Beach Management Platform

Applicazioni e Servizi Web

Daniele Commodaro
daniele.commodaro@studio.unibo.it
0000976618

30 Maggio 2022

1 Introduzione

Il progetto è un'applicazione web-based che nasce dall'idea di una piattaforma per la gestione degli stabilimenti balneari, che offra anche la possibilità di monitorare alcuni agenti atmosferici e di segnalare con degli alert quando i livelli di sicurezza previsti vengono superati. L'obiettivo è quello di ottenere i dati meteorologici da alcuni servizi terzi e di presentarli all'utente assieme alle schede dei vari stabilimenti balneari censiti sulla piattaforma. La web-application si compone quindi di un'interfaccia che presenta agli utenti i dati ottenuti in maniera semplice, questi sono relativi al giorno corrente e possono fare riferimento alla posizione dell'utente oppure ad un'altra città da lui scelta. Oltre a visualizzare le schede degli stabilimenti balneari presenti si offre la possibilità di poterle eliminare.

2 Requisiti

Prima di realizzare l'applicazione è stato necessario ottenere un elenco di requisiti che rispecchiassero gli interessi dei potenziali utenti finali. Progettazione e design sono stati basati sull'utente in accordo con le moderne tecniche per la valorizzazione della user experience. Attraverso la tecnica di Personas e Scenarios, sono stati definiti dei modelli per rappresentare un possibile gruppo di utenti come riferimento per l'applicazione. A ciascuna persona è stato assegnato uno scenario preciso che ricostruisse in maniera dettagliata una particolare situazione d'uso ipotetica. In seguito vengono riportate le personas individuate con i loro scenarios.

- **Adam**
  É un giovane surfista di 26 anni, lavora part-time in un negozio di abbigliamento. Quando esce a fare surf utilizza sempre delle app per controllare il vento e le condizioni atmosferiche.

- **Carla**
  É una nonna pensionata di 72 anni che ogni anno trascorre qualche settimana di vacanza al mare assieme a tutta la sua famiglia. Da poco Carla ha sostituito il suo vecchio telefono con uno smartphone che le è stato regalato dai suoi figli, e quest'anno sotto loro consiglio ha deciso che proverà a usarlo per prendersi cura dei suoi nipotini mentre sarà in spiaggia. La famiglia è preoccupata per quanto riguarda l'esposizione dei bambini al sole quando vengono lasciati soli con la nonna, e gradirebbero avere pertanto un'app che possa avvertirla quando la radiazione solare diventa troppo elevata.

- **Matteo**
  Matteo é un informatico di 33 anni che ogni anno durante l'estate va in vacanza al mare insieme ai suoi genitori. Quando sceglie un nuovo posto in cui andare questa famiglia si accerta sempre che sia possibile l'accesso alle persone disabili perché Matteo é un portatore di handicap.

- **Eros**
  Ha 46 anni ed è sempre stato un grande amante degli animali, da qualche anno ha deciso di fare della sua passione un lavoro e insieme alla sua compagna ha aperto un nuovo locale pet friendly sulla spiaggia. Purtroppo peró Eros non é ancora riuscito a creare un buon giro di clientela e si ritrova spesso a cercare nuove piattaforme o app su cui esporre il locale, in modo da farsi conoscere.

- **Chiara**
  Ha 26 anni ed é laureata in scienze dell'educazione. Ha da sempre avuto una forte passione per i bambini e lavora come educatrice presso un centro estivo durante l'estate. Chiara capisce molto bene la responsabilità del suo lavoro e ci tiene a farlo nel migliore dei modi. Il centro estivo organizza delle gite in spiaggia per i bambini durante la stagione, quest'anno Chiara ha deciso di affidarsi ad un'applicazione per tenere sotto controllo vento e radiazione uv in modo da capire quando dover prendere delle precauzioni.

A seguire vengono mostrati i requisiti individuati dopo l'analisi dei potenziali utenti.

