**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 dello studente (es. Postgres, MySQL, sqlite,…)
* PDFKIT (<https://pdfkit.org/>)

**Descrizione del progetto:**

Si realizzi un sistema che consenta di gestire il gioco della dama. In particolare, il sistema deve prevedere la possibilità di far interagire due utenti (autenticati mediante JWT). 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 specificando (codifica a scelta dello studente):
  + Dimensione della griglia (es. 10x10)
  + e-mail dell’avversario che sarà usata poi per autenticare le richieste mediante token JWT
* Per ogni partita viene addebitato un numero di token in accordo con quanto segue:
  + - 0.35 all’atto della creazione
    - 0.015 per ogni mossa da parte degli utenti
  + Il gioco 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 meno
* Creare una rotta per valutare lo stato di una data partita (di chi è il turno, se è terminata, vincitore,…)
* Creare una rotta per determinare l’abbandono di una partita da parte di un utente.
* Creare una rotta per restituire lo storico delle mosse di una data partita in formato CSV o JSON o PDF
* Dare all’utente la possibilità di ottenere lo storico delle partite fornendo per ogni partita dati quali numero di mosse utente, numero totale di mosse, tempo totale di gioco, sfidante) dando la possibilità di filtrare in modo opzionale per:
  + - Data inferiore ad un valore specificato dall’utente
    - Data superiore ad un valore specificato dall’utente
    - Data compresa tra due valori specificati dall’utente

Si chiede di dare la possibilità di ordinare per data (crescente o decrescente), per numero di mosse totali (crescente o decrescente) dando la possibilità di esportare in formati diversi come CSV o JSON o PDF.

* Creare una rotta che consenta di avere la classifica dei giocatori; la classifica può essere ordinata in modo crescente / decrescente. Per ogni giocatore devono essere riportate anche delle statistiche come il numero medio di mosse effettuate per vincere, il numero di partite perse, il numero di partite totale giocate, il numero di partite vinte per abbandono ed il numero di partite perse per abbandono. Tale rotta non deve essere autenticata (unica rotta “pubblica”)
* Creare una rotta che consenta ad un utente di ottenere il suo credito residuo.
* Creare una rotta che consente all’utente admin di ottenere la lista degli utenti con credito inferiore ad un valore scelto dall’utente stesso
* Creare una rotta che consenta ad un utente con privilegio di admin di ricaricare il credito di un dato utente

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

Appunti:

* regole dama italiana:
  + tutti i pezzi sulle caselle nere
  + pedoni solo avanti
  + dame anche indietro
  + se può mangiare un pezzo lo deve fare
  + quando un pezzo ne mangia un altro si sposta nella casella successiva
  + angolo in basso a destra nera per entrambi
  + se un pezzo che ha mangiato può mangiare un altro pezzo lo può mangiare (?)
  + se un pedone arriva in fondo diventa dama
  + iniziano sempre i neri (assegna a caso)
  + un pedone non può mangiare una dama (?)
  + un giocatore vince se non ha più pezzi o se non può più muoverli

ogni pedone ha un

id key,

posizione\_x foreign key,

posizione\_y foreign key,

posizione iniziale

lista di caselle possibili dove muoversi + check limiti e check occupata,

lista di caselle possibili dove andare dopo aver mangiato + check limiti e check occupata, dama, deve mangiare

table posizione

posizioni\_x 1 … 8 key

posizioni\_y 1…8 key

if r+1 > 8 or c+1 > 8:

break

if not dama or bianco:

break

if r+1 c+1 occupato

controllo fazione

if fazione uguale

break

else if r+2 c+2 esiste and è libera

registra mossa possibile

else

registra mossa possibile