Urbitek E-Commerce

Applicazioni e Servizi Web

Michael Camporesi - 879929 {michael.camporesi@studio.unibo.it}

16 Maggio 2020

0.1 Introduzione

Nella seguente relazione verrà illustrato il progetto relativo alla creazione di un'applicazione Web per gestire la parte di E-Commerce di una piccola azienda.

Nel dettaglio, l'azienda in questione è la Urbitek SRL [5], una piccola azienda di Russi che si occupa della vendita e del noleggio di impianti per la sosta. Questa azienda ha bisogno di mettere a disposizione dei suoi clienti un portale di E-Commerce per facilitare la vendita dei prodotti relativi alle macchine per la sosta vendute dall'azienda stessa.

0.2 Requisiti

Si deve dunque realizzare un'applicazione Web, improntata sul concetto di Mobile First, che possa gestire la parte di E-Commerce dell'azienda. Sono previste 3 tipologie di Utenti: gli Amministratori, che dovranno poter creare nuovi Utenti, modificarli e cancellarli, e creare nuovi Prodotti, modificarli e cancellarli. La registrazione di nuovi utenti è prevista previo invio di una mail al personale dedicato dell'azienda, che contenga Nome, Cognome, Azienda e Recapito del richiedente. In questo modo si evita che persone non realmente interessate possano registrarsi al sito.

La seconda tipologia di Utenti è quella degli Utenti Standard, che, dopo aver effettuato il Login, potranno visionare una lista contenente tutti i prodotti disponibili, visualizzando le caratteristiche di ognuno. Questi prodotti potranno poi essere inseriti in un carrello, in quantità a discrezione dell'utente. Dovrà essere data la possibilità all'utente di visionare il carrello con i prodotti da lui scelti, le quantità, i prezzi ed il totale, comprensivo di eventuali sconti. L'utente potrà poi scegliere se rimuovere degli oggetti dal carrello o confermare l'ordine.

Se l'ordine viene confermato, deve essere inviata una mail al personale dedicato dell'azienda con all'interno i dati dell'ordine, comprensivi di dati del cliente e dei vari prodotti che compongono l'ordine. L'utente inoltre dovrà avere a disposizione una pagina dove poter visionare tutti i suoi ordini, e una pagina dove poter modificare i suoi dati personali in caso cambi l'utilizzatore di un determinato account.

L'ultima tipologia di Utente è il Guest. Dovrà infatti essere predisposto l'accesso al portale di eventuali utenti guest. Questi utenti non registrati, potranno visionare l'elenco di tutti i prodotti, senza i prezzi, e potranno da questo elenco comporre un carrello dedicato. In questo carrello, però, non dovranno figurare i prezzi. Il carrello servirà solo a permettere all'utente di richiedere un preventivo per tali oggetti. La richiesta deve pervenire via mail al personale dedicato di Urbitek.

Il guest, prima di poter richiedere un preventivo, dovrà inserire una serie di informazioni che lo possano identificare (informazioni che andranno inserite nella mail di richiesta del preventivo).

0.3 Design

L'applicazione è stata sviluppata seguendo le linee guida della metodologia AGILE. Tale metodologia, in contrapposizione con quella Waterfall, permette di focalizzarsi sulla consegna al committente di un'applicazione funzionante nel più breve tempo possibile, tramite un approccio meno strutturato e rigido. Nella prima iterazione, il focus è stato centrato sul design della parte grafica essenziale dell'applicazione.

A questa fase hanno partecipato attivamente anche alcuni amministratori dell'applicazione e alcuni utenti finali. Il design iniziale prevedeva una pagina di login che contenesse la possibilità di far inserire all'utente username e password, insieme ad un pulsante che permetteva l'accesso degli utenti guest. E' emerso dagli utenti che sarebbe stato più comodo avere la form di login nascosta e visualizzabile alla pressione di un pulsante.

Il secondo aspetto grafico che è stato consigliato dagli utenti è l'aggiunta di una barra di navigazione posta sulla parte superiore della finestra, con pulsanti che contengono i vari collegamenti per le pagine dell'applicazione.

