Project 2 Report

Problem

Zion National Park hired me to build a mobile web app to help provide visitors with information before they arrive and make the experience of visiting the park better. The app is to be map centered, include the ability to search for features and locations within the park, be focused on themes of safety, accessibility and recreation, and have the ability for users to contribute to a database.

Data

Data for the project was mostly sourced from the National Parks Service online data store. Searching there, I was able to download shapefiles for the Park Boundary, Trails, and Trailheads. Relevant information in these layers include trailhead *names*, trail *length*, and whether a trail *allows horses*. All unnecessary attributes were deleted from the data to keep file sizes and loading times low. Additionally, I was able to create a point layer of suggested viewpoints for users to contribute to while they are hiking.

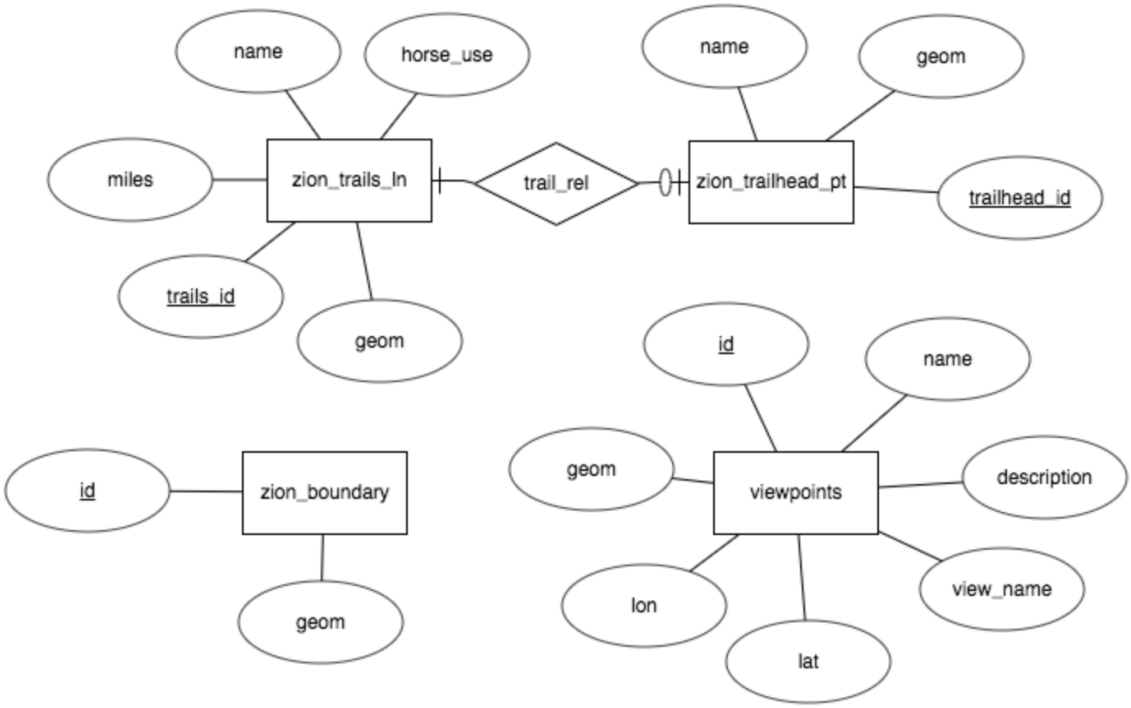

Figure 1. Entity-Relationship diagram, made using ERDPlus

The database structure for this project is relatively simple. The Zion Boundary layer is one standalone feature without any real attributes, the viewpoints layer is also stand alone, but contains multiple attributes for users to fill in before they contribute. These include latitude/longitude, their name, the name for their view, and a description.

The trails and trailhead layers have a loose link. When I downloaded them, I expected there to be a 1 to 1 relationship, but trails data is somewhat incomplete so the trails participation in the relationship is optional, as noted in the diagram above. I attempted to find more complete trails data before continuing with the project, but the data provided by the NPS was the best I could find. A future effort to complete this data will be required.

