


“Gioco della Sfortuna”

 **VERSIONE PRELIMINARE** – Lasciare eventuali domande come COMMENTI e non cancellare o risolvere nessun commento esistente. La versione finale sarà pubblicata il 2025-06-06.

Progettare e implementare un'applicazione web per giocare a una versione single player del gioco da tavolo “Stuff Happens”.

Nel gioco, ogni giocatore deve giocare contro il computer (l'applicazione web, nel nostro caso) per ottenere un totale di 6 carte, ognuna rappresentante una situazione orribile, secondo le regole descritte di seguito.

Il gioco si basa su un archivio di almeno 50 **carte con situazioni orribili** (da “ti si sfilta il costume in una piscina pubblica” a “uno squalo ti attacca e perdi una gamba”). Ogni carta include il *nome della situazione*, un'*immagine* rappresentativa e un *indice di sfortuna* da 1 a 100 (dove 1 è “niente di grave” e 100 è “ma perché proprio a me?”). Ogni carta ha un indice di sfortuna univoco, con una differenza minima di 0.5 tra ogni indice. Ciascun studente deve scegliere, per il proprio progetto, un *campo della vita* come tema delle carte da utilizzare nel gioco (per esempio, vita universitaria, sport e fitness, vita sentimentale, viaggi e turismo, ecc.).

L'applicazione permette di giocare più partite. La partita inizia con un giocatore che riceve 3 carte casuali (generate dal server e immediatamente visibili al giocatore) e si svolge in più round. Ogni round propone una nuova situazione orribile, come segue:

1. Il giocatore vede nome e immagine (ma non l'indice di sfortuna!) della situazione orribile a lui/lei assegnata, diversa da quelle presenti nelle sue carte e anch'essa generata casualmente dal server. Il giocatore, potendo vedere tutte le carte in suo possesso con tutti i relativi dettagli, deve indicare dove la situazione ricevuta si colloca, come indice di sfortuna, tra le carte in suo possesso. Per esempio, se il giocatore avesse carte con indice 1.5, 42.5 e 99, potrebbe ipotizzare che la situazione assegnata abbia un indice di sfortuna tra 42.5 e 99 e quindi collocarla tra quelle due carte.
2. Se il giocatore indovina la collocazione corretta entro 30 secondi, ottiene la carta di quella situazione, che viene quindi aggiunta all'insieme di carte in suo possesso e ne vengono mostrate tutte le informazioni.
3. Se il giocatore, invece, non indovina la collocazione corretta entro 30 secondi o il tempo scade senza che il giocatore abbia fatto una scelta, non riceve la carta (e non vede nessuna informazione su quella carta). Tale carta non dovrà essere presentata nuovamente nei round successivi della stessa partita.
4. Sia in caso di vincita o perdita di un round, l'applicazione mostra un messaggio appropriato e chiede al giocatore di confermare quando è pronto a iniziare un nuovo round.
5. La partita termina quando il giocatore ha 6 carte in suo possesso (vince) oppure non ha indovinato la collocazione corretta di tre situazioni orribili (perde).

Gli **utenti registrati** (cioè, dopo aver fatto login) giocheranno una partita finché quest'ultima non sarà completata (vinta o persa). Inoltre, tutte le loro partite completate saranno riportate in una cronologia visibile nella pagina del profilo utente. La cronologia deve mostrare, per ogni partita, i

nomi delle situazioni orribili delle carte coinvolte nella partita, con l'indicazione se sono state vinte o meno e in quale round, e l'esito della partita stessa con il numero totale di carte raccolte.

Gli **utenti anonimi** (cioè, i visitatori del sito), invece, possono visualizzare le istruzioni di gioco e giocare solo partite demo della durata di un solo round (una sola carta da giocare, partendo dalle 3 carte iniziali). Non avranno accesso a nessuna funzionalità degli utenti registrati.

Al termine della partita viene visualizzato un riepilogo che mostra le carte vinte (nome della situazione orribile, immagine e suo indice di sfortuna) e il giocatore può scegliere di cominciare una nuova partita.

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 "autoricararsi" 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 50 carte di situazioni orribili che possono capitare nel campo di vita scelto e almeno 2 utenti registrati. Uno degli utenti registrati deve aver giocato alcuni giochi con successo.

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 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. Due screenshot dell'applicazione, **uno con la cronologia dell'utente e uno durante una partita**. 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.