PPM Web App

Niccolò Parlanti

April 2022

Segue una breve ma il piu' possibile esaustiva presentazione della Web app Fast Ticket, partendo dalle funzionalità e dai casi d'uso, esponendo anche alcuni dettagli implementativi.

1 Abstract

Si propone di sviluppare una applicazione web che possa agevolare l'utente nella ricerca di biglietterie autorizzate per la vendita di titoli di viaggio per mezzi pubblici nell'area fiorentina, oltre che la ricerca di fermate di bus e tramvia.
La web app si propone di presentare una interfaccia semplice ed immediata per gli utenti.

2 Utilizzo

2.1 Connessione

Deve essere possibile caricare l'applicazione sotto una copertura 3g-4g in quanto si prevede il suo utilizzo in ambiente urbano con una copertura decente; E' necessario l'utilizzo del servizio di localizzazione Gps.

2.2 Caso d'uso

- Il sito web è caricato
- viene richiesto consenso di accedere a posizione (necessario nei moderni Sistemi Operativi)
- l'utente da il consenso, viene visualizzata una mappa interattiva centrata nella posizione Gps
- viene automaticamente visualizzato il percorso verso la biglietteria più vicina
- sullla mappa è presente un bottone che serve a selezionare i layer da mostrare sulla mappa

- nella mappa i punti di interesse sono rappresentati da un Marker, l'utente sceglie un Marker
- cliccando sul marker l'utente può consultare alcuni informazioni sul punto di interesse
- l'utente può avviare la navigazione verso il punto di interesse
- l'utente verra guidato verso il punto di interesse seguendo le indicazioni che compaiono sulla mappa

3 Funzionalità

Vengono individuate le seguenti funzionalit'a da implementare :

- **Caricamento della mappa Leaflet**
  Inizialmente si importa leaflet.css

```
<script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"
integrity="sha512-XQoYMqMTK8LvdxXYG3nZ448hOEQiglfqkJs1NOQV44cW
nUrBc8PkAOcXy20w0vlaXaVUearIOBhiXZ5V3ynxwA=="
crossorigin=""></script>
```

  si crea una variabile baseMap ed un layer L.tileLayer collegato con un url

```
var mymap = L.map('map', {
 center: [43.78, 11.23],
 zoom: 14
});


//Mappa base OpenstreetMap
 var baseMap = L.tileLayer('https://api.mapbox.com/styles/v1/
 {id}/tiles/{z}/{x}/{y}?access_token={accessToken}',
 {attribution: 'Map data &copy; <a href="https//www.
 openstreetmap.org/copyright">OpenStreetMap</a> contributors,
 Imagery © <a href="https://www.mapbox.com/">Mapbox</a>',
    maxZoom: 18,
    minZoom: 5,
    id: 'mapbox/streets-v11',
    tileSize: 512,
    zoomOffset: -1,
    accessToken: 'myaccestoken'
}).addTo(mymap);
```

- **Creazione dei markers**
  Si creano i marker rispettivamente per:
  - Biglietterie:

```
var ticketIcon = L.icon({
 iconUrl: "images/ticket.jpg",
 iconSize: [30, 30],
 iconAnchor: [25, 40],
 popupAnchor: [0, -30],
});
```

  - Fermate della tramvia:

```
var tramIcon = L.icon({
 iconUrl: "images/tram.jpg",
 iconSize: [30, 30],
 iconAnchor: [25, 40],
 popupAnchor: [0, -30],
});
```

  - Fermate del bus:

```
var stopsIcon = L.icon({
 iconUrl: "images/stops.png",
 iconSize: [30, 30],
 iconAnchor: [25, 40],
 popupAnchor: [0, -30],
});
```

  - Icona utente:

```
var myicon = L.icon({
 iconUrl: "images/icon.jpg",
 iconSize: [50, 50],
 iconAnchor: [25, 40],
 popupAnchor: [0, -30],
});
```

  I marker verrano poi caricati sulla mappa e uniti in dei MarkerCluster-Group importando

```
<link rel="stylesheet" href="../dist/MarkerCluster.css" />
```

- **Dati utilizzati**

  I dati che utilizzo sono stati presi da tre diversi database, rispettivamente per le fermate del bus nell'area fiorentina, le biglietterie e le fermate della tramvia.

