Applicazioni Web I – Esame # 3 (scadenza 2020-09-09 alle 23:59)

"Scheduling oral tests"

VERSIONE FINALE – Aggiunte e correzioni sono mostrate in rosso

Progettare e implementare un'applicazione web per consentire a degli insegnanti e degli studenti di definire in maniera semplice uno schedule per dei test orali, in cui un gran numero di studenti deve essere allocato in slot temporali individuali su un certo numero di giorni.

Ci sono due tipi di utenti nel sistema: insegnanti e studenti. Gli insegnanti devono sempre essere autenticati nel sistema (tutte le operazioni disponibili per gli insegnanti devono essere autenticate). Le operazioni effettuate dagli studenti non necessitano di essere autenticate (sono sempre disponibili).

Si assuma che ogni insegnante sia associato con esattamente un solo corso, per il quale la lista degli studenti è immagazzinata nel database. Dopo il login, l'insegnante ha accesso alle seguenti funzionalità:

- **Creare un esame e uno schedule**. Questa azione richiede 4 differenti fasi:
    1. L'insegnante seleziona un sottoinsieme degli studenti del corso (può selezionare solamente studenti che non hanno ancora passato l'esame): gli studenti selezionati dovranno sostenere il test orale.
    2. Successivamente, l'insegnante definisce T1, cioè, la durata di uno "slot" che corrisponde ad un singolo test orale (per es. 15 minuti).
    3. L'insegnante definisce uno o più "sessioni" (definite da: giorno, ora di inizio, durata – con durata multiplo di T1). Mentre le sessioni vengono definite, il sistema mostra (in tempo reale) il numero di studenti selezionato al passo 1, e il numero di slot disponibili (che può essere una differenza positiva o negativa).
    4. Se la differenza è positiva o zero, l'insegnante può salvare e chiudere lo schedule.
- **Eseguire un test orale.** L'insegnante seleziona uno degli slot (per facilità di debug, si assuma che corrisponda con la data e ora corrente, senza controllarla) in cui è schedulato uno studente. L'insegnante specificherà se lo studente è Presente o Assente. Se è presente, allora l'insegnante specificherà un voto: 18-30, 30L, Insufficiente, Ritirato. Gli studenti che hanno ottenuto un voto positivo (da 18 a 30L) non saranno in grado di prenotare altri slot nello stesso esame.
- **Vedere un riassunto dei risultati**. L'insegnante vede la lista di tutti gli studenti selezionati. Per ogni studente viene mostrato o la data e l'ora dello slot a cui lo studente è prenotato (se lo studente è prenotato ma non è ancora passato) oppure il voto assegnato (per gli orali già svolti), o il fatto che manca per gli studenti che non hanno ancora prenotato.

Le funzionalità mostrate agli studenti non richiedono autenticazione, esse necessitano solamente che lo studente inserisca il suo numero di matricola (senza password). Gli studenti possono:

- **Prenotare uno slot disponibile** (non ancora preso da altri studenti). Per vedere la lista degli slot, lo studente deve scegliere uno degli esami possibili.
- **Vedere il proprio slot correntemente prenotato**. Se l'orale è già stato sostenuto, verrà mostrato il risultato. Se l'orale non è ancora stato sostenuto, verranno mostrate la data e l'ora, con la possibilità di **DISDIRE** l'appuntamento.

Requisiti del progetto

- L'architettura dell'applicazione e il codice sorgente devono essere sviluppati adottando le migliori pratiche (best practices) di sviluppo del software, in particolare quelle per le single page applications (SPA) che usano React e REST.
- Il progetto deve essere realizzato come applicazione React, che interagisce con un'API REST 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`" and "`cd client; npm start`". Viene fornito uno scheletro delle directory del progetto.
- L'intero progetto deve essere consegnato tramite GitHub, nel repository creato da GitHub Classroom.
- Il progetto non deve includere le directories in `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 `moment.js`, `react-bootstrap`, ecc.), se applicabili e utili. Tali librerie devono essere correttamente dichiarate nel file `package.json` cosicché il comando `npm install` le possa scaricare.
- L'autenticazione dell'utente e l'accesso devono essere realizzati tramite token JWT, memorizzato in un cookie http-only. Non è richiesto alcun ulteriore meccanismo di protezione.
- E' possibile assumere che ci sia un solo studente alla volta che usa il sistema, perciò non è necessario considerare operazioni concorrenti da parte di più utenti.
- Il database del progetto deve essere definito dallo studente e deve essere precaricato con almeno 2 insegnanti/corsi di esempio (la procedura di registrazione non è richiesta), almeno 10 studenti per ogni corso (parzialmente sovrappposti), almeno 3 esami già definiti, e alcuni studenti già prenotati.

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 REST offerte dal server, con una breve descrizione dei parametri e delle entità scambiate
3. Una lista delle tabelle del database, con il loro scopo
4. Una lista dei principali componenti React usati nell'applicazione
5. Uno screenshot della **pagina per definire un nuovo slot** (embeddando una immagine messa nel repository)
6. Username e password dei 2 insegnanti di test, e il numero di matricola di 3-4 studenti per ogni corso.

Procedura di sottomissione (importante!)

Per sottomettere correttamente il progetto è necessario:

- essere **iscritti** all'appello
- fare il **push del progetto** nel branch `master` 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`

Nota: per taggare un commit, si può 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 alternative, 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 master  # just in case the default branch is not master
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` .

Il progetto sarà testato sotto Linux (si faccia attenzione al fatto che Linux è case-sensitive nei nomi dei files, mentre MacOS-X e Windows non lo sono).