HÖGSKOLAN
I GÄVLE

Karttjänster med Open Source, 5 HP VT19

Projektet är en webbaserad kartapplikation för vandringsled i Sverige

av
Ferhat Sevim

Akademin för teknik och miljö
Högskolan i Gävle

S-801 76 Gävle, Sweden

Innehållsförteckning

Sammanfattning

Det här projektet handlar om att skapa en webbaserad kartapplikation där användaren kan se närmaste hotell, vandrarhem och restauranger till en vald vandringsled. Användaren kan även se information från andra platsers namn och koordinater. **Preclicked** fungerar som när man klickar på kartan som visar före något annat händer. **Hovering** ska visa information om alla lager som finns på kartan. **Buffer** skapar en polygon runt den valda vandringsleden. **Doubleclick** används för att gå tillbaka till den ursprungliga kartan efter resultat. **ZoomTo** gör att man går närmare till den valda vandringsleden. ExtJS är ett JavaScript ramverk för att skapa webbsida med färdigställde rutor.

Introduktion

Jag använder **Geoserver** och **Leaflet** för att skapa en webbaserad kartapplikation. Jag kommer att använda vektor data för projektet och raster data för kartbild. Jag använder olika bibliotek som är **Turfjs** och **Leaflet-GeometryUtil** för att skapa buffert runt den valda linjen och få fram vilka punkter som är inne bufferten. Jag testar också att integrera **Leaflet** med **ExtJS**.

Målet är att kunna visa närmaste platser till den valda vandringsleden.

Följande frågeställningarna är centrala:

- Vad är Geoserver?
- Vad är Leaflet?
- Vad är Turfjs?
- Vad är Leaflet-GeometryUtil?
- Vad är ExtJS?

Metoder

Jag laddar ner öppna data i formatet shapefil för vandringsleder i *Miljödataportalen* från *Naturvårdsverket (Naturvårdsverket, u.å.)*. Jag utnyttjar shapefilen points från inlämningsuppgifters data. Jag hämtar punkter för hotell, vandrarhem och restauranger. Jag söker även på *geodatakatalogen* i *Länstrylsen* och laddar ner vandrarhem (Länstyrelsen, u.å.). Jag skapar en egen shapefil för vandrarhem och sätter ihop de två shapefilerna i **QGIS** eftersom dessa shapefiler har olika fältnamn. Jag skapar geojson filer för shapefiler i **QGIS**. Jag ändrar filändelserna till .js och skapar ett variabel namn som filnamn i filen för att det ska kunna fungera med Leaflet. Jag anropar mina .js filer under head tag på följande sätt ”`<script src="projekt_data/hikings.js"></script>`” i html filen sedan skapar jag Geojson layers i JavaScript filen och olika stilar för att visa.

En av optioner till GeoJSON är **pointToLayer** vilket är en funktion som definierar hur punkt data passeras till GeoJSON punkt feature och sina koordinater sedan returnerar en cirkel markering med sin färg, fyllnadsfärg och radius. Denna funktionen gäller bara för punkter. Linjer använder sig en option som är **style** för att sätta färg och få det synas på kartan.

Den andra optionen till GeoJSON är **onEachFeature** vilket är en funktion som körs för varje skapade feature. Den är rätt användbar för att kunna använda event, popup, hovering och andra funktioner till feature. Resultat är i **figur 1**.

Jag skapar **tooltip** att se olika layers namn när jag flyttar över (**hovering**) punkterna och linjerna och **popup** för att visa koordinaterna för den valda punkten i en ruta på kartan i **figur 2. Preclicked** är en event som startar innan användare klickar på kartan. Ibland är det användbart när man vill något att hända på klick innan något annat existeras.

Jag använder metoder **mymap.fitBounds()** och **layer.getBounds()** för att zooma in (**zoomTo**) den valda vandringsleden och du får en buffert polygon runt den i **figur 3**. Bufferten skapade jag med hjälp av metoden **turf.buffer()** och använder metoden **turf.within()** för att veta vilka punkter som ligger i in bufferten. Dessa är metoder i **Turfjs**. Eftersom **turf.within()** endast läser från ett GeoJSON lager, så sparar jag längden av features i bufferten till en Array sedan räknar jag med hjälp av en lambda funktion. Nu använder jag en metod av **Leaflet-GeometryUtil**. Metoden **nClosestLayers()** sparar antal närmaste lager (utnyttjar längden av features i bufferten) från den valda positionen till en list och returnerar det. Jag använder for loop för att skriva ut. Efter figur 3 klickar du på den valda vandringsleden en gång till, får du se en polygon buffert och se de närmaste punkterna nära den valda vandringsleden. Du kan även se vilka andra punkter som är nära till punkten du väljer en punkt inuti bufferten. Resultat visas i högre sidan under rubriken information. Det innehåller typen, namnet och avståndet i pixlar och meter med avrundning till 2 decimaler med hjälp av metoden **toFixed()** i **figur 4.** En metod **capitalize()**; jag har skapat för att få första bokstaven med stor bokstav eftersom fältnamnet har små bokstäver. För att gå tillbaka ursprungliga kartan dubbelklickar jag på kartan annars man inte kan se andra vandringsleder utan den valda vandringsleden.

