**Folder Structure of the Backend**

**app:** Contain the Model and Route of the application

**app > model:** Contain the schema of the database document. (Modeling tool use: Moongose)

**app > routes:** Contain routes or URL of the application

**config:** Contain the authentication for Client, Trainer, and oauth

**config > client-authentication:** Contain the client authentication for login and sign up using passport.js local for authentication, csrf(cross site request forgery) for csrf protection

**config > oauth-detail:** Contain the API key and API secret for facebook, twitter or google

**public:** Contain images, css, and javascript

**views:** Contain the view that will be rendered from when routes will be called from the server

**view > client:** Contain the client view, for logging in, signing up and profile

**view > trainer**: Contain the trainer view for logging in, signing up and profile

**view > index.ejs**: The Home page of the application

**server.js:** The server of the application, PORT used is 3001

**Routes for Client**

**/client/login (get)** – will render the view for client login form, each request from this url will contain new csrf token

**/client/login (post)** – this will validate the form from the get request, and if validated, will redirect/navigate to /client/profile

**/client/signup (get)** – will render the view for client sign up form, each request from this url will contain new csrf token

**/client/sign up (post) –** this will validate the form from the get request, and if validated, will redirect or navigate to /client/profile

**/client/auth/facebook** – this is the route that will call the facebook oauth and ask for facebook credentials

**/client/profile (get) –** will render the profile view of the client. It will also contain the data of the user that was save from the sessions (req.user) generated by passport.js

**/client/logout (get) –** will logout the user and destroy the session

**Routes for Trainer**

**/trainer/login (get)** – will render the view for trainer login form, each request from this url will contain new csrf token

**/trainer/login (post)** – this will validate the form from the get request, and if validated, will redirect/navigate to /trainer/profile

**/trainer/signup (get)** – will render the view for client sign up form, each request from this url will contain new csrf token

**/trainer/sign up (post) –** this will validate the form from the get request, and if validated, will redirect or navigate to /trainer /profile

**/ trainer /auth/facebook** – this is the route that will call the facebook oauth and ask for facebook credentials

**/trainer/profile (get) –** will render the profile view of the trainer. It will also contain the data of the user that was save from the sessions (req.user) generated by passport.js

**/trainer/profile/:id (get) –** will render the profile view of the trainer. That can be viewed by client even if the client is not logged in

**/trainer/contact (get)** – will render the contact form of the trainer, this will allow user to send an email to trainer

**/trainer/list (get)** – will render the list of trainers

**/trainer/update (get)** – will render the update form for trainer, can only update basic trainer’s information excluding the password and username

**/trainer/update (put)** – this will execute update after everything is validated, will only update basic trainer’s information excluding the password and username

**/trainer/logout (get) –** will logout the user and destroy the session

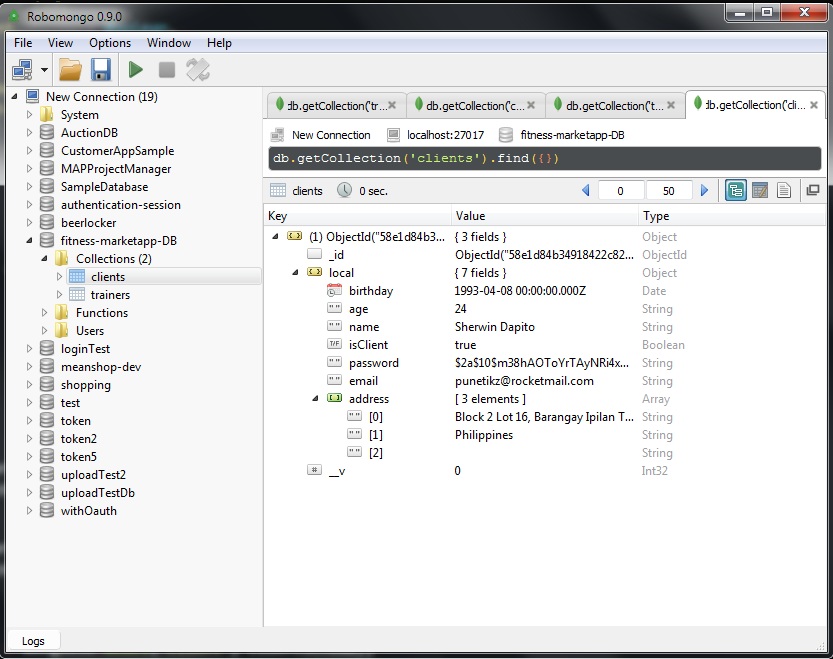
**Routes for Gym**

**/gym/create (get)** – will render the create view for creating a new Gym. For now only **one gym per trainer will be allowed** but the validatation for unique in trainers and mongoose schema has not been set up yet

