Relazione su ilverometeo.it

Corso di Programmazione web e mobile

Anno accademico 2022

Luca Marazzi (951779)

16 luglio 2022

1. Introduzione

ilverometeo.it è un applicazione web che fornisce informazioni relative al meteo nella propria località o alternativamente in ogni città che l'utente possa richiedere.

Lo scopo principale del servizio è quello appunto di fornire l'informazione relativa al meteo nel modo più semplice possibile, in modo tale che sia possibile, per qualunque utente, dai più ai meno esperti, accedervi con facilità.

Le principali tecnologie utilizzate per il progetto sono ExpressJS, ExpressWinston, EJS, git, Bootstrap e Leaflet.

1.1. Analisi dei requisiti

1.1.1 Destinatari

I destinatari del progetto sono tutti gli utenti che abbiano un sufficiente livello di esperienza per comprendere come funzioni una pagina web in quanto l'interazione con l'applicazione è piuttosto basilare, pertanto qualunque utente potrà accedervi con facilità.

Gli utenti a cui è destinato il servizio dovranno conoscere ciò che stanno cercando, nello specifico la città per la quale vogliono conoscere il meteo, ma sarà anche possibile esplorare la mappa presente all'interno del sito per scoprire il meteo nelle città limitrofe.

Potranno usare qualsiasi dispositivo per connettersi in quanto, dato che abbiamo scelto di utilizzare Bootstrap, il layout della pagina si adatterà allo schermo utilizzato dal utente.

Gli utenti dovranno avere una sufficiente conoscenza del Italiano per comprendere a cosa servano le varie viste accessibili attraverso la navbar e una minima conoscenza del inglese per quanto riguarda il nome della città della quale si sta visualizzando il meteo.

La motivazione che può spingere un utente ad utilizzare il servizio è prevalentemente di tipo informazionale.

L'utente avrà una predisposizione alla ricerca attiva in quanto dovrà attivamente voler ricercare questo applicativo web per individuare le informazioni relative al meteo di cui necessita, potrà però inoltre individuare informazioni aggiuntive rispetto a quelle di cui necessita ricercando il meteo di una qualche altra città utilizzando la barra di ricerca presente nel applicazione o alternativamente la mappa.

Pertanto le aree del modello di Bates che descrivono maggiormente l'interazione che ci aspettiamo dal utente con il nostro sistema sono quelle di 'Searching' e 'Browsing'.

1.1.2. Modello di valore

Il servizio principale che fornisce valore al utente è quello appunto di ottenere informazioni precise relative al meteo attuale per la città da lui richiesta.

Gli elementi invece che possono generare valore per gli investitori potrebbero essere gli ads, qualora fossero introdotti all'interno della pagina, e potrebbero venire identificate le transazioni che generano questo valore andando a monitorare il quantitativo di utenti che hanno cliccato su di essi.

1.1.3. Flusso di dati

Il flusso di dati si basa principalmente su queste fonti di dati:

- OpenWeather API utilizzando il livello di sottoscrizione gratuito "Current Weather Data", che utilizziamo per fornire le previsioni meteo attuali per le città richieste dal utente
- Unsplash API per ottenere un immagine rappresentativa della città richiesta dal utente
- Leaflet che viene utilizzata per visualizzare la mappa del mondo all'interno del servizio

Tutte queste fonti di dati sono completamente gratuite e facilmente reperibili sul mercato, pertanto il nostro servizio andrà semplicemente ad interpolare i dati forniti da queste fonti per creare le pagine con cui l'utente interagirà.

Non sarà quindi necessario un metodo di archiviazione in quanto questi dati verranno richiesti ogniqualvolta l'utente li necessiti.

Non sono neanche presenti particolari vincoli alla pubblicazione se non quella di attribuire la fonte della mappa, ovvero Leaflet stessa, e utilizzare un link diretto alle immagini fornite dal API di Unsplash.

Le immagini possono essere riutilizzate purché ne venga accreditato l'autore.

1.1.4. Aspetti tecnologici

Il formato utilizzato per le immagini statiche, ovvero quelle che non vengono richieste al API di Unsplash, è il WebP, questo per avere le immagini con un peso molto inferiore rispetto ai formati standard e quindi venire incontro alla necessità di un ridotto utilizzo di banda che è tipica degli utenti

mobile. Mentre invece per quanto riguarda i contenuti che sono richiesti dal server alle API esterne, useremo il formato JSON per reindirizzare alla nostra pagina la risposta che avremo ottenuto.

