ENACTest - Manutenzione evolutiva

Achievement e Sito Documentazione

A13.2

Submitted by:

Carmine Palmese

M63001624

28.11.2024

Contents

1 Introduzione

La progettazione e sviluppo che verranno di seguito presentati puntano ad introdurre nel progetto didattico **ENACTest** un sistema per la gestione degli **Achievement** o **Traguardi**. Per poterli visualizzare e riconoscerli a particolari giocatori, si è pensato di introdurre anche una **pagina di profilo utente**.

Gli Achievement sono un argomento ampiamente trattato nel contesto della **gamification** di procedure o compiti da far svolgere ai fruitori di un servizio, al fine di renderne l'utilizzo piacevole e stimolante.

Dal momento che il progetto si trova in uno stato già avanzato di sviluppo, si è progettata una **manutenzione evolutiva** per integrare questa funzionalità, dando luogo alla stima della bontà dell'attuale architettura di progetto e quindi alla facilità di integrazione di funzionalità di gamification.

1.1 Gamification

Per **Gamification** si intende l'impiego di elementi mutuati dai giochi ma in contesti non ludici, o più specificamente, l'introduzione di **meccaniche di gioco** al fine di coinvolgere gli utilizzatori a sentirsi gratificati nel risolvere un particolare problema.

Con questa tecnica, è possibile adattare vari processi affinché diano livelli di stimoli e coinvolgimento simili alle attività ludiche.

L'implementazione di un'infrastruttura di gamification avviene attraverso i cosiddetti **game elements** o elementi di gioco, che sono distinguibili in tre diverse categorie, in base al livello di astrazione: **dinamiche**, **meccaniche** e **componenti**:

- una **dinamica** rappresenta l'elemento di gioco a più alto livello di astrazione, che può essere quindi associata all'introduzione di **regole** o di una **narrativa** di base. Tra le più importanti dinamiche vi sono anche quella di **progressione**, che puntano a far percepire il senso di miglioramento dei giocatori con il susseguirsi dei giochi, e la dinamica di **relazione**, che punta a far confrontare e relazionare i vari giocatori;
- una **meccanica** rappresenta una serie di processi che, legandosi a *più di-*

namiche di gioco, sono in grado di portare avanti le azioni dei giocatori. Tra le meccaniche più gettonate ci sono sicuramente la **competizione**, la **cooperazione** e le **sfide**, ma sono frequenti anche quelle di acquisizione di **ricompense**;

- un **componente** rappresenta una specifica manifestazione delle meccaniche di gioco, ad esempio un **traguardo** o **achievement**, che può eventualmente prendere la forma di un **distintivo** da sfoggiare su un pagina di **profilo utente**, in una sezione in cui sarà possibile avere delle **collezioni** di traguardi.

A valle di questa breve presentazione di un processo di gamification, si andranno a presentare le soluzioni pensate per essere introdotte in questa versione.

1.1.1 Statistiche

La **statistica** è il concetto portante di tutto il processo di progettazione di questa soluzione. Nel contesto della gamification, una statistica rappresenta una dinamica di progressione. Si punta ad utilizzare le statistiche per ottenere conteggi relativi ai progressi fatti da un giocatore nel corso del suo storico di gioco.

Ad esempio, una statistica potrebbe essere il **numero di partite vinte** da un giocatore, o il **punteggio cumulativo** ottenuto in una determinata modalità di gioco.

1.1.2 Achievement

Gli **achievement** vengono associati ad una ed una sola statistica, in modo da poter definire un valore di soglia oltre il quale si possa ottenere un distintivo.

Ad esempio, un achievement chiamato "giocatore in erba" potrebbe essere associato alla statistica di **numero di partite vinte**, e si potrebbe considerare ottenuto una volta che le partite vinte siano 3.

1.1.3 Profilo Utente

La **pagina di profilo utente** è utile a favorire la competizione tra i giocatori. Questa è quindi uno strumento per visualizzare gli attuali progressi delle statis-

tiche di un giocatore e gli achievement ottenuti.

Dovrebbe inoltre essere possibile visualizzare alcune informazioni sommarie dell'utente, come **dati anagrafici**, **username**, **immagine di profilo** e un'eventuale **biografia**.

1.2 Sito di documentazione

La documentazione di progetto, così come si presenta quella che si sta leggendo in questo momento, è suddivisa in molti PDF sparsi che comprendono le varie versioni dell'applicativo, e spesso la navigazione tra questi rende il tutto estremamente dispersivo.

Si è pensato all'introduzione di un **sito di documentazione** su cui ospitare le versioni più aggiornate delle specifiche dell'applicativo.

Lo scopo è quello di creare un sistema di navigazione della documentazione più fruibile per i nuovi sviluppatori.

2 Analisi dei requisiti

In questa sezione si andranno a formalizzare i requisiti funzionali richiesti precedentemente, corredandoli ad un'opportuna descrizione che agevolerà il processo di progettazione e implementazione.