2.1 Requisiti dell'utente

I requisiti percepiti dall'utente sono:

- riconoscimento della città in cui si utilizza l'applicazione
- inserimento di una città per cui si desidera ottenere i dati meteo
- visualizzazione dei dati meteorologici del giorno corrente. Questi possono essere relativi alla città in cui l'utente si trova, oppure ad una città diversa se viene inserita. I dati meteo che vengono mostrati sono:
  - periodo del giorno
  - temperatura attuale, minima, massima e percepita
  - città di riferimento
  - descrizione qualitativa dello stato del cielo, ad esempio "cielo sereno" oppure "nubi sparse"
  - quantitá di pioggia
  - stima dell'umidità rilevata
  - stima della quantitá di nuvole presenti in cielo
  - velocitá del vento
  - indice UV rilevato
- ricezione di alert che descrivono l'intensitá dei parametri di vento e radiazione ultravioletta

- ottenimento di notifiche quando vengono compiute delle azioni nell'applicazione
- visualizzazione degli stabilimenti balneari censiti sulla piattaforma
- eliminazione degli stabilimenti censiti

2.2 Requisiti funzionali

- geo-localizzazione dell'utente
- integrazione con i servizi OpenWeatherMap e WeatherBit per l'ottenimento dei dati meteorologici aggiornati all'ultima mezz'ora trascorsa
- temperature espresse in gradi Celsius
- la quantitá di pioggia viene espressa in millimetri su ore ed é associata ad una descrizione qualitativa per esempio "leggera pioggia"
- generazione di alert per comunicare all'utente come l'indice UV rilevato viene classificato rispetto all'indice universale della radiazione UV solare(da 0 a 11+)
- generazione di alert per comunicare come il valore del vento rilevato viene classificato rispetto alla scala di Beaufort, una misura empirica utilizzata per misurare la forza del vento
- creazione di notifiche verso l'utente quando le azioni che vengono compiute hanno successo, in situazioni dove si necessita di un avvertimento oppure quando si verifica un errore
- registrazione sulla piattaforma degli stabilimenti balneari
- cancellazione degli stabilimenti balneari
- integrazione con il servizio Angular Google Maps[3] per disegnare la mappa del locale in base alle sue coordinate

2.3 Requisiti non funzionali

I requisiti non funzionali del sistema sono:

- l'applicazione deve risultare semplice da usare anche per gli utenti non esperti
- l'interfaccia dev'essere responsiva e avere un aspetto visibilmente gradevole
- il sistema dev'essere progettato in maniera tale da poter essere facilmente estendibile a nuove funzionalitá

3 Design

Durante la fase di realizzazione del progetto é stato adottato il modello di design User Centerd Design con utenti virtualizzati. Sono stati individuati gli utenti target dell'applicazione e su di essi sono state sviluppate le Personas che sono servite a comprendere quelle che sono le abitudini e le necessitá degli utenti. Il team ha scelto di intraprendere la realizzazione di questo progetto cercando di seguire una metodologia di tipo Agile; le funzionalitá sono state sviluppate in modo incrementale sulla base della prioritá assegnata.

3.1 Design delle interfacce

Dal punto di vista del design delle interfacce si é cercato di seguire i principi KISS e Less is More con l'obiettivo di realizzare un'interfaccia semplice ed intuitiva. Sono stati utilizzati anche i principi della Responsive Design per poter rendere le interfacce utilizzabili da qualsiasi dispositivo e migliorare la User Experience. Per realizzare i mockup é stato utilizzato il web tool di app.moqups.com [1] che ha permesso di ottenere un'anteprima dell'applicazione in tempi rapidi. Sono una pagina dedicata ai dati meteorologici ed una che mostra un'anteprima degli stabilimenti con un carosello. Le ultime due pagine sono per il profilo e per la conferma dell'eliminazione di una scheda.

