Vita Bucolica

Applicazioni e Servizi Web

Andrea La Rosa - 0000951374 {andrea.larosa2@studio.unibo.it}

23 Maggio 2022

1 Introduzione

Il progetto nasce dall'esigenza di un gruppo di divulgatori di potere pubblicare informazioni di diverso tipo riguardanti l'ambito agrario. L'obiettivo è quello di creare una web application che consenta una semplice creazione di articoli corredati di foto e/o video, da parte di "utenti esperti" e la consultazione dei medesimi da parte degli utenti base. La web application si compone di una interfaccia attraverso cui l'utente esperto può inserire i nuovi articoli e al contempo visualizzare gli articoli precedenti. La medesima interfaccia sarà a disposizione degli utenti base per la visualizzazione degli articoli e la loro ricerca, che potrà essere effettuata tramite tag o search word.

2 Requisiti

L'utente rappresenta il punto centrale sul quale è stata basata sia la progettazione che il successivo design, in accordo con le moderne tecniche che valorizzino la User Experience. I requisiti sono stati raccolti intervistando gli utenti esperti, che hanno commissionato la web application, dopodiché data la difficoltà nell'intervistare dei potenziali utilizzatori, sono stati definiti, tramite la tecnica di Personas e Scenarios, alcuni modelli che rappresentassero un possibile gruppo di utenti di riferimento per l'applicazione. A ciascuna personas è stato assegnato uno scenario preciso che ricostruisse in maniera dettagliata una particolare situazione d'uso ipotetica. Riportiamo di seguito le personas individuate con i loro scenarios.

- Tomas [Reale] Agricoltore, commerciante diretto e divulgatore, ha necessità di pubblicare informazioni riguardo i diversi aspetti del comparto agrario, in particolare la meccanizzazione e le coltivazioni. Ha necessità di pubblicare sul sito i video che pubblica sul suo canale YouTube "Vita-Bucolica".

- Carlo [Reale] Responsabile contenuti e gestore dei social di Vita Bucolica, è appassionato di avicoltura e possiede un piccolo allevamento di razze avicole locali. Ha necessità di potere scrivere approfondimenti riguardanti gli aspetti dell'avicoltura.

- Michela [Reale] Addetta alle vendite dello spaccio aziendale "Boutique Contadina", ha interesse a promuovere i prodotti di stagione.

- Maria [Personas] Utente Esperto che vuole contribuire alla divulgazione della storia dell'agricoltura mediante articoli in stile giornalistico, corredandoli di link ad eventuali video su vimeo.

- Gianni [Personas] Utente base che vuole tenersi informato sulle curiosità agrarie e sente la necessità di esprimere un giudizio su ciò che legge. Gradisce in particolare la possibilità di non dovere cercare su Youtube i video di "Vita Bucolica" ma di visualizzarli direttamente sul sito dal suo smartphone.

- Lucia [Personas] Utente base che coltiva un piccolo orto, vuole tenersi informata sulle curiosità agrarie riguardanti in particolare la frutta e la verdura di stagione, è una utente discontinua quindi ha necessità di potere vedere subito quali sono i nuovi articoli degli argomenti di suo interesse. Allo stesso modo vuole potere spulciare tra tutti gli articoli per cercare curiosità.

Dall'analisi dei potenziali utenti così definiti sono stati individuati i seguenti requisiti.

2.1 Requisiti Funzionali

Il sistema si occupa di gestire gli aspetti di pubblicazione e fruizione di articoli divulgativi. Esiste una separazione tra chi pubblica e fruisce e chi può solo visualizzare gli articoli. Dall'analisi degli scenari d'uso possibili è emerso che le funzioni dell'utente di base sono e devono essere disponibili anche all'utente esperto divulgatore. Un utente che non sia registrato potrà solo visualizzare la pagina principale con gli ultimi articoli disponibili ma non potranno avere ulteriori interazioni.