Den sista metoden är att skapa en kontroller (**layers control**) för att kunna välja olika bas lager och GeoJSON lager och den är överst till höger i hörnet på kartan **figur 5**. Det blir radio knapp för bas lager och checkbox för GeoJSON lager i kontroller så du kan släcka eller tända olika lager. Under bilagor kan du se de olika osm bas lager **figur 6,7,8**. Jag har också fixat att div tag kan ha scroller i html, så jag skall kunna bläddra ner. Jag har även skapat JavaScript dokumentation där kan du läsa korta förklaringar om koder (JSDoc, 2017).

Att använda **ExtJS** med **Leaflet**, använder jag **Ext.define()**. Det används för definiera klasser i **ExtJS**.
Så skapar jag en klass och döper till **projekt_data.LmapPanel** som ärver **Ext.panel.Panel**. Jag lägger till en class alias och config variable för karta referensen. Jag börjar skriva koder för **Leaflet** efter **afterRenderer**. Jag skapar en till JavaScript fil och döper det till **projekt_leaflet_extjs.js.** Jag konfigurerar och lägger till nödvändiga filer sedan **Ext.application()** för att köra applikationen och **Ext.create()** för att skapa och visa paneller. Metoden updateMapLocation() är för att få platsen när du klickar på knappar i tabb listan på överst av webbsidan. Resultat kan du se i bilagor figur 9-10.

Resultat

Jag är helt nöjd med projektet och webbtjänsten. Jag fick kunna visa närmaste punkter till en vandringsled. Det ger mycket information mellan vandringsled och hotell, vandrarhem och restauranger. Det har mycket funktionalitet och är informerande till användaren. Det var mest komplicerade parten att kunna integrera **ExtJS** med **Leaflet**. Det var mycket att läsa och testa.

Geoserver är öppen källkod i server för att dela geospatial data (Geoserver, 2019).
Leaflet är en öppen källkod JavaScript baserat bibliotek för mobilvänliga interaktiva kartor (Leafletjs, 2018).
Turfjs är ett JavaScript bibliotek för avancerad geospatial analys för webbläsare och node.js (Turfjs, u.å.).
Leaflet-GeometryUtil är ett verktyg för avstånd och linjär referens (Leaflet-Geometry-Util, 2016).
ExtJS är ett JavaScript ramverk som ger många inbyggda komponenter till en ruta som är mycket anpassningsbara (Sencha, 2013).

Analys

Projektet handlar om att användaren ska kunna hitta närmaste platser från en vandringsled. Det vill visa att användaren får veta vilka platser är nära på vandringsleder utan att välja en viss vandringsled. Man ska kolla på kartan vilka vandringsleder har olika punkter i närheten. Man ser även avståndet mellan punkt och linje.
Kartan med **ExtJS** kan man klicka på knappen Gävle eller Uppsala för att se närmare för vandringsled, vandrarhem, hotell och restauranger.

Diskussion

Projektet blev bra övning och har undersökt många saker samtidigt övat det. Det blev små fel men jag löste dessa med if else satser. Till exempel när jag valde en vandringsled, kunde inte det visa punkter som hade null.

Det kunde ha varit bättre om jag kunde lägga till andra punkt data. Till exempel caféer, bensinmackar, buss- och tågstationer. Det saknas också information för vissa linjer och punkter. Det är nackdelar med linjer. Man behöver zooma in för att se bättre annars blir det väldigt svårt att avgöra vilken vandringsled man ska välja och när man linjerna tjockare, ser inte det jättefint ut på kartan.

Det här projektet har likadana uppgifter som jag har haft på mina andra projektarbete.

Jag har också övat att skapa **Leaflet** med **ExtJS**. Jag har fått att lösa en del men det funkade inte att få popup när jag klickade på kartan. Till slut har jag fått lösa det. Det fungerar utmärkt nu.

Referenser

Geoserver. (2019). GeoServer is an Open Source Server for Sharing Geospatial Data. Hämtad 2019-01-24 från http://geoserver.org/

Leafletjs. (2018). A JavaScript Library for Interactive Maps. Hämtad 2019-04-30 från https://leafletjs.com/

Leaflet-GeometryUtil. (2016). Leaflet Geometry Utilities for Distances and Linear Referencing. Hämtad 2019-04-30 från http://makinacorpus.github.io/Leaflet.GeometryUtil/index.html

Länsstyrelsen. (u.å.). Geodatakatalogen. Hämtad 2019-04-03 från https://ext-geodatakatalog.lansstyrelsen.se/GeodataKatalogen/

