GEOINFORMATICS LABORATORY

Project Report

Raspberry PI GPS Data Logger and Field Notebook

Asad Ahmed
Ralph Florent

19.03.2019

CONTENTS

0.1 *Outline*

This report outlines the steps taken to produce the system referred to as *GNS (Geospatial Notebook System)*.
GNS aims to provide a field notebook experience for researchers by enabling location-tracking of their field notes. A *raspberry pi* computer was used for the main functionality. A GPS device, and battery were connected to Pi. The GPS was used for fetching location data in the field, and the battery ensures operation of the device in the field.

0.2 *Setup and Configuration*

The setup of the system includes the following steps:

1. Connect a GPS device/puck and install *gpsd*
2. Configure GPSD with options:

   ```
   GPSD_OPTIONS="-b -n"
   ```

   in the */etc/default/gpsd*
3. Setup the Wi-Fi hotspot by installing *hostapd* and *dnsmasq*
4. Configure a static IP by editing the configuration file: */etc/dhcpcd.conf*
5. Configure the access point by editing the */etc/hostapd/hostapd.conf* file. Ensure that the bridge is disabled.
6. Reboot the system and all services to access the wireless point.

Once the system is setup, development can take place. This would include the development of a backend system and a frontend web system. These systems are documented in preceeding sections of this report.

0.3 *Basic Workflow*

The basic work flow of GNS is given as follows:

1. Start up the Raspberry Pi system, with the connected GPS, and battery.
2. Connect to the Raspberry Pi's Wi-Fi hotspot: *GNS-Hotspot*.
3. Navigate to the GNS web service on any device near the Pi. This IP address is *10.10.10.11*.
4. A list of notes will be presented that are already in the system.
5. Navigate using the web application to view, create, update, and delete notebooks as required.

0.4 *Functionality*

The system was built to provide the following functionality to the user:

1. **View Notes**
   The user would be allowed to login and view the notes that have been created so far. Each note would have in addition to the text information, geo-spatial data such as the latitude and longitude along with the data and time (UTC) of the note.

2. **View Note**
   The user would be allowed to view specific details on a specific note. This includes its text information, the user who created the note, and the geo-spatial information.

3. **Edit Note**
   The user would be allowed to make changes to a note. This includes editing the note's text data.

4. **Add Note**
   The user would be allowed to add a new note to the system. The date and time and location will be fetched from the GPS device.

5. **Delete Note**
   The user would be allowed to delete a note from the system. The note will no longer be visible to the users of the system.

0.5 *Web Interface*

As mentioned above, GNS consists mainly of two parts: firstly, a background running process to load and process the raw GPS data, and on the other hand, a web interface to visualize the pre-processed data and perform certain actions with them. This section will focus on detailing the different components joint together to produce the expected end-results. GNS was never conceived as server-to-server application, but as a client-to-server application. Although the server-to-server module remains eventually a considerable aspect to take the project to a distribution level, a normal end-user will enjoy better a WYSIWYG-based experience using an HTTP Web Client, mostly known as a web browser.

0.5.1 *Choosing a tool to build the web app*

Unlike the old times, web developers nowadays exposed to innumerable sets of tools easing up their lives when it comes to create web pages and applications.

