Applicazioni Web I – Esame # 1 (scadenza 2022-06-22 alle 23:59)

“Piano degli studi”

VERSIONE FINALE – le modifiche sono in rosso

Progettare e implementare un’applicazione web per gestire il piano di studi di uno studente universitario.

L’applicazione deve soddisfare le seguenti specifiche.

L’università offre una serie di corsi. Ogni corso è caratterizzato da un codice univoco di 7 caratteri, un nome, e un numero (intero) di crediti.

Il piano di studi di uno studente è un sottoinsieme dei corsi offerti dall’università. Il totale dei crediti dei corsi inseriti nel piano di studi deve essere compreso tra 60 e 80 crediti (estremi inclusi) per l’opzione full-time, o tra 20 e 40 (estremi inclusi) per l’opzione part-time.

Un corso può essere soggetto ad uno o più vincoli riguardo al suo inserimento nel piano degli studi:

- Un corso può essere *incompatibile* con uno o più altri corsi. Essi non possono essere selezionati insieme.
- Un corso può avere un corso (uno solo) *propedeutico* obbligatorio, che deve già essere presente nel piano di studi.
- Un corso può avere un numero massimo di *studenti* che possono inserirlo nel piano di studi.

Nella pagina iniziale (home-page) dell’applicazione tutti gli utenti, anche non autenticati (cioè anonimi), possono vedere la lista completa dei corsi offerti dall’università. La lista dei corsi deve essere visualizzata in ordine alfabetico di nome corso. Per ogni corso, la lista mostra la sua descrizione: il codice, il nome, il numero di crediti, il numero di studenti che hanno già scelto il corso e, se presente, il numero massimo di studenti che possono selezionarlo. Ogni descrizione di corso può essere espansa/richiusa dall’utente, per mostrare il dettaglio dei corsi incompatibili e/o propedeutici (visualizzare almeno il codice corso). Più di un corso può essere visualizzato in stato espanso allo stesso tempo.

Una volta autenticato, l’utente continua a vedere la stessa lista completa dei corsi (in una home-page logged-in). In questa stessa pagina, se il piano di studi non è stato ancora creato, l’utente può crearne uno vuoto, specificando l’opzione full-time o part-time; questa lista vuota può essere editata secondo le istruzioni nel seguito. Se un piano di studi è già stato creato e salvato in modo persistente, esso dovrà essere immediatamente mostrato (nella stessa pagina) e può essere editato come nel seguito.

L’editing del piano di studi avviene sempre nella stessa pagina (home-page logged-in) dove sono visualizzate contemporaneamente sia la lista dei corsi sia il piano di studi. L’editing

- Mostra sempre il numero di crediti che corrispondono al totale dei corsi nel piano di studi, e i valori di minimo e massimo da rispettare.
- Consente di aggiungere un corso al piano di studi, preso dalla lista completa. Solo i corsi che soddisfano tutti i vincoli possono essere aggiunti.
- Consente di rimuovere un corso dal piano di studi, se ciò non viola alcun vincolo di propedeuticità (altrimenti, l’applicazione deve mostrare la ragione)

- Se un corso non può essere aggiunto, deve essere marcato in maniera differente nella lista completa dei corsi, e l'applicazione deve indicare la ragione.

Durante la sessione di editing, l'utente può "Salvare" il piano di studi in modo persistente (questo rimpiazzerà qualsiasi precedente versione). L'utente può "Annullare" le modifiche correnti e in tal caso la copia persistente (se presente) non deve essere modificata.

Quando si salva, il piano di studi deve essere validato per verificare il rispetto di tutti i vincoli.

In aggiunta, l'utente può "Cancellare" l'intero piano di studi, inclusa la copia persistente.

Dopo ognuna di queste azioni, l'applicazione dovrà trovarsi nella home-page logged-in.

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 implementato a cura dello studente, e deve essere precaricato con *almeno 5* studenti, di cui almeno 1 con un piano di studi part-time e almeno 1 con un piano di studi full-time. Almeno due corsi devono aver raggiunto il massimo numero di studenti iscritti.
- Il database del progetto deve includere, almeno, i seguenti corsi:

Codice	Nome	Crediti	Max studenti	Incompatibilità	Propedeuticità
02GOLOV	Architetture dei sistemi di elaborazione	12		02LSEOV	
02LSEOV	Computer architectures	12		02GOLOV	
01SQJOV	Data Science and Database Technology	8		01SQMOV 01SQLOV	
01SQMOV	Data Science e Tecnologie per le Basi di Dati	8		01SQJOV 01SQLOV	
01SQLOV	Database systems	8		01SQJOV 01SQMOV	
01OTWOV	Computer network technologies and services	6	3	02KPNOV	
02KPNOV	Tecnologie e servizi di rete	6	3	01OTWOV	
01TYMOV	Information systems security services	12		01UDUOV	
01UDUOV	Sicurezza dei sistemi informativi	12		01TYMOV	
05BIDOV	Ingegneria del software	6		04GSPOV	02GOLOV
04GSPOV	Software engineering	6		05BIDOV	02LSEOV
01UDFOV	Applicazioni Web I	6		01TXYOV	
01TXYOV	Web Applications I	6	3	01UDFOV	
01TXSOV	Web Applications II	6			01TXYOV
02GRSOV	Programmazione di sistema	6		01NYHOV	
01NYHOV	System and device programming	6	3	02GRSOV	
01SQOOV	Reti Locali e Data Center	6			
01TYDOV	Software networking	7			
03UEWOV	Challenge	5			
01URROV	Computational intelligence	6			
01OUZPD	Model based software design	4			
01URSPD	Internet Video Streaming	6	2		

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 home-page logged-in mentre è in corso una sessione di editing. Lo screenshot deve essere inserito nel README linkando un'immagine committata nel repository
    b. Username e password degli utenti creati, e loro caratteristiche (part-time, full-time)

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/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()`.