

Corso di Programmazione Web e Mobile

A.A 2021-2022

Angello Fernando Pomayay Gabonal
Matricola: 984605



Introduzione

Il meteo è l'insieme di fenomeni atmosferici in un determinato tempo e in un determinato luogo e generalmente indica eventi tipici quali temperatura, nuvolosità, insolazione, umidità, precipitazioni giornaliere e vento

L'applicazione web “Meteo” permette di vedere che tempo fa nella propria città tramite la geolocalizzazione browser o in altre città tramite coordinate per la giornata attuale e per le prossime 7 giornate. I dati vengono acquisiti tramite richiesta a un API: <https://openweathermap.org/>

Oltre alle API sono stati usati NodeJS, CSS, HTML, JavaScript

Destinatari

Capacità e possibilità tecniche

L'utente che userà l'applicazione non avrà bisogno di particolari capacità tecniche per visualizzare la propria città, in quanto l'interfaccia è semplice ed intuitiva, ma comunque accattivante dal punto di vista grafico, invece se si vuole visualizzare una città diversa, se non è indicata nell'elenco bisognerà inserire le coordinate che sono facilmente recepibili su internet. Una volta caricata l'applicazione sul web questa potrà essere accessibile da qualunque dispositivo connesso ad internet dotato di un browser web (computer, smartphone ecc.). L'applicazione è responsive e si adatta alla dimensione del dispositivo che ne richiede l'utilizzo.

Linguaggio

All'utente finale è richiesta la conoscenza della lingua italiana e una conoscenza minima della lingua inglese in quanto la descrizione del tempo che si riceve dall'API è in inglese (in ogni caso il tipo di tempo è facilmente intuibile grazie alle icone). Essendo l'interfaccia abbastanza intuitiva anche qualcuno che non conosce la lingua italiana può usufruire del servizio.

Motivazione

L'utente viene molto probabilmente spinto ad utilizzare questa applicazione per necessità di conoscere che tempo fa nella sua città oppure nel luogo che intende visitare lo stesso giorno o nei successivi sette giorni.

Modello di valore

L'applicazione grazie alla geolocalizzazione del browser permette di accedere nell'immediato e molto facilmente alla pagina con le previsioni della propria città, un click e sei dentro. Anche la pagina con l'elenco delle città è molto intuitiva, cerchi la città che desideri nell'elenco la clicchi e guardi le previsioni, e nel caso non ci sia potrai inserire le coordinate della città nelle apposite caselle e questa verrà aggiunta automaticamente all'elenco ordinato alfabeticamente.

Flusso dei dati

I dati delle città e delle previsioni meteo vengono da richieste API a <https://openweathermap.org/> che fornisce appunto questo servizio di informazioni meteo a client di terze parti, i quali li ricevono in formato JSON. Ogni volta che l'applicazione richiede le previsioni per una città non presente nell'elenco (sia tramite coordinate inserite dall'utente o nella pagina "la mia città"), questa viene aggiunta automaticamente al database nel quale sono salvate le coordinate delle città.

Aspetti tecnologici

L'applicazione è conforme al modello MVC, infatti il controller è il server Node.js che interagisce con il model (il sito che fornisce l'API) e invia i risultati al view.

I risultati view sono per l'utente in HTML5 e CSS3. Le pagine che l'utente vede sono frutto dell'elaborazione del render che le converte da EJS (Embedded JavaScript templates) a pagine HTML comprensibili dal browser. I dati che passano dal server alle pagine sono codificati in JSON. Le richieste che vengono effettuate dal client verso il server usano i metodi

GET e POST che sono i più diffusi del protocollo HTTP. Il database che contiene le coordinate delle città è un database MongoDB

Interfacce

L'applicazione è composta da 3 diverse schermate:

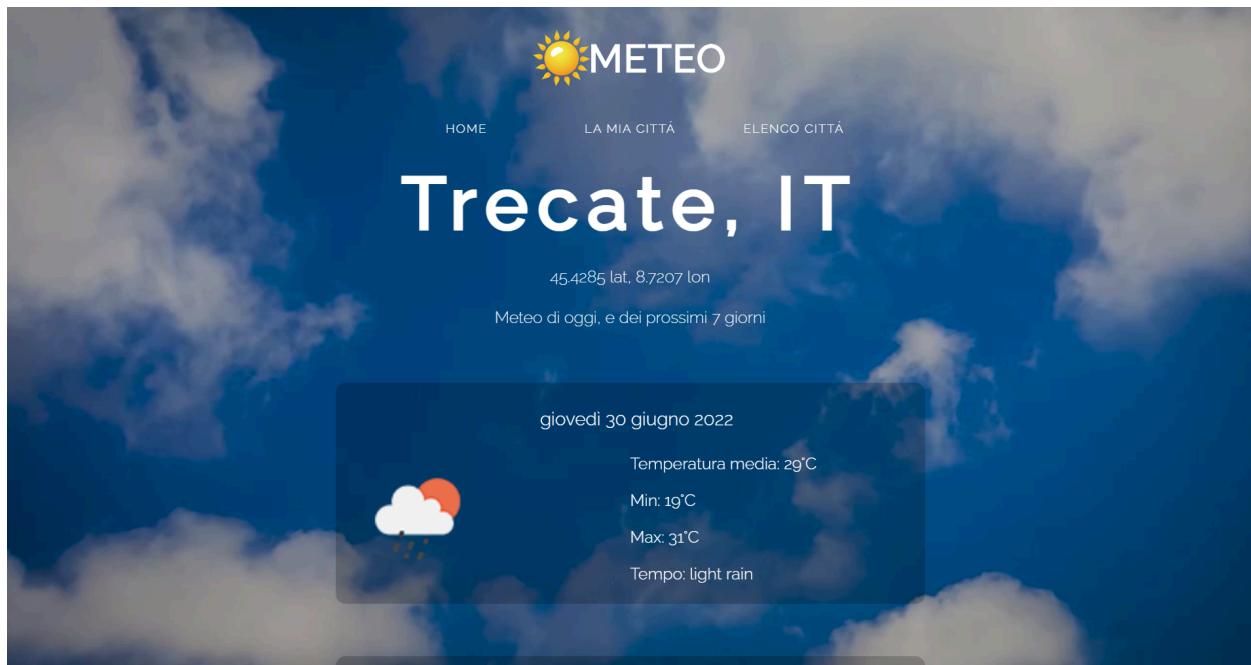
La pagina principale (index)



In questa pagina viene presentata l'applicazione. Sono presenti il logo, una navbar e dei buttoni per le altre pagine.

Grazie al logo con il sole e lo sfondo è facilmente intuibile l'argomento principale della pagina.

La pagina del meteo (city)



In questa pagina viene mostrato il meteo del giorno attuale e dei prossimi 7 giorni della città che sarà la città dell'utente se è arrivato a questa pagina tramite il pulsante “la mia città”, oppure della città scelta dall’elenco o tramite le coordinate.

Sulla pagina viene visualizzato il nome della città, lo stato e le coordinate geografiche. Per ogni giornata viene visualizzata la data comprensiva di anno, un’icona che indica il tempo, la temperatura media, la temperatura minima, la temperatura massima e una breve descrizione del tipo di tempo. La barra di navigazione è presente anche in questa pagina perchè è fondamentale per la navigazione tra di esse.

