Progetto d’esame

Tecnologie e applicazioni web, a.a 2019/2020

Si realizzi un’applicazione web, comprensiva di server con API stile REST e front-end di tipo SPA, che permetta la compravendita (con un modello ad asta) di libri universitari usati da parte degli studenti.

L’applicazione gestisce due tipologie di utenti: **studenti** e **moderatori**. Gli studenti possono accedere all’applicazione per vendere oppure comprare dei libri. I moderatori possono manualmente intervenire sui libri in vendita, ad esempio per rimuovere un’inserzione o forzarne la modifica dei contenuti.

Il workflow dell’applicazione è simile a quello offerto da siti d’aste stile EBay (www.ebay.it). Un utente, anche se non registrato, può accedere per vedere la lista dei libri attualmente in vendita. Deve essere possibile restringere la ricerca secondo vari parametri, tra cui il titolo del libro, il corso in cui esso è utilizzato (e in quale Università), l’area geografica del venditore e il prezzo attuale d’asta.
Previa registrazione e autenticazione al sistema, uno studente può fare un’offerta ad una o più aste in corso. Alla scadenza dell’asta, lo studente che ha offerto il prezzo più alto (purché superi il prezzo di riserva) risulta vincitore e può procedere all’acquisto contattando lo studente venditore.
Ogni studente può creare una nuova inserzione, specificando i dati del libro, la durata dell’asta, il prezzo di partenza e il prezzo di riserva[1], cioè il prezzo che deve essere superato perché la vendita vada a buon fine. Una volta creata l’inserzione, essa può essere rimossa soltanto da un moderatore o alla scadenza della stessa.

Gli studenti possono comunicare, anche in tempo reale, con gli inserzionisti lasciando dei messaggi. I messaggi sono associati a ciascun libro in vendita e possono essere privati, visibili cioè solo da mittente e destinatario, o pubblici, visibili cioè da tutti gli studenti che osservano un’inserzione.

[1] Nota: Il prezzo di riserva è visibile solo all’inserzionista. Gli altri studenti che osservano l’inserzione non possono conoscerlo fino ad asta ultimata.

Architettura

Il sistema deve essere composto da:

- Un web service di **backend** con API in stile REST, implementato in TypeScript su ambiente Node.js. Il servizio deve inoltre utilizzare:
    - Il DBMS MongoDB per gestire la persistenza dei dati
    - Il middleware Express.js per la gestione del routing

- Una web application di **front-end** in stile SPA realizzata con il framework Angular. L'applicazione, oltre alla versione web standard, deve comprendere anche:
    - Un'applicazione mobile ibrida realizzata con Apache Cordova
    - Un client desktop realizzato con Electron

Le applicazioni mobile e desktop sono sostanzialmente identiche alla web application Angular browser-based ma possono includere funzionalità aggiuntive (ad esempio l'applicazione mobile può presentare elementi di interfaccia tipici del contesto mobile).
Il front-end realizzato è unico per ciascuna tipologia di utente precedentemente descritta. A seconda del ruolo dell'utente a seguito del login (moderatore o studente) verranno presentate pagine e funzionalità diverse.

Funzionalità

Il sistema deve implementare le seguenti funzionalità:

1. Gestione degli utenti
    a. Registrazione di nuovi studenti
    b. Registrazione di nuovi moderatori "su invito". I moderatori non possono registrarsi autonomamente ma devono essere registrati da un altro moderatore inserendo un nome ed una password temporanea. Al primo login il nuovo moderatore dovrà quindi impostare le proprie credenziali (password, nome, cognome, etc.)
    c. Cancellazione di studenti esistenti da parte dei moderatori
    Nota: Tutte le funzionalità del sistema svolte dai moderatori sono accessibili previo login obbligatorio.
2. Gestione delle inserzioni
    a. Inserimento di una nuova inserzione, da parte di uno studente autenticato, per iniziare l'asta.
    b. Visualizzazione delle aste attive da parte di tutti gli utenti, anche non registrati
    c. Possibilità di filtrare le inserzioni attive su diversi parametri, come titolo del libro, corso di laurea, utente, prezzo, etc.
    d. Possibilità di fare un'offerta ad un'inserzione. L'offerta deve ovviamente superare il prezzo attuale d'asta.
    e. Invio di messaggi al venditore (gestore dell'inserzione). Il messaggio può essere privato oppure pubblico. Il venditore può rispondere al messaggio, e la visibilità della risposta sarà la stessa del primo messaggio inviato dal

