Corso di Programmazione Web e Mobile

A.A. 2019-2020

GeoRating Milan

Marco, Galli, 941816

Andrea, Zaffanella, 941818

1. Introduzione

Il sito web realizzato permette agli utenti di consultare una mappa nella quale sono presenti servizi e attività sul territorio milanese; è stato pensato in una chiave turistica, fornendo nella stessa pagina informazioni utili come le posizioni di AP wifi liberi, alberghi, musei e parchi. E' inoltre possibile scrivere recensioni e commenti e consultare quelli già presenti.

In modo più tecnico, la app è costruita per geolocalizzare e recensire dati in formato JSON, Il sito presenta un'unica pagina con menu che permette di passare da una categoria di dati all'altra senza ricaricare. Le posizioni delle location nel dataset vengono visualizzate come marker sulla mappa, realizzata grazie alla API di Google Maps. Cliccando su un marker si apre una finestra che mostra la valutazione media, tutte le recensioni e permette di scriverne una nuova. Il sistema di recensione si appoggia ad un server Node.js che gestisce le query ad un database MySQL, il quale contiene tutte le recensioni.

1.1. Breve analisi dei requisiti

1.1.1. Destinatari

Capacità e possibilità tecniche.

Per utilizzare il sito non è necessario alcun tipo di esperienza pregressa, in quanto esso è molto intuitivo e una guida introduttiva non è quindi necessaria. Il sito può inoltre essere utile sia a chi ha un ricerca particolare in mente (come ad esempio trovare un wifi aperto nelle vicinanze) sia a chi invece sta solo cercando attivita da svolgere/luoghi da visitare nella zona.

Tipologie di device supportati

Il sito può essere visitato da ogni dispositivo connesso alla rete, ma è stato pensato in particolare per laptop/desktop PC.

Linguaggio

Il sito è in inglese, ma grazie al intuitività dell'interfaccia, è possibile utilizzarlo senza particolari conoscenze linguistiche, in quanto utilizza per la maggior parte parole note ed utilizzate internazionalmente.

Vision e mission della app

Il sito è stato pensato per scopi diversi, può rappresentare un modello educational, per esempio per fornire un servizio gratuito ai turisti oppure business se si dovesse inserire pubblicità per trarre profitto dalle visualizzazioni.

L'utente di questo sito è tipicamente un turista, invogliato ad accedervi perché può visualizzare in unico posto punti di interesse per lui e allo stesso tempo accedere alle recensioni di ognuno. Secondo il modello di Bates questo tipo di utenza è di tipo attiva ma non diretta, e quindi i contenuti del sito sono stati organizzati in modo da essere consultabili nella forma di una mappa interattiva.

1.1.2. Modello di valore

Il valore principale dell’applicazione è quello di fornire in un'unica pagina la geolocation di una gamma di servizi e attività utili per il cittadino e per il turista.

Un altro elemento di valore è la possibilità di inserire commenti e recensioni per ogni servizio, che, fornendo esperienze e punti di vista altrui, permettono all’utente di valutare al meglio e di farsi una propria idea a riguardo al servizio o attività in questione. Più cresce il numero di recensioni più si incrementa il valore.

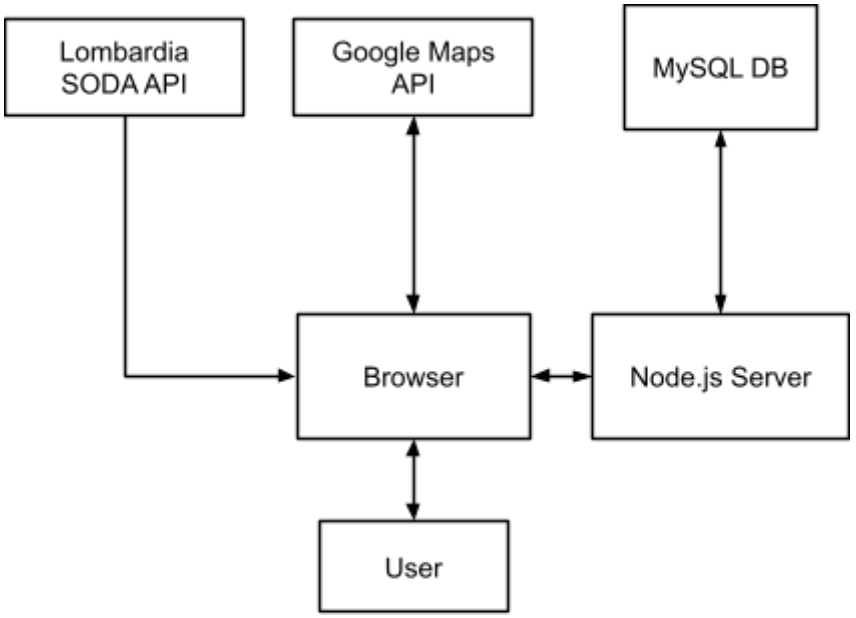
Infine un altro aspetto chiave è la modularità della app, che consente a questo “proof of concept” di espandersi con nuovi : è infatti possibile aggiungere nuovi dataset in modo rapido e senza particolari cambiamenti al codice.

1.1.3. Flusso dei dati

I dati delle location visualizzate all'interno del sito provengono da terze parti, in questo caso dagli archivi open data della regione lombardia (https://www.dati.lombardia.it). Questi vengono quindi esportati in formato JSON e ne viene effettuato il "parsing". Dal JSON vengono estratti il nome della location e le sue coordinate e questo viene rappresentato sulla mappa. Questi dati vengono inoltre salvati in localStorage.

