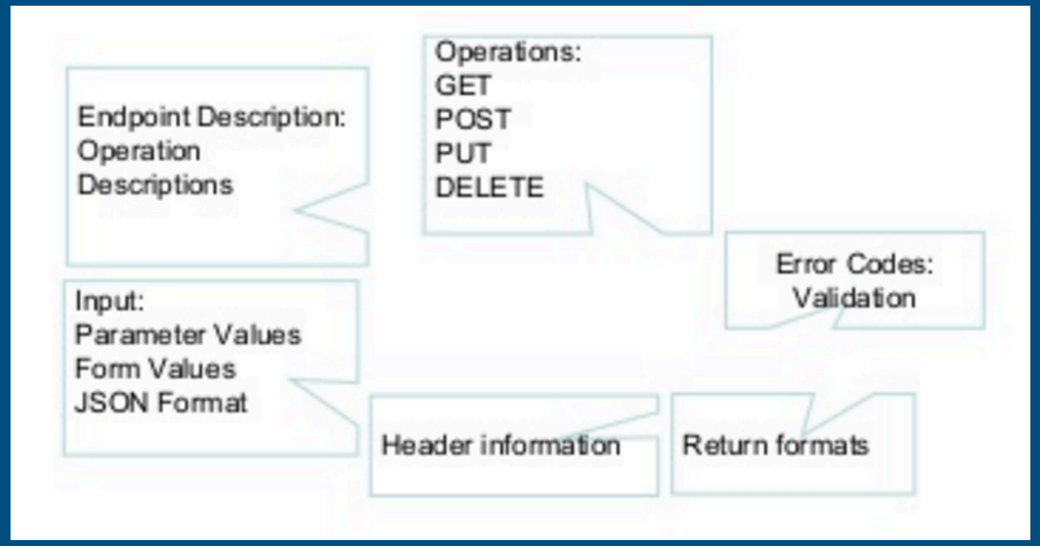
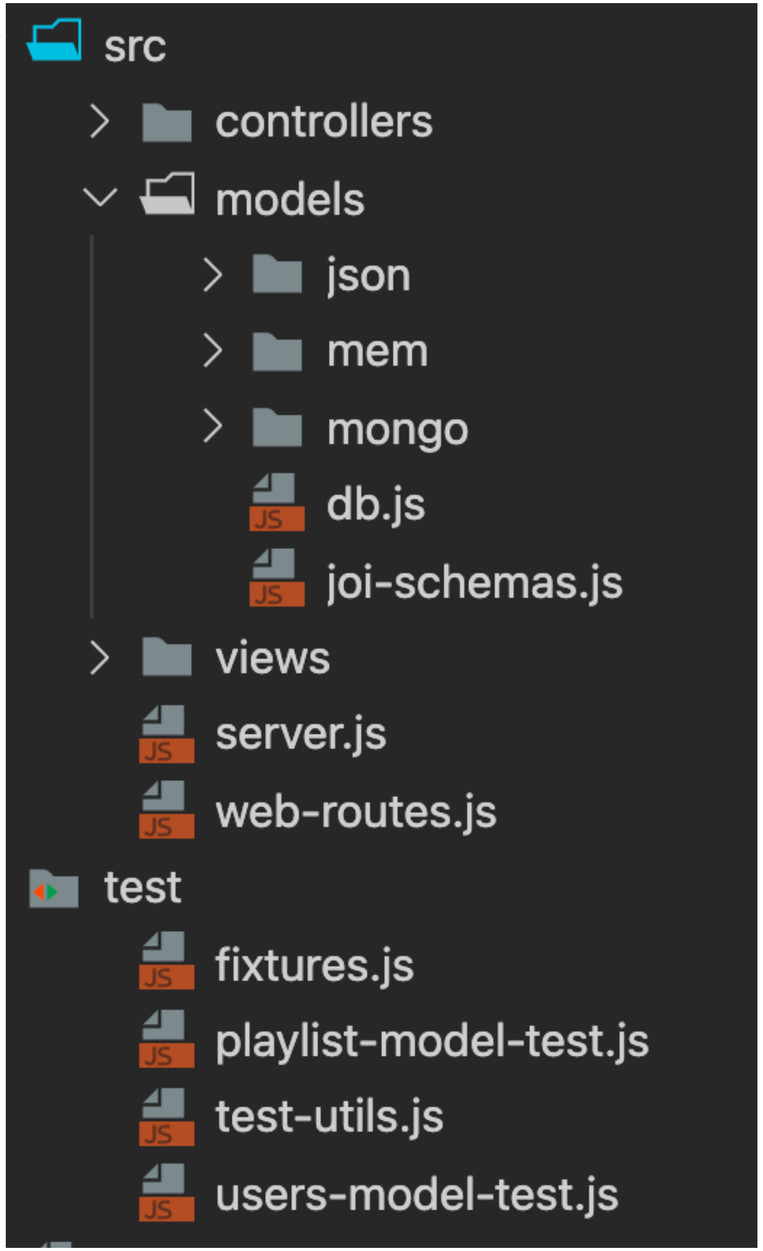
