UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

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

Software Architecture Design

Documentazione Task R2 - Gruppo B13: Implementazione classifiche

Anno Accademico 2024/2025

Salvatore Santoro sal.santoro@studenti.unina.it
Andrea Russo andrea.russo48@studenti.unina.it
Andrea Giaquinto andr.giaquinto@studenti.unina.it
Repo github: https://github.com/SalvatoreSantoro/A13

Indice

1 Introduzione

1.1 ENACTEST

Il progetto di cui il nostro team si è occupato va contestualizzato nell'ambito dell'ENACTEST, acronimo di European iNnovation AllianCe for TESting educaTion, ovvero un progetto europeo il cui obiettivo è la progettazione e realizzazione di materiali didattici a supporto dell'insegnamento delle tecniche di testing.
In particolare è stato realizzato un Educational Game in cui i giocatori hanno la possibilità di scrivere la propria test suite e provare a battere tool di generazione automatica di test, come Randoop o Evosuite.

1.2 Nuovi requisiti assegnati

Il nostro team si propone di sviluppare classifiche dei giocatori basate su specifiche statistiche di merito relative alle diverse modalità di gioco, rendendole facilmente consultabili dagli utenti. Queste classifiche saranno accessibili direttamente dalla homepage del gioco e offriranno la possibilità di selezionare la classifica desiderata, oltre a individuare rapidamente la posizione di un giocatore all'interno di essa.

In particolare, ci siamo proposti di implementare la visualizzazione delle classifiche unicamente per la modalità di gioco "Sfida" e per le statistiche "Partite giocate" e "Partite vinte", proponendoci di garantire che i componenti aggiunti e le scelte di design consentano l'estendibilità futura a nuove statistiche e nuove modalità di gioco. Maggiori dettagli in merito sono disponibili nella sezione Sviluppi futuri (Capitolo 6)

1.3 Processo di sviluppo

Il processo di sviluppo adottato segue a grandi linee i principi del framework SCRUM. Per l'analisi del task abbiamo iniziato con una definizione di alto livello dei requisiti del progetto, articolandoli in **User Stories**. Le stories sono state concepite come descrizioni sintetiche ma efficaci delle funzionalità richieste, focalizzandoci sul valore che queste avrebbero apportato agli utenti finali.

Successivamente, abbiamo proceduto alla decomposizione delle User Stories in sotto-task per mantere una visione chiara e organizzata dell'avanzamento. Ogni sotto-task è stato analizzato per stimarne il costo temporale necessario al completamento. Per mantenere una visione organizzata e chiara dell'avanzamento, abbiamo

definito un **backlog**. Questo backlog ha funzionato come una lista dinamica e ordinata delle attività da svolgere, consentendoci di concentrarci su poche attività alla volta.

Abbiamo quindi adottato un approccio iterativo e incrementale per lo sviluppo, organizzando il lavoro in sprint con cadenza settimanale. Durante ciascuno sprint, abbiamo selezionato dal backlog i sotto-task prioritari da completare, assicurandoci di mantenere un ritmo costante e di produrre risultati tangibili al termine di ogni iterazione. Nel capitolo successivo, in Tabella 4 è stata riportata la stima sul costo dei sotto-task e nelle Figure 3, 4, 5 sono stati riportati i backlog corrispondenti ai 3 sprint completati.

Infine, abbiamo adottato in più occasioni la metodologia **Pair Programming**, in cui un componente scriveva il codice mentre gli altri osservavano, rivedevano e fornivano feedback in tempo reale. I ruoli si sono alternati frequentemente, favorendo la condivisione di conoscenze, la riduzione degli errori e una maggiore qualità del codice.

Figura 1: User Story 1

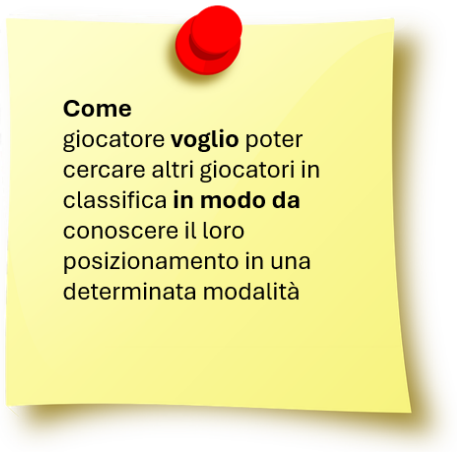


Figura 2: User Story 2

2 Analisi dei requisiti

2.1 User Stories

Per la formalizzazione dei requisiti sono state identificate due User Stories in Figura 1 e Figura 2.

Queste User Stories svolgono un ruolo fondamentale nel processo di analisi, poiché consentono di chiarire i requisiti dal punto di vista dell'utente finale. La prima User Story descrive un caso d'uso specifico legato alla visualizzazione di una classifica relativa ad una specifica modalità di gioco, mentre la seconda si focalizza sulla ricerca di altri giocatori in una delle classifiche.

2.2 Requisiti funzionali, non funzionali e sui dati

Per avere un quadro più dettagliato dei requisiti necessari alla creazione e visualizzazione delle classifiche, abbiamo identificato i requisiti funzionali, non funzionali e i requisiti sui dati.

Si chiarisce che all'interno dell'applicativo non viene memorizzato alcun username o nickname dei giocatori, pertanto si è scelto di utilizzare l'email come identificativo univoco su cui basare la ricerca e la visualizzazione dei posizionamenti.

ID	**R**equisito funzionale
RF1	Il sistema deve permettere ai giocatori di visualizzare le classifiche relative alle due modalità di gioco attualmente presenti
RF2	Il sistema deve permettere ai giocatori di accedere alle classifiche tramite un apposito bottone
RF3	Il sistema deve permettere di scegliere la modalità di gioco e su quale statistica ordinare la classifica
RF4	Il sistema deve mettere in risalto la posizione dell'utente corrente in classifica
RF5	Il sistema deve permettere la ricerca in classifica di altri utenti tramite e-amil

Tabella 1: Requisiti funzionali

ID	**R**equisito non funzionale
RNF1	Le classifiche devono essere organizzata in pagine in modo da favorire la navigabilità
RNF2	La generazione e il retrieval delle classifiche deve avvenire gestendo in maniera efficiente le risorse di calcolo, di memoria e di banda per favorire la scalabilità e le performance

Tabella 2: Requisiti non funzionali

ID	**R**equisito sui dati
RD1	Ciascuna classifica è caratterizzata dalla modalità di gioco, dalla statistica di merito e dall'elenco dei posizionamenti dei giocatori
RD2	Ogni posizionamento è caratterizzato dalla posizione del giocatore in classifica, e-mail del giocatore ed il valore della statistica

Tabella 3: Requisiti sui dati

ID	**Story/Task**	**Stima**
1	Come studente voglio visualizzare le classifiche relative alle modalità di gioco in modo da conoscere il mio piazzamento rispetto agli altri giocatori	
1.1	Creazione UI per visualizzare la classifica	1
1.2	Creazione logica di retrieval dei dati necessari	3
1.3	Implementazione del caching delle pagine nel front end	2
2	Come giocatore voglio poter cercare altri giocatori in classifica in modo da conoscere il loro posizionamento in una determinata modalità	
2.1	Implementazione della barra di ricerca utente	2

