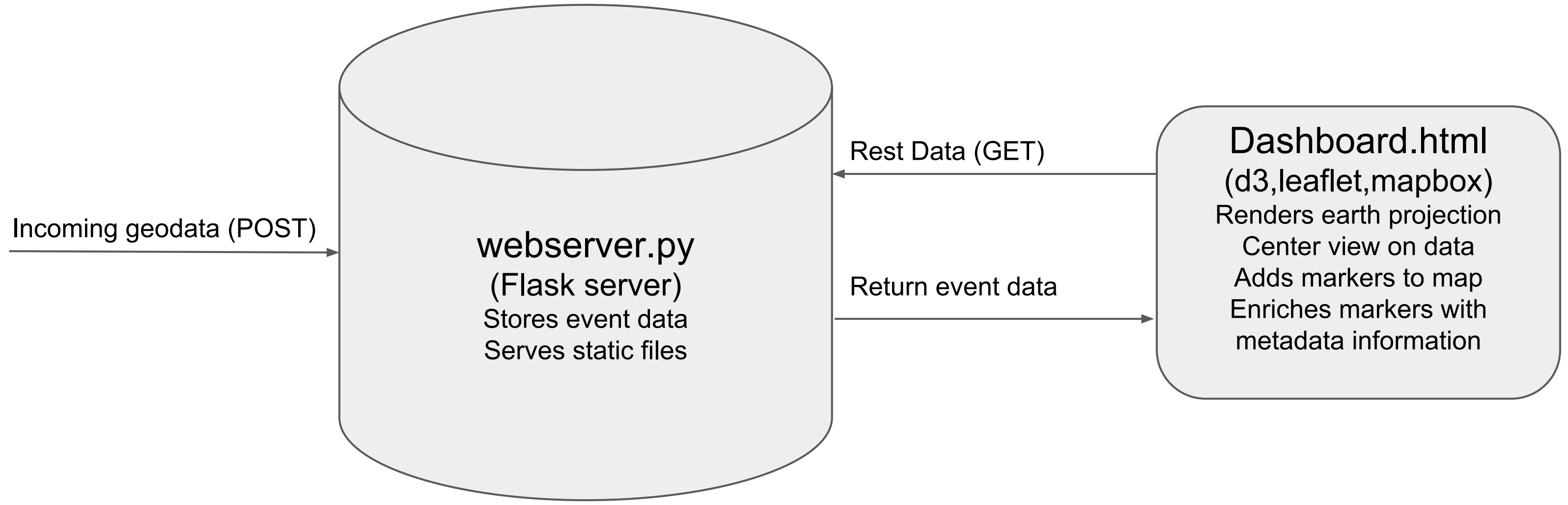
Lore

A simple geo-event visualizer

Ryan Stepanek

Lore - Summary



Lore - Explained

The Lore web server listens for incoming posts and caches them in memory. When the dashboard is started, it begins querying the web server periodically for the next event’s data, gradually emptying the server’s cache.

Each data event is turned into a marker and placed on a map against a tile backdrop of satellite imagery. Zoom level is left in the hands of the user via an intuitive, “google map-like” interface. The very first data event centers the initial view for the map. The user can watch their events filter in and click on them to see the detailed information for the event.

Lore - Running the System

1. Start up lore server by running "python webserver.py" in the root directory of Lore. This starts a web server at localhost:5000. If you open the page now, you will see a map centered in the UK, this will update when the script ingests its first real data from the server.

2. Start up your simulation by running "python main.py" in the root of the Legend project: if you have left in the config "server_uri = http://127.0.0.1:5000" the sim will stream events to the Lore server in addition to logging them in the output directory.

3. Once the simulation is finished running, simply visit http://127.0.0.1:5000 which will load up a dashboard that begins visually representing your simulation events (click on an event to pop-up detailed information).

Lore - Dependencies

Lore requires:

- Python - 2 or 3 will work
- Flask - A python module that manages server requests
- Internet Access - Due to space constraints, it was infeasible to store map tiles for the entire dataset within the project, so Lore relies on the Mapbox api to download map tiles as needed.

Lore - Next Steps

My original vision of Lore went far beyond what time constraints allowed me to implement, here are the next steps to bring the software inline with my vision:

- Database - Lore should include the option of using a database instead of tracking events in memory to prevent it from being overwhelmed by large datasets. While a NoSQL approach would work, the data geodata tends be highly structured, making a traditional SQL database an equally viable option. This would allow easy saving and comparison f data from multiple runs.
- Convert time from sim to real-world - Entity information should display real world time, rather than sim time by default.
- Add global statistics - Add a data display to show total number of entities, types, states, etc...
- Add crossfilter - It should be possible to filter based on entity state, type, or process.
- Add time filtering and playback - Allow the user to play through their scenario similar to a kml movie
- Allow the user to customize visual markers and data displays.
- Change javascript libraries to local libraries that are package with the application.
- Host Mapbox tile server locally with application to decrease loading time and allow deployment in non-internet connected environments.