Robot Scout : Tiger Tiger

Dante Baiunco
Capolavoro 5°anno ITIS G.Peano 2023/2024

Descrizione
Un'applicazione desktop in C#, sviluppata con il motore Unity, che manipola i dati di un database attraverso delle RESTful API fornite da un server scritto in Java. Il server utilizza il formato JSON per lo scambio di dati.
All'interno dell' applicazione desktop oltre strumenti per la visualizzazione e manipolazioni dati è presente un videogame integrato.

Tuto il progetto è reperibile a
https://github.com/PolloRequem/RobotTigerTigerEX

1. Database

Logico
Missioni(id, nome, *robot*, *player*, punteggio, dataInizio, dataFine).
Ritrovamenti (id, *missione, materiale,* data)
Materiali (id, colore, descrizione)
Players (id, username, role, email, punteggioTotale, missioniCompletate, hash, salt)
Robots (nome, *seriale, nPorte, nRuote)*

-Players

Ogni player tiene il conteggio dell totale del proprio punteggio e delle missioni che ha completato.
Il ruolo influirà sulle possibilità del player.
Per ottenere più sicurezza e non mostrare in chiaro le password dei player, ho deciso di utilizzare la crittografia hash con l'aggiunta del salt, una sequenza di caratteri casuali , aggiunto alla password nella funzione unidirezionale.

Quando viene cancellato un player vengono cancellate tutte le sue missioni , di conseguenza anche tutti i ritrovamenti.

-Missioni

Ogni missione viene iniziata con punteggio 0 e senza data di fine. Una volta iniziata il player che l'ha creata può aggiungere un numero illimitato di ritrovamenti ad essa finché non viene chiusa. Una missione si può concludere solo attraverso la simulazione di completamento che termina la missione rendendo impossibile aggiungere altri ritrovamenti ed assegnando un punteggio e la dataFine.

2. Applicazione Desktop

Il progetto vuole essere una sorta di gestore software per una realtà che gestisce missioni spaziali con dei robot.

Comprende quindi un sistema di autenticazione dell'utente , chiamato “player” , e un sistema di ruoli.Ci sono due ruoli possibili “user” e “admin”.È possibile anche registrarsi. Una volta aver fatto il login si può accedere a diverse attività che possono variare a seconda del ruolo

dell'utente.Inoltre è presente una LeaderBoard che mostra la classifica dei cinque giocatori con più punti all'interno del DataBase.

2.1 Profilo

L'utente visualizza i propri dati e le missioni che ha iniziato .Può inoltre modificare la sua è mail.

2.2 Missions

La prima opzione, “Missions”, permette di visualizzare tutte le missioni nel database. Qui non ci sono distinzioni tra i ruoli: tutte le missioni vengono visualizzate indipendentemente dal player che le ha iniziate o dal fatto che siano concluse.

2.3 Simulation

Una volta cliccato su “Simulation” il player deve scegliere se vuole completare una missione o vuole andare in “esplorazione”.

2.3.1 *Esplorazione*

Qui , vengono proposte , di nuovo ,due opzioni : iniziare una nuova missione o cercare un ritrovamento in una missione già esistente.
Nel caso si scelga la prima opzione, verrà visualizzata una schermata dove sarà possibile configurare una nuova missione. Qui è necessario specificare il nome della missione e il robot con cui si vuole iniziare. Altri dati, come la data di inizio, il player e l'ID, verranno inseriti automaticamente. Tuttavia, l'ID potrà essere modificato dall'utente, purché rispetti il formato richiesto e sia unico.
Dopodiché l'utente verrà reindirizzato alla pagina della missione appena creata dove potrà aggiungere ritrovamenti.

Se, invece, nell'hub di simulazione, si opta per "existing missions", l'utente avrà la possibilità di selezionare una missione da lui precedentemente

avviata dall'elenco di quelle ancora non completate e aggiungervi un ritrovamento.
Nel caso l'utente fosse un admin potrà visualizzare tutte le missioni non completate tra l'elenco indipendentemente dal player che le ha iniziate.