Tabella 4: Sprint backlog con stima dell'effort

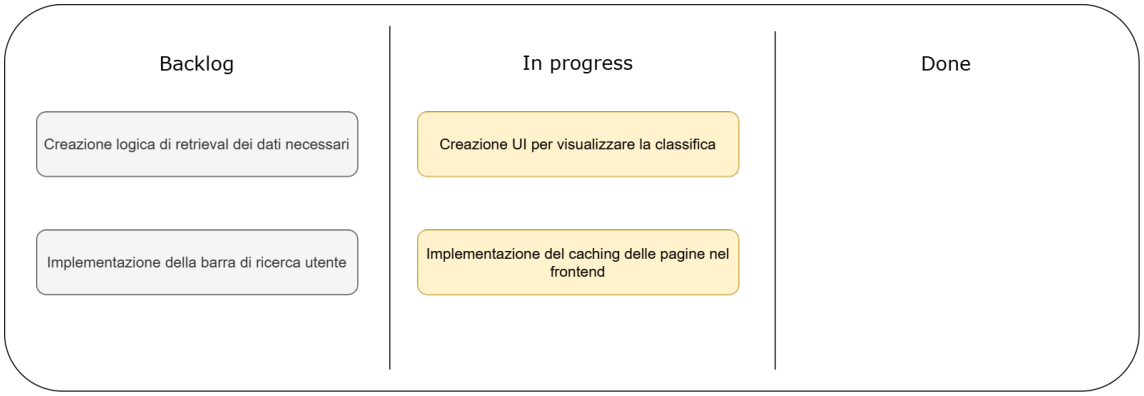


Figura 3: Sprint 1 - implementazione UI e meccanismi di caching

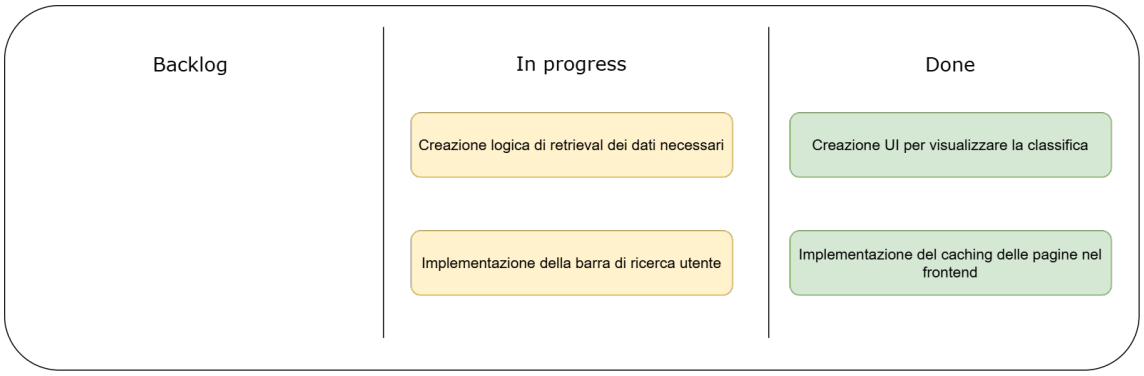


Figura 4: Sprint 2 - completato il front-end

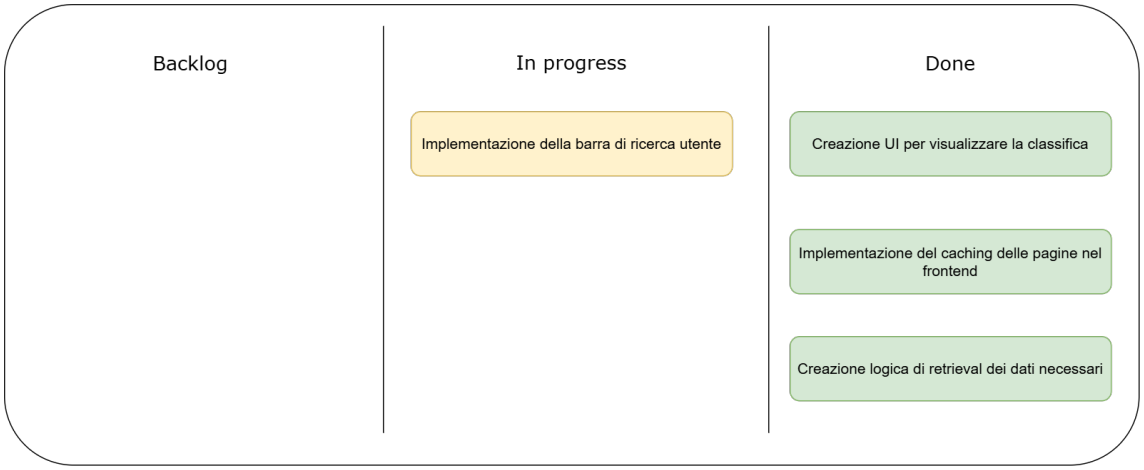


Figura 5: Sprint 3 - completato il back-end, implementazione della funzionalità di ricerca

2.3 Class Diagram

Il Class Diagram in Figura 6 chiarisce come verranno soddisfatti i requisiti sui dati presenti in Tabella 3.

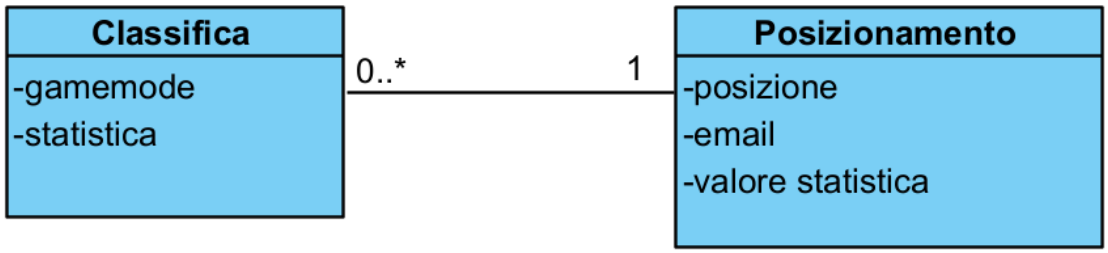


Figura 6: Class Diagram

2.4 Use Case Diagram

Lo Use Case Diagram riportato in Figura 7 definisce i due casi d'uso individuati a partire dai requisiti funzionali.

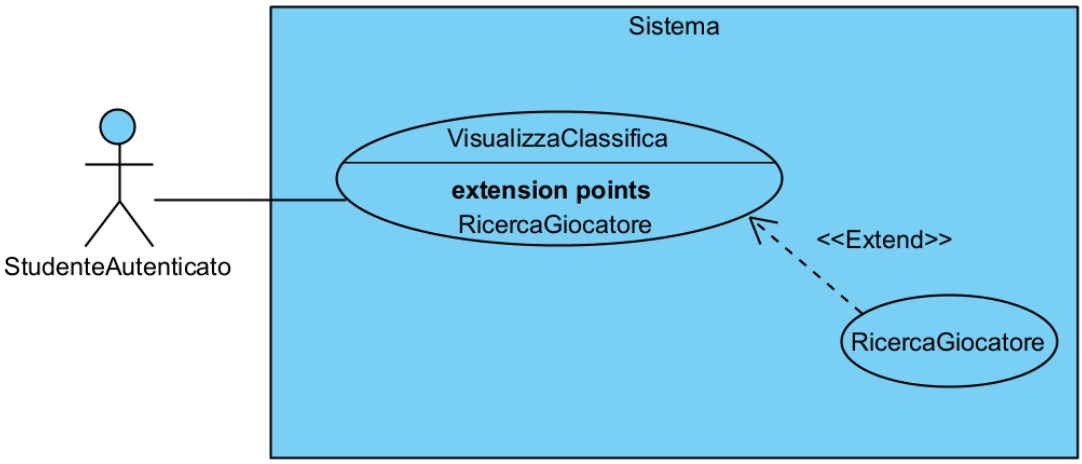


Figura 7: Use Case Diagram

3 Analisi dell'impatto dei requisiti assegnati

3.1 Componenti coinvolti

L'implementazione delle classifiche impatta principalmente i microservizi **T4** e **T5**, in quanto il microservizio T4 si occupa di gestire il Game Repository contenente i dati delle partite e la loro relazione con i giocatori, mentre il microservizio T5 si occupa di fornire le pagine web all'utente.

In aggiunta, anche il microservizio **T23** è impattato in quanto al momento non prevede un endpoint per cercare un giocatore tramite email.

3.2 Impatto dei nuovi requisiti

Il microservizio **T4** dovrà fornire accesso ai dati delle partite aggregati per modalità di gioco e tipologia di statistica. Tali dati devono essere esposti tramite specifiche API REST progettate per la lettura e la consultazione delle classifiche. Per soddisfare i requisiti non funzionali, sarà necessario implementare endpoint che permettano il recupero di sotto-intervalli delle classifiche.
Il microservizio **T5** avrà il compito di integrare una vista dei dati aggregati provenienti dal Game Repository all'interno della Homepage. La nuova vista dovrà garantire un'esperienza utente ottimale, supportando sia la navigabilità sia un caricamento efficiente dei dati.
Il microservizio **T23**, infine, dovrà invece fornire un endpoint per cercare un giocatore a partire dalla sua email.

4 Progettazione

4.1 Scelte di Design

4.1.1 Microservizio T4

Il microservizio T4 attualmente utilizza un Controller diverso per ciascun tipo di risorsa accessibile. E' stato quindi necessario creare un nuovo Controller per esporre le API di lettura delle classifiche ed un nuovo Model per incapsulare i dati ed implementare la logica di business per la lettura.

La scelta progettuale più rilevante ha riguardato il metodo di calcolo delle statistiche a partire dai dati sulle partite. Nel Game Repository le partite giocate sono salvate nella tabella "Game", mentre la relazione tra il giocatore e la partita giocata, insieme all'informazione sulla vittoria/sconfitta che viene aggiornata attraverso una UPDATE, sono presenti in un'altra tabella "PlayerGame". Di conseguenza ogni richiesta di retrieval di una classifica consisterebbe di due operazioni:

- **Aggregazione** delle partite giocate/vinte da un certo giocatore in base al suo **PlayerID**
- **Ordinamento** decrescente in base alla statistica selezionata

Queste due operazioni potrebbero pesare molto sul Game Repository se la tabella PlayerGame dovesse contenere molte entry, causando la violazione del requisito non funzionale **RNF2**.

Per risolvere il problema dell'aggregazione è stato scelto di creare una tabella **PlayerStats** con le statistiche già aggregate, la quale viene aggiornata attraverso dei **TRIGGER SQL** sulla tabella "PlayerGame".

Per evitare invece di eseguire frequenti ordinamenti, sono state individuate due opzioni:

1. Creare un **Indice secondario SQL** per ciascuna statistica presente in PlayerStats
    - **Pro**: la richiesta della classifica o parti di classifiche beneficerebbe di un ordinamento già effettuato tramite indici. La classifica sarebbe quindi aggiornata in real-time
    - **Contro**: ad ogni entry salvata nella tabella "PlayerGames", gli indici devono essere aggiornati (overhead)

2. Creare una **Routine GO** all'interno del microservizio T4 che ad intervalli regolari (es. 1-2 minuti) legge l'intera tabella PlayerStats e ne mantiene una copia ordinata per ciascuna statistica in memoria principale.

    - **Pro**: il DB viene scaricato delle operazioni di ordinamento e può continuare a servire richieste
    - **Contro**: overhead di spazio occupato in memoria. Inoltre la tabella non è aggiornata in real-time

Per scegliere tra le due soluzioni è stato stimato l'overhead temporale dovuto all'aggiornamento degli indici menzionati nella soluzione 1. Sono state effettuate **100.000 INSERT** nella tabella PlayerGames misurando i tempi di esecuzione senza la dichiarazione degli indici secondari e successivamente con gli indici secondari. I risultati mostrati in Figura 8 e Figura 9 indicano un overhead temporale di 22 secondi in totale, ovvero circa **0,2 ms per ciascuna INSERT**.