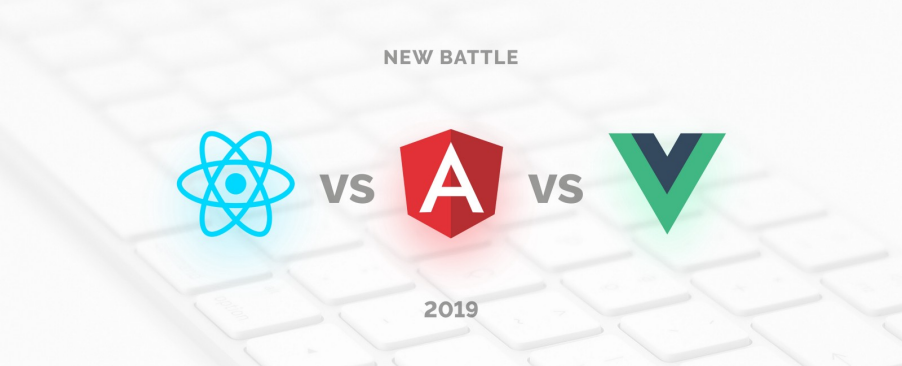

Figure 1: Client-side JavaScript tools for building web applications

In fact, building a web application is no longer a luxury for web developers due to the amount of frameworks and libraries currently available. Big companies like Google, Facebook, among others, dedicate their strengths and efforts in building powerful tools to enhance fast development and deployment across all platforms [1].

Being exposed to so many options, choosing the right tool for the GNS web interface can be a difficult task. The so-called analysis paralysis. Therefore, a fair comparison between the most used frameworks and libraries seems to be a good startup before diving deep into the technical details of a web application.

Today's most famous frameworks and libraries are Angular (powered by Google), React (developed by Facebook), and Vue.js (developed by Evan You) - see figure 1. All three (3) are open-source client-side JavaScript tools to create, build and deliver on-demand web applications. A few examples of web applications built with those tools are: Facebook, Gmail, Airbnb, Amazon, Netflix, etc. Besides that, the community of developers is huge and contributes to the improvement of these tools on a daily basis.

Angular and React outperform Vue.js for fanaticism reasons. Hence, they remain the two highly considered competitors on the market.

Angular vs React

One main relevant difference between Angular and React is their core concept: Angular is a framework for JavaScript whereas React is a JavaScript library. This explains why React is faster, lightweight, and painless when creating interactive user interfaces (UI). Of course, other JavaScript libraries can be that powerful too, but there are some important core features that make React much more outstanding. These are:

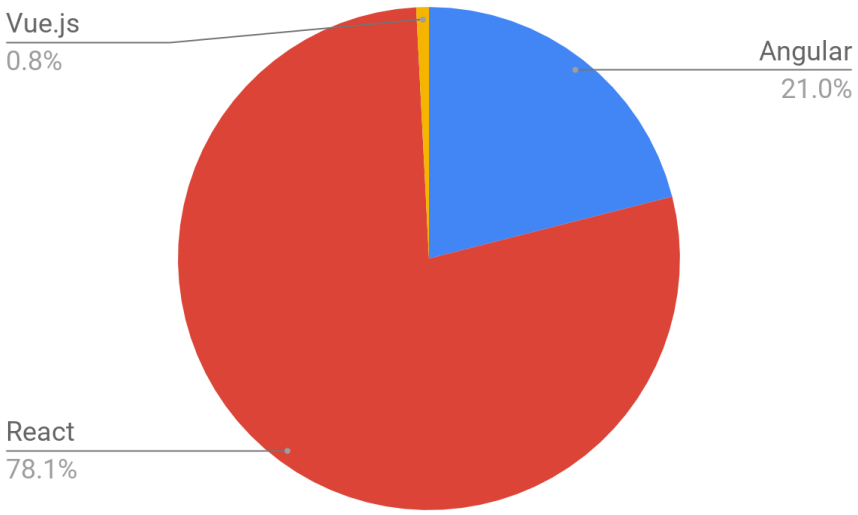

Figure 2: React and Angular as the two main competitors in indeed.com

1. Fast and responsive: React can scale to large and complex UIs because of its efficiency about how and when it manipulates the DOM (via Virtual DOM)

2. Composable: React allows to compose components, i.e., components can be easily nested within other components and communicate data down to sibling or child components via props.

3. Pluggable: unlike larger full-featured frameworks like Angular, Vue.js, or even Ember, React is totally pluggable into existing applications. Since it's a view layer, it can be easily integrated with other technologies.

