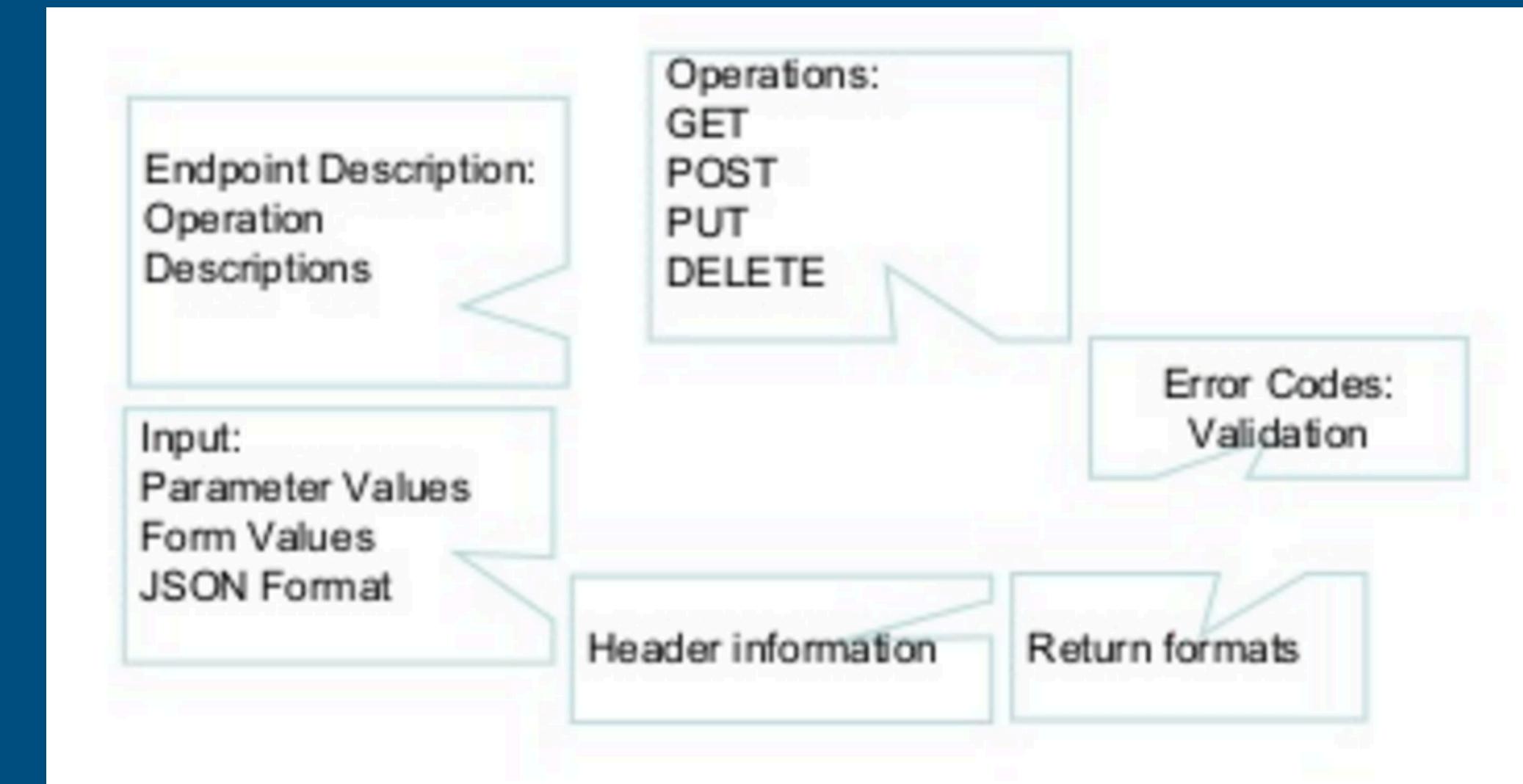
