Applicazioni Web I – Esame #3 (scadenza 2024-09-17 alle 23:59)

“Lotto Instantaneo”

VERSIONE FINALE – Le modifiche sono riportate in rosso.

Si realizzi un’applicazione web per giocare a una versione del lotto istantaneo. Il gioco consiste nell’estrazione periodica di un insieme di numeri e nella possibilità da parte dei giocatori di puntare su alcuni numeri per tentare di vincere. I giocatori sono utenti loggati al sito. Un utente non loggato vedrà solo un messaggio che spiega le regole del gioco e avrà la possibilità di fare login.

In dettaglio, ogni due minuti, l’applicazione web presenta ai giocatori un’**estrazione** di cinque numeri nell’intervallo 1-90, generati in maniera *casuale* dal server. Tutti i giocatori collegati all’applicazione web in quel momento devono ricevere, quindi, gli stessi numeri.

Nell’attesa dell’estrazione, ogni giocatore può **puntare** su 1, 2 o 3 numeri distinti. Un giocatore può fare una sola puntata per estrazione. Il controllo sui numeri indovinati deve essere compiuto dal server al momento dell’estrazione.
Le puntate si effettuano con dei punti a disposizione di ogni giocatore. Puntare su un numero costa 5 punti, su due 10 punti e su tre 15 punti. Un giocatore può effettuare una puntata solo se ha sufficienti punti a disposizione per coprire il costo di quella puntata. Ogni giocatore ha un budget iniziale di 100 punti. Il budget si può incrementare solo vincendo al gioco; una volta azzerato non si può più giocare.

A ogni estrazione, possono capitare tre casi:

1. Il giocatore indovina **tutti i numeri** su cui ha puntato. In questo caso l’applicazione mostra un messaggio appropriato e il giocatore vince il doppio dei punti che ha usato per la puntata.
2. Il giocatore non indovina **nessun numero**. L’applicazione mostra un messaggio appropriato e il giocatore non vince alcun punto.
3. Il giocatore indovina **alcuni numeri** ma non tutti. L’applicazione mostra un messaggio appropriato e il giocatore vince punti in maniera proporzionale rispetto ai numeri indovinati nella puntata (secondo la formula *k/n*2*punti_usati_puntata*, dove k è il numero di numeri indovinati e n è il numero di numeri giocati — quindi 2 o 3).

La tabella sottostante riassume i punteggi ottenibili nei tre casi, secondo tutte le puntate possibili.

Puntata	**Numeri indovinati**	**Punti guadagnati**
1	1	10 punti
2	1	10 punti
2	2	20 punti
3	1	10 punti
3	2	20 punti
3	3	30 punti

1, 2 o 3	0	0 punti

L'applicazione include, infine, una pagina con la *classifica* dei 3 giocatori con più punti, ordinata a partire dal giocatore con più punti a disposizione.

L'applicazione dovrebbe prevenire comportamenti malevoli (accesso a informazioni private, puntate al di fuori della finestra corretta, o manomissione dei punteggi, per esempio) tramite appropriate API e controlli vari.

L'organizzazione di queste specifiche in diverse schermate (e possibilmente su diverse route) è lasciata allo studente.

Requisiti del progetto

- L'architettura dell'applicazione e il codice sorgente devono essere sviluppati adottando le migliori pratiche (best practice) di sviluppo del software, in particolare per le single-page application (SPA) che usano React e HTTP API. Le API devono essere protette con cura e il front-end non dovrebbe ricevere informazioni non necessarie.
- L'applicazione deve essere pensata per un browser desktop. La responsività per dispositivi mobile non è richiesta né valutata.
- Il progetto deve essere realizzato come applicazione React, che interagisce con API HTTP implementate in Node.js+Express. La versione di Node.js deve essere quella usata durante il corso (20.x, LTS). Il database deve essere memorizzato in un file SQLite. Il linguaggio di programmazione deve essere JavaScript.
- La comunicazione tra il client ed il server deve seguire il pattern dei "due server", configurando correttamente CORS e con React in modalità "development" con lo Strict Mode attivato.
- La valutazione del progetto sarà effettuata navigando all'interno dell'applicazione. Non saranno testati né usati il bottone di "refresh" né l'impostazione manuale di un URL (tranne /), e il loro comportamento non è specificato. Inoltre, l'applicazione non dovrà mai "autoricaricarsi" come conseguenza dell'uso normale dell'applicazione stessa.
- La directory radice del progetto deve contenere un file `README.md` e contenere due subdirectories (`client` e `server`). Il progetto deve poter essere lanciato con i comandi: "`cd server; nodemon index.mjs`" e "`cd client; npm run dev`". Un template con lo scheletro delle directory del progetto è disponibile nel repository dell'esame. Si può assumere che `nodemon` sia già installato a livello di sistema. Nessun altro modulo sarà disponibile globalmente.
- L'intero progetto deve essere consegnato tramite GitHub, nel repository creato da GitHub Classroom.
- Il progetto **non deve includere** le directory `node_modules`. Esse devono essere ricreabili tramite il comando "`npm install`" subito dopo "`git clone`".
- Il progetto può usare librerie popolari e comunemente adottate (per esempio, `day.js`, `react-bootstrap`, ecc.), se applicabili e utili. Tali librerie devono essere correttamente dichiarate nei file `package.json` cosicché il comando `npm install` le possa scaricare ed installare.
- L'autenticazione dell'utente (login e logout) e l'accesso alle API devono essere realizzati tramite `Passport.js` e cookie di sessione. Le credenziali devono essere memorizzate in formato hashed e con sale. La registrazione di un nuovo utente non è richiesta né valutata.