4. Isomorphic friendly: a feature that allows React to render in client or server side.

So far, the benefits of using React have been mentioned. However, one needs to think of the tradeoffs too: Framework vs Library. This is where Angular comes up with all the power. Because React is a library, its major focus is on components. However, a framework is more complete. It covers more cross-team consistency and avoids setup overhead. In other words, a framework covers more features such as routing, testing, http library, animation, etc., than a library does. See *Fig. 3* for a more descriptive analysis.

In order to take advantage of React, some pieces need to be pulled out from the list and plugged in under certain configurations to develop a clean, lightweight web app. However, Angular appears to be fully featured and doesn't

Features	React	Angular
Components	✔	✔
Testing	Jest – Mocha	✔
HTTP Library	Fetch, Axios	✔
Routing	react-router	✔
I18n	react-intl	✔
Animation	react-motion	✔
Form Validation	react-forms	✔
CLI	create-react-app	angular-cli

Figure 3: Features of React and Angular

require extra configurations. Thus, GNS UI will have a better feature if developed with Angular. Other pointer to such a decision is the well-documented (A-Z) guide provided by the Angular team. Have a quick look here.

0.5.2 *Using Angular to build GNS-UI*

No wonder, Angular is the chosen one. GNS UI is the HTTP web client module for the GNS project, used to access geospatial data from the GNS API and take notes.

This web application, known as GNS Web App, is used to manage different tasks when it comes to taking notes and archiving them. Among others, certain tasks feature the following: locating the current position of the set of Raspberry PI + GPS puck; adding, editing, and/or viewing existing notes; creating base maps based on the path traveled during the process of note taking, and so forth. To build the website from scratch, follow the instructions below from a development standpoint up to production.

Getting Started
Assuming that the reader has the basic programming/DevOps knowledge, this section aims to explain the procedure implemented to build the web application up to deploying it on the Apache web server.

Prerequisites This web interface module is built with Angular Framework v7.2.1, powered by Google, Inc. If not yet installed, please install these following task runners using a preferred command line console *(linux terminal, Git Bash, Windows Powershell, Iterm2, etc.)*:

1. Download and install Node and npm
2. Once node and npm installed, this reference can be used, or run the command 'npm install -g yarn' to install Yarn globally
3. Install Angular CLI by running this command 'npm install -g @angular/cli' or using this link.

In case of errors during the installation, one can always google-search for quick fixes or use the references (mentioned above) of the respective official web pages for more information on those errors.

If everything goes well, the next step is to proceed with setting up a development environment (dev-env) so the app can be tested and run locally. But if the user is not interested in visualizing the app on his/her local machine, he or she can always skip this next section and go directly to the deployment section.

Create a workspace
To run the web page, the developer should set up a development environment. To do so, he or she needs an IDE such as IntelliJ IDEA, Visual Studio Code, WebStorm, and any related workspace enabling developing features like intellisense, for example. His or her preference is what really matters.

Since the web app is an Angular web-based application, Visual Studio Code seems to work quite well with it and contains some useful features at the time of developing. Other considerations could be IntelliJ IDEA as a strong, stable IDE for development.

IMPORTANT: The instructions detailed in the following sections target a Visual Studio Code IDE. For other IDEs, the developer (reader) shall inquire details on how to configure an angular-adapted environment for a specific IDE.

Running GNS UI locally
Running the GNS UI is an easy-to-do computing task. Basically, a command. However, keep in mind that it requires to set a proper environment while having the GNS API service running in parallel in order to have a first look.

The steps that follow below are a guide for developers willing to reproduce the development environment so that they may add more features and write tests for future releases.

1. Clone (or download) the repository here using any CLI or any preferred 'Application'.
2. Checkout to the 'develop' Git branch and pull the changes in case of any updates.

