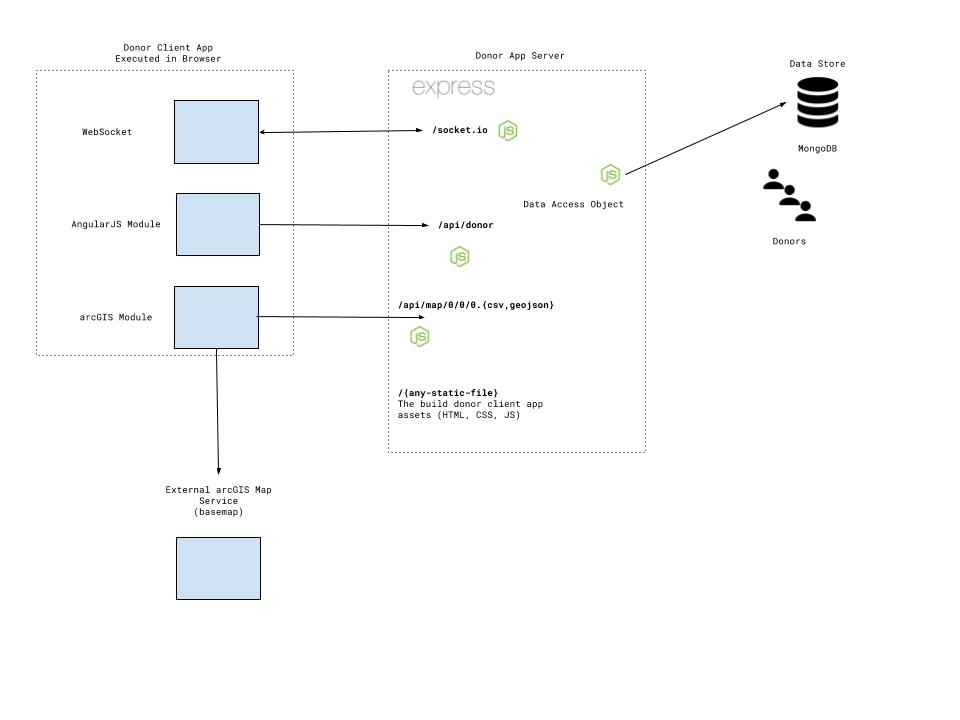
# Blood Donor Design

This document will mainly contain visual explanations of the architecture.

**Major Components**



The Donor Server is responsible for:

1. Serving the built JS with corresponding assets (this is done initially when entering the URL)
2. Opening a WebSocket (via special root) and keeping it open
3. After the base map has loaded, it provides the feature layers in a tile format (not implemented fully)
4. It provides a REST interface for CRUD operations on the donor
5. Using a Data Access Object to for CRUD operations in the Data Store (i.e. MongoDB Collection). This is linked to the REST endpoints for ease of use.

The Angular Module is responsible for:

1. Internal routes (#/donors, #/patients) that do not go to the Donor Server
2. Triggering CRUD operations via REST endpoints provided by the Donor Server
3. Interactivity between widgets (e.g. modals, header button click)

The arcGIS module will inject itself into an Angular Component and will know how to:

1. Fetch the base map (this is stored in the ArcGIS cloud)
2. Fetch the layer with features from the Donor Server via /api/map/\* endpoints
3. On an event basis will update the feature layer

Will trigger a click event will location data to be received by the Angular Module?

**Proposal for a lazy load**

We need a feature layer that is provided via vector map tiles. They are vector because we do not do any server side rendering (like raster tiles) and the styling is done in the browser providing quick response times.

The tile service is composed of a load balancer and numerous map service instances. The load balancer will make sure that not any map service is having too much to do.

The map service is a HTTP NodeJS (ExpressJS) server that connects to the Data Service and fetches data to be displayed in the requested tiles. There is heavily process intensive task, as clipping billion of features can be.

The Data Service is responsible for storing the feature data. One could devise them in Read and Read-Write nodes to further speed up things. The Read nodes will synchronize periodically with the Read-Write nodes.