  - http://dati.toscana.it/dataset/rt-oraritb
  - https://opendata.comune.fi.it/?q=metarepo/datasetinfoid=07d7d15b-b121-48e0-87c2-53adac0a5c2a
  - https://opendata.comune.fi.it/?q=metarepo/datasetinfoid=607b8031-252b-4442-950a-7b4452d9db8f

  Tutti i file presenti sono stati poi convertiti in GeoJson per poter essere utilizzati sulla mappa Leaflet. In particolare i dati delle fermate del bus sono stati generati unendo con criteri specifici alcuni file di tipo .txt, a cui poi è stato applicato un parse a CSV (realizzato in python).

- **Geolocalizzazione**

  La geolocalizzazione è fondamentale, dato che l'intera web app è basata sul conoscere la posizione dell'utente per poter poi consigliare la biglietteria più vicina, e dare indicazioni verso i punti di interesse.

  La funzione che ho utilizzato controlla se la geolocalizzazione è consentita sul browser, per poi aggiornare la stessa ogni 2000 ms

```
    //Geolocalizzazione
if (!navigator.geolocation){
 console.log("Browser does not support geolocation");
}else {
 setInterval(() => {
 navigator.geolocation.getCurrentPosition(getPosition)
 }, 2000);
}
```

  La funzione getPosition è quella che effettivamente controlla la posizione e gestisce lo spostamento dell'icona dell'utente sulla mappa.

```
var marker;
check = true

function getPosition(position){
 var lat = (position.coords.latitude) ;
 var lng = (position.coords.longitude);
 var accuracy = position.coords.accuracy;
```

```
 if (check){
 marker = L.marker([lat,lng], {icon: myicon}).bindPopup("I'm here!")
 .openPopup().addTo(mymap);
 check = false;

 }
 else{
 marker.slideTo( [lat, lng], {
 duration: 1700
});
 }

};
```

Per spostare l'icona in maniera fluida utilizzo la funzione slideTo, utilizzabile importando

```
<script src='https://unpkg.com/leaflet.marker.slideto@0.2.0/
Leaflet.Marker.SlideTo.js'></script>
```

- **Funzione riposizionamento utente**

  Tramite un bottone presente sulla mappa è possibile riposizionare la mappa centrata sulla posizione dell'utente.

```
<button id = "buttonPosition" onclick ="backToPosition()" title="Riposiziona" >
 <img src="images/geolocation.jpg" width="30" height="30">
 </button>
```

  La funzione BackToPosition utilizza le coordinate delll'utente in tempo reale, e usa la funzione flyTo() per ragiungere quella posizione sulla mappa.

```
function backToPosition(){
 if(navigator.geolocation){
     navigator.geolocation.getCurrentPosition(function(position){
     var lat = (position.coords.latitude) ;
     var lng = (position.coords.longitude);
     var latlng = L.latLng(lat, lng);
     mymap.flyTo([lat,lng], 16);
     })
   }
}
```

- **Gestione del Routing**

  Cliccando su ogni punto di interesse sulla mappa è possibile cliccare il bottone "Portami Qui"

```
"<button type='button' class='btn btn-outline-primary' id = 'getDirection'
onclick = 'createRoute(["+feature.geometry.coordinates+"])'>
Portami Qui</button>"
```

  Cliccando sul bottone, questo richiama la funzione createRoute()

```
//Routing

        var Route,latUser,lngUser;

        function createRoute(coordinates) {

         if(navigator.geolocation){
             navigator.geolocation.getCurrentPosition(function(position){
             latUser = (position.coords.latitude) ;
             lngUser = (position.coords.longitude);
             WaitAndRoute();
             })
         }else console.log("geolocation does not work");

        lat = coordinates[1];
        lng = coordinates[0];

         function WaitAndRoute(){
         if(Route){
             mymap.removeControl(Route);
         }

         Route = L.Routing.control({
             waypoints: [
                 L.latLng(latUser, lngUser),
                 L.latLng(lat, lng)
             ],
             routeWhileDragging: true,
             show: true
         }).addTo(mymap);

         mymap.closePopup();

         }
```

```
};
```

Questa funzione, dopo aver salvato le coordinate reali dell'utente, richiama la funzione WaitAndRoute(), che dopo aver controllato la presenza di un altro routing attivo, eliminandolo, crea una nuova Route utilizzando i dati in tempo reale dell'utente e quelli del punto di interesse.