Naturvårdsverket. (u.å.). Kartor, Data och Rapporter om Natur och Miljö. Hämtad 2019-04-03 från http://mdp.vic-metria.nu/miljodataportalen/

Npmjs. (2017). JSDoc 3 An API Documentation Generator for JavaScript. Hämtad 2019-04-30 från https://www.npmjs.com/package/jsdoc

Plnkr. (u.å.). ExtJS 4 Panel /w Leaflet. Hämtad 2019-04-05 från http://embed.plnkr.co/Fo9nOYH3KUHRqw4Soq6F/

Sencha. (2013). Integrating ExtJS with 3rd Party Libraries. Hämtad 201-04-07 från https://www.sencha.com/blog/integrating-ext-js-with-3rd-party-libraries-2/

Turfjs. (u.å.). Advanced Geospatial Analysis. Hämtad 2019-04-30 från https://turfjs.org/

Bilagor

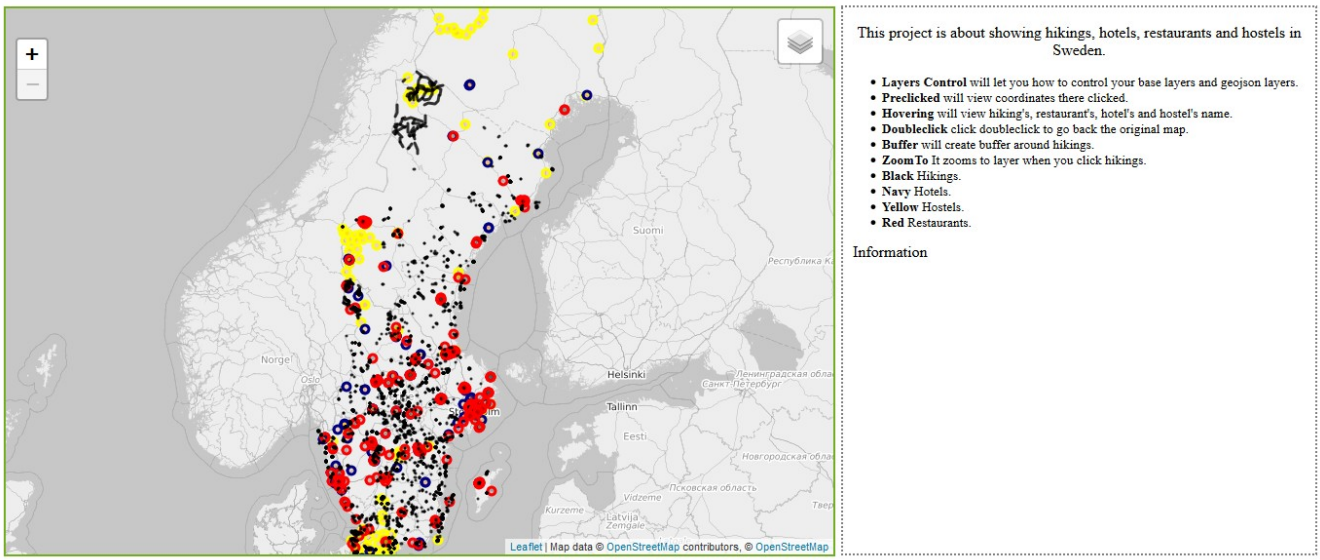

Figure 1: Start map and description