Un utente non registrato può:

- Navigare la pagina principale del sito
- Accedere alla registrazione come utente di base o utente esperto

Un utente base registrato può:

- Visualizzare gli articoli
- Filtrare gli articoli mediante tag
- Cercare gli articoli mediante termine di ricerca
- Mettere e togliere un "like" ad un articolo pubblicato
- Aggiornare i parametri utente
- Aggiornare la password

Un utente esperto può:

- Effettuare tutte le operazioni consentite all'utente base
- Pubblicare un articolo
- Modificare un articolo da lui pubblicato
- Cancellare un articolo da lui pubblicato

Altri requisiti:

- Gli utenti loggati al sito vengono avvisati della pubblicazioni di un articolo inerente il loro argomento di interesse mediante una notifica
- Fornire agli utenti, sotto forma di notifiche, tutti gli articoli che si sono persi tra un login e il successivo

2.2 Requisiti Non Funzionali

Il requisito non funzionale principale della applicazione web è la scelta di progettazione Mobile First. L'approccio applicato al web design è pensato principalmente per l'ottimizzazione dell'applicazione su dispositivi mobili. Questo requisito è stato individuato dall'analisi degli scenari d'uso. Si cerca di favorire l'utenza di base, non esperta, per la forte caratterizzazione divulgativa della applicazione. Si è ipotizzato che gli utenti ne facciano uso anche durante gli acquisti di frutta e verdura o durante le proprie sessioni hobbistiche, per esempio durante la coltivazione dell'orto o la potatura di piante ornamentali.

3 Design

La progettazione dell'intero sistema si focalizza sul modello iterativo basato sullo User Centered Design, il cui focus riguarda utenti reali e "virtualizzati". Come introdotto nel paragrafo precedente, sono state individuate sei Personas, con annessi Scenarios, per capire a quali funzionalità l'applicazione dovesse rispondere. Avendo il progetto un unico sviluppatore si è comunque potuto usufruire di alcune metodologie AGILE durante lo sviluppo, in particolare:

- Sviluppando in piccoli sprint
- Contattando spesso il cliente, in questo caso contattando Tomas, Carlo e Giulia i tre soggetti interessati.

3.1 Design delle interfacce

Si è cercato di aderire ai principi KISS e "Less Is More" con l'obiettivo di realizzare una interfaccia semplice ed intuitiva nell'utilizzo da parte degli utenti. Infine, per rendere le interfacce utilizzabili da qualsiasi dispositivo e migliorare la User Experience, sono stati applicati i principi che stanno alla base del responsive design. Si sono analizzati, insieme ai committenti, dei wireframe e, dopo un primo sviluppo, si è effettuato un test di usabilità con l'utenza e sono state apportate le modifiche ricavate dai suggerimenti ricevuti dai clienti.

3.2 Design Architetturale

Il sistema si compone di un servizio back-end che mantiene i dati in modo persistente e gestisce le logiche di business e, un'interfaccia front-end per la visualizzazione e interazione lato utente. La comunicazione e l'accesso ai dati persistenti sono affidate ad un back-end mediante API Restful, i vantaggi che ne derivano sono:

- Indipendenza dalle tecnologie adoperate
- Indipendenza tra i dati persistenti e i modelli di comunicazione
- Comunicazione client-server stateless
- Accesso alle risorse mediante identificazione

Ṕer rispondere alle richieste del sistema, è stato necessario implementare la gestione delle notifiche sia a back-end che a front-end, mediante la libreria socket.io(4).

3.3 Accessibilità: Scelta dei colori per Daltonici

Per migliorare l'esperienza degli utenti affetti da una qualsiasi forma di daltonismo si è effettuata una piccola ricerca sui colori di selezione dei pulsanti per i tag, e, dopo diverse prove si è selezionato l'abbinamento di colore arancione/giallo e rosso che maggiormente poteva essere identificabile per l'utenza citata. I colori scelti sono poi stati adottati in tutto il progetto. Per effettuare il confronto ci si è avvalsi del sito Coblis, che mette a disposizione gli strumenti per sperimentare come risulta una immagine ad un daltonico.