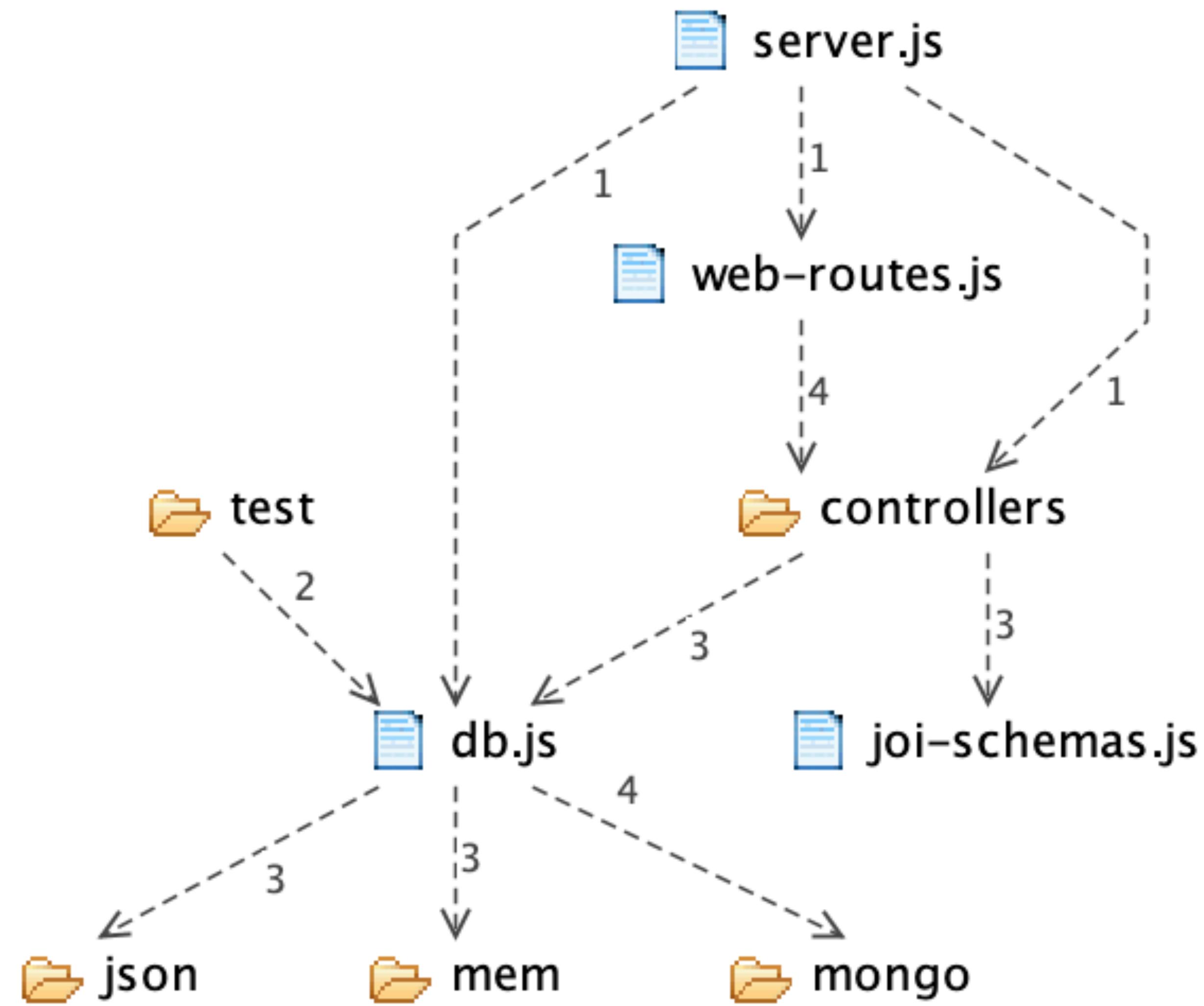