Le recensioni vengono visualizzate nell'apposita finestra attraverso ad una query al database MySql che le contiene dal server Node.js. La query restituisce a Node.js tutti i commenti della location interessata.

Quando si scrive una nuova recensione, i dati inseriti nella form HTML vengono "parsati" da Javascript e inviati al server Node.js che ne gestisce l'archiviazione nel database MySql.



1.1.4. Aspetti tecnologici

Tecnologie utilizzate

HTML5/CSS

- La pagina principale è sviluppata in HTML5 ed è stata validata.
- Ogni elemento della pagina HTML è stato stilizzato con CSS.
- L'applicazione si serve dell' API HTML5 local storage per contenere la lista di location.

AJAX

- Nel progetto sono presenti diverse chiamate XMLHTTPRequest.
- La chiamate ai server della lombardia e al server Node.js interrogano un file JSON.

Node.js (Express)

- Il server Node.js interroga e carica dati su un database, ricevendo risposte in formato JSON, che verranno poi a sua volta inviate al client richiedente.
- Abbiamo usato il framework Express per semplificare lo sviluppo.

Tecnologie aggiunte

MySQL

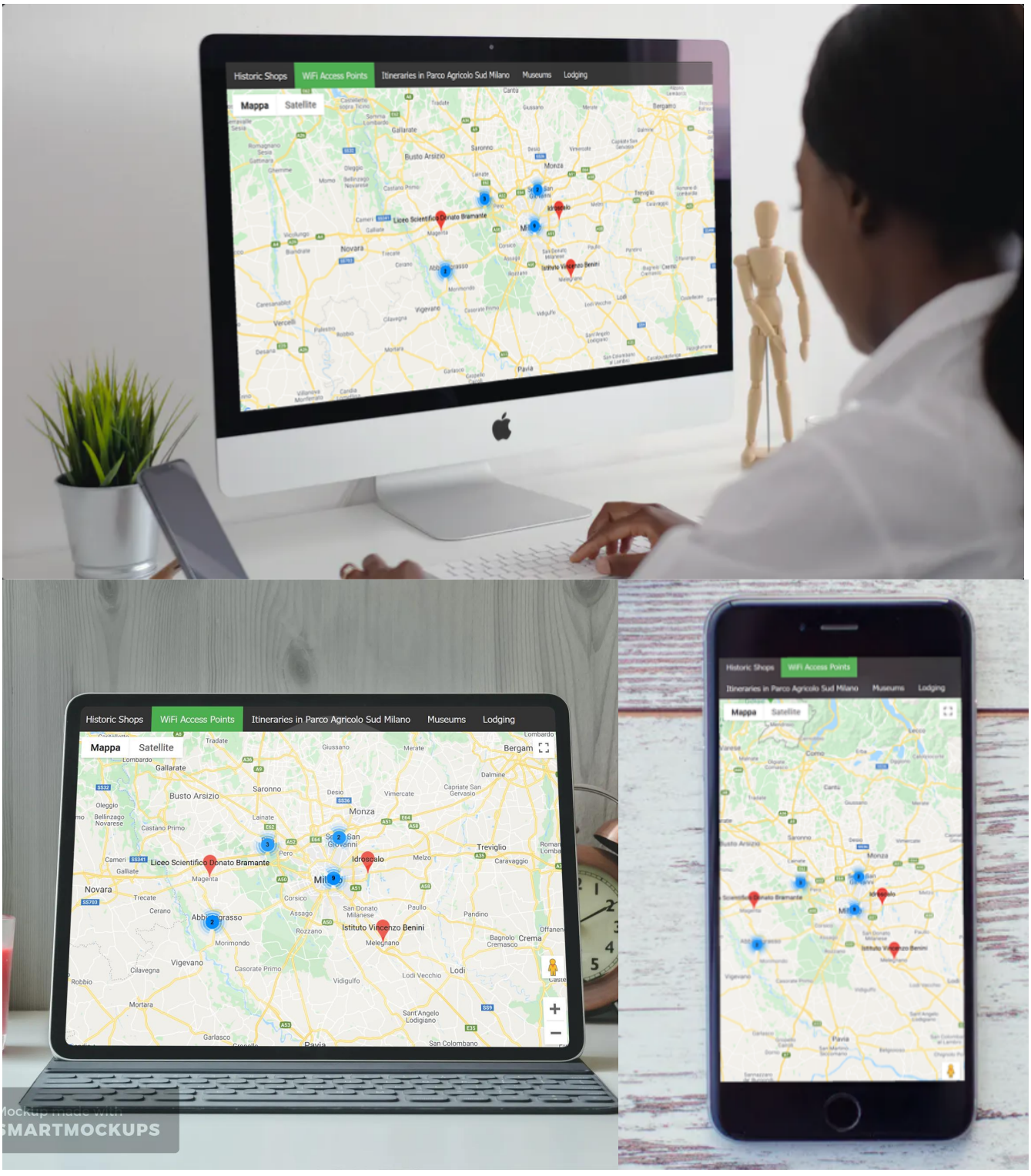
- Il database mySQL immagazzina i dati, come relativi alle recensioni (autore, data, commento, voto, location, id). Su di esso vengono eseguite query, per mostrare le recensioni relative ad una location o per aggiungere una nuova recensione.

