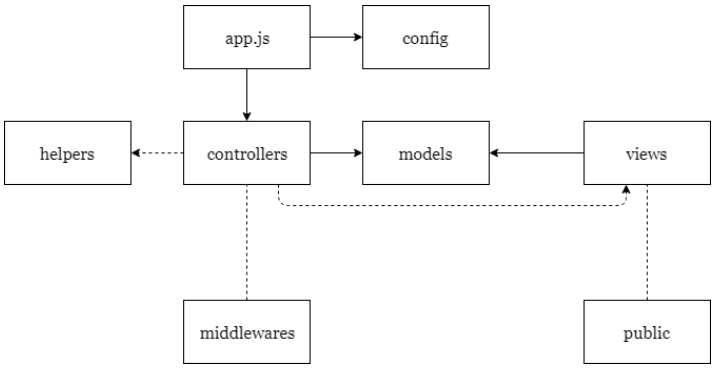
Emergency Social Network Team A1

The goal is to provide civilians with a social network that they can use during emergency situations. The system is different from other existing social networks because it is specifically designed to effectively support small communities of civilians seriously affected in case of natural disasters like earthquake, tsunami, tornado, wildfire, etc.

Technical Constraints

- HTML, CSS, and JavaScript for the UI and front-end functionality.
- Structured JavaScript framework AngularJS used for dynamic one-page applications.
- Node.js with express.js and any other supporting JS-based frameworks for the back-end
- A database (MongoDB) for persisting data.
- Project deployed on a Cloud platform, Heroku.
- System has a RESTful API - should function with and without UI
- System supports real-time dynamic updates



High-Level Functional Requirements

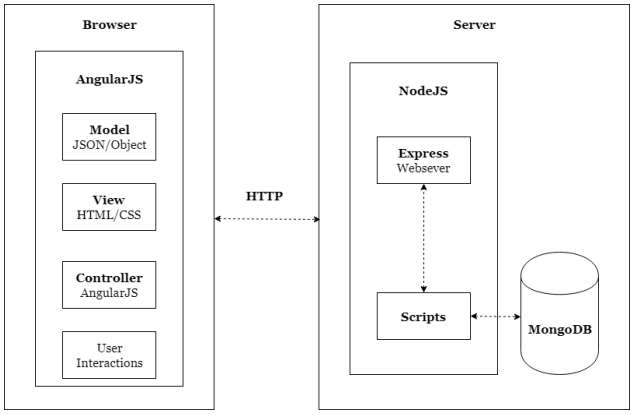
- Join community
- Chat publicly
- Share status
- Chat privately
- Post announcement
- Search information
- Administer user profile
- Post announcement
- Map
- Voice Message
- Activity

Top 3 Non-Functional Requirements

Usability > Reliability > Scalability

Architectural Decisions with Rationale

- Client-Server as main architectural style
- Server-side JS (node.js) for small footprint and performance
- Lightweight MVC on the server side using **express** framework
- RESTful API provides core functionality and reduces coupling between UI and back-end
- Web-sockets allow event-based fast dynamic updates
- MongoDB as database



Design Decisions with Rationale

- Encapsulate data and behavior in models for easy testing and better modularization
- Dependency Injection makes code more readable and reusable as the implementations are configured in the XML file.
- Factories provides a generic interface for creating objects.

Responsibilities of Main Components

- **models:** encapsulate data and behavior for entities of the system, main models are User, Message.
- **controllers**: Responsible for handling incoming requests and returning a response to the client.
- **views**: how data lay out, how it is displayed.
- **middlewares**: Access to the request object (req), the response object (res), and the next middleware function in the application's request-response cycle.
- **helpers:** helper functions