Requisiti di qualità

In aggiunta all'implementazione delle funzionalità richieste dell'applicazione, saranno valutati i seguenti requisiti di qualità:

- Progettazione e organizzazione del database.
- Progettazione delle HTTP API.
- Organizzazione dei componenti React e delle route.
- Uso corretto dei pattern di React (comportamento funzionale, hook, stato, contesto ed effetti). Questo include evitare la manipolazione diretta del DOM.
- Chiarezza del codice.
- Assenza di errori (e warning) nella console del browser (tranne quelli causati da errori nelle librerie importate).
- Assenza di crash dell'applicazione o eccezioni non gestite.
- Validazione essenziale dei dati (in Express e in React).
- Usabilità e facilità d'uso basica.
- Originalità della soluzione.

Requisiti del database

- Il database del progetto deve essere realizzato dallo studente e deve essere precaricato con almeno cinque utenti registrati, di cui almeno tre con delle puntate già effettuate.

Contenuto del file README.md

Il file README.md deve contenere le seguenti informazioni (un template è disponibile nel repository del progetto). In genere, ogni spiegazione non dovrebbe essere più lunga di 1-2 righe.

1. Server-side:
    a. Una lista delle API HTTP offerte dal server, con una breve descrizione dei parametri e degli oggetti scambiati.
    b. Una lista delle tabelle del database, con il loro scopo.
2. Client-side:
    a. Una lista delle route dell'applicazione React, con una breve descrizione dello scopo di ogni route.
    b. Una lista dei principali componenti React implementati nel progetto.
3. In generale:
    a. Due screenshot dell'applicazione, **uno durante la fase di puntata e uno dopo l'estrazione dei numeri**. Le immagini vanno embeddate nel README linkando due immagini da inserire nel repository stesso.
    b. Username e password degli utenti registrati.

Procedura di consegna

Per sottomettere correttamente il progetto è necessario:

- Essere **iscritti** all'appello.
- Usare il **link** fornito per **unirsi alla classroom** di questo appello su GitHub Classroom (cioè, correttamente **associare** il proprio nome utente GitHub con la propria matricola studente) e **accettare l'assignment**.
- Fare il **push del progetto** nel **branch `main`** del repository che GitHub Classroom ha generato per ognuno. L'ultimo commit (quello che si vuole venga valutato) deve essere **taggato** con il tag **`final`** (nota: `final` deve essere scritto tutto minuscolo e senza spazi ed è un 'tag' git, non un 'messaggio di commit').

Nota: per taggare un commit, si possono usare (dal terminale) i seguenti comandi:

```
# ensure the latest version is committed
git commit -m "...comment..."
git push

# add the 'final' tag and push it
git tag final
git push origin --tags
```

In alternativa, è possibile inserire un tag dall'interfaccia web di GitHub (seguire il link 'Create a new release').

Per testare la propria sottomissione, questi sono gli esatti comandi che saranno usati per scaricare ed eseguire il progetto. Potreste volerli provare in una directory vuota:

```
git clone ...yourCloneURL...
cd ...yourProjectDir...
git pull origin main  # just in case the default branch is not main
git checkout -b evaluation final # check out the version tagged with
'final' and create a new branch 'evaluation'
(cd server ; npm install; nodemon index.mjs)
(cd client ; npm install; npm run dev)
```

Assicurarsi che tutti i pacchetti (package) necessari siano scaricati tramite i comandi `npm install`. Fate attenzione: se alcuni pacchetti sono stati installati a livello globale, potrebbero non apparire come dipendenze necessarie. Controllare sempre con un'installazione pulita.

Fate attenzione al fatto che Linux è case-sensitive nei nomi dei file, mentre macOS e Windows non lo sono. Pertanto, si controllino con particolare cura le maiuscole/minuscole usate nei nomi dei file e negli `import`.