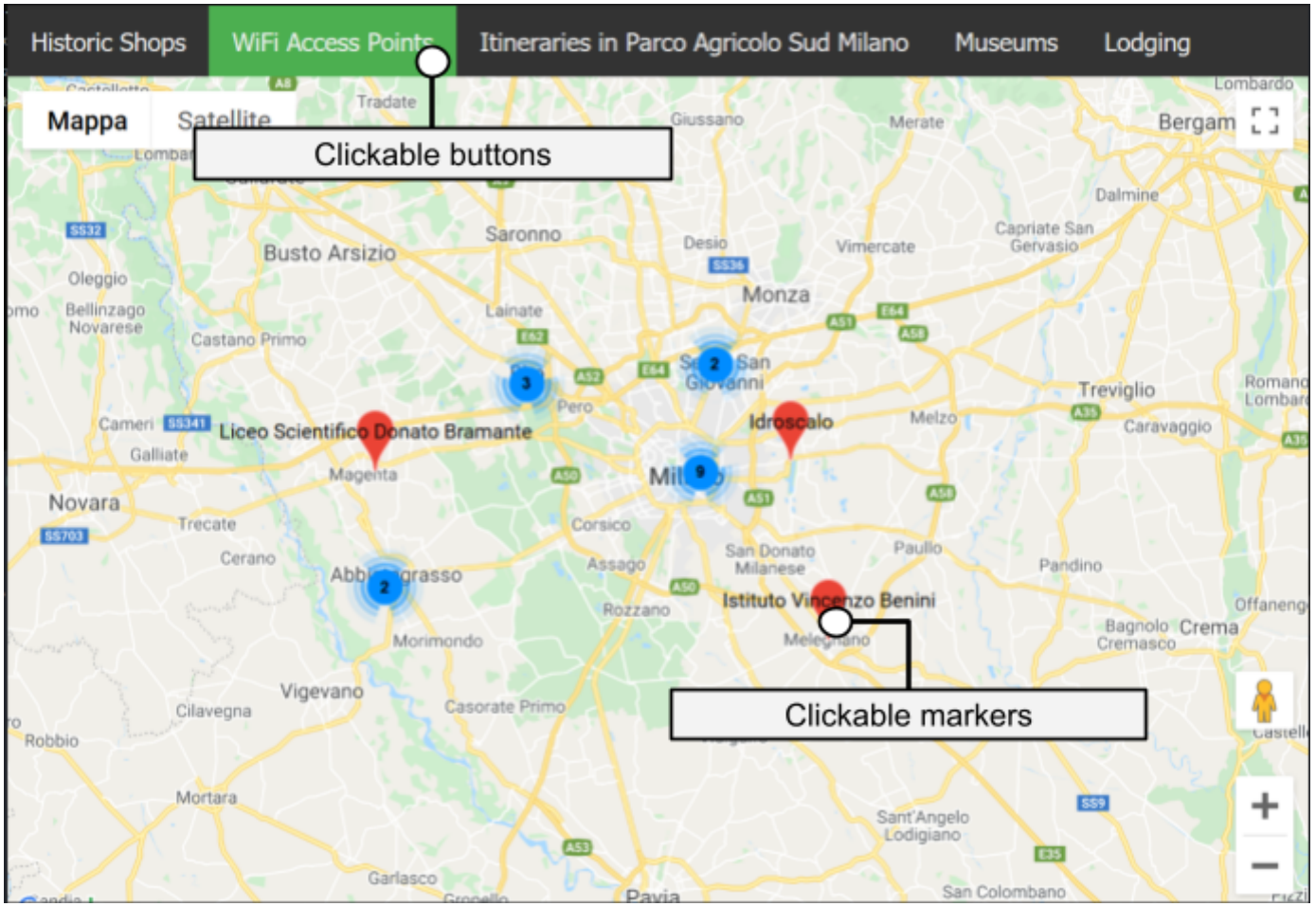
Google Maps API

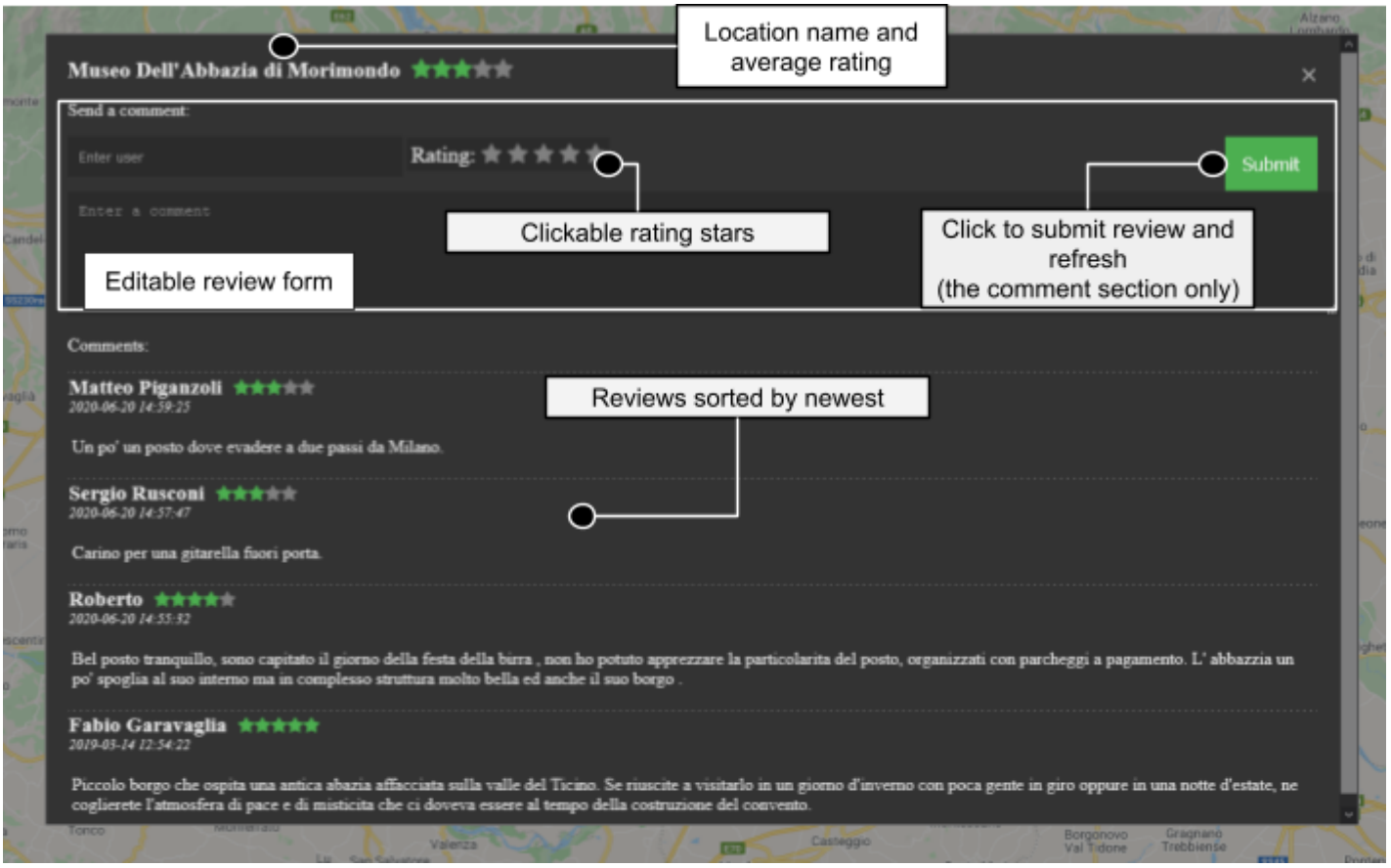
- Sono stati utilizzati due script, uno per rappresentare la mappa in sé, e un altro per gestire il raggruppamento dei marker al variare dello zoom.

2. Interfacce

Interfaccia principale:




Historic Shops
WiFi Access Points
Itineraries in Parco Agricolo Sud Milano
Museums
Lodging
Mappa
Satellite
Clickable buttons
Liceo Scientifico Donato Bramante
Idroscalo
Istituto Vincenzo Benini
Clickable markers