3. Open the project using the Visual Studio(VS) Code IDE.
4. Locate the Terminal of the IDE, make sure it's open in the current working directory, which is 'gns-ui'.
5. Run the following command 'yarn'. Being installed globally, Yarn will scan the package.json file and install all the dependencies accordingly. This might take a little while.
6. After installing the dependencies, the developer can run the following command 'yarn start' to run the app locally. The Angular CLI will be in charge and open the web app in the default web browser.

***Notes**: In order to simulate a production-like environment while at the development environment, some changes might be necessary in the package.json file.*

- The default port used by Angular CLI is 4200. To avoid conflicts with existing apps running in port 4200, set the following starting script

  `'ng serve --port [port_number]' in the 'package.json' file.`

- GNS Web App requires the GNS API, which is an independent service located in another port, to load GPS data, and perform specific tasks.

Finally, running the command 'yarn start' in the selected terminal should start the app.

Deploying the app to the web server

The GNS Web App, at the time of writing this document, is at its 'v1.0.1' version. This section covers specific headache at the moment of running and deploying the web application on the Apache server of the Raspberry PI.

1. First off, make sure the Apache server is up and running (Read more here on how to set and run Apache2 on Raspberry PI)
2. Build the GNS Web app for production by running 'yarn run build' command in the CLI. Once built successfully, the GNS UI is ready to be deployed (go to production). It should be noted that when built with Angular, using 'ng build –prod' generates hashed files under a 'gns-ui' folder.
3. Zip the folder 'gns-ui/dist/gns-ui/*' to 'gns-ui.zip'
4. Transfer the 'gns-ui.zip' via a client (S)FTP or via SCP over ssh connection. It is recommended the use of FileZilla to avoid complex client-side configurations. It should also be noted that '*scp*' or '*sftp*' via a terminal can be used to copy the zip file 'gns-ui.zip' to the Raspberry PI. For example, using the PI as hotspot: 'scp gns-ui.zip pi@10.10.10.11: /Projects'.

5. Connect to the PI using *'ssh'* via a linux terminal. For example, using the PI as hotspot: 'ssh -l pi 10.10.10.11'.
6. Copy and unzip the zip file 'gns-ui.zip' to '/var/www/html'.
7. Enable rewriting module for Apache2 web server (Read more about it here)
8. Finally, go the browser, try to load and reload the page.

IMPORTANT: If it runs on the local machine, it should also run on the web server. Always make sure the tests are passed. But if any errors, the developer can always check the console output.

[Preview of the GNS UI first-look]

How does GNS UI work?
At this point, the technical details of how to run GNS UI are no longer necessary. In other words, this section discusses the part where a simple end-user can take advantage of GNS Web App

0.6 *Backend System*

This section will highlight the development process of the backend system. The back-end system outlined in this report refers to the system residing on the Raspberry Pi. This system manages the database, and the GPS data streaming. It is also responsible for providing the web application with the required data and functionality.

0.6.1 *Architecture*

The GNS backend was built using *Java 11* and the Spring Boot Framework[8]. The system was developed to provide a *REST based micro-service* running on the Raspberry-Pi. Spring Boot is bundled with its own *Apache Tomcat Java Server*[2], and the final artifact from the development was a *jar* file that was run on Raspberry Pi under the *JVM*.

These REST endpoints were exposed to interface with the system for carrying out basic operations on the notebook data, namely to create, read, update and delete notebooks. The user interface was a web application that was a client of this backend-system.

In addition to the REST API, the system also connected to the GPS system program, *gpsd*. This was accomplished with the use of a Java library: gps4j[3].

This allowed the system to become a client of the *gpsd* server, that was also running on the Pi. The library was used to *stream* the GPS data every second. This data was dumped to a *CSV file*, and was renamed according to the date. Hence, the location of the Pi is always being tracked, and is stored by day in CSV files. These log files can be found in the *gps_logs* folder.

