Corso di Programmazione Web e Mobile A.A. 2018-2019

Fast Weather

Matteo Salvi, 909286

Sommario

Introduzione

Fast Weather è una semplice applicazione web per la visualizzazione dei dati principali metereologici di una data città del mondo.

Il funzionamento è estremamente semplice e permette tramite l'inserimento del nome di una città la visualizzazione di alcuni dati meteorologici:

- Umidità
- Temperatura attuale
- Temperatura minima e massima della giornata
- Velocità del vento

I dati vengono recuperati dal servizio openweathermap.com tramite le loro api.

Breve analisi dei requisiti

Destinatari

L'interfaccia è pensata per essere minimalista, intuitiva e semplice da usare. Per la costruzione visiva di essa, si è preferito uno stile grafico minimalista e flat, con pagine dai colori vivaci.

Simile ad uno stile Google, l'home page presenta poche informazioni (titolo e sottotitolo), un input box per la città e il bottone di controllo meteo. Mentre la pagina del meteo, presenta informazioni basilari in maniera minimalista, con l'uso di immagini animate per dare maggior enfasi al tempo attuale.

Non richiede particolari livelli di esperienza da parte dei suoi utilizzatori. L'applicazione è usabile su qualsiasi dispositivo connesso a internet dotato di web browser. E' richiesta una conoscenza minima della lingua inglese, in quanto l'applicazione è pensata per essere utilizzata da utenti da ogni parte del mondo.

La motivazione che spinge un utente a usare questa applicazione è quello di avere informazioni basilari e senza fronzoli in modo rapido e diretto. L'utente è predisposto ad una ricerca attiva delle informazioni di cui ha bisogno.

Modello di valore

Il modello ideale di valore per Fast Weather è un modello di business basato sulla pubblicità. Quindi il ricavo è derivato dalla vendita di un piccolo spazio (banner orizzontale) nelle pagine dell'applicazione. Il valore di rendimento dipenderà dal volume di traffico generato.

Flusso dei dati

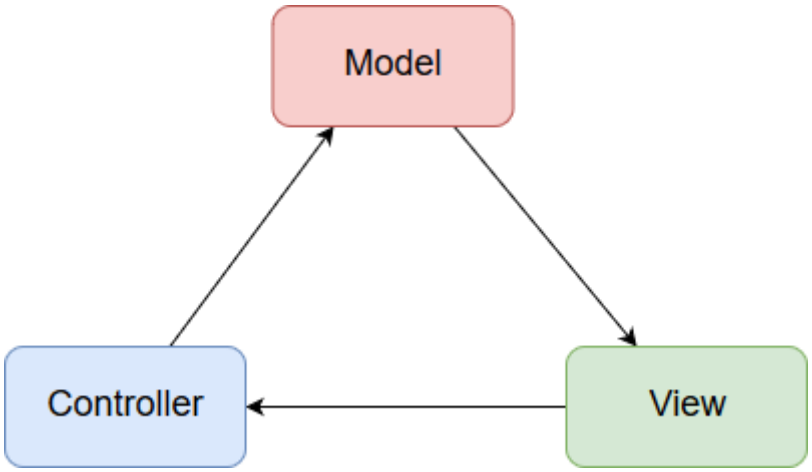
L'applicazione è basata sul servizio di OpenWeatherMap. Ad ogni richiesta, l'applicazione richiede le informazioni tramite una delle loro API. La risposta in formato JSON verrà quindi acquisita e parsata in modo da costruire successivamente l'interfaccia grafica di visualizzazione del meteo.

I dati ricevuti vengono elaborati al momento e non vengono in nessun modo archiviati.

L'uso delle API di OWM è soggetto alla licenza Creative Commons Attribution-ShareAlike 4.0 Generic License, come scritto nei termini di servizio di OWM.

Aspetti tecnologici

L'applicazione è sviluppata seguendo il pattern architetturale MVC (Model View Controller). Esso è diviso in due parti fondamentali, in modo da separare il Controller (server) dal View (client). Mentre il Model è rappresentato dal servizio OpenWeatherMap con il loro database, a cui l'applicazione accede tramite API.



Server

Il server è sviluppato usando il framework express.Js, basato sul microframework Node Js, in modo che possa gestire la parte http e l'invio delle request verso il servizio OpenWeatherMap. Una volta ricevuti i dati, essi verranno trasmessi al client.

Client

Il client è sviluppato usando la libreria ReactJS. Grazie ad esso, viene generata l'interfaccia in HTML 5, formattata seguendo le regole definite dal foglio di stile CSS3. Inoltre il client si occuperà di ricevere i dati dal server (precedentemente ottenuti da OWM e parsati come oggetto JSON). Essi verranno poi utilizzati per la interfaccia grafica.

