Test project

Application is quite simple. And can be explained with this diagram:



Donor sends info to DonorsController.createDonor method, data is saved to MongoDb and is sent to connected via WebSocket patients.

Patient connect to WebSocket and requests donors info from DonorsController.getDonors.

# Server side

Server side is built using next libraries and frameworks:

* NodeJs v6.6.0 as js interpreter
* Express framework v4.14.0 as http server
* MongoDb v2.2.10 as data storage
* Socket.Io v1.4.8 to keep patients updated
* Babel v6.16.0 is used to transpile es6 modules
* And other few libraries to parse http and logging

## Server logic

www/bin is used as server’s entry point:

* it registers babel-register to transpile server code on the fly (this is for dev/test project only, we need to pre-transpile for production)
* calls server/startup.js
* add error handlers to log errors to http server

server/startup.js configures server and start http server:

* creates Express app and configures different middlewares
* create Socket.Io server and saves it as app.locals.io
* connects to MongoDb
* adds our custom routes to app

server/routes contains routes for one Donors controller and rule to send all other requests to client/index.html

server/controllers/donors is one controller in app. It contains two methods:

* createDonor saves donor to MongoDb and sends donor to patients via WebSocket
* getDonors returns existing donors from MongoDb

There are no checks for MongoDb results and there are no donor data validation. I didn’t have time to add it.

lib/async is very similar to <https://github.com/tj/co> and uses generators to write async code as sync.

# Client side

Client side is built using next libraries and frameworks:

* ArcGis v4.1
* Dojo as ArcGis dependency is used as AMD loader
* React v15.3.2, react-router as separation of components and single page application routing
* Babel v6.16.0 is used to transpile es6 modules and react jsx templates
* Gulp v3.9.1 is used to run babel against client side code
* Socket.Io v1.4.8 is used to keep patients updated

client/index.html is entry point for client app

* includes ArcGis and Socket.Io directly to page
* includes config for Dojo AMD loader
* includes client/src/index.js

client/src/index.jsx is startup for application

* creates Map and MapView for main map
* creates additional widgets to search address and determine current user location
* contains routing config
* contains App component as view presentation for routing and nothing else

client/src/donor/popup.jsx is main component for Donor page

* adds map event handlers to show popup to fill in donor info
* contains donor info popup and logic to save info

client/src/patient/donors.jsx is main layer for existing donors

* creates FeatureLayer from ArcGis library
* contains donors representation logic on map
* requests donors on each map update (when user zoom or move map) to load donors
* contains donor info popup