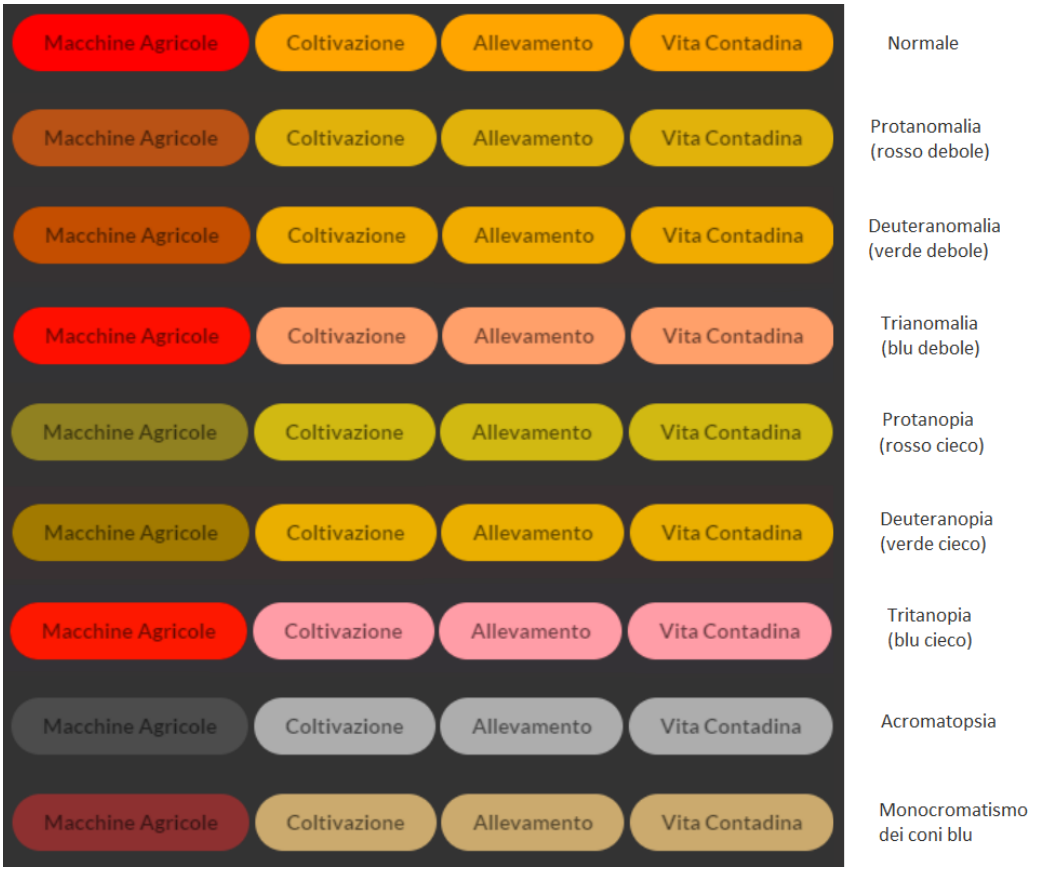

Figure 1: comparazione tra le diverse forme di daltonismo per i colori scelti. Si può notare che il colore scelto per il pulsante in selezione è distinguibile più o meno marcatamente per tutti i soggetti

4 Tecnologie

Lo stack utilizzato è MERN scelto tra i moderni stack di tecnologie Javascript-based per la realizzazione di Single Page Applications. Il sistema si compone quindi di un servizio di back-end basato su un server in Node.js con Express.js come framework di gestione delle richieste e un layer di persistenza implementato da un database non relazionale con motore MongoDB. Il sistema è poi reso disponibile agli utenti tramite una interfaccia sviluppata tramite ReactJS (2) nella sua accezione Typescript(5).

4.1 Front-end

Sviluppo software

React JS rende la creazione di UI interattive facile e indolore. Attraverso la progettazione di interfacce per ogni stato della applicazione, React JS consente ad ogni modifica l'aggiornamento efficiente solamente delle parti della UI che dipendono da tali dati. La natura dichiarativa dell'UI rende il codice prevedibile e facile da debuggare. I componenti creati sono isolati e componibili per creare UI complesse.

Le interazioni e logica per i componenti sono implementate in JavaScript, si possono facilmente passare, ed accedere, strutture dati complesse in vari punti della applicazione senza dover salvare informazioni sul DOM.

Si è scelto di utilizzare TypeScript (che aggiunge sintassi a JavaScript) per la tipizzazione delle variabili, questo permette di catturare gli errori a tempo di editing.

Il codice TypeScript viene convertito in JavaScript, che viene eseguito ovunque venga eseguito JavaScript. TypeScript è uno strumento efficace che utilizza l'inferenza del tipo senza appesantire troppo il codice.

Stile

Parte delle componenti di base sono state importate da Semantic Ui React. L'interfaccia utente semantica tratta le parole e le classi come concetti scambiabili. Le classi utilizzano la sintassi dei linguaggi naturali come le relazioni nome/modificatore, l'ordine delle parole e la pluralità per collegare i concetti in modo intuitivo. Questo ha semplificato e accorciato notevolmente i tempi di sviluppo. Laddove erano necessarie modifiche all'appeal grafico portato da Semantic si sono utilizzati fogli di stile in SCSS. In particolare si sono aggiunte declinazioni per il mobile first.