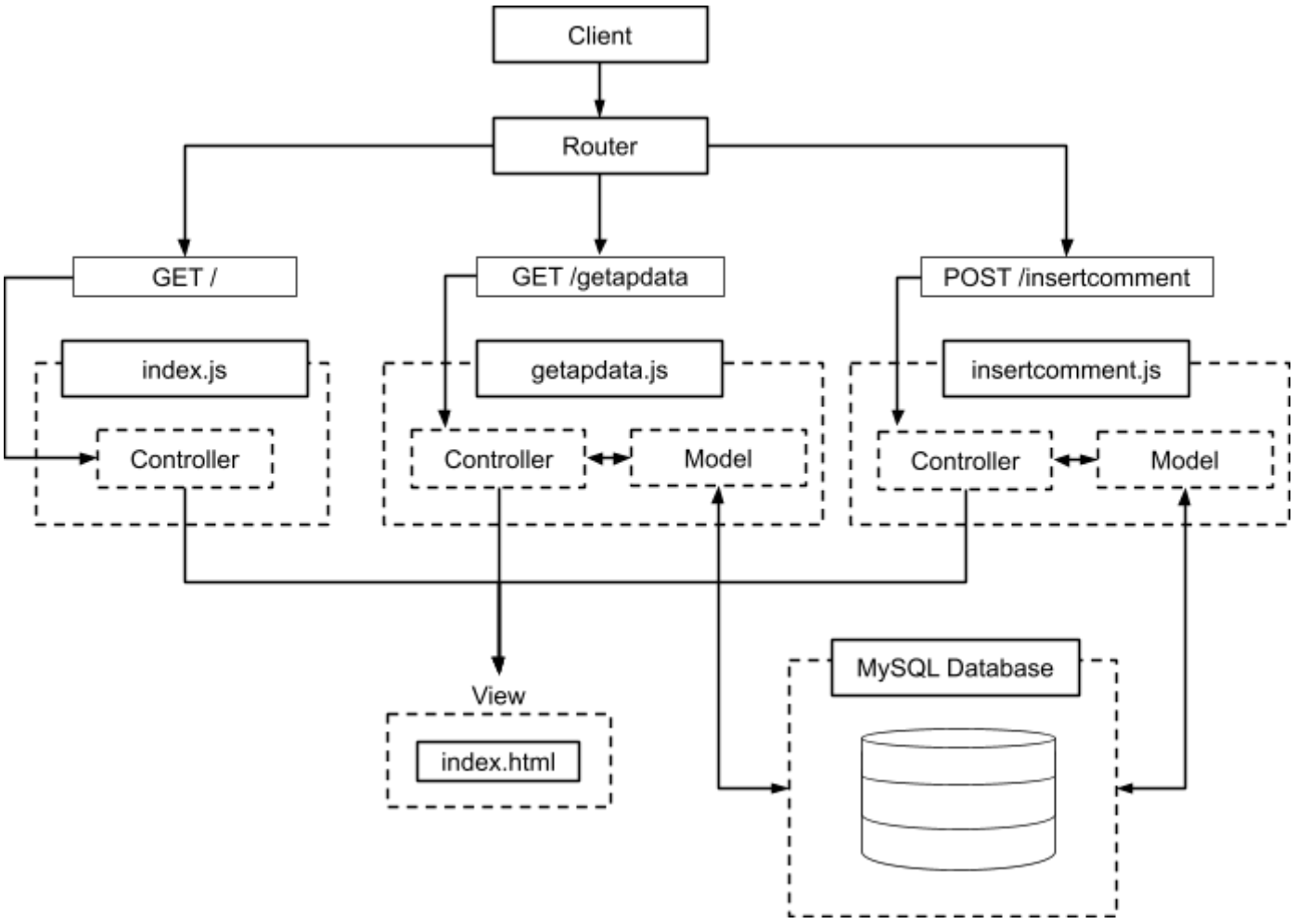

Location name and average rating
Museo Dell'Abbazia di Morimondo
Send a comment:
Enter user
Rating:
Submit
Enter a comment
Clickable rating stars
Click to submit review and refresh (the comment section only)
Editable review form
Comments:
Reviews sorted by newest
Matteo Piganzoli
2020-06-20 14:59:25
Un po' un posto dove evadere a due passi da Milano.
Sergio Rusconi
2020-06-20 14:57:47
Carino per una gitarella fuori porta.
Roberto
2020-06-20 14:55:32
Bel posto tranquillo, sono capitato il giorno della festa della birra , non ho potuto apprezzare la particolarita del posto, organizzati con parcheggi a pagamento. L' abbazzia un po' spoglia al suo interno ma in complesso struttura molto bella ed anche il suo borgo .