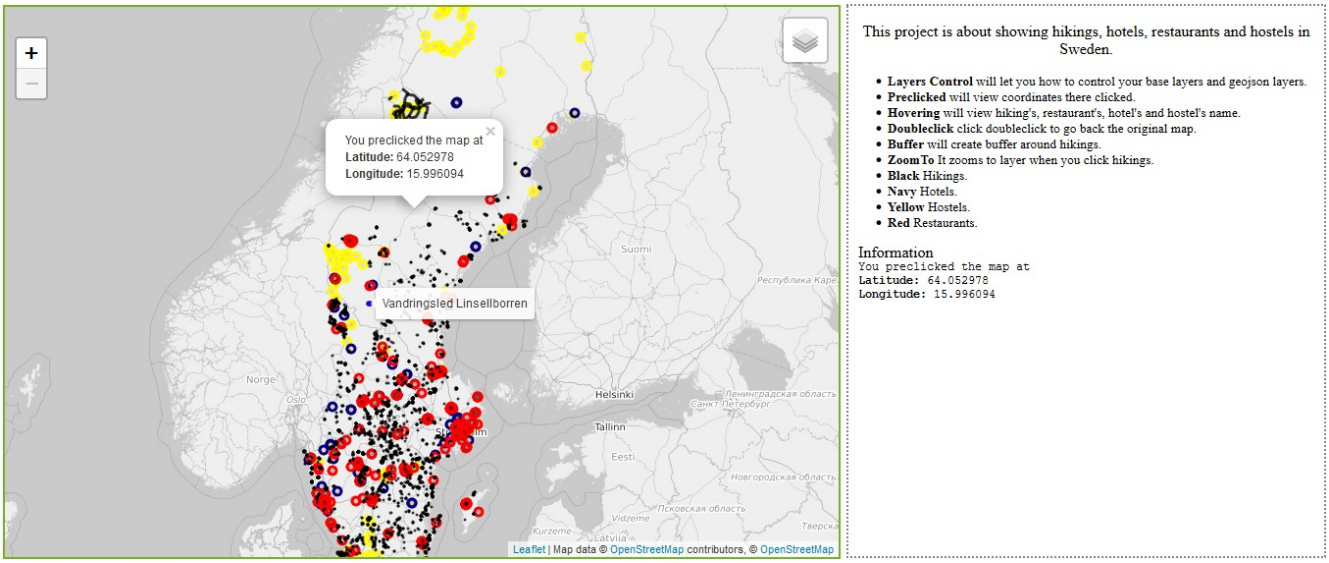

Figure 2: Hovering and Popup

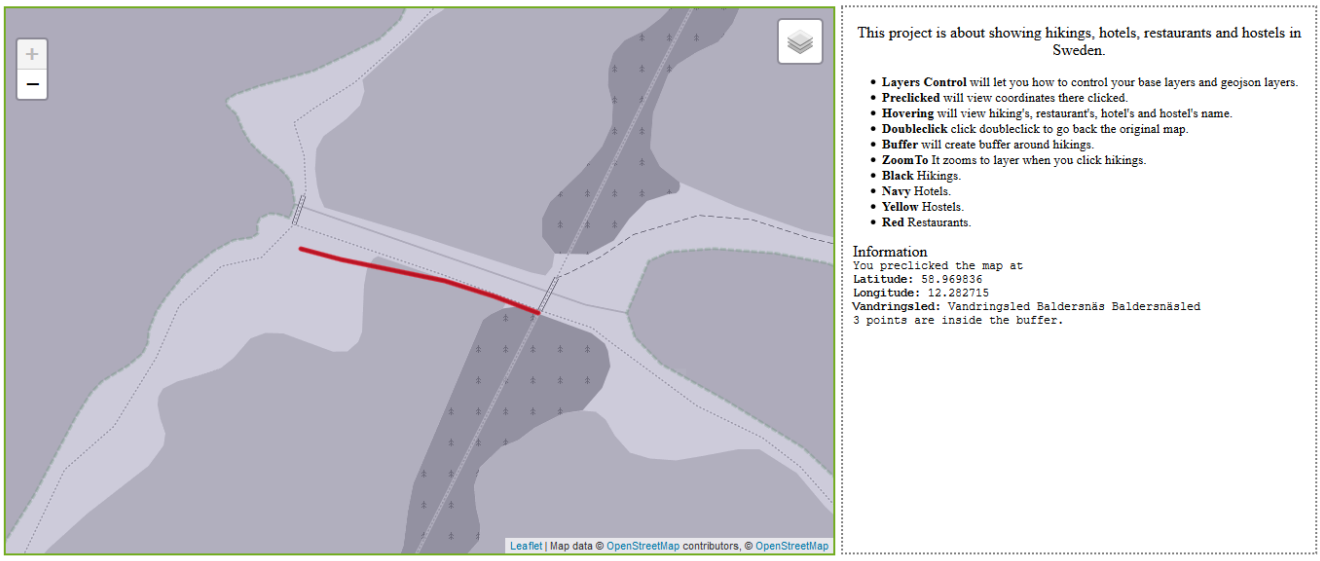

Figure 3: Chosen hiking

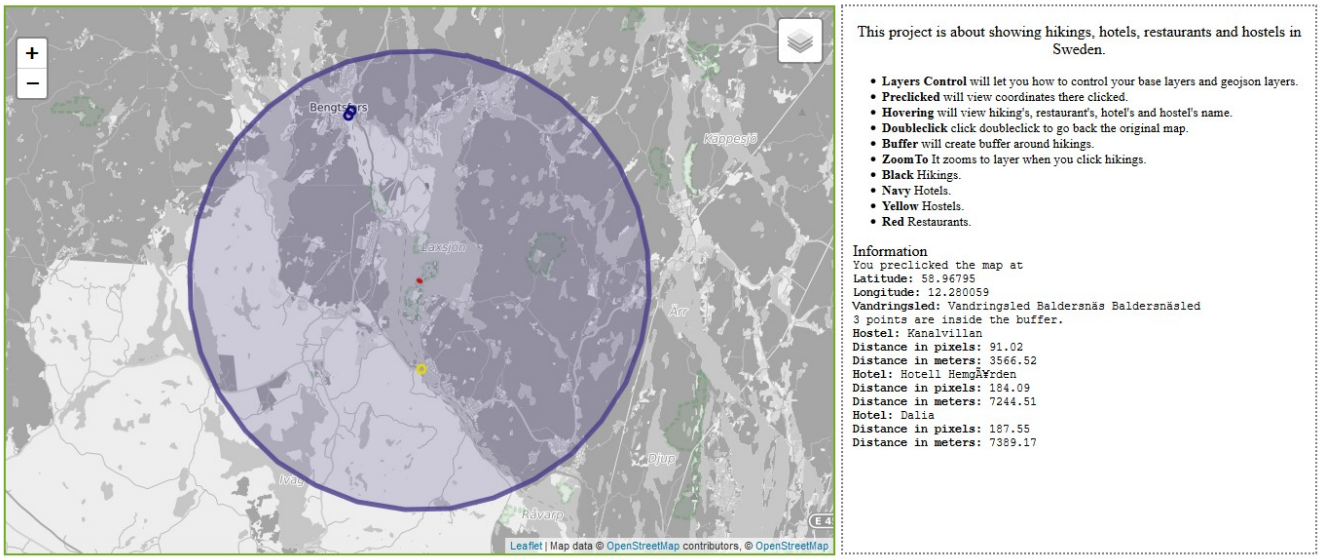

Figure 4: The result of chosen hiking

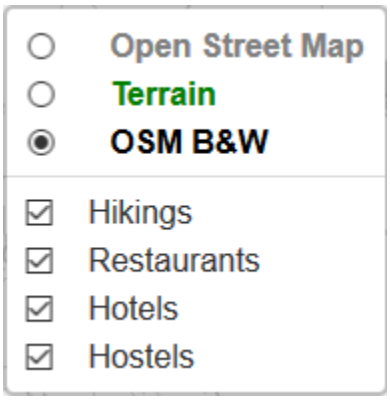

