Introduzione

Klotski è un puzzle composto da blocchi scorrevoli che mette alla prova la logica. L'obiettivo del gioco è quello di spostare un blocco speciale verso una posizione specifica all'interno del piano di gioco. La vera sfida risiede nell'abilità di muovere abilmente gli altri pezzi per creare una via libera al blocco speciale, cercando di raggiungere la posizione finale con il minor numero di mosse possibili. Questo richiede un'attenta pianificazione e un approccio ingegnoso per risolvere il puzzle in modo efficiente. Il gioco è stato implementato come un sito web SPA, la persistenza dei dati degli utenti è stata implementata tramite un sistema di autenticazione. Vi è una netta separazione tra front-end e back-end, questo è stato fatto di proposito al fine di poter riutilizzare il back-end qualora si volesse ricreare il progetto con un front-end diverso, per esempio JavaFX.

Tecnologie e librerie utilizzate

backend

Linguaggio utilizzato

Java	17	Linguaggio di programmazione usato per sviluppare il backend dell'applicazione

Sorgenti esterni utilizzati

SpringBoot	3.0.2	Framework per lo sviluppo di applicazioni con design pattern MVC o MC (come nel nostro caso)
Gson	2.10.1	Libreria per la serializzazione e deserializzazione del json
Lombok	1.18.28	Annotation preprocessor per semplificare la generazione di metodi (Getter, Setter, Builder etc…)
JUnit		

Sistema di gestione delle dipendenze

Gradle	7.6	Sistema per la gestione delle dipendenze del backend. È basato su Maven

frontend

Linguaggio utilizzato

Typescript	5.0.4	Linguaggio di scripting usato per sviluppare il frontend dell'applicazione

Sorgenti esterni utilizzati

Vue	3.2.47	Framework per il templating di pagine web in typescript
Tailwind	3.3.2	Libreria per la semplificazione della scrittura del css
Axios	1.3.6	Libreria per la semplificazione dell'invio di richieste http al backend
Vuex	4.1.0	Framework per lo state managing delle pagine web del frontend
Vite	4.3.2	Bundler

Sistema di gestione delle dipendenze

NodeJs	versione immagine di docker: 18 versione in docker: 18.16.1	Sistema per la gestione delle dipendenze per librerie Javascript

Descrizione del progetto e implementazione delle specifiche

Una volta entrati nella home del sito per iniziare a giocare è necessario registrarsi, se siamo dei nuovi utenti, e fare il login.

Scelta della difficoltà

E' possibile scegliere la difficoltà del puzzle tra semplice, media, difficile e impossibile, in base al livello di sfida che si vuole affrontare nel risolvere il klotski.

Counter delle mosse effettuate

Esiste un contatore che tiene traccia di quante mosse abbiamo fatto fino ad ora per cercare di risolvere il puzzle.

Tasto reset

Permette di azzerare il counter delle mosse e di ritornare alla conformazione iniziale del gioco.

Tasto undo

Nel caso in cui si fosse fatto uno spostamente errato è sempre possibile tornare indietro alla posizione precedente, facendo diminuire di uno il counter delle mosse di questa partita.

Tasto hint

In caso di stallo, in cui non si è in grado di risolvere il puzzle, è possibile usare questo tasto per richiedere un aiuto che consisterà nel muovere un pezzo del gioco (verrà aggiunta una mossa al counter).

Storico delle partite passate

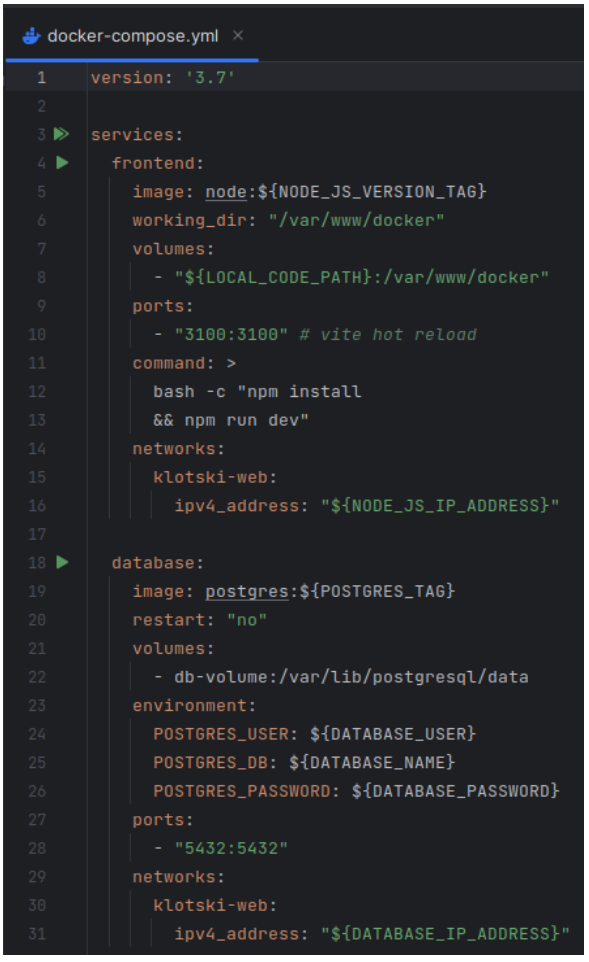
Sono presenti, grazie all'utilizzo di un database, le informazioni sulle ultime 10 partite con il rispettivo numero di mosse impiegate e con la loro data e ora di inizio.