Fabio Garavaglia
2019-03-14 12:54:22
Piccolo borgo che ospita una antica abazia affacciata sulla valle del Ticino. Se riuscite a visitarlo in un giorno d'inverno con poca gente in giro oppure in una notte d'estate, ne coglierete l'atmosfera di pace e di misticita che ci doveva essere al tempo della costruzione del convento.

3. Architettura

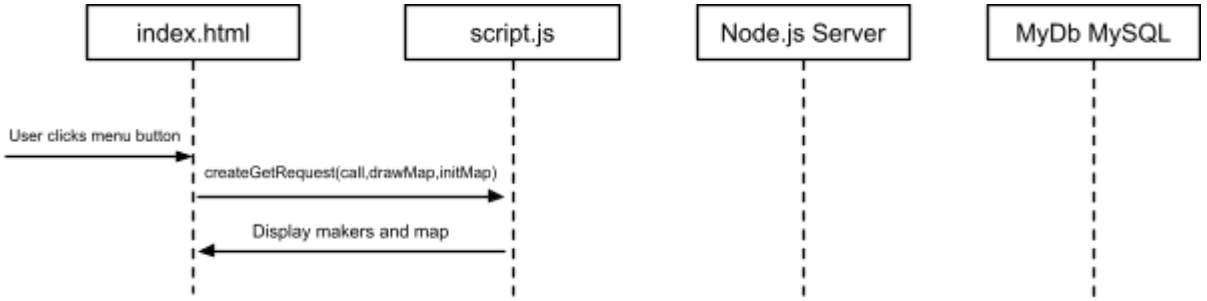
3.1. Diagramma dell'ordine gerarchico delle risorse

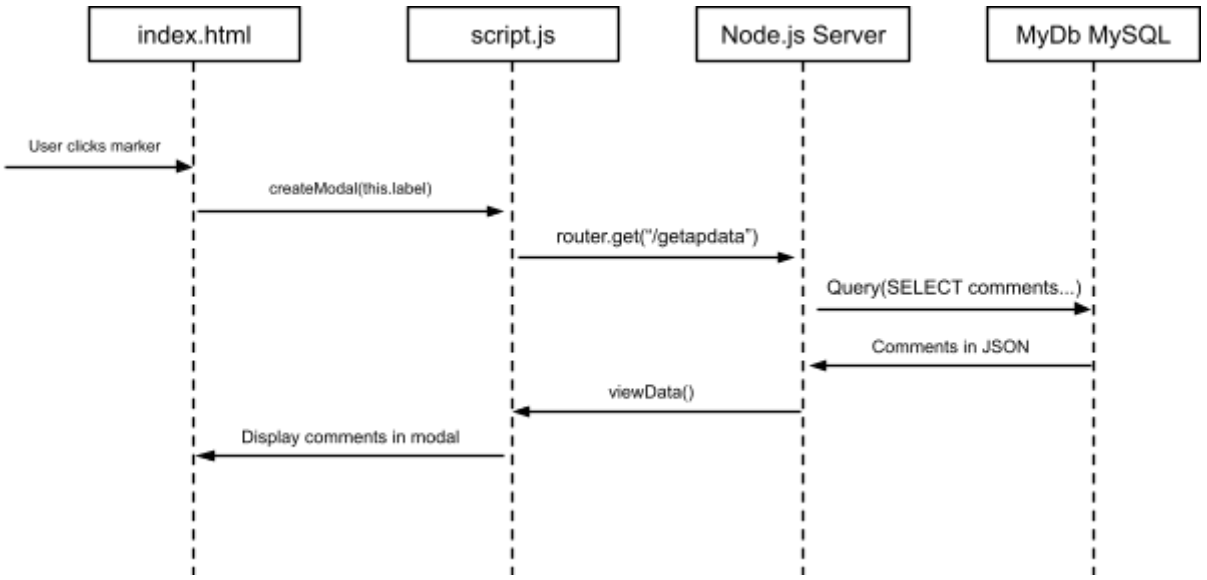
Il progetto sviluppato è un sito one page, che presenta quindi solo una pagina principale, dove è presente la mappa e tutte le azioni disponibili.