Figure 1: Mockup dell'applicazione

3.2 Design architetturale

Il sistema dispone di un servizio backend per la persistenza dei dati e per gestire la logica di business, e di un'interfaccia frontend per la visualizzazione dei dati e l'interazione con l'utente. L'accesso ai dati avviene con delle chiamate lato client verso delle API REST esposte dal servizio di backend. I vantaggi nell'utilizzo di questo tipo di approccio sono:

- indipendenza dei dati e dei modelli di comunicazione
- semplicitá nell'identificare e nell'accedere alle risorse
- indipendenza dalle tecnologie utilizzate

Le API REST esposte dai servizi esterni OpenWeatherMap e WeatherBit forniscono i dati meteo ai client dai quali vengono contattati. Una parte dei dati ottenuti viene trasmessa dai client al servizio di BE usando un modulo di comunicazione event-based, il quale riceve i dati, li elabora per poi inviare in broadcast un messaggio di alert.

4 Tecnologie

Per realizzare il sistema é stato adottato un solution stack di tipo MEAN, le tecnologie impiegate sono:

- Node + Express per lo sviluppo della componente server e la gestione delle richieste
- Angular per lo sviluppo dell'interfaccia
- MongoDB per la gestione dei dati persistenti

É stato scelto Angular come framework perché viene adottato spesso nell'azienda in cui lavorano i membri del team. Inoltre, sicché risulta essere una competenza fortemente richiesta dal mercato del lavoro si voleva maturare una prima esperienza con esso.

4.1 Frontend

A seguire vengono elencate le tecnologie implementate come supporto alla programmazione lato frontend.

Sviluppo software e testing

Si é scelto di ricorrere all'uso dell'Angular CLI che ha permesso di migliorare la struttura e l'organizzazione del codice lato client. L'Angular CLI facilita la gestione di tutte le attivitá noiose e comuni oltre a fornire una struttura di progetto scalabile. Questo tipo di supporto é uno strumento di base tra gli standard appartenenti all'ecosistema di Angular.

Stile

Per quanto riguarda lo stile delle interfacce si è scelto di utilizzare Bootstrap per facilitare la creazione delle componenti piu' comuni come finestre, barre, pulsanti e il carosello. Sono state impiegate le finestre modali per presentare i dati letti dal db e per realizzare una finestra di conferma da mostrare durante l'azione di cancellazione per una scheda. In seguito alle azioni compiute dall'utente possono verificarsi delle situazioni di: successo, errore o che necessitano di un avvertimento. Per comunicare con l'utente e rendergli noto ciò che accade, è stato impiegato MaterialSnackBar[4] un servizio usato per visualizzare delle notifiche sullo schermo. SnackBar è stato gestito in combinazione con un altro servizio HTTP Interceptor che intercetta le response e le request delle chiamate HTTP effettuate dalla Single Page Application. Altre componenti dell'applicazione sono state realizzate in modo nativo come ad esempio la navigation bar o il banner animato che viene usato per mostrare i messaggi di alert. Il progetto utilizza le classi di CSS Flexbox e sono state create delle funzioni di SASS Mixin per lo stile delle notifiche snackBar.

Gestione dei dati e dello stato

Le attivitá che operano sui dati sono state definite in appositi servizi, ogni servizio ha uno scopo ben definito. Segregare le operazioni in servizi ha permesso di ottenere delle funzionalitá che sono indipendenti rispetto ai componenti, e ha permesso di condividere logica e dati. I servizi risultano anche piú facili da testare e sono piú facilmente riutilizzabili. Quelli realizzati sono:

- AlertService: utilizza la libreria socket.io per generare degli eventi con cui vengono comunicati i dati meteorologici al servizio di BE e si ricevono in risposta i messaggi di alert
- BeachService: espone le operazioni adibite a recuperare le schede censite sul db e a cancellarle
- SnackbarinterceptorService: come giá accennato sfrutta i servizi HTTP Interceptor per intercettare le chiamate HTTP e MaterialSnackBar per disegnare le notifiche. Questo servizio si puó attivare per ogni chiamata e ha l'obiettivo di riconoscere situazioni che risultano essere di particolare importanza, quando le rileva comunica con l'utente spiegandogli ció che sta accadendo. Rappresenta il punto del codice sorgente in cui vengono globalmente gestiti tutti gli errori derivanti da chiamate HTTP ritenuti essere rilevanti per l'applicazione.

