# Documentazione Tecnica

## Artigianato Online

Grieco Giacomo – 752466
Chiara De Rosa – 754162

## Introduzione

Abbiamo realizzato questa Web App di un sito e-commerce pensato per gli artigiani che decidono di pubblicare i propri articoli online. Abbiamo cercato di mantenere una struttura semplice per non rendere troppo complicata la navigazione e la consultazione del catalogo.

Per quanto riguarda il lato **front-end** abbiamo utilizzato **HTML** e **CSS** per la struttura e lo stile della pagina, utilizzando anche **JavaScript** per l'interattività e dinamicità; mentre Per il **back-end**, abbiamo adottato **Node.js** e il framework **Express.js**, che ci hanno permesso di costruire un server scalabile e di definire facilmente delle **API RESTful** per comunicare con il database.

Il **database** scelto è **PostgreSQL**, un sistema relazionale potente e open source

Il tutto lo abbiamo realizzato utilizzando **Docker** per garantire portabilità tra diversi sistemi operativi. In Docker troviamo 3 container:

- frontend: dove appunto gira un server http-server che fornisce tutte le pagine sulla porta 8080, e qui sono contenuti tutti i file HTML, CSS e JS con anche tutte le immagini utilizzate per la realizzazione;
- backend: dove appunto gira il server che serve sulla porta 3000, con il quale si interagisce anche con il Database tramite l'utilizzo di RESTful API;
- db: dove appunto è contenuto il Database che contiene tutte le tabelle con le informazioni relative a utenti, prodotti e ordini.

Abbiamo anche reso tutta la Web App **Progressive** così da poterla avere a disposizione anche su Desktop come se fosse una app nativa.

Durante lo sviluppo abbiamo utilizzando anche diverse tecniche come il **debounce** per non sovraccaricare il server con richieste o l'**Event Delegation**.

## Struttura del progetto

Il progetto è strutturato in 2 cartelle principali:

- **back-end**
  Strutturato in diverse sottocartelle:
  - coverage: cartella contente informazioni generate dai file di test;
  - db: che contiene il file db.js con il quale il server si connette a questo;
  - immaginiProdotti: cartella che contiene tutte le immagini dei prodotti caricati sul sito;
  - middleware: che contiene i due file con le due funzioni per verificare che l'utente nel sito abbia un token (che viene generato nel login o registrazione) e che sia valido e per verificare il ruolo per non far accedere utenti non autorizzati a zone per i quali non hanno il permesso;
  - routes/api: contenute tutte le API RESTful con il quale vengono effettuate le richieste al DB e mandate le risposte al client
  - test: contiene tutti i file per testare le API
  - utils: contiene un file funzioni.js che contiene varie funzioni usate nelle API come la funzione per l'hashing delle password e la funzione per validare la carta di credito
  - Dockerfile: file contenuto in ogni container che specifica tutte le informazioni necessarie per eseguire il programma su Docker
- **front-end**
  Anche questo strutturato in diverse sottocartelle:
  - public: che contiene tutte le cartelle con i relativi file CSS e JS per ogni file HTML e che contiene anche le immagini utilizzate per lo sviluppo del progetto
  - views: cartella che contiene tutti i file HTML di ogni pagina
  - Dockerfile: stesso funzione che ha quello nel backend

    - index.html e index.js: file HTML e JS della pagina principale, questa è la pagina che fornisce il server frontend *http-server* appena si entra nel sito

Nella cartella principale abbiamo vari file come:

- docker-compose.yml: file fondamentale per l'avvio del Docker, contiene le informazioni su tutti i container con le relative porte e informazioni
- init.sql: script per la creazione del DB
- **README.md: contiene tutte le informazioni per l'avvio della Web App**
- reset.bat/start.bat: script per l'inizializzazione del Docker su Windows
- reset.sh/start.sh: script per l'inizializzazione del Docker su macOS/Linux
- ER.pdf: file con lo schema ER del Database