Al primo accesso alla pagina tramite il pulsante “la mia città” verrà molto probabilmente chiesto all’utente di permettere l’accesso della posizione all’applicazione web, in ogni caso se l’utente non accettasse il servizio rimane disponibile attraverso la lista o le coordinate

La lista delle città e le coordinate

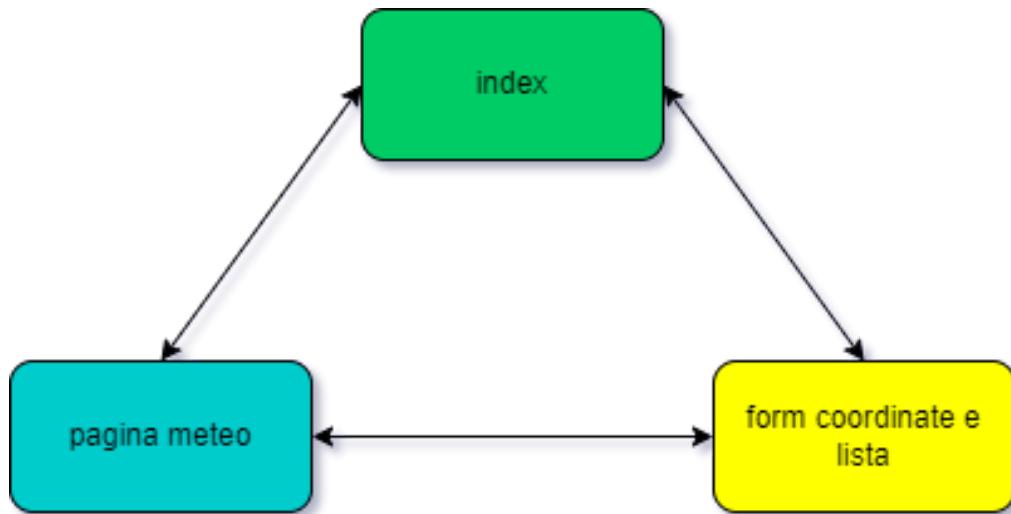


Questa pagina contiene un form nel quale si possono inserire dei valori di coordinate geografiche seguendo i criteri di inserimento indicati, infatti il range di valori che la latitudine può assumere vanno da -90 a +90 mentre la longitudine va da -180 a +180 (ovviamente con valori decimali intermedi), dopo che i valori sono stati inseriti sarà possibile inviare i valori con il pulsante “INVIA”, il server dopo che ha acquisito la richiesta reindirizzerà l’utente verso la pagina che conterrà le previsioni del luogo alle coordinate ricevute.

La pagina contiene inoltre una serie di pulsanti che formano un elenco ordinato alfabeticamente. Su ogni pulsante è indicato il nome della città e la sigla dello stato a cui appartiene in quanto al mondo possono esserci città con lo stesso nome. Anche tramite questi pulsanti si viene reindirizzati alle previsioni del luogo scelto.