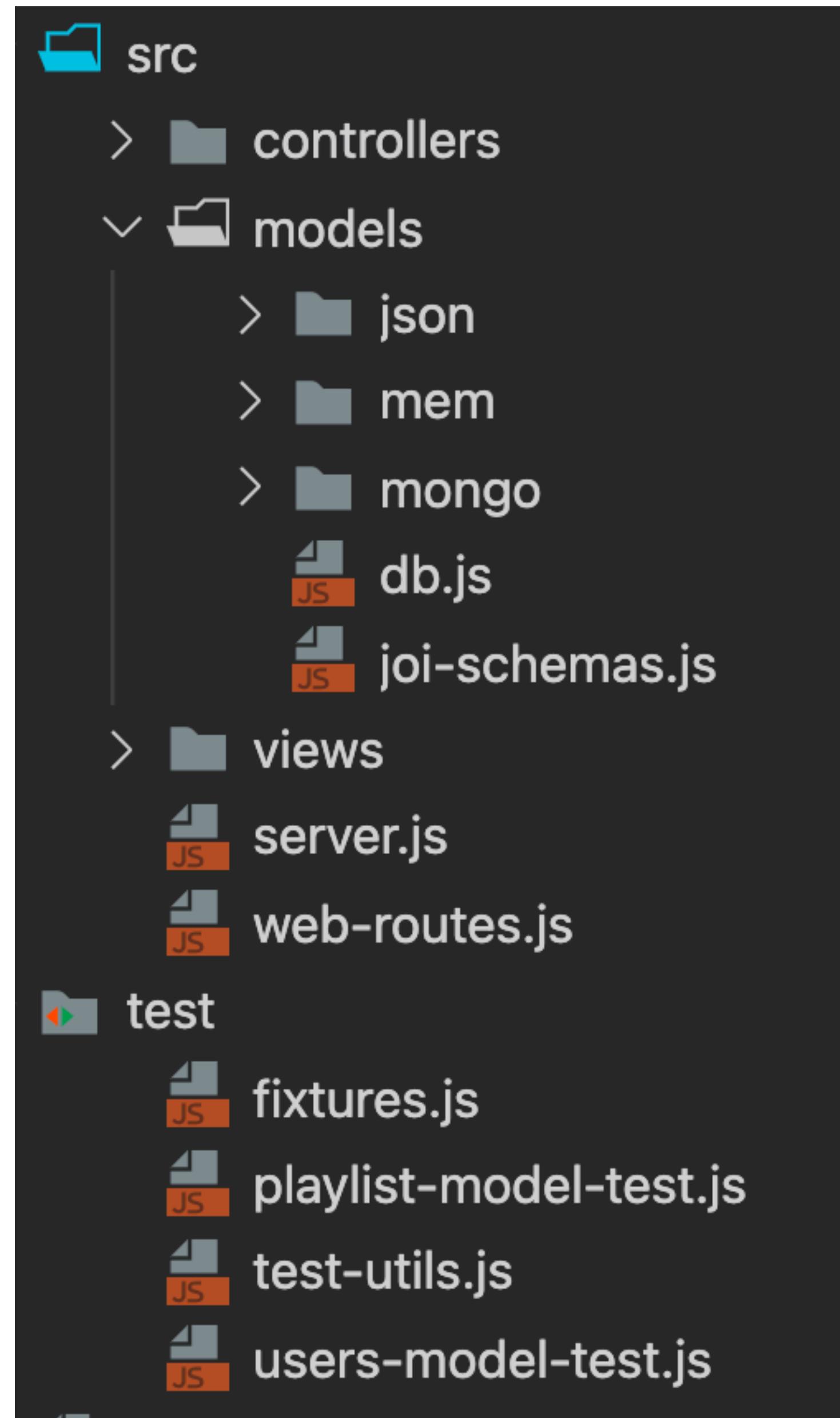


