Interlogica Website Pasticceri

Documentazione tecnica per l'applicazione Full stack Mern

Introduzione:

L'applicazione Full Stack è un'applicazione web che utilizza le tecnologie React.js, Redux.js, MongoDB, Node.js e Express.js per fornire un applicativo E-commerce di una pasticceria.

Requisiti di sistema:

L'applicazione può essere eseguita su qualsiasi sistema operativo. Tuttavia, per il corretto funzionamento dell'applicazione, è necessario disporre delle seguenti componenti software:

- React.js
- Node.js
- MongoDb
- Express.js

Installazione:

Per installare l'applicazione, seguire i seguenti passaggi:

- Clonare il repository dall'URL https://github.com/Dcapalbo/MERN-stack-website-pastry-shop.git
- Aprire due finestre del terminale e spostarsi nelle relative directory dell'applicazione (client/server)
- Seguire i passaggi contenuti dentro i due file di ReadMe, dentro le rispettive cartelle client e server.

Utilizzo:

Una volta avviata l'applicazione, l'utente atterra sulla home, dalla quale è possibile navigare attraverso una nav bar contenuta nell'header, nella sezione in alto dell'applicativo. Per iniziare il giro dell'applicativo occorre creare un utenza. I processi possibili sono due, è possibile lanciare i seeders per popolare il database

Con i dati relativi alle utenze e i dolci, oppure è possibile creare un utenza grazie all'utilizzo dell'interfaccia utente visitando la rotta "crea account". Una volta creato le credenziali, l'utente viene redirezionato sulla pagina login dove è possibile effettuare l'autenticazione, nonché la creazione del token(JWT), che verrà utilizzato per filtrare voci, link di navigazione e azioni, le quali risulteranno nascoste ad utenti non autenticati. Completata l'autenticazione, l'utente, viene redirezionato dentro l'area di back office, dove, attraverso un interfaccia, sarà possibile eseguire le seguenti operazioni:

- Aggiungere nuovi Dolci
- Modificare dei Dolci precedentemente creati
- Eliminare dei Dolci precedentemente creati
- Aggiungere o eliminare unità di un dolce specifico

Nel caso in cui i relativi seeders siano stati eseguiti, come da ReadMe (server folder), al momento del caricamento della pagina "/" , verrà effettuata una chiamata Get, per eseguire una fetch e di conseguenza popolare la home con tutti i dolci disponibili all'interno del database.
Qualora non fossero stati lanciati i seeders, occorre inserirli uno ad uno manualmente. Tornando sulla home "/" è possibile filtrare i dolci presenti all'interno della rotta, attraverso una select centrata nel DOM, inoltre, cliccando su uno dei dolci presenti, l'utente viene redirezionato verso una pagina di dettaglio in cui è possibile osservare tutti i dettagli del dolce, come per esempio gli eventuali sconti, la quantità, la descrizione e gli ingredienti. L'applicativo è in doppia lingua, in termini di navigazione, ma per il momento i database nella seconda lingua, ENG, non sono stati ancora creati, quindi è consigliabile effettuare tutte le operazioni di CRUD solo ed esclusivamente da lingua Italiana, nonché lingua base dell'applicativo. Per concludere il giro è consigliabile eseguire il logout dall'applicativo una volta terminato il giro dello stesso.

Struttura dell’applicazione:

L'applicazione Full Stack MERN è stata implementata seguendo il pattern MVC (model, view, controller), ed è suddivisa come segue:

- Frontend: (V) la parte dell'applicazione che gestisce l'interfaccia utente. Questa parte utilizza React e Redux per la gestione dello stato dell’applicazione. React ha il ruolo della view in questo applicativo. La Web app è una SPA (single page application), dunque l’applicativo e i suoi relativi componenti non vengono serviti al cambio di ogni rotta, bensì, tutto il javascript viene caricato quando l’utente atterra sull’applicativo. Una chiamata GET è stata predisposta al momento in cui viene montato l’applicativo. Terminata la GET, viene popolato lo store di redux, attraverso un azione dello slice, la quale, grazie ad un persistor, si occupa di mantenere accessibili gli state e i relativi payload durante la navigazione dell’applicativo. React utilizza React Hook Form sia per la gestione dei form dinamici che di quelli statici. La validazione degli specifici Schema è in mano a Zod, una libreria di validazione. Una volta che l’autenticazione viene eseguita, redux inserisce i dati di autenticazione nello store, affinché sia possibile riutilizzarli per renderizzare dinamicamente determinate rotte e/o azioni non disponibili per un utente non autenticato. Dentro l’area di back office, è presente un ulteriore chiamata GET, la quale viene innescata in due precise occasioni, quando l’utente esegue una UPDATE di un dolce e quando esegue una CREATE, in questi casi, l’utente viene redirezionato all’area di back office in cui la chiamata GET dei dolci torna a inserire nuovamente i dati aggiornati nello store di redux. Redux, lato front-end è il fulcro del flusso dei dati dell’applicativo e lo ritroviamo anche in altre occasioni. Nella creazione della pagina di dettaglio del dolce, ad esempio, al click dell’immagine, Redux inserisce nello stato i dati del dolce singolo, grazie ai quali ne renderizza il componente in una pagina apposita e infine, durante le UPDATE/DELETE dei dolci e delle quantità dei dolci, il processo di gestione del flusso dei dati rimane il medesimo.

- Backend: la parte server dell'applicazione che gestisce la logica (Controller) legata alle CRUD e lo schema delle collections per mappare (Model). L'applicativo utilizza MongoDB e Node.js con il framework Express.js per la creazione di un'API REST, Sono stati implementati due model, uno per i dolci e uno per gli utenti, ognuno di essi possiede il relativo seeder per popolare e rendere attivo l’applicativo, i controller dei dolci e degli utenti si occupano di gestire la logica delle CRUD, sia per i dolci che per gli utenti. Node.js grazie al core event loop di javascript riesce a mettere in coda e svolgere in parallelo le chiamate HTTP, limitando i tempi di attesa probabili e potenziali. Nel back-end per soddisfare una delle richieste, nell’implementazione del website pasticceria, è stato usato un modulo che pianifica determinate azioni in una linea temporale (node-cron), in questo caso è stato utilizzato per rimuovere i dolci dopo quattro giorni dalla loro creazione. Un altra funzionalità che fa capo al back-end, è quella di calcolare, prima di eseguire una GET dei dolci, da quanto tempo essi sono stati creati e inseriti nel “magazzino”, in base a questo lasso di tempo vengono applicati determinati sconti. Infine, come best practices, sono stati utilizzati vari middleware, quali body-parser, un middleware Express.js che analizza i dati delle richieste input in entrata rendendoli disponibili nell'oggetto di richiesta, Multer, un middleware di gestione di file multipli che consente di caricare file attraverso richieste HTTP. Cors, un meccanismo di sicurezza del browser che limita le richieste HTTP da un'origine diversa dall'origine del server. il quale permette di accedere alle risorse da un domain ad un altro, infine express validator, utile a semplificare le validazione dei dati di input in arrivo dal front-end nelle richieste HTTP.

Conclusioni:

L'applicativo è una SPA che consente agli utenti di navigare attraverso una nav bar nell'header e di eseguire varie operazioni, come creare un account, autenticarsi e accedere all'area di back office per aggiungere, modificare ed eliminare dolci. Grazie alla validazione degli specifici Schema in mano a Zod, la gestione dei form dinamici e di quelli statici è in grado di soddisfare le esigenze degli utenti. Tuttavia, è importante notare che, poiché l'applicativo è in doppia lingua, i database nella seconda lingua non sono ancora stati creati e quindi, per il momento, è consigliabile eseguire tutte le operazioni di CRUD solo ed esclusivamente in lingua italiana. Infine, una volta terminato il giro dell'applicativo, è consigliabile eseguire il logout.