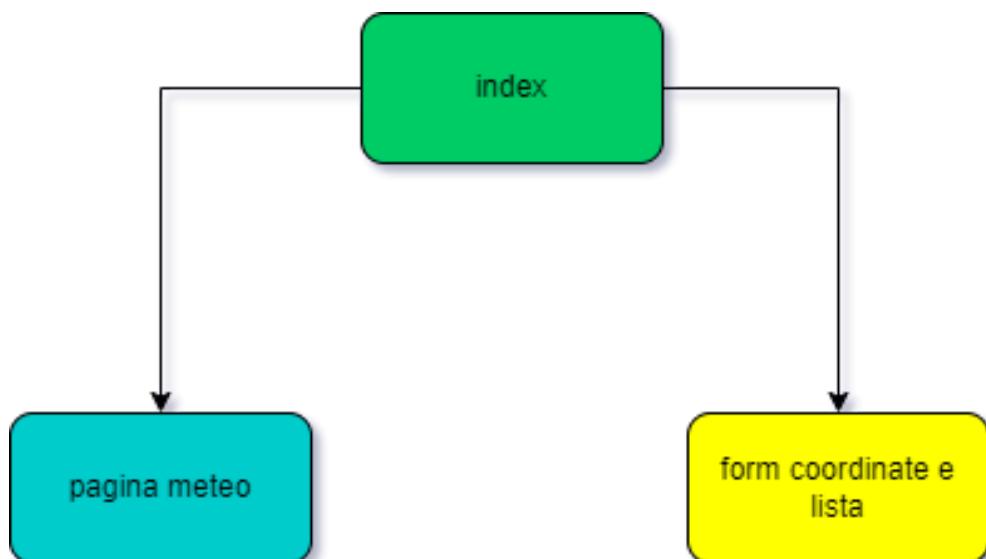
Architettura

Accessibilità delle risorse



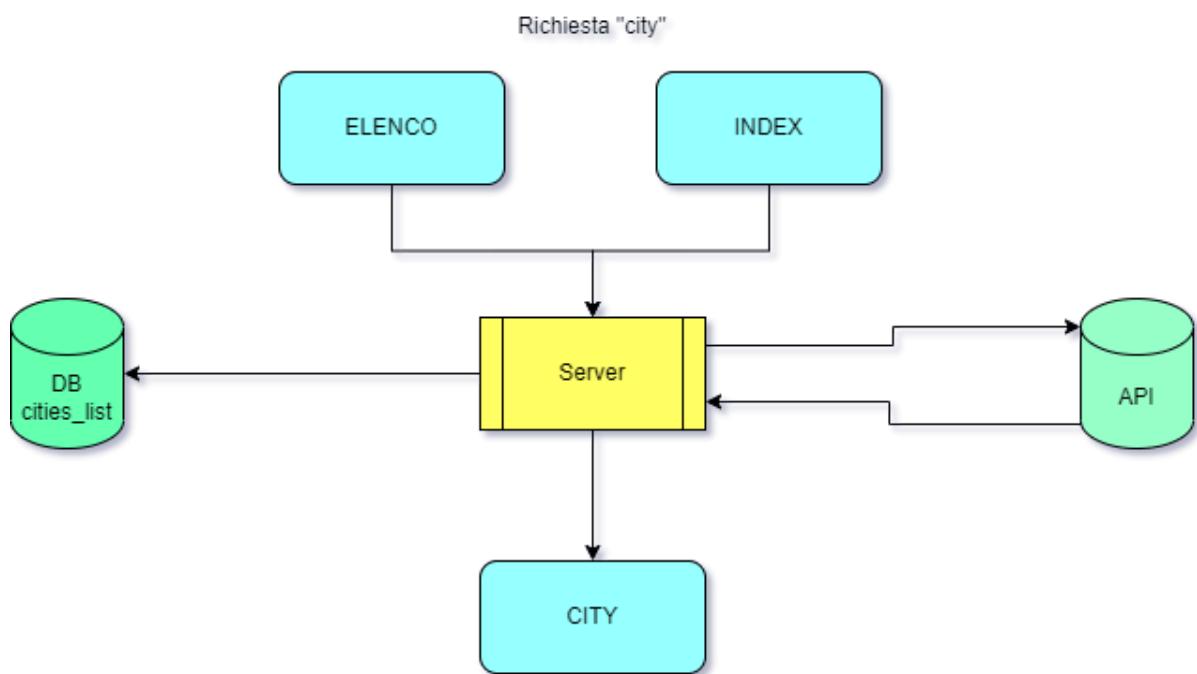
Grazie alla navbar presente in ogni pagina, ognuna di esse è accessibile direttamente dall'altra con un'unica limitazione: dalla pagina "index" e "pagina meteo" è accessibile solamente la pagina meteo della propria città