End Points



Full Stack Web Development



 Playtime

Log in

Email

Password

 Playtime

Sign up

Name

Email

Password

 Playtime

Dashboard About Logout

Beethoven Piano Sonatas

 Playtime

Dashboard About Logout

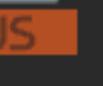
Beethoven Piano Sonatas

Track	Artist	Duration	
Piano Sonata No. 7	Beethoven	54	

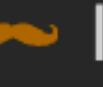
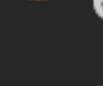
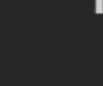
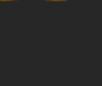
Enter Track Details:

Enter Title
Enter Artist
Enter duration

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

Pass

Enter

Sub



Sign up

Name

Enter first name

Email

Enter email

Password

Enter Passw

Submit

models

json

- playlist-json-store.js
- playlists.json
- track-json-store.js
- tracks.json
- user-json-store.js
- users.json

mem

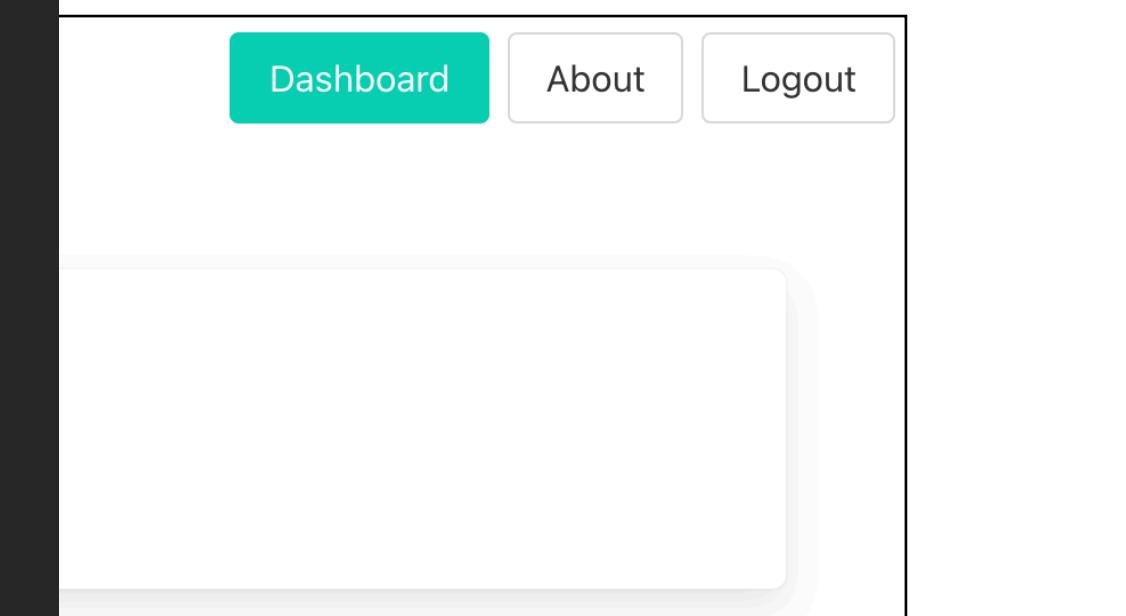
- playlist-mem-store.js
- track-mem-store.js
- user-mem-store.js