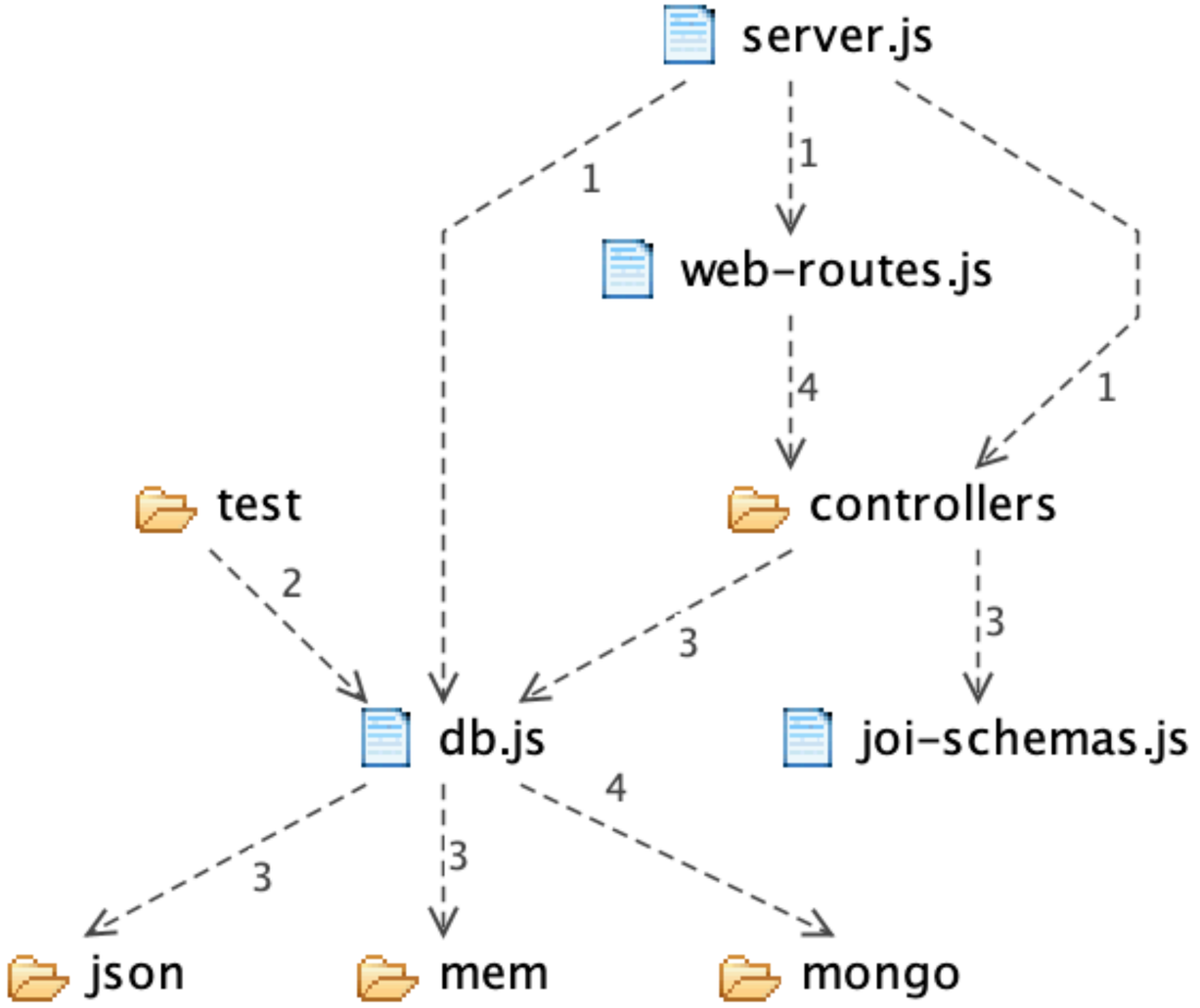
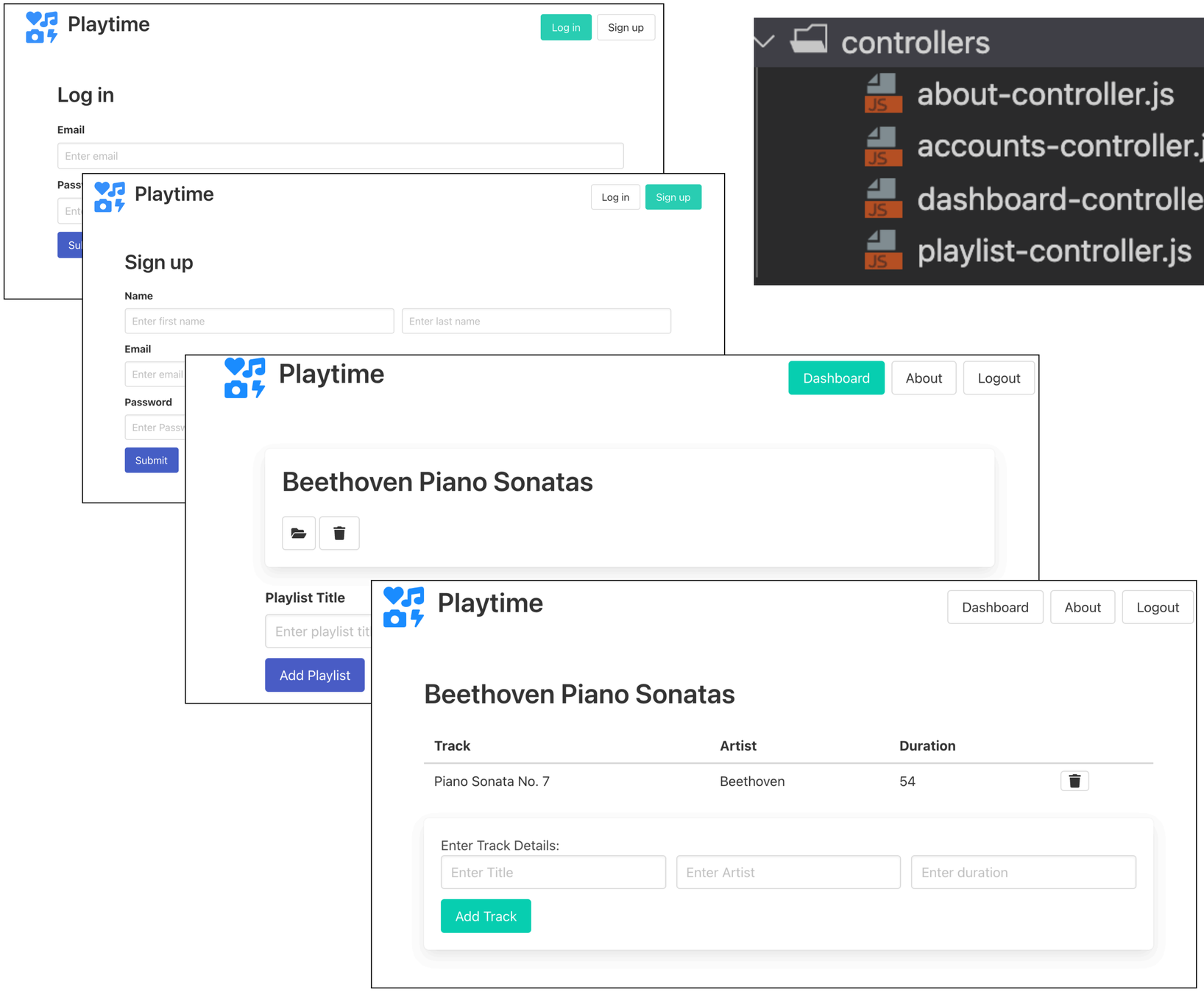
End Points