4.2 Backend

Come giá anticipato il backend é 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 é stata ottenuta attraverso un db non relazionale. Si é usata la libreria Mongoose che si occupa di gestire il mapping alle collection di MongoDB e di definire la forma dei documenti all'interno di tale collection. Il documento realizzato per la gestione dei clienti censiti sulla piattaforma si compone dei seguenti campi:

- id
- titolo: corrisponde al nome del locale
- cittá
- indirizzo
- servizi offerti dal locale che sono:
  - bar
  - ristorante
  - animali ammessi
  - carte
  - doccia calda
  - wifi
  - beach volley
  - cabine
  - area giochi
  - animazione
  - accesso per disabili
- foto: é il riferimento all'immagine di copertina
- galleria: sono i riferimenti alle immagini da esporre nella galleria
- coordinate: é la posizione del posto in latitudine e longitudine

Propagazione degli alert in tempo reale

Nell'implementazione delle funzionalitá real-time per la gestione dei messaggi di alert si é fatto uso della libreria socket.io. I client ricevono i dati meteo da alcuni servizi esterni e con l'uso di socket.io li trasmettono al BE generando un evento. Quando questo avviene il BE elabora i dati che ha ricevuto e compone un messaggio che attraverso un evento viene propagato in broadcast a tutti i client connessi alla piattaforma.

5 Codice

Durante lo sviluppo del software si é cercato di rendere il sistema scalabile avendo cura di segregare le operazioni piú importanti in appositi servizi o in funzioni all'interno delle varie componenti. Il modello di comunicazione event-based usato per i messaggi di alert ha un'esecuzione che rimane separata rispetto alla logica delle chiamate RESTful, questo permette al flusso principale di rimanere libero nel gestire le richieste HTTP da parte dei client.
Quando le applicazioni effettuano richieste HTTP e queste falliscono devono essere gestite. L'intercettore di Angular é un servizio che viene implementato per aggiungere comportamento alle richieste HTTP dell'applicazione.

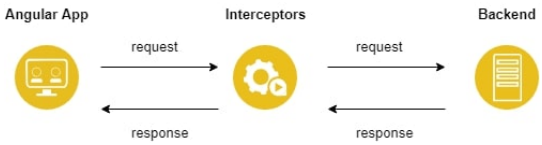

Figure 2: HTTP interceptor di Angular

Come mostra il diagramma sopra, gli intercettori sono sempre nel mezzo di una richiesta HTTP. In qualitá di intermediari consentono di eseguire operazioni sulle richieste di andata e ritorno dal server, creando un luogo perfetto per centralizzare il codice per cose come l'aggiunta di intestazioni, il passaggio di token, la memorizzazione nella cache e la gestione degli errori.[2]

6 Test

Le funzionalitá del sistema sono state testate sui browser Chromium e Firefox sia in versione desktop che mobile(mediante i tool di sviluppo integrati nel browser). I test hanno voluto verificare che le funzionalitá, i task principali e la parte grafica funzionassero correttamente.
Le API sviluppate lato server sono state opportunamente testate utilizzando lo strumento Postman che ha permesso di verificare che il comportamento fosse quello desiderato.

6.1 User Experience

Per offrire usabilitá e una buona user experience si é cercato di applicare quanto indicato dalle euristiche di Nielsen [5]:

1. Visibilitá dello stato del sistema: icone ed elementi vengono sotto intensificati oppure evidenziati a seconda che vengano selezionati o meno. Notifiche inviate all'utente in caso di errori, warning o azioni completate con successo

2. Corrispondenza tra sistema e mondo reale: si cerca di usare un linguaggio familiare all'utente con frasi e concetti a lui familiari. Le icone sono intuitive e convenzionali

3. Controllo e libertá: l'utente ha il controllo del contenuto informativo e puó muoversi liberamente tra i vari argomenti. L'applicazione non presenta percorsi predefiniti senza scorciatoie e non é possibile commettere azioni non volute