Ordine gerarchico



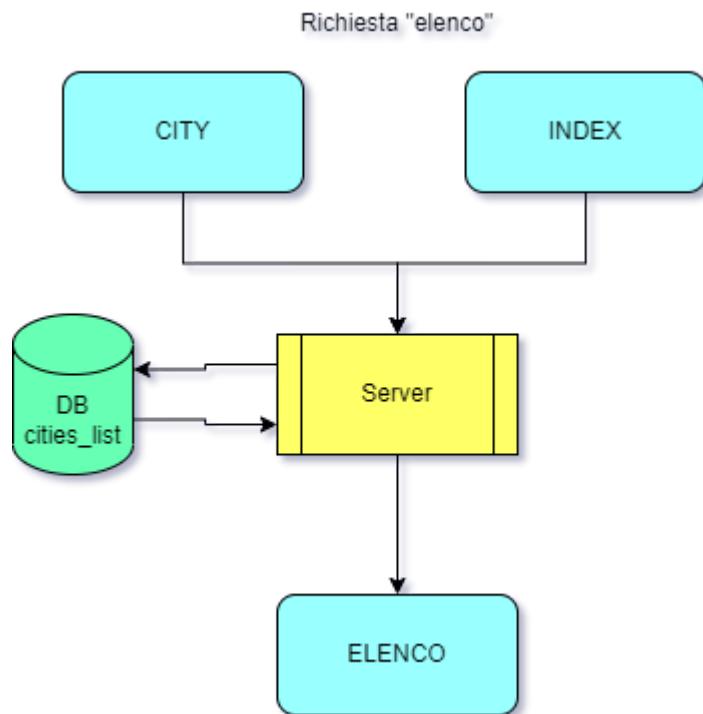
Se si volesse dare un ordine gerarchico per l'accesso alle risorse sarebbe questo, infatti l'index è ovviamente la prima che è immediatamente accessibile all'avvio del servizio e le altre due pagine sono accessibili da essa.

Chiamata “city”



La pagina INDEX o ELENCO inviano una richiesta “city” al server con la latitudine e longitudine passati in POST, il server elabora la richiesta, invia una richiesta all’API, e una volta ricevuta la risposta elabora i dati, invia una richiesta al DB per un eventuale aggiornamento della lista e infine risponde all’utente con la pagina “city” al quale sono stati passati i dati ricevuti dall’API

Chiamata “elenco”



Quando viene fatta una richiesta “elenco” dalle pagine city o index, questa viene ricevuta dal server che la elabora, invia una richiesta al DB che risponde con il contenuto dell’intera collezione “cities_list” che verrà successivamente inviata alla pagina “elenco” restituita all’utente

Database



Il database denominato “programmazione-web-mobile” ha al suo interno una “collection” denominata “cities_list” che è formata dall’attributo chiave _id. l’attributo “city_name”, l’attributo “latitude” e l’attributo “longitude”. In questa collection sono salvati i nomi e le coordinate di tutte i luoghi che sono stati richiesti all’applicazione

Codice

L’applicazione funziona tramite un server Node.js che comunica con l’api e risponde con pagine HTML al client. La pagina del meteo viene visualizzata dopo che il server ha ricevuto risposta dall’API e ha inviato i dati tramite JSON al client. Il client controlla che ci sia il contenuto e che non gli sia stato inviato un messaggio di errore:

```

<% if(weather !=null){ %>
<div id="my_city_title">
    <h1><%= place %></h1>
    <p class="subtitle"><%= coordinates %></p>
    <p class="subtitle">Meteo di oggi, e dei prossimi 7 giorni</p>
</div>

    <% for(var i=0; i<weatherDays.length; i++) {%
<!-- per ogni giorno verrà stampato -->
<div class="sing_cont">
    <div class="top">
        <p> <%= weatherDays[i].time %></p>
    </div>

    <div class="flex_container">
        <!-- foto e data -->
        <div class= "meteo_sinistra">
            <img src=<%= weatherDays[i].icon %> height="250px" width="250px">
        </div>

        <!-- informazioni -->
        <div class = "meteo_destra">
            <p>Temperatura media: <%= weatherDays[i].temp %>°C</p>
            <p>Min: <%= weatherDays[i].temp_min %>°C</p>
            <p>Max: <%= weatherDays[i].temp_max %>°C</p>
            <p>Tempo: <%= weatherDays[i].description %></p>
        </div>
    </div>
</div>
<% } %>

<% } %>
<% if(error !=null){ %
    <p>
        <%= error %>
    </p>
<% } %>