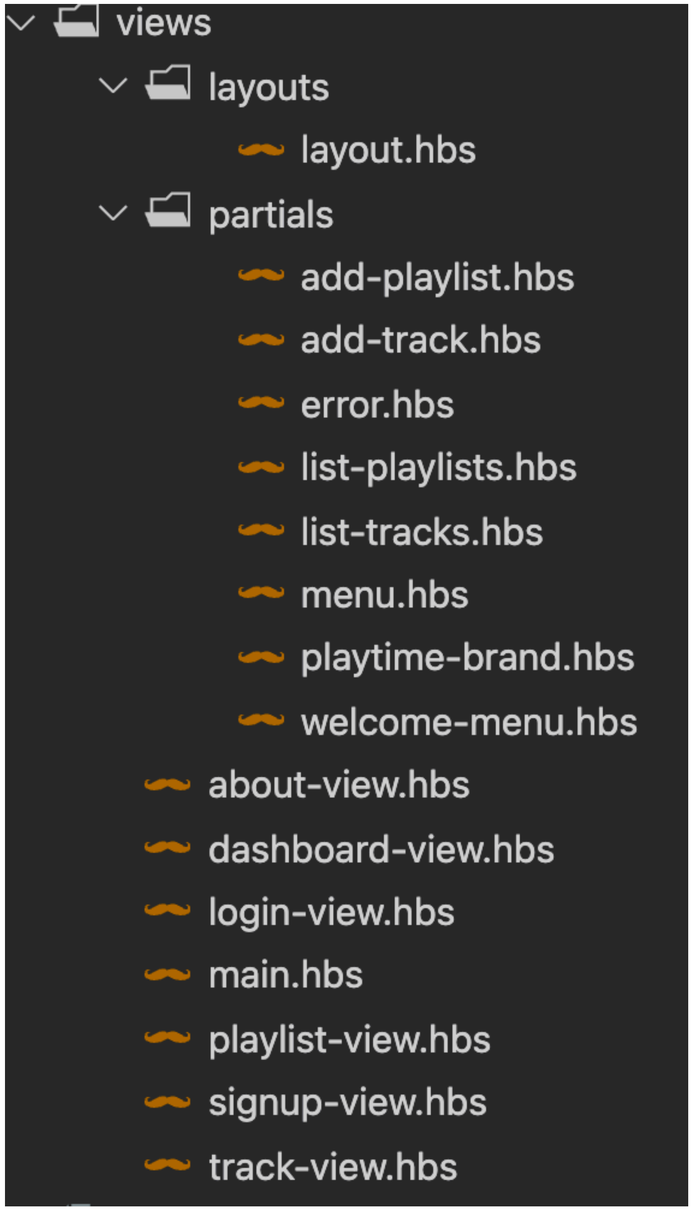
Full Stack Web Development

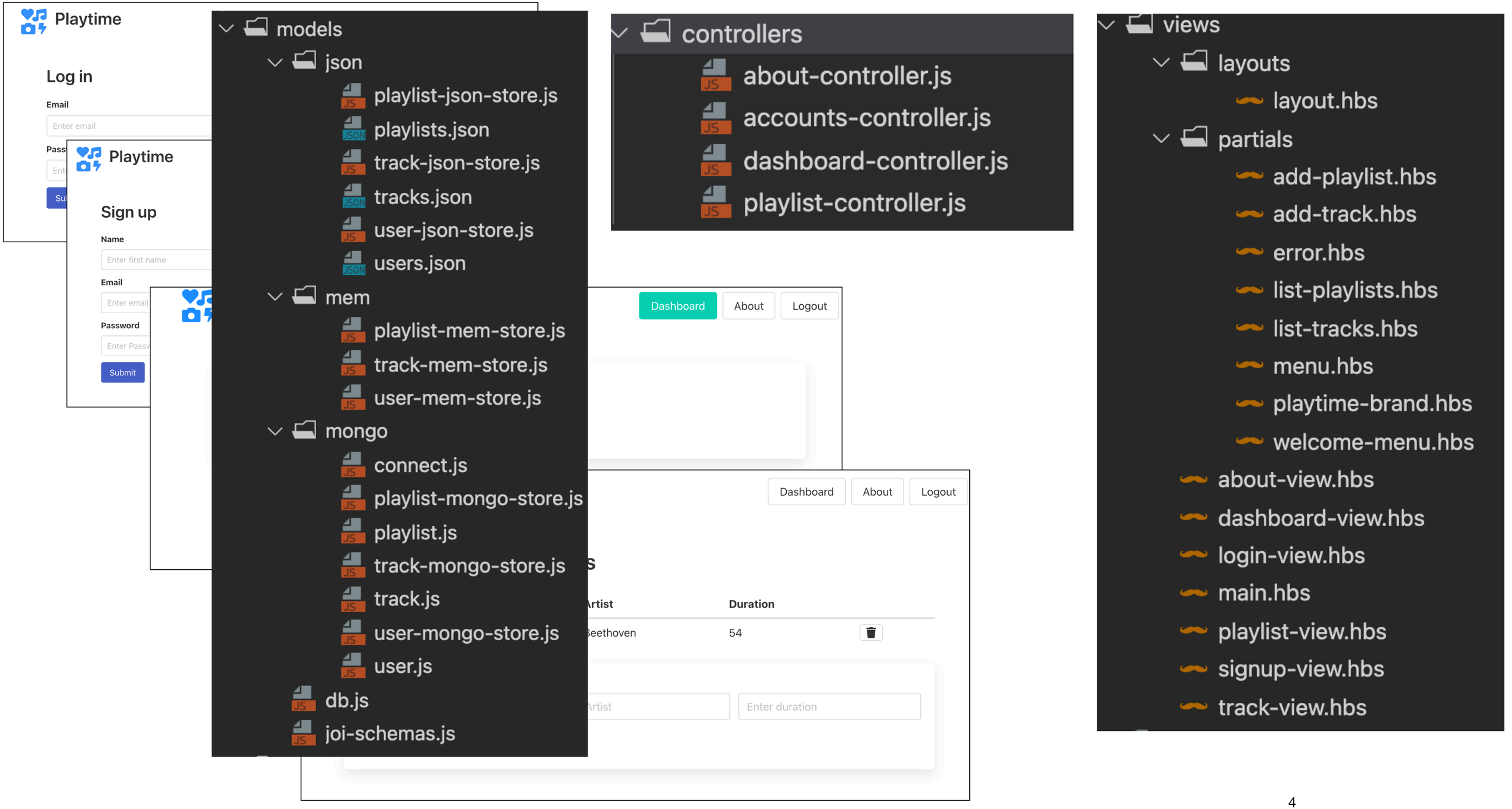

src
controllers
models
json
mem
mongo
db.js
joi-schemas.js
views
server.js
web-routes.js
test
fixtures.js
playlist-model-test.js
test-utils.js
users-model-test.js

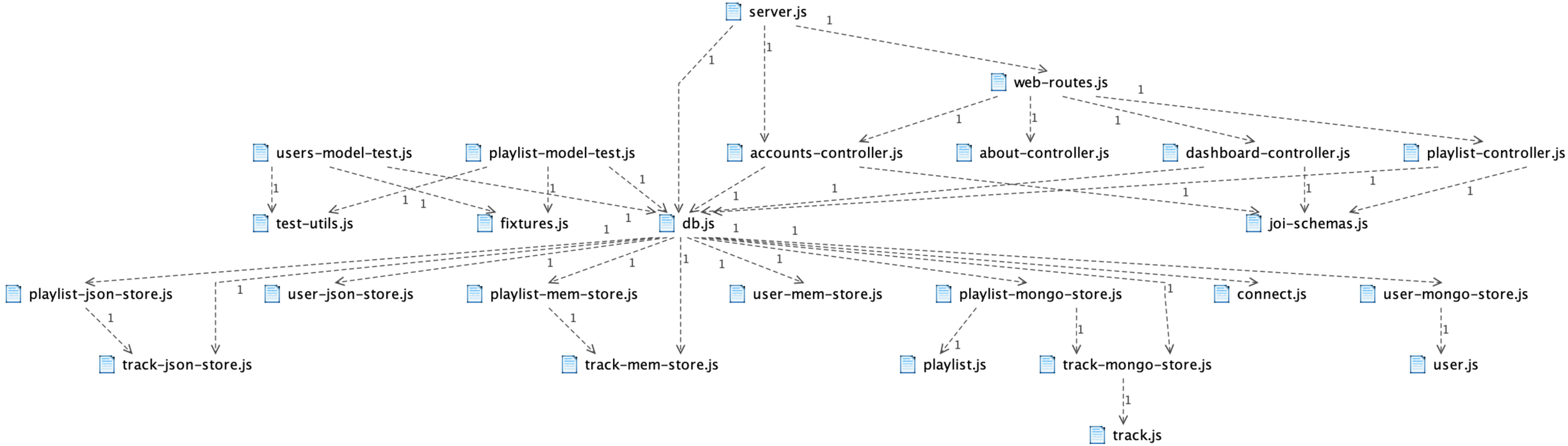

server.js
1
1
1
web-routes.js
4
1
test
controllers
2
3
3
db.js
joi-schemas.js
4
3
3
json
mem
mongo