Tasto "Nuova partita"

Da la possibilità di iniziare una nuova partita anche se quella corrente non è stata terminata.

Scaricare ed eseguire il software

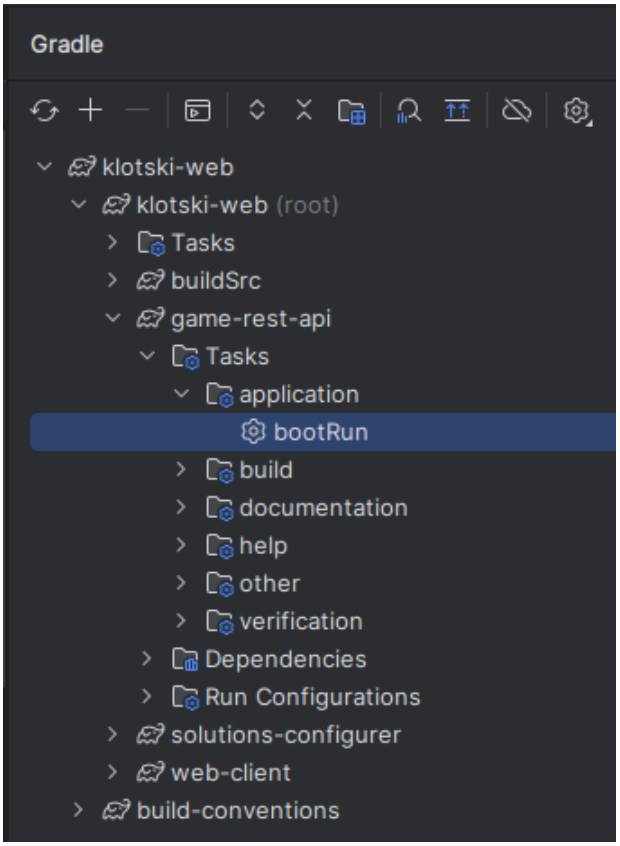
Innanzitutto è necessario preparare l'ambiente installando Docker e Amazon Corretto 17 (versione modificata da Amazon di JAVA 17). Successivamente è necessario fare il download del codice andando sul seguente link che porta alla repository GitHub del progetto e, una volta scaricato, aprirlo usando l'IDE IntelliJ. Ora è necessario aprire il file docker-compose.yml e cliccare sul pulsante verde play che si trova alla riga 4 per avviare il frontend e 18 per avviare il database.



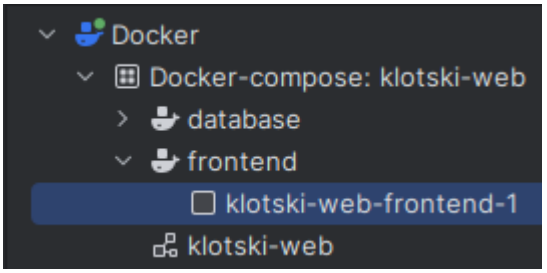
Per avviare il backend è necessario innanzitutto cliccare sull'icona di Gradle (quella evidenziata in blu nell'immagine a lato) che si trova nella parte destra dell'IDE e navigare nelle cartelle nel seguente ordine:

klotski-web/klotski-web/game-rest-api/Tasks/application

Ora è necessario fare doppio clic sull'eseguibile bootRun (visualizzato nell'immagine successiva) presente all'interno del percorso precedentemente descritto per avviare il backend.



Per ottenere il link è necessario cliccare su Services in basso a sinistra ed entrare su klotski-web-frontend-1; si potrà così vedere il link ed entrare nel sito web.



Design patterns

Model Controller
Il "Model-Controller" è un pattern di progettazione architetturale comunemente utilizzato nel contesto dello sviluppo di applicazioni software. È una variante del pattern Model-View-Controller (MVC) che si concentra sulla separazione delle responsabilità tra il modello (Model) e il controller (Controller). Nel nostro caso non possiamo parlare di MVC in quanto la view viene completamente separata dalla logica del progetto esponendo un'API.

Dependency injection con inversion of control
Tutte le dipendenze sono sempre ottenute tramite il costruttore delle classi. Si evita di usare dei Singleton per esprimere appieno le potenzialità di Composizione ed Ereditarietà. L'inversione di controllo è un principio di progettazione che sposta la responsabilità della creazione e gestione degli oggetti dalla classe cliente a un framework o un contenitore. In pratica, questo significa che invece di un oggetto che crea le proprie dipendenze, l'oggetto riceve le dipendenze dall'esterno.

Visitor design pattern
Il Visitor è un design pattern comportamentale che consente di separare l'algoritmo di elaborazione da una struttura di oggetti su cui opera. In altre parole, il Visitor consente di definire nuove operazioni per ogni oggetto della struttura. Nel nostro caso è stato usato per definire dei comportamenti generici dei vari tile presenti nel gioco.

Repository design pattern

Il Repository design pattern è un pattern di progettazione che viene utilizzato per gestire la persistenza dei dati in un'applicazione. Si tratta di una componente dell'architettura del software che si occupa di interagire con la sorgente dati, come un database, un file system o un servizio web, nascondendo i dettagli di implementazione specifici della persistenza dal resto dell'applicazione.