- **Routing verso il negozio più vicino**

  All'apertura della pagina, la web app calcola automaticamente il percorso verso la biglietteria più vicina alla posizione in tempo reale dell'utente.

```
function getNearest(){
     if(navigator.geolocation){
         navigator.geolocation.getCurrentPosition(function(position){
         var latUser = (position.coords.latitude) ;
         var lngUser = (position.coords.longitude);
         var shop = L.geoJson(atafLayer);
         shopIndex = leafletKnn(shop);
         var nearestResult = shopIndex.nearest([lngUser,latUser], 1)[0];
         lng = nearestResult.lon;
         lat = nearestResult.lat;
         createRoute([lng, lat])

     })
     }
}
getNearest()
```

- **Cancellazione del percorso visualizzato sulla mappa**

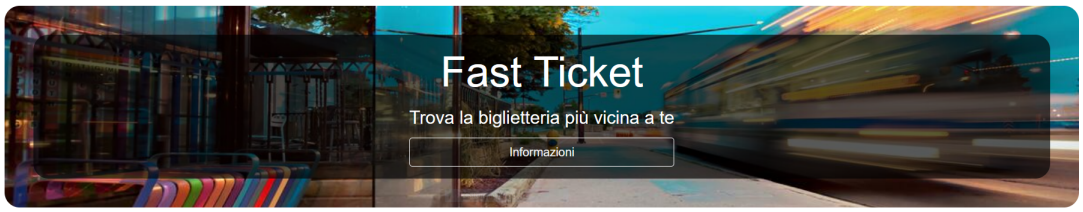
  Tramite un bottone presente sulla mappa è possibile eliminare il percorso visualizzato sulla mappa, eliminado di fatto la route precedentemente creata.

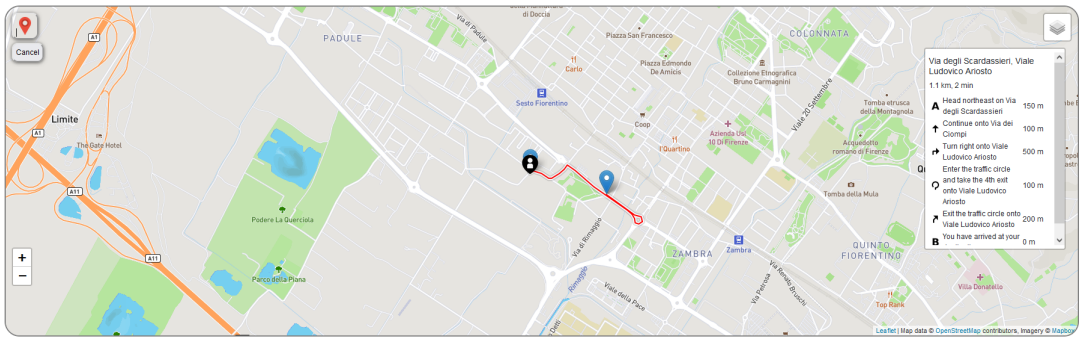
```
<button id="cancelRoute" onclick ="removeRoute()"
title="Cancella itinerario"> Cancel </button>
```

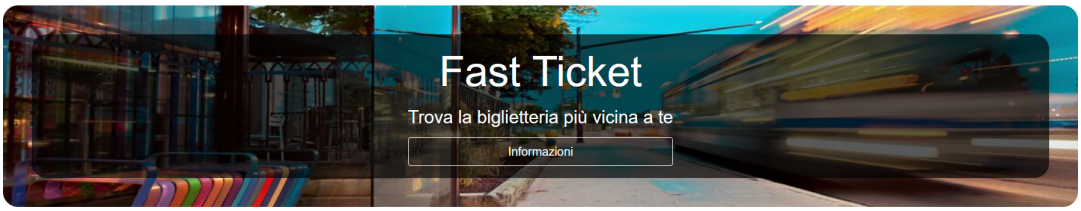
  La funzione controlla la presenza di una Route, ed in caso ci fosse una Route attiva, procede ad aliminarla dalla mappa.

