Progetto di laboratorio – IUM+TWEB – 2020/21 (12 CFU)

Si sviluppi un'applicazione java web che gestisca **prenotazioni di ripetizioni online** essendo *dotata sia di interfaccia utente per browser web che per accesso mobile.*

- Si assuma che per ogni corso di cui si offrono ripetizioni ci siano uno più docenti alternativi tra cui scegliere con chi fare la ripetizione (per es., lunedi dalle 9.00 alle 10.00 la lezione di matematica può essere tenuta sia da Mario Rossi che da Gianni Verdi).
- Si assuma che ogni docente possa insegnare più di un corso (per es., Mario Rossi insegna sia matematica che scienze).
- Si assuma che l'applicazione presenti le ripetizioni disponibili di una settimana pre-fissata, dal lunedi al venerdi, al pomeriggio (15.00 – 19.00). In altre parole, non è richiesto di gestire un vero calendario ma solo una griglia di slot temporali di dimensioni limitate (1h ciascuno). Ogni ripetizione disponibile è associata a un orario (dalle ore X alle ore Y del giorno Z) e ai docenti che la potrebbero tenere in quell'orario in quanto non occupati.

Seguono i dettagli richiesti per il progetto.

L'applicazione deve gestire due tipi di ruolo utente: utente registrato (cliente del servizio di ripetizioni) e amministratore. Si assuma di avere 1 amministratore e 2 clienti pre-registrati.

Nel dettaglio, l'applicazione deve permettere all'utente di eseguire le seguenti azioni:

- Visualizzare il catalogo delle ripetizioni disponibili, specificando i docenti disponibili per ogni ripetizione prenotabile - *tutti i ruoli, interfaccia web e mobile.*
- Prenotare una o più ripetizioni, tra quelle disponibili, scegliendo il corso e il docente desiderato - *tutti i ruoli, interfaccia web e mobile.*
- Disdire una prenotazione specifica - *tutti i ruoli, interfaccia web e mobile.*
- Segnare una ripetizione come effettuata – *ruolo cliente, interfaccia web e mobile.*
- Visualizzare la propria lista delle prenotazioni. La lista deve includere sia le ripetizioni ancora da fruire che lo storico delle ripetizioni prenotate in passato (NB: se si rimuove un docente, o un corso, lo storico deve rimanere). Per ogni ripetizione in elenco, l'applicazione deve visualizzare lo stato della ripetizione (attiva/effettuata/disdetta) - *tutti i ruoli, interfaccia web e mobile.*
- Visualizzare tutte le prenotazioni attive, effettuate e cancellate dei vari clienti - *solo amministratore, solo interfaccia web.*
- Inserire/rimuovere corsi e docenti, e associazioni corso-docente ai fini delle ripetizioni - *solo amministratore, solo interfaccia web.*

NB: assumete che, quando parte l'applicazione, tutti i docenti siano disponibili in tutti gli slot temporali della griglia. Le loro disponibilità verranno cancellate man mano che i clienti prenoteranno le loro lezioni. Non sviluppate nessuna interfaccia utente per i docenti, essi non devono eseguire nessuna operazione sull'applicazione.

NB: la scelta del dominio applicativo (prenotazioni) non è vincolante e sono ammessi altri tipi di dominio purché si rispettino le specifiche funzionali e tecniche descritte in questo documento.

Requisiti tecnici:

- L'applicazione deve essere basata su **architettura MVC**, con Controller + viste e Model. Si noti che non deve esserci comunicazione diretta tra viste e model: ogni tipo di comunicazione tra questi due livelli deve essere mediato dal controller.
- È obbligatorio gestire le **sessioni utente**.
- L'applicazione deve salvare in un **database relazionale a scelta** i seguenti tipi di informazione:
    - account, password e ruolo degli utenti registrati;
    - titolo dei corsi di cui si offrono le ripetizioni;
    - nome e cognome dei docenti che tengono le ripetizioni;
    - associazioni corso-docente;
    - prenotazioni di ripetizioni.
- L'applicazione deve controllare l'inserimento di input utente sia lato client che lato server per evitare che l'utente inserisca dati parziali o errati nei form (per esempio, per evitare che l'utente cerchi di collegarsi senza inserire login e password).
- L'applicazione deve controllare sia lato client che lato server che gli utenti non eseguano operazioni illecite. Per es., gli utenti non autenticati possono vedere il catalogo delle ripetizioni disponibili, ma non possono segnare come effettuate, o disdire, le prenotazioni; solo gli amministratori devono poter inserire/rimuovere corsi in catalogo; ogni utente (tranne l'amministratore) deve poter vedere solo le proprie ripetizioni e non quelle altrui.

Requisiti generali dell'interfaccia utente (sia web che mobile):

- L'interfaccia utente deve essere:
    - Comprensibile (**trasparenza**). Per esempio, a fronte di errori, deve segnalare il problema; quando un'operazione viene eseguita con successo, deve visualizzare la conferma di esecuzione, a meno che la conferma non sia ridondante (in quanto il risultato si vede direttamente sull'interfaccia utente).
    - Ragionevolmente efficiente per permettere all'utente di eseguire le operazioni con un numero minimo di click e di inserimenti di dati.
    - In caso di errore durante l'inserimento di dati nelle form, l'interfaccia deve permettere all'utente di correggere i dati e ripetere l'operazione senza perdere i dati precedentemente inseriti (cioè, senza riempire d'accapo i moduli online).

Requisiti specifici per l'interfaccia utente per il web:

- **L'interfaccia web deve essere implementata come Single Page Application utilizzando Vue.js** e **HTML5;** Il layout delle pagine dell'interfaccia utente deve essere specificato con **CSS** e deve essere **fluido**.
- Il controllo dell'input utente lato client deve essere effettuato utilizzando **i tag di HTML5 e/o JavaScript.**
- Il backend dell'applicazione deve essere implementato in Java.

- **Requisiti specifici per l'interfaccia utente mobile:**
    - vincoli implementativi: l'applicazione deve essere scritta in Java/Kotlin usando le librerie Android. La comunicazione col lato Server avviene tramite il formato JSON.

- **NB:** l'App deve essere sviluppata usando i componenti **native della UI di Android. NON** è permesso usare **Browser Web** o un loro emulatore come ad esempio **"WebView".** Nel dubbio chiedete prima.
- L'interfaccia Mobile comunica col server del progetto TWEB, 1 server condiviso tra 2 client.

Progetto per il linguaggio python:

Si può realizzare a scelta uno dei seguenti 2 progetti:

Progetto 1:

- traendo spunto dal TicTacToe, pensiamo ad un gioco simile generalizzato: "filetto"
- Data una matrice di dimensione NxN, i due giocatori cercano di costruire sequenze di caselle tutte uguali disposte in righe/colonne/diagonali il più lunghe possibile con i seguenti punteggi :
  - 0 punti per sequenze di lunghezza 1 o 2,
  - 2 punti per l=3,
  - 10 per l= 4,
  - 50 per L=5.
- Vince chi per primo totalizza 50 punti
- *il programma deve visualizzare la matrice con le caselle occupate e stampare ad ogni mossa i punteggi dei due giocatori; consentirà di inserire da prompt le coordinate della casella.*
-
- *SE il progetto è fatto in GRUPPO, si richiede inoltre che:*
- *la tastiera sia rappresentata tramite interfaccia grafica*
- *Tutti I valori numerici che ho inserito come spunto siano parametrici.*
- *Ci possano essere K giocatori*
-
- *Sono benvenute, anche se non richieste, funzionalià' aggiuntive, esempio:*
- *Giocare contro il computer, che ha una strategia*
- *scegliere i simboli da usare per le caselle*
- *ogni altro ampliamento*

Progetto 2:

realizzare una interfaccia grafica in Python per il DB delle ripetizioni, funzionalmente simile a quella Android.