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



Artist	Duration	
Beethoven	54	
Artist	Enter duration	

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

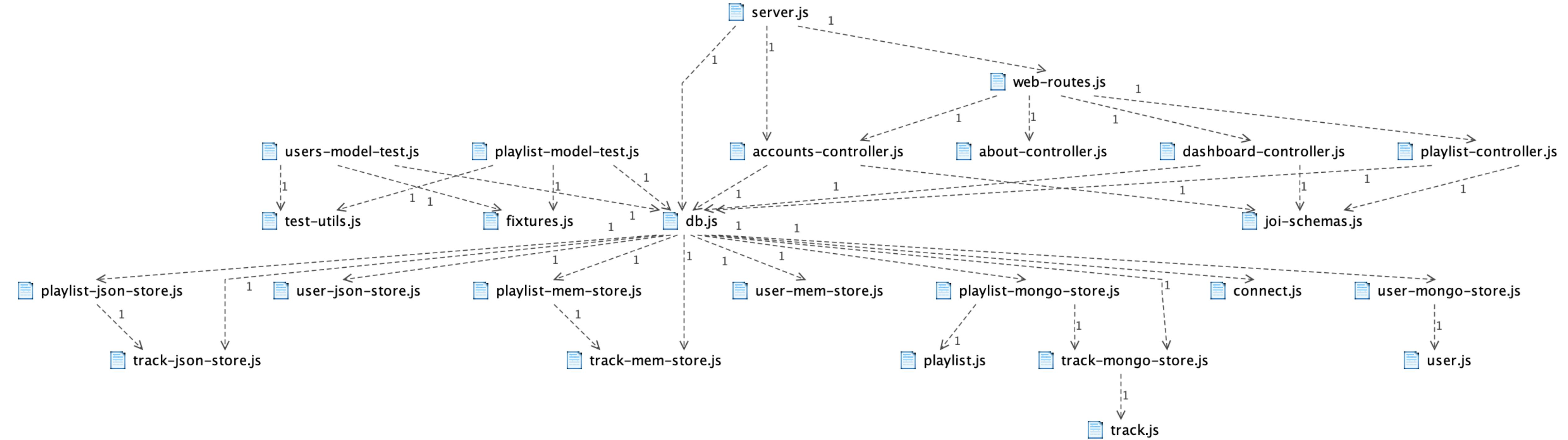
login-view.hbs

main.hbs

playlist-view.hbs

signup-view.hbs

track-view.hbs



web-routes.js

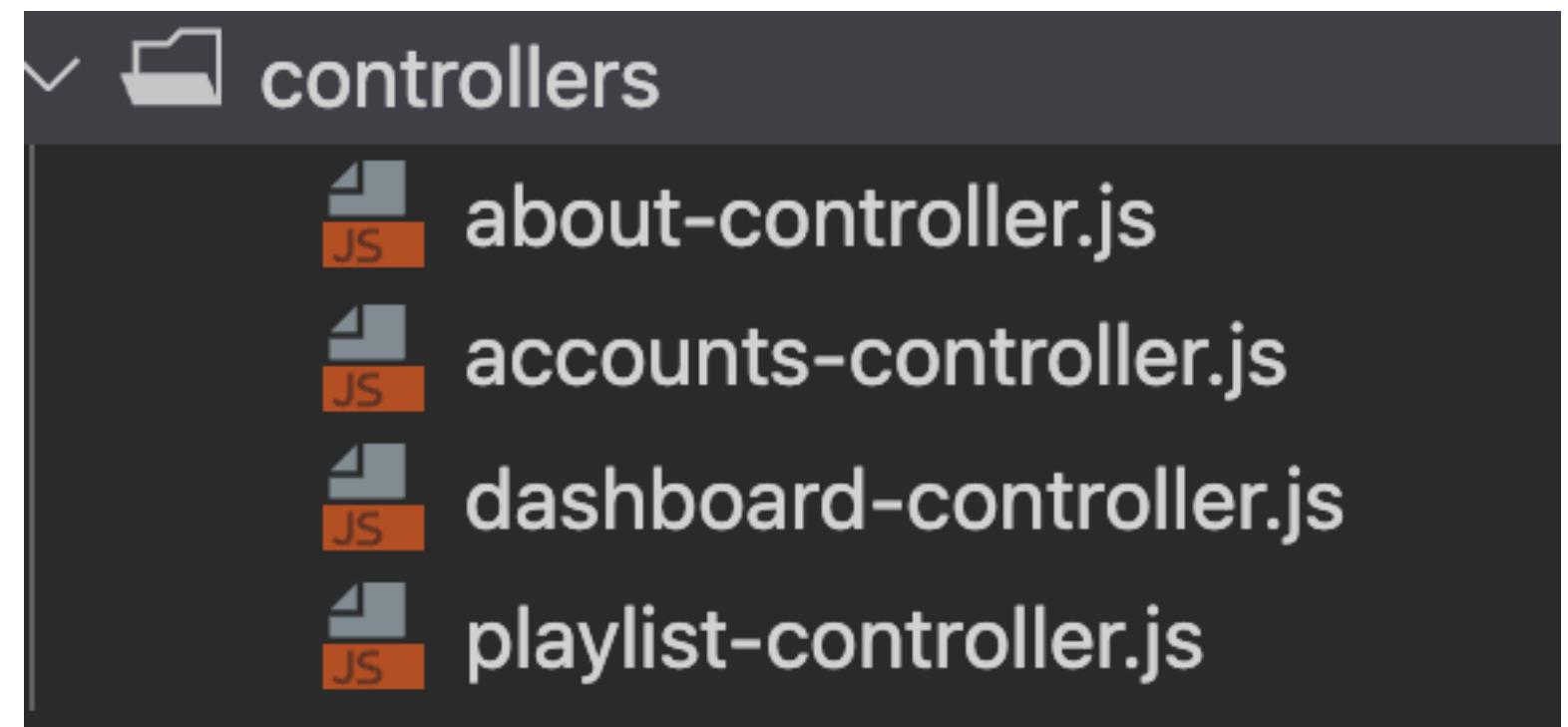
```
import { aboutController } from "./controllers/about-controller.js";
import { accountsController } from "./controllers/accounts-controller.js";
