:8080
/t1	http://manvsclass-controller-1:8080
/uploadClasseAndTest	http://manvsclass-controller-1:8080
/uploadTest	http://manvsclass-controller-1:8080
/login	http://t23-g1-app-1:8080

/logout	http://t23-g1-app-1:8080
/register	http://t23-g1-app-1:8080
/mail_register	http://t23-g1-app-1:8080
/password_change	http://t23-g1-app-1:8080
/password_reset	http://t23-g1-app-1:8080
/t23	http://t23-g1-app-1:8080
/students_list	http://t23-g1-app-1:8080
/main	http://t5-app-1:8080
/editor	http://t5-app-1:8080
/report	http://t5-app-1:8080
/t5	http://t5-app-1:8080
/tests	http://prototipo20-t8_generazione-1:3080/tests
/robots	http://t4-g18-app-1:3000
/turns	http://t4-g18-app-1:3000
/games	http://t4-g18-app-1:3000
^~ /api	http://api_gateway-gateway-1:8090

Tabella 3-1: Tabella Routing UI-Gateway. In rosso sono indicate le operazioni aggiunte/modificate

Quando l'ui-gateway "passa" la rotta all'api gateway, il servizio richiesto è servito secondo le regole di questa tabella:

API GATEWAY:

Servizio	PATH	PROXY-PASS
Jacoco-service	/api/getJaCoCoReport/**	http://t6-g12-app-1:80/getJaCoCoReport
ReceiveClass-service	/api/receiveClassUnderTest/**	http://t6-g12-app-1:80/receiveClassUnderTest
SendInfo-service	/api/sendInfo/**	http://t6-g12-app-1:80/sendInfo
SaveData-service	/api/save-data/**	http://t5-app-1:8080/save-data
DownloadFile-service	/api/downloadFile/**	http://manvsclass-controller-1:8080/downloadFile
Run-service	/api/run/**	http://t6-g12-app-1:80/run
Loadingresult-service	/api/VolumeT8/FolderTreeEvo/**	http://prototipo20-t8_generazione-1:3080/api/VolumeT8/FolderTreeEvo

Tabella 3-2: Tabella Routing UI-Gateway. In rosso sono indicate le operazioni aggiunte/modificate

3.3 Component Diagram

Il diagramma dei componenti è uno strumento utile per capire come le diverse parti del sistema interagiscono tra di loro durante l'esecuzione per svolgere le funzioni necessarie. È importante notare che i componenti responsabili delle funzioni degli studenti sono separati da quelli dedicati all'amministrazione. Questa separazione è dovuta a causa di un un approccio diverso nell'autenticazione.

Per quanto riguarda l'accesso degli studenti, è stato implementato un ulteriore livello di sicurezza grazie all'utilizzo dei token JWT forniti dal componente Student Repository. Inoltre, è stato garantito un controllo di sicurezza aggiuntivo grazie all'API Gateway. Tuttavia, è importante sottolineare che quest'ultimo non offre la stessa protezione per le API relative all'amministrazione.

In figura la situazione è ben chiara:

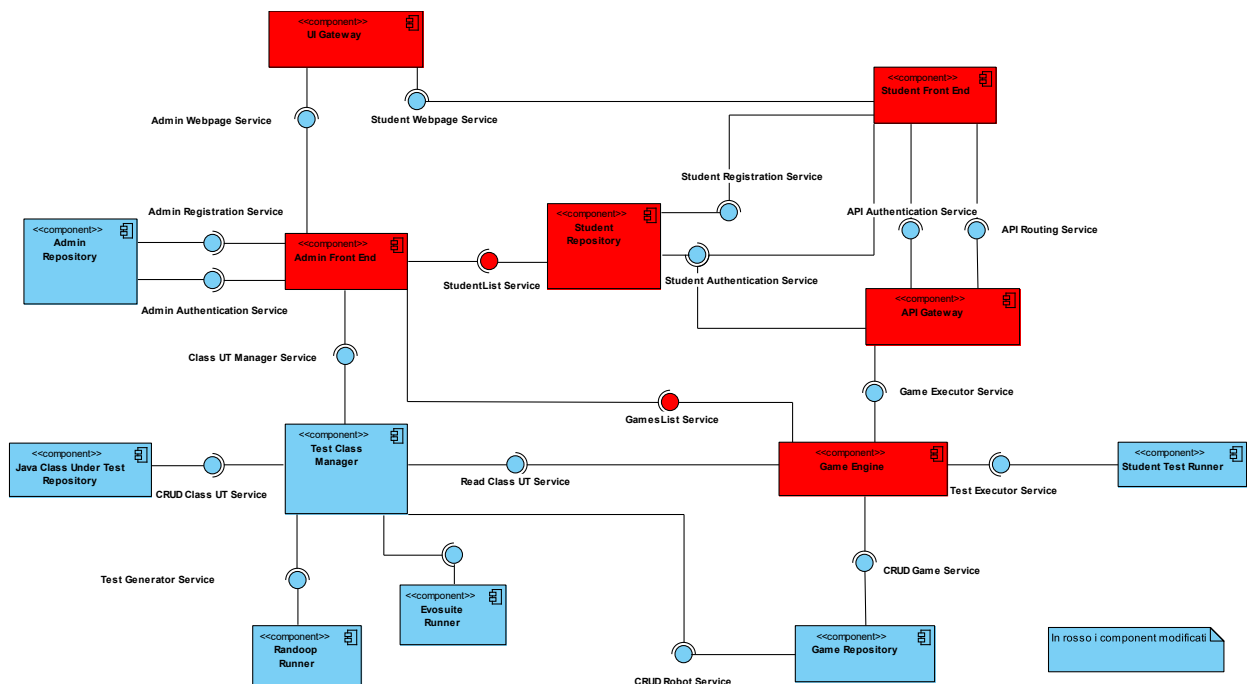


Figure 3.5: Component Diagram

3.3.1 Composite Diagram

Nel seguente diagramma, è stata rappresentata la mappatura di chi, tra i vari task, ha sviluppato quale componente visibile al momento dell'esecuzione. La notazione scelta, in coerenza con il Component Diagram, prevede l'inserimento di un package per ogni componente, specificando quali task ne sono responsabili e contribuiscono al suo corretto funzionamento.

Sebbene alcuni componenti siano interamente realizzati da singoli task, vi sono casi come lo Student Front End e il Game Engine che sono distribuiti su due task. Il task 2-3 è responsabile della realizzazione dello Student Front End per la parte relativa all'accesso al gioco, includendo le schermate di login, registrazione e recupero password. Per la sezione del gioco vero e proprio, il front end è stato sviluppato dal task 5.

Il Game Engine, invece, è stato realizzato nella fase di salvataggio iniziale della partita dal task 5 e per le restanti funzionalità dal task 6, tra cui l'esecuzione, la compilazione, eccetera.

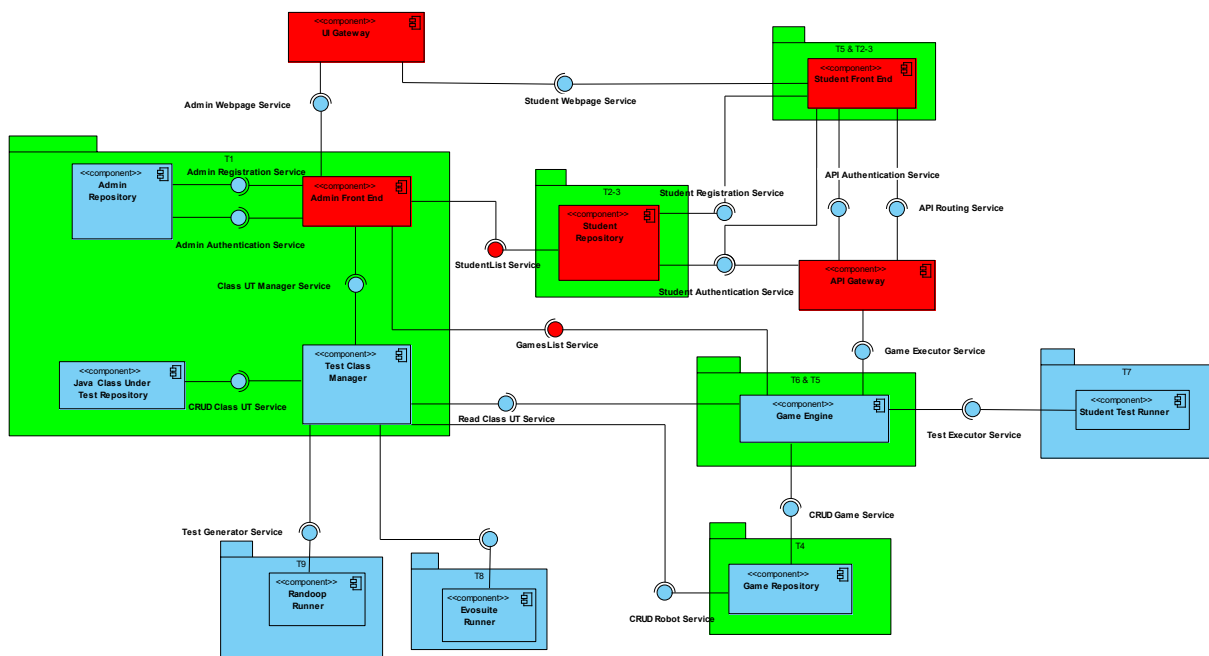


Figure 3.6: Composite Diagram. In verde i task modificati

3.4 Sequence Diagram

Al fine di ottenere una visione completa del funzionamento dell'applicazione web desiderata, vengono presentati i principali diagrammi di sequenza, tenendo conto delle modifiche apportate alle funzioni implementate dai task e delle aggiunte effettuate. Si rimanda alla documentazione dei singoli task (microservizi) per un dettaglio maggiore.

UI Gateway

Ogni volta che un client richiede l'accesso a una pagina web, interagisce con l'UI Gateway, il quale si occupa di instradare la richiesta. L'interazione è descritta nel seguente diagramma di sequenza. Il client richiede una pagina web e l'UI Gateway, dopo aver consultato la propria tabella di routing, effettua la richiesta al microservizio necessario per conto del client e gli inoltra la risposta.

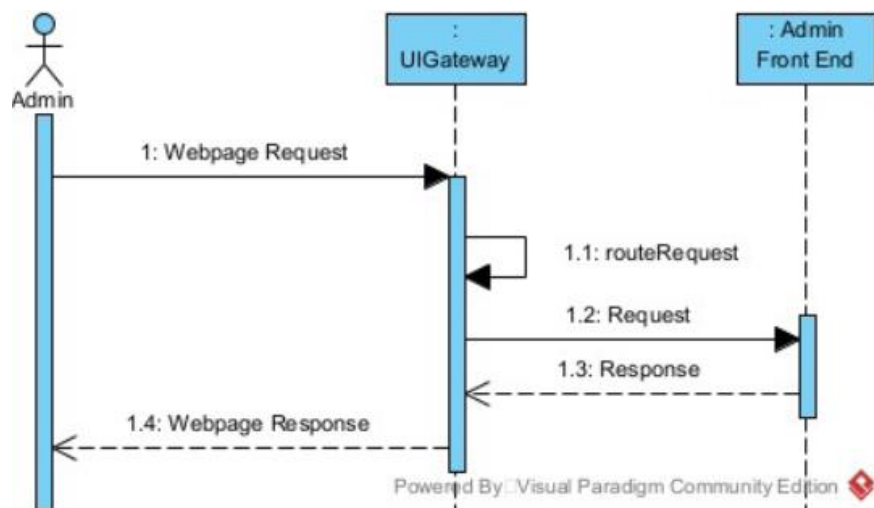


Figure 3.7: UI Gateway Sequence Diagram (Admin).

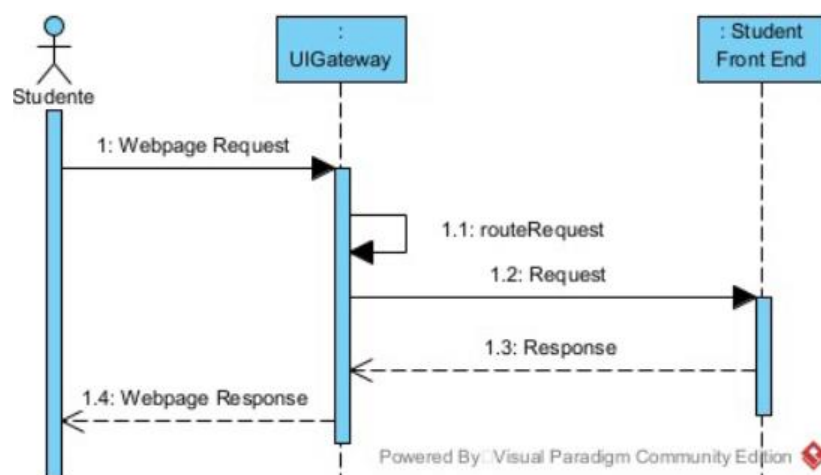


Figure 3.8: UI Gateway Sequence Diagram (Student)

API Gateway

Ogni volta che un client richiede il servizio di un'API, interagisce con l'API Gateway, che si occupa di instradare la richiesta, di verificare l'autenticazione e di preparare l'autorizzazione. L'interazione è descritta nel seguente diagramma di sequenza. Quando il client richiede l'utilizzo di un'API, l'API Gateway esegue le seguenti operazioni:

1. Estrae il token JWT contenuto in un cookie della richiesta.
2. Verifica che il token JWT sia valido, utilizzando il servizio offerto dallo Student Repository (nel caso in cui non lo sia, restituisce un errore "Not Authorized").
3. Estrae i claim contenuti all'interno del token JWT (cioè l'ID dello studente) e li aggiunge come Header di autorizzazione della richiesta.
4. Consulta la propria tabella di routing.
5. Esegue la richiesta al microservizio richiesto per conto del client e gli inoltra la risposta.

Esecuzione (Run)

Questo caso d'uso consente la conclusione e il salvataggio della partita, nonché il calcolo del vincitore. Si attiva quando lo studente preme il tasto "Run" nell'editor. Le operazioni eseguite sono le seguenti:

1. L'Editor UI effettua una richiesta POST all'API **/run** (implementata dal Task 6) presente nel Game Engine Controller. Quest'ultimo si occupa di svolgere tutte le azioni necessarie, inserendo le informazioni sulla partita e il codice sottomesso nel corpo della richiesta.
2. Il Game Engine Controller effettua a sua volta una richiesta POST all'API **/compileand-codecoverage** (implementata dal Task 7), la quale ha il compito di compilare il codice dell'utente e calcolarne la copertura, restituendo quest'ultima insieme all'output di compilazione e al report CSV.
3. Il Game Engine Controller effettua una richiesta GET all'API **/robots** verso il Game Repository (realizzato dal Task 4), ottenendo il punteggio del robot scelto dall'utente.
4. Successivamente, il Game Engine Controller effettua una serie di richieste PUT alle API **/turns**, **/rounds** e **/games** verso il Game Repository (realizzato dal Task 4) per aggiornare le informazioni sulla partita, sul turno e sul round (indicando nel caso del turno anche chi è il vincitore).
5. Vengono mostrati i risultati della partita, comprese tutte le metriche di coverage di EvoSuite e le linee di codice coperte dal test.