4.2 Gestione dei dati e dello stato

Per la gestione dei dati e dello stato si è utilizzato Redux(3). Redux è un contenitore di stati deterministici per le applicazioni JavaScript, nato per risolvere l'annoso problema della gestione dello stato dell'applicazione. Lo stato di un'applicazione è l'insieme delle condizioni interne in uno specifico istante che determinano il risultato delle interazioni con l'esterno, è l'insieme delle informazioni che determinano l'output in corrispondenza di un dato input in uno specifico istante. Nello sviluppo di applicazioni abbiamo sempre a che fare con la gestione dello stato e proprio questa gestione è uno dei punti più critici responsabili di buona parte della complessità del software. Le Single Page Application si trovano a dover gestire:

- Dati provenienti dal server e memorizzati in una cache locale;
- Dati generati dall'applicazione stessa che devono essere inviati al server;
- Dati che devono rimanere in locale per rappresentare, ad esempio, la situazione corrente dell'interfaccia utente o le preferenze espresse dall'utente.

La gestione di questi dati è complessa in quanto possono variare per cause diverse: interazioni con l'utente, interazioni con il server, modifiche scatenate da altre modifiche e così via. I principi alla base di Redux sono:

- Esiste una singola fonte di verità: lo stato dell'intera applicazione è memorizzato in un unico oggetto
- Lo stato è in sola lettura: non è possibile modificare direttamente le informazioni memorizzate nello stato dell'applicazione; l'unico modo per farlo è tramite azioni esplicite
- Le modifiche allo stato vanno fatte con funzioni pure: lo stato corrente viene sostituito da un nuovo stato generato da funzioni esclusivamente in base ad una azione ed allo stato precedente

Il rispetto di questi principi consente di ottenere dei benefici come ad esempio:

- La predicibilità delle transizioni di stato
- L'organizzazione del codice è obbligata a seguire uno specifico pattern, il che crea uno standard di codifica

- É possibile definire uno stato iniziale a piacere e far partire l'applicazione da quello stato
- É possibile tenere traccia delle transizioni di stato
- Dal momento che le modifiche allo stato sono effettuate tramite funzioni pure, risulta più semplice scrivere unit test per le applicazioni

4.3 Session Storage

Le informazioni riguardanti lo user e alcune informazioni di utilità sono state salvate come cookie dentro l'application storage.

4.4 Back-end

Il back-end è stato realizzato con Node e Express per realizzare la business logic lato server e MongoDB per il layer di persistenza.

Persistenza

La persistenza dei dati è stata ottenuta tramite un database non relazionale. Per l'interfaccia al database si è utilizzata la libreria Mongoose(1). Si tratta di una delle librerie più utilizzate che consente di creare in codice una rappresentazione dello schema dei dati che rispecchia quello che deve essere mantenuto nel database. É possibile definire lo schema dei dati tramite Mongoose in modo da poter creare le collezioni quando necessario senza ulteriori setup.

Autenticazione e Autorizzazione

Per la trasmissione delle informazioni tra client e server, al login viene rilasciato dal server un Json Web Token (JWT) per garantire l'identità dell'utente loggato. Qualsiasi richiesta successiva avrà in allegato lo stesso JWT generato in partenza così che l'utente possa accedere a qualsiasi risorsa dell'applicazione consentita da quel preciso token. I token vengono salvati sul browser del client in modo da poter essere recuperati e garantire di nuovo l'accesso all'utente fino alla scadenza dei token stessi.

Comunicazione Real-Time

Per l'implementazione delle funzionalità real-time quali l'invio delle notifiche al sistema relative alla creazione di nuovi articoli, si è deciso di fare uso della libreria socket.io presentata a lezione.

5 Codice

Dopo alcune prime difficoltà incontrate nella curva di apprendimento di ReactJS nella sua accezione Typescript, lo sviluppo ha proceduto in modo abbastanza spedito. Le maggiori battute di arresto si sono avute nell'implementazione di redux lato front-end e nell'integrazione delle notifiche mediante socket.io sia back-end che front-end. Per la gestione di autenticazione e autorizzazione delle richieste, che come detto in precedenza è basata su Json Web Token, aggiunto nella pipeline delle richieste creata mediante Axios. Per quanto riguarda il front-end, la gestione dello stato globale è stata affidata a Redux. Mediante questo tool sono state memorizzate informazioni da rendere disponibili alle componenti. Il token di autenticazione è stato salvato tuttavia nel local storage sottoforma di cookie insieme ad altri dati utili come lo user e le notifiche.