Playtime
Log in
Sign up
Log in
Email
Enter email
Playtime
Log in
Sign up
Sign up
Name
Enter first name
Enter last name
Email
Password
Submit
Playtime
Dashboard
About
Logout
Beethoven Piano Sonatas
Playlist Title
Add Playlist
Playtime
Dashboard
About
Logout
Beethoven Piano Sonatas
Track
Artist
Duration
Piano Sonata No. 7
Beethoven
54
Enter Track Details:
Enter Title
Enter Artist
Enter duration
Add Track

controllers
about-controller.js
accounts-controller.js
dashboard-controller.js
playlist-controller.js

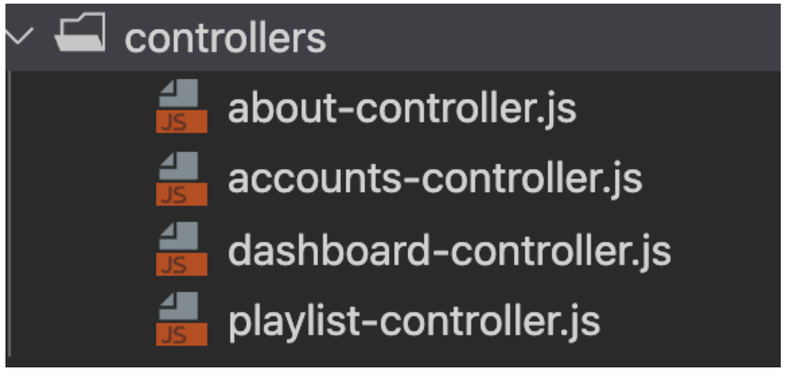

views
layouts
layout.hbs
partials
add-playlist.hbs
add-track.hbs
error.hbs
list-playlists.hbs
list-tracks.hbs
menu.hbs
playtime-brand.hbs
welcome-menu.hbs
about-view.hbs
dashboard-view.hbs
login-view.hbs
main.hbs
playlist-view.hbs
signup-view.hbs
track-view.hbs


Playtime
Log in
Email
Enter email
Playtime
Sign up
Name
Enter first name
Email
Password
Submit
Dashboard
About
Logout
Artist
Duration
Beethoven
54
Enter duration
models
json
playlist-json-store.js
playlists.json
track-json-store.js
tracks.json
user-json-store.js
users.json
mem
playlist-mem-store.js
track-mem-store.js
user-mem-store.js
mongo
connect.js
playlist-mongo-store.js
playlist.js
track-mongo-store.js
track.js
user-mongo-store.js
user.js
db.js
joi-schemas.js
controllers
about-controller.js
accounts-controller.js
dashboard-controller.js
playlist-controller.js
views
layouts
layout.hbs
partials
add-playlist.hbs
add-track.hbs
error.hbs
list-playlists.hbs
list-tracks.hbs
menu.hbs
playtime-brand.hbs
welcome-menu.hbs
about-view.hbs
dashboard-view.hbs
login-view.hbs
main.hbs
playlist-view.hbs
signup-view.hbs
track-view.hbs

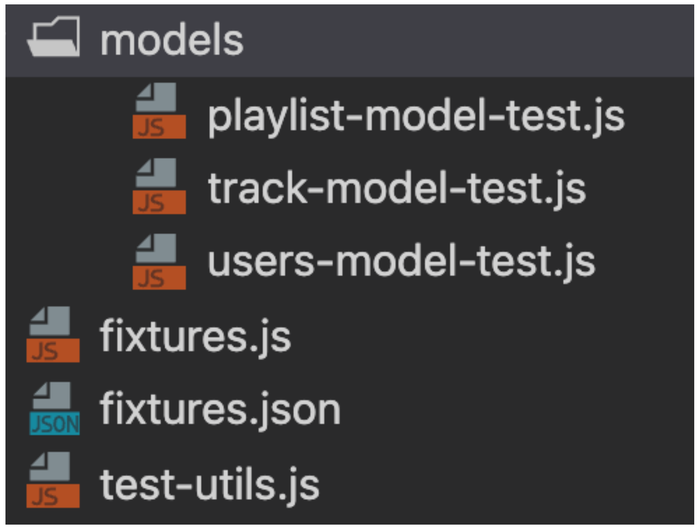

server.js
web-routes.js
users-model-test.js
playlist-model-test.js
accounts-controller.js
about-controller.js
dashboard-controller.js
playlist-controller.js
test-utils.js
fixtures.js
db.js
joi-schemas.js
playlist-json-store.js
user-json-store.js
playlist-mem-store.js
user-mem-store.js
playlist-mongo-store.js
connect.js
user-mongo-store.js
track-json-store.js
track-mem-store.js
playlist.js
track-mongo-store.js
user.js
track.js