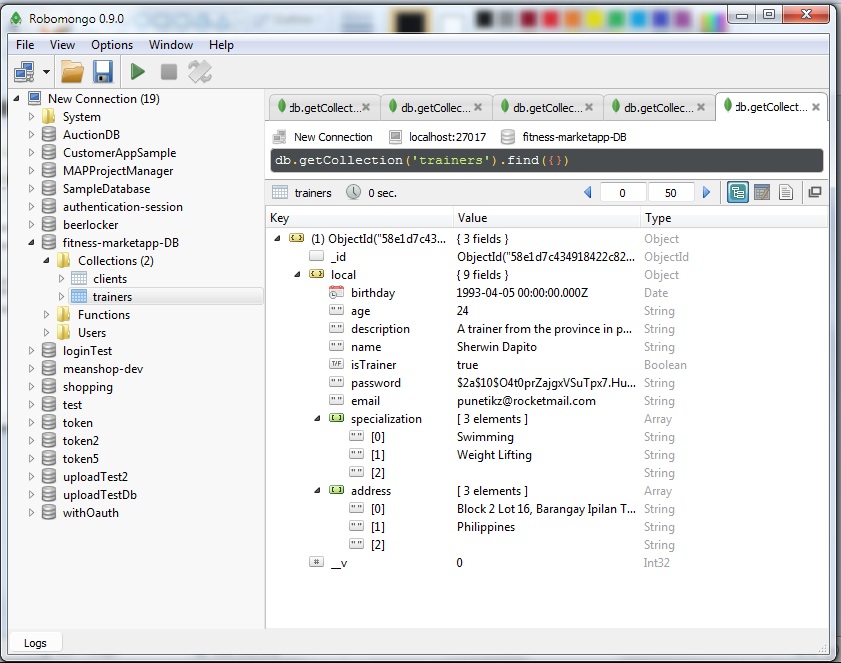
**/gym/create (post)** – this will execute after everything is validated, after successful creation will navigate to /gym. This will also update the gymInfo of trainer.

**/gym/** – Will render the view and generate the list of gym

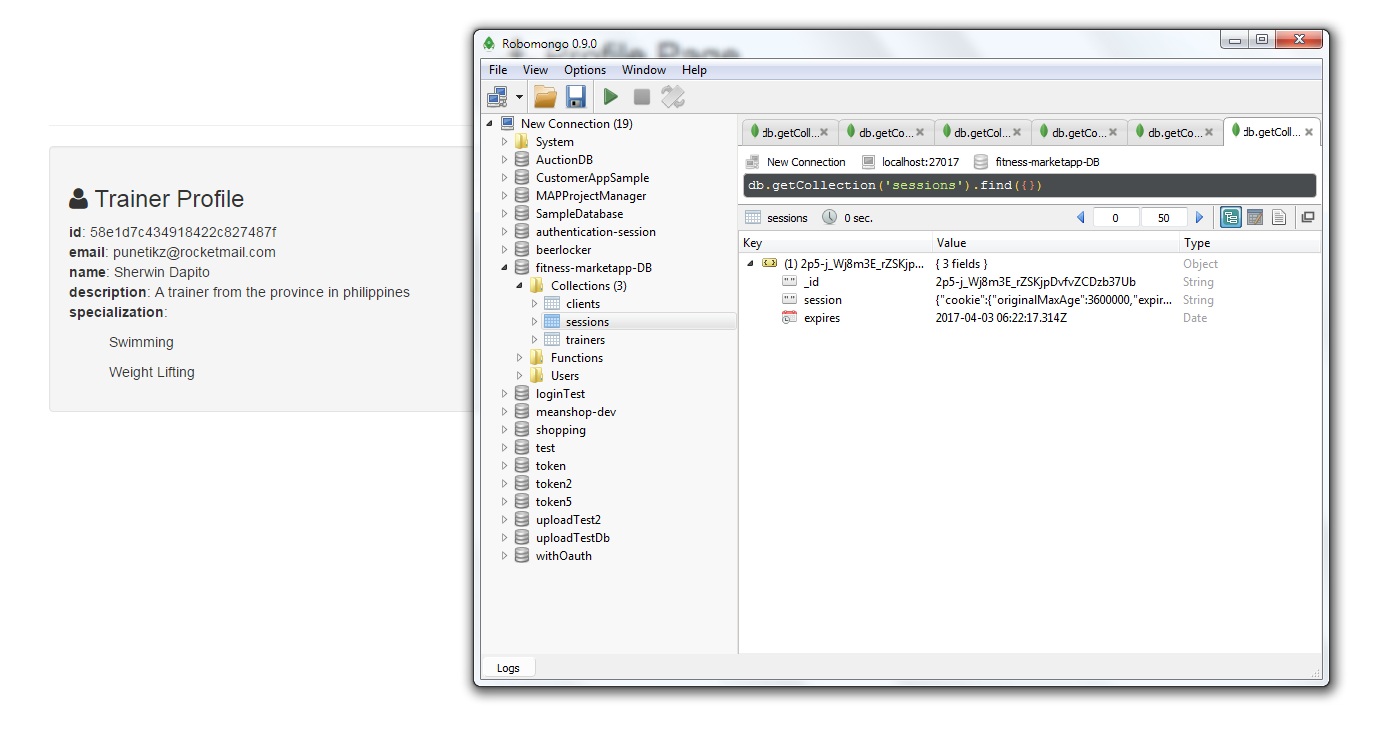
**/gym/:id** – Will render the view for gym profile