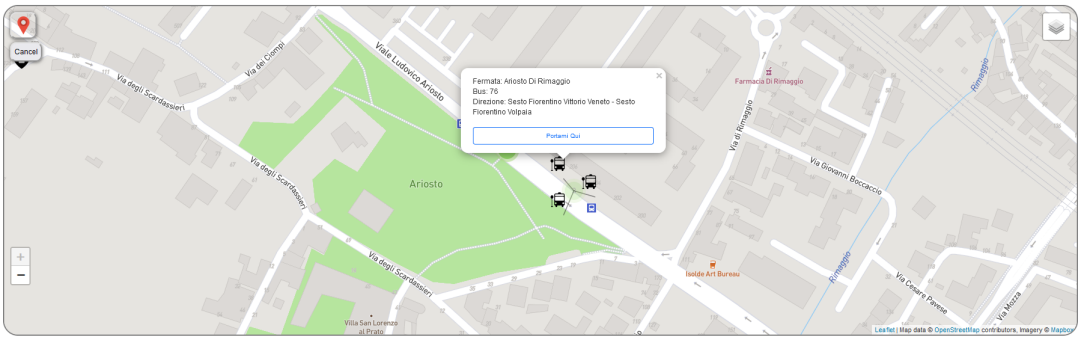
```
function removeRoute(){
     if(Route){
         mymap.removeControl(Route);
     }else
         console.log("No Route");
    }
```

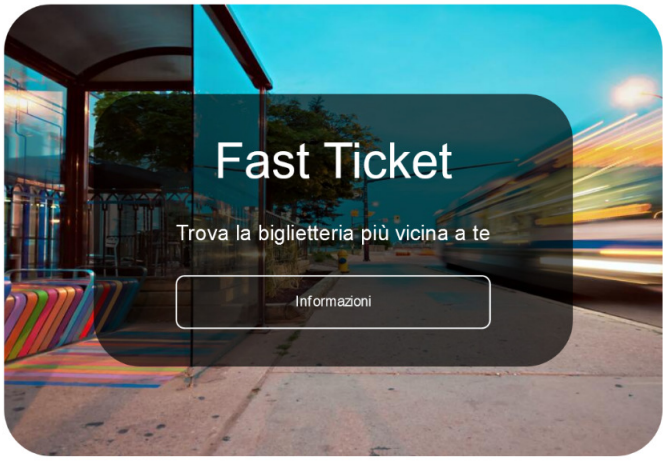
4 Screenshot dell'applicazione

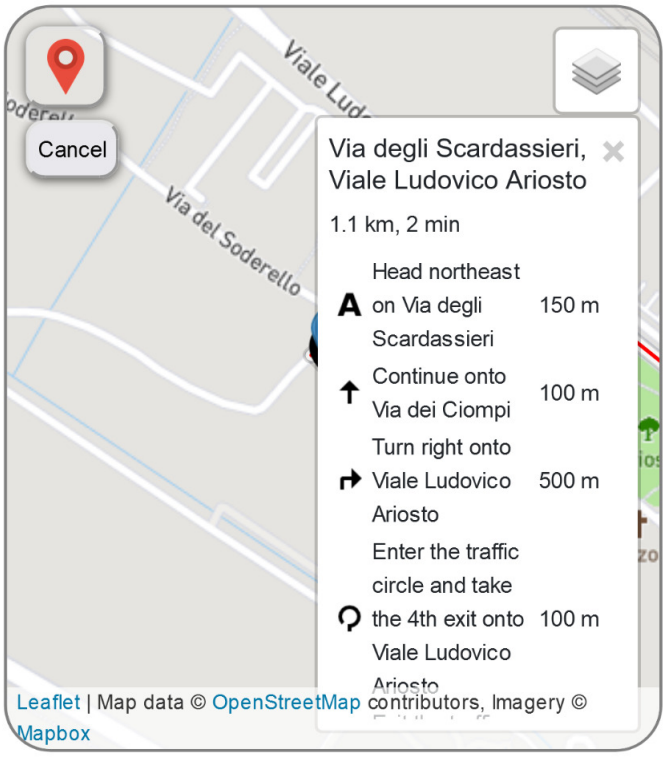









Fast Ticket
Trova la biglietteria più vicina a te
Informazioni


Cancel
Via del Soderello
Via degli Scardassieri, Viale Ludovico Ariosto
1.1 km, 2 min
Head northeast on Via degli Scardassieri 150 m
Continue onto Via dei Ciompi 100 m
Turn right onto Viale Ludovico Ariosto 500 m
Enter the traffic circle and take the 4th exit onto Viale Ludovico Ariosto 100 m
Leaflet | Map data © OpenStreetMap contributors, Imagery © Mapbox