Figure 5: Control options in the topright corner on the map

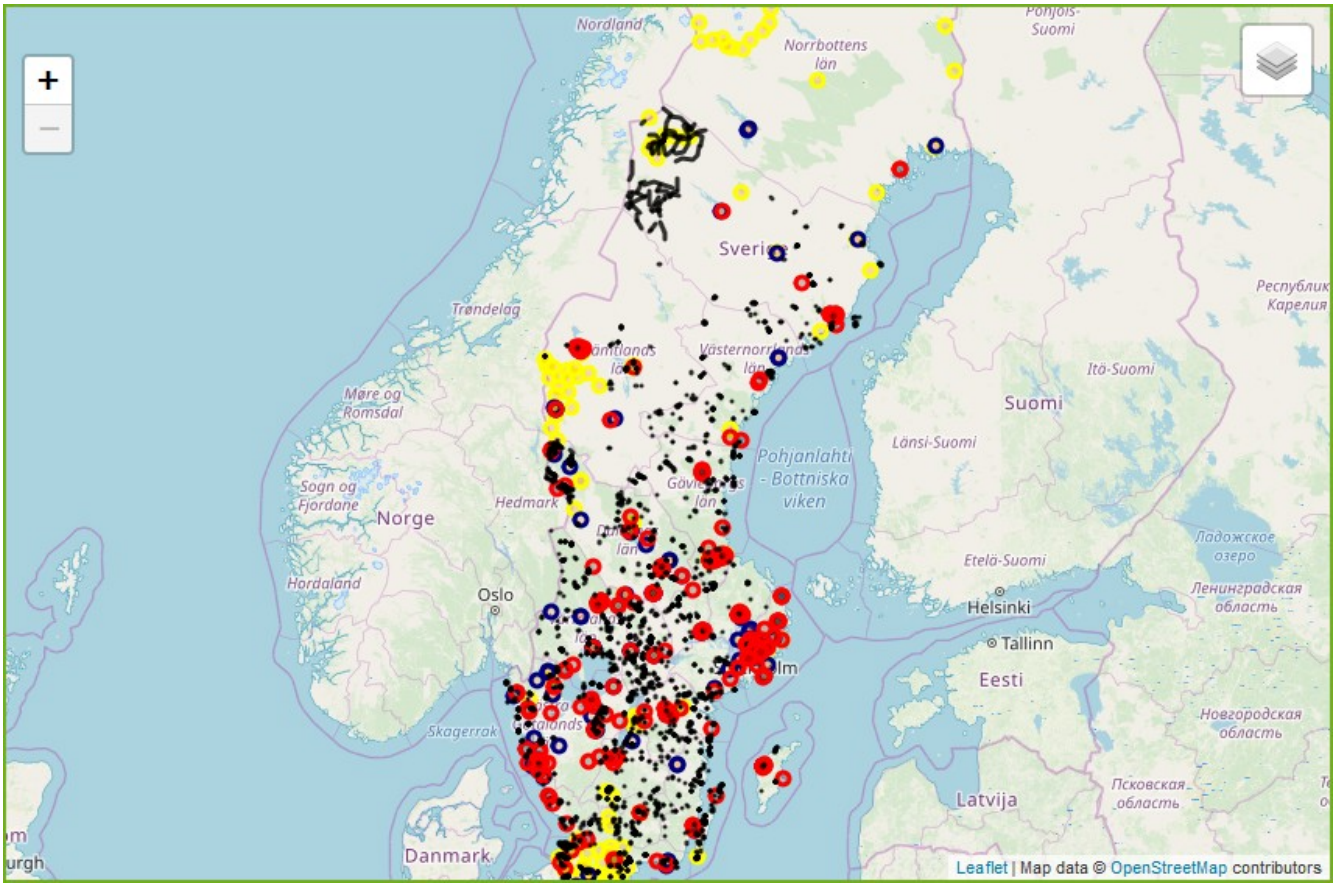

Figure 6: OSM Base Layer

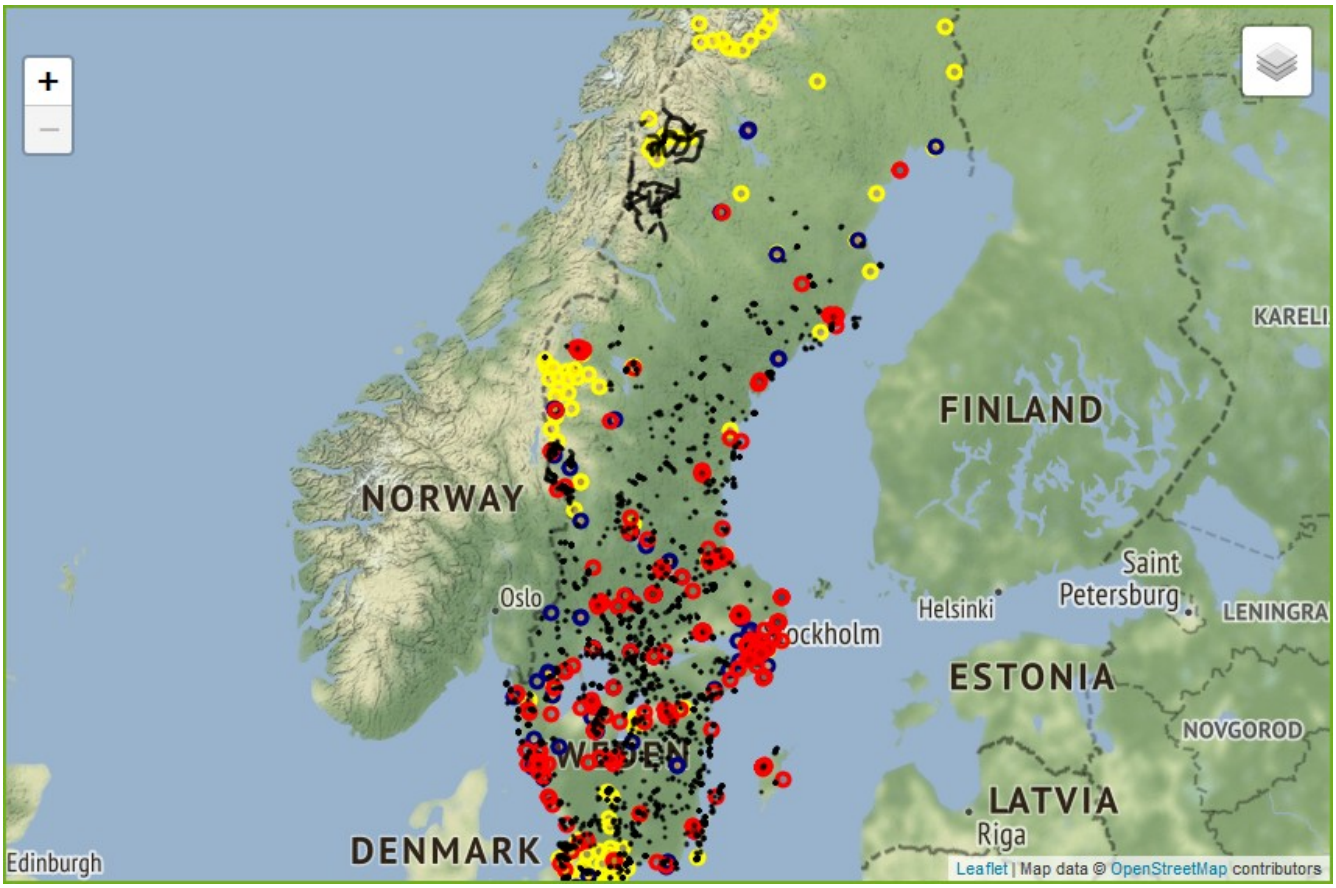

Figure 7: Terrain Base Layer

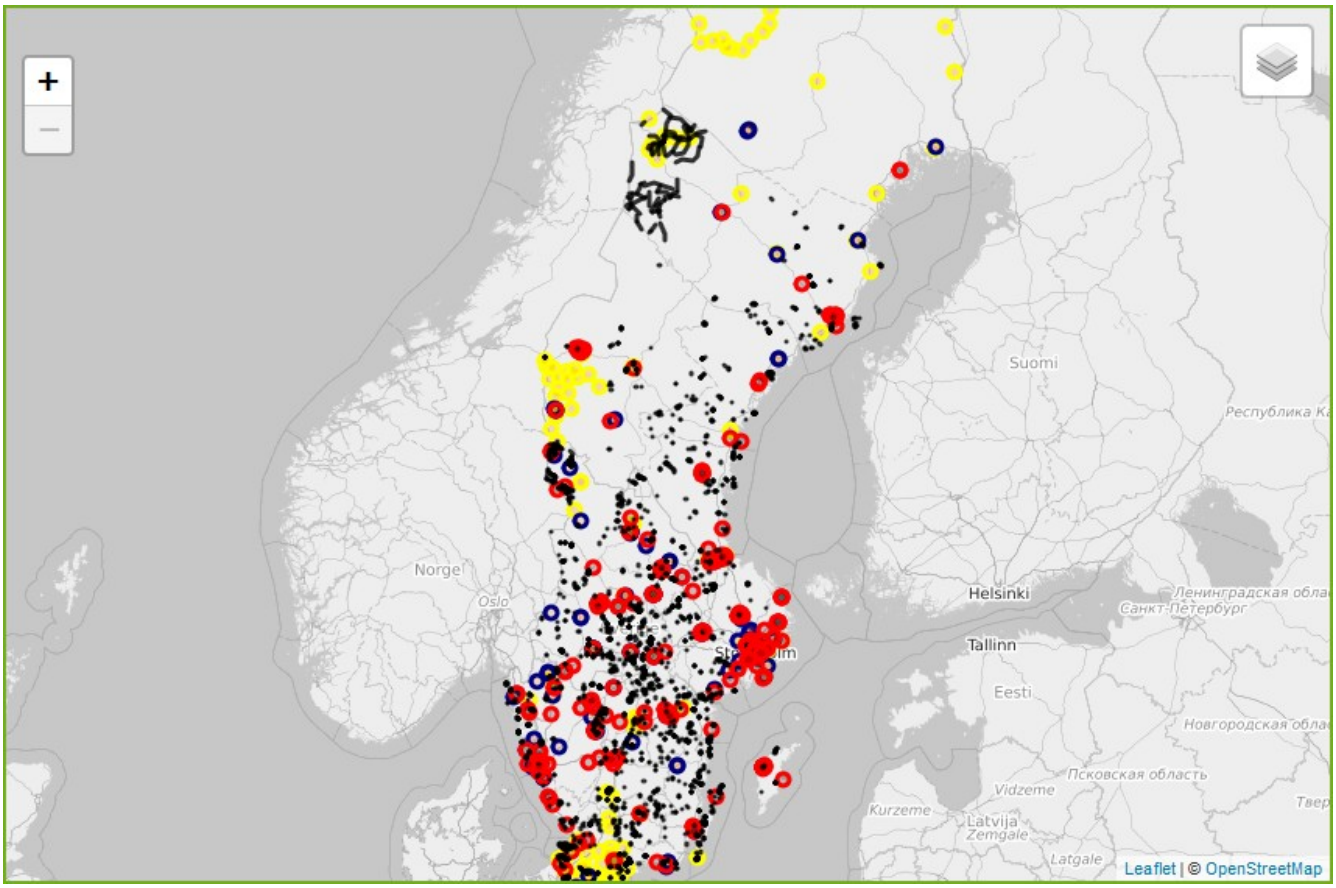