Quando si aggiunge un ritrovamento ad una missione si avvia la simulazione dove si impersonifica il robot con cui è stata iniziata la missione.Per aggiungere il ritrovamento basterà far muovere il robottino con le frecce direzionali verso un materiale e premere il pulsante “E” per scannerizzarlo , per completare la registrazione del ritrovamento basterà tornare verso la base , la struttura rossa , ed uscire. Il materiale registrato sarà l'ultimo materiale scannerizzato dal robot , che è possibile verificare dall'interfaccia di gioco quando si è ancora nella simulazione. Il parziale del ritrovamento verrà generato casualmente.

2.3.2 *Completare una missione*

L'utente avrà la possibilità di selezionare una missione da lui precedentemente avviata dall'elenco di quelle ancora non completate e di completarla.
Nel caso l'utente fosse un admin potrà visualizzare tutte le missioni non completate tra l'elenco indipendentemente dal player che le ha iniziate.

Quando si clicca su “Complete” si avvierà la simulazione di completamento dove si impersonifica il robot con cui è stata iniziata la missione scelta dall'utente.
L'obiettivo della simulazione di completamento è quella di superare il percorso ad ostacoli facendo più punti possibili.Il gioco è diviso in due fasi: nella prima, la telecamera si muoverà automaticamente verso il basso, mentre nella seconda fase si muoverà verso l'alto ad una velocità maggiore ed inoltre la velocità del robot verrà diminuita.

L'unico modo per perdere ed essere costretti a riprovare la simulazione è uscire dalla visuale della telecamera a scorrimento automatico.Ci sono due modi principali per guadagnare punti: raccogliendo monete e diamanti. Le monete sono sparse lungo tutto il percorso, mentre i diamanti sono di due tipi: piccoli (sei in totale) che garantiscono più punti delle monete, e un diamante grande posizionato alla fine del percorso. Raccogliere il diamante grande non garantisce punti, ma avvia la seconda fase del gioco.
Vi sono anche modi per perdere punti , infatti se il robot dovesse incappare in uno dei tanti ostacoli presenti in tutto il percorso verranno tolti dei punti al player. È possibile finire la missione con punti negativi.

Dopo aver perso o attraverso il menu di pausa, attivabile con il tasto "Esc", è possibile regolare la difficoltà del gioco. Ci sono tre livelli di difficoltà: facile, normale e difficile. Ogni livello influisce sulla velocità della telecamera e del giocatore, e determina il punteggio iniziale della simulazione: -500 per il livello facile, 0 per il livello normale e 500 per il livello difficile.

Una volta terminata la simulazione, comparirà la schermata finale dei punti. Qui saranno visualizzati i punti ottenuti durante la missione appena completata, il totale dei punti accumulati fino a quel momento e il numero di missioni totali completate dal giocatore, aggiornati con la nuova missione. Nel caso in cui un admin avvii una missione non iniziata da lui, i punti e il conteggio delle missioni completate verranno attribuiti a lui ma la missione verrà considerata completata comunque dall'utente che ha iniziato la missione.È possibile giocare alla simulazione anche senza aver effettuato il login ,nel menu principale.

2.4 Players
Attivita disponibile solo agli utenti admin. Permette di visualizzare l'elenco dei players. Inoltre è possibile customizzare o cancellare player non admin. Scegliendo di customizzare un player verrà data la possibilità di cambiare il nome e l'email , di promuovere ad admin e di resettare la password del player selezionato.Cancellando un player vengono cancellate pure tutte le sue missioni ad esso associate.

3. Web Service

Il WebService sfrutta l'architettura REST e offre API per l'applicazione desktop, permettendo la manipolazione e visualizzazione dei dati. Il server è sviluppato in Java e utilizza l'applicazione server di glassfish.Questo compone di tre parti principali:

3.1 JavaBean: Queste classi vengono utilizzate per incapsulare i dati del database in oggetti. Ogni tabella del database ha un corrispondente Bean, garantendo così una rappresentazione strutturata e facile da gestire dei dati.Utilizziamo i jar jackson per serializzare i bean in json e viceversa.

3.2 DAO (Data Access Object): Queste classi sono responsabili delle operazioni di interrogazione e manipolazione del database. Implementano i metodi necessari per eseguire le operazioni CRUD (Create, Read, Update, Delete) sui dati, assicurando un accesso ai dati efficiente e sicuro.