Quindi ho provveduto a realizzare le varie pagine dell'applicazione, collegandole tra loro e realizzando l'applicazione server che potesse effettuare il delivery delle pagine. A questo punto avevo la pagina di login funzionante (senza ancora nessun meccanismo per il login), e le varie pagine dell'applicazione che, seppur vuote, potevano già permettere agli utenti di navigare nel sito.

Ora, ho sottoposto agli utenti il sito per raccogliere feedback sul funzionamento e per capire se ci potessero essere modifiche opportune. Una modifica, ad esempio, ha riguardato il colore del tasto "ESCI", che era inizialmente colorato come gli altri, ma che ho notato portasse confusione e non venisse identificato correttamente.

In seguito, ho realizzato il database e ho cominciato a popolare le pagine con dell'effettivo contenuto. Ho realizzato innanzitutto la parte dedicata agli utenti guest, in quanto ad essi sono disponibili funzioni limitate, per poi passare agli utenti normali ed infine agli amministratori. Ho scelto di effettuare un iterazione del processo AGILE per ogni tipologia di utente sviluppata.

Il goal principale dell'applicazione era quello di avere un'interfaccia semplice e chiara per l'utente. Per questo motivo, ho scelto di adottare una tipologia di design partecipativo, in quanto ho cercato di coinvolgere utenti di varie tipologie e capacità con l'obiettivo di realizzare un'interfaccia che potesse soddisfare il bacino più ampio possibile di utenti. Per quanto riguarda gli amministratori, ho sottoposto loro l'applicazione ad ogni iterazione dell'approccio AGILE, tramite l'utilizzo di Focus Group. Questi sono stati svolti per via telematica a causa dell'emergenza sanitaria in corso al momento dello sviluppo, ma comunque si sono rivelati efficaci e il dialogo con gli utenti è sempre stato produttivo.

Per quanto riguarda gli utenti standard e guest, ho utilizzato l'Experience Prototyping. Ho cercato di coinvolgere utenti di età diverse (da un minimo di 28 anni ad un massimo di 65) e di capacità diverse (dagli utilizzatori abituali di computer a persone poco avvezze alla navigazione sul web).

Per realizzare un'applicazione che fosse il più possibile user friendly, l'intera

applicazione è stata sviluppata seguendo il principio del mobile first. Ogni interfaccia, cioè, è stata pensata prima su mobile e poi studiata per adattarsi ad altri dispositivi come tablet e computer. L'interfaccia è responsiva e si adatta al supporto sul quale viene visualizzata.

Il principio alla base dello stile minimalista dell'interfaccia è quello del Less is More. Ho preferito mantenere una grafica minimalista per non distrarre l'utente e fornirgli solo le informazioni necessarie, senza elementi che potrebbero confondere gli utenti e rendere l'utilizzo dell'applicazione complicato. Infine, sono stati eseguiti test di funzionamento dell'intera applicazione seguendo anche il principio dello Think Aloud. Durante questi test, soprattutto con gli amministratori, sono emersi problemi di funzionamento dell'applicazione, come ad esempio quello di controllare l'esistenza di un oggetto nel database prima di inserirlo.

0.4 Tecnologie

Per la realizzazione dell'applicazione, ho scelto di adottare lo stack MEVN. Questo si compone di Mongo DB [2] come database, un database non relazionale semplice e veloce, Express [3], framework Javascript per la programmazione lato server, Vue [6], framework Javascript per la programmazione lato client, e Node JS [1], per la realizzazione del server.

La peculiarità di questo solution stack è che è completamente open source e non è legato a nessun sistema operativo.

Come pattern di design, ho utilizzato MVVM, che si compone di Model, View e ViewModel. Il Model è la parte dell'applicazione che si occupa di gestire i dati, la View è la parte dell'applicazione che mostra i dati all'utente e riceve i suoi input, mentre il ViewModel agisce da "collante" tra la View e il Model, ricevendo i dati dal Model e modellandoli per presentarli alla View. Il ViewModel, inoltre, agisce anche nel senso opposto, catturando gli input dell'utente e definendo poi operazioni che andranno a cambiare i dati del Model e, se necessario, aggiornare la View.