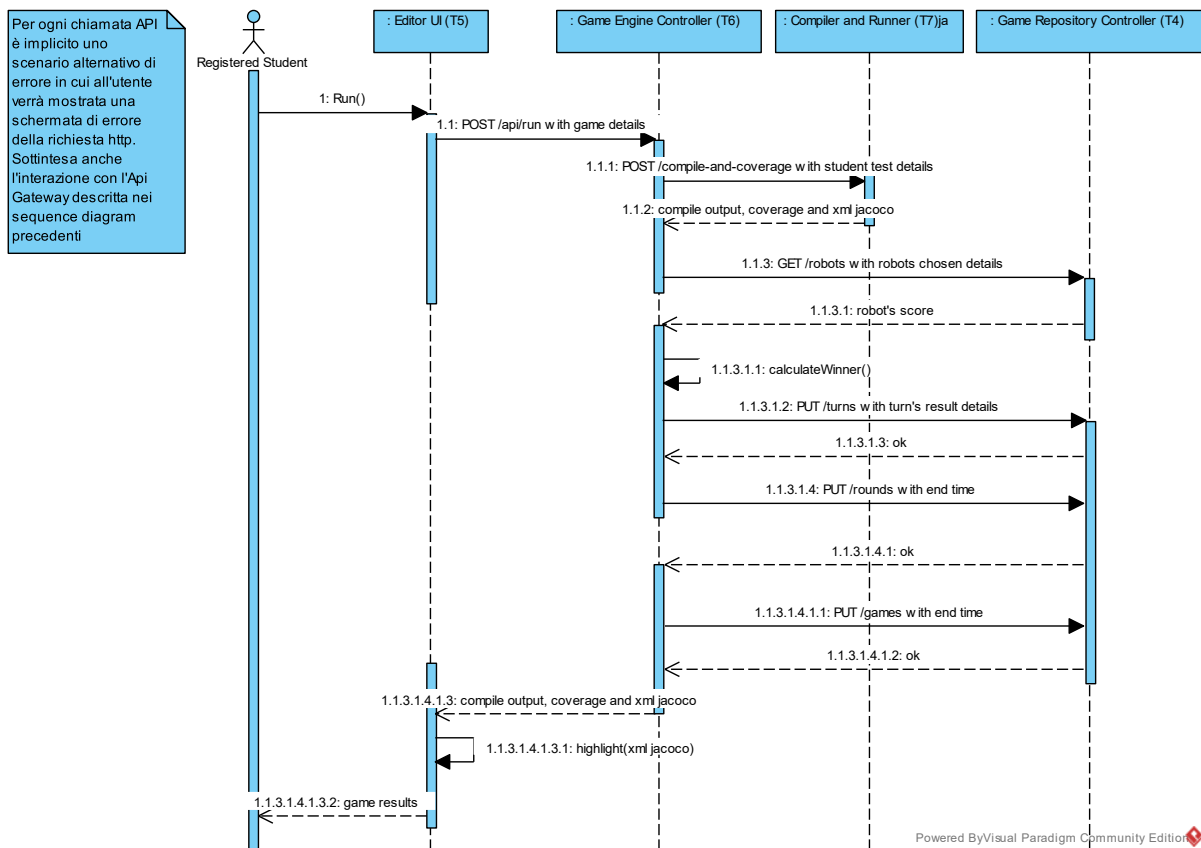


Figure 3.9: Run Sequence Diagram

CRUSCOTTO ADMIN

Per implementare un cruscotto amministratore, è necessario recuperare i dati di gioco interfacciandosi con due database del sistema, gestiti da due microservizi distinti.

Il Task T4 implementa un database con PostgreSQL e memorizza tutti i dati relativi al game-engine (partita, robot, turni, ecc.).

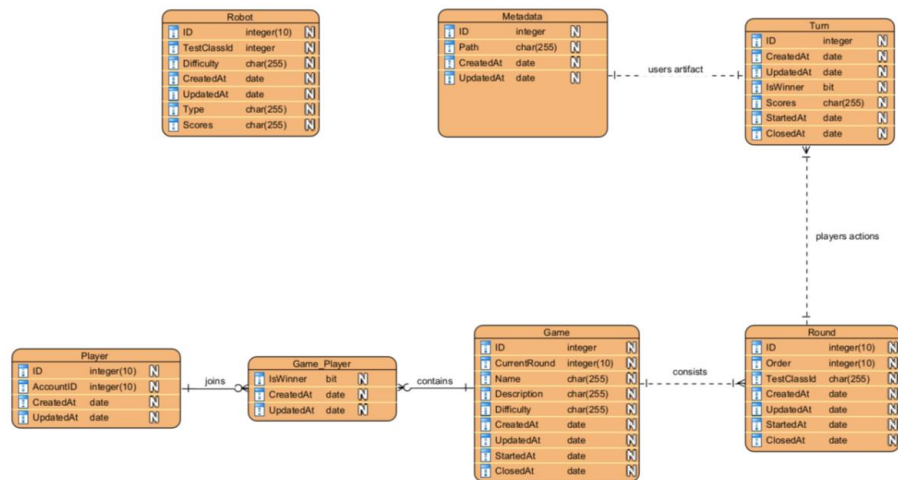


Figure 3.10: Database Task T4

Il Task T2-3 implementa con MySQL un database dedicato allo Student Repository, che contiene i dati anagrafici degli studenti.

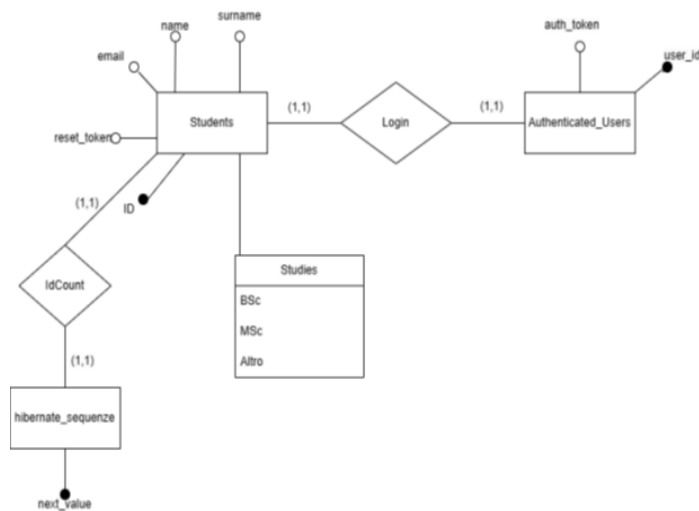


Figure 3.11: Database Task T2-3

Dato che i database sono gestiti da due servizi diversi, e inoltre il cruscotto amministratore è responsabilità di un ulteriore microservizio (T1), il recupero e l'aggregazione dei dati non può avvenire in modo "classico". È necessario ricorrere alle API fornite dai due database.

Supponendo di poter unire tutte le entità dei due database in un macro-database, necessitiamo di eseguire la seguente query:

```
CREATE VIEW AS TEMP
SELECT ID AS PLAYER
FROM PLAYER

SELECT [PUNTI], NAME, SURNAME, EMAIL, STUDIES, COUNT(*) AS PARTITE_GIOcate, SUM(CLOSEDAT-STARTEDAT)
FROM STUDENTS S JOIN TEMP T ON S.ID=T.PLAYER
```

L'idea alla base è la seguente:

1. Richiedere al database degli studenti tutti gli ID, Nome, Cognome.
2. Richiedere al database del game-engine tutti i dati di gioco nel database.

A questo punto, dobbiamo eseguire l'operazione di JOIN. Non potendola fare in maniera "SQL" ci avvaliamo dei seguenti passaggi:

1. Confrontiamo "l'account ID" del primo database con il "player.accountID" del secondo database. Questo confronto rappresenta la nostra clausola di JOIN tra le due tabelle.
2. Per ogni accountID, ricaviamo le informazioni di interesse: COUNT, TEMPO DI GIOCO, ECC.

La formalità del seguente diagramma UML non lascia dubbi al caso:

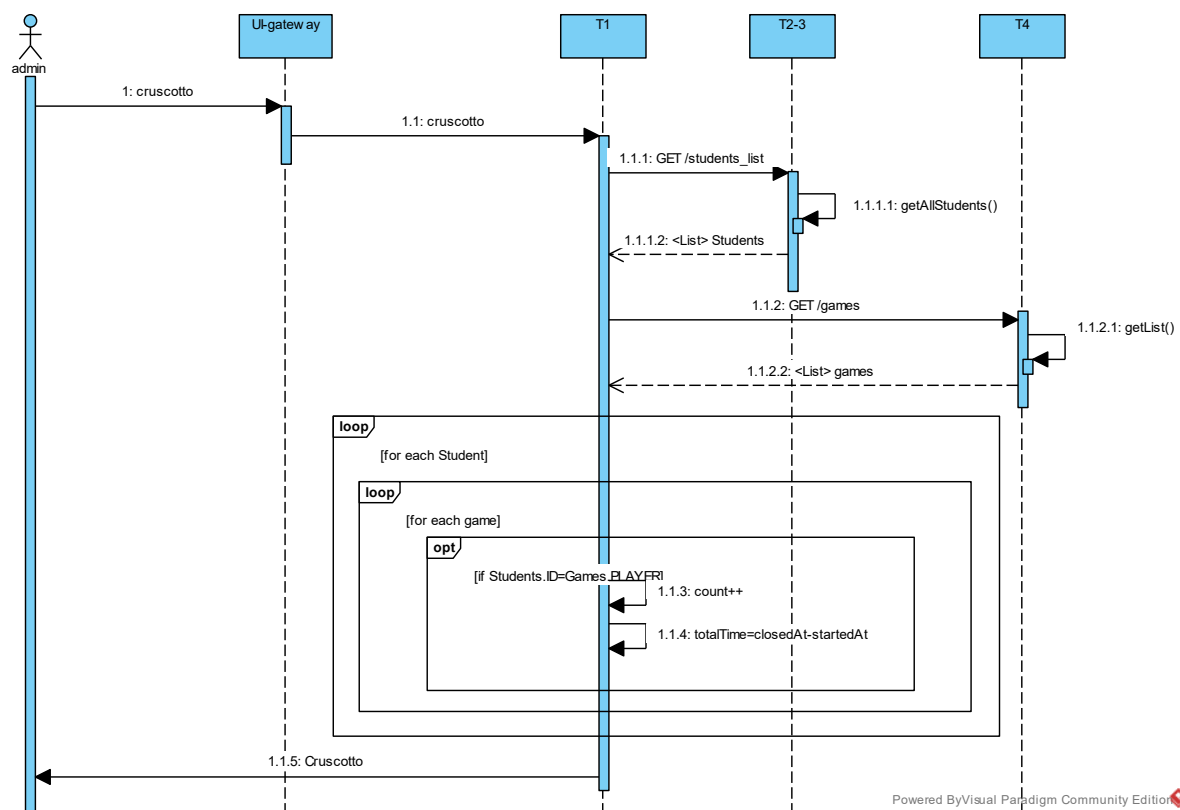


Figure 3.12: Realizzazione query nel task T1

3.5 Communication Diagram

Di seguito il communication Diagram tra i vari componenti nell'operazione di run:

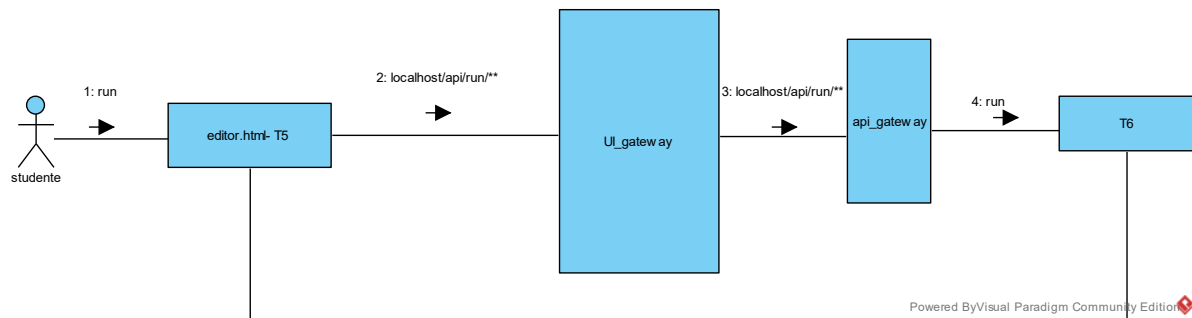


Figure 3.13: Communication Diagram per tasto run

Le chiamate tra i diversi microservizi avvengono attraverso il protocollo HTTP. Ogni richiesta effettuata da un microservizio è intercettata dall'UI-Gateway, il quale ha il compito di instradarla ai vari componenti.

Nella versione di partenza, per come sono strutturate le varie chiamate, l'applicazione, per forza di cose, può funzionare solo su localhost (indirizzo inchiodato nel codice).

E' stato necessario aggiornare, per tutte le funzionalità, questo tipo di chiamate.

La comunicazione dell'operazione run si svolgerà, quindi, come segue:

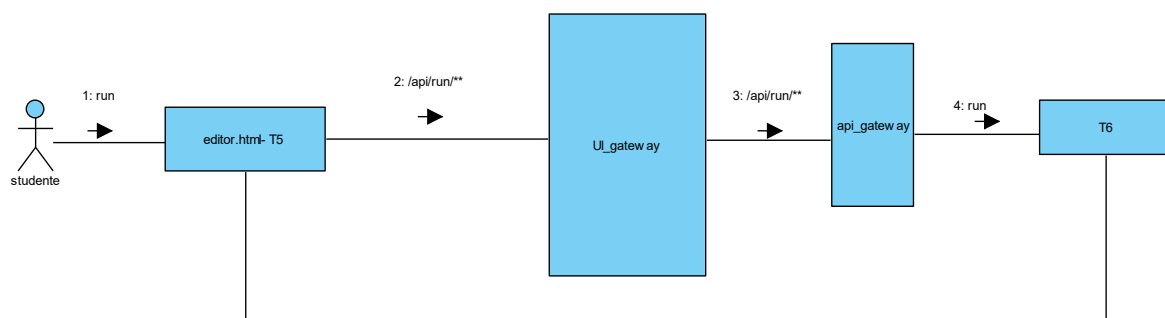


Figure 3.14: Communication Diagram dopo le modifiche

Nella versione attuale, le chiamate continuano ad utilizzare il protocollo HTTP, ma senza specificare nel codice la dipendenza da localhost. L'indirizzo viene dinamicamente aggiornato in base all'indirizzo della macchina in esecuzione, consentendo così la distribuzione dell'applicazione.

Deployment Diagram

In questo capitolo, si presentano diversi deployment diagram per una comprensione approfondita delle modifiche e del funzionamento di ogni soluzione. Per ciascun diagramma si mette in mostra la disposizione fisica dei componenti di un sistema software e la loro interazione all'interno dell'ambiente di esecuzione. L'applicazione è ospitata su un server che utilizza il sistema operativo Windows 11. Nello specifico, il software viene eseguito nell'ambiente Docker, che permette la distribuzione su più container, in modo da avere una certa indipendenza tra i vari task. I container evidenziati in giallo sono quelli che hanno subito modifiche.

4.1 Esposizione su Rete Pubblica

Il primo requisito è quello di rendere l'applicazione accessibile da qualsiasi macchina, non solo da localhost. Analizziamo il diagramma di deployment presentato nella versione precedente:

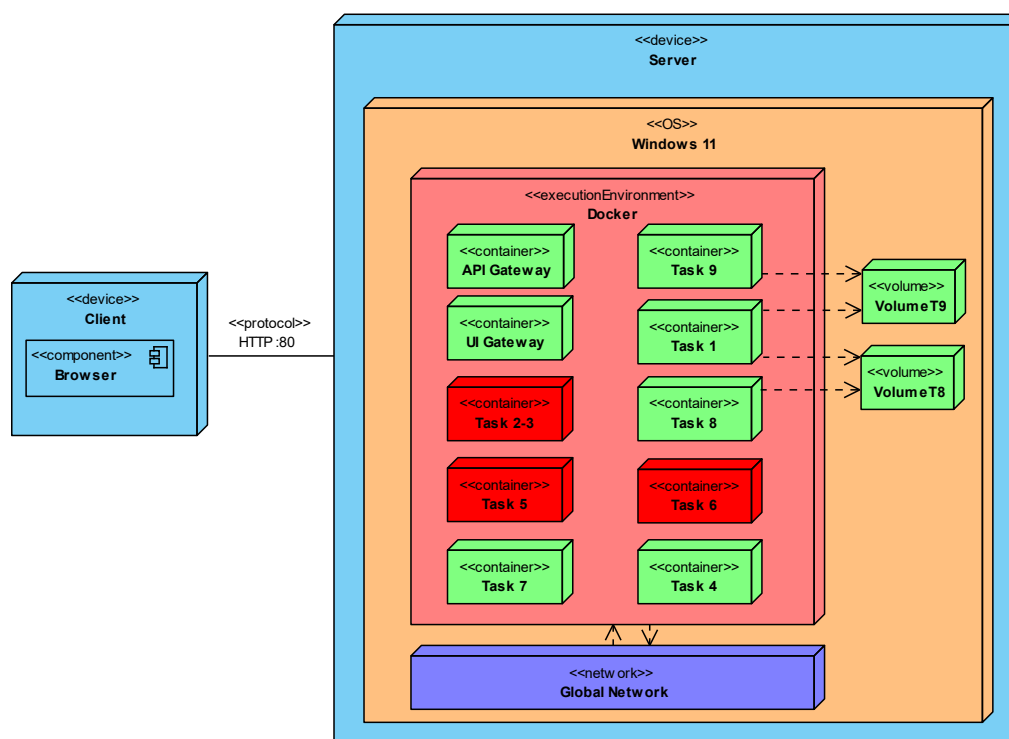


Figure 4.1: Diagramma di Deployment, in rosso i container modificati per esporre l'applicazione correttamente

Per l'esposizione su rete pubblica sono proposte due possibili soluzioni, che impattano sul diagramma di deployment. La prima è attraverso l'utilizzo di un servizio di Tunneling, mentre la seconda è tramite un server esterno.

