Specifiche Esame per superamento del Corso Programmazione Avanzata

A.A. 2021/2022

Gruppo: SI

Si chiede di realizzare un back-end utilizzando i seguenti framework / librerie:

- Node.JS
- Express
- Sequelize
- RDBMS a scelta del gruppo (es. Postgres, MySQL, sqlite,…)
- https://www.npmjs.com/package/js-chess-engine

Descrizione del progetto:

Si realizzi un sistema che consenta di gestire il gioco di scacchi. In particolare, il sistema deve prevedere la possibilità di far interagire due utenti (autenticati mediante JWT) o un utente (sempre autenticato con JWT) che gioca contro l'intelligenza artificiale. Ci possono essere più partite attive in un dato momento. Un utente può allo stesso tempo partecipare ad una ed una sola partita.

- Dare la possibilità di creare una nuova partita seguendo https://www.npmjs.com/package/js-chess-engine ; la partita può essere:
  - Utente contro utente (le email possono fungere da chiave)
  - Utente contro intelligenza artificiale (IA) scegliendo il livello di difficoltà.
- In particolare, è necessario validare la richiesta di creazione della partita. Per ogni partita viene addebitato un numero di token in accordo con quanto segue:
    - 0.40 all'atto della creazione
    - 0.01 per ogni mossa da parte degli utenti (anche IA)
  - Il modello può essere creato se c'è credito sufficiente ad esaudire la richiesta (se il credito durante la partita scende sotto lo zero si può continuare comunque).
- Creare la rotta per effettuare una mossa in una data partita verificando se questa è ammissibile o meni (si consiglia di valutare quanto presente in *Board Configuration – JSON*)
- Creare una rotta per verificare le partite svolte riportando se sono state vinte o perse, la modalità di gioco (utente contro utente o utente contro IA), il numero di mosse totali, filtrando anche per data di avvio della partita
- Creare una rotta per valutare lo stato di una data partita (di chi è il turno, se è terminata, scacco, scacco matto,…); una partita si considera chiusa quando:
  - C'è uno scacco matto.
  - I due utenti (solo per modalità utente contro utente) concordano sulla chiusura. Per far ciò creare una rotta che consenta di effettuare richieste distinte (una per utente) per confermare l'intenzione di chiudere forzatamente la partita.
  - Per ogni partita vinta al giocatore si da 1 punto; per ogni partita interrotta con comune accordo: 0.1 punti;
- Creare una rotta per restituire lo storico delle mosse di una data partita con la possibilità di esportare:
  - JSON
  - FEN
- Restituire la classifica dei giocatori dando la possibilità di scegliere l'ordinamento ascendente / discendente. Questa rotta è pubblica e non deve essere autenticata.

Le richieste devono essere validate (es. utente che scelga un evento che non esistente).

Ogni utente autenticato (ovvero con JWT) ha un numero di token (valore iniziale impostato nel seed del database).

Nel caso di token terminati ogni richiesta da parte dello stesso utente deve restituire 401 Unauthorized.

Prevedere una rotta per l'utente con ruolo admin che consenta di effettuare la ricarica per un utente fornendo la mail ed il nuovo "credito" (sempre mediante JWT). I token JWT devono contenere i dati essenziali.

Il numero residuo di token deve essere memorizzato nel db sopra citato.

Si deve prevedere degli script di seed per inizializzare il sistema.

Si chiede di utilizzare le funzionalità di middleware.

Si chiede di gestire eventuali errori mediante gli strati middleware sollevando le opportune eccezioni.

Si chiede di commentare opportunamente il codice.

Note:

Nello sviluppo del progetto è richiesto l'utilizzo di Design Pattern che dovranno essere documentati opportunamente nel Readme.MD. È preferibile una implementazione in typescript.

I token JWT da usare possono essere generati attraverso il seguente link: https://jwt.io/

La chiave privata da usare lato back-end deve essere memorizzata un file .env e caricata mediante la libreria

Specifiche Repository

- Il codice deve essere reso disponibile su piattaforma github con repo pubblico
- Nel repository è obbligatario inserire un Readme.md che descriva:
    - Obiettivo del progetto
    - Progettazione
        - diagrammi UML
        - descrizione dei pattern usati motivandone la scelta
    - Come avviare il progetto mediante docker o docker-compose (preferibile) per comporre i servizi richiesti.
    - Test del progetto mediante chiamate effettuate con curl o wget o con Postman
- Il Readme.MD può essere redatto in lingua italiana o inglese (non vi saranno differenziazioni nel processo di valutazione)

Specifiche Consegna

- La consegna avviene esclusivamente mediante moodle all'indirizzo di seguito riportato dove dovranno essere indicati:
    - URL del repository pubblico
    - Commit id che verrà usata dal docente per effettuare la valutazione.
    - Data per lo svolgimento dell'esame
- Indirizzo per la consegna: https://learn.univpm.it/mod/assign/view.php?id=332114

Buon lavoro 😉

Il docente

Adriano Mancini