Applicazioni Web I – Esame # 2 (scadenza 2021-07-12 alle 23:59)

"Generatore di meme"

VERSIONE FINALE – Aggiunte e correzioni sono riportate in rosso

Progettare e implementare un'applicazione web per gestire la creazione di *meme* e la loro visualizzazione.

L'applicazione deve soddisfare le seguenti specifiche.

Un meme è composto dalle seguenti proprietà:

- un **titolo**,
- un'**immagine**,
- una o più **frasi** sovrapposte all'immagine,
- una **visibilità** (pubblico o protetto),
- ha associato il **nome** del suo creatore.

Per semplicità, si supponga che il sito disponga già di un insieme di immagini "di sfondo" predefinite (e non modificabili, servite da React o Express) a partire dalle quali si possono creare i meme. Ciascuna immagine predefinisce anche il numero di campi di testo supportati (1, 2 o 3), e la posizione di tali campi di testo nell'immagine stessa (immagini diverse possono posizionare i campi in posti diversi). La definizione di un meme, quindi, consiste semplicemente nella scelta dell'immagine di sfondo e nella definizione del contenuto dei campi di testo previsti per tale immagine. Non è richiesta la creazione di una "bitmap" con il meme completato, ma più semplicemente la visualizzazione dell'immagine con il testo sovrapposto nelle posizioni predefinite.

I meme possono essere creati unicamente da utenti di tipo **creatore**. I meme con visibilità "pubblico" saranno visibili a tutti i visitatori dell'applicazione web, mentre quelli "protetti" saranno visibili unicamente agli altri creatori.

Chiunque visiti il sito (quindi qualunque utente non autenticato), visualizza l'elenco[1] dei meme *pubblici* nella home page, e selezionandone uno potrà vederne le proprietà: titolo, immagine con testo sovrapposto (cioè il meme vero e proprio) e il nome del creatore.

Gli utenti **creatori** devono autenticarsi con username/password. Una volta autenticato, un creatore vede l'elenco dei meme *pubblici e protetti*, e selezionandone uno ne potrà vedere tutte le proprietà. Inoltre, dalla pagina con l'elenco dei meme potrà:

- Creare un nuovo meme, definendo il **titolo**, l'immagine di sfondo e le frasi da utilizzare. In questa fase, le azioni possibili sono:
    - Selezionare un'**immagine** (scegliendo tra quelle predefinite) che serva come sfondo per il meme.
    - Scrivere il **testo** del meme, che comparirà nelle aree predefinite dall'immagine. Tutti i campi di testo sono da considerarsi opzionali (ad esempio, se un'immagine definisce due campi di testo, il creatore può decidere di compilarne uno solo), ma almeno uno deve

[1] Il titolo è richiesto, la visualizzazione dell'anteprima (thumbnail) è permesso ma non richiesto.

essere compilato. Nello specificare il testo dell'*intero* meme, il creatore deve poter scegliere tra almeno 2 tipi di font (di dimensione prefissata) e 4 colori diversi.
    - Salvare il meme così creato, scegliendo la sua visibilità ('pubblico' o 'protetto').
- Copiare un meme, considerando i seguenti due casi:
    - Se il meme è proprio (dello stesso creatore), eseguirne una copia e permettere il cambiamento di titolo, frasi, visibilità, attributi del testo (cioè, tutto *tranne* l'immagine).
    - Se il meme non è proprio, eseguirne una copia e permettere il cambiamento di titolo e testo (inclusi i suoi attributi). Se il meme da copiare è protetto, non deve essere possibile cambiare la visibilità; se è pubblico, è possibile salvare la copia come pubblica o protetta. La copia apparterrà al nuovo creatore.
- Eliminare un proprio meme.

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 "React Development Proxy" 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 server.js`" e "`cd client; npm start`". Viene fornito uno scheletro delle directory del progetto. Si può dare per scontato che `nodemon` sia già installato a livello di sistema operativo.
- 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 tutte.
- L'autenticazione dell'utente (login) e l'accesso alle API devono essere realizzati tramite `passport.js` e cookie di sessione. Non è richiesto alcun ulteriore meccanismo di protezione. La registrazione di un nuovo utente non è richiesta.
- Il database del progetto deve essere definito dallo studente e deve essere precaricato con *almeno* 3 utenti creatori, i quali abbiano creato *almeno* 3 meme ciascuno, di cui uno tramite copia di un meme di un altro utente. Devono essere predefinite almeno 6 immagini, di cui almeno 2 con 1 campo, 2 con 2 campi e 2 con 3 campi di testo.

Contenuto del file README.md

Il file README.md deve contenere le seguenti informazioni (un template è disponibile nello scheletro del progetto – in generale, ogni spiegazione non dovrebbe essere più lunga di 1-2 righe):

1. Una lista delle route dell'applicazione React, con una breve descrizione dello scopo di ogni route
2. Una lista delle API HTTP offerte dal server, con una breve descrizione dei parametri e degli oggetti scambiati
3. Una lista delle tabelle del database, con il loro scopo
4. Una lista dei principali componenti React usati
5. Uno screenshot della **pagina per creare un meme** (embeddando una immagine messa nel repository)
6. Username e password degli utenti creatori, indicando anche quali creatori hanno creato quali meme.

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 alla propria 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
```

In alternativa, è 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 gli `import` e i `require()`.