Applicazioni Web I 2024/25 – Esame #3 (scadenza 2025-09-16 alle 23:59)

“Indovina la Frase”

VERSIONE FINALE – Le modifiche sono riportate in rosso.

Sviluppare un’applicazione web per giocare a un gioco di “indovina la frase”, le cui regole[1] sono dettagliate di seguito.

A ciascun giocatore viene assegnata una dotazione iniziale di 100 monete, che possono essere spese per giocare oppure guadagnate vincendo le partite.

Un giocatore autenticato (con un numero positivo di monete) può iniziare una nuova **partita**, nella quale deve indovinare una **frase segreta** (in inglese). La frase è scelta dal server, è di significato compiuto ed è composta da una sequenza di lettere dell’alfabeto e/o spazi, con una lunghezza compresa tra 30 e 50 caratteri. La frase non distingue tra maiuscole e minuscole.

Una volta iniziata la partita, l’applicazione mostra al giocatore una **griglia** contenente una cella vuota per ciascuna lettera dell’alfabeto che compone la frase. Gli spazi sono già posizionati nei punti corretti. Nel momento in cui la griglia viene mostrata, parte un timer che concede 60 secondi per indovinare la frase.

Fino allo scadere del tempo, il giocatore può eseguire una sequenza di azioni; ogni azione può essere una delle seguenti:

- **Indovinare una lettera**. Il giocatore sceglie una lettera; se questa compare almeno una volta nella frase, tutte le occorrenze di quella lettera vengono rivelate (le celle mostreranno la lettera al posto del vuoto). Indovinare una lettera costa al giocatore un numero di monete che dipende dalla frequenza della lettera nella lingua inglese[2]: 5 monete per le consonanti più frequenti fino a 1 moneta per quelle meno frequenti. Le vocali costano 10 monete. Il costo di ogni lettera deve essere mostrato al giocatore. Nel caso in cui la lettera scelta non sia presente nella frase, il suo costo viene raddoppiato (se le monete non fossero sufficienti a coprire il doppio del costo, si useranno tutte le monete del giocatore). Come vincolo, in una partita può essere scelta una sola vocale. Se il giocatore esaurisce le proprie monete, non può indovinare ulteriori lettere.
- **Indovinare l’intera frase**. Il giocatore scrive una frase completa (ad esempio in una casella di testo) e l’applicazione verifica se coincide esattamente con la frase segreta. Se è uguale, il giocatore vince 100 monete e la partita termina. Se non è uguale, il giocatore riceve un messaggio appropriato e la partita continua senza penalità.
- **Abbandonare la partita**. Il giocatore può interrompere la partita in corso senza alcuna penalità. In questo caso la partita termina.

Se il tempo scade prima che il giocatore indovini la frase corretta, la partita si chiude con una penalità di 20 monete (o di tutte le monete del giocatore, se fossero meno di 20).

In ogni caso, al termine di una partita, l’applicazione mostra la griglia con la frase corretta.

[1] Il gioco descritto è un mix tra la “Ruota della fortuna” e il gioco “L’impiccato”.

[2] Per esempio: https://www.sttmedia.com/characterfrequency-english

Commentato [1]: basta averne 1?

Commentato [2]: sì, perché basta per tentare di indovinare una consonante non frequente

Commentato [3]: Vogliamo evitare un signor "Giancarlo"? :D

Commentato [4]: Vogliamo evitare lettere accentate, apostrofi & Co. :)

Commentato [5]: È possibile che ad un utente venga riproposta una frase che ha già indovinato in una partita precedente? In particolare se inizio una nuova sessione, è possibile che mi venga riproposta una frase che ho indovinato in una partita che ho giocato in una sessione precedente?

Commentato [6]: È possibile ricevere una frase ottenuta in una sessione di gioco precedente. Il testo non specifica come le frasi vengono selezionate, solo

Commentato [7]: quindi:
ho 1 moneta posso giocare

Commentato [8]: O non ho capito o è errato. Se il giocatore iniziasse una partita con 1 moneta, come è

Commentato [9]: Ho un dubbio su come vengono attribuiti i costi alle consonanti, ovvero come associ a

Commentato [10]: Il come sta a voi, è una scelta di progettazione. Per il costo, quelle più frequenti costan

Commentato [11]: Se ho capito bene il costo in monete viene calcolato dopo che l'utente ha tentato di

Commentato [12]: Non andiamo in negativo. Se non ha monete sufficienti, non può giocare. Inoltre, il costo

Commentato [13]: quindi per permettere una "puntata" dobbiamo prevedere il caso peggiore in cui non si

Commentato [14]: Buona osservazione. Ho aggiornato il testo così potete calcolare il costo subito e, nel caso

Commentato [15]: Mi scusi nel momento in cui io inserisco una lettera devo controllare se è presente o

Commentato [16]: La prima. Per esempio: scelgo la lettera L. Se c'è nella frase da indovinare, appare e m

Commentato [17]: Se scelgo una vocale non presente nel testo, ne posso scegliere un'altra o il vincolo di una

Commentato [18]: Una sola. Indipendentemente dalla sua presenza nella frase.

Commentato [19]: Il giocatore può completare la frase solo indovinando lettera per lettera? vince sempre 100

Commentato [20]: Per vincere, può indovinare tutte le lettere o direttamente l’intera frase. Le lettere sono un

Commentato [21]: Quando si parla di interrompere la partita senza penalità, si deve gestire una sorta di

Commentato [22]: Nessun rollback. Si esce, senza perdere ulteriori monete ma senza neanche recuperar

Commentato [23]: Cosa succede se il giocatore raggiunge 0 monete e perde la partita? Rimarrà per

Commentato [24]: Le partite iniziano con le monete a disposizione del giocatore, non "sempre con 100" (dic

Un utente non autenticato può giocare a una versione semplificata del gioco, senza monete in gioco, e con tre frasi segrete dedicate disponibili.

Commentato [25]: Quindi come funziona in questo caso lo sblocco delle consonanti? C'é un limite? Per le vocali resta sempre il limite di una?

Commentato [26]: Non c'è un limite sulle consonanti. Tutto il resto (tranne per il fatto che ci siano 3 frasi dedicate, diverse dalle altre che riceve un giocatore loggato) rimane uguale: una sola vocale, timer di 30 secondi, regole del gioco, … Semplicemente non spende né guadagna monete.

Note:

- Per evitare imbrogli, il client dell'applicazione **non deve mai ricevere alcuna informazione** sulla frase segreta che non sia strettamente e immediatamente necessaria.
- Più utenti possono giocare contemporaneamente e ciascuno di essi gioca con una frase indipendente e con un timer indipendente.

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 (22.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 base.
- Originalità della soluzione.

Requisiti del database

- Il database del progetto deve essere realizzato dallo studente e deve essere precaricato con almeno 3 giocatori (uno che non ha giocato alcuna partita, uno che ha terminato le proprie monete, uno che ha giocato alcune partite). L'applicazione deve anche avere almeno 20 diverse frasi da indovinare (più le 3 per la modalità semplificata per utenti non autenticati).

Contenuto del file README.md

Il file README.md deve contenere le seguenti informazioni, seguendo il template disponibile nel repository del progetto. In genere, ogni spiegazione non dovrebbe essere più lunga di 2-3 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 dell'applicazione, **con una partita in corso**. L'immagine va embeddata 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 (e corso) 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`.