Applicazioni Web I – Esame #4 (scadenza 2022-11-03 alle 23:59)

“Form griglia”

VERSIONE FINALE – non ci sono state modifiche

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

L’applicazione deve soddisfare le seguenti specifiche.

Un form è composto da un **titolo** e da una **domanda** nella forma di una griglia.

La domanda a griglia si compone dalle intestazioni di riga e di colonna, il cui numero e contenuto è scelto, da chi crea la domanda stessa, e da una griglia di caselle selezionabili o no (es. checkbox). Esempio di domanda: attività sportive sulle righe, nomi di persone sulle colonne, la casella selezionata indica l’interesse per una certa attività).

Per la domanda a griglia, chi crea la domanda imposta, per ogni riga, un numero minimo e massimo di caselle che potranno essere selezionate, da 0 a M, dove M è il numero di colonne. Affinché sia possibile inviare la risposta è necessario soddisfare i vincoli per ogni riga di ogni griglia.

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

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

- Creare un nuovo form, definendo il titolo e la domanda. In questa fase, le azioni possibili sono, dopo l’indicazione del titolo:
    - Creare la domanda (specificando le informazioni sulle intestazioni di riga, sulle intestazioni di colonna, e sul valore minimo e massimo di caselle).
    - Pubblicare il form. Da questo momento, il form non sarà più modificabile, e diventerà visibile agli utilizzatori nella pagina principale del sito.
- Visualizzare i risultati dei propri form pubblicati. Le azioni possibili sono:
    - Visualizzare l’elenco dei propri form pubblicati, affiancati dal *numero* di compilazioni ricevute.
    - Selezionando uno di tali form, permettere di visualizzare le risposte fornite dagli utilizzatori, tutte sulla stessa pagina.

L’**utilizzatore** non deve autenticarsi al sito. Dalla pagina principale, può scegliere uno dei form pubblicati, ed iniziarne la compilazione. All’avvio della compilazione, l’utilizzatore deve inserire il proprio *nome* (campo di testo libero, di cui non si garantisce l’univocità), e procedere alla compilazione delle risposte. La domanda visualizzata dovrà chiaramente indicare i vincoli di compilazione (minimo e massimo di ogni riga) e garantirne il rispetto. Il form potrà essere inviato solo se tutti i vincoli sono stati rispettati.

L'organizzazione delle funzionalità di questo testo in differenti schermate (e potenzialmente in differenti routes) è lasciata allo studente ed è oggetto di valutazione.

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 "two servers", configurando correttamente CORS, 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 index.js`" e "`cd client; npm start`". Viene fornito uno scheletro delle directory del progetto. Si può assumere che `nodemon` sia già installato a livello di sistema.
- 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 ed installare tutte.
- L'autenticazione dell'utente (login) e l'accesso alle API devono essere realizzati tramite `passport.js` e cookie di sessione, utilizzando il meccanismo visto a lezione. Non è richiesto alcun ulteriore meccanismo di protezione. La registrazione di un nuovo utente non è richiesta.

Requisiti del database

- Il database del progetto deve essere definito dallo studente e deve essere precaricato con *almeno* 2 utenti amministratori, i quali abbiano creato almeno 1 form ciascuno, e *almeno* quattro compilazioni (almeno due per lo stesso form).

Contenuto del file README.md

Il file README.md deve contenere le seguenti informazioni (un template è disponibile nello scheletro del progetto creato con il repository). 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. Uno screenshot della **pagina per creare un form in cui sia già stata inserita la domanda a griglia** (embeddando una immagine messa nel repository)
    b. Username e password degli utenti amministratori, indicando anche quali amministratori hanno creato quali form.

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 al proprio nome e 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
```

NB: **il nome del tag è "final", tutto minuscolo, senza virgolette, senza spazi, senza altri caratteri**, e deve essere associato al commit da valutare.

E' anche 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 le maiuscole/minuscole usate negli `import` e nei `require()`.