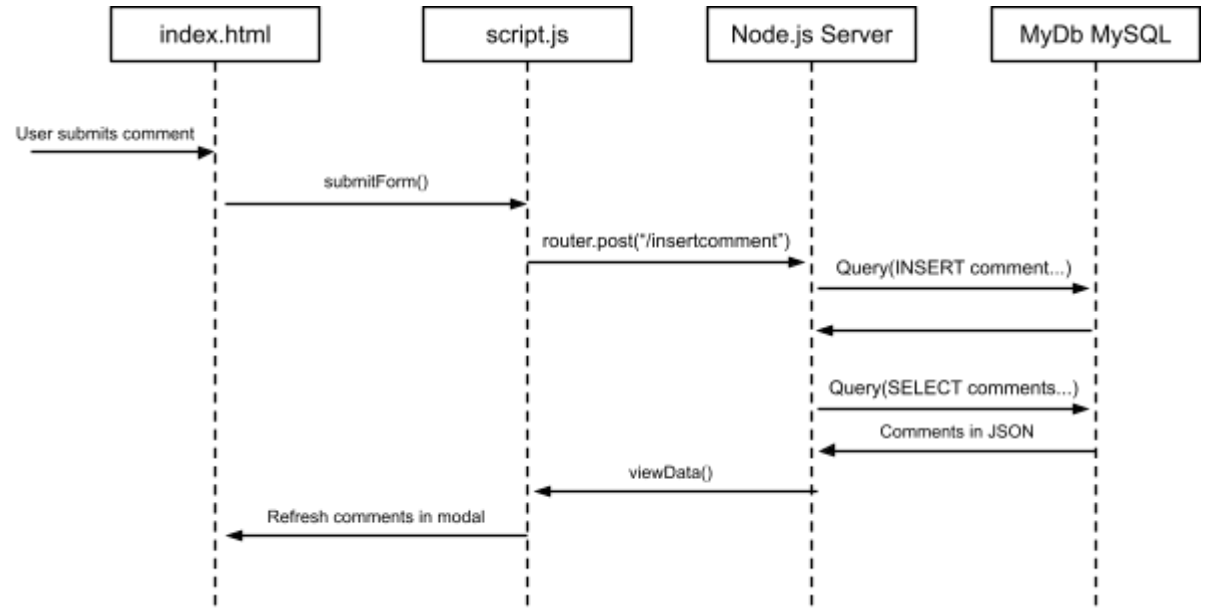
Architettura del server Node.js secondo il modello MVC



3.2. Descrizione delle risorse







3.3. Struttura del database

Il database è stato realizzato e progettato utilizzando mySQL.
A causa della semplicità dell'informazione da salvare, abbiamo deciso di utilizzare un'unica tabella.

DATABASE STRUCTURE:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
commentid	INT	☑	☑	☐	☐	☐	☐	☑	☐	
user	TINYTEXT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
apname	TINYTEXT	☐	☑	☐	☐	☐	☐	☐	☐	
text	TEXT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
rating	INT	☐	☑	☐	☐	☐	☐	☐	☐	
date	DATETIME	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP
		☐	☐	☐	☐	☐	☐	☐	☐	

EER diagram:

4. Codice

4.1. HTML

Codice per mostrare la barra di navigazione superiore, per passare da una categoria all'altra.
Ognuno di questi pulsanti chiama due funzioni:

- *CreateGetRequest:* cambia gli elementi visualizzati nella mappa
- *setColor*: cambia il colore del bottone una volta premuto, e resetta il colore degli altri

```
<div class="topnav">
   <button onclick="CreateGetRequest(negoziStoriciGetReq, drawMap, initMap);
setColor(0)" class="buttonPressed" id="0">
   Negozi Storici</button>

   <button onclick="CreateGetRequest(wifiAPGetReq, drawMap, initMap);
setColor(1)" class="buttonNotPressed" id="1">
   WiFi Access Points </button>

   <button onclick="CreateGetRequest(puntiParcoAgricoloSudMilanoGetReq, drawMap,
initMap); setColor(2)" class="buttonNotPressed" id="2">
   Parco Agricolo Sud Milano </button>

   <button onclick="CreateGetRequest(museiMilanoGetReq, drawMap,
initMap);setColor(3)" class="buttonNotPressed" id="3">Musei</button>

   <button onclick="CreateGetRequest(struttureRicettiveMilanoGetReq, drawMap,
initMap);setColor(4)" class="buttonNotPressed" id="4">Strutture
Ricettive</button>
</div>
```

Creazione del modal con all'interno recensioni e comment form, che di default è nascosto, verrà mostrato solo alla pressione di un marker sulla mappa, grazie ad una funzione in javascript

```
<div id="myModal" class="modal">
      <div class="modal-content">
```

Comment Form, per creare e inviare una recensione. I campi presenti sono:

- Nome Utente (da 5 a 20 caratteri)*
- Valutazione (da 1 a 5 stelline)*
- Testo (fino a 500 caratteri)