Alla luce di ciò, l'overhead temporale causato dall'utilizzo degli indici secondari si è rivelato essere trascurabile rispetto all'overhead spaziale e alla maggiore complessità di implementazione della soluzione 2. Di conseguenza si è scelto di **optare per la soluzione 1**.

```
app-1          | 2024/12/14 15:56:13 took 1m24.236901493s
app-1          | 2024/12/14 15:56:13 listening on 0.0.0.0:3000
```

Figura 8: Tempo di esecuzione di 100k INSERT senza indici secondari

```
app-1          | 2024/12/14 16:27:18 took 1m46.545199139s
app-1          | 2024/12/14 16:27:18 listening on 0.0.0.0:3000
```

Figura 9: Tempo di esecuzione di 100k INSERT con indici secondari

4.1.2 Microservizio T5

Nel microservizio T5 le rotte destinate al retrieval di pagine web e dati relativi al Front-end sono gestite dal **GuiController**. E' stato quindi necessario introdurre una nuova rotta GET che consenta al Front-end di richiedere dinamicamente i dati delle classifiche da mostrare nel nuovo componente UI riservato alla visualizzazione delle classifiche.
In ottemperanza al pattern MVC, è stato quindi creato il nuovo componente UI

nella View della Homepage ed un nuovo Model **LeaderboardSubinterval** per rappresentare i dati delle classifiche sotto forma di sotto-intervalli. La business logic è stata incapsulata in un nuovo servizio **LeaderboardService** che gestisce le interazioni con il T4 per ottenere gli intervalli di una classifica selezionata e con il T23 per recuperare le e-mail dei giocatori. Il GuiController, dopo aver validato i parametri della richiesta GET, può quindi delegare l'elaborazione della richiesta al nuovo servizio.

4.2 Progettazione delle modifiche in T4

4.2.1 Modello ER aggiornato

In Figura 10 si riporta il nuovo modello ER del database presente in T4 in cui è stata aggiunta la nuova tabella **PlayerStats**. Questa tabella contiene una colonna per ciascun tipo di classifica, ovvero per ciascuna coppia (modalità di gioco, statistica di merito). Attualmente, sono state previste soltanto le statistiche "Partite giocate" e "Partite vinte" in modalità di gioco "Sfida" ma il procedimento è facilmente estendibile a nuove classifiche come spiegato nel Capitolo 6 (Sviluppi futuri).

- La colonna **SfidaPlayedGames** viene aggiornata attraverso un TRIGGER SQL attivato sulle **INSERT** nella tabella "Game_Player" (chiamata "player_games" nel database). Ad ogni nuova partita salvata, il trigger legge il PlayerID associato e incrementa di 1 il valore presente in questa colonna (oppure crea una nuova riga inizializzando questo valore ad 1 se è la prima partita giocata).
- La colonna **SfidaWonGames** viene aggiornata attraverso un TRIGGER SQL attivato sulle **UPDATE** della tabella "Game_Player". Ad ogni UPDATE di una riga viene letto l'attributo IsWinner: se è stato aggiornato a True, il trigger legge il PlayerID associato e incrementa di 1 il valore presente in questa colonna.

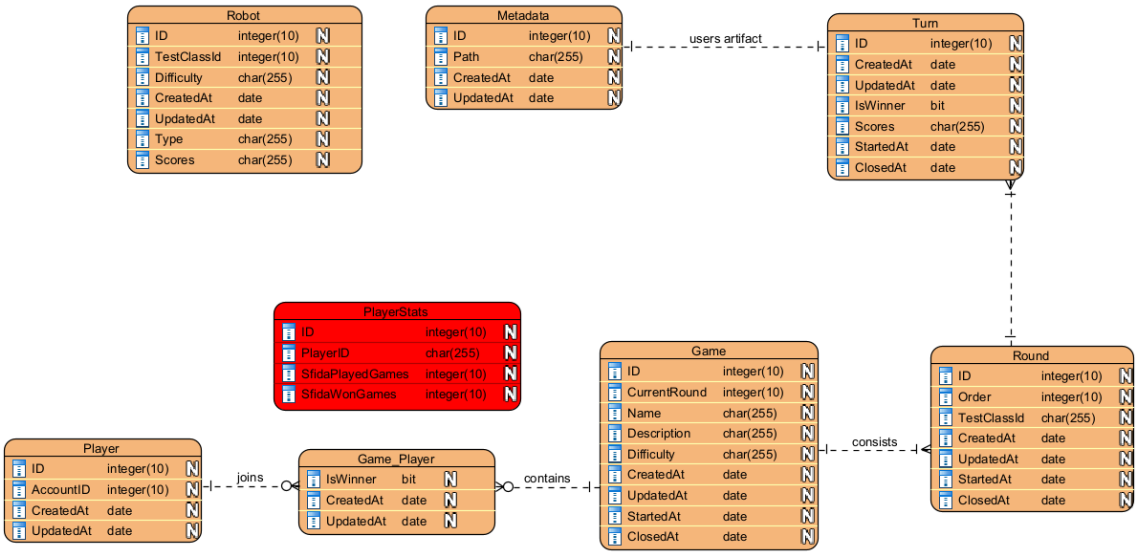


Figura 10: Modello dei dati aggiornato

4.2.2 Composite Diagram

In Figura 11 si riportano le classi aggiunte al microservizio T4. In particolare, è stato aggiunto il nuovo Controller **Leaderboard** per il processamento delle richieste in ingresso, i nuovi Model **Leaderboard** e **Row** per la gestione e il mapping su JSON dei dati provenienti dal database ed il nuovo Service per l'implementazione delle logiche di business e l'interazione con il database. Inoltre, è stato aggiunto il Model **PlayerStats** necessario all'ORM *GORM* per l'inizializzazione della tabella nel database e la definizione dei constraints e degli indici secondari.

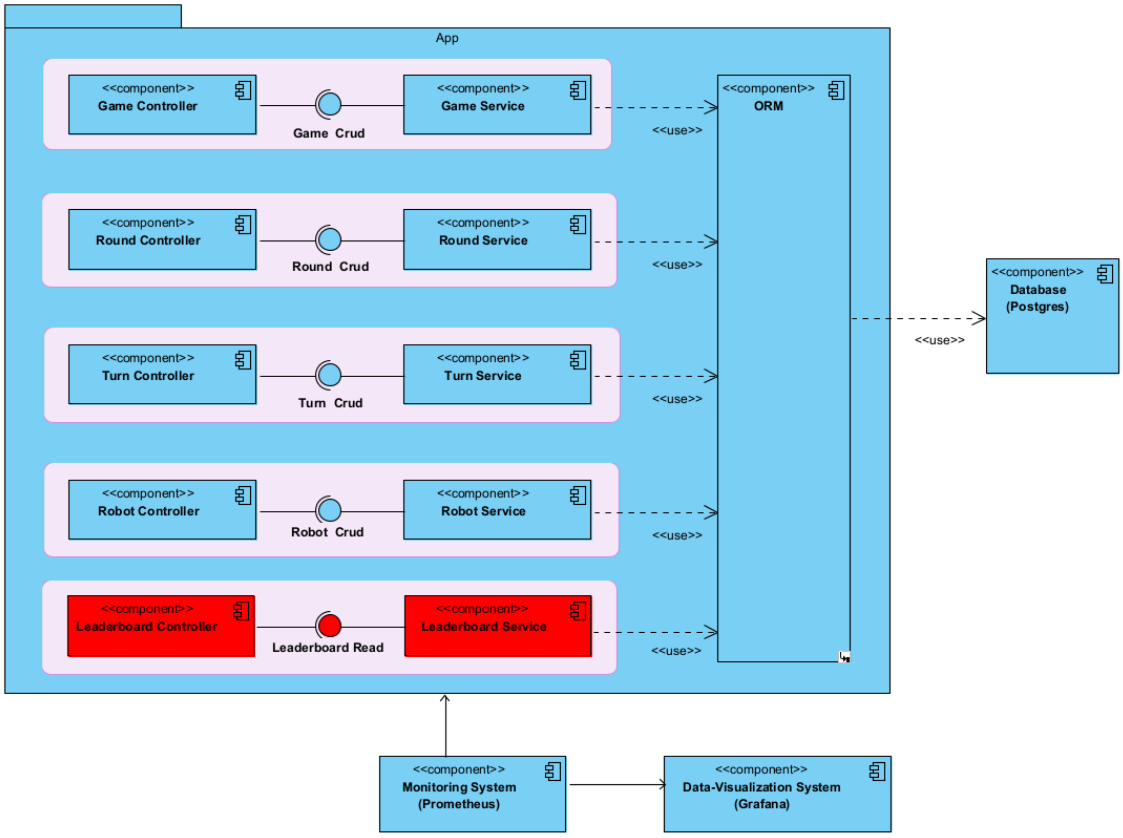


Figura 11: Composite diagram T4

4.3 Progettazione delle modifiche in T5

4.3.1 View

Nella progettazione della View sono stati tenuti in conto i requisiti non funzionali di usability e scalability, che sono stati realizzati attraverso meccanismi di:

- **selezione**: l'utente può scegliere la classifica da visualizzare in base ad un selettore per la modalità di gioco (ad es. "Sfida") ed un selettore per la statistica (ad es. "Partite vinte")
- **paginazione**: le classifiche sono facilmente navigabili per pagine
- **caching**: il front-end richiede un numero di pagine alla volta (fissato a 5) e durante la navigazione richiede ulteriori pagine solo se non presenti in cache. Anche la ricerca di un giocatore beneficia del caching: l'email inserita nell'apposito campo viene prima cercata tra le pagine in cache e poi richiesta al back-end se non viene trovata.

Inoltre, abbiamo inserito un bottone di "refresh" per aggiornare la classifica invalidando la cache ed un meccanismo che evidenzia la riga corrispondente al giocatore cercato.

In Figura 12 è riportata la User Interface realizzata, accessibile a scomparsa tramite un bottone "Classifiche" inserito nella navbar.