Le richieste e i response vengono effettuati con parametri in formato JSON, in modo da essere facilmente recuperati ed elaborati dall'applicazione e dal servizio OpenWeatherMap.

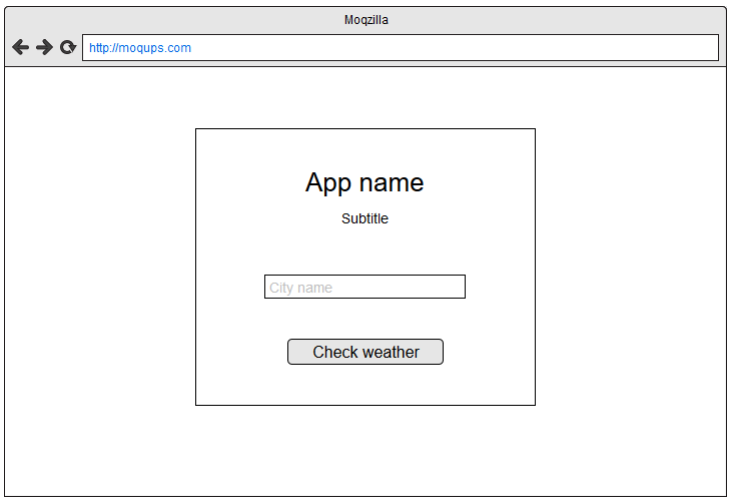
Interfacce

Mockup

L'interfaccia si compone di tre viste o schermate.

Home page

La home page è la vista principale. In questa schermata troviamo il nome della pagina, l'input box per l'inserimento del luogo e il bottone per passare alla schermata successiva.

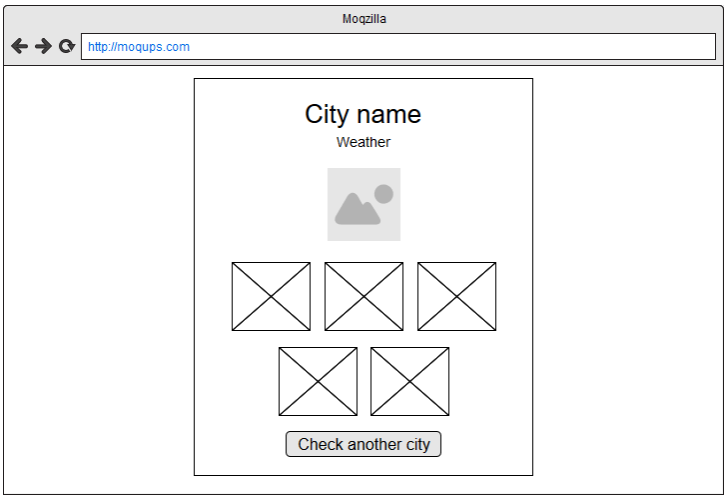


Weather page

In questa schermata, verranno mostrati i dati metereologici principali del luogo scelto dall'utente.

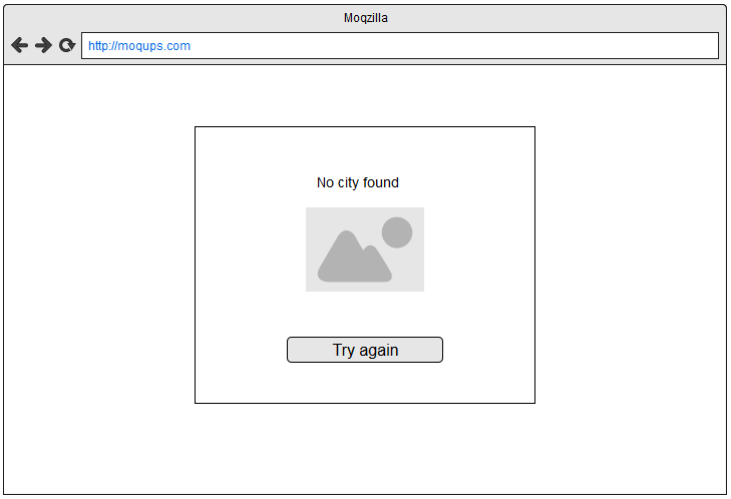
Dati metereologici visualizzati in questa schermata:

- Tempo attuale
- Umidità
- Temperatura attuale
- Temperatura minima e massima
- Vento



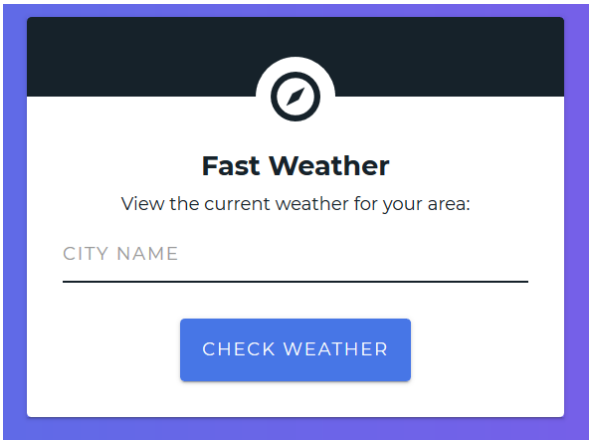
Error page

Semplice pagina d'errore nel caso l'utente sbagli ad immettere una città, ovvero un nome errato o una città non riconosciuto dal servizio OpenWeatherMap.

