Applicazioni Web I – Esame # 3 (scadenza 2021-09-07 alle 23:59)

"Gruppi studio"

VERSIONE FINALE – Le modifiche sono riportate in rosso

Progettare e implementare un'applicazione web per dei *gruppi di studio* per studenti universitari.

L'applicazione deve soddisfare le seguenti specifiche.

Un *gruppo di studio* (SG) è un insieme di studenti che decide di studiare insieme per preparare un certo esame universitario. Perciò, un SG è caratterizzato dall'informazione riguardo al corso (codice, nome, crediti)[1], da una lista di studenti che partecipano al gruppo, e da un calendario di incontri (passato e futuri) del gruppo. Ogni corso può avere al massimo un gruppo di studio ~~attivo~~, e per chiarezza esso è associato a un colore unico; questo colore dovrà essere usato ogni volta che l'applicazione mostrerà informazioni associate a quel SG.

Tutti gli utenti del sistema hanno il ruolo 'studente', ogni utente può avere ruoli extra ~~(ogni ruolo ha anche accesso alle funzionalità dei ruoli inferiori)~~. I ruoli sono:

- Amministratore generale. Nota: può esserci più di un amministratore generale.
- Amministratore di gruppo (per uno o più gruppi di studio). Nota: ogni SG può avere più di un amministratore di gruppo.
- Studente (tutti gli utenti hanno questo ruolo, quindi tutti gli utenti hanno accesso alle sue funzionalità).

Un utente con il ruolo di amministratore generale può:

- Vedere la lista di tutti i gruppi di studio
- Vedere le informazioni di un gruppo di studio (SG): lista dei membri, lista degli amministratori del gruppo, lista degli incontri passati e futuri.
- Creare un nuovo SG, inserendo i dati relativi ad un corso
- Cancellare un SG
- Aggiungere o rimuovere amministratori del gruppo, per ogni SG, selezionandoli dalla lista dei membri correnti dell'SG

Un utente con il ruolo di amministratore di gruppo può:

- Vedere la lista degli SG che può amministrare, e i suoi membri (questa è la stessa funzionalità dei primi 2 punti elencati sopra per l'amministratore generale, ma limitata agli SG amministrati dallo studente)
- Definire gli incontri futuri (data, ora, durata, luogo) per il gruppo di studio
- Vedere la lista degli studenti dello SG
- Approvare le richieste di studenti di partecipare allo SG
- Rimuovere uno studente da un SG

[1] Non è necessario avere alcuna informazione pre-inserita nell'applicazione riguardo ai corsi.

Un utente con il ruolo di studente può:

- Vedere la lista di tutti gli SG
- Chiedere di partecipare ad uno SG (la richiesta deve essere approvata da un amministratore di gruppo)
- Vedere la lista degli incontri futuri per tutti gli SG a cui partecipa; tutti gli incontri devono essere mostrati nella stessa lista/tabella/calendario, usando i colori corretti per identificare ogni SG
- Registrarsi per un incontro futuro per uno SG a cui partecipa, o cancellare la registrazione per un incontro a cui si è precedentemente registrato. Nel caso in cui l'utente cerchi di registrarsi per un incontro che si sovrappone ad un altro a cui si è già registrato, il sistema richiederà una conferma ulteriore prima di precedere.

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.
- 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 5 utenti* e *2 gruppi di studio*, con la seguente mappatura (il nome degli utenti e dei gruppi può essere scelto liberamente).
    - Utente 1: Amministratore generale, studente
    - Utente 2: Amministratore generale, amministratore di gruppo per SG A, studente
    - Utente 3: Amministratore di gruppo per SG A, amministratore di gruppo per SG B, studente

- Utente 4: Amministratore di gruppo per SG B, studente
- Utente 5: Studente

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 con la lista di tutti gli incontri futuri** (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()`.