4. Consistenza e standard: le convenzioni grafiche adottate rimangono valide per tutta l'interfaccia e viene data all'utente la sensazione di essere sempre nello stesso ambiente

5. Prevenzione dell'errore: l'utente non viene messo in situazioni ambigue, critiche o che possono portare all'errore. Esiste sempre la possibilitá di tornare indietro

6. Riconoscimento anziché ricordo: i layout sono semplici e schematici, l'utente si orienta facilmente

7. Flessibilitá ed efficenza d'uso: si offre all'utente la possibilitá di un uso differenziale dell'interfaccia in base al livello di esperienza

8. Design e estetica minimalista: si cerca di non confondere l'utente. Il design é minimale e non sono presenti elementi superflui

9. Aiuto dell'utente: i messaggi di errore sono descrittivi in modo tale da esplicitare il tipo di errore e consentire possibili soluzioni. Viene chiesta una conferma per le azioni importanti

10. Documentazione: l'utente non ha bisogno di documentazione per fruire dell'applicazione

6.2 Test di Usabilitá

I test sono avvenuti con uno stretto campione di utenti che si é prestato alla verifica dell'utilizzo del sistema. Questi hanno fornito feedback grazie ai quali si é potuto capire quale aspetto poteva necessitare di alcuni miglioramenti. In generale gli utenti hanno trovato la disposizione dei vari elementi adeguata. Le osservazioni sollevate hanno riguardato perlopiú aspetti stilistici.

7 Deployment

Per avviare il sistema bisogna lanciare l'applicazione Angular, il server Node ed importare il dump del db. I passi da fare sono i seguenti, va bene eseguirli in questo ordine:

- posizionarsi con un terminale dove si vuole scaricare il progetto e clonare il repository

```
git clone https://github.com/pankake/BeachManagementPlatform.git
```

- importare il dump nel db

  ```
  mongorestore -d beachmp_db BeachManagementPlatform/backend/database/dump_db/beaches.bsc
  ```

- spostarsi nella directory del BE

  ```
  cd BeachManagementPlatform/backend/
  ```

- installare le dipendenze con npm

  ```
  npm install
  ```

- avviare il server node

  ```
  node index.js
  ```

- aprire un nuovo terminale e posizionarsi nel path dove é stato scaricato il progetto. Spostarsi nella seguente directory ed eseguire l'applicazione

  ```
  cd BeachManagementPlatform/src/app
  ng serve
  ```

- é possibile collegarsi da browser attraverso il seguente indirizzo

  ```
  https://localhost:4200
  ```

8 Conclusioni

Il team si ritiene soddisfatto del lavoro svolto avendo raggiunto tutti gli obiettivi previsti. In fase di testing si é potuto constatare che le interfacce utente realizzate sono sufficientemente intuitive. Il prodotto finale é semplice da utilizzare, svolge i compiti richiesti e fornisce un'esperienza d'uso piacevole agli utenti come qualitá visiva generale. Per quanto riguarda le funzionalitá aggiuntive future un'evoluzione del sistema potrebbe prevedere l'integrazione con altri servizi per aggiungere dati come ad esempio per la qualitá dell'acqua o dell'aria.

8.1 Commenti

Lo sviluppo di questo progetto ha permesso al team di arricchire il proprio bagaglio di conoscenze tecniche, dando la possibilitá di mettersi alla prova con diverse tecnologie dello sviluppo web moderno che risultano essere molto apprezzate nel mondo del lavoro.

Bibliography

[1] app.moqups web tool. *https://app.moqups.com/*.

[2] dev academy.com. *https://dev-academy.com/how-to-use-angular-interceptors-to-manage-http-requests/*.

[3] Angular Google Maps. *https://angular-maps.com/api-docs/agm-core/components/agmmap/*.

[4] MatSnackBar. *https://material.angular.io/components/snack-bar/overview/*.

[5] Jakob Nielsen. *Usability Engineering. (English). San Diego: Academic Press, 1994*.