Inoltre, ho utilizzato le Socket per implementare un servizio di notifiche real time che comunicano all'utente la creazione di un nuovo prodotto e la modifica o la cancellazione di un prodotto esistente. Così facendo l'utente viene messo al corrente in tempo reale dei cambiamenti che avvengono al catalogo dei prodotti. Tali modifiche ovviamente si propagano sul carrello e sulla lista dei desideri degli utenti.

Un'altra tecnologia utilizzata è quella dei cookies, tramite i quali viene gestita la sessione degli utenti. Quando si accede alla pagina di login, viene creato un cookie all'interno del browser. Quando un utente esegue il login, viene cambiato lo stato del cookie in modo tale da permettergli di navigare tra le pagine a lui dedicate. In questo modo, non è necessario effettuare il login ogni volta che si chiude il browser, garantendo una maggiore comodità, e inoltre viene aumentata le sicurezza dell'applicazione facendo sì che solo chi abbia effettuato correttamente il login possa accedere alle pagine a lui dedicate.

A proposito di sicurezza, viene anche utilizzato un plugin che permette di effettuare un hash delle password, in modo che queste siano salvate all'interno del database criptate. Quando viene creato l'utente, la password viene hashata e poi salvata nel database. Quando l'utente esegue il login, la password immessa viene hashata e poi confrontata con quella presente nel database. Se questa è corretta, viene garantito il login.

L'ultima tecnologia utilizzata è un plugin di GMAIL che permette di costruire una mail e, fornendo un account dal quale inviare le mail, tale plugin consente di spedire i messaggi costruiti in precedenza.

0.5 Codice

Gli aspetti considerevoli del codice a mio avviso sono quelli legati a Vue [6]. Vue [6] è un framework Javascript che permette di gestire molti elementi legati agli aspetti di presentazione a lato client e non a lato server, così da sgravare il server da diverse operazioni. Vue [6] implementa il modello MVVM, concentrandosi sul ViewModel, fornendo un "collante" tra View e Model. Ho utilizzato il data binding fornito da vue [6] per collegare aspetti presentazionali a delle variabili.

Ho utilizzato le direttive di vue [6], come v-bind, insieme alle direttive con parametro come v-on. Ho utilizzato inoltre le computed properties, il rendering condizionale e i componenti.

Molto comodo infatti è stato creare un componente per la barra di navigazione e poterlo riutilizzare nel codice senza ripetizioni.

Infine, è stato molto comodo gestire le form con vue [6], per catturare l'input dell'utente e poter eseguire funzioni personalizzate.

0.6 Test

Non sono stati svolti test automatici. Sono stati effettuati solo test con utenti reali. Sono stati effettuati test ad ogni iterazione dell'approccio AGILE. Inizialmente sono state testate tutte le interfacce di base e il passaggio tra le pagine, con utenti amministratori, insieme, e con gli utenti guest e standard, singolarmente. Poi sono state eseguite e testate le funzionalità legate ai guest. Queste sono state testate con utenti singoli e anche con multipli utenti connessi allo stesso tempo. Successivamente sono state realizzate e testate le funzionalità legate agli utenti standard. Sono state testate tutte le funzionalità e infine, sono state realizzate le funzionalità legate agli amministratori.

Sono stati eseguiti test dagli utenti per quanto riguarda anche tutte le funzionalità che riguardano il database. Infine, sono state testate le notifiche, e l'invio delle mail.

Ho utilizzato la piattaforma NGROK [4] per poter mettere online il server Node.js [1] in modo da poter eseguire i test con utenti remoti.

L'applicazione è stata testata sui browser Chrome, Firefox, Safari e Microsoft Edge su PC, e sugli smartphone android e IOS.

0.7 Deployment

Innanzitutto, scaricare la cartella contenente il progetto dal sito https://github .com/Sfigaman/ASW-2020-CAMPORESI-URBITEK-ECOMMERCE.

Per quanto riguarda sistemi Windows e Mac, occorre innanzitutto scaricare ed installare Node JS dal sito https://nodejs.org/en/. In seguito, per controllare l'avvenuta installazione, eseguire il comando node -v. Quindi, occorre scaricare ed installare Mongo DB dal sito https://www.mongodb.com/download-center/community. Una volta fatto ciò, importare le due collection dai file .bson forniti con i comandi mongorestore –collection products –db test "path to products.bson" e mongorestore –collection users –db test "path to users.bson". Fatto ciò, è possibile far partire l'applicazione spostandosi nella cartella principale, contenente il file app.js, tramite il comando node app.js. A questo punto è possibile navigare il sito partendo dalla home raggiungibile all'indirizzo localhost:3000.