2.1 Stato attuale del progetto

Per progettare al meglio i requisiti evolutivi, sarà necessario asserire uno stato attuale di progetto. Siccome il progetto è abbastanza ampio e modulare, si approfondiranno solo i dettagli di alcuni dei moduli che lo compongono. Si veda innanzitutto il **Deployment Diagram** attuale:

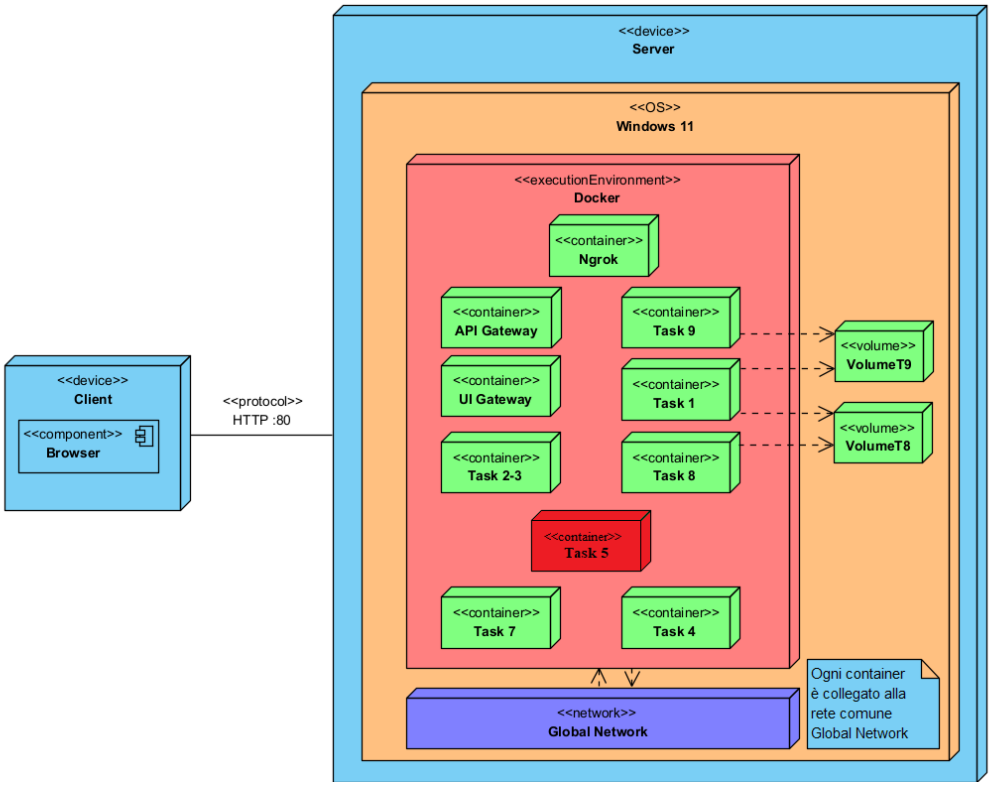

Figure 2.1: Deployment Diagram del progetto

I container chiamati **task** si riferiscono ai diversi moduli di progetto, ognuno con un suo contesto funzionale. Si farà riferimento in particolare ai moduli:

- **Modulo T1:** servizio che si occupa dell'autenticazione degli amministratori e del relativo cruscotto per l'aggiunta di nuove classi da testare e per la

configurazione delle modalità di gioco,

- **Modulo T4:** servizio che si occupa di mantenere una repository dei dati di gioco, contenendo quindi uno storico delle partite giocate, dei round di ogni partita e varie associazioni tra esse.
- **Modulo T5 o T5/6:** servizio che si occupa di realizzare l'interfaccia utente e permettere all'utente di selezionare una modalità di gioco e di cimentarsi in una particolare sfida attraverso un editor di testo.

Successivamente si vedranno nel dettaglio gli stati progettuali di questi servizi e come verranno modificati per lo scopo del task evolutivo attuale.

2.2 Dettaglio dei requisiti

Per quanto riguarda la parte di **gamification**, i requisiti funzionali che dovranno essere implementati sono i seguenti:

1. progettare l'entità **statistica**, rappresentata da un **nome** (a scopo di visualizzazione) e da un **ruolo**, che ne indicherà il tipo di valore rappresentato dal suo progresso (es.: punteggio, numero di partite giocate, ecc...); la statistica è inoltre associata eventualmente ad una **modalità di gioco** o da uno specifico **robot**;
2. fornire la possibilità ad un **amministratore** di **creare** una statistica o di poterla **eliminare** da un opportuno pannello di gestione;
3. progettare l'entità **achievement**, che verrà associato a una ed una sola statistica, indicandone un valore di soglia oltre il quale l'achievement può ritenersi ottenuto;
4. fornire la possibilità ad un **amministratore** di **creare** un achievement e di poterlo **eliminare**;
5. realizzare un'**associazione** in grado di collegare una statistica a un particolare giocatore ed associarvi un **progresso**;
6. realizzare una **pagina di profilo** contenente informazioni sommarie su un utente dato il suo *ID*, i progressi delle sue statistiche e dei suoi achievement;
7. progettare un sistema di **aggiornamento delle statistiche**, che opportu-

namente innescato provvederà a calcolare tutte le statistiche definite dagli amministratori, dato lo storico di un utente;

8. progettare un sistema di **aggiornamento degli achievement**, che opportunamente innescato provvederà a calcolare quali achievement sono stati ottenuti da un particolare utente.

Per quanto riguarda il **sito di documentazione**, i requisiti funzionali che dovranno essere implementati sono i seguenti:

1. progettare una **piattaforma di deployment** su cui ospitare un sito contenente la documentazione dell'intero progetto;
2. progettare un **template di documentazione** che verrà ospitato sulla piattaforma, per rendere omogenea la stesura della documentazione.

2.3 Diagrammi dei casi d'uso

Dai precedenti requisiti, l'obiettivo è quello di estrarre gli opportuni **diagrammi dei casi d'uso**.

