
UNIVERSITÀ
DI PARMA

## DIPARTIMENTO DI INGEGNERIA E ARCHITETTURA

INGEGNERIA DEI SISTEMI INFORMATIVI

Tecnologie Internet

# E-COMMERCE PER LA VENDITA DI LIBRI


Studente: Romano Cosimo

Matricola: 284867

Github: Cosimo7 (github.com)

Anno Accademico 2019/2020

# INTRODUZIONE

La libreria Romanelli è una **single-page application** di e-commerce per la vendita di libri sviluppata con il framework React-JS per il front-end mentre per la gestione back-end la piattaforma Google Firebase.

# TECNOLOGIE DI SUPPORTO

**Visual Studio Code:** editor di testo proprietario ideato da Microsoft che permette di accedere a diverse funzionalità tipicamente proprie degli IDE, come ad esempio il syntax highlighting, controllo del codice, integrazione con Git, snippets di codice e estensioni per molti linguaggi. Inoltre la possibilità di accedere al terminale da VS Code è particolarmente comoda per il debugging del progetto o di qualsiasi altra applicazione web.

**Github:** servizio di hosting per sofware. Gli utenti possono condividere il loro codice ed interagire tra loro tramite fork con cui modificare altro codice senza alterare l'originale e contribuire inviando un eventuale fix di questo attraverso pull requests. Sono molto attivi anche forum di discussione su Github stesso.

# TECNOLOGIE FRONT-END

**ReactJS:** è un web framework Javascript per la creazione di user interfaces, utilizzato prevalentemente per lo sviluppo di single-page applications che si occupa esclusivamente del rendering dei dati. Infatti React si avvale di altre librerie per l'implementazione di funzioni necessarie per il front-end. Per la realizzazione del progetto sono state scelte le seguenti:

- **React Router:** libreria finalizzata alla gestione del routing con l'ausilio delle Routes e Switch, components che si introducono nel JSX del nodo principale dell'applicazione (App.js) per fissare i diversi path rendendo la navigazione particolarmente fluida con l'utilizzo del **Link**. Il componente Link è un' alternativa all'attributo HTML **href** e al passaggio ad un altro componente non ricarica l'intera applicazione ma solamente la pagina di riferimento rendendola più rapida nei caricamenti.

- **Redux:** state container per applicazioni Javascript eseguibile sia lato client che server. Permette di scrivere funzioni utili alla gestione dei dati(**actions**), ad esempio per aggiungere un prodotto al carrello si può settare **initialState** con relativo array vuoto per poi popolarlo attraverso il metodo **dispatch**. Rappresenta la logica dei dati forniti con Context

- **Context:** API che consente di rendere globali alcune variabili evitando props drilling, cioè una criticità che sorge quando si vogliono estrapolare attributi(props) di un dato dalla componente figlia a partire da un nodo distante nella struttura ad albero delle componenti dell'applicazione.

# TECNOLOGIE BACK-END

La scelta della piattaforma per la gestione lato server è ricaduta su **Firebase**, suite sviluppata da Google che offre diverse funzionalità come autenticazione, database noSQL in cloud, servizio di machine learning, analytics a supporto della propria applicazione web o mobile e altre ancora. Tuttavia per lo svolgimento del progetto ho ritenuto opportuno utilizzare due di questi servizi:

- **Firebase authentication:** servizio back-end utilizzato per la gestione degli utenti all'interno di una web app in real-time. Supporta l'autenticazione attraverso indirizzo mail e password, ma anche numero di telefono. Per la mia applicazione web ho creato un form e una funzione login con l'ausilio di authentication. I metodi di verifica password e mail sono facilmente utilizzabili grazie all'SDK fornita da Firebase.

- **Cloud Firestore:** database cloud noSQL scalabile, utilizzato per l'archiviazione dei dati. Nello specifico è un **document-oriented database** che sfrutta valori JSON con diversi tipi di dato per la rappresentazione dei dati manipolati con operazioni concettualmente sovrapponibili a quelle di un database relazionale ma eseguite più flessibilmente in virtù della natura slegata del noSQL. Le query vengono lanciate con metodi offerti da Firestore come **where()** o **limit()**, che richiamano esattamente le query SQL.

Per il mio progetto ho creato tre **collection**: FirstRowBooks, SecondRowBooks, ThirdRowBooks. Ogni collection è composta da dieci **documents**, per meglio dire il singolo prodotto descritto dalla notazione JSON. I campi dei libri sono: author, genre, image, price, title, year. L'id viene generato automaticamente da firestore ed è la key.

# STRUTTURA DEL SITO

Dato l'utilizzo di React, lo stile del progetto è stato concepito con JSX, una metodologia ibrida per la creazione grafica dell'applicazione composta da codice Javascript e HTML.

Con la convenzione di scrittura BEM (block-element-modifier) gli elementi html sono stati associati al relativo CSS con className, in modo da rendere leggibile il codice con sigle intuitive nel caso in cui abbia voluto apportare modifiche successive. Di seguito riporto alcune components del progetto:

- ## HEADER

L'header presenta il logo del progetto e le voci per: accedere ai tre generi del sito (horror, thriller, fantasy), novità, login e carrello.

- ## HOME

Nella Home sono rappresentati alcuni titoli estrapolati con l'utilizzo della query **limit()** dalle tre collection disposte tramite la Product component in cui vi è la logica per la grafica del singolo libro.

- ## HORROR, FANTASY, THRILLER

Il Link non è applicato alla voce "Genere" ma ad ogni genere del menù a tendina. Per cui vi è un link per i libri horror, uno per i fantasy e l'ultimo per i thriller. Per le tre pagine vengono eseguiti tre **where()** diversi relativi al genere desiderato, in questo modo nel genere specifico vengono rappresentati tutti i libri di quel tipo.

- ## NOVITA'

La pagina novità funziona in maniera analoga ai generi però il where richiede l'anno, che in questo caso sarà "2021" trattandosi di nuovi titoli.

- ## LOGIN

Nel login è presente un form che richiede indirizzo mail e password, un button per accedere al sito e fare acquisti e un altro per creare l'account. Tutta la gestione viene affidata a **Firebase authentication**.

- ## CARRELLO

L'icona carrello presa da **Material-UI** è dotata di counter ogniqualvolta venga aggiunto o rimosso un libro. Se il carrello è vuoto compare una scritta che lo indica, mentre se sono presenti libri ci sono le card con informazioni, un button per la rimozione ed un'altra card che indica l'importo totale dell'acquisto con il button per finalizzare l'acquisto.

L’applicazione forza l’accesso se si tenta di acquistare i prodotti senza essere loggati. Dopo l’accesso o la creazione di un nuovo account si viene riportati alla home per fare altri acquisti oppure comprare i libri precedentemente scelti.