(I campi con "" sono obbligatori)*

```
<form id="commentForm" >
   <p>Send a comment:</p>
   <input type="text" placeholder="Enter user" id="getUser" name="user" >
   <span id=starFormWrap class="valid">
      <span id=starForm class=star>
         Rating:
         <i class="fa fa-star unchecked" id="one"></i>
         <i class="fa fa-star unchecked" id="two"></i>
         <i class="fa fa-star unchecked" id="three"></i>
         <i class="fa fa-star unchecked" id="four"></i>
          <i class="fa fa-star unchecked" id="five"></i>
      </span>
   </span>
   <span>
      <input type="button" onclick="submitForm()" value="Submit" class="button">
       <br>
      <textarea placeholder="Enter a comment" id="getComment" name="comment"
rows="5" cols="33" class="valid"></textarea>
      ...
   </span>
</form>
```

Se una delle sopracitate condizioni non viene rispettata (il controllo viene effettuato dalla funzione *validateForm()* in javascript), la recensione non verrà inviata, ma verrà al contrario stampato a schermo un avviso, e il campo errato verrà evidenziato in rosso.

4.2. CSS

Classe modal, per l'inserimento di un nuovo commento e la visualizzazione delle recensioni, hidden by default

```
.modal {
  display: none; /* Hidden by default */
  position: fixed; /* Stay in place */
  z-index: 1; /* Sit on top */
  left: 0;
  top: 0;
  width: 100%; /* Full width */
  height: 100%; /* Full height */

  background-color: rgb(0,0,0); /* Fallback color */
  background-color: rgba(0,0,0,0.4); /* Black w/ opacity */
}
```

Classe modal-content

```
.modal-content {
  color: #f2f2f2;
  background-color: #333;
    position: fixed;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
  padding: 20px;
  border: 1px solid #888;
  overflow-y: auto; /* Enable scroll if needed */
  overflow-x: hidden;
  width: 60%;
  height: 70%;
}
```

Classe topnav, per la visualizzazione della barra di navigazione superiore

```
.topnav {
  background-color: #333;
  overflow: hidden;
}

.topnav button {
  float: left;
}

.topnav button:hover {
  background-color: #ddd;
  color: black;
  cursor:pointer
}
```

4.3 API

Di seguito gli url per accesso all'endpoint SodaAPI di Regione Lombardia, per recuperare i dati necessari in formato JSON, attraverso una Ajax XMLHttpRequest

```
const negoziStoriciGetReq = {
  method: 'GET',
  //url: 'https://www.dati.lombardia.it/resource/ny3p-f7jd.json',
  url: 'https://www.dati.lombardia.it/resource/pccq-vbbq.json',
  asynchronous: true
}
const wifiAPGetReq = {
  method: 'GET',
  url: 'https://www.dati.lombardia.it/resource/ny3p-f7jd.json',
  asynchronous: true
}
const puntiParcoAgricoloSudMilanoGetReq = {
  method: 'GET',
  url: 'https://www.dati.lombardia.it/resource/ejc4-m7dx.json',
  asynchronous: true
}
const museiMilanoGetReq = {
  method: 'GET',
  url: 'https://www.dati.lombardia.it/resource/seen-dpws.json',
  asynchronous: true
}
const struttureRicettiveMilanoGetReq = {
  method: 'GET',
  url: 'https://www.dati.lombardia.it/resource/t9pt-x49i.json',
  asynchronous: true
}
```

Inclusione di uno script dalla Google Maps API per generare la mappa in modo ***asincrono (async)*** e chiamare come ***callback*** la funzione initMap con ***(callback=initMap)***, il tutto però dopo aver eseguito il parsing di tutto il documento html (***defer***)

```
<script async defer src=
"https://maps.googleapis.com/maps/api/js?key=APIKEY&callback=initMap"> </script>
```

Inclusione dello script markerclusterer dell'API di Google Maps, per la gestione e il raggruppamento di marker per livelli di zoom different

```
<script src=
"https://developers.google.com/maps/documentation/javascript/examples/markerclus
terer/markerclusterer.js"> </script>
```

4.4. Javascript

Questa funzione prende in ingresso il risultato di una query del database e costruisce la lista delle recensioni di una specifica location.

```
function viewData(data){
    ...
    jdata=JSON.parse(data);
    switchStars();          //Create interactive stars
    et apname =
document.createTextNode(document.getElementById("getAPName").value);
    function extract(item){
        commentsAmount++;
        targ=document.getElementById("entry"); //Get review and append to entry
        let wrap = document.createElement('p');
        wrap.setAttribute("class", "commentBox");
        ...
        document.createTextNode(JSON.parse(JSON.stringify(item.text)));
        let
author=document.createTextNode(JSON.parse(JSON.stringify(item.user)));
        let datetext = JSON.stringify(item.date);
        ...
        let date = document.createTextNode(datetext);
        let rating=item.rating;
        ratingSum+=rating;
        targ.appendChild(wrap);
        wrap.appendChild(authorWrap).appendChild(author);
        displayStars(wrap,rating); //Display stars for each comment
        wrap.appendChild(dateWrap).appendChild(date);
        wrap.appendChild(bodyWrap).appendChild(textCont);
    }
    jdata.forEach(extract);    //Display each comment from the db response

    let summaryWrap = document.createElement('p');
    summaryWrap.setAttribute("id","modalTitle");
    modalHeader.appendChild(summaryWrap).appendChild(apname); //Write Header
    displayStars(modalHeader,ratingSum/commentsAmount);  //Display overall rating
}
```