0.6.2 *Location Data*

The location data that is dumped to the CSV file every second, was used in a test run. The device was started, and notes were created around campus. During this test walk, the backend system was streaming the GPS data continously to the CSV file. The first few records of this CSV file are given below:

```
Latitude,Longitude,Date,Time,UTC
53.168051667,8.653143333,2019-03-22,14:42:35,2019-03-22T14:42:35.580
53.168033333,8.653375,2019-03-22,14:42:36,2019-03-22T14:42:36.580
53.167991667,8.653343333,2019-03-22,14:42:37,2019-03-22T14:42:37.580
53.167906667,8.653568333,2019-03-22,14:42:38,2019-03-22T14:42:38.580
```

```
53.167656667,8.653776667,2019-03-22,14:42:39,2019-03-22T14:42:39.580
53.167758333,8.65356,2019-03-22,14:42:40,2019-03-22T14:42:40.580
53.167838333,8.653538333,2019-03-22,14:42:41,2019-03-22T14:42:41.580
53.16783,8.653433333,2019-03-22,14:42:42,2019-03-22T14:42:42.580
53.167825,8.653615,2019-03-22,14:42:43,2019-03-22T14:42:43.580
53.167843333,8.653485,2019-03-22,14:42:44,2019-03-22T14:42:44.580
53.167858333,8.65339,2019-03-22,14:42:45,2019-03-22T14:42:45.580
53.167863333,8.653286667,2019-03-22,14:42:46,2019-03-22T14:42:46.580
53.168181667,8.653945,2019-03-22,14:43:28,2019-03-22T14:43:28.580
53.168081667,8.653755,2019-03-22,14:43:29,2019-03-22T14:43:29.579
53.16796,8.653941667,2019-03-22,14:43:30,2019-03-22T14:43:30.579
53.167905,8.653795,2019-03-22,14:43:31,2019-03-22T14:43:31.579
53.167876667,8.653578333,2019-03-22,14:43:32,2019-03-22T14:43:32.578
53.167851667,8.653726667,2019-03-22,14:43:33,2019-03-22T14:43:33.579
53.167811667,8.653626667,2019-03-22,14:43:34,2019-03-22T14:43:34.579
53.167855,8.653723333,2019-03-22,14:43:35,2019-03-22T14:43:35.579
53.167918333,8.653641667,2019-03-22,14:43:36,2019-03-22T14:43:36.579
53.167865,8.653491667,2019-03-22,14:43:37,2019-03-22T14:43:37.579
53.167946667,8.653496667,2019-03-22,14:43:38,2019-03-22T14:43:38.579
53.167876667,8.653625,2019-03-22,14:43:39,2019-03-22T14:43:39.579
53.167865,8.653586667,2019-03-22,14:43:40,2019-03-22T14:43:40.579
53.167841667,8.653498333,2019-03-22,14:43:41,2019-03-22T14:43:41.579
53.167906667,8.65341,2019-03-22,14:43:42,2019-03-22T14:43:42.579
53.167905,8.65336,2019-03-22,14:43:43,2019-03-22T14:43:43.579
53.167921667,8.653295,2019-03-22,14:43:44,2019-03-22T14:43:44.579
53.16793,8.65318,2019-03-22,14:43:45,2019-03-22T14:43:45.579
53.167918333,8.653248333,2019-03-22,14:43:46,2019-03-22T14:43:46.579
53.167915,8.653338333,2019-03-22,14:43:47,2019-03-22T14:43:47.579
53.167913333,8.65323,2019-03-22,14:43:48,2019-03-22T14:43:48.579
53.167901667,8.653353333,2019-03-22,14:43:49,2019-03-22T14:43:49.579
53.167895,8.65347,2019-03-22,14:43:50,2019-03-22T14:43:50.579
53.16789,8.653568333,2019-03-22,14:43:51,2019-03-22T14:43:51.579
53.167886667,8.65376,2019-03-22,14:43:52,2019-03-22T14:43:52.579
53.167885,8.653856667,2019-03-22,14:43:53,2019-03-22T14:43:53.579
53.16788,8.653871667,2019-03-22,14:43:54,2019-03-22T14:43:54.579
53.167878333,8.653825,2019-03-22,14:43:55,2019-03-22T14:43:55.579
53.167893333,8.653611667,2019-03-22,14:43:56,2019-03-22T14:43:56.579
53.167896667,8.65356,2019-03-22,14:43:57,2019-03-22T14:43:57.579
53.167898333,8.653533333,2019-03-22,14:43:58,2019-03-22T14:43:58.579
53.167888333,8.653661667,2019-03-22,14:43:59,2019-03-22T14:43:59.579
53.16789,8.653595,2019-03-22,14:44:00,2019-03-22T14:44:00.579
53.167885,8.653641667,2019-03-22,14:44:01,2019-03-22T14:44:01.579
53.16787,8.65384,2019-03-22,14:44:02,2019-03-22T14:44:02.579
53.167866667,8.6538,2019-03-22,14:44:03,2019-03-22T14:44:03.579
```