## Database

Come detto nell'Introduzione, abbiamo utilizzando il database relazionale **PostreSQL** per la creazione del nostro DB. Per la struttura di ogni tabella rimandiamo a guardare lo schema ER contenuto nella root del progetto. Troviamo 6 tabelle:

- **DatiCarte**: che contiene tutte le informazioni sulle carte degli utenti registrati
- **ElencoProdotti**: che contiene tutti i prodotti caricati sul sito
- **ElencoUtenti**: che contiene tutte le informazioni degli utenti registrati
- **Ordini**: che contiene tutti gli ordini in corso
- **OrdiniProdotti**: che contiene i dettagli di ogni ordine, con i prodotti, quantità e prezzo
- **Segnalazioni**: che contiene tutte le segnalazioni effettuate

## Ruoli e Authentication/Authorization

Sono 3 i ruoli che si possono avere sul sito:

- **Cliente**: pensato per l'utente che vuole semplicemente consultare il catalogo ed effettuare acquisti; nella pagina principale è disponibile il catalogo con tutti i prodotti in vendita, se desidera cercare un prodotto preciso può effettuare la ricerca nella apposita barra a comparsa nella pagina principale, mentre se vuole fare una ricerca più approfondita può inserire i filtri con l'apposito pulsante; appena finisce di acquistare, può recarsi alla pagina del carrello dove può vedere tutto il resoconto e successivamente andare alla pagina dove può inserire tutte le informazioni per la spedizione. Nella pagina principale è disponibile anche il pulsante per effettuare segnalazioni e di fianco il pulsante per andare nella pagina del proprio account, dove può visualizzare le sue informazioni ed eventualmente modificarle;
- **Artigiano**: pensato, appunto, per tutti gli artigiani che producono prodotti e vogliono vederli; nella pagina principale, a differenza del cliente, è disponibile il pulsante per caricare un prodotto, dove si può inserire immagine e le relative informazioni, poi il pulsante per andare alla schermata del proprio account (uguale a quella del cliente) e un pulsante per visualizzare tutti gli ordini e segnare quando un ordine è stato spedito. Anche lui fornito della pagina per effettuare segnalazioni;
- **Amministratore**: account unico con il quale si accede inserendo le credenziali presenti nel README.md, qui sono disponibile informazioni sugli utenti, i prodotti e le segnalazioni effettuate.

Ogni volta che si fa il login oppure ci si registra viene assegnato all'utente un **token** con il quale ha l'accesso a tutte le pagine per il quale ha i permessi. Per il sistema dei token abbiamo utilizzato la libreria **JWT** e il token viene salvato nei cookies del sito. Ogni utente, dopo il login o la registrazione, riceve un token JWT che viene salvato nei **cookie HttpOnly**. Il server verifica questo token tramite i middleware

verificaToken e verificaRuolo, garantendo così l'accesso alle API solo se il token è valido e se il ruolo dell'utente è autorizzato.

## Test

Per effettuare i testing delle API abbiamo utilizzato la libreria **Jest** e in ogni test vengono utilizzati degli account di prova sia per il Cliente che per l'Artigiano. Tutti i test sono ripetibili dato che alla fine di ognuno vengono cancellate dal Database tutti i profili e le eventuali righe create. Per l'esecuzione dei test consultare il README.

## PWA (Progressive Web App)

La web app è stata trasformata in una Progressive Web App (PWA) con manifest JSON e service worker. In questo modo può essere installata su desktop o dispositivi mobili e usata anche offline per la visualizzazione delle pagine statiche principali.

## CSS Responsive

I file CSS della Web App sono stati ottimizzati per la corretta visualizzazione del sito su molteplici tipi di schermi. Sono state utilizzate **media query**, che adattano le componenti delle varie pagine alle dimensioni dello schermo, facendo si che l'esperienza dell'utente sia migliore in termini di usabilità, portabilità e accessibilità.

## Conclusione

L'applicazione è stata sviluppata con attenzione alla modularità, sicurezza e semplicità d'uso. Pur trattandosi di un progetto didattico, la struttura è pensata per essere estendibile con funzionalità future come notifiche, pagamenti integrati o dashboard analitiche.