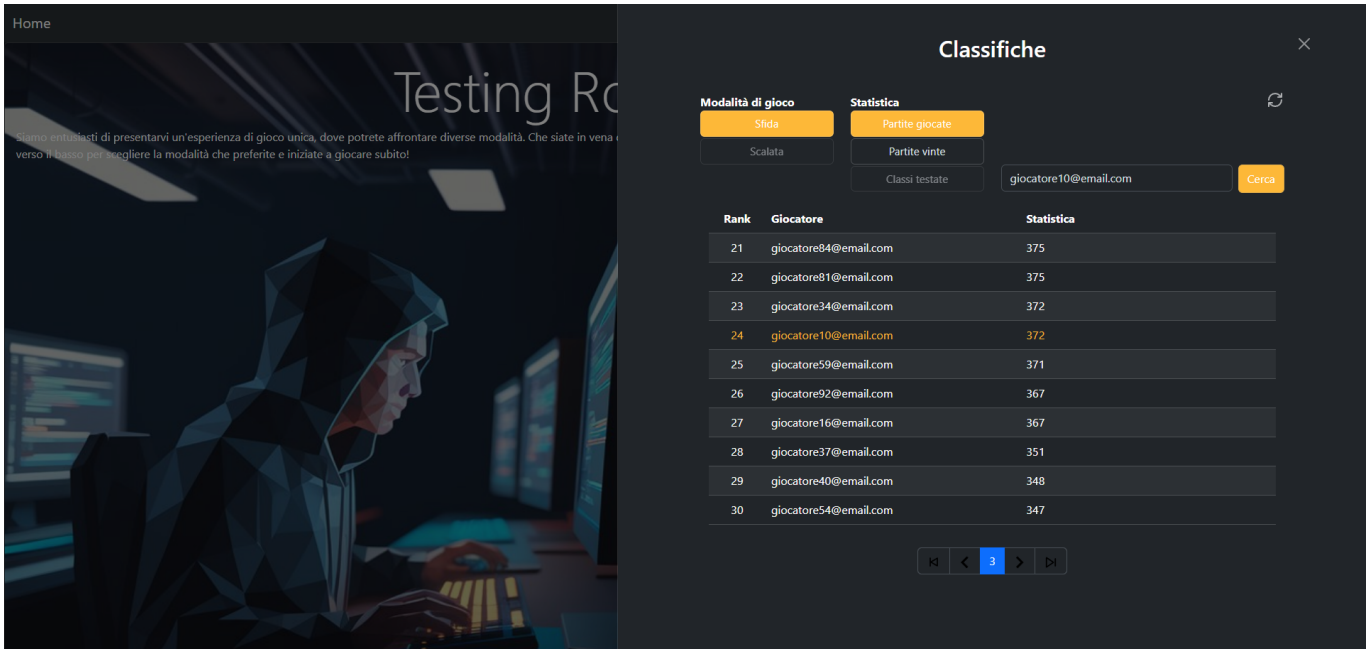


Figura 12: User Interface

4.3.2 Package Diagram

In Figura 13 sono mostrate solo le aggiunte effettuate.

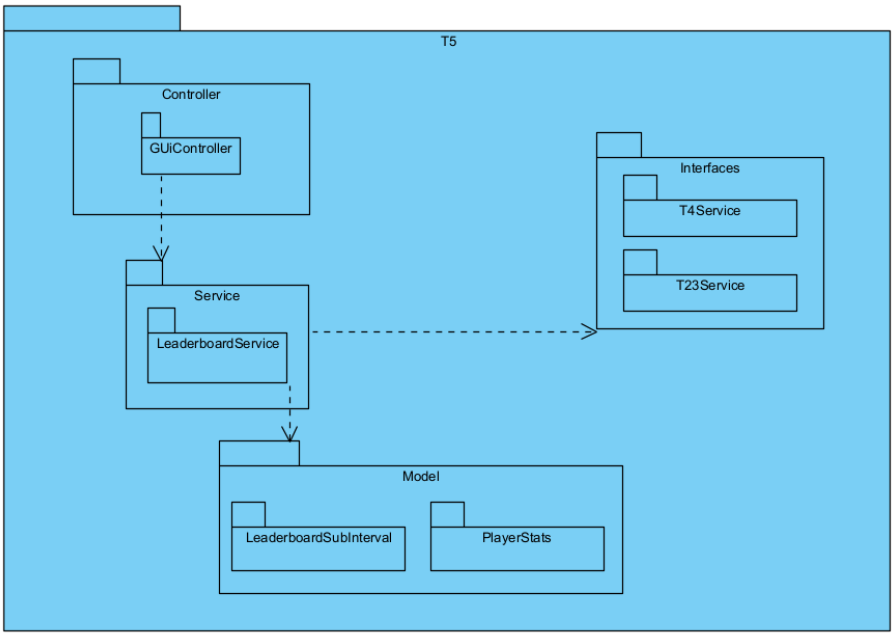


Figura 13: Package diagram T5

4.3.3 Sequence Diagrams

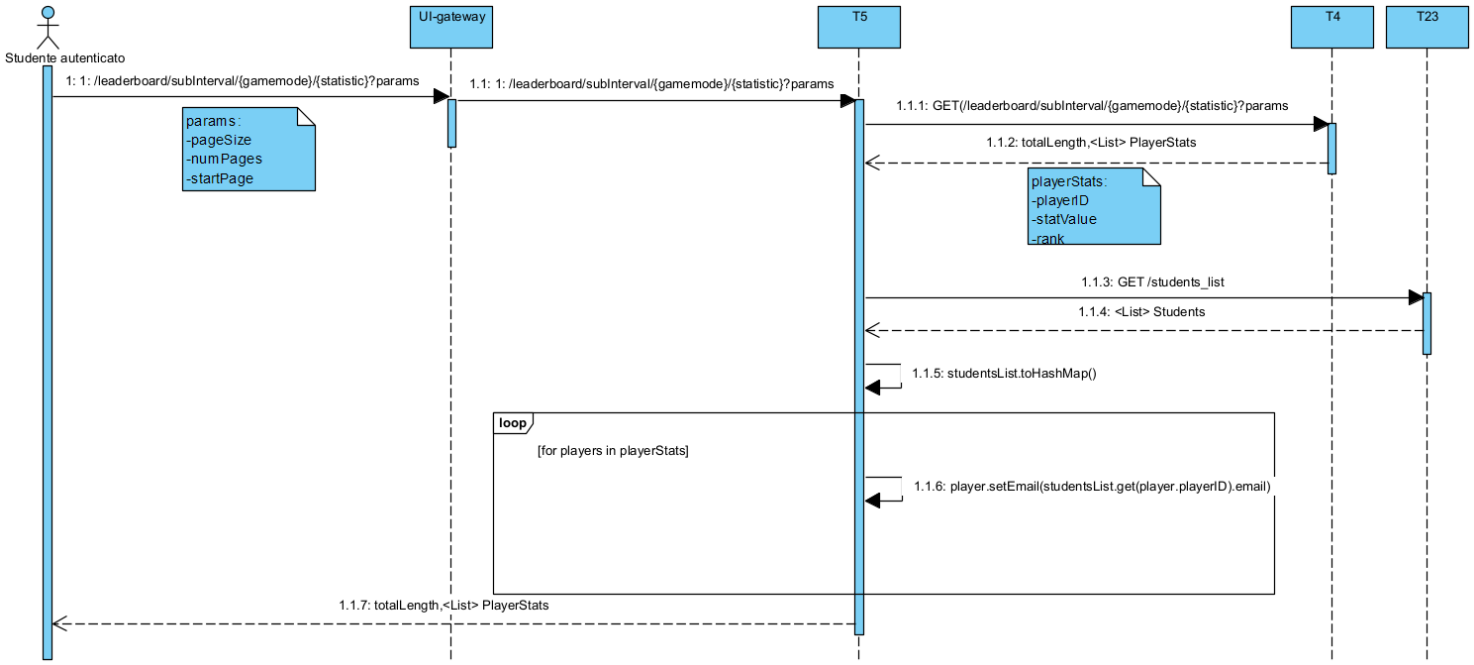


Figura 14: Sequence diagram per la visualizzazione delle classifiche

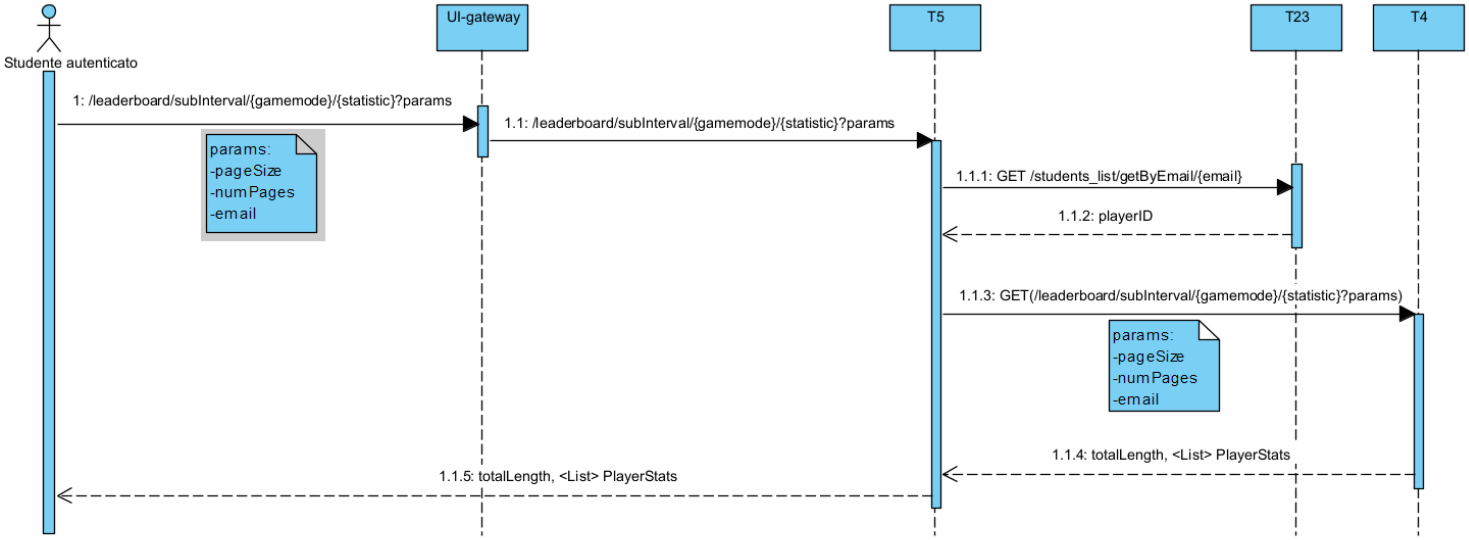


Figura 15: Sequence diagram per la ricerca di un giocatore nel caso in cui non viene trovato nella cache front-end

4.4 Design delle nuove REST API

Le nuove API implementate in T4 e in T5 sono riassunte in Figura 16.