4.1.1 Soluzione 1: Servizi di Tunneling

Il Tunneling è una tecnica che consente di instradare il traffico di rete attraverso una connessione crittografata, creando così un “tunnel” sicuro tra l’indirizzo locale e un indirizzo pubblico. Con questa tecnica è possibile in maniera rapida e sicura l’applicazione, facilitando il test delle funzionalità anche da remoto.

Un servizio che utilizza questa tecnica è [Ngrok](#), un reverse Proxy che si interfaccia direttamente con Docker.

Ngrok, con l’installazione (descritta nel paragrafo 7.2 Installazione Ngrok), crea un proprio container all’interno di Docker, fornisce un indirizzo pubblico e gestisce le richieste indirizzandole all’indirizzo locale.

Di seguito è illustrato il diagramma con questa soluzione.

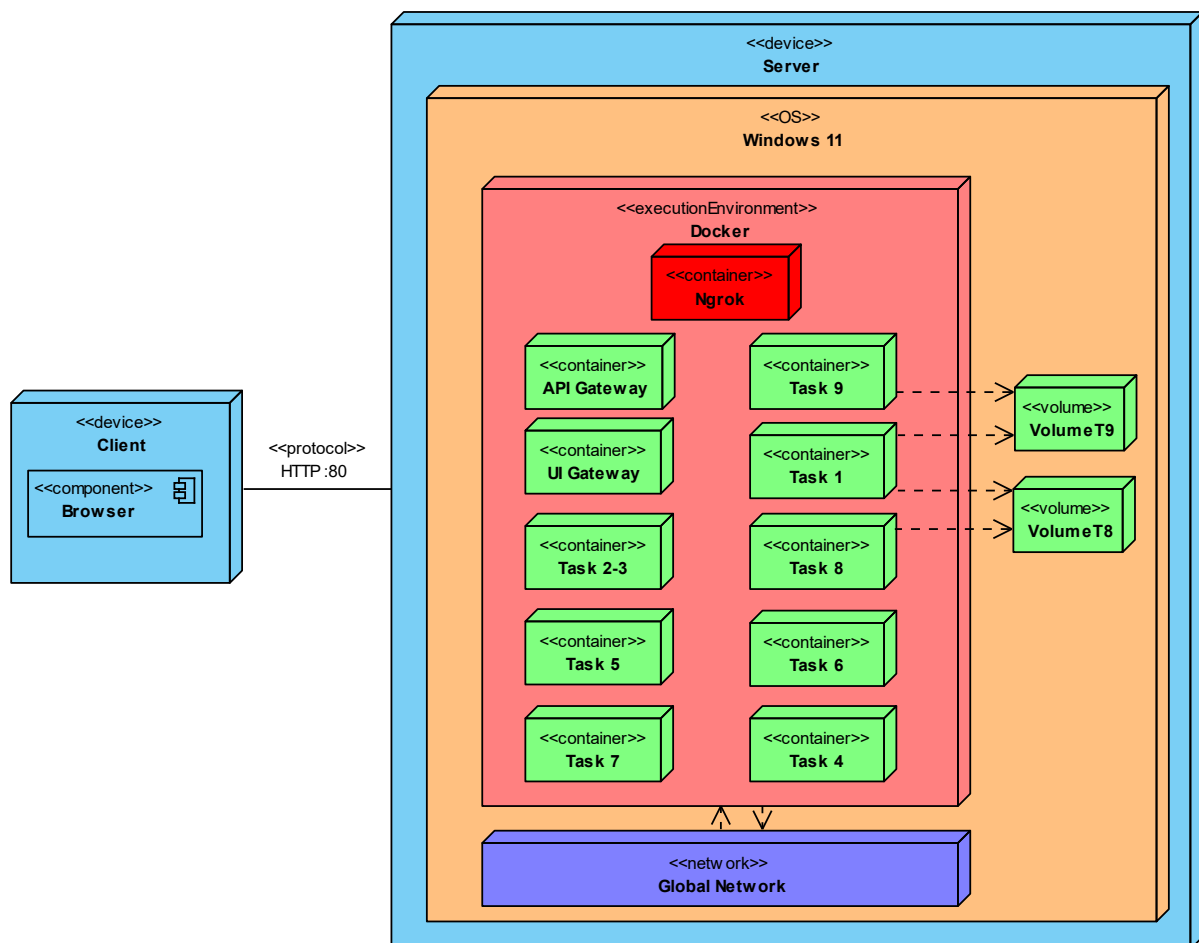


Figure 4.2: Diagramma di Deployment (Ngrok), in rosso i container modificati/aggiunti

Un’alternativa a Ngrok è [Pinggy](#), che offre la possibilità di avere un indirizzo pubblico per un tempo limitato pari a 60 minuti. Rispetto a Ngrok non è necessario iscriversi né installare nulla, ma con una sola riga di comando è possibile instradare localhost su un indirizzo pubblico temporaneo. Pinggy si basa sul protocollo SSH, e si interfaccia direttamente con l’indirizzo locale sul porto locale utilizzato dall’applicazione web.

Entrambe le soluzioni presentate sono da tenere in considerazione per testare l'applicazione web durante il suo sviluppo, ma non sono praticabili come soluzioni definitive.

ATTENZIONE: Si consiglia di non diffondere il link generato da Pinggy poiché all'interno è presente il proprio indirizzo IP pubblico. UTILE SOLAMENTE IN CASO DI TEST IN AMBIENTE CONTROLLATO!

4.1.2 Soluzione 2: Server esterno

La soluzione con l'appoggio ad un server esterno per la nostra applicazione comporta diversi vantaggi in termini di praticità e versatilità.

Possiamo concepire il server esterno come una macchina in esecuzione nel cloud, le operazioni di configurazione e installazione della nostra applicazione sono equivalenti a quelle eseguite sul nostro computer personale.

Il seguente diagramma illustra perfettamente la situazione:

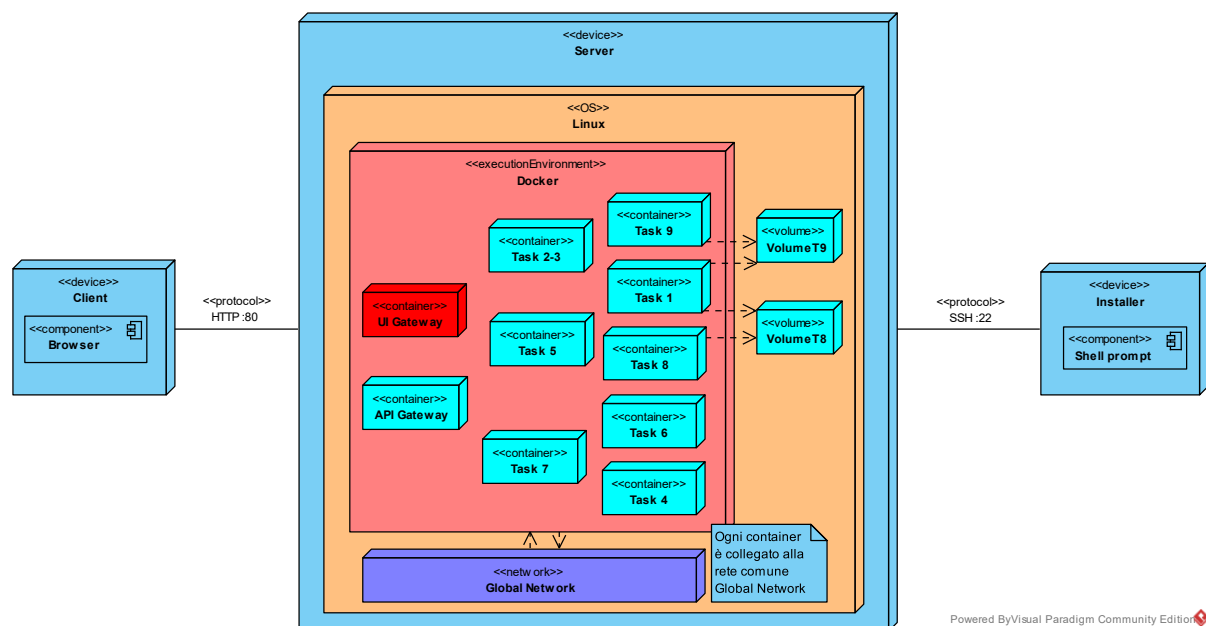


Figure 4.3: Diagramma di Deployment (server esterno), in rosso il punto di accesso al sistema

In una fase iniziale di configurazione, il collegamento al server esterno avviene tramite il protocollo SSH, utilizzando la riga di comando per impartire i giusti comandi al fine di installare correttamente l'applicazione (Si rimanda al Capitolo 7).

Una volta installata e configurata, l'UI-Gateway sarà in ascolto sul porto 80 all'indirizzo IP associato al server esterno. Il client (studente/admin) non dovrà fare altro che collegarsi a questo indirizzo IP per accedere all'applicazione.

Quando il client invia una richiesta HTTP all'indirizzo del server, l'UI-Gateway, come precedentemente spiegato, intercetta tutte le richieste in arrivo e le reindirizza al servizio

appropriato. Inoltre, le comunicazioni tra i vari task avvengono sempre attraverso chiamate HTTP, che quindi presenteranno (dinamicamente) l'indirizzo IP del server in esecuzione.

Quindi, al passo successivo, sarà il microservizio a prendere il posto del client ed effettuare la chiamata HTTP.

È responsabilità dell'UI-Gateway, che comunica con i diversi container attraverso la GLOBAL NETWORK condivisa, instradare la richiesta al servizio corretto.

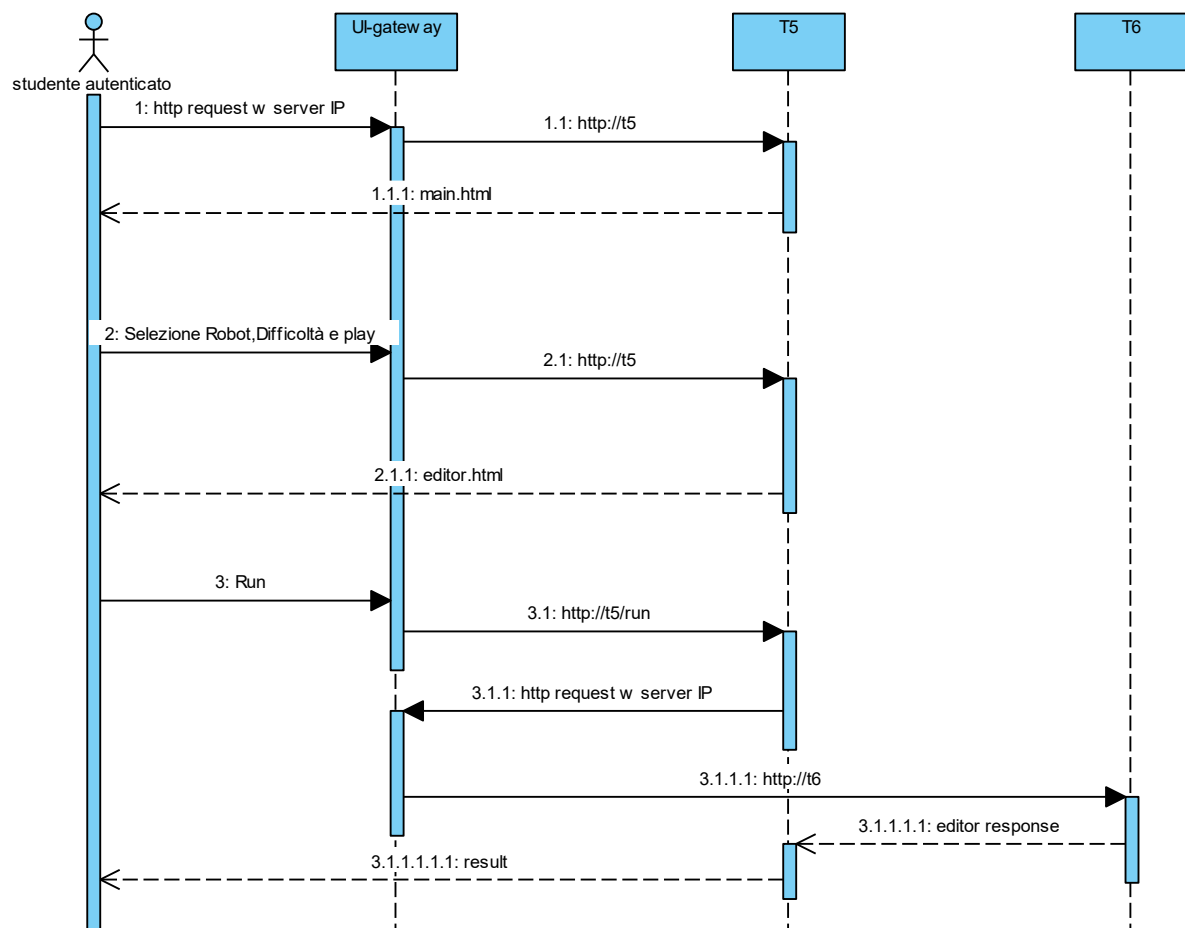


Figure 4.4: Sequence Diagram interazioni tra i componenti

Il sequence diagram, semplificato, illustra come durante la fase di run/submit della partita, il Task 5 (T5) contatta il Task 6 (T6) "passando" attraverso l'UI-Gateway con chiamate http all'indirizzo IP del server (nodo in esecuzione).

Lo svantaggio principale nell'utilizzo di un server esterno è rappresentato dai costi associati, i quali possono essere irrilevanti, se non totalmente assenti, con l'adozione di un servizio di tunneling. Il servizio di tunneling può risultare particolarmente vantaggioso durante le fasi di sviluppo per rendere l'applicazione accessibile a tutti. Tuttavia, una volta rilasciata la versione finale, l'impiego di un server esterno si candida come la migliore delle scelte.

Implementazione

Per informazioni dettagliate e precise sulla manutenzione effettuata, evitando di appesantire questo documento con schermate di codice dispersive e non formali, si prega di far riferimento al documento allegato [Modifiche Effettuate.txt](#), dove abbiamo cercato di indicare in modo chiaro e conciso dove e come sono state apportate tutte le modifiche che riguardano i requisiti esposti in questo capitolo.

5.1 Applicazione distribuita

Per adempiere a questo primo requisito, come evidenziato chiaramente dal diagramma di comunicazione nel capitolo 3, è fondamentale rivedere la progettazione di tutti i messaggi scambiati tra le diverse entità quando si rivolgono a un altro microservizio. In particolare, l'invocazione dei vari servizi avviene attraverso una chiamata a servizi tramite un URL all'indirizzo localhost.