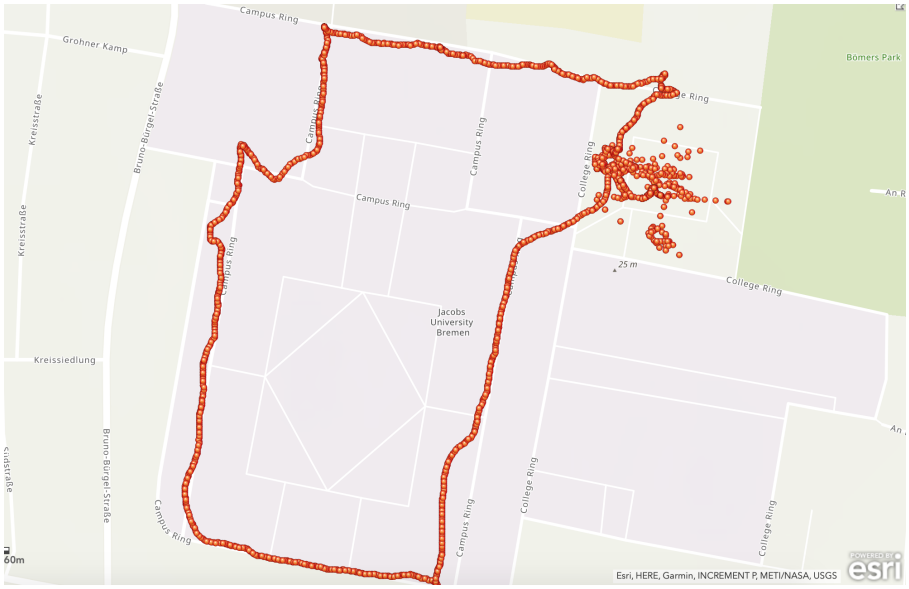

Figure 4: The GPS track loaded into ArcGIS

This CSV file was uploaded to ArcGIS as a hosted layer. The layer was imported into a map and is given in *Fig. 5*.

REFERENCES

[1] Angularjs. https://angular.io/. Accessed: 2019-03-20.

[2] Apache tomcat. http://tomcat.apache.org. Accessed: 2019-03-29.

[3] gpsd4j. https://github.com/ivkos/gpsd4j. Accessed: 2019-03-29.

[4] React: The big picture. https://app.pluralsight.com/library/courses/react-big-picture/table-of-contents. Accessed: 2019-03-20.

[5] Reactjs. https://reactjs.org/. Accessed: 2019-03-20.

[6] Reactjs. https://react.parts/. Accessed: 2019-03-20.

[7] Reactjs. https://medium.com/TechMagic/reactjs-vs-angular5-vs-vue-js-what-to-choose-in-2018-b91e028fa91d. Accessed: 2019-03-20.

[8] Spring boot framework. https://spring.io. Accessed: 2019-03-29.

[9] What are the most popular react plugins? https://www.quora.com/What-are-the-most-popular-React-plugins. Accessed: 2019-03-20.