Siccome la configurazione degli achievement e delle statistiche, secondo i requisiti, deve essere una cosa quanto più flessibile possibile, si è ritenuto opportuno aggiungere dei casi d'uso al **modulo T1** in modo da estendere le possibilità di un amministratore nel suo pannello di controllo. Attualmente, il diagramma dei casi d'uso di T1 è il seguente:

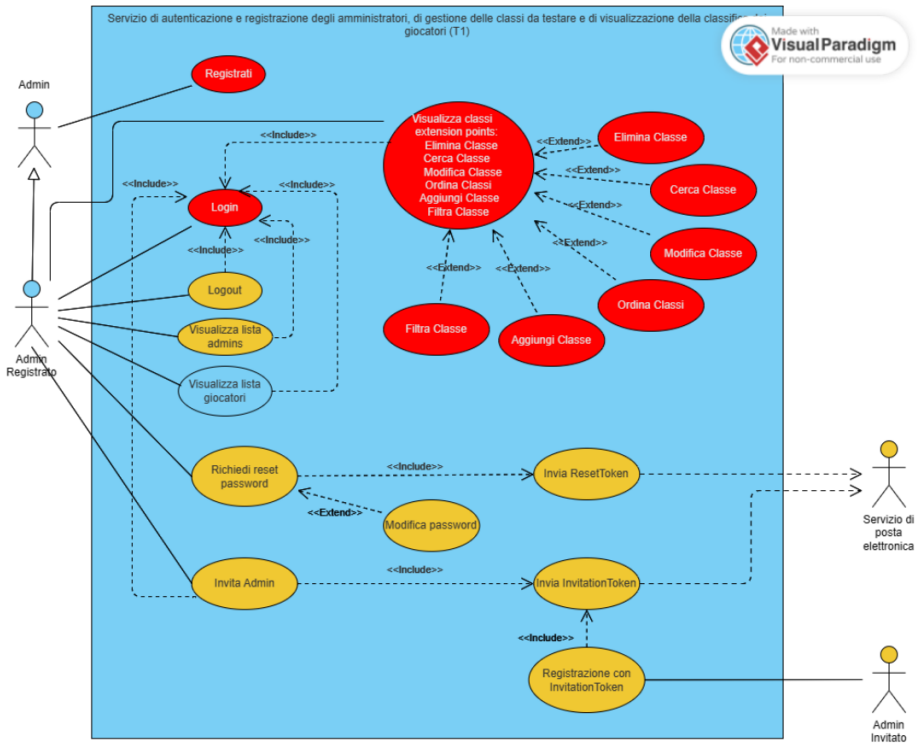

Figure 2.2: diagramma dei casi d'uso di T1, prima delle modifiche

Dal momento che nessuna delle nuove funzionalità intacca qualcuna delle precedenti, si è preferito realizzare un nuovo diagramma dei casi d'uso più sintetico, per evidenziare i nuovi (in verde) rispetto ai precedenti:

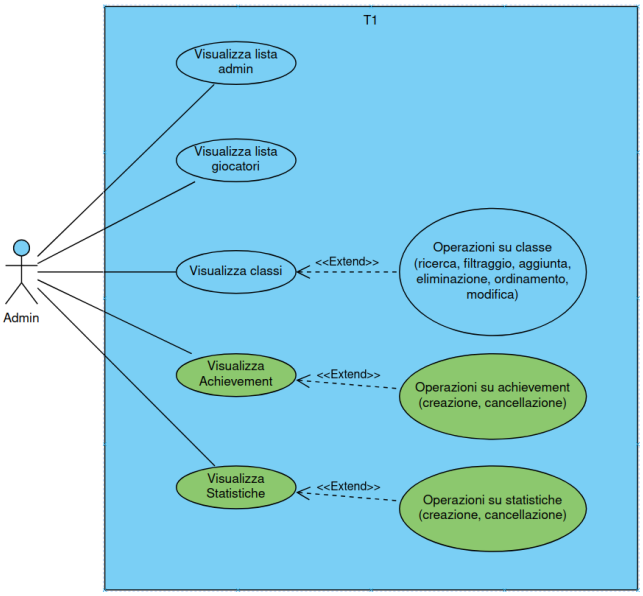

Figure 2.3: diagramma (sintetico) dei casi d'uso di T1, dopo le modifiche

Per quanto riguarda il salvataggio dei progressi delle statistiche per ogni giocatore, si è deciso di operare sul **modulo T4**, che è in effetti il contenitore del repository di gioco. Il diagramma dei casi d'uso del task è stato modificato come segue:

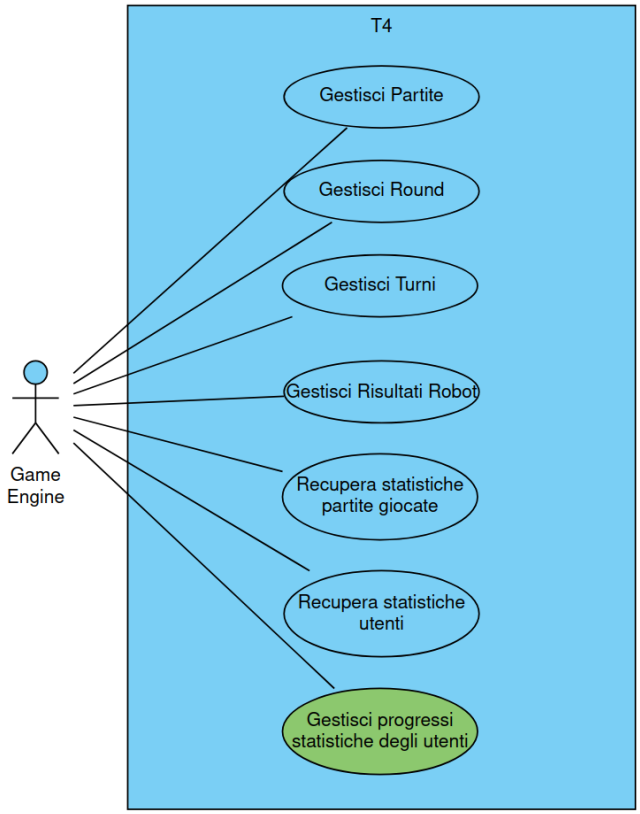

Figure 2.4: diagramma dei casi d'uso di T4

Nota: anche se il Game Engine non può propriamente definirsi un attore, è stato commesso questo abuso di concetto per segnalare che i casi d'uso sono attivabili da un componente esterno, che è proprio il Game Engine.