3.3 API: Queste classi utilizzano la libreria Jakarta per esporre i servizi Web RESTful. Le API sono progettate per essere intuitive e facili da usare, fornendo endpoint per varie operazioni sui dati, come recupero, aggiornamento e cancellazione.Le API sono suddivise in due parti distinte:

3.3.1 Authentication: Queste API sono utilizzate per il login e la registrazione degli utenti. Entrambe le operazioni utilizzano il metodo POST per l'autenticazione. Gli endpoint per l'autenticazione si trovano aggiungendo "/authentication" all'URL del server. Queste API garantiscono che solo gli utenti autorizzati possano accedere e interagire con il sistema.

3.3.2 Data: Queste API sono utilizzate per lo scambio di dati relativi agli elementi del database. Utilizzano JSON per il formato di scambio dati, facilitando l'integrazione con l'applicazione desktop e altre applicazioni. Gli endpoint offrono funzioni per recuperare, aggiornare, creare e cancellare dati nel database, assicurando una gestione completa delle informazioni.

4 Perché ho scelto Unity?

Ho deciso di sviluppare il progetto in Unity per quattro motivi principali:
-Conoscenza approfondita: Conosco bene l'engine di Unity e ho colto l'opportunità di dimostrare le mie competenze.
-Interfaccia intuitiva: La creazione di un'interfaccia gradevole è più diretta e veloce da realizzare in Unity rispetto ad altri linguaggi.
-Applicazione desktop: Desideravo sviluppare un'applicazione desktop eseguibile, poiché finora mi sono occupato solo di programmi e siti web.
Integrazione di un videogioco: Volevo inserire un videogioco nel progetto e Unity è la piattaforma migliore per questo scopo.

5. Percorso

Il progetto si è basato su un'esperienza PCTO svolta in classe sotto la supervisione del Professore di Informatica. L'obiettivo di questa esperienza era simulare un'applicazione capace di gestire missioni nello spazio o su altri pianeti, coordinate da ricercatori e svolte da robot, con lo scopo di raccogliere materiali estranei.

Noi studenti dovevamo sviluppare il modello concettuale del database contenente informazioni riguardo ai robot, ricercatori, missioni e così via. Inoltre, dovevamo creare un'applicazione web in grado di visualizzare questi dati. L'obiettivo finale era sviluppare un'applicazione in Java per pilotare un robot EV3 Brick, che tramite un sensore potesse scannerizzare colori. Questa applicazione avrebbe poi dovuto comunicare con un server, il quale avrebbe inserito i dati nel database, utilizzando una connessione TCP.

Partendo da questo concetto, ho iniziato a sviluppare una versione del progetto che implementasse una simulazione virtuale del robot fisico. Successivamente, questa versione è continuata a evolversi e ho iniziato a orientare il progetto verso l'ambito videoludico, dato che stavo sviluppando il progetto in Unity. Quando ho studiato il modello REST e le RESTful API, ho deciso di sostituire la comunicazione TCP con le API.

Ho quindi modificato la struttura del database, aggiungendo elementi videoludici come punteggi, missioni completate e altri elementi simili.

Quindi ho sviluppato le API necessarie e le scene per la manipolazione dati in Unty. Successivamente ho adattato il videogame in base al database.

6 Documentazione Unity

Dato che la maggiorparte del lavoro è stato fatto in Unity ecco alcune cose che può essere importante menzionare

6.1 UnityWebRequest

La classe di Unity che ho utilizzato per inviare richieste API al server supporta tutti i metodi http.

6.2 Newtonsoft.Json

Pacchetto si Unity utilizzato per serializzare classi in json e viceversa.

Come Provarlo

Dato che il progetto è orientato in localhost dovranno essere eseguiti questi passaggi per poter provare il progetto:
(se il server è hostato in internet saltare all ultimo passaggio)

- Scaricare la zip del progetto da https://github.com/PolloRequem/RobotTigerTigerEX
- Scaricare il progetto del JavaWebService locato nella omonima cartella.
- Aprire il progetto con NetBeans, importare i jar necessari che sono all interno del file del Server
- Aggiungere un server GlassFish di versione 6.25 o inferiore
- Hostare il servizio DBMS (per esempio attraverso MySql con Xamp), ed importare il file .sql presente nel progetto
- Dopodichè installare la build dell'applicazione del proprio OS