Applicazioni Web I – Esame # 3 (scadenza 2022-09-01 alle 23:59)

"Categorie"

VERSIONE FINALE – le modifiche sono in rosso

Progettare e implementare un'applicazione web per gestire una variante del gioco di parole "Categorie".

Il gioco consiste nell'escogitare, in un tempo limitato, il maggior numero di parole valide distinte tra loro che appartengano ad una data categoria e che inizino con una data lettera. Per esempio, in un turno di gioco ("round"), potrebbe essere richiesto di scrivere il maggior numero di parole possibili che inizino con la lettera "D" e che appartengano alla categoria "Animali". Alla fine di ogni round viene calcolato un punteggio.

L'applicazione deve soddisfare le seguenti specifiche.

Un giocatore inizia un nuovo round di gioco facendo le seguenti cose:

- Selezionando la difficoltà da un insieme di 4 livelli, in cui ogni livello consiste di un differente numero minimo di parole da scrivere per completare con successo il round: minimo 2 parole per il livello di difficoltà 1, 3 per il livello di difficoltà 2, 4 per il livello di difficoltà 3, e 6 per il livello di difficoltà 4.
- Selezionando una categoria tra "Animali", "Nazioni", o "Colori"[1].

La lettera è generata in maniera casuale dall'applicazione web e mostrata all'inizio di ogni round. Solamente le parole che iniziano con quella lettera saranno accettate dall'applicazione. Ogni round ha una durata fissa pari a 60 secondi.

Nel round di 60 secondi, il giocatore deve inserire quante più parole possibile che appartengano alla categoria scelta, che inizino con la lettera data. Al termine del tempo, il round è fallito o passato, con un certo punteggio.

Il round è fallito se ~~nessuna delle parole inserite appartiene alla categoria oppure~~ il numero delle parole inserite che appartengono alla categoria corretta è meno del numero minimo di parole, il cui valore dipende dal livello di difficoltà selezionato. Per il round fallito non è assegnato alcun punteggio.

Altrimenti, il round è passato e il punteggio totale viene mostrato al giocatore. Il punteggio è calcolato come segue:

- 10 punti, moltiplicato per il livello di difficoltà (1…4), per ogni parola corretta non ancora usata in quella categoria negli ultimi 2 round giocati, con la stessa lettera iniziale, da qualsiasi utente. Se tale numero di rounds non sono stati ancora giocati, questa regola non si applica.

---

[1] Le parole che appartengono ad ogni categoria devono essere immagazzinate nel server, controllate al termine del round, e NON devono essere trasferite al client. Le liste di parole categorizzate si possono trovare su diversi siti web, come per esempio:
https://github.com/sroberts/wordlists/blob/master/animals.txt (Animali),
https://gist.github.com/dariusz-wozniak/656f2f9070b4205c5009716f05c94067 (Nazioni), e
https://gist.github.com/mordka/c65affdefccb7264efff77b836b5e717 (Colori).

- 5 punti, moltiplicato per il livello di difficoltà (1…4), per ogni parola corretta per la quale non si applica la regola precedente.
- 0 punti per ogni parola che non appartiene alla categoria.

Il giocatore deve poter interrompere il round in qualsiasi momento prima della scadenza dei 60 secondi. In questo caso, si applicheranno le stesse regole del round completo.

Ogni giocatore può giocare più rounds~~, senza limite~~. Il gioco può essere giocato in modo anonimo (cioè senza richiedere il login, e i punteggi non vengono salvati) o in modo personalizzato. In quest'ultimo caso, l'utente deve effettuare il login prima di iniziare il round, e tutti i punteggi di tutti i round giocati, insieme alla corrispondente categoria, sono registrati nella propria storia personale.

L'applicazione mostra anche una "hall of fame", accessibile senza autenticazione, ~~per ogni categoria,~~ con il nome e il punteggio totale del miglior giocatore per ogni categoria.

L'organizzazione delle funzionalità di questo testo in differenti schermate (e potenzialmente in differenti routes) è lasciata allo studente ed è oggetto di valutazione.

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.
- Il progetto deve essere realizzato come applicazione React, che interagisce con un'API HTTP implementata in Node+Express. Il database deve essere memorizzato in un file SQLite.
- La comunicazione tra il client ed il server deve seguire il pattern "two servers", configurando correttamente CORS, e React deve girare in modalità "development".
- 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.js`" e "`cd client; npm start`". Viene fornito uno scheletro delle directory del progetto. Si può assumere che `nodemon` sia già installato a livello di sistema.
- 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 (come per esempio `day.js`, `react-bootstrap`, ecc.), se applicabili e utili. Tali librerie devono essere correttamente dichiarate nei file `package.json` e `package-lock.json` cosicché il comando `npm install` le possa scaricare ed installare tutte.
- L'autenticazione dell'utente (login) e l'accesso alle API devono essere realizzati tramite `passport.js` e cookie di sessione, utilizzando il meccanismo visto a lezione. Non è richiesto alcun ulteriore meccanismo di protezione. La registrazione di un nuovo utente non è richiesta.

Requisiti del database

- Il database del progetto deve essere implementato a cura dello studente, e deve essere precaricato con *almeno 3 utenti*, che abbiano giocato almeno 5 volte ciascuno.

Contenuto del file README.md

Il file README.md deve contenere le seguenti informazioni (un template è disponibile nello scheletro del progetto creato con il repository). 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. Uno screenshot della **pagina per giocare al gioco, con alcune parole già inserite**. Lo screenshot deve essere inserito nel README linkando un'immagine committata nel repository
    b. Username e password degli utenti creati.

Procedura di consegna (importante!)

Per sottomettere correttamente il progetto è necessario:

- Essere **iscritti** all'appello.
- **Avere accettato l'invito** su GitHub Classroom, **associando** correttamente il proprio utente GitHub al proprio nome e matricola.
- Fare il **push del progetto** nel **branch "`main`"** del repository che GitHub Classroom ha generato per lo studente. L'ultimo commit (quello che si vuole venga valutato) deve essere **taggato** con il tag `final`.
  NB: recentemente GitHub *ha cambiato il nome del branch di default da master a main*, porre attenzione al nome del branch utilizzato, specialmente se si parte/riutilizza/modifica una soluzione precedentemente caricata su un sistema git.

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
```

NB: **il nome del tag è "final", tutto minuscolo, senza virgolette, senza spazi, senza altri caratteri**, e deve essere associato al commit da valutare.

E' anche possibile inserire un tag dall'interfaccia web di GitHub (seguire il link 'Create a new release').

Per testare la propria sottomissione, questi sono i comandi che useremo per scaricare 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 client ; npm install)
(cd server ; npm install)
```

Assicurarsi che tutti i pacchetti (package) necessari siano scaricati tramite i comandi `npm install`. Fare attenzione se alcuni pacchetti sono stati installati a livello globale perché potrebbero non apparire come dipendenze necessarie: potreste voler testare la procedura su un'installazione completamente nuova (per es. in una VM).

Il progetto sarà testato sotto Linux: si faccia 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 negli `import` e nei `require()`.