```

il tag `<%` è un tag che nelle pagine EJS indica la presenza di codice javascript, qua si può notare il controllo dell'oggetto “weather” e se è presente vengono stampati i dati di ogni giornata che si trovano all'interno dell'oggetto “weatherDays”. Se “weather” è null viene controllato se anche l'oggetto “error” è null altrimenti viene stampato

```

getForecastCity = async (forecastUrl, latitude, longitude) => {
  const options = { weekday: 'long', year: 'numeric', month: 'long', day: 'numeric' };

  try {
    let output = [];
    const response = await axios.get(forecastUrl)
    const result = response.data;
    if (result.daily == undefined) {
      return Promise.reject(error);
    }

    result.daily.forEach(element => {

      let outputObject = {
        temp: Math.round(` ${element.temp.day}` - 273.15),
        description: element.weather[0].description,
        time: `${new Date( value: element.dt * 1000).toLocaleDateString( locales: 'it-IT', options)}`,
        icon: `http://openweathermap.org/img/wn/${element.weather[0].icon}@2x.png`,
        humidity: element.humidity,
        clouds: element.clouds,
        visibility: element.visibility,
        pressure: element.pressure,
        longitude: longitude,
        latitude: latitude,
        temp_min: Math.round(` ${element.temp.min}` - 273.15),
        temp_max: Math.round(` ${element.temp.max}` - 273.15),
        _id: element.dt,
      };
      // push to output
      output.push(outputObject);
    });
    return Promise.resolve(output);
  } catch (error) {
    return Promise.reject(error);
  }
}

```

Questa funzione viene chiamata nel server quando gli viene fatta una richiesta POST “/city” e serve ad acquisire i dati riguardanti le previsioni meteo attraverso una richiesta all’API tramite l’URL passato alla funzione.

I dati di cui si ha bisogno vengono salvati in un oggetto javascript “outputObject” che viene poi inserito in seguito inviato come risposta di ritorno. Nei dati salvati si può notare come le temperature che vengono ricevute in fahrenheit vengono convertite in gradi Celsius sottraendo 273.15 al valore.

```
UpdateList = async (collectionName, elementToInsert) => {

    try {
        await mongoClient.connect();
        const database = mongoClient.db(DBName);
        const cities = database.collection(collectionName);
        let result = await cities.updateMany(
            filter: { city_name: elementToInsert.city_name },
            update: { $set: elementToInsert },
            options: { upsert: true });
        return Promise.resolve(result);
    } catch (error) {
        return Promise.reject();
    }
}
```

Questo codice permette di aggiornare il database che contiene le coordinate delle città, se la città è già presente le coordinate vengono aggiornate, invece se non è presente la città viene aggiunta. Questa funzione viene chiamata ogni volta che si fa una richiesta di tipo “city” al server.

```
getList = async (collectionName) => {
    try {
        await mongoClient.connect();
        const database = mongoClient.db(DBName);
        const cities = database.collection(collectionName);
        let result = await cities.find( filter: {} ).sort( sort: { "city_name": 1 } ).toArray();
        return Promise.resolve(result);
    } catch (error) {
        return Promise.resolve( value: [] );
    }
}
```

Questa funzione viene chiamata ad ogni richiesta di tipo “elenco”,

restituisce l'intero contenuto del database ordinato alfabeticamente in base a "city_name".

Conclusioni

Il meteo è un servizio in continuo utilizzo dalle persone, avevo immaginato questo progetto in modo diverso, avrei voluto aggiungere una mappa per automatizzare le richieste al posto del form che richiede le coordinate, ma purtroppo con openstreetmaps non è molto supportata una funzione di questo tipo. Alla fine guardando il funzionamento della mia applicazione l'ho pensata come un'app personale, infatti se migliorata con un sistema di login e quindi ognuno con uno spazio personale si otterrebbe un'app che permette di tenere sotto controllo il meteo dei luoghi desiderati che vengono salvati nella lista (ovviamente il sistema di gestione della lista andrebbe cambiato per permettere solo l'aggiunta tramite form e la cancellazione di un record).