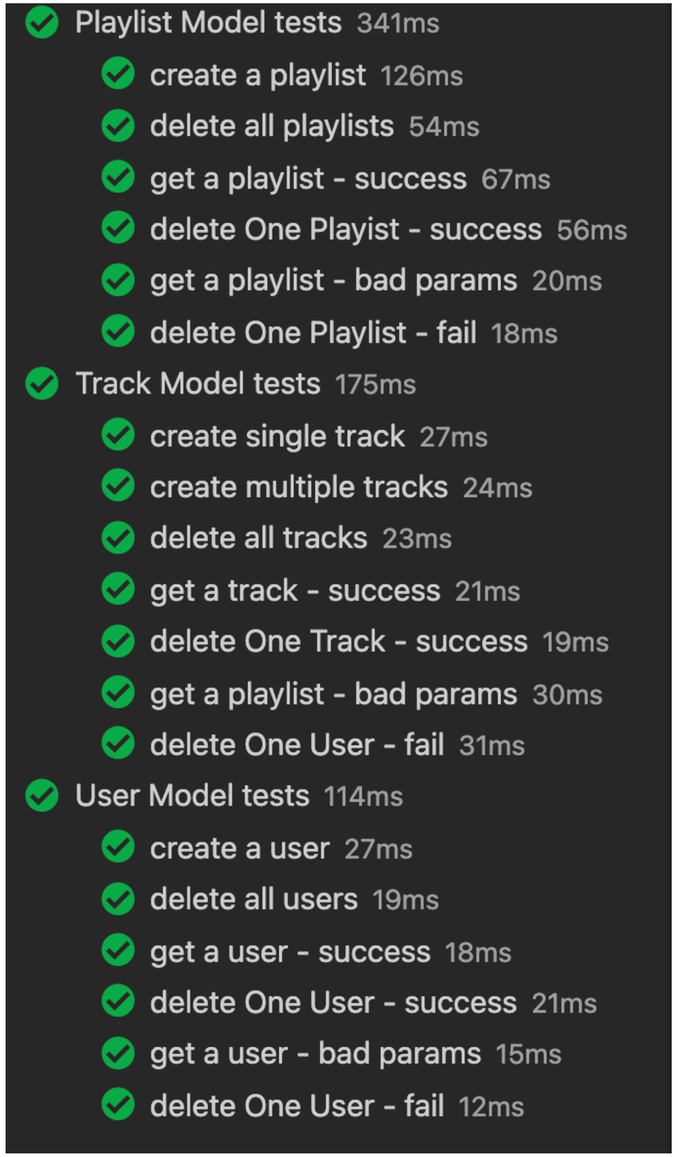
web-routes.js

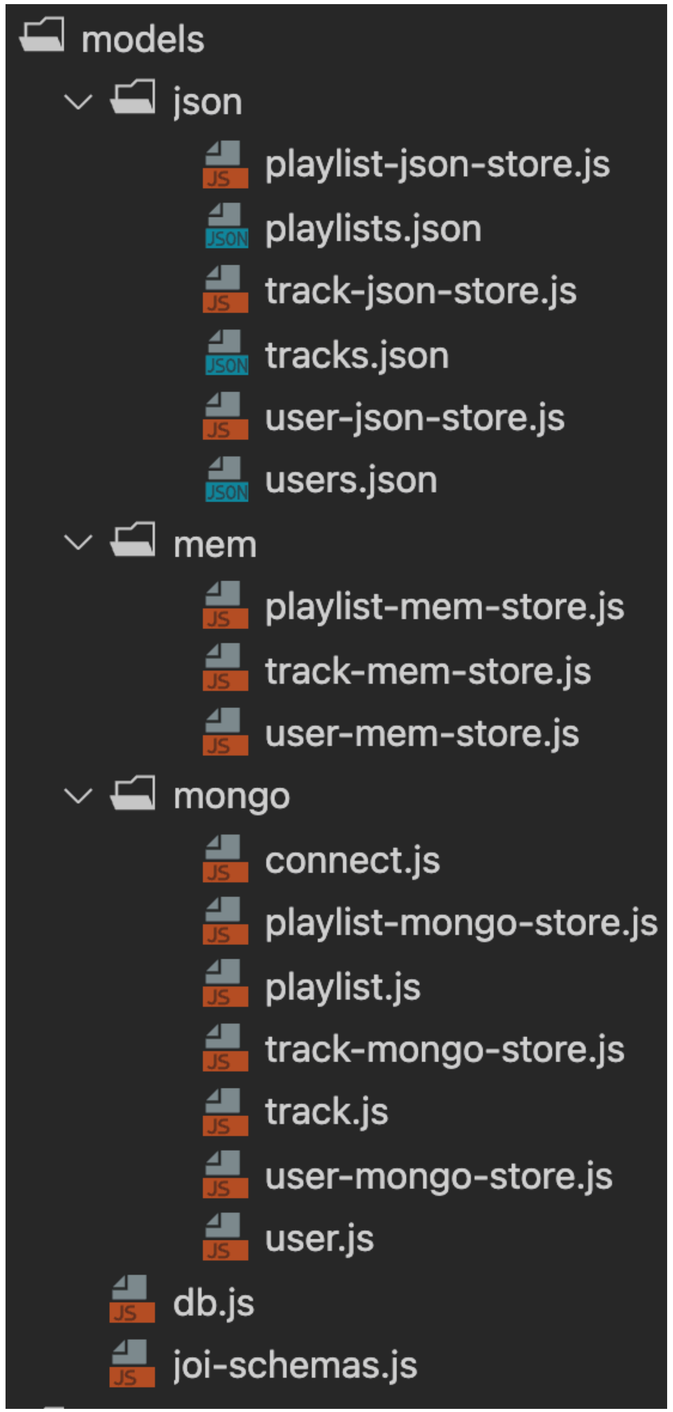
```
import { aboutController } from "./controllers/about-controller.js";
import { accountsController } from "./controllers/accounts-controller.js";
import { dashboardController } from "./controllers/dashboard-controller.js";
import { playlistController } from "./controllers/playlist-controller.js";

export const webRoutes = [
  { method: "GET", path: "/", config: accountsController.index },
  { method: "GET", path: "/signup", config: accountsController.showSignup },
  { method: "GET", path: "/login", config: accountsController.showLogin },
  { method: "GET", path: "/logout", config: accountsController.logout },
  { method: "POST", path: "/register", config: accountsController.signup },
  { method: "POST", path: "/authenticate", config: accountsController.login },

  { method: "GET", path: "/about", config: aboutController.index },

  { method: "GET", path: "/dashboard", config: dashboardController.index },
  { method: "POST", path: "/dashboard/addplaylist", config: dashboardController.addPlaylist },
  { method: "GET", path: "/dashboard/deleteplaylist/{id}", config: dashboardController.deletePlaylist },

  { method: "GET", path: "/playlist/{id}", config: playlistController.index },
  { method: "POST", path: "/playlist/{id}/addtrack", config: playlistController.addTrack },
  { method: "GET", path: "/playlist/{id}/deletetrack/{trackid}", config: playlistController.deleteTrack },
];
```