Questa funzione prende in ingresso un elemento HTML e un numero intero da 1 a 5. La funzione “appende” all’elemento in ingresso le stelle di valutazione correttamente illuminate.

```
function displayStars(targ,stars){
    let starsWrap=document.createElement('span');
    starsWrap.setAttribute("class", "stars");
    for(let i=1;i<=5;i++){
    if(i<=stars){
       starsWrap.insertAdjacentHTML( 'beforeend', starsLayout.starChecked);}
      else {
      starsWrap.insertAdjacentHTML( 'beforeend', starsLayout.starUnchecked);
      }
    } targ.appendChild(starsWrap); }
```

Funzione che crea una get request a un server passato per parametro nell’oggetto call, con due funzioni di callback come parametro.

```
function CreateGetRequest(call, callback1, callback2){
   let xhttpr = new XMLHttpRequest(); // LombardiaGetReq
   xhttpr.onreadystatechange = function(){
      if(xhttpr.readyState === 4){
         if(xhttpr.status === 200){
            callback1(xhttpr.response);
            callback2(xhttpr.response);
         }
         else {
            let message = document.createTextNode('Error getting the
data!');
         }
      }
   }
   xhttpr.open(call.method,call.url,call.asynchronous); // HTTP call
   xhttpr.send(); // send the request to the server
}
```

Funzione che crea una post request a un server specificato in call, con una funzione di callback come parametro

```
function CreatePostRequest(call, callback){

  let xhttpr = new XMLHttpRequest();

  xhttpr.onreadystatechange = function(){
      if(xhttpr.readyState === 4){
            if(xhttpr.status === 200){
                  callback(xhttpr.response);
                  }
            else {
            console.log("Error getting the data")
            }
            }
  }

  xhttpr.open('POST',call.url,call.asynchronous); // HTTP call

  xhttpr.send(call.parameterString); // send the request to the server
}
```

4.5. Node.js

app.js

Inclusione dei file con le routes necessarie per l'applicazione

```
var indexRouter = require('./routes/index');
var getApDataRouter = require('./routes/getapdata');
var insertCommentRouter = require('./routes/insertcomment');
```

```
app.use('/', indexRouter);
app.use('/getapdata', getApDataRouter);
app.use('/insertcomment', insertCommentRouter);
```

index.js

Funzione di risposta alla GET request della pagina principale del sito

```
router.get('/', function(req, res, next) {
   res.render('index', { title: 'Express' });
});
```

Getapname.js

Informazioni di accesso al database mySQL

```
var conn = mysql.createConnection({
    host: "localhost",
    user: "exampleuser",
    password: "******",
    database: "mydb"
});
```

Funzione che permette la connessione con il database, ed esegue il throw di errori se si presentano

```
conn.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
});
```

Funzione di risposta alla GET request (su /getapdata), che risponde con l'invio della pagina principale html.

```
router.get('/', function(req, res){
  conn.query('SELECT * FROM comments WHERE apname=? ORDER BY date DESC',
[req.query.apname],function(err, results) {
  res.send(results);
  if(err)
    console.log("Database error");
  });
});
```

insertcomment.js

Funzione di risposta alla richiesta POST (originariamente a /insertcomment), che invia al server la recensione, così da poter essere salvata nel database; la funzione chiama PopulateDb che gestisce l'inserimento nel database e successivamente invia una risposta con l'elenco delle recensioni per la location selezionata

```
router.post('/', urlencodedParser, function(req, res){
      response = { text : req.body.comment,
                   rating  : req.body.rating,
                   user  : req.body.user,
                   apname  : req.body.apname};
      PopulateDb(response);
      conn.query('SELECT * FROM comments WHERE apname=? ORDER BY date
DESC', [req.body.apname],function(err, results) {
      res.send(results);
      console.log(results);
      if(err)
          console.log("Database error");
      });
});
```

Funzione che inserisce il commento nel database

```
function PopulateDb (CommentForm){
     var sql = "INSERT INTO comments (user,apname,text,rating) VALUES ?";
     var values = [
         [CommentForm.user,
          CommentForm.apname,
          CommentForm.text,
          CommentForm.rating]];
     conn.query(sql, [values], function (err, result) {
         if (err) console.log("Database error");
          console.log("Number of records inserted: " + result.affectedRows);
     });}
```

5. Conclusioni

Siamo da sempre stati affascinati da internet e da come computer in parti opposte del mondo possano comunicare e la realizzazione di questo progetto ci ha permesso di imparare molto a riguardo. Durante lo sviluppo abbiamo affrontato molte sfide e problemi che ci hanno aiutato a comprendere di più l'argomento nonostante a volte abbiano causato rallentamenti e notevoli riscritture di codice.
Nonostante ciò siamo riusciti comunque a terminare un "proof of concept" di quella che potrebbe essere un'applicazione web di grande utilità. Di grande utilità perché raggruppa la geolocation di tutta una serie di servizi nello stesso sito web, anche non presenti sull'ormai super usato Google Maps (come gli access point wifi).
L'abbiamo definita "proof of concept" invece perchè questo progetto è tutt'altro che terminato: abbiamo infatti definito uno scheletro di base da cui poi espandersi in modo semplice e rapido, senza ulteriori complete riscritture di codice, grazie alla modularità ed espandibilità con cui l'app è stata pensata.
Future aggiunte potrebbero includere:

- Inserimento di ulteriori dataset, per rendere il servizio più completo
- Un migliorato mobile browser support
- Database differenti per ciascun dataset
- Una searchbar

Inoltre il sito è stato anche hostato su un computer e, grazie ad un IP statico e all'apertura e port forwarding di una porta, siamo riusciti anche ad accedervi da diversi dispositivi, sia nella rete locale, sia attraverso internet.