Al fine di eliminare queste dipendenze da localhost, abbiamo re-implementato tutte le chiamate. In particolare, abbiamo convertito tutte le chiamate fetch in chiamate AJAX, e gli indirizzi non contengono più il riferimento a localhost. Ora le invocazioni alle API avvengono correttamente utilizzando le rotte gestite dall' UI-gateway (vedi Figure 4.4). I task che hanno subito modifiche relative ai riferimenti a localhost sono stati principalmente T5, T2-3, T6 e i gateway.

Si rimanda ai [video](#) allegati che illustrano come l'applicazione riesce ad essere accessibile da diversi dispositivi tutti indipendenti e non connessi tra loro.

5.2 Cruscotto amministratore

Il seguente requisito richiede una manutenzione additiva, poiché nella versione precedente questa funzionalità non era disponibile. Seguendo le regole del progetto, abbiamo apportato leggere modifiche ai vari task (T2-3, T1, T4) per recuperare i dati necessari ed elaborarli correttamente prima di presentarli agli utenti finali, in questo caso, gli amministratori. Per soddisfare il requisito e presentare all'amministratore un elenco conforme alle specifiche, abbiamo scelto di implementare la soluzione utilizzando la libreria JavaScript *DataTables*. Questo ci consente di creare una lista o tabella ordinabile su diverse colonne, consentendo così di ordinare i giocatori in base ai loro punteggi e di visualizzare una classifica.

L' applicazione è già stata predisposta per ottenere i punti da ciascun giocatore, quando la funzionalità di salvataggio sarà disponibile sarà sufficiente decommentare alcune parti di codice per integrare la "reale classifica" del sistema.

Elenco Studenti

Show 10 entries

Search:

Posizione	Punti	Name	Surname	Email	Studies	Partite Giocate	Tempo Totale (min)
1	null	Giorgio	DiCostanzo	giorgi.dicostanzo@studenti.unina.it	MSc	3	1.04
2	null	Mario	Rossi	mariorossi@gmail.it	BSc	0	0.00

Showing 1 to 2 of 2 entries

Previous 1 Next

5.3 Ulteriori modifiche

In questo paragrafo sono presentate tutte le modifiche che abbiamo effettuato che non impattano l'architettura del sistema, quali miglioramenti grafici, ottimizzazioni e miglioramenti dell'esperienza d'uso.

1. Schermata di Caricamento

Nella versione precedente, durante il caricamento della classe da parte di un amministratore o durante l'esecuzione di qualsiasi funzione dell'editor da parte dello studente, mancava completamente un feedback grafico. Questo problema è stato risolto.

Description

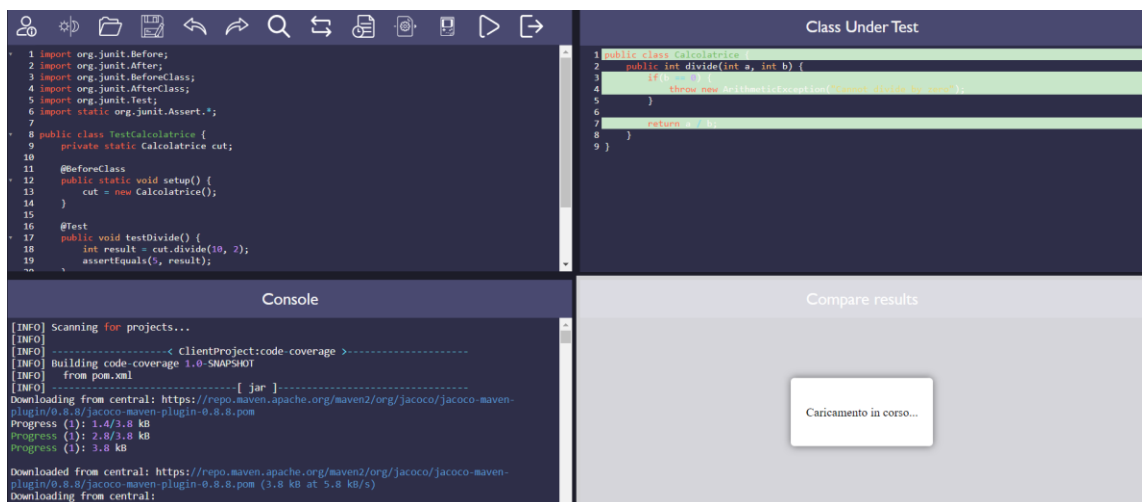
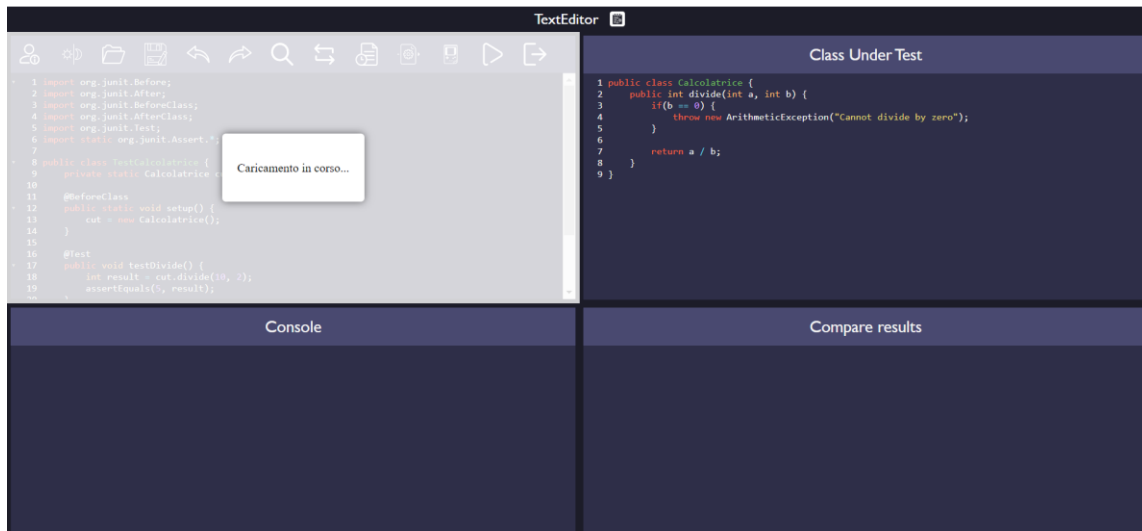
Category 1

Category 2

Category 3

Upload your file (.java) : Calcolatrice.java

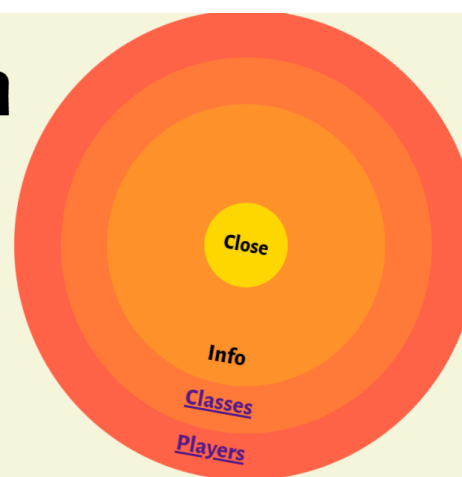
Caricamento in corso...



2. Menù Principale Admin

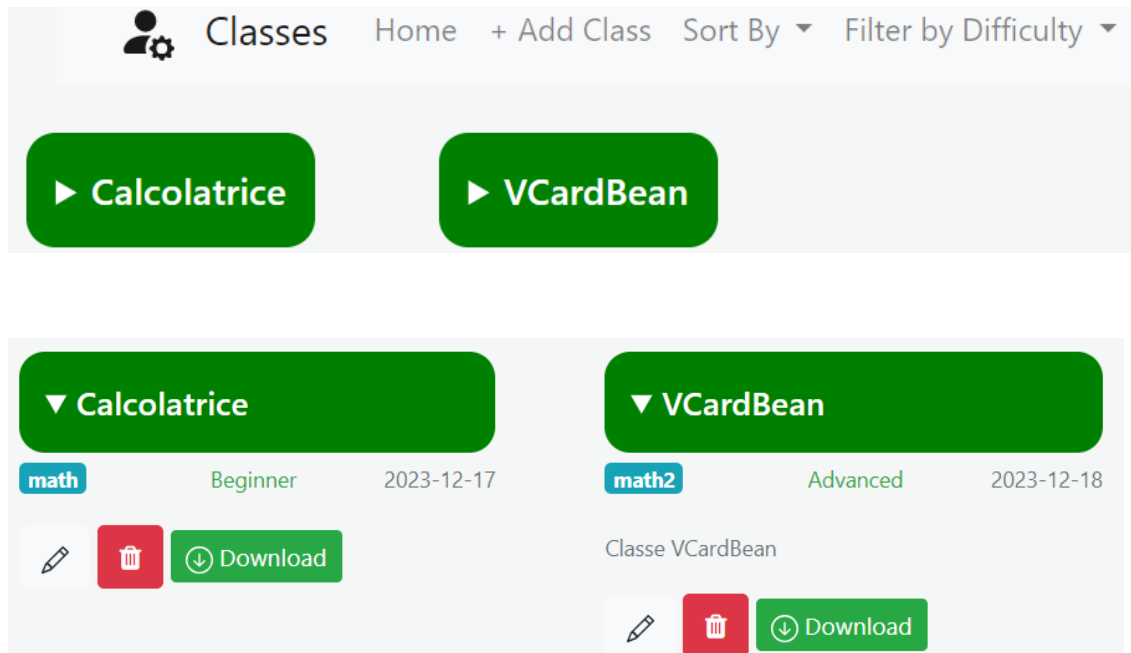
Il lato amministratore è stato migliorato con l'aggiunta di un menu principale che indirizza ai vari servizi che l'amministratore può utilizzare.

Home Admin
Welcome
Choose an option



3. Visualizzazione Compatta delle Classi

Nella versione precedente, la schermata di visualizzazione delle classi era dispersiva e poco scalabile, specialmente con un elevato numero di classi. L'interfaccia grafica è stata aggiornata per una visualizzazione più compatta delle classi.



4. Avviso su formattazione password

Precedentemente, durante la fase di registrazione, non vi era alcuna indicazione su come dovesse essere strutturata la password, lasciando all'intuito e alla fortuna la ricerca dei giusti requisiti. Da ora, ogni qual volta, si effettua la registrazione con una password non conforme il sistema ricorderà i requisiti necessari.

5. Divisione file nel task T5

Poiché il file "editor.html" è stato fortemente impattato dalle modifiche effettuate, si è presa la decisione di suddividerlo nei tre file tipici di una pagina web: "editor.html", "editor.css", e "editor.js".

6. Rotta /error

Al fine di ottenere un riscontro grafico per gli errori, è stata inserita una pagina HTML che viene visualizzata quando il gateway è indirizzato, per vari motivi, sul percorso /error.



7. Rimosso Captcha

La funzionalità del codice di verifica CAPTCHA durante la fase di registrazione è stata rimossa. Durante lo sviluppo, si è notato che, al di fuori dell'indirizzo localhost, questa funzionalità non è disponibile. Pertanto, la decisione è stata quella di eliminarla completamente.

8. Aumento Timeout del gateway

Durante la fase di caricamento delle classi, specialmente quelle più strutturate, la generazione dei test da parte dei robot può risultare onerosa in ottica temporale. Nella versione precedente dell'applicazione, il timeout delle richieste dell'ui-gateway era impostato a un valore relativamente breve, circa 5 minuti, il che risultava insufficiente per gestire correttamente le elaborazioni più pesanti.

Per affrontare questo problema e consentire una gestione più adeguata dei tempi di elaborazione, è stata apportata una modifica sostanziale. Il timeout delle richieste dell'ui-gateway è stato notevolmente esteso, portandolo a 50 minuti. Questa soluzione offre un margine di tempo più ampio per affrontare i casi in cui la generazione dei test richiede un periodo prolungato, garantendo così un funzionamento più robusto e efficiente dell'applicazione.

9. Rimozione Classe

Durante l'ultima integrazione, è stato rilevato un malfunzionamento nella funzionalità responsabile dell'eliminazione di una classe. In precedenza, la classe veniva eliminata dal database, ma le cartelle con i nomi delle classi nei Volumi condivisi T8 e T9 persistevano. In un successivo upload della stessa classe eliminata, il sistema restituiva giustamente un errore in quanto le cartelle nei volumi condivisi che riguardavano i test erano ancora presenti. Questo errore è stato corretto, e ora il sistema esegue il seguente scenario:

Caso d'uso Elimina Classe	
Attore primario	Registered Admin
Attore Secondario	-
Descrizione	Permette ad un amministratore registrato di eliminare una classe
Pre-condizioni	L'amministratore è loggato e la classe da eliminare è presente nel sistema
Sequenza di eventi principale	<ol style="list-style-type: none">1. L'admin richiede la rimozione di una classe2. Vengono eliminati i file e i riferimenti relativi alla classe dal database3. Vengono eliminati i file e le cartelle relative alla classe nei Volumi T8, T9
Post-Condizioni	Non vi sono più tracce della classe eliminata nell'intero sistema (Databases, Volumi e front-ends)

10. Installer Modificato

Nella versione precedente, dopo aver lanciato lo script di installazione, era necessario eseguire dei prompt manuali all'interno del Docker.

Inoltre, durante un'analisi più approfondita, è emerso un errore, probabilmente legato a un errore di trascrizione, nella creazione degli indici delle tabelle. Questo errore è stato corretto, e ora tutte le operazioni vengono eseguite in modo automatico durante l'installazione, senza richiedere prompt manuali.

11. Uninstaller creato

La mancanza di una procedura automatizzata che permettesse di disinstallare l'applicazione, soprattutto durante la fase di sviluppo e aggiunta modifiche, è diventata non trascurabile. È stata nostra cura aggiungere anche l'uninstaller del sistema.

Testing

Il Testing è una fase cruciale nello sviluppo software, consente di identificare eventuali problemi che possono verificarsi.

Si è effettuato sia un Testing manuale per identificare eventuali criticità nell'affidabilità e robustezza del sistema, sia un Testing automatico per capire come le soluzioni rispondono a un elevato numero di richieste.

Di seguito sono riportati i risultati derivanti dai Test.

6.1 Testing manuale

Per il testing manuale si sono provati i seguenti scenari:

- Login in contemporanea da più dispositivi
- Giocare in contemporanea da più dispositivi
- Login con lo stesso account da più dispositivi
- Giocare in contemporanea da più dispositivi
- Registrarsi in contemporanea

6.1.1 Login da più dispositivi per ogni provider

Il primo test effettuato prevede il solo scenario di Login da più dispositivi.