Da questo punto è possibile navigare il sito come utente guest tramite l'apposito pulsante, accedere come utente standard immettendo al login user e user come username e password, oppure accedere come amministratore immettendo admin e admin come username e password. A questo punto è possibile creare, modificare e cancellare liberamente prodotti e utenti, così come effettuare ordini e mettere oggetti nel carrello e nella lista dei desideri.

Da notare che se si immette un indirizzo mail valido, verrà inviata una mail di conferma a tale indirizzo se viene correttamente eseguito un ordine o una richiesta di preventivo.

0.8 Conclusioni

Sono fiero di poter dire che tutti i requisiti sono stati rispettati e che tutte le funzionalità pensate all'inizio sono state implementate. Nonostante l'emergenza sanitaria in corso, ho notato come, grazie all'utilizzo dei moderni strumenti di comunicazione, si riesca comunque a coinvolgere gli utenti nel progetto, e che questo coinvolgimento porti sempre benefici. Mi sono trovato spesso a non pensare assolutamente a determinati casi di utilizzo o situazioni, che invece agli utenti sono venute subito in mente. In questo modo credo che l'applicazione abbia raggiunto l'obbiettivo iniziale di essere il più user friendly possibile, e di adattarsi all'utilizzo di un ampio bacino di persone, con differenti livelli di abilità nell'ambito tecnologico. In definitiva, credo che sia importantissimo mettere al centro gli utenti e cercare di realizzare applicazioni per l'utente, o addirittura con l'utente come è avvenuto in questo caso, mettendo la sua comodità e le sue esigenze prima di tutto il resto.

Dal punto di vista tecnico, non conoscevo lo stack MEVN ma mi sono trovato molto bene nell'utilizzarlo, sebbene fosse la prima volta per me, e credo che sia uno stack davvero molto efficace. In primo luogo, per il fatto di non essere legato ad una piattaforma specifica. Ho sviluppato il progetto in parte su Mac e in parte su Windows, senza incontrare alcuna difficoltà. In secondo luogo, trovo che

Mongo DB [2] sia una piattaforma davvero performante e semplice da utilizzare, soprattutto se confrontata con i classici database relazionali. Inoltre, Vue [6] ha facilitato notevolmente tutto lo sviluppo delle funzionalità dell'applicazione, permettendo di gestire la presentazione dei dati provenienti dal database in modo preciso, semplice, veloce e flessibile.

Infine, la possibilità di realizzare uno strumento da inserire realmente nel sito dell'azienda per cui lavoro, la possibilità di farlo testare ai miei colleghi e a clienti reali, ha impreziosito questa esperienza.

Ho pensato anche ad alcuni possibili aggiornamenti futuri del sito. In particolare un aggiornamento utile potrebbe essere quello di cambiare il layout della pagina, in modo da descrivere maggiormente le caratteristiche e i dettagli dei prodotti in catalogo. Si potrebbe aggiungere un pulsante per aprire una finestra in cui visualizzare maggiori dettagli, sia caratteristiche tecniche che foto, del prodotto. Per certi prodotti disponibili in varie tipologie (diversi formati, colorazioni o altro) invece del pulsante "Aggiungi al carrello", si potrebbe direttamente inserire un pulsante "Dettagli" che apra una finestra in cui visualizzare le diverse scelte possibili relative al prodotto. Una volta selezionati i dettagli del prodotto scelto si potrebbe aggiungere al carrello direttamente da quella finestra, avendo quindi già selezionato le preferenze.

Bibliography

[1] OpenJS Foundation. The node.js docs, 2020.

[2] MongoDB Inc. The mongodb 4.2 manual, 2008.

[3] StrongLoop Inc. The express manual, 2020.

[4] inconshreveable. About ngrok, 2020.

[5] Urbitek SRL. Urban solutions, 2020.

[6] Evan You. The vue manual, 2020.