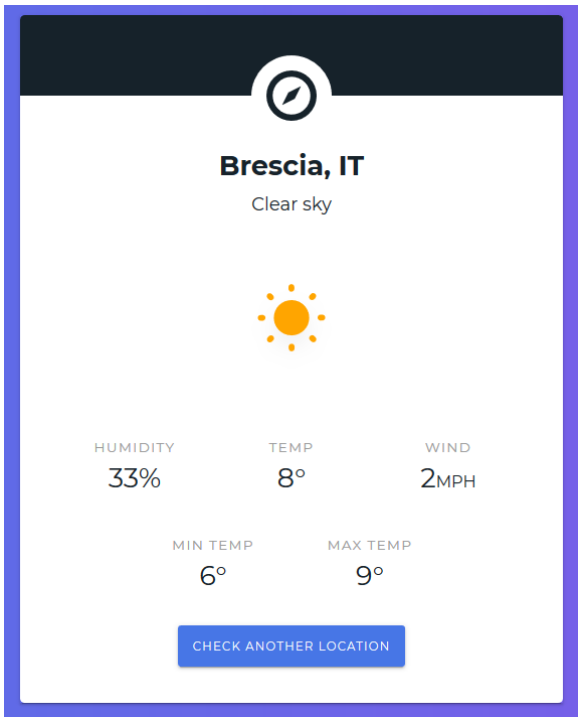


Interfaccia grafica definitiva

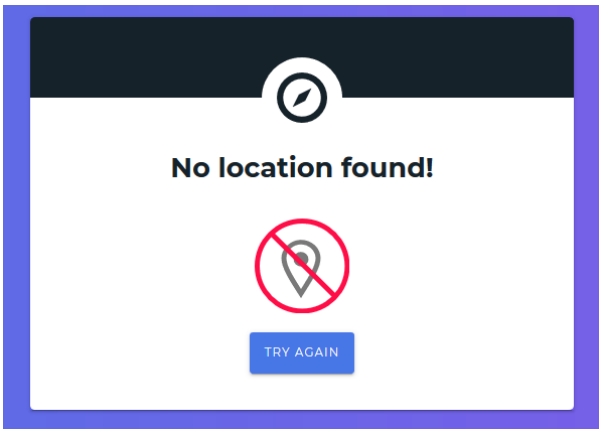
Home Page



Weather Page

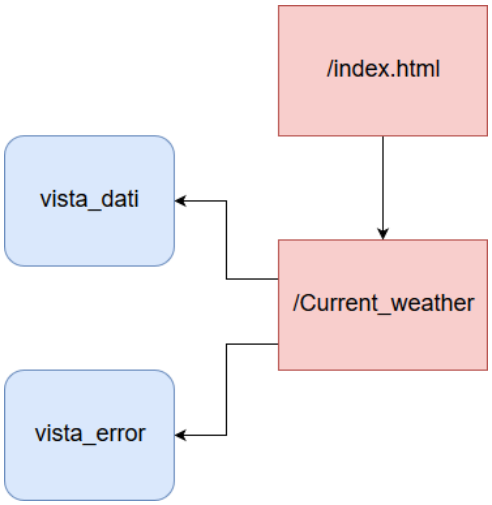


Error Page



Architettura

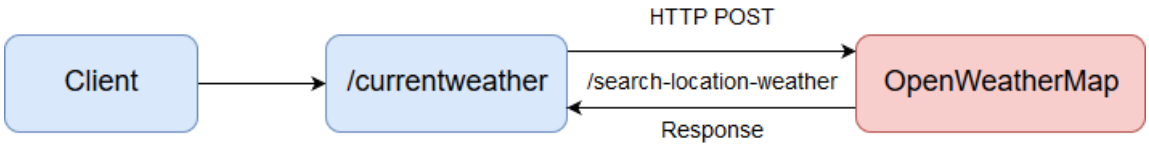
Diagramma delle risorse



L'applicazione è formata da due pagine differenti:

- index.html, pagina principale dell'applicazione che coincide la schermata principale
- current_weather, con due differenti schermate. La prima è la pagina dei dati metereologici e la seconda è la schermata di errore.