Username	Password
<ul style="list-style-type: none">• Stringa che può contenere: caratteri, numeri, punti• Contengono una chiocciola• finisce con suffissi convenzionali (.it, .com...)	<ul style="list-style-type: none">• Almeno un carattere maiuscolo• Almeno un carattere minuscolo• Almeno un numero• Lunghezza minima di 8 caratteri

Test Case ID	Descrizione	Classi di equivalenza coperte	Elemento di Sistema provider	Pre-condizioni	Input	Output attesi	Post-condizioni attesi	PASS/FAIL
1	Tutti input validi	Username valido Password valida	Ngrok	L'utente è regolarmente registrato al sistema	{Username:" gior.gi.dicostanzo@studenti.unina.it " Password:"Mario rossi12" }	Login effettuato con successo	Reindirizzamento alla pagina /main	PASS
2	Tutti input validi	Username valido Password valida	Pinggy	L'utente è regolarmente registrato al sistema	{Username:" gior.gi.dicostanzo@studenti.unina.it " Password:"Mario rossi12" }	Login effettuato con successo	Reindirizzamento alla pagina /main	PASS

3	Tutti input validi	Username valido Password valida	Server esterno	L'utente è regolarmente registrato al sistema	{Username:" gior.gi.dicostanzo@studenti.unina.it " Password:"Mario rossi12" }	Login effettuato con successo	Reindirizzament o alla pagina /main	PASS
4	Password non valida	Username valido Password non compatibile	ngrok	L'utente è regolarmente registrato al sistema	{Username:" gior.gi.dicostanzo@studenti.unina.it " Password:"abc" }	Password non corretta!	Messaggio di errore con password incorretta	PASS
5	Password non valida	Username valido Password non compatibile	pinggy	L'utente è regolarmente registrato al sistema	{Username:" gior.gi.dicostanzo@studenti.unina.it " Password:"abc" }	Password non corretta!	Messaggio di errore con password incorretta	PASS
6	Password non valida	Username valido Password non compatibile	Server esterno	L'utente è regolarmente registrato al sistema	{Username:" gior.gi.dicostanzo@studenti.unina.it " Password:"abc" }	Password non corretta!	Messaggio di errore con password incorretta	PASS
7	Username non valido	Username non valido Password -	Ngrok	L'utente non è registrato al sistema	{Username:"abc" Password:"abc" }	Email not found	Messaggio di errore con Email non trovata	PASS
8	Username non valido	Username non valido Password -	Pinggy	L'utente non è registrato al sistema	{Username:"abc" Password:"abc" }	Email not found	Messaggio di errore con Email non trovata	PASS
9	Username non valido	Username non valido Password -	Server esterno	L'utente non è registrato al sistema	{Username:"abc" Password:"abc" }	Email not found	Messaggio di errore con Email non trovata	PASS

6.1.2 Giocare in contemporanea da dispositivi diversi

Il secondo scenario prevede che utenti diversi possano giocare da dispositivi diversi in remoto.

Codice

- Tipo stringa
- Segue le regole della sintassi del linguaggio (in questo caso java)

Test Case ID	Descrizione	Classi di equivalenza coperte	Elemento di Sistema provider	Pre-condizioni	Input	Output attesi	Post-condizioni attesi	PASS/FAIL
1	Codice del test vincente per entrambi i giocatori	Vittoria contro il robot per entrambi i giocatori	Ngrok	Gli utenti sono regolarmente loggati al sistema, hanno scelto la classe calcolatrice ed EvoSuite come robot da sfidare	Utente1:{Codice: TestCalcolatrice.java }; Utente2:{Codice: TestCalcolatrice.java }	Entrambi vincitori	Messaggio di vittoria, con esito dei risultati per tutti e due i giocatori	PASS
2	Codice del test vincente per entrambi i giocatori	Vittoria contro il robot per entrambi i giocatori	Pinggy	Gli utenti sono regolarmente loggati al sistema, hanno scelto la classe calcolatrice ed EvoSuite come robot da sfidare	Utente1:{Codice: TestCalcolatrice.java }; Utente2:{Codice: TestCalcolatrice.java }	Entrambi vincitori	Messaggio di vittoria, con esito dei risultati per tutti e due i giocatori	PASS
3	Codice del test da vittoria per entrambi i giocatori	Vittoria contro il robot per entrambi i giocatori	Server Esterno	Gli utenti sono regolarmente loggati al sistema, hanno scelto la classe calcolatrice ed EvoSuite come robot da sfidare	Utente1:{Codice: TestCalcolatrice.java }; Utente2:{Codice: TestCalcolatrice.java }	Entrambi vincitori	Messaggio di vittoria, con esito dei risultati per tutti e due i giocatori	PASS
4	Codice del primo utente vincente Codice del secondo utente perdente	Vittoria per l'utente 1 e sconfitta per l'utente 2	Ngrok	Gli utenti sono regolarmente loggati al sistema, hanno scelto la classe calcolatrice ed EvoSuite come robot da sfidare valida	Utente1:{Codice: TestCalcolatrice_1.java }; Utente2:{Codice: TestCalcolatrice_1.java }	Utente1 Vincitore Utente2 sconfitto	Messaggio di vittoria, con esito dei risultati per Utente1 Messaggio di sconfitta per Utente2	PASS
5	Codice del primo utente vincente Codice del secondo utente perdente	Vittoria per l'utente 1 e sconfitta per l'utente 2	Pinggy	Gli utenti sono regolarmente loggati al sistema, hanno scelto la classe calcolatrice ed EvoSuite come robot da sfidare	Utente1:{Codice: TestCalcolatrice_1.java }; Utente2:{Codice: TestCalcolatrice_1.java }	Utente1 Vincitore Utente2 sconfitto	Messaggio di vittoria, con esito dei risultati per Utente1 Messaggio di sconfitta per Utente2	PASS
6	Codice del primo utente vincente Codice del secondo utente perdente	Vittoria per l'utente 1 e sconfitta per l'utente 2	Server Esterno	Gli utenti sono regolarmente loggati al sistema, hanno scelto la classe calcolatrice ed EvoSuite come robot da sfidare	Utente1:{Codice: TestCalcolatrice_1.java }; Utente2:{Codice: TestCalcolatrice_1.java }	Utente1 Vincitore Utente2 sconfitto	Messaggio di vittoria, con esito dei risultati per Utente1 Messaggio di sconfitta per Utente2	PASS

6.1.3 Login con lo stesso account da dispositivi diversi

Il terzo scenario che si è previsto è quello di effettuare il login da dispositivi diversi ma con lo stesso account.

Username	Password
<ul style="list-style-type: none">• Stringa che può contenere: caratteri, numeri, punti• Contengono una chiocciola• finisce con suffissi convenzionali (.it, .com...)	<ul style="list-style-type: none">• Almeno un carattere maiuscolo• Almeno un carattere minuscolo• Almeno un numero• Lunghezza minima di 8 caratteri

Test Case ID	Descrizione	Classi di equivalenza coperte	Elemento di Sistema provider	Pre-condizioni	Input	Output attesi	Post-condizioni attesi	PASS/FAIL
1	Tutti input validi e uguali per due dispositivi diversi	Username1 valido Password1 valida	Ngrok	L'utente è regolarmente registrato al sistema	{Username: "giorgi.dicos tanzo@stud enti.unina.it" Password:" Mariorossi1 2" }	Login effettuato con successo da entrambi i dispositivi	Reindirizzamento alla pagina /main su entrambi i dispositivi	PASS
2	Tutti input validi e uguali per due dispositivi diversi	Username1 valido Password1 valida	Pinggy	L'utente è regolarmente registrato al sistema	{Username: "giorgi.dicos tanzo@stud enti.unina.it" Password:" Mariorossi1 2" }	Login effettuato con successo da entrambi i dispositivi	Reindirizzamento alla pagina /main su entrambi i dispositivi	PASS
3	Tutti input validi e uguali per due dispositivi diversi	Username1 valido Password1 valida	Server Esterno	L'utente è regolarmente registrato al sistema	{Username: "giorgi.dicos tanzo@stud enti.unina.it" Password:" Mariorossi1 2" }	Login effettuato con successo da entrambi i dispositivi	Reindirizzamento alla pagina /main su entrambi i dispositivi	PASS

6.1.4 Giocare con lo stesso account contemporaneamente da due dispositivi diversi

Il quarto scenario è quello di giocare con lo stesso account contemporaneamente da due dispositivi diversi

Codice

- Tipo stringa
- Segue le regole della sintassi del linguaggio (in questo caso java)

Test Case ID	Descrizione	Classi di equivalenza coperte	Elemento di Sistema provider	Pre-condizioni	Input	Output attesi	Post-condizioni attesi	PASS/FAIL
1	Codice del test da vittoria sul primo dispositivo e sconfitta sul secondo dispositivo	Codice del test con cui si ha già vinto	Ngrok	L'utente è regolarmente registrato al sistema, entrambi scelgono EvoSuite come robot da sfidare e la compilazione del codice è risultata valida	Utente1:{Codice: TestCalcolatrice.java }; Utente2:{Codice: TestCalcolatrice.java }	Entrambi vincitori	Messaggio di vittoria sul primo dispositivo, e messaggio di sconfitta sul secondo dispositivo	PASS
2	Codice del test da vittoria sul primo dispositivo e sconfitta sul secondo dispositivo	Codice del test con cui si ha già vinto	Pinggy	L'utente è regolarmente registrato al sistema, entrambi scelgono EvoSuite come robot da sfidare e la compilazione del codice è risultata valida	Utente1:{Codice: TestCalcolatrice.java }; Utente2:{Codice: TestCalcolatrice.java }	Entrambi vincitori	Messaggio di vittoria sul primo dispositivo, e messaggio di sconfitta sul secondo dispositivo	PASS
3	Codice del test da vittoria sul primo dispositivo e sconfitta sul secondo dispositivo	Codice del test con cui si ha già vinto	Server Esterno	L'utente è regolarmente registrato al sistema, entrambi scelgono EvoSuite come robot da sfidare e la compilazione del codice è risultata valida	Utente1:{Codice: TestCalcolatrice.java }; Utente2:{Codice: TestCalcolatrice.java }	Entrambi vincitori	Messaggio di vittoria sul primo dispositivo, e messaggio di sconfitta sul secondo dispositivo	PASS

6.1.5 Registrazione in contemporanea

Il quinto scenario prevede la registrazione in contemporanea di più utenti su reti diverse.

Indirizzo e-mail	Password	Nome	Cognome	Studi
<ul style="list-style-type: none"> Stringa che può contenere: caratteri, numeri, punti Contengono una chiocciola finisce con suffissi convenzionali (.it, .com...) 	<ul style="list-style-type: none"> Almeno un carattere maiuscolo Almeno un carattere minuscolo Almeno un numero Lunghezza minima di 8 caratteri 	<ul style="list-style-type: none"> Stringa di soli caratteri Senza spazi 	<ul style="list-style-type: none"> Stringa di soli caratteri Senza spazi 	<ul style="list-style-type: none"> Può assumere solo uno dei tre valori: BSc, MSc o Altro.

Test Case ID	Descrizione	Classi di equivalenza coperte	Elemento di Sistema provider	Pre-condizioni	Input	Output attesi	Post-condizioni attesi	PASS/FAIL
1	Tutti gli input sono validi	E-mail: valida Password: valida Nome: valido Cognome: valido Studi: valido	Ngrok	Gli utenti riescono ad accedere alla pagina regolarmente	Utente1: {E-mail: "giorgi.dicostanzo@studenti.unina.it" Password: Mariorossi12 Nome: Giorgio Cognome: DiCostanzo Studi: Msc } Utente2: { "mario@gmail.it" Password: Mariorossi12 Nome: Mario Cognome: Rossi Studi: Bsc }	Registrazione effettuata con successo	Messaggio di avvenuta registrazione e ricezione email di avvenuta registrazione	PASS
2	Tutti gli input sono validi	E-mail: valida Password: valida Nome: valido Cognome: valido Studi: valido	Pinggy	L'utente riesce ad accedere alla pagina regolarmente	Utente1: {E-mail: "giorgi.dicostanzo@studenti.unina.it" Password: Mariorossi12 Nome: Giorgio Cognome: DiCostanzo Studi: Msc } Utente2: { "mario@gmail.it"	Registrazione effettuata con successo	Messaggio di avvenuta registrazione e ricezione email di avvenuta registrazione	PASS

					l.it" Password: Mariorossi12 Nome: Mario Cognome: Rossi Studi: Bsc }			
3	Tutti gli input sono validi	E-mail: valida Password: valida Nome: valido Cognome: valido Studi: valido	Server Esterno	L'utente riesce ad accedere alla pagina regolarmente	Utente1: {E-mail: "giorgi.dicostanzo@studenti.unina.it" Password: Mariorossi12 Nome: Giorgio Cognome: DiCostanzo Studi: Msc } Utente2: { "mario@gmail.l.it" Password: Mariorossi12 Nome: Mario Cognome: Rossi Studi: Bsc }	Registrazione effettuata con successo	Messaggio di avvenuta registrazione e ricezione email di avvenuta registrazione	PASS
4	Tutti gli input sono validi	E-mail: valida Password: valida Nome: valido Cognome: valido Studi: valido	Ngrok	Gli utenti riescono ad accedere alla pagina regolarmente	Utente1: {E-mail: "giorgi.dicostanzo@studenti.unina.it" Password: Mariorossi12 Nome: Giorgio Cognome: DiCostanzo Studi: Msc } Utente2: {E-mail: "giorgi.dicostanzo@studenti.unina.it" Password: Mariorossi12 Nome: Giorgio Cognome: DiCostanzo Studi: Msc }	Registrazione effettuata con successo per un solo dispositivo, Registrazione non effettuata per il secondo utente	Messaggio di avvenuta registrazione e ricezione email di avvenuta registrazione per un solo dispositivo, messaggio di Email già in uso per il secondo dispositivo	PASS
5	Tutti gli input sono validi	E-mail: valida Password: valida Nome: valido Cognome: valido Studi: valido	Pinggy	Gli utenti riescono ad accedere alla pagina regolarmente	Utente1: {E-mail: "giorgi.dicostanzo@studenti.unina.it" Password: Mariorossi12 Nome:	Registrazione effettuata con successo per un solo dispositivo	Messaggio di avvenuta registrazione e ricezione email di avvenuta registrazione per un solo dispositivo,	PASS

					Giorgio Cognome: DiCostanzo Studi: Msc } Utente2: {E-mail: "giorgi.dicostanzo@studenti.unina.it" Password: Mariorossi12 Nome: Giorgio Cognome: DiCostanzo Studi: Msc }	o, Registrazione non effettuata per il secondo utente	messaggio di Email già in uso per il secondo dispositivo	
6	Tutti gli input sono validi	E-mail: valida Password: valida Nome: valido Cognome: valido Studi: valido	Server Esterno	Gli utenti riescono ad accedere alla pagina regolarmente	Utente1: {E-mail: "giorgi.dicostanzo@studenti.unina.it" Password: Mariorossi12 Nome: Giorgio Cognome: DiCostanzo Studi: Msc } Utente2: {E-mail: "giorgi.dicostanzo@studenti.unina.it" Password: Mariorossi12 Nome: Giorgio Cognome: DiCostanzo Studi: Msc }	Registrazione effettuata con successo per un solo dispositivo o, Registrazione non effettuata per il secondo utente	Messaggio di avvenuta registrazione e ricezione email di avvenuta registrazione per un solo dispositivo, messaggio di Email già in uso per il secondo dispositivo	PASS

6.2 Testing automatico

Il testing automatico consente l'esecuzione rapida ed efficiente di diversi test. Per effettuare tali test è stato utilizzato il software Postman il quale permette in maniera automatica di effettuare richieste su applicazioni web di tipo GET, POST, DELETE, PUT, HEAD, PATCH e OPTIONS.

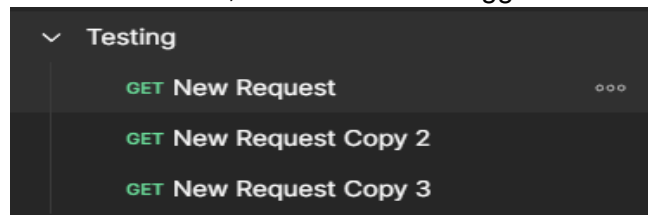
Il testing non è stato effettuato sul server esterno ma solo su pinggy e ngrok. Su ognuna di queste due soluzioni sono stati effettuati tre test con queste caratteristiche:

- 5 Utenti Virtuali per 3 minuti con richieste di tipo GET
- 20 Utenti Virtuali per 5 minuti con richieste di tipo GET
- 100 Utenti Virtuali per 10 minuti con richieste di tipo GET

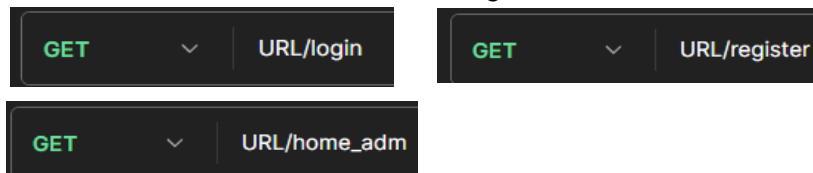
Si testa quindi quanto le soluzioni proposte riescano a rispondere in maniera efficiente a un numero di richieste sempre più elevato.

