Applicazioni Web I – Esame # 1 (scadenza 2021-06-28 alle 23:59)

"Questionario"

VERSIONE FINALE – Aggiunte e correzioni sono riportate in rosso

Progettare e implementare un'applicazione web per gestire la creazione di un questionario online e la raccolta di informazioni tramite tale questionario.

L'applicazione deve soddisfare le seguenti specifiche.

Un questionario è composto da un **titolo** e da un insieme non vuoto di **domande**. Le domande possono essere di due tipi diversi:

- Domanda a **risposta chiusa**. È caratterizzata dal testo di un **quesito** e da una serie di testi per le possibili **risposte**. Alla domanda sono associati il numero *minimo* ed il numero *massimo* di risposte[1] che saranno accettate in fase di compilazione. Ogni domanda può avere al massimo 10 possibili risposte.
- Domanda a **risposta aperta**. È caratterizzata dal testo di un quesito e da un'area di **testo libero** per la risposta (max 200 caratteri). La domanda può essere indicata come **obbligatoria** o **opzionale**.

Vi sono due tipi di utenti: **amministratore** (che può creare questionari e visualizzare i risultati) ed **utilizzatore** (che può compilare un questionario).

L'**amministratore** deve autenticarsi con username/password. Una volta autenticato, l'amministratore può:

- Creare un nuovo questionario, definendo il titolo e le domande. In questa fase, le azioni possibili sono:
  - Creare una nuova domanda (raccogliendo tutte le informazioni ed opzioni relative alla domanda stessa).
  - Modificare l'ordine delle domande inserite (con azioni su/giù disponibili per ciascuna domanda).
  - Eliminare una domanda.
  - Pubblicare il questionario. Da questo momento, il questionario non sarà più modificabile, e diventerà visibile agli utilizzatori nella pagina principale del sito.
- Visualizzare i risultati dei propri questionari pubblicati. Le azioni possibili sono:
  - Visualizzare l'elenco dei propri questionari pubblicati, affiancati dal *numero* di compilazioni ricevute.

[1] Agendo su questi valori, si possono ottenere diversi tipi di domande:
- min = 0, max = 1 → domanda facoltativa, a scelta singola
- min = 1, max = 1 → domanda obbligatoria, a scelta singola
- min = 0, max > 1 → domanda facoltativa, a scelta multipla
- min = 1, max > 1 → domanda obbligatoria, a scelta multipla

- Selezionando uno di tali questionari, permettere di navigare tra le risposte fornite dagli utilizzatori. Nella pagina sarà mostrato il nome dell'utilizzatore, seguito da tutte le risposte fornite[2], e ci saranno funzionalità di navigazione (precedente/successivo) per potersi spostare sulle altre compilazioni ricevute.

L'**utilizzatore** non deve autenticarsi al sito. Dalla pagina principale, può scegliere uno dei questionari pubblicati, ed iniziarne la compilazione. All'avvio della compilazione, l'utilizzatore deve inserire il proprio *nome* (campo di testo libero), e procedere alla compilazione delle risposte. Le domande dovranno chiaramente indicare i vincoli (min/max/obbligatorio) di compilazione. Il questionario potrà essere inviato solo se tutti i vincoli sono stati rispettati. Una volta inviato il questionario, esso diventa non più modificabile, e l'utilizzatore ritorna alla pagina principale.

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 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 nel file `package.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 definito dallo studente e deve essere precaricato con *almeno* 2 utenti amministratori, i quali abbiano creato almeno 1 questionario ciascuno, e *almeno* quattro compilazioni (almeno due per lo stesso questionario).

[2] **Suggerimento**: a tale scopo, è possibile riutilizzare i componenti usati per la creazione del questionario configurandoli in modalità di sola lettura.

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 per creare un questionario, in cui siano inserite 2 domande, una per tipo** (embeddando una immagine messa nel repository)
6. Username e password degli utenti amministratori, indicando anche quali amministratori hanno creato quali questionari.

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()`.