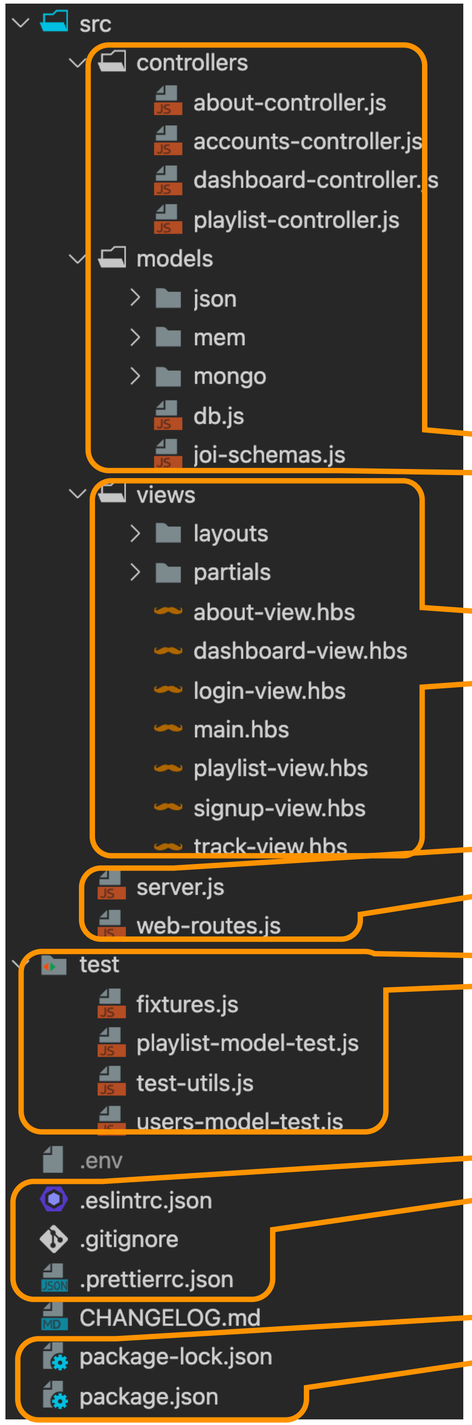
Unit tests exercise the model directly



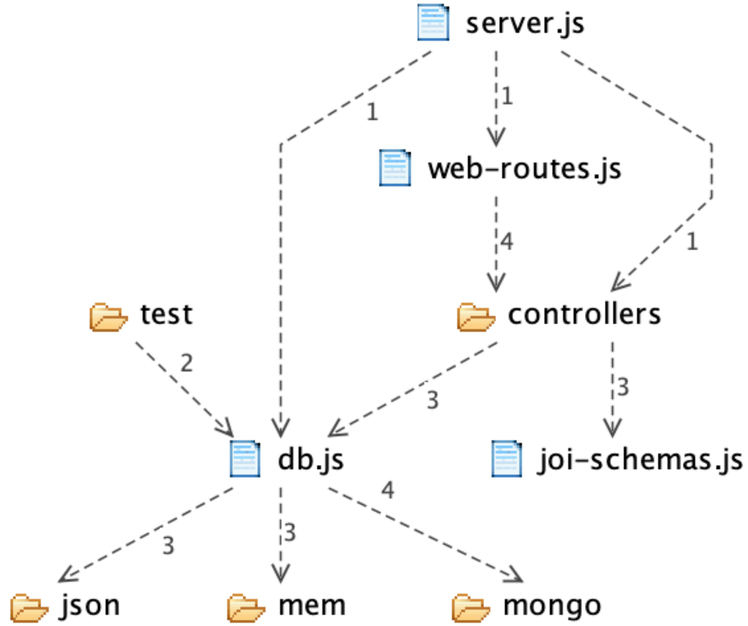


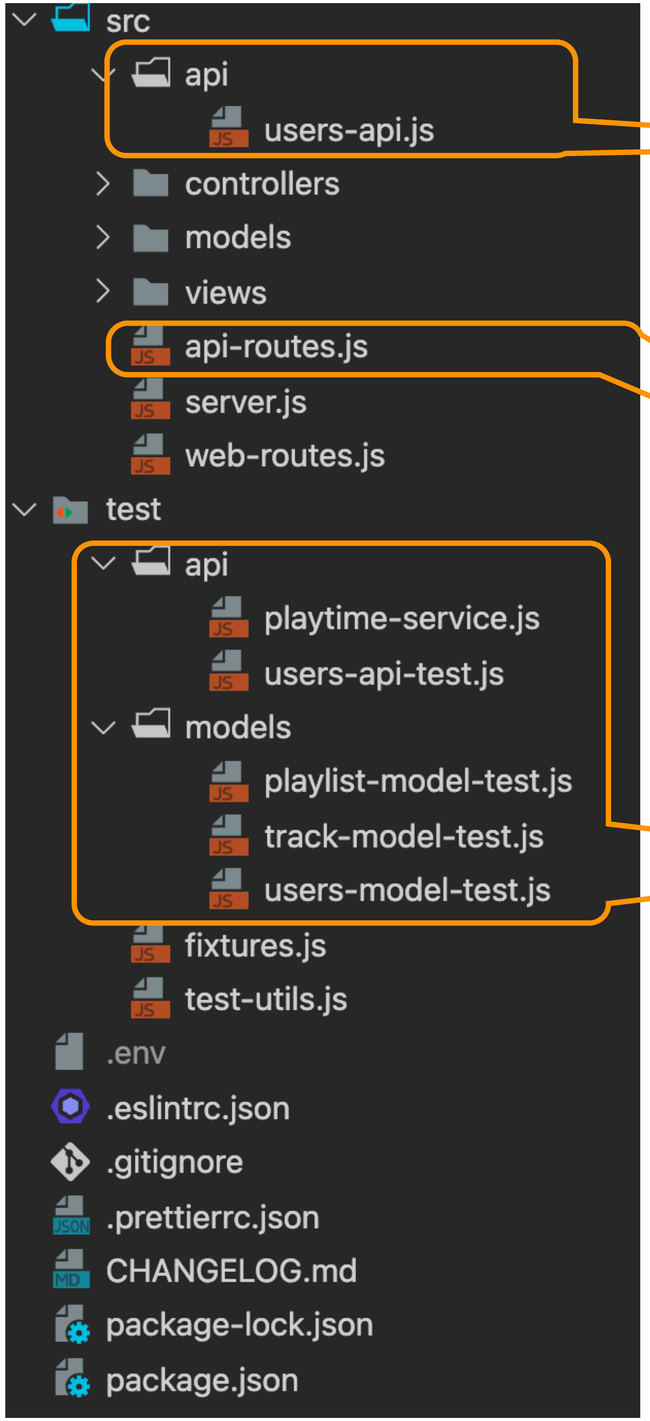


Project Structure



- Back End
- Front End
- Main + Routes
- Model Tests
- Code Quality Config
- Project Dependencies





- Define controllers to handle API Routes
- Define API Routes
- Refactor Tests to Incorporate separate Model + API Tests

Introduce API

Get Users Endpoint

server.js

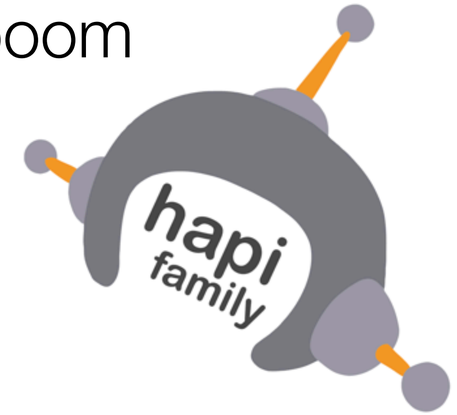
```
import { apiRoutes } from "./api-routes.js";

...
  db.init("mongo");
  server.route(webRoutes);
  server.route(apiRoutes);
  await server.start();
...
```

api-routes.js

```
import { usersApi } from "./api/users-api.js";

export const apiRoutes = [
  { method: "GET", path: "/api/users", config: usersApi.find },
];
```

https://github.com/hapijs/boom

@hapi/boom



HTTP-friendly error objects.

boom is part of the **hapi** ecosystem and was designed to work seamlessly with the hapi web framework and its other components (but works great on its own or with other frameworks). If you are using a different web framework and find this module useful, check out hapi – they work even better together.

