Chat

# Stack

jQuery

backbone.js

underscore.js

socket.io

**Client (browser)**

**chat logic**

node.js

socket.io

express

mongoose

**Server**

MongoDB

Redis

**app.js**

**WebSocket**

**HTTP**

# Architecture

node

http

websocket

MongoDB

Redis

iOS

Android

Windows Phone

Home

Profiles

(room, user)

Chat

User account

Backend

Web

Mobile

# Client Server communication

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **emitter** | **event** | **data** | **description** |
|  | CLIENT | *[connect]* | {} |  |
|  | CLIENT | *[disconnect]* | {} |  |
|  | SERVER | welcome | {  username: *username*,  avatar: *avatar*,  rooms : *[String]*  } | Trigger on connection, provide needed information for interface construction |
|  | CLIENT | room:join | {  name: *name*  } | Client join ask to join a room.  Create the room if not already exist. |
|  | CLIENT | room:leave | {  name: *name*  } | Client ask to leave a room |
|  | SERVER | room:join | {  name: *name*  } | Server ask to client to join this room(probably joined from another device) |
|  | SERVER | room:leave | {  name: *name*  } | Server ask to client to leave this room (probably leaved from another device) |
|  | SERVER | room:welcome | {  name: *name*,  topic: *topic,*  users: *[User]*  } | Received after a successful room joining |
|  | CLIENT | room:topic | {  name: *name*,  topic: *topic*  } | A client try to change a room topic |
|  | SERVER | room:topic | {  name: *name*,  topic: *topic*  } | Server inform room user that topic has changed |
|  | SERVER | room:in | {  name: *name*,  username: *username*,  avatar: *avatar*  } | Server informs client that this user has joined the room |
|  | SERVER | room:out | {  name: name,  username: *username*  } | Server informs client that this user has leaved the room |
|  | CLIENT | room:message | {  name: *name*,  message: *message*  } | Client post a message in room |
|  | SERVER | room:message | {  name: name,  time: *time*,  message: *message*,  username: *username*,  avatar: *avatar*  } | Server send a posted message to the client |
|  | CLIENT | room:search | {  search: *search*  } |  |
|  | SERVER | room:searchsuccess | [  room  {  count: *count*,  name: *name*,  topic: *topic*  }  ] |  |
|  | SERVER | room:searcherror | {  } |  |
|  | CLIENT | user:message | {  to: *user\_id*,  message: *message*,  } |  |
|  | SERVER | user:message | {  from: *user\_id*,  time: *time*,  message: *message*,  username: *username*,  avatar: *avatar*  } |  |
|  | CLIENT | user:search | {} |  |
|  | SERVER | user:online | {  username: *string,*  avatar*: string*  } |  |
|  | SERVER | user:offline | {  username: *string*  } |  |
|  | AJAX | room:profile |  | // @todo : should send JSON via websocket and handle rendering with template |
|  | AJAX | user:profile |  | // @todo : should send JSON via websocket and handle rendering with template |

# Models

|  |  |  |  |
| --- | --- | --- | --- |
|  | Client | Server |  |
| User:: |  |  |  |
| \_id |  |  |  |
| email |  |  |  |
| username |  |  |  |
| password |  |  |  |
| avatar |  |  |  |
| bio |  |  |  |
| location |  |  |  |
| website |  |  |  |
| time\_created |  |  |  |
| time\_logged |  |  |  |
| time\_updated |  |  |  |
|  |  |  |  |
| Room:: |  |  |  |
| \_id |  |  |  |
| name |  |  |  |
| description |  |  |  |
| rules |  |  |  |
| baseline |  |  |  |
| permanent |  |  |  |
| private |  |  |  |
| time\_created |  |  |  |
| time\_updated |  |  |  |
|  |  |  |  |
| Connection:: |  |  |  |
| \_id |  |  |  |
| user\_id |  |  |  |
| room\_id |  |  |  |
| time |  |  |  |
|  |  |  |  |
| Message:: |  |  |  |
| user\_id |  |  |  |
| room\_id |  |  |  |
| time |  |  |  |
| message |  |  |  |
|  |  |  |  |
| Message:: |  |  |  |
| user\_id |  |  |  |
| to\_user\_id |  |  |  |
| time |  |  |  |
| message |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# BACKBONE

« The layers are : **Server** communication, **Model** and **View**.

The **Model** bind the **Server.**

The **View** bind the **Model**.

The View bind the **User** actions. »

**Server**

**Model**

**Views**

**Collection**

**Bind**

**Trigger**

# Samples

* Entities
  + Users
    - Damien
    - David
    - Lili
    - Elio
    - Elvire
    - <from log>
  + Rooms
    - <from log>
* Activity
  + User connect (random 50%)
    - User join room (random)
    - User leave room (random)
    - User message (random)
    - User one to one to static users (random)
    - User disconnect (random 5%)

Content : <http://mg.pov.lt/maemo-meeting-irclog/index.html>