Figure 8: OSM Black & White Base Layer

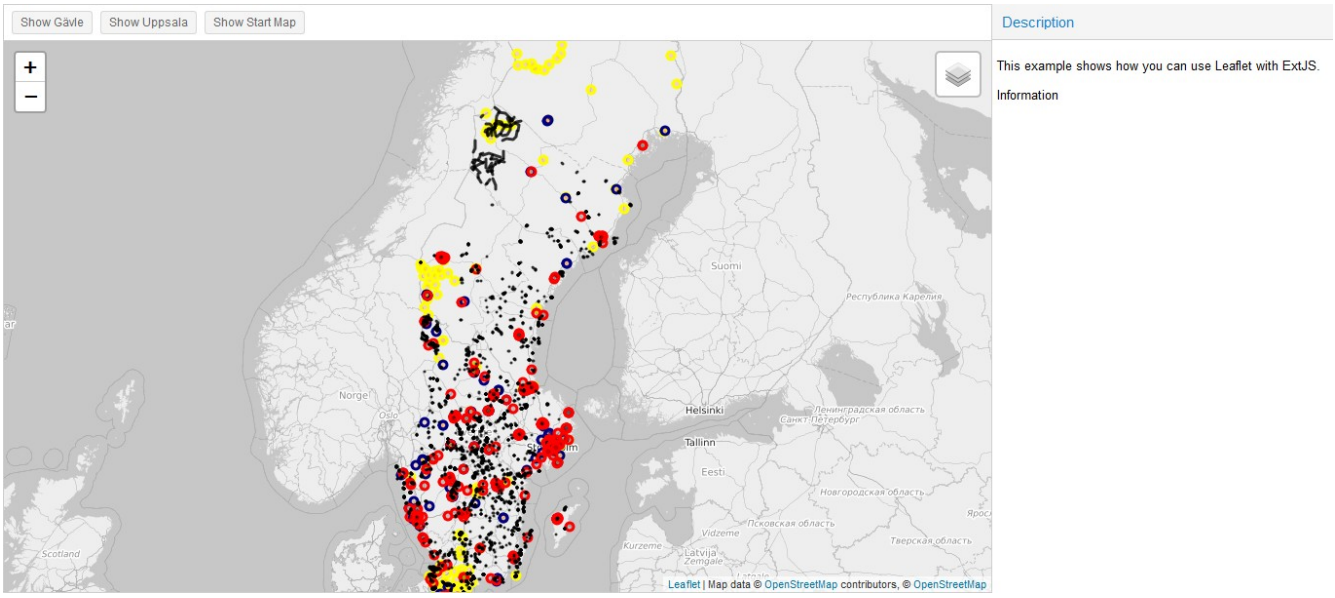

Figure 9: Example of Leaflet with ExtJS

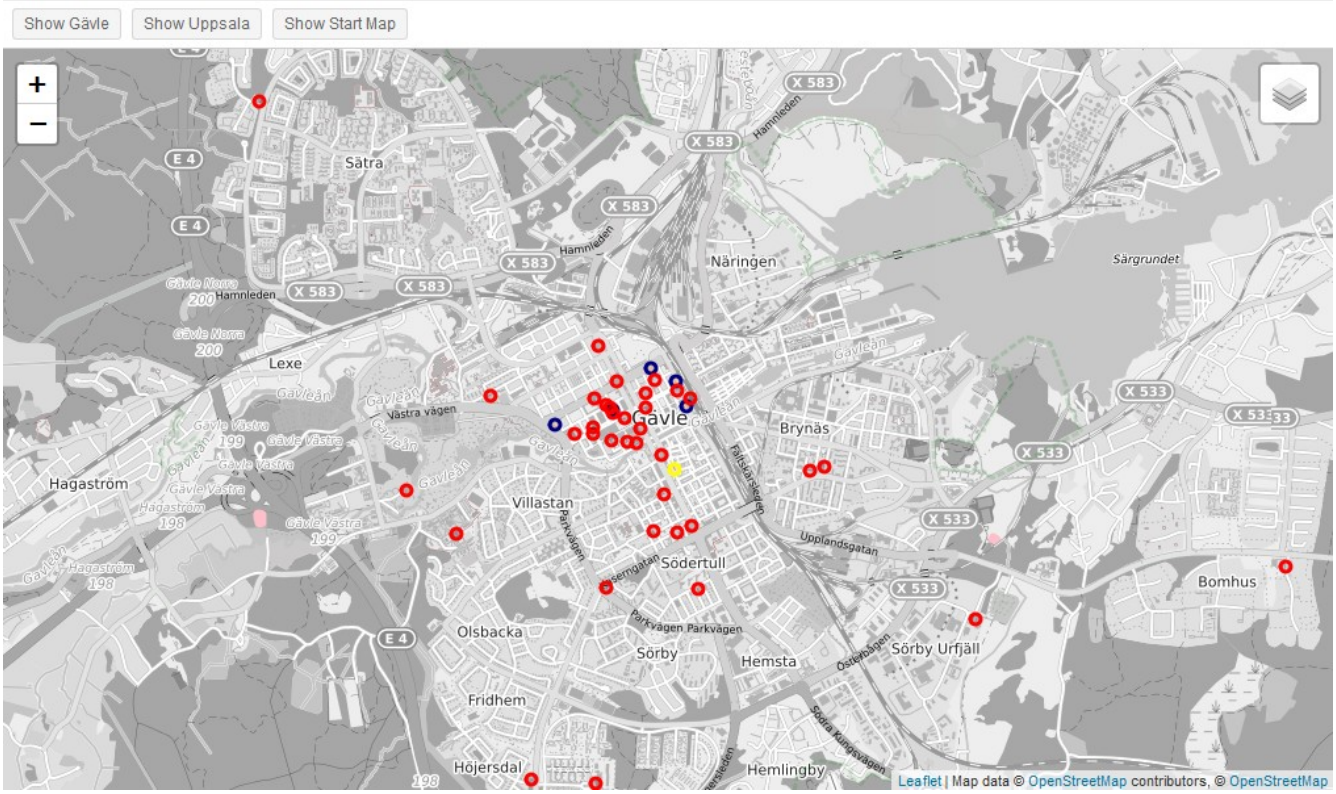

Figure 10: Knappar som zoomar till positionen och knapp för start karta