Applicazioni Web I – Esame #5 (scadenza 2023-01-19 alle 23:59)

“Poke”

VERSIONE FINALE – non ci sono state modifiche

Progettare e implementare un’applicazione web per gestire gli ordini take-away in un negozio di poke. Si noti che, per semplicità, non è necessario gestire le date: si assume che tutti gli ordini siano effettuati nello stesso giorno (non specificato).

L’applicazione deve soddisfare i seguenti requisiti.

Le ciotole di poke sono essere di 3 dimensioni differenti: Regular, Medium, Large (R, M, L). Ogni ciotola ha una base (riso) e può essere ordinata scegliendo tra differenti quantità di proteine ed ingredienti, secondo la dimensione della ciotola:

- La ciotola “Regular” include una proteina e fino a 4 ingredienti.
- La ciotola “Medium” include due proteine e fino a 4 ingredienti.
- La ciotola “Large” include tre proteine e fino a 6 ingredienti.

Le proteine sono: tonno, pollo, salmone, tofu.

Gli ingredienti sono: avocado, ananas, anacardi, cavolo, mango, peperoni, mais, wakame, pomodori, carote, insalata.

Un ordine può contenere **una o più ciotole**. Più ciotole della stessa dimensione e con le stesse proteine ed ingredienti possono essere selezionate modificando un apposito contatore durante la fase di ordinazione: per esempio, deve essere possibile ordinare 3 ciotole R con salmone, avocado e cavolo senza reimmettere tre volte le stesse proteine e gli stessi ingredienti ma semplicemente selezionando 3 come numero di ciotole.

Ogni giorno il negozio può produrre solamente un **numero massimo di ciotole di ciascuna dimensione**: rispettivamente, 10 R, 8 M, 6 L. Questa informazione deve essere immagazzinata nel database, insieme ai prezzi delle ciotole, discussi nel seguito.

Qualsiasi utente (autenticato o no) può **visionare liberamente** la disponibilità delle 3 dimensioni di ciotole, così come la lista delle proteine ed ingredienti che possono essere usati.

Un utente autenticato può creare il suo ordine creando una **lista di ciotole di poke**, specificando le caratteristiche in una pagina di configurazione interattiva. Tutta l’interazione per la configurazione delle ciotole deve essere gestita sul client, ad eccezione del controllo delle disponibilità delle dimensioni delle ciotole che deve essere effettuato in tempo reale quando l’ordine è completo e sta per essere inviato, per evitare di lasciar configurare all’utente delle ciotole che poi non sarebbero disponibili.

Mentre l’utente configura l’ordine, il prezzo totale è aggiornato in tempo reale a ogni cambiamento (aggiunta/rimozione, o modifica delle proteine/ingredienti o la quantità).

Il **prezzo totale** è la somma del costo di tutte le ciotole nell’ordine. Ogni dimensione di ciotola (R, M, L) ha un prezzo che è immagazzinato nel database: R costa 9 euro, M costa 11 euro, L costa 14 euro.

Ogni *ingrediente* in aggiunta alle quantità previste (cioè 4 per le ciotole R ed M, 6 per le ciotole L) aumenta del 20% il prezzo iniziale della ciotola. Si applica uno sconto del 10% sul totale dell'ordine se più di 4 ciotole sono incluse nell'ordine.

Quando l'utente invia l'ordine da eseguire al negozio, il negozio verifica che un numero sufficiente di ciotole delle dimensioni richieste siano ancora disponibili e conferma l'ordine, altrimenti un opportuno messaggio informativo viene mostrato (per es. non ci sono sufficienti ciotole L), e l'utente è reinviato al configuratore che deve consentire di correggere l'ordine appena inviato.

Gli utenti autenticati devono poter sempre controllare la lista dei propri **ordini passati**: la lista deve mostrare una riga per ogni ordine. La riga mostra il numero totale di ciotole e il prezzo totale pagato. Una vista dettagliata dell'ordine che include le stesse informazioni presentate nel configuratore deve essere mostrata (e nascosta) quando la riga viene cliccata.

Suggerimento: una vista immutabile del configuratore può essere riciclata a questo scopo se si ritiene appropriato.

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" con lo Strict Mode attivato.
- La valutazione del progetto sarà effettuata navigando nell'applicazione. Non saranno testati né il comportamento del bottone "refresh" del browser, né l'immissione manuale di un URL (ad eccezione di /), e il loro comportamento non è da ritenersi specificato. Inoltre, l'applicazione non dovrà mai "ricaricarsi" (ossia ricaricare la pagina del browser) come conseguenza dell'uso normale dell'applicazione stessa.
- 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 logout) e l'accesso alle API devono essere realizzati tramite `passport.js` e cookie di sessione, utilizzando il meccanismo visto a lezione. Le credenziali devono essere immagazzinate in forma cifrata e "con sale". 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 3 utenti, con almeno un utente che abbia effettuato 2 ordini, uno che abbia effettuato 1 ordine, e un totale di 10 ciotole ordinate nel giorno corrente (non specificato).

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 configurare una ciotola di poke.** Lo screenshot deve essere embeddato nel README linkando una immagine messa nel repository.
    b. Username e password degli utenti.

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: negli ultimi anni 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()`.