Infine, è stato opportuno apportare delle modifiche anche al modulo T5. In questo modulo, è stata necessario introdurre non solo la visualizzazione del profilo di un particolare utente, ma anche un'estensione della procedura di "salvataggio partita" che includesse anche l'aggiornamento dei progressi del giocatore che la stava giocando. A questo scopo, è stato anche qui riportato un diagramma dei casi d'uso semplificato, che evidenzia in verde i nuovi casi d'uso:

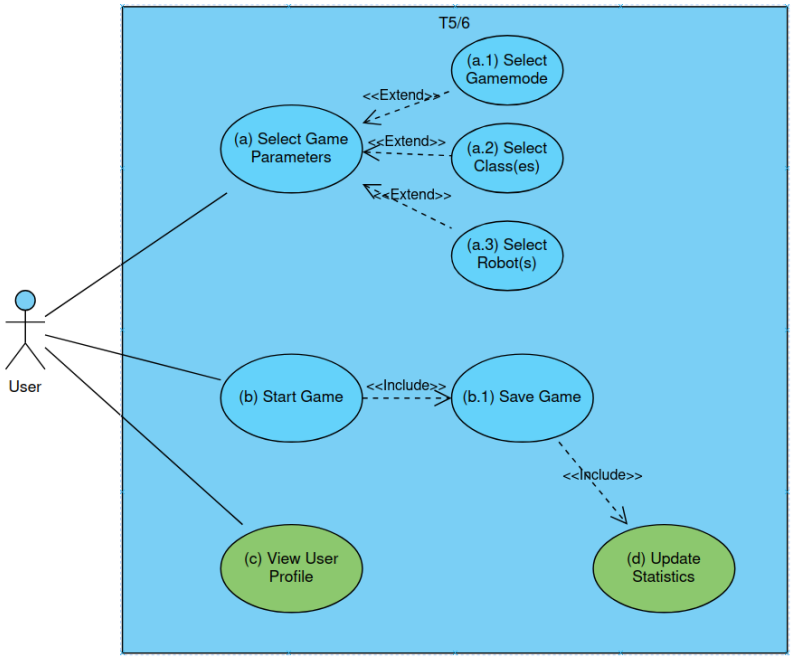

Figure 2.5: diagramma dei casi d'uso di T5/6

3 Progettazione

In questo capitolo si discuteranno delle scelte progettuali effettuate a valle dell'analisi dei requisiti.

3.1 Progettazione Achievement

Per avere una visione d'insieme dell'architettura che verrà distribuita sui vari moduli di progetto, si presenta un diagramma delle classi comprensivo di tutte le relazioni ed entità che compongono la gestione del sistema degli achievement:

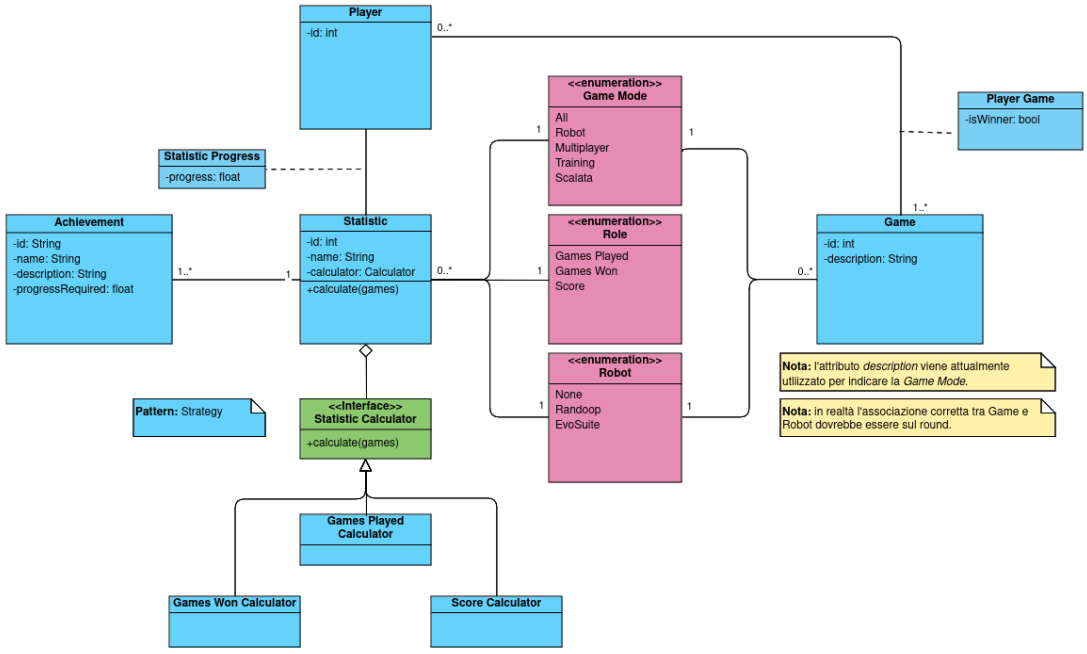

Figure 3.1: diagramma delle classi degli achievement

Si noti che il punto di snodo centrale del diagramma è proprio la statistica. Il calcolo del progresso della statistica è stato delegato a un'interfaccia con pattern di tipo **Strategy**, per diversificare le diverse tecniche di calcolo a seconda del ruolo della statistica. Si noti inoltre che il calcolo della statistica è basato su una lista di *Game*, che può essere preventivamente filtrata per **modalità di gioco** e per **robot** a seconda del tipo di statistica.