potenziale compratore. I messaggi sono associati ad una specifica inserzione.
    f. Modifica di tutti i dati di un'inserzione da parte di un moderatore.
3. Compravendita
    a. Allo scadere dell'asta, lo studente che ha offerto il prezzo più alto (purché superi il prezzo di riserva) risulta vincitore. Il sistema deve notificare sia lo studente vincitore sia l'inserzionista sull'esito dell'asta.
4. Statistiche
    a. Il sistema deve fornire a ciascuno studente la lista di inserzioni che ha effettuato, le aste a cui ha partecipato e quelle in cui è risultato vincitore.
    b. Il sistema deve fornire a ciascun moderatore il numero di aste attive, concluse con successo e concluse senza il raggiungimento del prezzo di riserva.

I gruppi sono liberi di implementare funzionalità aggiuntive oltre quelle precedentemente elencate. L'implementazione o meno di funzionalità aggiuntive non preclude né garantisce il conseguimento della lode ma contribuisce a definire un giudizio positivo.

Il sistema deve, in fase di avvio, precaricare nel database con una lista (anche ristretta) di studenti, moderatori ed inserzioni per testarne il funzionamento durante l'esame.

Consegna

La consegna avviene esclusivamente mediante piattaforma Moodle alla pagina:

https://moodle.unive.it/mod/assign/view.php?id=163442

Ciascun gruppo deve consegnare un solo file, in formato zip, chiamato `<nomegruppo>.zip`.

Il file zip al suo interno deve contenere:

- La relazione di ciascuno studente, in formato pdf, chiamata `<nome>_<cognome>_<matricola>.pdf`
- Un file README.txt che spiega chiaramente come eseguire l'intera applicazione. Ad esempio, la lista di comandi da dare per installare le dipendenze (npm install o altro), eseguire il processo del DBMS, il server e i relativi client;
- Tutti i sorgenti necessari alla sua esecuzione.

NOTA: Non consegnare le librerie esterne installate attraverso il gestore pacchetti. In altre parole, tutte le directory `node_modules` devono essere eliminate.

Relazione

Congiuntamente al progetto, ciascuno studente deve consegnare una relazione.
La relazione va compilata in forma strettamente individuale e porrà le basi per la discussione orale del progetto.

La relazione deve contenere:

- Una descrizione dell'architettura del sistema, quali sono i componenti e in che modo questi concorrono a soddisfare le features richieste;
- Una descrizione del modello dei dati utilizzato. Quali sono le collezioni e qual'è la struttura dei documenti di ciascuna collezione che vengono memorizzati nel database;
- Una descrizione delle API fornite dalla componente server. La descrizione deve contenere in modo chiaro la lista degli endpoints, degli eventuali parametri e il formato dei dati (JSON) che vengono scambiati nelle richieste HTTP.
- Una descrizione di come è stata realizzata l'autenticazione degli utenti, con relativo workflow;
- Una descrizione del client web realizzato con il framework Angular. In particolare, la lista dei vari Components, Services e delle Routes;
- Alcuni esempi, corredati possibilmente da screenshots, del workflow tipico dell'applicazione.

Altre informazioni

Per domande o chiarimenti è sempre possibile contattare il professore via mail all'indirizzo filippo.bergamasco@unive.it

Per domande sulla modalità di esame si faccia inoltre riferimento alla seguente pagina:

https://moodle.unive.it/mod/page/view.php?id=131341

Filippo Bergamasco,
Tecnologie e Applicazioni Web, a.a. 2019/2020