Descrizione delle risorse



Tramite il form della pagina principale, l'utente inserisce il nome di una città. Una volta partita la procedura, l'applicazione carica la pagina adibita al risultati, ovvero currentweather. Essa invia una richiesta http post a OpenWeatherMap tramite http endpoint /search-location-weather della parte server dell'applicazione.

Quindi l'endpoint si occuperà di gestire la richiesta e la ricezione dei dati da OWM. I dati ricevuti verranno parsati in un oggetto json e trasferiti al client, che potrà quindi decidere quale delle due viste utilizzare ed elaborare i dati.

Codice

Il seguente codice sorgente del client richiama la funzione API weatherAppAPI del server, con i parametri di ricerca weatherSearchData in formato JSON (parametri quali locationType e locationName).

```
65  componentDidMount() {
66    const weatherSearchData = this.props.location.state;
67    const handleApiResponse = this.handleApiResponse;
68
69    weatherAppAPI({}, weatherSearchData, function(err, data) {
70      if (err) {
71        handleApiResponse(err);
72      } else {
73        handleApiResponse(null, data);
74      }
75    });
76  }
77
```

Il seguente codice sorgente rappresenta la funzione API weatherAppAPI del server. Questa API si occupa di gestire (inviare e ricevere dati) la richiesta http Post con i parametri ricevuti dal client.

Principalmente crea la XMLHTTPRequest, impostando header (specificando l'utilizzo del formato JSON per i dati scambiati), body (parametri ricevuti dal client) e l'evento OnReadyStateChange (con un codice per la gestione del cambio di stato della richiesta).

Alla fine la richiesta verrà mandata all'endpoint /search-location-weather che si occuperà di indirizzare la richiesta a OpenWeatherMap.

Una volta ricevuti i dati in modo asincrono, i dati ricevuti verranno parsati in oggetto JSON e verranno usati come parametro per il metodo callback. Di fatto, l'oggetto JSON verrà quindi trasferito al client che potrà elaborare i dati.

```
export function weatherAppAPI(requestHeaders, requestBody, callback) {
  var xhr = new XMLHttpRequest();// eslint-disable-line no-undef
  const requestEndpoint = WEATHER_API_ENDPOINT;
  const requestOptions = {
    method: 'post',
  };

  if (requestBody) {
    requestOptions.body = requestBody;
  }

  if (requestHeaders) {
    requestOptions.headers = requestHeaders;
  }

  xhr.open(requestOptions.method, requestEndpoint, true);

  xhr.setRequestHeader('Content-Type', 'application/json');

  xhr.onreadystatechange = function() {
    if (xhr.readyState === 4) {
      const resposeData = JSON.parse(xhr.response);
      if (xhr.status !== 200 || resposeData.data.cod !== 200) {
        return callback(resposeData);
      }

      return callback(null, resposeData);
    }
  };

  xhr.send(JSON.stringify(requestOptions.body));
}
```

Il seguente codice, invece, si occupa di generare l'url e di indirizzare la richiesta http Post verso OpenWeatherMap.

```
const fetch = require('node-fetch');
const generateWebAppURL = require('server/utils').generateWebAppURL;

module.exports = (app) => {

  app.post('/search-location-weather', (req, res) => {
    const requestBody = req.body;
    const apiUrl = generateWebAppURL(requestBody.locationType, requestBody.locationData);

    fetch(apiUrl)
      .then(res => res.json())
      .then(data => {
        res.send({ data });
      })
      .catch(_err => {
        res.redirect('/error');
      });
  });
};
```

Conclusioni

Fast Weather è un progetto semplice ed immediato per la visualizzazione meteorologica. Il progetto è stato sviluppato tenendo bene a mente il principio KISS, ovvero Keep It Simple, Stupid. La semplicità prima di tutto.

Nonostante il buon punto a cui sono arrivato nello sviluppo, il progetto si può considerare ancora nelle fasi iniziali.

Lo sviluppo si può ulteriormente estendere sia nel front end che nel back end. Nel front end si può pensare di aggiungere ulteriori informazioni sulle prossime previsioni orarie o giornaliere di un dato luogo, con l'aggiunta di grafici (per esempio, andamento della temperatura). Come anche la rilevazione automatica della geolocalizzazione dell'utente.

Per non parlare di ulteriori ottimizzazioni del back end. Per esempio, una delle possibili tecniche di ottimizzazione delle richieste verso OpenWeatherMap è quello di salvare i dati ricevuti ed elaborati in uno database (ad esempio, mongodb, in modo da sfruttare ulteriormente il formato JSON) in modo da risparmiare sulle richieste api verso OpenWeatherMap. Naturalmente i dati archiviati dovrebbero essere validi solo per un certo lasso di tempo, per non fornire informazioni troppo datati.