L'ambiente Postman è stato settato nel seguente modo:

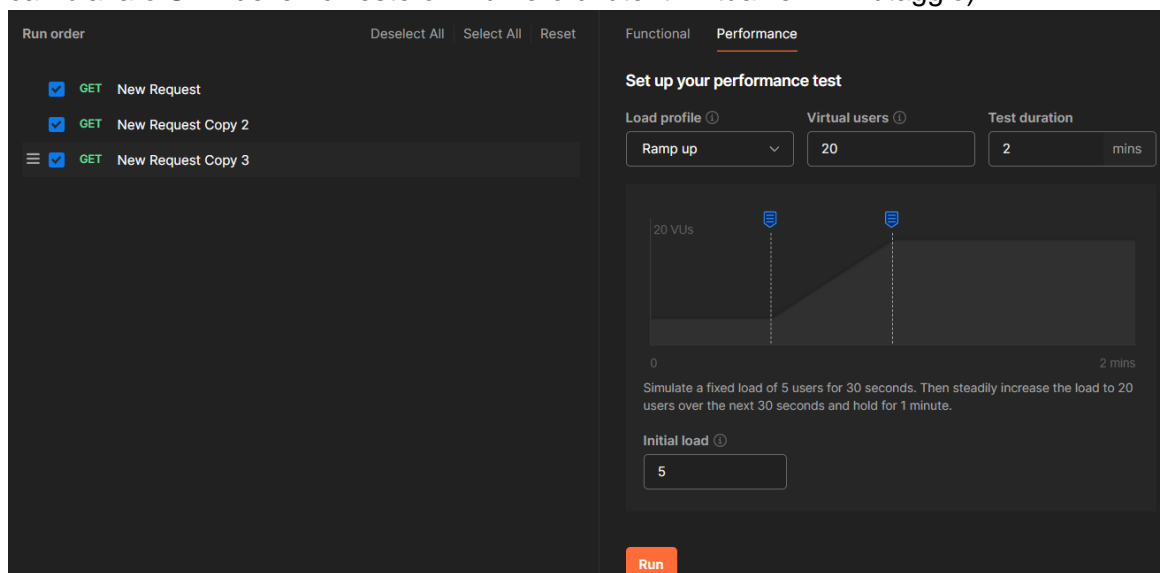
- Creazione di una nuova Collection, in cui sono state aggiunte tre richieste di GET:



- Le richieste di GET sono state settate nel seguente modo:



- I test sono stati settati nel seguente modo (il riferimento è solo al primo, per gli altri cambiava o URL delle richieste o il numero di utenti virtuali e il minutaggio):



Di seguito è riportata la tabella su come siano stati settati tutti i test automatici.

Utenti Virtuali	Richiesta	URL	Tempo	Profilo di carico*
<ul style="list-style-type: none"> Numero intero che va da 1 a 100 	<ul style="list-style-type: none"> Può essere una delle seguenti: GET, POST, PUT, DELETE, PATCH o OPTIONS 	<ul style="list-style-type: none"> Stringa inizia con il tipo di protocollo (http o https) Nessuna lettera maiuscola Nessun carattere speciale Nessuno spazio formato standard: protocollo://dominio_o_ip/altro 	<ul style="list-style-type: none"> Stringa di soli caratteri Senza spazi 	<ul style="list-style-type: none"> Può essere di questo tipo: Fixed, Ramp Up, Spike

Test Case ID	Tipo di richiesta	Utenti Virtuali	Tempo	Elemento di Sistema provider	URL	Profilo di carico
1	GET	5	3 minuti	Ngrok	"URL_Ngrok/home_adm" "URL_Ngrok/login" "URL_Ngrok/register"	Ramp-up
2	GET	5	3 minuti	Pinggy	"URL_Pinggy/home_adm" "URL_Pinggy/login" "URL_Pinggy/register"	Ramp-up
3	GET	20	5 minuti	Ngrok	"URL_Ngrok/home_adm" "URL_Ngrok/login" "URL_Ngrok/register"	Ramp-up
4	GET	20	5 minuti	Pinggy	"URL_Pinggy/home_adm" "URL_Pinggy/login" "URL_Pinggy/register"	Ramp-up
5	GET	100	10 minuti	Ngrok	"URL_Ngrok/home_adm" "URL_Ngrok/login" "URL_Ngrok/register"	Ramp-up
6	GET	100	10 minuti	Pinggy	"URL_Pinggy/home_adm" "URL_Pinggy/login" "URL_Pinggy/register"	Ramp-up

* Il profilo di carico in Postman rappresenta come il carico su un'applicazione o servizio API varia nel corso del tempo durante i test di carico. I tre tipi principali di profili di carico sono:

1. **Fixed (Fisso):** Il numero di richieste al secondo rimane costante per tutta la durata del test.
2. **Ramp Up (Graduale):** Il carico aumenta gradualmente nel corso del tempo, simulando un incremento graduale di utenti o richieste.
3. **Spike (Picco):** Si verifica un repentino aumento del carico per un periodo limitato, simulando picchi improvvisi di traffico.

6.2.1 Risultati Test Ngrok

Di seguito sono presentati i test effettuati su Ngrok.

TEST 1

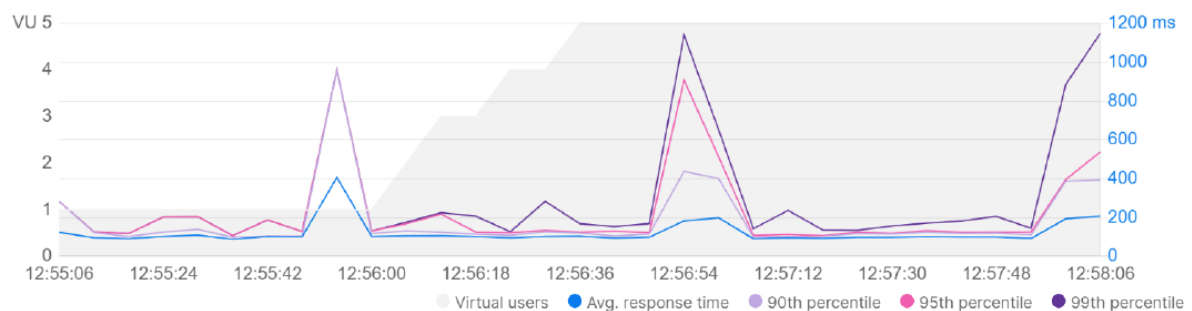
Risultati:

1. Summary

Total requests sent	Throughput	Average response time	Error rate
1,341	6.86 requests/second	117 ms	0.00 %

1.1 Response time

Response time trends during the test duration.



TEST 3

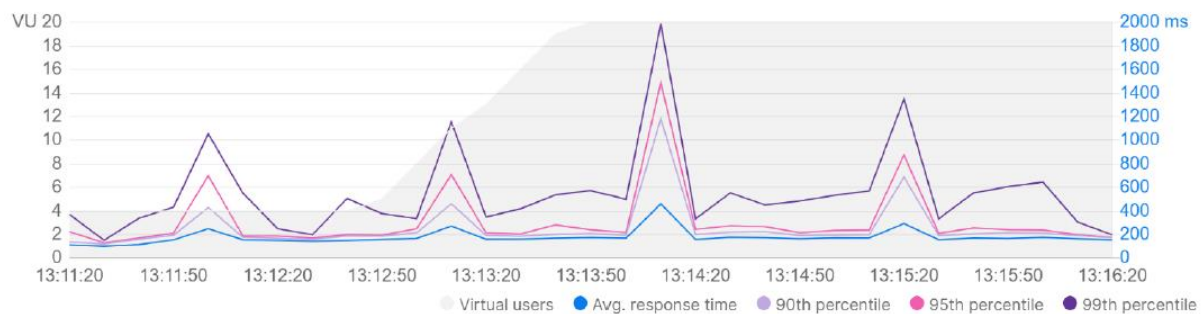
Risultati:

1. Summary

Total requests sent	Throughput	Average response time	Error rate
7,378	23.26 requests/second	184 ms	0.01 %

1.1 Response time

Response time trends during the test duration.



TEST 5

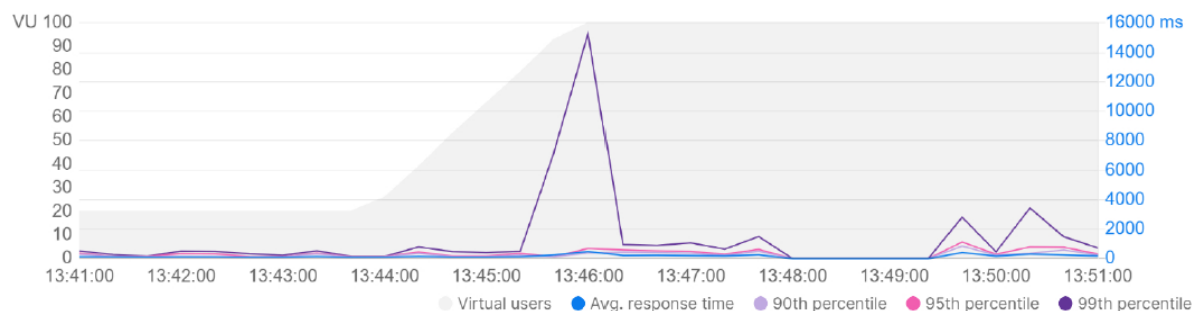
Risultati:

1. Summary

Total requests sent	Throughput	Average response time	Error rate
35,054	56.85 requests/second	219 ms	48.13 %

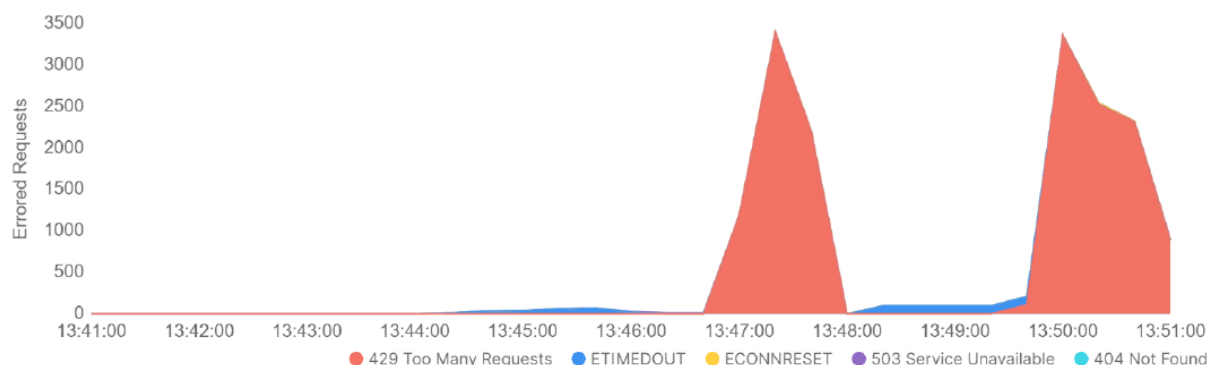
1.1 Response time

Response time trends during the test duration.



3.1 Error distribution over time

Top 5 error classes observed during the test duration.



A seguito dei risultati dei test si evince come la soluzione funzioni ottimamente con un basso numero di richieste e un basso numero di utenti. Con l'aumentare del numero di richieste

l'applicazione risponde più lentamente, inoltre si evince che non si può avere un certo numero di richieste in un ristretto lasso di tempo.

6.2.2 Risultati Test Pinggy

Di seguito sono presentati i test effettuati su Pinggy.

TEST 2

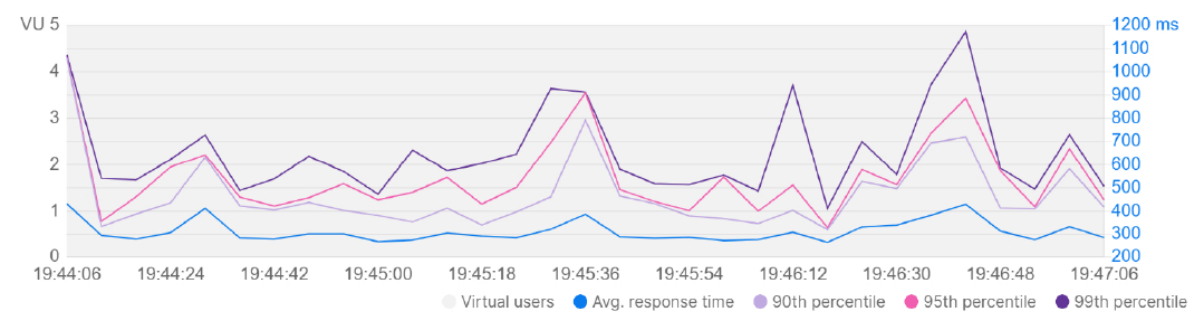
Risultati:

1. Summary

Total requests sent	Throughput	Average response time	Error rate
1,276	6.27 requests/second	309 ms	9.40 %

1.1 Response time

Response time trends during the test duration.



3.1 Error distribution over time

Top 5 error classes observed during the test duration.



TEST 4

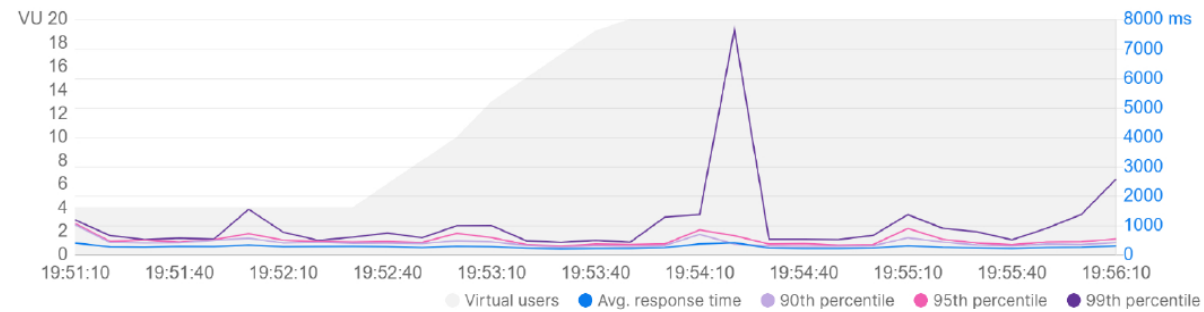
Risultati:

1. Summary

Total requests sent	Throughput	Average response time	Error rate
5,833	17.91 requests/second	276 ms	3.09 %

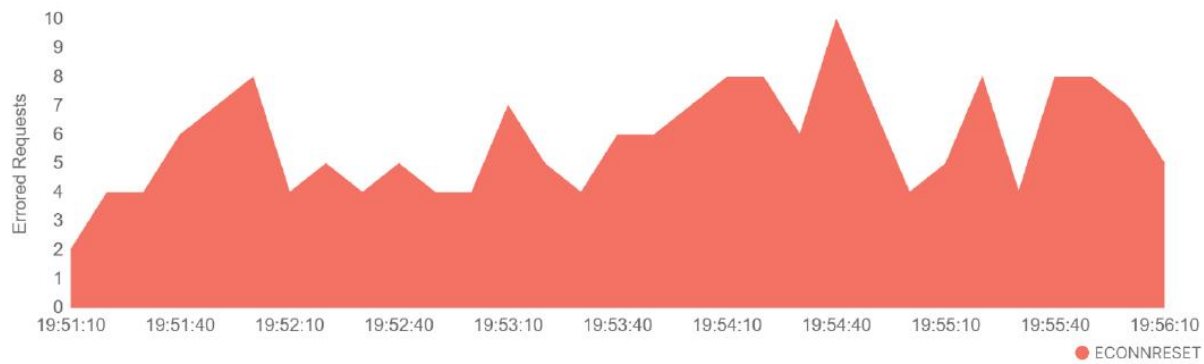
1.1 Response time

Response time trends during the test duration.