Per la progettazione delle nuove REST API in T4 è stato seguito il seguente procedimento:

1. **Identificazione della risorsa da servire**: L'obiettivo principale è fornire accesso a una classifica, definita dai seguenti parametri principali:

   - **Modalità di gioco** (`gamemode`)
   - **Statistica di merito** (`statistic`)

   Per gestire in modo efficiente classifiche potenzialmente molto ampie, l'API deve supportare la consultazione di **sotto-intervalli** specifici, determinati da un intervallo di posizioni predefinito (ad esempio, dalla posizione 1 alla 50).

2. **Definizione della struttura della risorsa**: Un sotto-intervallo di classifica è rappresentato come un insieme ordinato di posizioni. Ogni posizione è modellata come una tupla avente il formato seguente:

   - `rank`: Posizione nella classifica
   - `playerID` Identificativo del giocatore
   - `valoreStatistica` Valore della statistica di riferimento

T5				
Descrizione	Endpoint	Path params	Query params	Response

Richiesta di un intervallo di pagine	**/leaderboard/subInterval**	/{*gamemode*}/{*statistic*}/	• *numPages*
			• *pageSize*
			• *startPage*
				{
				"totalLength": ?
				"positions": [{
				rank: ?,
				userId: ?,
				email: ?,
				statistic: ?
				},
				...
				]
				}

Richiesta dell'intervallo di pagine in cui è presente il giocatore cercato	**/leaderboard/subInterval**	/{*gamemode*}/{*statistic*}/	• *numPages*
			• *pageSize*
			• *email*
				{
				"totalLength": ?
				"positions": [{
				rank: ?,
				userId: ?,
				email: ?,
				statistic: ?
				},
				...
				]
				}

T4				
Descrizione	Endpoint	Path params	Query params	Response

Richiesta di un intervallo di pagine	**/leaderboard/subInterval**	/{*gamemode*}/{*statistic*}/	• *numPages*
			• *pageSize*
			• *startPage*
				{
				"totalLength": ?
				"positions": [{
				rank: ?,
				userId: ?,
				statistic: ?
				},
				...
				]
				}

Richiesta dell'intervallo di pagine in cui è presente il giocatore cercato	**/leaderboard/subInterval**	/{*gamemode*}/{*statistic*}/	• *numPages*
			• *pageSize*
			• *playerId*
				{
				"totalLength": ?
				"positions": [{
				rank: ?,
				userId: ?,
				statistic: ?
				},
				...
				]
				}

Figura 16: REST API per il retrieval delle classifiche

3. **Definizione dello scopo della API**: Le classifiche devono poter essere visualizzate nella View front-end dedicata che consente:

   - **Navigazione paginata** delle classifiche, in linea con il requisito non funzionale RNF1
   - **Caching di più pagine contemporaneamente** in liena con il requsito RNF2.

4. **Funzionalità della API**: L'API deve quindi permettere

   - Il recupero di un numero definito di pagine (`numPages`), a partire da una pagina iniziale (`startPage`) **oppure** a partire da un `playerID`
   - La determinazione della dimensione di ogni pagina (`pageSize`) richiesta

   Inoltre:

   - La risposta deve includere un campo che indichi la dimensione totale della classifica richiesta (`totalLength`), per consentire l'accesso diretto all'ultima pagina dalla View.

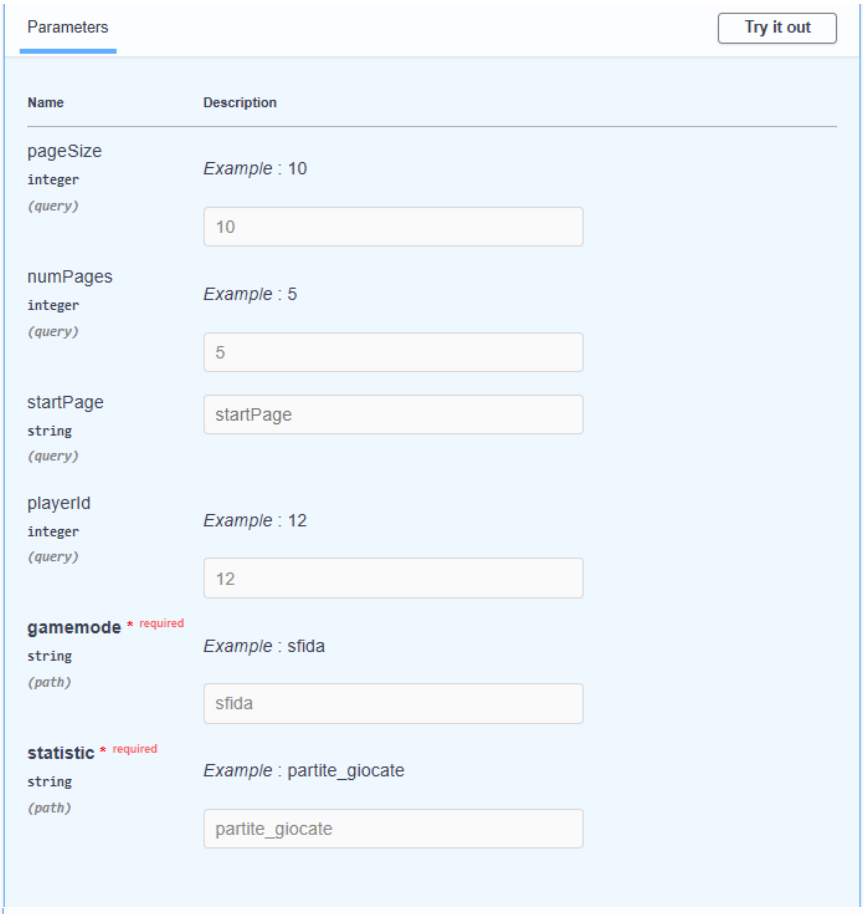
5. **Progettazione degli endpoint**: L'endpoint scelto per l'accesso alle classifiche è:**/leaderboard/subInterval**, a cui vanno aggiunti i parametri della richiesta distinti in:

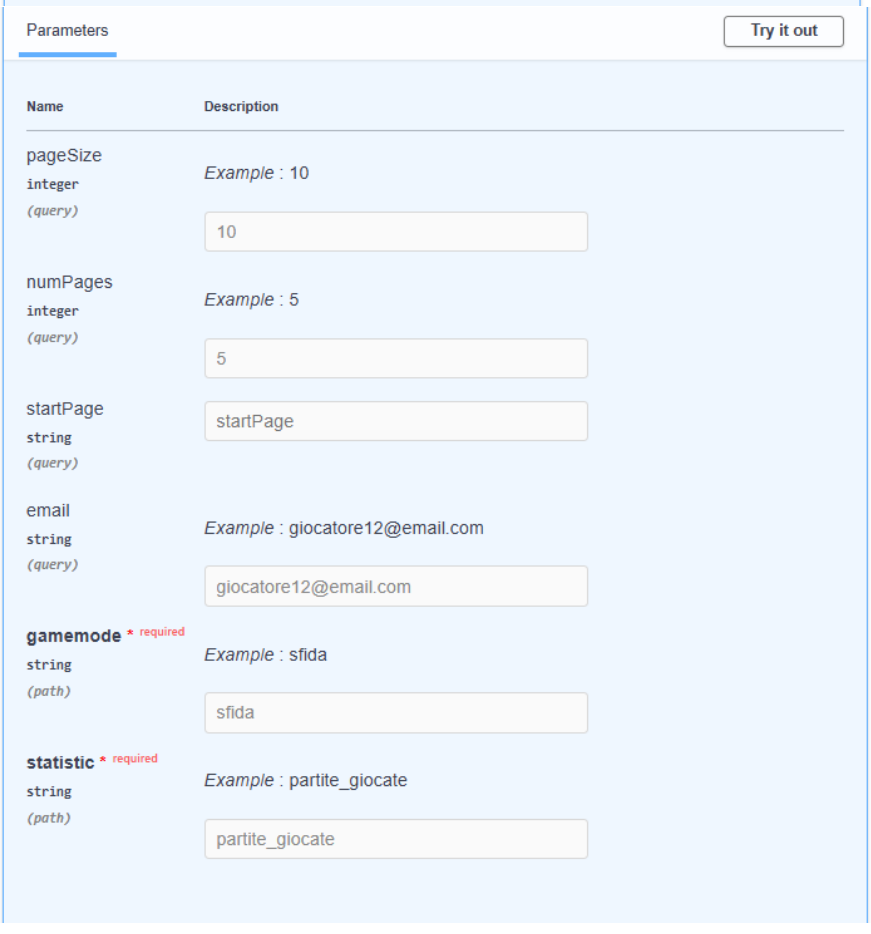
   - **Path parameters**:
     - {`gamemode`}: Modalità di gioco
     - {`statistic`}: Statistica di riferimento
   - **Query parameters**:
     - `numPages`: numero di pagine da recuperare
     - `pageSize`: numero di righe per ogni pagina
     - `startPage`: pagina iniziale per il recupero (opzionale)
     - `playerID`: ID del giocatore da cercare nella classifica (opzionale)

   I due parametri `startPage` e `playerID` sono mutuamente esclusivi.