Le principali tecnologie utilizzate sono:

- Bootstrap, per implementare le views con cui l'utente andrà ad interagire e renderle reattive, in modo tale che si possano adattare ad ogni schermo utilizzato dagli utenti
- ExpressJS, per implementare il web server su cui si regge il nostro applicativo
- ExpressWinston, per gestire il logging delle richieste rivolte dagli utenti al web server backend nonché per loggare tutti gli eventuali errori che possono incorrere. In ambedue i casi verrà utilizzata una string in formato JSON (in modo tale da avere un sistema di logging strutturato per poi analizzarlo con più facilità)
- EJS, per gestire la renderizzazione delle pagine di template
- Git, per il versioning della nostra web app
- Leaflet, per gestire la mappa nella nostra web app. L'ho scelto in quanto è un servizio completamente open source e inoltre è molto immediato da utilizzare.

2. Interfacce

Le interfacce, secondo l'obbiettivo che mi sono prefissato in partenza, ovvero quello di garantire un servizio il più semplice possibile, sono molto basilari.

In ogni vista avremo una navbar presente nella parte alta dello schermo che permetterà al utente di accedere alle altre viste e quindi alle informazioni in esse contenute.

Per il resto la pagina di index fornirà al utente alcune informazioni relative a come utilizzare l'applicazione e ulteriori riferimenti per accedere ad altre viste nelle quali saranno visualizzabili le sopracitate informazioni.

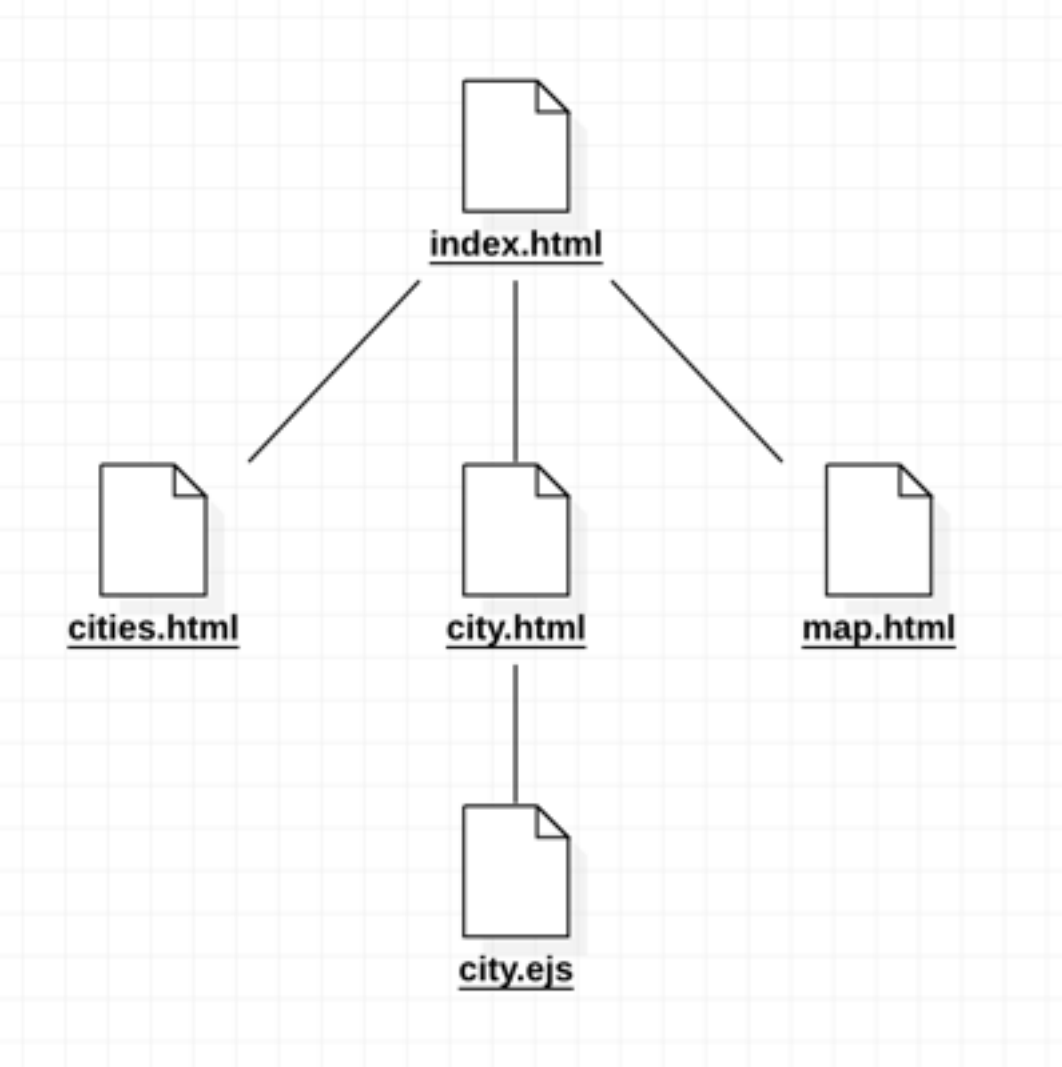
La pagina denominata "La tua città", accessibile attraverso la navbar o cercando una città nella barra di ricerca, richiederà al utente di condividere la propria posizione, nel caso in cui esso rifiutasse, verrà reindirizzato sulla pagina di indice, altrimenti l'utente accederà ad una vista nella quale saranno contenute le informazioni relative al meteo della sua città, nonché una mappa che gli permetterà di visualizzare la sua posizione corrente e "saltare" ad un'altra pagina selezionando un punto qualsiasi sulla mappa per ottenere le previsioni della città su cui ha premuto.