3.1 Error distribution over time

Top 5 error classes observed during the test duration.



TEST 6

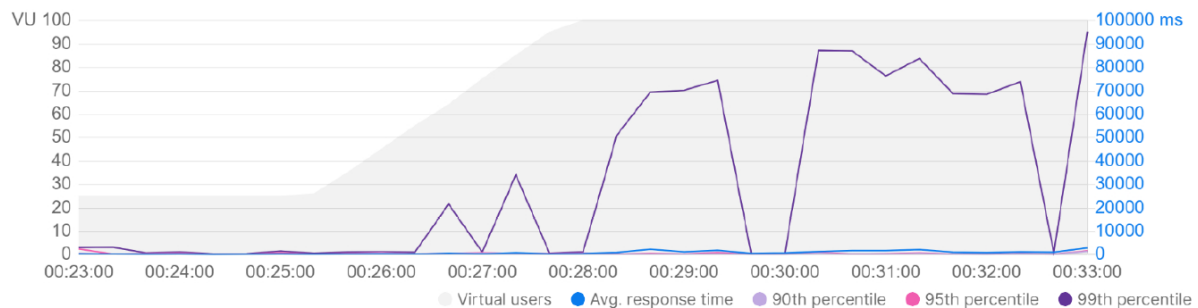
Risultati:

1. Summary

Total requests sent	Throughput	Average response time	Error rate
29,568	48.06 requests/second	924 ms	0.88 %

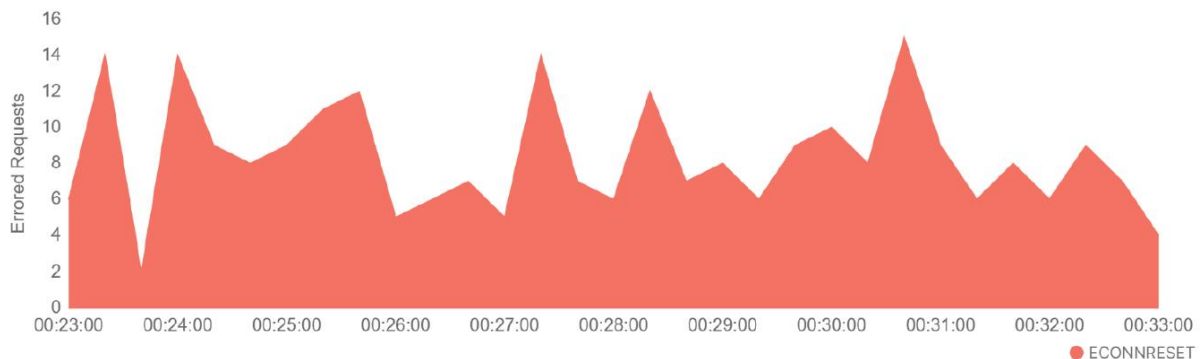
1.1 Response time

Response time trends during the test duration.



3.1 Error distribution over time

Top 5 error classes observed during the test duration.



Dai test si evince come abbia risultati inferiori a Ngrok in generale, con un tempo di risposta più lento e un margine di errore presente spesso, ma non si è arrivati al punto in cui il sistema ha smesso di funzionare come Ngrok.

Installazione

Di seguito è riportata la procedura di installazione automatica dell'applicazione. All'avvio del file "installer.bat" i Task vengono installati uno alla volta seguendo i file di configurazione "docker-compose.yml". Per maggiori dettagli si rimanda alla lettura del file [README](#).

7.1 Installazione Windows/Unix

Avviare lo script "installer.bat" o "installer.sh". Saranno effettuate le seguenti operazioni:

1. creazione della rete "global-network" comune a tutti i container
2. creazione del volume "VolumeT9" comune ai Task 1 e 9
3. creazione del volume "VolumeT8" comune ai Task 1 e 8
4. installazione di ogni singolo container
5. esecuzione dei file di installazione nei container del task T8 e T7
6. avvio dei container
7. **Comandi di inizializzazione del db del task T1**

7.2 Installazione Ngrok

L'installazione è la seguente:

1. **Registrazione:** Ci si registra sul sito ufficiale.
2. **Accesso alla Dashboard:** <https://dashboard.ngrok.com/get-started>.
3. **Scelta agente:** Si consiglia di scegliere Docker ma è possibile scegliere anche il proprio sistema operativo.
4. **Esecuzione del comando di installazione:** Copia il comando e esegilo mentre Docker è in funzione (Si consiglia di scegliere il dominio statico).

Installation

Install ngrok via Docker with the following command:

```
$ docker pull ngrok/ngrok
```



Deploy your app online

Ephemeral Domain Static Domain

Deploy with your static domain!

```
$ docker run --net=host -it -e NGROK_AUTHTOKEN= ngrok/ngrok:latest h
```



5. **Creazione nuovo container ed installazione:** Dopo l'esecuzione del comando, verrà installato un nuovo container all'interno di Docker. L'utilizzo di un dominio statico è consigliato per evitare la creazione di un nuovo container ogni volta che Ngrok viene avviato con un nuovo dominio.

6. **Visualizzazione del dominio associato:** Nel prompt dei comandi comparirà questa schermata dove specifica il dominio al quale accedere:

```
Version      3.5.0
Region      Europe (eu)
Latency      1251ms
Web Interface http://0.0.0.0:4040
Forwarding   https://horse-causal-basically.ngrok-free.app -> http://localhost:80

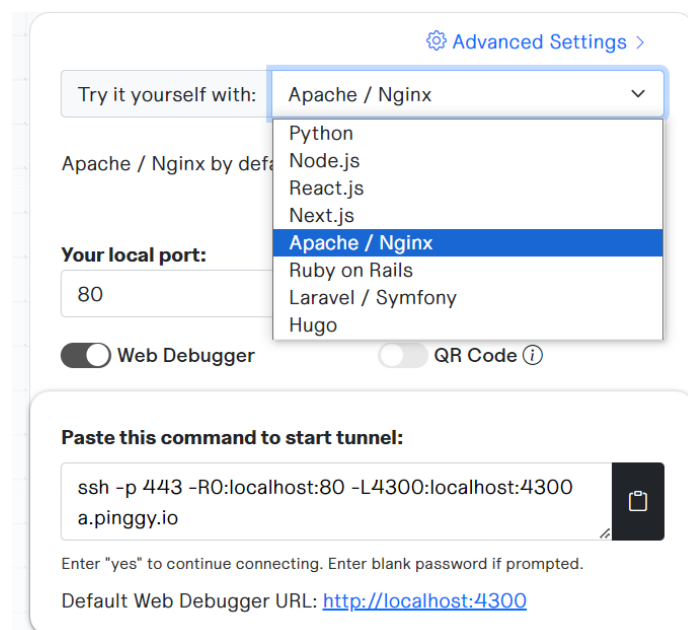
Connections  ttl    opn    rt1    rt5    p50    p90
              0      0      0.00   0.00   0.00   0.00
```

Nel caso il **container di Ngrok** sia **già installato**, si può avviare direttamente il container e collegarsi all'indirizzo fornito durante l'installazione.

7.3 Installazione Pinggy

In alternativa si può installare Pinggy il quale ha un'installazione molto più rapida:

1. Accedere al sito: <https://pinggy.io/>
2. Selezionare Apache/Nginx e porta 80:



3. Copiare il comando ed inserirlo nel prompt dei comandi
4. Di seguito chiederà una password, premere direttamente invio senza scrivere nessun carattere.

```
C:\Users\Giorgio>ssh -p 443 -R0:localhost:80 -L4300:localhost:4300 a.pinggy.io
giorgio@a.pinggy.io's password:
```

5. Infine, fornisce gli indirizzi pubblici sui quali è possibile accedere:

```
You are not authenticated.
Your tunnel will expire in 60 minutes. Upgrade to Pinggy Pro to get unrestricted tunnels. https://dashboard.pinggy.io

http://rnrxv-93-34-36-98.a.free.pinggy.online
https://rnrxv-93-34-36-98.a.free.pinggy.online
http://rnrxv-93-34-36-98.a.free.pinggy.link
https://rnrxv-93-34-36-98.a.free.pinggy.link
```

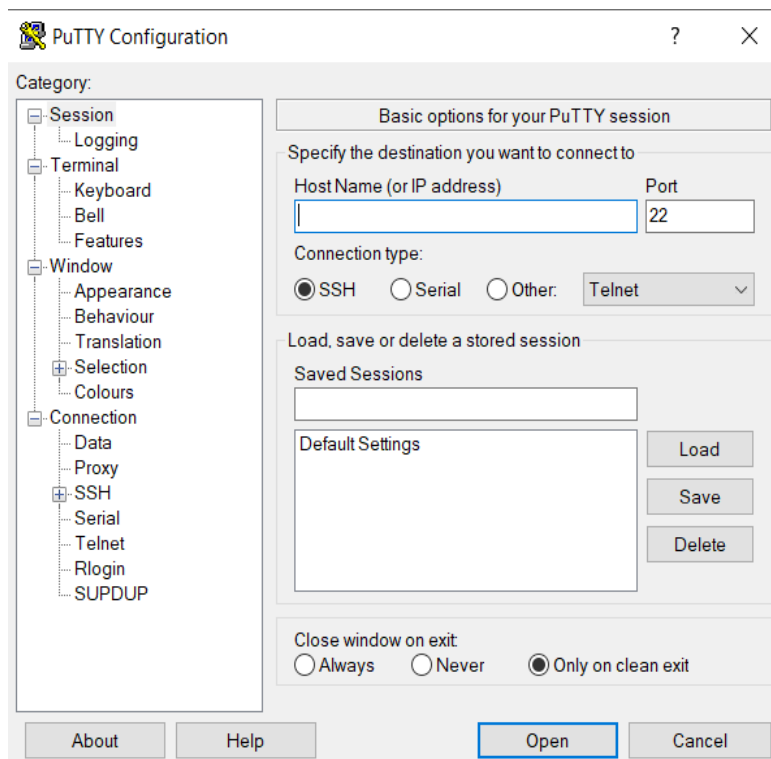
7.4 Installazione su server esterno

Il server utilizzato dal nostro gruppo è ospitato da [Kamatera](#), che offre un mese gratuito di utilizzo. Tuttavia, la seguente guida è formulata in modo generale ed adatta a qualsiasi server Linux, (con alcune modifiche intuitive, se necessario, anche per Windows).

Il server è stato configurato con queste specifiche: 4GB di RAM, 2 CPU, 40 GB MEMORIA

Name	Zone	IP	CPU	RAM	Storage	State	Actions
Sad_game	EU-ML	113.30.151.218	2B	4096MB	40GB		Actions Open

Dopo aver configurato il server, è necessario collegarsi ad esso attraverso il protocollo SSH. Lo strumento utilizzato da noi è PuTTY, ma è possibile utilizzare qualsiasi programma in grado di stabilire una connessione SSH. La connessione viene effettuata all'indirizzo IP pubblico fornito dal server e, dopo una fase di autenticazione (lo username solitamente è root, ma per sicurezza è possibile reperirlo sulla home del Server in Open ->Connect->Username), veniamo indirizzati sul prompt dei comandi:



```
root@sadgame: ~
--driver kamatera \
--kamatera-api-client-id {APIClientId} \
--kamatera-api-secret {APISecretId}

Optional:
--kamatera-datacenter {datacenter Code, default: EU}
--kamatera-billing {billing cycle, default: hourly}
--kamatera-cpu {cpu type, default: 1B}
--kamatera-ram {ram in MB, default: 512}
--kamatera-disk-size {disk size in GB, default: 10}
--kamatera-image {server OS Image, default: ubuntu_server_20.04_64-bit}

You can also set API ClientID/Secret:
export KAMATERA_API_CLIENT_ID={APIClientId}
export KAMATERA_API_SECRET={APISecretId}

=====

If you found any issue with this installation or have an idea how to improve this, please email us to: devteam@kamatera.com, thanks!

To delete this message of the day: rm /etc/update-motd.d/98-description
Last login: Fri Jan 12 17:26:18 2024 from 79.42.149.19
root@sadgame:~#
```

L'installazione sul server prevede i seguenti passaggi (riferiti a un sistema Ubuntu). Per altre distribuzioni, i comandi saranno sicuramente diversi.

```
sudo apt update
sudo apt install git

git clone https://github.com/Testing-Game-SAD-2023/A10-2024
cd ./A10-2024
chmod +x installer_docker.sh
chmod +x installer-linux-server.sh
chmod +x uninstaller.sh
./installer_docker.sh
./installer-linux-server.sh
```

A questo punto, l'installazione è completa. L'utente finale può collegarsi al servizio accedendo all'indirizzo IP pubblico reso disponibile dal server.

Di seguito sono riportati alcuni comandi di utilità per risolvere problemi comuni:

```
#Mostrare tutti i container
docker ps
docker container ls -a

#Avviare tutti i container
docker restart $(docker ps -q)
docker restart $(docker ps -q -a) #anche quelli già attivi
```

```
#Fermare tutti i container
docker stop $(docker ps -a -q)

#log container
docker logs ~idcontainer #sostituire id container con quello che si vuole il log
```

7.4.1 Problemi comuni

Dopo l'installazione, alcuni container potrebbero non partire in automatico, soprattutto per l'UI-Gateway. Si consiglia di controllare prima di riavviare tutti i container con il comando sopra specificato. Verificare quali container non sono ancora attivi usando il comando `docker container ls -a`, appuntarsi l'id, e avviarli manualmente tramite: `docker restart #codicecontainerid`.

Qualora i container non riuscissero ad avviarsi neanche con la procedura manuale è consigliato eseguire un riavvio dell'intero OS (server), nella maggior parte delle volte il problema viene risolto in questo modo.

Nota: L'UI-Gateway deve essere l'ultimo container da far partire, altrimenti lo stato EXIT sarà sempre presente.

Nota 2: Il container "progetto-sad-g19-master" è normale che risulti non in esecuzione.

7.5 Uninstaller

Al fine di facilitare l'operazione di disinstallazione dell'applicazione è stato creato un uninstaller.

È necessario eseguire lo script `uninstaller.bat` (Windows) o `uninstaller.sh` (Unix):

1. Vengono fermati tutti i container in esecuzione
2. Verrà effettuata la rimozione di tutti i container fermi, i volumi, le immagini e le reti create anche non inerenti alla seguente applicazione!).

Guida alle future integrazioni

Di seguito sono indicate delle procedure per effettuare l'integrazione di ulteriori futuri Task, mantenendo l'architettura proposta.

8.1 Integrazione API

Per quanto spiegato e presentato in questo documento, è essenziale effettuare le invocazioni corrette per le prossime integrazioni, sviluppi e funzionalità che faranno uso delle API dei vari microservizi che compongono l'applicazione:

INVOCAZIONE CORRETTA: `"/api/service"`

INVOCAZIONE ERRATA: `"http://localhost/api/service"` oppure `"http://qualunque_indirizzo/api/service"`

*Ovviamente le rotte delle API sono opportunamente gestite dai due gateway

Per una comprensione più chiara dei motivi, si consiglia di rileggere il paragrafo 3.2 Gateway Pattern sul Proxy gateway e il 0 con il deployment diagram e il rispettivo sequence diagram.

8.2 Integrazione Container

Per procedere con l'integrazione del container all'interno della stessa rete condivisa utilizzare il seguente file di configurazione `"docker-compose.yml"`:

```
1 version: '2'
2 services:
3   #sostituire "app" con il nome del container
4   app:
5     build: ./
6     expose:
7     # sostituire "8000" con la porta utilizzata
8     - 8000
9     networks:
10      - global-network
11
12 networks:
13   global-network:
14     external: true
```

8.3 Integrazione con UI Gateway

Per procedere con l'integrazione di un nuovo Front End, si deve modificare il file di configurazione `"/ui gateway/default.conf"` utilizzato da Nginx andando ad aggiungere un nuovo blocco `"location"`:


```

1 ...
2 # sostituire "webpage" con l'endpoint utilizzato che serve il frontend
3 location ~ ^/(webpage) {
4     include /etc/nginx/includes/proxy.conf;
5     # sostituire "app" con il nome del container e "8000" con la porta utilizzata
6     proxy_pass http://app:8000;
7 }
8 ...