3.1.1 Modulo Commons

Ai fini di semplicità di progettazione e implementazione, è stato aggiunto un modulo **commons** che raggruppa una serie di classi a cui accedono più moduli diversi. Questo ha favorito la **riusabilità** del codice e la **manutenibilità** in quanto una modifica su una singola classe ha reso effettive le modifiche su tutti i moduli che la utilizzassero.

Il modulo non viene deployato come gli altri, ma viene semplicemente compilato prima di essi, per poter essere utilizzato come dipendenza esterna comune a tutti.

3.1.2 Modulo T1

Per il modulo T1, non sono state apportate sensibili modifiche all'architettura già esistente. Ciò che è stato fatto è stato principalmente realizzare nuovi documenti per il database **MongoDB** che rappresentassero **Achievement** e **Statistiche**. A questi sono state abbinate delle **repository** per l'implementazione delle operazioni di CRUD, e una serie di endpoint di **API** per concedere l'accesso a queste risorse da altri servizi.

È stato inoltre integrato, nell'interfaccia utente già esistente, un cruscotto per la creazione di statistiche e achievement.

3.1.3 Modulo T4

Il modulo T4 contiene in particolare un database, con un diagramma ER così definito:

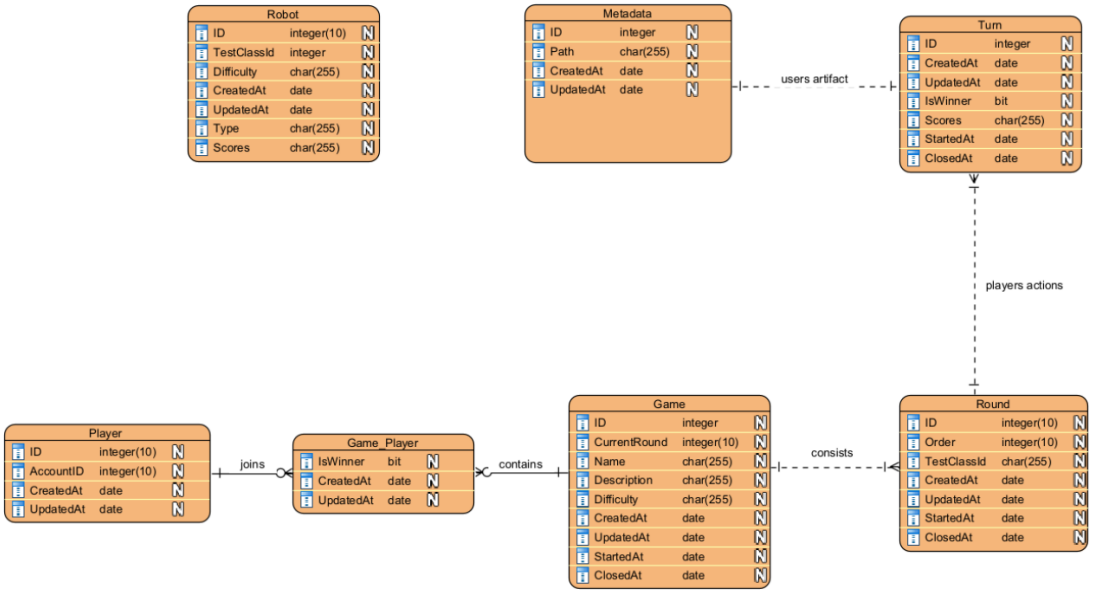

Figure 3.2: diagramma ER di T4, prima delle modifiche

Per aggiungere il progresso associato a una statistica come entità, è stato necessario semplicemente modificare il diagramma in questo modo:

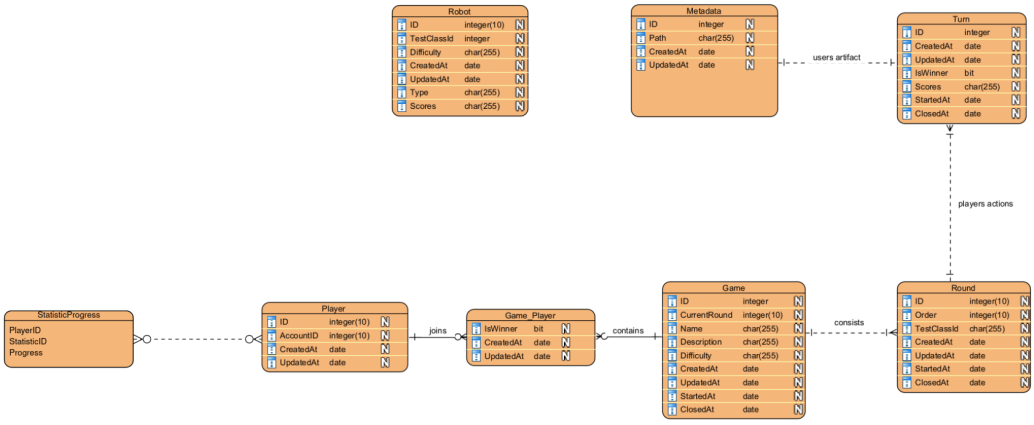

Figure 3.3: diagramma ER di T4, dopo le modifiche

A valle di ciò, all'interno dell'architettura del servizio è stato necessario sia inserire il nuovo **model** dell'entità che rappresentasse il progresso (così come indicato nel diagramma ER) sia integrare un **service** e un **controller** per servire le richieste API di accesso a queste entità.

