Forest Service API Project Documentation

CS 481 - Senior Design

12/13/19

Developers:

Alex Simons, Brenna Leonard, Peter Henggeler, Seema Marahatta

Sponsors:

Kelly Hopping, Cathie Olschanowsky

Table of Contents

Project Overview

When ecologists are working in the field, they often record their data by hand. This hard-won information then tends to remain trapped in physical copies of datasheets or else becomes lost in a proliferation of unwieldy Excel files, thereby preventing further analysis. The U.S. Forest Service, for example, has valuable information about historical environmental conditions that could be used to help track how ecosystems have changed over time, but getting this information out of the filing cabinet and onto a computer in a format that can be queried and analyzed remains a key constraint to this type of research.

The goal of this project was to develop a relational database populated with environmental monitoring data from around Sun Valley, Idaho. This work was in collaboration with the Sawtooth National Forest, which has collected decades of data on historical vegetation and soil conditions. This database will also serve as the foundation for future work to create an app that scientists can use to collect these same data today. By making nearly a century of environmental data ready for statistical analysis, this project will allow Forest Service and BSU scientists to answer important questions about how some of Idaho’s most spectacular landscapes have been affected by climate change, sheep grazing, and natural resource management decisions across areas and timeframes that were previously impractical to tackle.

Where We Left Off

We had finalized our schema fully with the sponsor this semester. We had just started working on the following sections:

- Building out a REST api to allow users to interact with the database
- Creating a front end UI
- Started researching what is required to host the app on a server
- Started creating a process for bulk entry of data

How To's

Setup

- Ensure that Docker is installed on your machine

- Ensure that docker-compose is installed on your machine

- Clone the git repo
  github.com/alexsimons/ForestryServiceDatabase.git

- From the root directory, run the following command:
  `docker-compose up --build`

Database

To contribute to the database, reference the following example:

- Create a new file `CreateExample.sql` in the
  `./sql-scripts/` directory

- Add the following sql query into the file:
  `CREATE TABLE example(example_name varchar(25), example_description varchar(25));`

- Save `CreateExample.sql`

- From the root directory, run the following command:
  `docker-compose up --build`

Node Application

To contribute to the node application:

- Make file changes in the `web-app` directory

- From the root directory, run the following command:
  `docker-compose up --build’

VM / Server hosting the App

We focused mostly on the schema design and api setup. We were not ready to host the app on a server. That said, the sponsors indicated that we could likely host the app on a BSU-provided server. When it comes time to host the app on a shared server, verify if the BSU-provided server still works for the sponsor. If so, Ben Peterson should be able to provide a server to host the app.

Persisting Data

Please note that the database is not persisted currently. So when the docker Postgres image is running, you can insert some data. When it is stopped and restarted, all of that data is wiped out.

- In order to persist data, the docker-compose.yml file will need to have a "volumes" created in your docker-compose.yml

- For instance, the following code snippet when added to a docker will persist data from the 'data' directory on your local machine to 'var/lib/postresql' on the Docker container
    - `volumes: - ./data:/var/lib/postgresql/`

- Reference this stackoverflow for more info: https://stackoverflow.com/questions/46504902/postgres-docker-persistence

- Please note that you do not need to persist data when developing and testing. When your sponsor has given you permission to run the app on a Boise-state provided VM, can add the 'volumes' flag to the docker-compose ym to persist data.

Adding Endpoints

To add or change existing endpoints:

- Navigate to ForestryServiceDatabase\web-app\server.js to find existing endpoints
- Reference the example of endpoint on lines 69-88
- Modify return query from "`SELECT * FROM cover`" to desired query
- Modify "`'/info'`" to desired endpoint
- After running a docker build, the endpoint should be locally accessible **ex**:**localhost:8081/info**

How to add a view

The view system used in this project is Pug. Pug allows one to do a lot of dynamic HTML templating. Check out their documentation HERE. To add a view to the app:

- Create a new route in server.js. (If it is a form you will need 2 routes: A GET route and a POST route).
- Create a new pug file in the /views folder.
- Refer to the `new-view-template.pug` file as a starting point
- If you'd like to add a link to the view in the nav bar, add an `li` element in the `layout.pug` file.

Known Bugs / Limitations

- Can only insert report summary at the moment, can’t insert individual plots/transects yet

- Data is not currently persisted in the docker-compose file. This was done for development reasons.

- The tests written for the app do not run in an automated fashion. More work needs to be done to make sure the tests run on every commit.

- The query builder forms don’t actually run queries on the database yet. The queries are written and the form is gathering the query filter data but these need to be linked.

- The transect query builder and full-report query builder have not been completely built out. The routes exist and there are some filters in the form but these need some more development work. Discuss with sponsor to determine what to add to these forms.

- In further examining the other example forms, it seems the `total_grass`, `total_forbes`, and `total_browse` fields are not used consistently (they are not used in the newer forms). Perhaps they could be removed from the schema (in the report table). It also seems like it is a summation which doesn’t have to be stored as an attribute. If we have the attributes that are being summed (it seems to be the summation of every type (grass/forb/browse) in both the `Prod.Dry Weight by Acre` column and the `Percent Composition` column. These fields can be removed from the schema.

Requests from sponsors that were not implemented:

- Convert the metric ‘aspect’ to degrees as opposed to using cardinal directions

- Export data to CSV as opposed to JSON data

- Being able to find datasheets based on a species name (needs both query support and to be added to the query builder form)

Bulk Data Entry Process

Bulk data uploads are not supported by the application. We have created a basic csv template that can be used to enter data for a report that could be easily read by a python script and transformed into sql code that could be inserted into a database. We wanted to get a template set up for data entry so that this can be worked on until bulk data entry is developed and supported on the application. The csv file can be found in the git repository.

NOTES:

- When you come across a `‘T’` in the form, it represents ‘a trace amount’. Since the field we are using to capture these numbers is an integer, we simply replace the `‘T’` with a `‘0’`.

- There are two csv’s present in the git repository - one is the csv report data template, the other is a semi-filled example to give an idea of how the form should be filled out. The form used to fill out this information is the original datasheet shared with us, with the writeup id of `‘A-13’`.

- This script is designed to be read in by a scripting language like python that could then pull the data from this form and then insert it into the database, once it has been set up to persist.

- Each block of data in the form follows a title of the data below it as a description of where the data belongs in the database. This should be useful for the script writer.

Future Development Ideas

- Ability to scan in old reports and have them read into the database
- Web-based application for scientists to collect data
- Adding mobile CSS support so the web app renders well on tablets. If this app were used to collect data in the fields it would likely be on a tablet.
- Breaking out the insert datasheet into a multi-step form instead of being one gigantic form