6. **Considerazioni Aggiuntive**: Nell'integrazione con il T5, l'API è identica ma il parametro `playerID` è sostituito da `email`. Quest'ultima viene utilizzata per ottenere l'ID del giocatore corrispondente attraverso il T23.

Alla luce di questo procedimento, sono state formalizzate le seguenti API di cui si fornisce una vista creata usando Swagger (standard OpenAPI).


Parameters
Try it out
Name
Description
pageSize
integer
(query)
Example : 10
10
numPages
integer
(query)
Example : 5
5
startPage
string
(query)
startPage
playerId
integer
(query)
Example : 12
12
gamemode * required
string
(path)
Example : sfida
sfida
statistic * required
string
(path)
Example : partite_giocate
partite_giocate


Parameters
Try it out
Name
Description
pageSize
integer
(query)
Example : 10
10
numPages
integer
(query)
Example : 5
5
startPage
string
(query)
startPage
email
string
(query)
Example : giocatore12@email.com
giocatore12@email.com
gamemode * required
string
(path)
Example : sfida
sfida
statistic * required
string
(path)
Example : partite_giocate
partite_giocate

4.5 Component Diagram aggiornato

Il Component Diagram in Figura 17 evidenzia in rosso le modifiche effettuate, in particolare la nuova interfaccia esposta dal Game Repository.

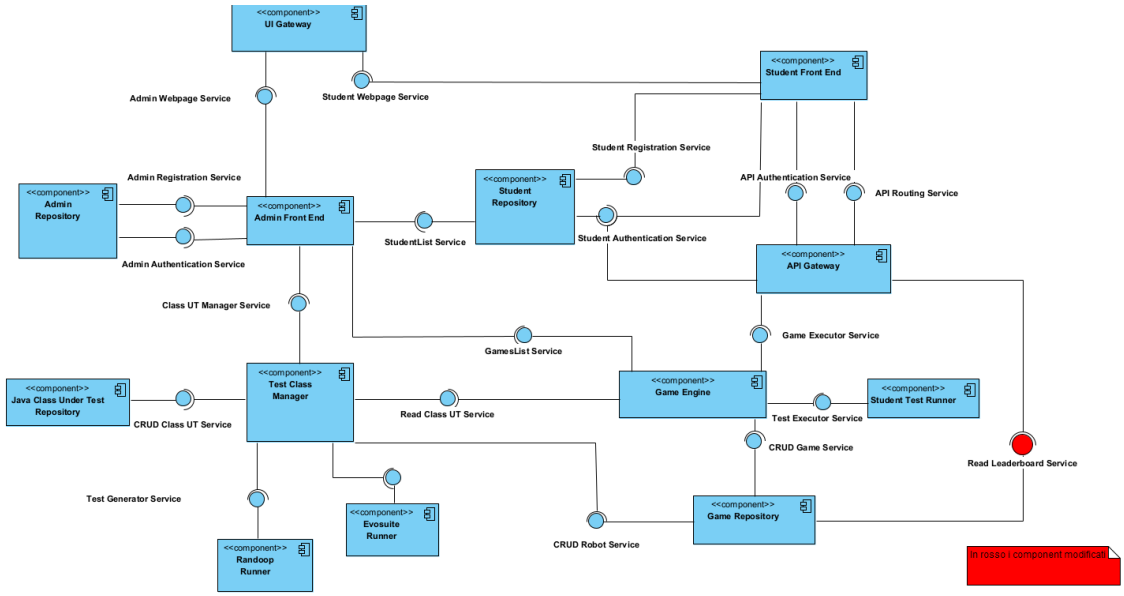


Figura 17: Component Diagram aggiornato

4.6 Issue rilevate durante lo sviluppo

Per effettuare delle prime prove, abbiamo provato a giocare delle partite per verificare l'aggiornamento delle statistiche ma abbiamo riscontrato alcuni problemi.

1. **Creazione di un Game (risolto)**: quando si comincia a giocare una partita, questa non viene memorizzata nella tabella Game nel Game Repository (T4). Questo problema è stato facilmente risolto osservando che nel `GameController` presente in T5 alla richiesta POST `/StartGame` era presente in riga 220 l'istruzione commentata `//gameLogic.CreateGame();` responsabile dell'inoltro della richiesta POST verso il T4 per la creazione del Game. Decommentando questa riga, il problema sembra essersi risolto.

2. **Registrazione della vittoria (non risolto)**: quando si vince una partita, il T5 dovrebbe richiedere al T4 l'aggiornamento dell'attributo `IsWinner` alla riga corrispondente al GameID della partita vinta. Dai log del T4 si è evinto che la richiesta viene ricevuta ma la risposta riporta lo status code `405 METHOD_NOT_ALLOWED`: infatti, il T5 richiede l'aggiornamento della partita attraverso una richiesta di tipo POST, mentre il Controller delle partite in T4

espone una rotta di tipo PUT causando il rigetto della richiesta. In questo caso, pur cambiando il tipo di richiesta inviata dal T5 la vittoria non viene comunque registrata correttamente, suggerendo la necessità di un'ispezione più approfondita dei meccanismi di gestione di fine partita.

3. **Risoluzione dell'associazione tra l'ID di un giocatore e la sua email (parzialmente risolto)**: poichè il T4 contiene la relazione tra ID del giocatore e ID della partita giocata mentre nella View delle classifiche si vuole visualizzare l'email del giocatore, questa viene recuperata attraverso il T23 come precedentemente menzionato. Poichè le classifiche sono servite per sottointervalli, va risolta l'associazione soltanto per un sottoinsieme di giocatori. Al momento, il T23 espone API per avere le informazioni di un giocatore dato il suo ID e per avere l'intera lista di giocatori. Abbiamo quindi risolto l'associazione in T5 chiedendo l'intera lista (piuttosto che tante singole chiamate). Per evitare ciò, sarebbe utile aggiungere una rotta in T23 per avere le informazioni di un sottoinsieme di giocatori data una lista di ID.

4. **Presenza di altre View in T5 probabilmente inutilizzate (non risolto)**: nella cartella `/resources/templates` sono presenti due View `leaderboard.html` e `leaderboardScalata.html`, mentre nella cartella `/resources/static/js_old` è presente il sorgente `leaderboardScalata.js`. Questi file sembrano essere relativi a vecchi task non più portati avanti e dovrebbero probabilmente essere eliminati per non generare confusione con il nuovo fragment aggiunto da noi.

5. **Assenza di uno username (non risolto)**: al fine di identificare univocamente i giocatori evitando di esporre dati sensibili come l'email, sarebbe utile far scegliere uno username univoco a ciascun giocatore in fase di registrazione da mostrare anche nelle classifiche pubbliche.

4.7 Moduli aggiunti e modificati

- **T4**:
  - `/api/leaderboard`: nuova cartella in cui sono stati aggiunti i nuovi Controller, Service e Models, oltre ai test di unità per il Controller.
  - `/main.go`: aggiunte inizalizzazioni della nuova tabella PlayerStats, dei TRIGGER per l'aggiornamento delle statistiche, del nuovo Controller e del nuovo endpoint '/leaderboard/subInterval'
  - `/model/model.go`: aggiunta definizione GORM della nuova tabella SQL PlayerStats.
  - `/api/utils.go`: aggiunta funzione di utility per la selezione della colonna della tabella PlayerStats in base alla classifica richiesta
- **T5**:
  - `/src/main/java/com/g2/t5/GuiController.java` : aggiunta nuova rotta GET '/leaderboard/subInterval/gamemode/statistic'
  - `/src/main/java/com/g2/Service`: creato nuovo service Leaderboard-Service.java
  - `/src/main/java/com/g2/Interfaces/T4Service.java`: registrata la nuova rotta getLeaderboardSubinterval
  - `/src/main/java/com/g2/Interfaces/T23Service.java`: registrata la nuova rotta GetUserByEmail
  - `/src/main/java/com/g2/Game/GameController.java`: decommentata riga 220 (chiamata a funzione per il salvataggio della partita)
  - `/src/main/resources/static/t5/` : è stato aggiunto il fragment "leaderboard.html", il sorgente "leaderboard.js" e lo stylesheet "leaderboard.css", è stato aggiunto il riferimento allo stylesheet nel fragment "header.html", infine è stato aggiunto il bottone "Classifiche" ed il nuovo fragment "leaderboard.html" è stato incluso nel fragment "navbar.html"
  - `/src/main/resources/static/t5/Icone/` : aggiunte icone utilizzate nella UI delle classifiche
  - `/src/main/resources/lang/` : aggiunte traduzioni italiano, inglese e spagnolo per i testi presenti nella UI delle classifiche

- **T23**:
  - `/db_setup/Controller.java`: aggiunta rotta GET '/students_list/getByEmail/{email}'
- **API-Gateway**
  - `/src/main/resources/application.yml` : registrato il nuovo service Leaderboard (rotta '/api/leaderboard/subInterval/')

Inoltre, è stata creata la cartella **TestClassifica** nella directory principale A13 che contiene i test delle nuove API effettuati con Postman. Il file `README.md` chiarisce il contenuto delle cartelle e la procedura per ripetere i test.

