Applicazioni Web I – Esame # 2 (scadenza 2024-07-14 alle 23:59)

“Budget Sociale”

VERSIONE FINALE – Le modifiche sono riportate in rosso.

Un’associazione senza scopo di lucro deve definire le attività per il prossimo anno e desidera allocare il budget disponibile a una serie di iniziative proposte e selezionate dai membri dell’associazione tramite un processo partecipativo.

Per fare ciò, si realizzi un’applicazione web che consenta all’associazione di definire il budget attraverso un processo in più fasi:

- Fase 0: un utente speciale registrato del sistema (‘amministratore’) definisce il **budget** per il prossimo anno (in Euro). Nessun’altra azione è possibile in questa fase. Non appena il budget viene definito, il sistema passa alla Fase 1. Gli altri utenti vedranno un messaggio che informa che la fase di definizione delle proposte è ancora chiusa (vedi dopo).
- Fase 1: ogni membro dell’associazione, dopo aver effettuato l’accesso all’applicazione, può inserire una **proposta** (massimo 3 proposte per ogni membro). Ogni proposta consiste in una descrizione di una riga e un costo stimato (in Euro), che non può essere superiore al budget definito nella fase precedente. In questa fase, ogni membro vede l’importo del budget definito, può visualizzare, modificare, eliminare e aggiungere le proprie proposte, ma non può vedere alcuna proposta degli altri membri. L’amministratore, oltre alle funzioni disponibili per ogni altro membro, può terminare il processo di proposta e passare alla Fase 2.
- Fase 2: le proposte non possono essere modificate e i loro dettagli sono visibili a tutti i membri. Ogni membro può esprimere una **preferenza** per qualsiasi proposta pubblicata tranne la propria; la preferenza è espressa come un punteggio da 1 a 3 (un valore di 0 è assunto per default). Non c’è limite al numero di preferenze espresse. Ogni membro può vedere solo le proprie preferenze e può ~~modificarle o~~ revocarle in qualsiasi momento, ma non può vedere le preferenze degli altri membri. L’amministratore, oltre alle funzioni disponibili per ogni altro membro, può terminare il processo di votazione e passare alla Fase 3.
- Fase 3: non è possibile apportare ulteriori modifiche alle proposte né alle preferenze, e l'insieme finale delle **proposte approvate** è visibile a tutti. Le proposte approvate sono calcolate ordinandole per il punteggio totale di preferenze e prendendo quelle la cui somma cumulativa è minore o uguale al budget definito. L’applicazione mostrerà l’elenco di tutte le proposte, classificate in base al numero totale dei punteggi delle preferenze, fino a coprire il budget definito; questo elenco includerà la descrizione della proposta, il membro che l’ha proposta, il suo costo e il suo punteggio totale. Le proposte non approvate (cioè quelle al di fuori del budget complessivo) sono anch’esse elencate, ordinate per punteggio, ma senza mostrare l’autore. L'amministratore può decidere di riavviare il processo tornando alla Fase 0; in questo caso, si effettua un reset completo, con tutte le proposte e le loro preferenze che vengono eliminate.

Gli utenti anonimi possono vedere l’elenco delle proposte approvate nella Fase 3 ma non vedranno quelle non approvate. Nelle Fasi 0-2, gli utenti anonimi vedranno solo un messaggio che informa che la fase di definizione delle proposte è in corso.

L'applicazione gestirà una sola associazione. Ci deve essere un solo utente che è anche amministratore. Tutti i membri dell'associazione sono utenti registrati nell'applicazione.

L'applicazione dovrebbe prevenire comportamenti malevoli (accesso a informazioni private, votazione al di fuori della fase 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 4 utenti registrati, incluso l'amministratore. Lo stato iniziale dell'applicazione deve essere prima della Fase 0.

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 durante la Fase 1 e la Fase 2, rispettivamente**. 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 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`.