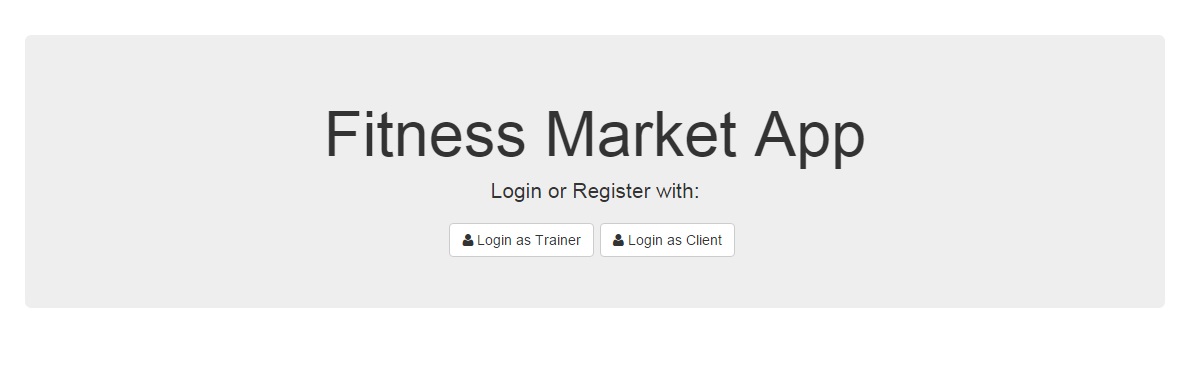
Sample document that was created after successful sign up using passport.js for clients

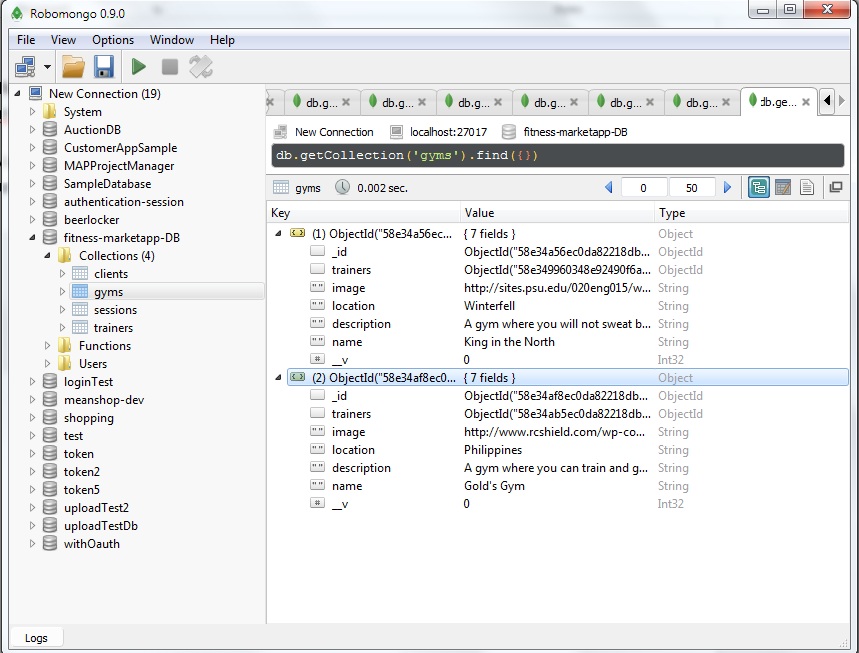


Sample document that was created after successful sign up using passport.js for trainers

Every successful log in will be stored in a session. Each session will be stored on a MongoDB. The middleware that was used for this is the connect-mongo module. Cookie will expire in one hour.

**Note**: You can now use the app without setting up a mongo database, I used mongolab or mlab.com to host my mongoDB online.

**Mongolab account**: sarwinmongo, **password**: 01610715awdrgEZCQ@B#N  
  
Authentication for facebook is still on test mode, can only be authenticated by using my facebook (dapito.sherwin@yahoo.com) account

This is the sample of the relationship of Gym and Trainer by their Object(ID), the data of the trainer can be populated using the populate keyword