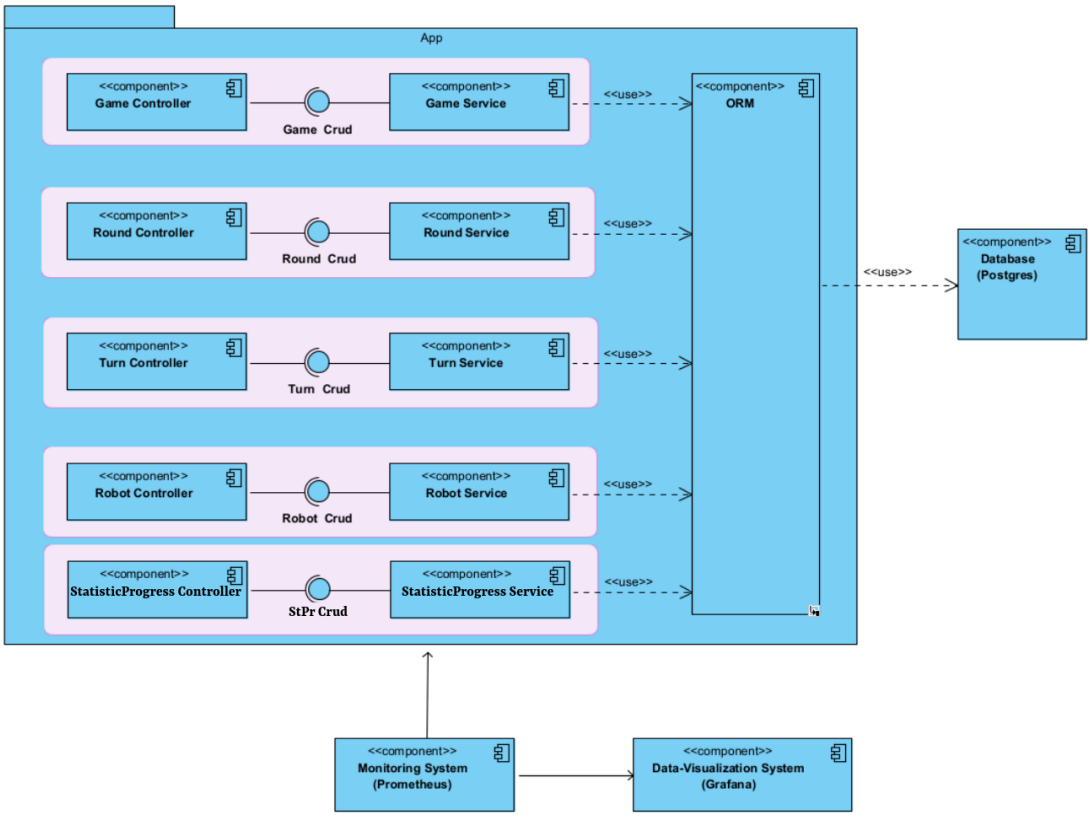

Figure 3.4: Component Diagram dei controller di T4

È stato infine necessario includere gli endpoint API per le nuove richieste, associandole agli opportuni controller.

3.1.4 Modulo T5

Il modulo T5 è quello che ha subito le modifiche più significative, seppur poco invasive in termini dell'architettura pre-esistente.

Innanzitutto è stato necessario realizzare un **Service** che contenesse le logiche di aggiornamento e di prelievo degli achievement. Questo servizio viene utilizzato dall'attuale **Game Controller** per eseguire l'aggiornamento delle statistiche per un determinato giocatore a valle del salvataggio di una partita, e dalle schermate di **Profilo Utente** per ottenere informazioni riguardo i progressi di un giocatore. Con riferimento alla versione su cui si basa questa evoluzione, si riporta il diagramma dei componenti del Game Controller:

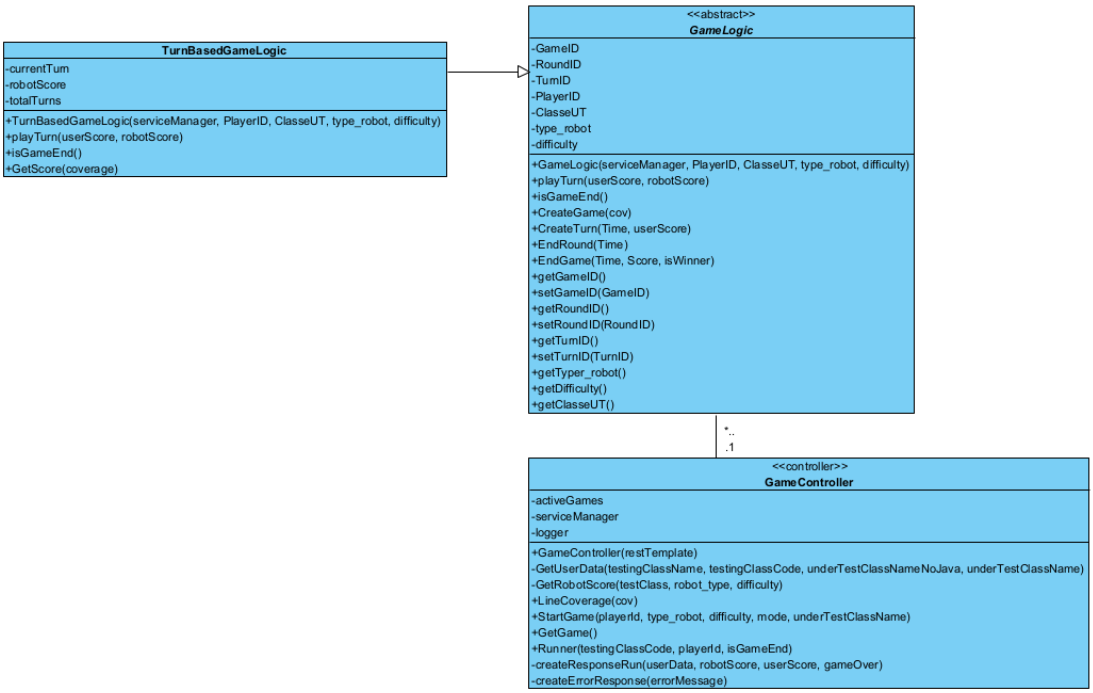

Figure 3.5: diagramma dei componenti del Game Controller

All'interno del metodo *Runner* del GameController, è stata inserita una chiamata al servizio di gestione degli achievement. In particolare, si riporta (a valle della procedura di *run*) il processo di aggiornamento degli achievement attraverso un Sequence Diagram:

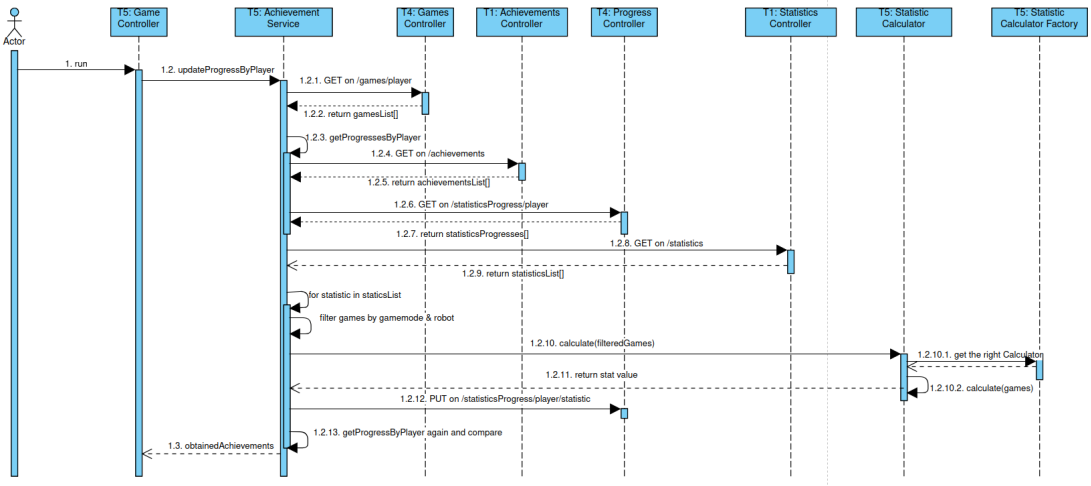

Figure 3.6: sequence diagram del processo di *updateProgressByPlayer*

L'**Achievement Service** contiene anche altre funzionalità, come ad esempio

getProgressesByPlayer, che fornisce la lista di tutti gli achievement con i progressi relativi a un determinato utente:

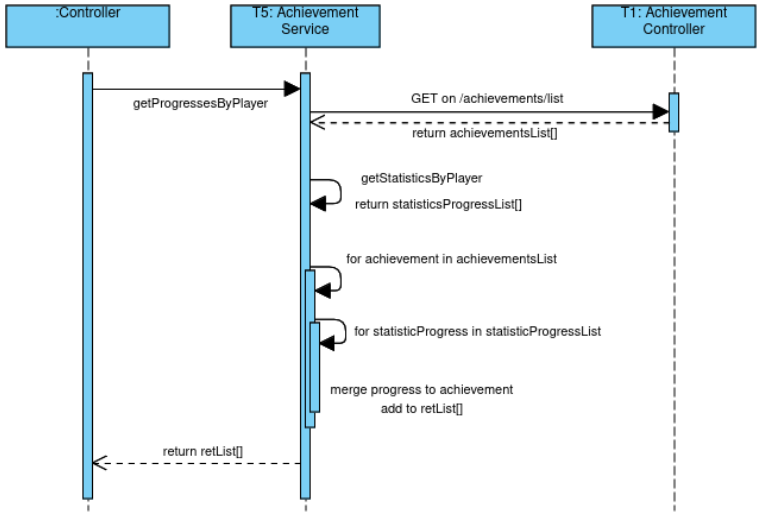

Figure 3.7: sequence diagram di *getProgressesByPlayer*

Allo stesso modo esiste anche il metodo *getStatisticsByPlayer*, che è equivalente ma restituisce una lista di statistiche anziché di achievement.

3.1.5 Architettura finale complessiva

Si riporta a fini di chiarezza un **Package Diagram** con l'architettura complessiva della gestione degli achievement:

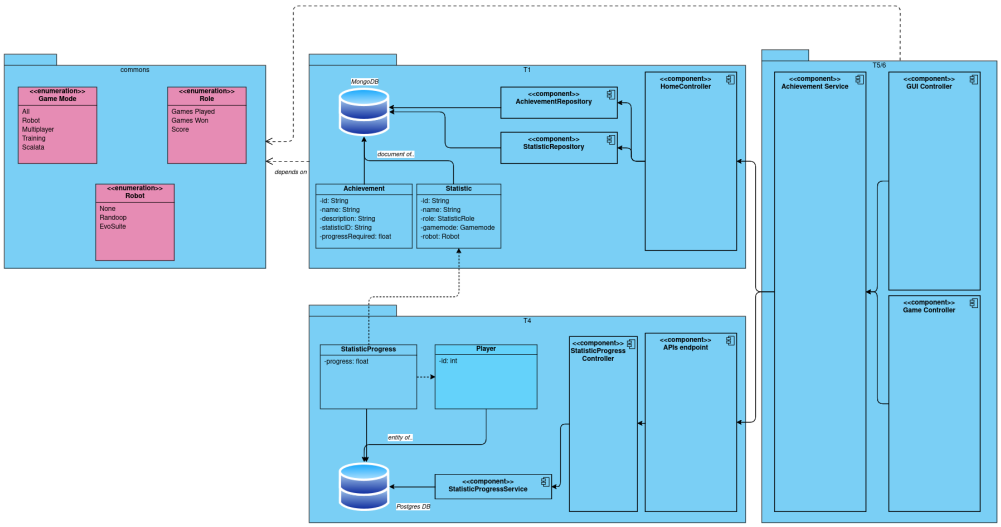

Figure 3.8: package diagram dell'architettura complessiva

Si noti come i due casi d'uso principali, cioè la visualizzazione della pagina di profilo utente e l'aggiornamento delle statistiche, siano rappresentati rispettivamente dal **GUI Controller** e dal **Game Controller**.

Entrambi, ai fini dei casi d'uso sopra citati, dipendono e utilizzano l'**Achievement Service**, sia per richiedere i progressi di statistiche e achievement sia per aggiornarli. Sarà poi l'Achievement Service ad occuparsi di comunicare con gli appositi moduli per ottenere le informazioni richieste.

Si presenta infine un Sequence Diagram che sintetizza il processo di costruzione della pagina utente, facendo uso di alcune funzioni precedentemente descritte:

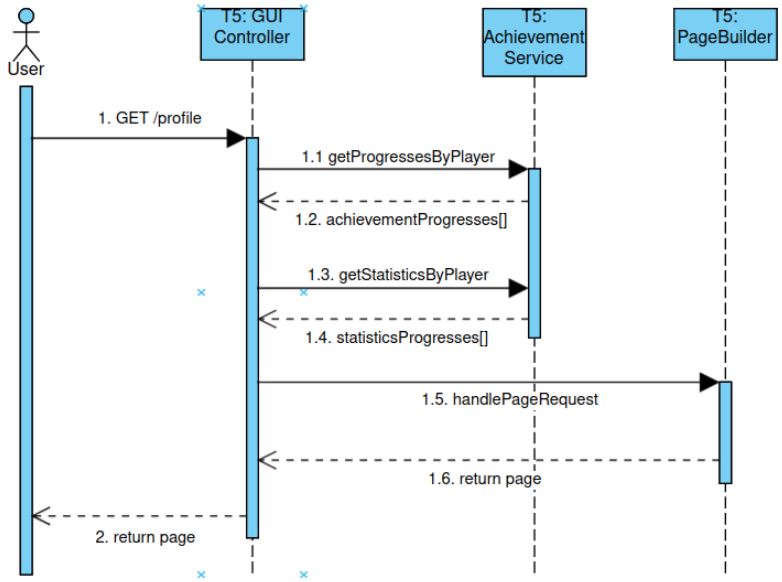

Figure 3.9: sequence diagram di richiesta pagina di profilo

3.2 Progettazione sito

Per la progettazione del **sito di documentazione**, si è scelto di fare affidamento sul servizio di hosting **GitHub Pages**, poiché gratuito e direttamente collegabile a una specifica repository di GitHub (in questo caso quella del progetto).

Il layout e la presentazione delle pagine del sito sono gestite attraverso l'utilizzo del tool **Docsify**, che mette a disposizione una serie di feature per l'organizzazione di una documentazione software.

Grazie a questo tool si può pensare di gettare le basi per uno **standard di versionamento** di documentazione del progetto. In questo modo le evoluzioni successive del software potranno essere documentate dinamicamente e lo stato complessivo del progetto sarà reperibile su di una sorgente centralizzata, e su cui sarà possibile tenere costantemente aggiornata ogni parte del progetto.

4 Conclusioni e sviluppi futuri

Dai test effettuati si evince che il sistema degli Achievement funziona correttamente, con aggiornamenti dettati dal salvataggio di una partita a valle del suo completamento. Ci sono tuttavia alcune funzionalità che non sono totalmente integrate nella soluzione, per via di alcune lacune di progettazione pregresse.

Innanzitutto, per il corretto calcolo di una statistica, il modello di una partita (*game*) dovrebbe essere corredato di specifici campi di modalità di gioco (*gamemode*) e di *robot* contro il quale la partita viene giocata. Mentre per il primo campo, in questa versione, si è fatto uso della **descrizione** di una partita (in quanto presente ma inutilizzata), il secondo campo ha dato diversi problemi di natura progettuale, in quanto un robot è associato individualmente a ogni *round* di una partita. Per l'introduzione di questa feature è quindi necessario portare avanti il concetto di associazione di robot ad una partita, di definirlo meglio a livello di dinamiche di gioco e solo poi filtrare opportunamente la lista dei giochi su cui calcolare la statistica.

Inoltre, la pagina di profilo utente è attualmente accessibile solo per l'utente che ha attualmente eseguito l'accesso, seppure la chiamata all'endpoint di */profile* mostri correttamente la pagina di un altro utente se viene corredato di opportuno ID. C'è quindi bisogno di collegare questa view in qualcosa che potrebbe essere una leaderboard, o una lista di amici, in modo da permettere agli utenti di confrontarsi tra loro e avere una ristretta cerchia di amici con cui poter condividere progressi.

Un'ulteriore dinamica che si potrebbe considerare per le statistiche è quella di sviluppare il concetto in modo da ottenere uno **storico di statistiche**, per presentare all'utente dei grafici di progresso e visualizzare il suo miglioramento progressivo durante le partite.