```

8.4 Integrazione con API Gateway

Per procedere con l'integrazione di nuovo endpoint REST API modificare il file di configurazione `"/api_gateway/src/resources/application.yml"` utilizzato da Netflix Zuul e aggiungere un nuovo blocco "route":

```

1 ...
2 routes:
3     // sostituire "app" con il nome del servizio che si vuole dare a questo endpoint
4     app-service:
5         sensitiveHeaders:
6         // sostituire "endpoint" con il nome dell'endpoint
7         path: /api/endpoint/**
8         url: http://app:8000/endpoint
9 ...

```

8.5 Integrazione Installer

Per procedere con l'integrazione all'interno dell'installer modificare lo script batch di installazione `"installer.bat"`, modificando la lista delle cartelle in cui sono presenti i file di configurazione `"docker-compose.yml"`:

```

1 ...
2 set list="./T1-G11/applicazione/manvsclass" ".\T23-G1" ".\T4-G18" ".\T5-G2/t5" ".\T6-
3   G12/T6" ".\T7-G31/RemoteCCC" ".\T9-G19\Progetto-SAD-G19-master" ".\api_gateway" ".\
  ui_gateway"
4 ...

```

Note e sviluppi futuri

A seguito dello sviluppo del seguente task e dei test effettuati sull'applicazione web si sono evidenziate delle criticità. Per questo, di seguito sono elencati alcuni sviluppi che possono essere implementati in futuro.

Caso d'uso modifica classe

Al momento è presente il pulsante di modifica classe; quando viene attivato, l'utente è reindirizzato alla schermata dedicata per la modifica della classe, ma all'atto del upload compare la seguente schermata:

```
[{"name":"Calcolatrice","date":"2024-01-09","difficulty":"Beginner","code_uri":"Files-Upload/Calcolatrice/Calcolatrice.java","description":"","category":["math",""],"category":["math",""]}], [{"name":"Calcolatrice","date":"2024-01-09","difficulty":"Beginner","code_uri":"Files-Upload/Calcolatrice/Calcolatrice.java","description":"","category":["math",""],"category":["math",""]}]
```

Caso d'uso login amministratore

Il login dell'amministratore al momento è presente, ma è possibile accedere alla pagina della homepage del admin senza effettuare il login. Questa situazione potrebbe consentire l'accesso a dati sensibili da parte di utenti non autorizzati.

Una soluzione è replicare il meccanismo del token, già implementato per la parte studente al fine di avere una reale protezione anche lato admin.

Caso d'uso login

Al momento, la schermata di login non è ben strutturata. Si consiglia in futuro di implementare una logica di validazione degli input lato front-end al fine di verificare e ottenere un riscontro grafico sulla correttezza della password. Inoltre, si suggerisce di introdurre un reindirizzamento alla pagina di login qualora la registrazione vada a buon fine, migliorando così l'esperienza utente.

Protocollo HTTPS per l'intera applicazione

Il protocollo HTTPS permette di avere una maggior sicurezza rispetto al protocollo HTTP e anche una maggiore compatibilità con diversi servizi software.

Modalità Multiplayer

Dopo aver reso distribuita e accessibile la nostra applicazione, la prossima sfida sarà implementare una modalità di gioco multiplayer. Di seguito, viene illustrata una prima idea per realizzare questo meccanismo:

1. Implementare all'interno del database T2-3 relativo ai giocatori un flag che indichi se un giocatore è online o meno.
 - a. Ad ogni autenticazione o logout, il valore del flag viene aggiornato.
2. A ogni giocatore viene mostrata una lista di giocatori online.
3. Viene effettuato un collegamento tra due giocatori.
4. I giocatori che si sfidano scrivono il loro codice.
 - a. Nel momento del submit, vengono sempre calcolate le statistiche.

- b. Le statistiche calcolate vengono confrontate non più con il robot, ma con quelle generate dall'avversario.
- c. Il sistema mostra il vincitore.

Un'altra modalità potrebbe consistere nello sfidare tutti il robot, ed il vincitore è colui che ha battuto il robot con una percentuale migliore.

L'impatto di questa manutenzione additiva riguarda soprattutto il task 2-3 relativo al database e il task T5 dell'editor che gestisce la partita.

Gestione sessioni utente: unico accesso per dispositivo

Con la strategia proposta per l'implementazione del multiplayer, durante la fase di login è necessario solo un ulteriore controllo per verificare se l'utente è già online.

Registrazione Admin

Al momento la registrazione dell'Admin è prevista per qualsiasi utente. In futuro si potrebbe implementare la conferma di registrazione tramite email e l'inserimento di un token specifico per la registrazione, in modo che solo gli utenti in possesso del token possono registrarsi come Admin.

Caricamento Classi: Gestione Concorrenza

Attualmente, poiché l'applicazione è esposta su un indirizzo pubblico e il lato admin manca di protezione, chiunque conosca la rotta può caricare le classi. Inoltre, anche in futuro, se il lato admin sarà dotato di un sistema di gestione degli accessi, sarà necessario rivedere e progettare la concorrenza durante la generazione dei test e il loro caricamento. Ciò è essenziale poiché un caricamento contemporaneo delle classi potrebbe causare problemi di concorrenza.

Schermata Home

Si propone la creazione di una schermata home unica, che offra agli utenti le seguenti opzioni:

- Registrarsi come studente
- Registrarsi come Admin
- Login come studente
- Login come Admin
- Info
- Varie ed eventuali.

Divisione file web

Quasi tutti i task includono file HTML che contengono sia la parte di stile (.css) che la parte di script (.js). È fortemente consigliato procedere con la suddivisione di questi file per migliorare la compressione, la manutenibilità e la pulizia del codice.

Considerazioni finali

La seguente applicazione, basata su microservizi, ha raggiunto un notevole grado di avanzamento, presentando una struttura modulare solida e altamente flessibile. La suddivisione delle funzionalità in microservizi distinti ha favorito l'isolamento delle componenti e una gestione efficiente delle responsabilità. Questo approccio ha semplificato lo sviluppo, la manutenzione e la scalabilità dell'applicazione. L'integrazione continua di nuove funzionalità, tra cui la gestione delle classi e l'ottimizzazione dell'interfaccia utente, sottolinea la capacità del modello a microservizi di adattarsi alle mutevoli esigenze del progetto.

Nonostante i successi finora raggiunti, è importante sottolineare l'importanza di concentrarsi sull'autenticazione dell'amministratore nelle future implementazioni. Questo aspetto diventa cruciale per garantire un accesso sicuro e controllato alle funzionalità riservate, proteggendo al contempo l'integrità del sistema. Le proposte di sviluppo future dovrebbero quindi essere orientate a una maggiore attenzione all'autenticazione dell'admin al fine di consolidare ulteriormente la robustezza complessiva dell'applicazione.

Inoltre, per quanto riguarda le integrazioni dei task svolti parallelamente ai nostri, si consiglia di adottare il seguente approccio.

Qualora la documentazione dell'altro gruppo sia esaustiva e le modifiche non siano sostanziali, si può valutare l'implementazione di quest'ultime nel presente progetto.

Altrimenti:

- se il task T5, in particolare editor.html, non è stato impattato o lo è stato in modo lieve, si suggerisce di implementare le modifiche descritte in questo documento nel task sviluppato dall'altro gruppo. Ad eccezione del suddetto task T5, la cui base di partenza è di questo gruppo (A10-2024).
- Se, invece, per soddisfare i requisiti, il task T5 dell'altra integrazione ha subito forti variazioni, si lascia al lettore/sviluppatore la facoltà di decidere quale sia la scelta migliore. Tuttavia, si ricorda che questo gruppo non solo ha aggiunto funzionalità al task T5, ma ha anche svolto un vero e proprio lavoro di refactoring in termini di divisione del codice e reimplementazione delle chiamate asincrone.

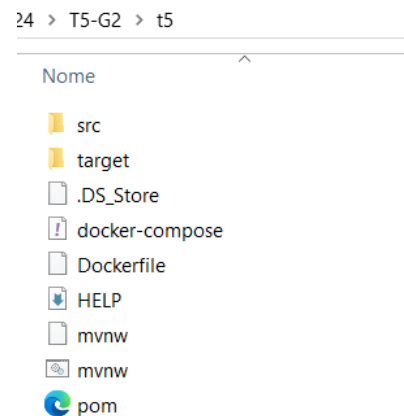
HOW TO: Come modificare l'applicazione

In questo ultimo paragrafo, cerchiamo di spiegare in modo semplice e rapido come modificare i vari task in modo corretto e rendere effettive le modifiche. Si consiglia sempre di consultare e comprendere la documentazione dei vari task per approfondire gli aspetti tecnologici e funzionali. Tuttavia, poiché l'applicazione è sviluppata interamente su container, possiamo descrivere dei passaggi comuni che renderanno più facile l'approccio iniziale all'applicazione.

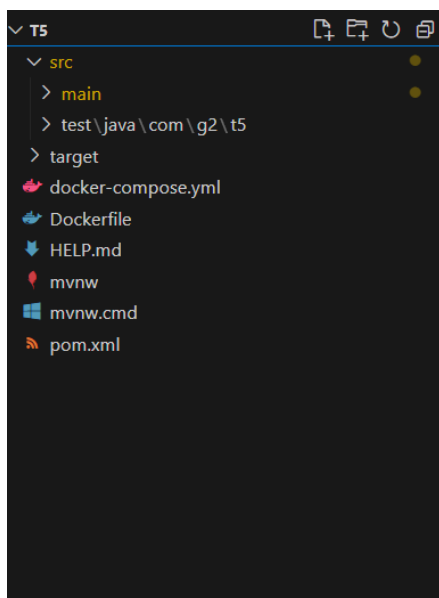
Le cartelle dei vari task seguono, più o meno, la stessa struttura:

-Nome task

- src
- target



Struttura cartelle



Cartella principale del task

Il codice che

l'applicazione eseguirà è presente nella cartella "target". Quindi, tutti i file "finali" (file .class, dipendenze e vari file html/css/js) risiederanno lì. Durante la modifica, è possibile utilizzare un IDE che, modificando i file in "src", salverà i rispettivi file necessari nella cartella "target". Si consiglia l'uso di Visual Studio Code: aprendo il programma nella cartella principale (dove sono presenti "src" e "target"), le modifiche effettuate in "src" verranno apportate anche in "target" (VScode si occuperà anche di compilare i file .java).

Generati i file modificati, il prossimo passo è generare il file .jar, l'artefatto che andrà in esecuzione nel docker. È richiesta l'installazione di Apache Maven. Una volta installato e configurate le variabili di sistema, posizionarsi all'interno della cartella principale del task (dove sono presenti i file mvnw) ed eseguire il seguente comando: `mvn package`

```
\A10-2024\T1-G11\application\manvsc\class>mvn package
```

Il nuovo file jar verrà creato nella cartella "target". Infine, occorre creare il container. Ciò è possibile farlo in due modi: utilizzare l'installer dell'intera applicazione e reinstallare tutto oppure installare solo il singolo task con il seguente comando, impartito sempre nella cartella principale (troverete la presenza del file Dockerfile): [*docker compose up -d --build*](#)

```
\A10-2024\T1-G11\application\manvsc\class>docker compose up -d --build
```

N.B. Ricordarsi che ogni prompt che coinvolga il docker richiede che il docker stesso sia in esecuzione

Seguendo questa guida, le modifiche ai task saranno subito effettive. Ci auguriamo che questa breve guida abbia contribuito a una migliore comprensione e a velocizzare le operazioni di modifica sulla nostra applicazione.

Strumenti Software e Tecnologie Utilizzate

Si riporta un elenco dei software e delle tecnologie utilizzate nel corso del progetto di sviluppo del software.

Microsoft Teams

Microsoft Teams è uno strumento software che combina chat di lavoro persistente, teleconferenza e condivisione di contenuti per agevolare la collaborazione a distanza. Ha consentito di ottimizzare i tempi di riepilogo del lavoro svolto singolarmente dai componenti del gruppo tramite riunioni della durata di circa tre ore tenute settimanalmente.

Microsoft OneDrive

Microsoft OneDrive è un servizio di cloud storage e backup, accessibile tramite browser e app desktop o app dispositivo mobile. Ha garantito la condivisione in tempo reale delle risorse del progetto e favorito la collaborazione dei membri del team.

Visual Paradigm

Visual Paradigm offre strumenti di progettazione, analisi e gestione per lo sviluppo di progetti. È stato fondamentale per la formalizzazione e documentazione della struttura e delle funzionalità dell'applicazione.

Miro

Miro è una piattaforma di collaborazione online che offre una lavagna virtuale e strumenti di collaborazione per il lavoro di squadra. Si è rivelato particolarmente utile per lo sprint planning e la formalizzazione dei requisiti sotto forma di storie utente.

Visual Studio Code ed Eclipse

Visual Studio Code ed Eclipse sono editor di codice gratuiti, open source e multiplatforma. Supportano molti linguaggi e funzionalità.

GitHub

GitHub è un servizio di hosting per progetti software. È stato uno strumento chiave per tenere traccia delle modifiche al codice, collaborare in modo efficiente e garantire la gestione delle versioni del software in modo coerente. L'approccio adottato ha favorito lo sviluppo parallelo e incrementale.

Maven

Maven è uno strumento di gestione di progetti software basati su Java e build automation. Maven usa un file XML che descrive le dipendenze fra il progetto e le varie versioni di librerie necessarie. La separazione tra le librerie e la directory di progetto promuove la coerenza e l'affidabilità del processo di sviluppo.

WSL

Windows Subsystem for Linux è un layer di compatibilità per l'esecuzione di binari Linux in modo nativo su Windows

Docker

Docker è un software progettato per eseguire processi informatici in ambienti isolabili, minimali e facilmente distribuibili chiamati container. Contribuisce all'efficienza del processo di deployment, garantendo la massima portabilità dell'applicazione indipendentemente dall'ambiente di esecuzione sottostante. Le immagini Docker sono snapshot di applicazioni con tutte le loro dipendenze e configurazioni e i container Docker sono istanze in esecuzione di queste immagini. I volumi Docker sono usati per memorizzare dati in modo persistente, anche dopo la chiusura di un container.

Servizi RESTful

I servizi RESTful costituiscono una categoria di servizi web che si basa sul protocollo HTTP, mirati a facilitare una comunicazione efficiente tra client e server. Delineano le operazioni che possono essere eseguite sulle risorse del server mediante l'utilizzo dei verbi HTTP, quali GET, POST, PUT e DELETE. Ad esempio, l'annotazione `@PostMapping` viene impiegata per definire i metodi responsabili della gestione delle richieste HTTP di tipo POST, mentre `@GetMapping` è utilizzata per definire i metodi dedicati alla gestione delle richieste HTTP di tipo GET.

Ngrok

Ngrok è un reverse proxy che permette di esporre la propria applicazione web su un indirizzo pubblico attraverso un servizio di tunneling, particolarmente utile per il testing. La scelta di Ngrok è stata motivata da diverse ragioni: la sua capacità di interfacciarsi direttamente con il software Docker e di fornire un dominio statico.

Pinggy

Pinggy è un servizio di tunneling che adopera il protocollo SSH per interfacciare l'indirizzo locale sul porto in cui è esposta l'applicazione con un indirizzo pubblico, disponibile per la durata di 60 minuti. Sempre molto utile per la fase di testing e particolarmente semplice e veloce da utilizzare.

Kamatera

Kamatera è una piattaforma cloud avanzata con server virtuali e servizi di hosting. È un'opzione robusta per ospitare applicazioni web, anche se presenta costi superiori rispetto ai servizi di tunneling. È disponibile una prova gratuita di un mese per coloro che desiderano testarla.

Postman

Postman è uno strumento software che permette di effettuare testing automatico sulle applicazioni web e API con la possibilità di effettuare diversi tipi.