5 Testing

5.1 Test di unità T4

Al fine di testare il Controller presente in T4 è stato utilizzato un Mock che sostituisce il service sottostante. In particolare tutti i seguenti casi di test suppongono come pre-condizione che la tabella PlayerStats sia popolata dai seguenti tre utenti:

- ID: 0, PlayerID: 1, SfidaWonGames: 5, SfidaPlayedGames: 7
- ID: 1, PlayerID: 2, SfidaWonGames: 3, SfidaPlayedGames: 6
- ID: 2, PlayerID: 3, SfidaWonGames: 2, SfidaPlayedGames: 2

Test case ID	Descrizione	Input	Output Attesi	Output Ottenuti	Esito
1	Player non esistente	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=10, NumPages=1, PlayerID=9, startPage=””	http.StatusNotFound	http.StatusNotFound	PASS
2	Player esistente	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=10, NumPages=1, PlayerID=1, startPage=””	http.StatusOK	http.StatusOK	PASS
3	PlayerID formato errato	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=10, NumPages=1, PlayerID=”aaa”, startPage=””	http.StatusBadRequest	http.StatusBadRequest	PASS
4	Modalità formato errato	Modalità=1, Statistica=”partite_giocate”, PageSize=10, NumPages=1, PlayerID=1, startPage=””	http.StatusBadRequest	http.StatusBadRequest	PASS
5	Statistica formato errato	Modalità=”sfida”, Statistica=2, PageSize=10, NumPages=1, PlayerID=9, startPage=””	http.StatusBadRequest	http.StatusBadRequest	PASS

6	PageSize vuoto	Modalità=”sfida”, Statistica=2, PageSize=””, NumPages=1, PlayerID=2, startPage=””	http.StatusBadRequest	http.StatusBadRequest	PASS
7	NumPages formato errato	Modalità=”sfida”, Statistica=2, PageSize=10, NumPages=”#”, PlayerID=1, startPage=””	http.StatusBadRequest	http.StatusBadRequest	PASS
8	Pagina richiesta non esistente	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=10, NumPages=1, PlayerID=””, StartPage=3	http.StatusNotFound	http.StatusNotFound	PASS
9	Pagina richiesta esistente	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=10, NumPages=1, PlayerID=””, StartPage=1	http.StatusOK	http.StatusOK	PASS
10	Pagina richiesta formato errato	Modalità=”sfida”, Statistica=”partite_vinte”, PageSize=10, NumPages= 1, PlayerID=””, StartPage=”aaa”	http.StatusBadRequest	http.StatusBadRequest	PASS
11	PlayerID e StartPage presenti entrambi corretti	Modalità=”sfida”, Statistica=”partite_vinte”, PageSize=10, NumPages=1, PlayerID=1, StartPage=1	http.StatusBadRequest	http.StatusBadRequest	PASS
12	PlayerID e StartPage entrambi assenti	Modalità=”sfida”, Statistica=”partite_vinte”, PageSize=10, NumPages=1, PlayerID=””, StartPage=””	http.StatusBadRequest	http.StatusBadRequest	PASS
13	NumPages negativo	Modalità=”sfida”, Statistica=”partite_vinte”, PageSize=10, NumPages=-1, PlayerID=1, StartPage=””	http.StatusBadRequest	http.StatusBadRequest	PASS

14	PageSize negativo	Modalità=”sfida”, Statistica=”partite_vinte”, PageSize=-10, NumPages=1, PlayerID=1, StartPage=””	http.StatusBadRequest	http.StatusBadRequest	PASS

5.2 Test API T4 con Postman

I seguenti test suppongono che la tabella PlayerStats in T4 sia popolata con 99 righe contenenti le statistiche di altrettanti giocatori, in particolare le partite giocate e le partite vinte.

Test case ID	Descrizione	Input	Output Attesi	Output Ottenuti	Esito
1	Input corretti con playerId nullo	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=10, NumPages=5, StartPage=1, PlayerID=””	Expected_Status=200 , Response content type=”application/json”, Response schema is valid, Page is included in the response	Expected_Status=200 , Response content type=”application/json, Response schema is valid, Page is included in the response	PASS
2	Input corretti con startPage nullo	Modalità=”sfida”, Statistica=”partite_vinte”, PageSize=10, NumPages=5, StartPage=””, PlayerID=3	Expected_Status=200 , Response content type=”application/json”, Response schema is valid	Expected_Status=200 , Response content type=”application/json”, Response schema is valid	PASS
3	Modalità formato errato	Modalità=10, Statistica=”partite_giocate”, PageSize=10, NumPages=5, StartPage=5, PlayerID=””	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
4	Modalità errata	Modalità=”sfid@”, Statistica=”partite_giocate”, PageSize=10, NumPages=5, StartPage=5, PlayerID=””	Expected_Status=400 , Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
5	Modalità non inserita	Modalità=””, Statistica=”partite_giocate”, PageSize=10, NumPages=5, StartPage=5, PlayerID=””	Expected_Status=400, Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
6	Statistica errata	Modalità=”sfida”, Statistica=”partite#vinte”, PageSize=10, NumPages=5, startPage=””, PlayerID=12	Expected_Status=400, Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
7	Statistica formato errato	Modalità=”sfida”, Statistica=-1, PageSize=10, NumPages=5, startPage=””, PlayerID=12	Expected_Status=400, Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
8	PageSize dimensione nulla	Modalità=”sfida”, Statistica=”partite_vinte”, PageSize=0, NumPages=5, startPage=””, PlayerID=12	Expected_Status=400, Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
9	PageSize dimensione negativa	Modalità=”sfida”, Statistica=”partite_vinte”, PageSize=-3, NumPages=5, startPage=””, PlayerID=12	Expected_Status=400, Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS

10	PageSize formato errato	Modalità=”sfida”, Statistica=”partite_vinte”, PageSize=”tre”, NumPages=5, startPage=””, PlayerID=12	Expected_Status=400, Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
11	PageSize vuoto	Modalità=”sfida”, Statistica=”partite_vinte”, PageSize=””, NumPages=5, StartPage=””, PlayerID=12	Expected_Status=400, Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
12	NumPages negativo	Modalità=”sfida”, Statistica=”partite_vinte”, PageSize=3, NumPages=-5, StartPage=””, PlayerID=12	Expected_Status=400, Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
13	NumPages nullo	Modalità=”sfida”, Statistica=”partite_vinte”, PageSize=3, NumPages=0, StartPage=””, PlayerID=12	Expected_Status=400, Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
14	NumPages formato errato	Modalità=”sfida”, Statistica=”partite_vinte”, PageSize=3, NumPages=”quattro”, StartPage=””, PlayerID=12	Expected_Status=400, Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
15	NumPages vuoto	Modalità=”sfida”, Statistica=”partite_vinte”, PageSize=3, NumPages=””, StartPage=””, PlayerID=12	Expected_Status=400, Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
16	StartPage nullo	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=10, NumPages=5, StartPage=0, PlayerID=””	Expected_Status=400, Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
17	StartPage negativo	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=10, NumPages=5, StartPage=-3, PlayerID=””	Expected_Status=400, Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
18	StartPage formato errato	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=10, NumPages=5, StartPage=”tre”, PlayerID=””	Expected_Status=400, Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS

19	StartPage vuoto con playerId vuoto	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=10, NumPages=5, StartPage=””, PlayerID=””	Expected_Status=400, Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
20	PlayerId nullo	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=10, NumPages=5, StartPage=””, PlayerID=0	Expected_Status=400, Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
21	PlayerId negativo	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=10, NumPages=5, StartPage=””, PlayerID=-3	Expected_Status=400, Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
22	PlayerId formato errato	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=10, NumPages=5, StartPage=””, PlayerID=”cinque”	Expected_Status=400, Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
23	PlayerId non esistente	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=10, NumPages=5, StartPage=””, PlayerID=200	Expected_Status=400, Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
24	StartPage non esistente	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=10, NumPages=5, StartPage=200, PlayerID=””	Expected_Status=400, Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
25	PlayerId overflow	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=10, NumPages=5, StartPage=””, PlayerID=9223372036854-775808	Expected_Status=400, Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
26	StartPage overflow	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=10, NumPages=5, StartPage=9223372036854-775808, PlayerID=””	Expected_Status=400, Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS

27	PageSize overflow	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=9223372036854-775808, NumPages=5, StartPage=3, PlayerID=””	Expected_Status=400, Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
28	NumPages overflow	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=10, NumPages=9223372036854-775808, StartPage=1, PlayerID=””	Expected_Status=400, Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS

5.3 Test API T5 con Postman

I seguenti test presuppongono che la tabella Students in T23 contenga 99 utenti e che la tabella PlayerStats in T4 sia popolata con 99 righe, ciascuna rappresentante le statistiche di altrettanti giocatori, in particolare "Partite giocate" e "Partite vinte".

Test case ID	Descrizione	Input	Output Attesi	Output Ottenuti	Esito
1	Input corretti con email nulla	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=10, NumPages=5, StartPage=1, Email=””	Expected_Status=200 , Response content type=”application/json”, Response schema is valid, Page is included in the response	Expected_Status=200 , Response content type=”application/json, Response schema is valid, Page is included in the response	PASS
2	Input corretti con startPage nullo	Modalità=”sfida”, Statistica=”partite_vinte”, PageSize=10, NumPages=5, StartPage=””, Email=”giocatore10@email.com”	Expected_Status=200 , Response content type=”application/json”, Response schema is valid, Email is included in the response	Expected_Status=200 , Response content type=”application/json, Response schema is valid, Email is included in the response	PASS
3	Email non esistente	Modalità=”sfida”, Statistica=”partite_vinte”, PageSize=10, NumPages=5, StartPage=””, Email=”marco_postiglione@gmail.com”	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
4	Pagina non esistente	Modalità=”sfida”, Statistica=”partite_vinte”, PageSize=10, NumPages=5, StartPage=200, Email=””	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
5	Modalità formato errato	Modalità=10, Statistica=”partite_giocate”, PageSize=10, NumPages=5, StartPage=5, Email=””	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
6	Modalità errata	Modalità=”sfid@”, Statistica=”partite_giocate”, PageSize=10, NumPages=5, StartPage=5, Email=””	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
7	Statistica errata	Modalità=”sfida”, Statistica=”partitevinte”, PageSize=10, NumPages=5, StartPage=””, Email=”giocatore10@email.com”	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
8	Statistica formato errato	Modalità=”sfida”, Statistica=-1, PageSize=10, NumPages=5, StartPage=””, Email=”giocatore10@email.com”	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS

9	PageSize nullo	Modalità=”sfida”, Statistica=”partite_vinte”, PageSize=0, NumPages=5, StartPage=””, Email=”giocatore10@email.com”	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
10	PageSize negativo	Modalità=”sfida”, Statistica=”partite_vinte”, PageSize=-3, NumPages=5, StartPage=””, Email=”giocatore10@email.com”	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
11	PageSize formato errato	Modalità=”sfida”, Statistica=”partite_vinte”, PageSize=”tre”, NumPages=5, StartPage=””, Email=”giocatore10@email.com”	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
12	PageSize vuoto	Modalità=”sfida”, Statistica=”partite_vinte”, PageSize=””, NumPages=5, StartPage=””, Email=”giocatore10@email.com”	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
13	NumPages negativo	Modalità=”sfida”, Statistica=”partite_vinte”, PageSize=3, NumPages=-5, StartPage=””, Email=”giocatore10@email.com”	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
14	NumPages nullo	Modalità=”sfida”, Statistica=”partite_vinte”, PageSize=3, NumPages=0, StartPage=””, Email=”giocatore10@email.com”	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
15	NumPages formato errato	Modalità=”sfida”, Statistica=”partite_vinte”, PageSize=3, NumPages=”quattro”, StartPage=””, Email=”giocatore10@email.com”	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
16	NumPages vuoto	Modalità=”sfida”, Statistica=”partite_vinte”, PageSize=3, NumPages=””, StartPage=””, Email=”giocatore10@email.com”	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
17	StartPage nullo	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=10, NumPages=5, StartPage=0, Email=””	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS

18	StartPage negativo	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=10, NumPages=5, StartPage=-3, Email=””	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
19	StartPage formato errato	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=10, NumPages=5, StartPage=”tre”, Email=””	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
20	StartPage ed email entrambe vuote	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=10, NumPages=5, StartPage=””, Email=””	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
21	StartPage ed email entrambe non vuote	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=10, NumPages=5, StartPage=4, Email=”giocatore10@email.com”	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
22	StartPage overflow	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=10, NumPages=5, StartPage=9223372036854775808, Email=””	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
23	PageSize overflow	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=9223372036854775808, NumPages=5, StartPage=3, Email=””	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS
24	NumPages overflow	Modalità=”sfida”, Statistica=”partite_giocate”, PageSize=10, NumPages=9223372036854775808, StartPage=1, Email=””	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	Expected_Status=200 , Response content type=”application/json”, Error schema is valid	PASS

5.4 Test End-to-End

I seguenti test presuppongono l'esistenza di classifiche composte da 99 giocatori, ciascuno con punteggi casuali assegnati alle statistiche di merito "Partite giocate" e "Partite vinte". Inoltre, si presuppone che i test vengano eseguiti dalla pagina inziale dopo il login.

Test case ID	Descrizione	Input	Output Attesi	Output Ottenuti	Esito
1	Visione della prima pagina della classifica "Partite giocate" nella modalità "Sfida"	Click su Classifiche	Prima pagina della classifica "Partite giocate" in modalità "Sfida"	Prima pagina della classifica "Partite giocate" in modalità "Sfida"	PASS
2	Visione della prima pagina della classifica "Partite vinte" nella modalità "Sfida"	Click su Classifiche, Click su Partite vinte	Prima pagina della classifica "Partite vinte" in modalità "Sfida"	Prima pagina della classifica "Partite vinte" in modalità "Sfida"	PASS
3	Visione della terza pagina della classifica "Partite giocate" nella modalità "Sfida"	Click su Classifiche, Click su Next, Click su Next	Terza pagina della classifica "Partite giocate" in modalità "Sfida"	Terza pagina della classifica "Partite giocate" in modalità "Sfida"	PASS
4	Visione della terza pagina della classifica "Partite vinte" nella modalità "Sfida"	Click su Classifiche, Click su Partite vinte, Click su Next, Click su Next	Terza pagina della classifica "Partite vinte" in modalità "Sfida"	Terza pagina della classifica "Partite vinte" in modalità "Sfida"	PASS
5	Visione dell'ultima pagina della classifica "Partite giocate" nella modalità "Sfida"	Click su Classifiche, Click su Last page	Ultima pagina della classifica "Partite giocate" in modalità "Sfida"	Ultima pagina della classifica "Partite giocate" in modalità "Sfida"	PASS
6	Visione dell'ultima pagina della classifica "Partite vinte" nella modalità "Sfida"	Click su Classifiche, Click su Partite vinte, Click su Last page	Ultima pagina della classifica "Partite vinte" in modalità "Sfida"	Ultima pagina della classifica "Partite vinte" in modalità "Sfida"	PASS
7	Tentata visione della pagina successiva oltre l'ultima della classifica "Partite giocate" nella modalità "Sfida"	Click su Classifiche, Click su Last page, Click su Next	Ultima pagina della classifica "Partite giocate" in modalità "Sfida"	Ultima pagina della classifica "Partite giocate" in modalità "Sfida"	PASS
8	Tentata visione della pagina successiva oltre l'ultima della classifica "Partite vinte" nella modalità "Sfida"	Click su Classifiche, Click su Partite vinte, Click su Last page, Click su Next	Ultima pagina della classifica "Partite vinte" in modalità "Sfida"	Ultima pagina della classifica "Partite vinte" in modalità "Sfida"	PASS
9	Ricerca posizionamento di un giocatore esistente nella classifica "Partite giocate" nella modalità "Sfida", tramite email valida	Click su Classifiche, Click su "Cerca giocatore", Scrivi "giocatore8@email.com", Click su "Cerca"	Pagina della classifica "Partite giocate" in modalità "sfida" contenente il giocatore "giocatore8@email.com"	Pagina della classifica "Partite giocate" in modalità "sfida" contenente il giocatore "giocatore8@email.com"	PASS
10	Ricerca posizionamento di un giocatore esistente nella classifica "Partite vinte" nella modalità "Sfida", tramite email valida	Click su Classifiche, Click su Partite vinte, Click su "Cerca giocatore", Scrivi "giocatore8@email.com", Click su "Cerca"	Pagina della classifica "Partite vinte" in modalità "sfida" contenente il giocatore "giocatore8@email.com"	Pagina della classifica "Partite vinte" in modalità "sfida" contenente il giocatore "giocatore8@email.com"	PASS

11	Ricerca posizionamento di un giocatore non esistente nella classifica "Partite giocate" nella modalità "Sfida", tramite email valida	Click su Classifiche, Click su "Cerca giocatore", Scrivi "giocatore@email.com", Click su "Cerca"	Messaggio "Non è stato trovato alcun giocatore"	Messaggio "Non è stato trovato alcun giocatore"	PASS
12	Ricerca posizionamento di un giocatore non esistente nella classifica "Partite vinte" nella modalità "Sfida", tramite email valida	Click su Classifiche, Click su Partite vinte, Click su "Cerca giocatore", Scrivi "giocatore8@email.com", Click su "Cerca"	Messaggio "Non è stato trovato alcun giocatore"	Messaggio "Non è stato trovato alcun giocatore"	PASS
13	Ricerca posizionamento di un giocatore esistente nella classifica "Partite giocate" nella modalità "Sfida", tramite email non valida	Click su Classifiche, Click su "Cerca giocatore", Scrivi "giocatore@email.com", Click su "Cerca"	Messaggio "Formato email non valido"	Messaggio "Formato email non valido"	PASS
14	Ricerca posizionamento di un giocatore esistente nella classifica "Partite vinte" nella modalità "Sfida", tramite email non valida	Click su Classifiche, Click su Partite vinte, Click su "Cerca giocatore", Scrivi "giocatore8@email.com", Click su "Cerca"	Messaggio "Formato email non valido"	Messaggio "Formato email non valido"	PASS

6 Sviluppi futuri

6.1 Possibili miglioramenti

Le possibili direzioni di miglioramento individuate possono riguardare:

- la UI: lo stile potrebbe essere reso più accattivante e meno sterile, in accordo con la gamification dell'applicativo;
- le Issues menzionate al Paragrafo 4.6: in particolare, implementare una API nel T23 che permetta il retrieval di un sotto-insieme di utenti eviterebbe di dover richiedere, e successivamente scorrere, l'intera lista di utenti (Issue n.3). Inoltre, l'eliminazione dei file inutilizzati menzionati nella Issue n.4 eviterebbe di generare confusione su quali file sono effettivamente inerenti alla View delle classifiche;
- l'implementazione di nuove classifiche secondo il procedimento menzionato nel Paragrafo 6.2.

6.2 Estensione a nuove classifiche

Il nostro gruppo ha progettato la struttura necessaria alla visualizzazione delle classifiche e la ricerca dei giocatori al loro interno, concentrandosi in particolare sulle partite giocate e vinte in modalità Sfida.
Per estendere la funzionalità a nuove modalità o nuove statistiche, lasciamo di seguito gli step da seguire:

- **T4**:
    1. Aggiungere una colonna alla tabella SQL PlayerStats nel Model GORM (`'T4-G18/model/model.go'`), ad esempio `ScalataPartiteConsecutive`;
    2. Definire nomi univoci per la nuova modalità/statistica ed aggiungerle alle Map presenti in `'T4-G18/api/leaderboard/service.go'`. Ciò servirà ad effettuare il mapping tra i valori dei Path param in ingresso alla API e la colonna del database;
    3. Definire i nuovi TRIGGER SQL nel sorgente `'T4-G18/main.go'` per implementare la logica di aggiornamento della nuova statistica nella tabella PlayerStats.

- **T5**:

    1. Aggiungere i selettori di modalità e statistica nel fragment `/src/main/resources/templates/fragments/leaderboard.html`
    2. Aggiornare l'oggetto `statisticOptions` nel sorgente `/src/main/resources/static/t5/js/leaderboard.js`