Visit the hapi.dev Developer Portal for tutorials, documentation, and support

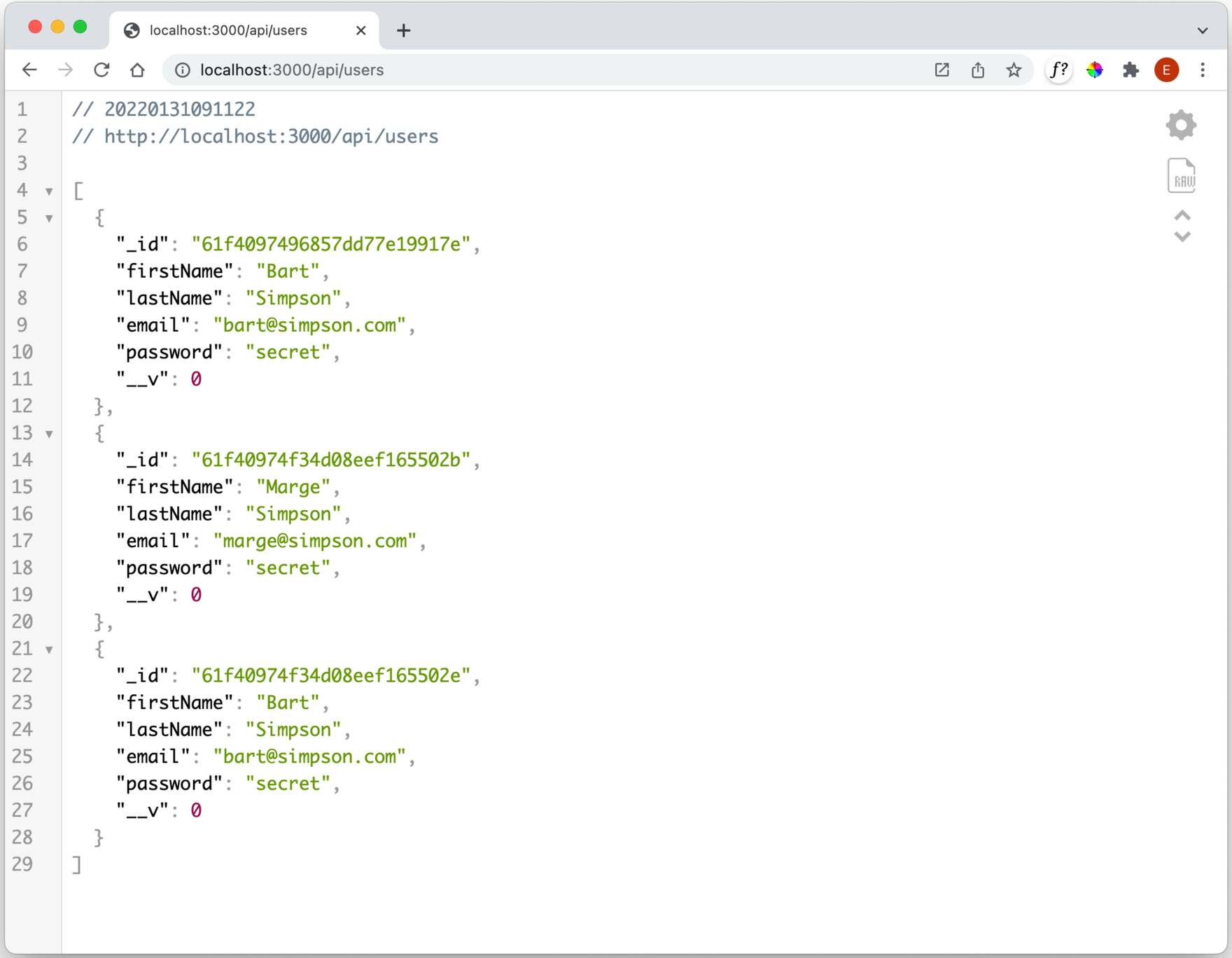
Useful resources

- Documentation and API
- Version status (builds, dependencies, node versions, licenses, eol)
- Changelog
- Project policies
- Free and commercial support options

Get Users Implementation

Exception in Database unavailable / in error state

```
import Boom from "@hapi/boom";
import { db } from "../models/db.js";

  find: {
    auth: false,
    handler: async function(request, h) {
      try {
        const users = await db.userStore.getAllUsers();
        return users;
      } catch (err) {
        return Boom.serverUnavailable("Database Error");
      }
    },
  },
};
```


localhost:3000/api/users
localhost:3000/api/users
// 20220131091122
// http://localhost:3000/api/users
[
{
"_id": "61f4097496857dd77e19917e",
"firstName": "Bart",
"lastName": "Simpson",
"email": "bart@simpson.com",
"password": "secret",
"__v": 0
},
{
"_id": "61f40974f34d08eef165502b",
"firstName": "Marge",
"lastName": "Simpson",
"email": "marge@simpson.com",
"password": "secret",
"__v": 0
},
{
"_id": "61f40974f34d08eef165502e",
"firstName": "Bart",
"lastName": "Simpson",
"email": "bart@simpson.com",
"password": "secret",
"__v": 0
}
]
RAW

Create User Endpoint

api-routes.js

```
import { usersApi } from "./api/users-api.js";

export const apiRoutes = [
  { method: "POST", path: "/api/users", config: usersApi.create },
  { method: "GET", path: "/api/users", config: usersApi.find },
];
```