La vista "Tutte le nostre città" conterrà delle card Bootstrap all'interno delle quali l'utente potrà visualizzare le previsioni per le ultime città che ha cercato (utilizzando la barra di ricerca presente nella navbar) e tre città da me inserite a scopo di prova.

La pagina "La mappa" permetterà al utente di visualizzare la mappa fornita da Leaflet e ottenere le previsioni delle città dove cliccherà.

3. Architettura

3.1. Diagramma del ordine gerarchico delle risorse



Architettura del sistema

Le richieste al web server per visualizzare le pagine verranno tutte fatte attraverso transazioni GET e l'unico parametro che viene scambiato tra le pagine è il nome della città qualora l'utente utilizzasse la barra di ricerca per cercare una città specifica o se selezionasse un punto all'interno della mappa. Questo parametro verrà utilizzato nella costruzione della pagina

rappresentata dal template ‘city.ejs’ per visualizzare il meteo della specifica città richiesta.

Inoltre le altre risorse saranno accessibili attraverso transazioni GET eseguite richiedendo il path ‘/city_photo/${cityName} per ottenere la foto di una specifica città, o ‘/city_info/${cityName}’ e ‘/weather_info’ per ottenere rispettivamente la latitudine e la longitudine di una città, e le informazioni relative al meteo in una località rappresentata dalla latitudine e dalla longitudine inviati come parametri GET al servizio.

4. Codice

4.1 Node

Questo codice viene utilizzato all’interno del server per reindirizzare le chiamate che faremo dal web server al API di Openweathermap in modo tale da nascondere le chiavi del API dal utente (le quali verranno estratte da un file dotenv):

```
weather_api_key = process.env.WEATHER_API_KEY
[…]
app.get('/weather_info', function (req, res) {
  let fullUrl = `https://api.openweathermap.org/data/2.5/weather?lat=${req.query.lat}&lon=${req.query.lon}&appid=${weather_api_key}&units=metric`;
  axios.get(fullUrl)
  .then(json => {
    res.json(json.data)
  })
  .catch(error => {
    return res.status(500).json(error);
  })
```

```
});
```

4.2 Javascript

Questo codice invece verrà utilizzato per gestire tramite un webworker le chiamate al API di Unsplash:

```
function getImages() {
    const ImageLoaderWorker = new Worker('js/image_worker.js')
    const imgElements = document.querySelectorAll('img[data-src]')

    ImageLoaderWorker.addEventListener('message', event => {
        const imageData = event.data
        const imageElement = document.querySelector(`img[data-src='${imageData.imageURL}']`)
        const objectURL = imageData.res.urls.regular
        imageElement.setAttribute('src', objectURL)
        imageElement.removeAttribute('data-src')
    })

    imgElements.forEach(imageElement => {
        const imageURL = imageElement.getAttribute('data-src')
        ImageLoaderWorker.postMessage(imageURL)
    })
}
```

Codice in 'handle_city_page.js'

```
self.addEventListener('message', async event => {
```

```
    const imageURL = event.data

    const response = await fetch(imageURL)
    let jsonObj = await response.json()

    self.postMessage({
      imageURL: imageURL,
      res: jsonObj
    })
  })
```

Codice in 'image_worker.js'

Il primo codice verrà utilizzato per inizializzare il web worker, il secondo per implementarne la logica.

5. Conclusione

Lavorare a questo progetto mi ha permesso di imparare molte cose che non conoscevo, quindi sono felice del lavoro che ho svolto, sebbene ci fossero altre funzionalità che avrei voluto implementare ma per le quali sfortunatamente non ho trovato il tempo.

Magari in futuro potrei pensare ad una versione 2.0 migliorata, ma fino ad allora spero possa piacere!