6 Test

Il sistema è stato testato su Browser Firefox, Chrome e Microsoft Edge. Riguardo alla visualizzazione mobile si è effettuata una prima scrematura dei bug dopo un primo release per evitare i bug grafici più lampanti. I test per le funzionalità principali, sia come user base che come user esperto, sono stati effettuati dal team di sviluppo supportato dal gruppo di Vita bucolica. Lato server le API sono state testate durante lo sviluppo sia mediante casi di uso sia avvalendosi di Postman, questo ha permesso di eliminare i bug più grossolani prima dell'integrazione con il front-end.

6.1 User Experience

Come già indicato oltre a dei test preliminari, pre-release, sono stati effettuati dei test con il Team di Vita Bucolica e con alcuni utenti istruiti sulle prove utili da fare, si sono quindi apportate le modifiche venute a galla dopo il primo test.

7 Usability Test

I risultati del test (consultabili al seguente link) hanno portato alla luce alcune problematiche legate alla intuitività di utilizzo, e a qualche bug grafico, in particolare, i suggerimenti diretti (già reinterpretati):

- "Non ho capito dove si modifica il profilo" - L'icona del profilo, indicata utilizzando la prima lettera del nome, non era intuitivamente chiara a tutti i partecipanti
- "Controlla il pulsante in alto a destra" - Il pulsante di ingresso ed uscita usciva dal bordo dello schermo di alcuni modelli mobile
- "Il bottone di ingresso e uscita non è chiaro" - L'icona di login/logout non era chiara ad alcuni partecipanti

Sono state quindi apportate le modifiche e sistemati gli eventuali bug di sistema trovati durante la fase di test. L'icona utente è stata modificata per essere una icona rappresentante una persona, l'icona in alto a destra è stata migliorata nel foglio di stile per rispondere alle esigenze di altri devices mobile, l'icona di ingresso e uscita sono state scelte in base a quelle maggiormente utilizzate per la medesima funzione.

8 Deployment

L'applicazione in download è creata per agire in locale.
É necessario avere installati sulla macchina:

- Docker 20.10.10 o superiore
- Mongo.DB 4.4.2 o superiore

Per avviare il progetto è necessario:

- Clonare il repository git clone https://github.com/AndreaLaRosa1983/vita-bucolica.git.
- Posizionarsi nella cartella ~\vita-bucolica\vita-bucolica, creare i container mediante il comando "docker-compose build" (tempo medio 1 minuto)
- Avviare il progetto mediante il comando "docker-compose up"(tempo medio 2 minuti)
- Da browser entrare in localhost:3000
- Cliccare il bottone accedi in alto a destra
- Inserire le seguenti credenziali :
  - Per accedere come creator
    - email: web@master.com
    - password: Web1234!!!
  - Per accedere come utente base
    - email: basic@user.com
    - password: Web1234!!!

9 Conclusioni

Sono abbastanza soddisfatto del risultato finale, nonostante non abbia portato a termine tutti gli obiettivi prefissati. Il prodotto finale è di semplice utilizzo e in grado di soddisfare tutti i requisiti principali ed ha dimostrato di sapere garantire una piacevole esperienza d'uso agli utenti, sia dal punto di vista delle funzionalità che dell'appeal. Non è da escludere che in futuro si vada ad implementare la funzionalità di commento ai vari post e la possibilità di iscriversi ad una mailing list che notifichi la pubblicazione di un nuovo articolo.

10 Commenti finali

Per quanto il progetto sia abbastanza basilare il doversi occupare interamente di tutto lo stack è stato particolarmente impegnativo e, se da un lato è stato molto formativo, dall'altro ha portato a diversi momenti di frustrazione. Il contatto diretto con i committenti e il loro coinvolgimento nelle fasi di progettazione e di testing ha reso più "leggero" compiere certe scelte.

Bibliography

[1] Mongoose. URL: https://mongoosejs.com/.

[2] Reactjs. URL: https://it.reactjs.org/.

[3] Redux. URL: https://redux.js.org/.

[4] Socket.io. URL: https://socket.io/.

[5] Typescript. URL: https://www.typescriptlang.org/.