Create User Implementation

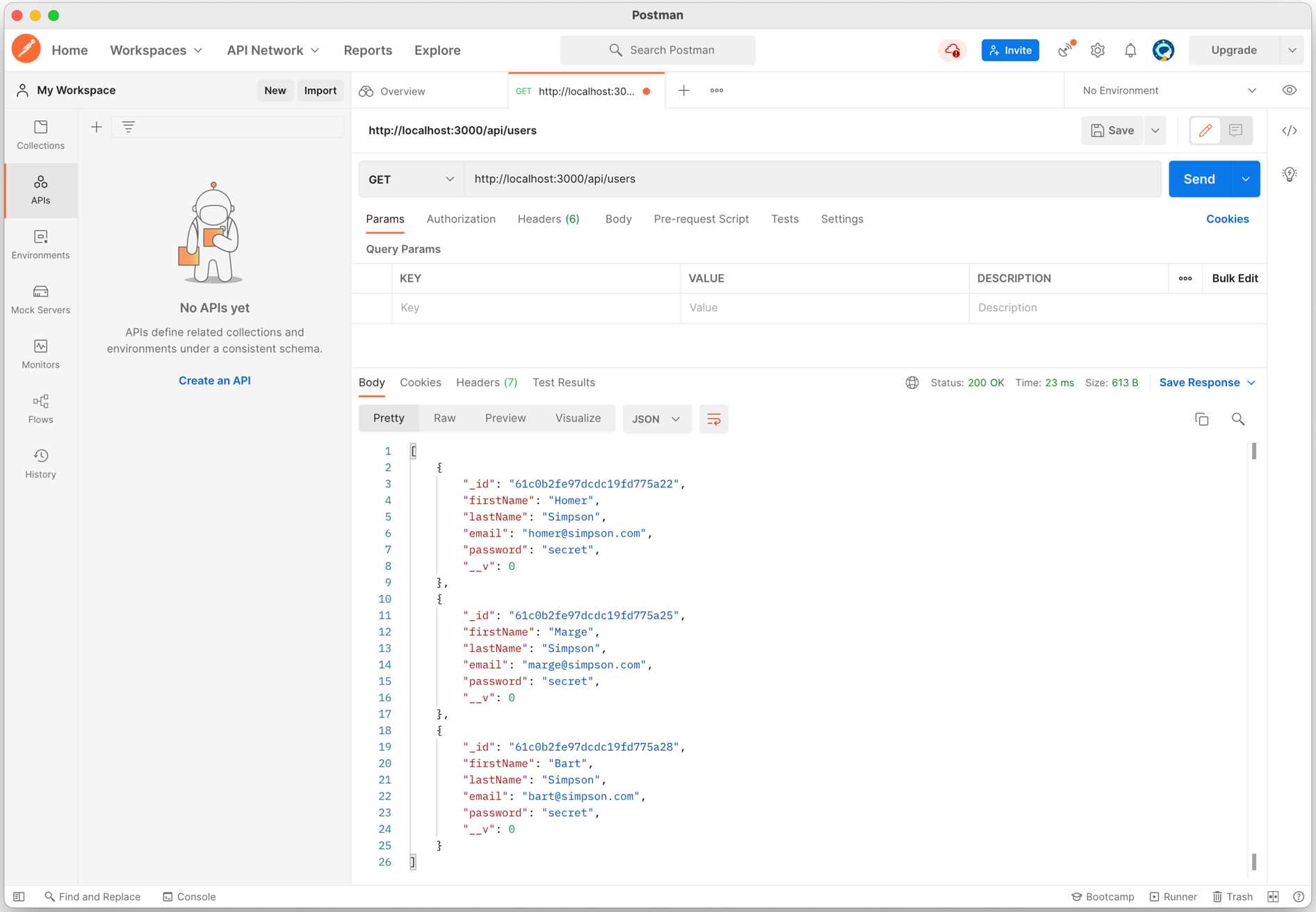
Exception creating user object

Exception if Database unavailable / in error state

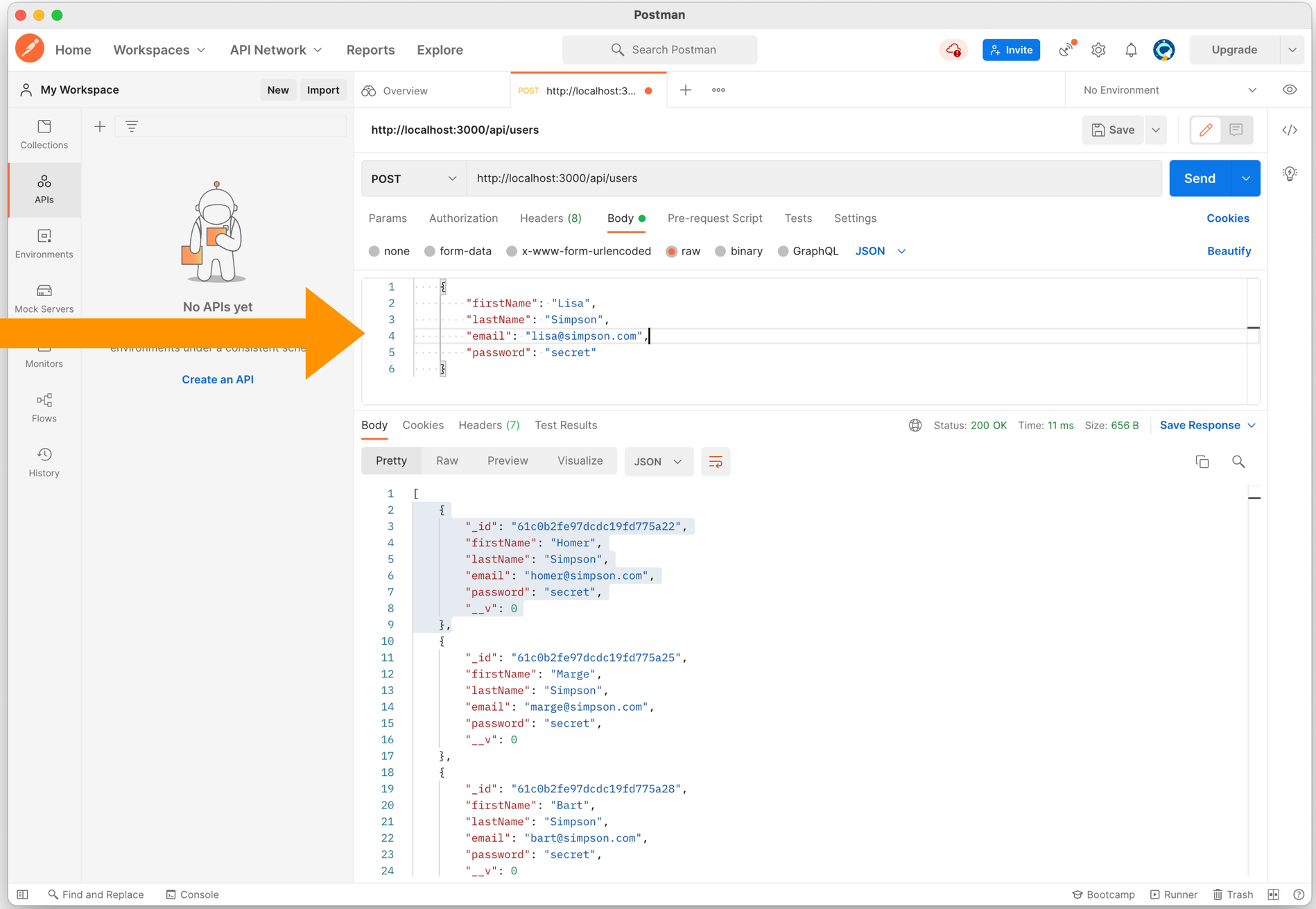
users-api.js

```
import Boom from "@hapi/boom";
import { db } from "../models/db.js";

export const usersApi = {
  create: {
    auth: false,
    handler: async function(request, h) {
      try {
        const user = await db.userStore.addUser(request.payload);
        if (user) {
          return h.response(user).code(201);
        }
        return Boom.badImplementation("error creating user");
      } catch (err) {
        return Boom.serverUnavailable("Database Error");
      }
    },
  },
};
```

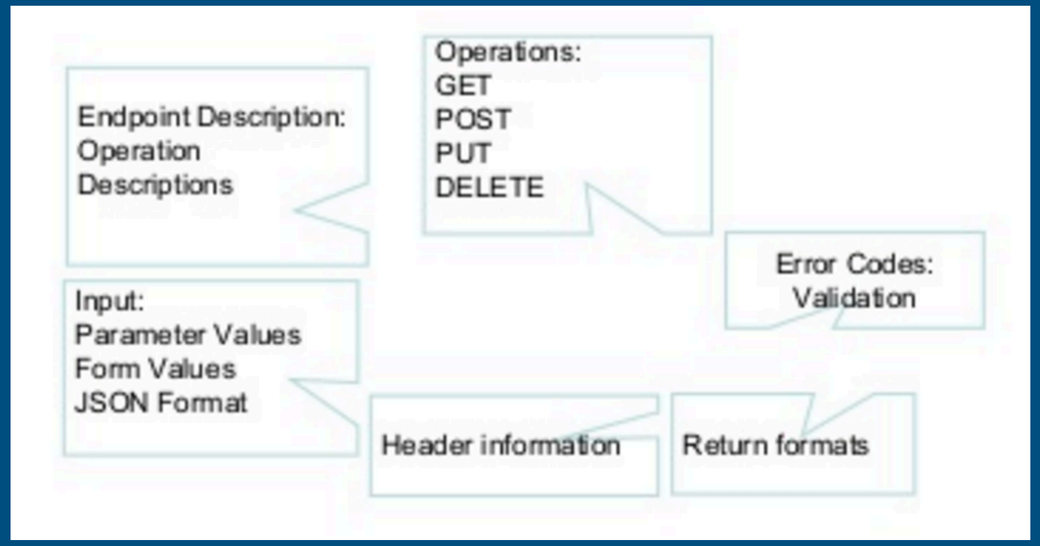
- To POST a user, need additional tools
- POSTMAN



- POST Request
- New User Details



End Points



Full Stack Web Development