Programmazione Web 2024/25

Requisiti progetto d'esame

Introduzione

Vogliamo realizzare una applicazione web che funga da **sito per aste**. L'applicazione permetterà ad un utente registrato di **mettere degli oggetti all'asta** e di **fare offerte** per altri oggetti. Sarà possibile fare delle **ricerche** per trovare le aste di interesse. Gli oggetti all'asta avranno un'**offerta corrente** (di base l'offerta minima), un **titolo** dell'inserzione, una **descrizione** ed una **data di fine asta**. Le offerte saranno valide solo se pervenute prima della data di fine asta. L'offerta più alta allo scadere vincerà l'asta.
Si potrà vedere lo storico di ogni asta con l'elenco delle offerte accettate.
Non sono previste forme di gestione dei pagamenti.
Gli utenti si registrano con uno **username** unico, **nome** e **cognome**. Si autenticheranno mediante una **password**.
Il progetto si comporrà della **parte server**, dove verranno memorizzati i dati e gestita la fase di autenticazione/autorizzazione, e della **parte client** che visualizzerà l'applicazione ed i dati.

Regole

Il progetto va sviluppato in **autonomia**. Potete usare i framework e le librerie che volete, l'importante è che **venga rispettata l'API fornita**. Potete usare un qualsiasi LLM per sviluppare il progetto, ma siete **responsabili** di ogni singola riga di codice. Dovete essere in grado di giustificare ogni scelta fatta (libreria, framework, scelta progettuale). Suggerimento: affidatevi ad un LLM se sapete cosa fare e come farlo, ma volete farlo meglio. Se non sapete come farlo, usate un LLM a vostro rischio e pericolo.

Funzioni principali

Legenda:
Necessaria autenticazione
Non necessaria autenticazione

ASTE

- **Elenco di tutte le aste:** sarà possibile visualizzare un elenco di tutte le aste, attive o scadute, eventualmente filtrando con un parametro di ricerca.
- **Creazione nuova asta:** un utente autenticato può fare una nuova inserzione per un'asta. L'asta conterrà:
  - Titolo
  - Descrizione dell'asta
  - Data di scadenza
  - Offerta di partenza

è da capire dove salvare l'informazione di chi è autore di quell'asta!!

- **Visualizzazione dei dettagli dell'asta:** sarà possibile vedere i dettagli di ogni asta, tra cui l'offerta corrente e l'utente vincitore
- **Modifica di un'asta:** l'utente che ha creato l'asta potrà modificare alcune informazioni (non tutte!), in particolare
  - Titolo

    - Descrizione
- **Eliminare un'asta:** l'utente che ha creato l'asta potrà eliminarla quando vuole (per semplicità).

OFFERTE

- **Nuova offerta**: un utente autenticato potrà fare una nuova offerta per un'asta. L'offerta sarà accettata solo se maggiore dell'offerta corrente, altrimenti restituirà un errore. Le offerte ricevute dopo la scadenza dell'asta saranno rifiutate e l'offerta più alta risulterà vincitrice dell'asta.
- **Elenco offerte**: per ogni asta sarà possibile vedere lo storico di tutte le offerte
- **Dettagli offerta**: per ogni offerta si potranno vedere i dettagli, come:
    - utente che ha fatto l'offerta
    - data dell'offerta
    - valore dell'offerta

UTENTI

- **Elenco utenti:** sarà possibile vedere un elenco degli utenti, eventualmente facendo una ricerca. Per ogni utente sarà possibile visualizzare le offerte vinte.
- **Registrazione nuovo utente**: un utente potrà registrarsi fornendo
    - nome utente (*username*)
    - password (*password)*
    - nome (*name*)
    - cognome (*surname*)

Interfaccia REST

Il progetto prevede che sia realizzata una interfaccia REST. L'API da implementare è riportata di seguito.

Metodo	API	Descrizione
POST	/api/auth/signup	Registrazione di un nuovo utente
POST	/api/auth/signin	Login di un utente
GET	/api/users/?q=*query*	Elenco degli utenti, si può filtrare con il parametro *q*
GET	/api/users/*:id*	Dettagli dell'utente con identificativo *id*
GET	/api/auctions?q=*query*	Elenco di tutte le aste, si può filtrare con il parametro *q*
POST	/api/auctions	Crea una nuova asta, solo per utente autenticati
GET	/api/auctions/*:id*	Dettagli dell'asta specifica con identificativo *id*
PUT	/api/auctions/*:id*	Modifica di un'asta esistente con identificativo *id*, previa autenticazione
DELETE	/api//api/auctions/*:id*	Elimina un'asta esistente con identificativo *id*, solo il creatore dell'asta può
GET	/api//auctions/*:id*/bids	Elenco delle offerte per l'asta con identificativo *id*
POST	/api/auctions/*:id*/bids	Nuova offerta per l'asta con identificativo *id,* previa autenticazione
GET	/api/bids/*:id*	Dettagli dell'offerta con identificativo *id*
GET	/api/whoami	Se autenticato, restituisce le informazioni sull'utente

Consegna del progetto

Per consegnare una demo funzionante del progetto consiglio caldamente di usare Docker. Questo strumento permette di realizzare dei *container*, degli ambienti virtuali che contengono tutto quello che serve per distribuire ed eseguire una applicazione.
Usate nomi di container che siano univoci, possibilmente con il vostro cognome!

Un esempio di configurazione di due container che usi node.js e mongodb:

```
version: "3"
services:
 app:
   container_name: app
   build: .
   command: nodemon --watch /usr/src/app -e js app.js
   ports:
     - "3000:3000"
   volumes:
     - ./app:/usr/src/app
   links:
     - "mongo:mongosrv"
 mongo:
   container_name: mongo
   image: mongo
   volumes:
     - ./data:/data/db
   ports:
     - '27017:27017'
```

Associato ad un Dockerfile per il container “app”:

```
FROM node:latest
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app
RUN npm install -g nodemon
COPY ./app/package.json /usr/src/app
RUN npm install
COPY ./app /usr/src/app
EXPOSE 3000
```

Potete anche usare altre soluzioni, non necessariamente MongoDB. Se preferite usare MySQL, fate pure, trovate i container con MySQL già pronto (va configurato).
Se dovete darmi una demo funzionante, allegate uno script che carichi i dati nel database.

Non inviatemi giga e giga di dati relativi al database. Soprattutto **non inviatemi i pacchetti di node**!