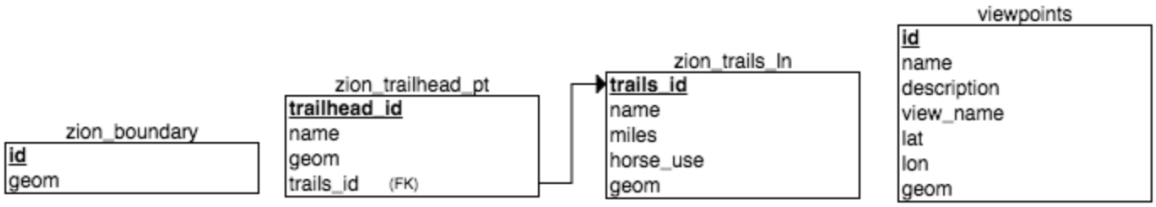

Figure 2. Logical Schema of entities on database, created using ERDPlus

After putting together the logical schema for the data, I created an account with Carto and loaded all 4 layers onto it. Using Carto over something like PostgreSQL or MongoDB streamlines the process of database creation and allowed me to focus on creating the front end without the need to worry about something breaking on the back end. After getting my head around the syntax, Carto's SQL API provided a quick and easy way to access the data in browser. However, due to the public facing nature of its queries I wouldn't use this on any project that contained sensitive information. Securing the data would be much easier using a different platform.

User Interface

Almost more important than the actual data in the map is having a user interface that allows park visitors to have a good experience. Luckily, Carto works well with Leaflet and I was able to hit the ground running since I've worked with Leaflet in previous courses. The biggest hurdle I faced was how to incorporate the way Carto handles data layers with the tools I wanted to create. The way Carto deals with layers is different enough from how Leaflet recognizes them that I was unable to use most of the default leaflet components I initially wanted . Instead, I had to build a few forms myself and rely heavily on some custom components that other leaflet users have developed and offered for free.

The main thing that makes the user interface work well between desktop and mobile browsers is the leaflet-sidebar. This incredibly useful plugin works seamlessly between mobile and desktop views and is customizable enough that I was able to build all the HTML forms I needed for the different functions without any issue.

Functions

This app contains standard functions that you could reasonably expect to find in any web map: Zoom, Pan, the ability to turn layers on and off, info popups when a layer is clicked, a home button to zoom to the initial map extent, and a marker in the mobile version that lets the user know where they are. In addition to these, the user can click through the tabs of the sidebar to gain access to some additional functionality:

-In the first tab the user can see the option to click layers on and off, as well as toggle between a standard base map and a custom base map I created, which I believe fits the National Park aesthetic well.

-In the second tab users can add a viewpoint to the database by filling in a form with attributes. When adding their new point to this layer they can use pre-populated coordinates from their browser, or they can select a location on the map.

-The third tab contains two functions to help users narrow down their trail options. First is the ability to query trails by length using a jQuery UI slider, and a radio selection for whether the trails allow horses. Second is the ability for users to choose a trailhead name from a pre-populated list which, when queried, removes all other trailheads and zooms to the selected location.

User Story

The typical user for the app will be able to look at it on their laptop computer before ever setting foot in Zion National Park. From there, they'll be able to easily browse around and filter the trails to see which trails seem short or long enough to be tackled on a trip. They'll also be able to see which areas park visitors liked the most by using the viewpoints layer and plan their trips around those things that sound most interesting.

Once at the park, users can open the map on their phone and see where they are in relation to where they would like to go, as a supplement to existing paper maps that are scattered around the park. While out on the trails, they can add a viewpoint they are particularly fond of to share with future park visitors. If a trail is too crowded, or they if have a change of plans, it is just as easy to search for trails on the phone as it is on a computer. The only real difference to the user is that the sidebar menu takes up the entire screen while opened on mobile.

Conclusion

This application is a good first step toward a mobile app for Zion National Park, and the way it's laid out with tabs in the side bar leaves room for plenty of additional functionality to be added in the future. One area where this current version could improve is the completion of the trails data. In my view this effort goes beyond of the scope of the project, but it would be essential in a real-world application.