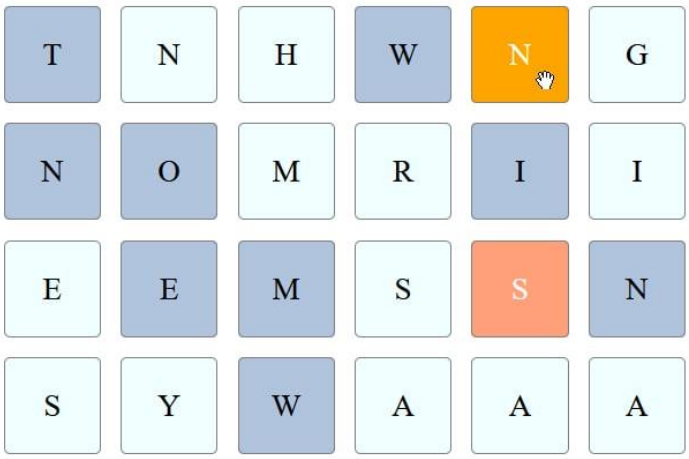
Applicazioni Web I – Esame # 5 (scadenza 2022-01-19 alle 23:59)

“Crucipuzzle”

VERSIONE FINALE – Le modifiche sono riportate in rosso

Progettare e implementare un’applicazione web per creare e risolvere un puzzle di ricerca di parole, conosciuto come “CruciPuzzle”. Il gioco consiste in una griglia rettangolare di lettere alfabetiche, in cui il giocatore deve scoprire delle parole presenti in una qualsiasi delle 8 direzioni possibili (2 orizzontali, 2 verticali, e 4 diagonali).



Esempio di griglia[1], con le parole TOM, WE, NEW e WIN già selezionate (ed evidenziate), e l’utente sta selezionando la parola SIN (la S è stata selezionata come punto di inizio, e la N sta per essere selezionata come punto di fine).

L’applicazione deve soddisfare le seguenti specifiche.

Un giocatore può iniziare una nuova partita selezionando la difficoltà tra 5 livelli differenti di difficoltà; ogni livello consiste di una griglia di dimensione differente (la difficoltà “1” è 4x6, la difficoltà “5” è 20x30). La griglia è riempita con lettere maiuscole, scelte a caso dal server, e la probabilità di scegliere una determinata lettera è proporzionale alla sua frequenza nella lingua inglese[2].

Le lettere devono essere mostrate graficamente all’utente sotto forma di una griglia rettangolare, così che sia possibile identificare facilmente righe, colonne e diagonali di lettere. Ogni partita ha una durata fissa pari a 60 secondi.

L’utente deve identificare una o più parole inglesi nella griglia, a partire da una qualsiasi lettera e andando in una qualsiasi delle 8 direzioni possibili. Per selezionare una parola, un utente deve cliccare sulla *prima* e *ultima* lettera della parola. Se le posizioni dei click sono compatibili (sulla stessa riga/colonna/diagonale),

[1] NON è richiesto di replicare esattamente questo design, è possibile sviluppare il proprio design (anche di aspetto migliore)

[2] E’ possibile trovare le frequenze pre-calcolate su Wikipedia https://en.wikipedia.org/wiki/Letter_frequency o altri siti web (per es., http://pi.math.cornell.edu/~mec/2003-2004/cryptography/subs/frequencies.html)

e se le lettere che collegano la posizione iniziale e finale compongono una parola valida in inglese (questo viene verificato tramite un dizionario[3] di parole valide), allora le lettere vengono "marcate" visivamente (per es. tramite uno sfondo colorato), e l'utente riceverà tanti punti quanti la lunghezza della parola. Nel selezionare una nuova parola, l'utente può riutilizzare alcune lettere delle precedenti parole.

Al termine di ogni partita di 60 secondi, il punteggio totale viene calcolato moltiplicando numero totale di lettere delle parole trovate per il livello di difficoltà (1…5), e viene mostrato all'utente. L'utente deve poter interrompere la partita in qualsiasi momento prima del termine dei 60 secondi. In questo caso, devono essere compiute le stesse azioni appena descritte.

Il gioco può essere giocato in modalità anonima (nessun login richiesto, e i punteggi non sono salvati) oppure in modalità personalizzata (l'utente deve effettuare il login prima di iniziare il gioco, e tutti i punteggi di tutte le partite sono immagazzinati nella storia personale).

L'applicazione deve anche mostrare la "hall of fame", ossia una pagina accessibile senza autenticazione con il nome e il punteggio dei migliori 5 giocatori.

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 "React Development Proxy" 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 server.js`" e "`cd client; npm start`". Viene fornito uno scheletro delle directory del progetto. Si può dare per scontato che `nodemon` sia già installato a livello di sistema operativo.
- 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 tutte.

[3] La lista di parole deve essere memorizzata sul server e NON deve essere trasferita al client. E' possibile trovare tale lista in svariati siti web, per es. http://www.mieliestronk.com/wordlist.html, https://github.com/dwyl/english-words, http://www.gwicks.net/dictionaries.htm, …

- L'autenticazione dell'utente (login) e l'accesso alle API devono essere realizzati tramite `passport.js` e cookie di sessione. Non è richiesto alcun ulteriore meccanismo di protezione. La registrazione di un nuovo utente non è richiesta.
- Il database del progetto deve essere incluso con la sottomissione, e deve essere precaricato con *almeno 3 utenti* che abbiano giocato almeno 5 volte.

Contenuto del file README.md

Il file README.md deve contenere le seguenti informazioni (un template è disponibile nello scheletro del progetto – in generale, ogni spiegazione non dovrebbe essere più lunga di 1-2 righe):

1. Una lista delle route dell'applicazione React, con una breve descrizione dello scopo di ogni route
2. Una lista delle API HTTP offerte dal server, con una breve descrizione dei parametri e degli oggetti scambiati
3. Una lista delle tabelle del database, con il loro scopo
4. Una lista dei principali componenti React usati
5. Uno screenshot della **pagina della partita in corso, con alcune parole già trovate** (embeddando una immagine messa nel repository)
6. Username e password degli utenti creati (vedere sopra).

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 alla propria 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
```

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 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 gli `import` e i `require()`.