import { dashboardController } from "./controllers/dashboard-controller.js";
import { playlistController } from "./controllers/playlist-controller.js";

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 },

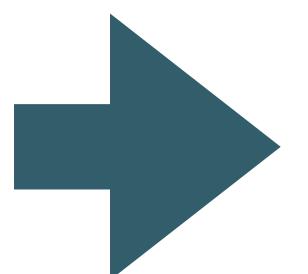
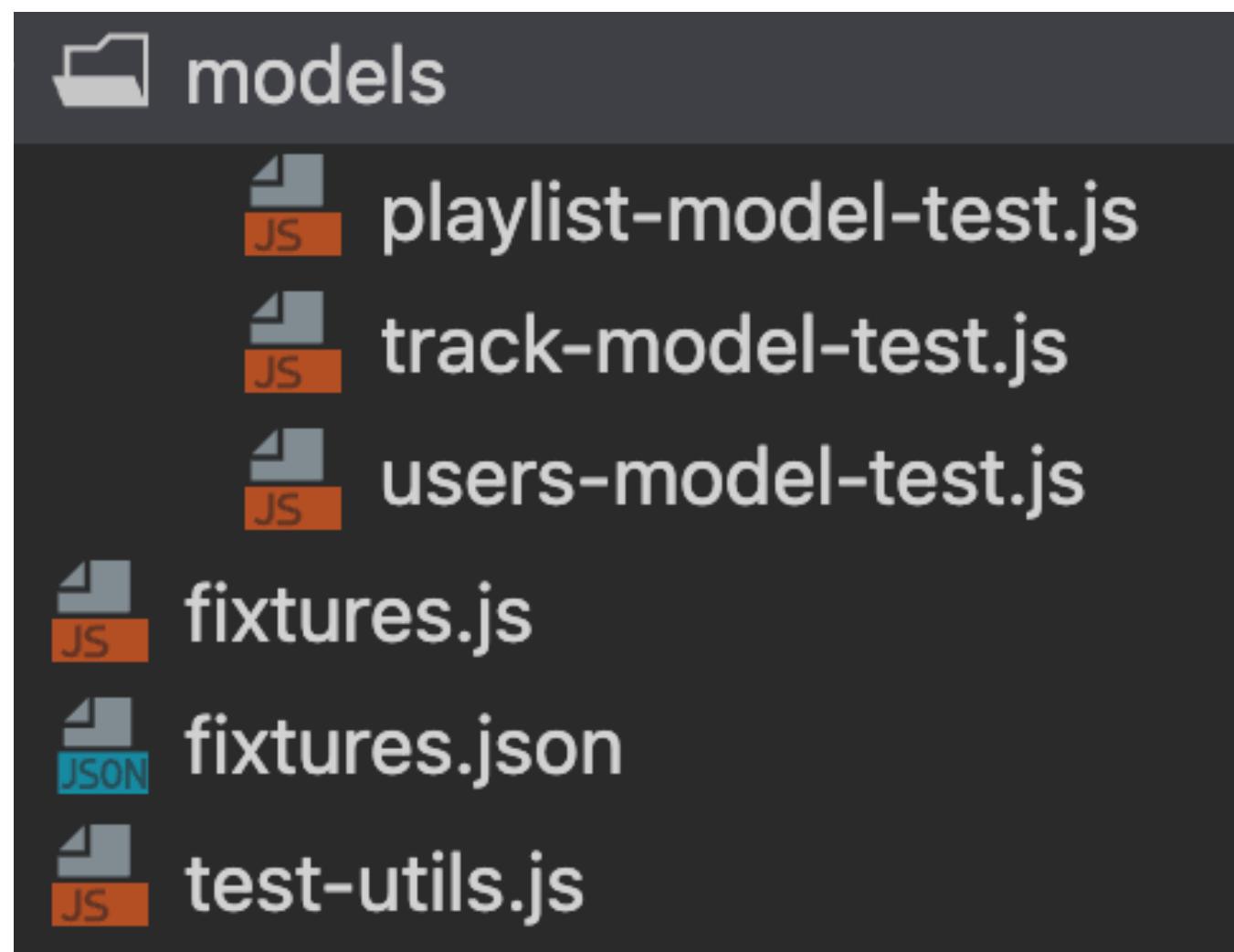
  { method: "GET", path: "/about", config: aboutController.index },

  { 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



✓ Playlist Model tests 341ms

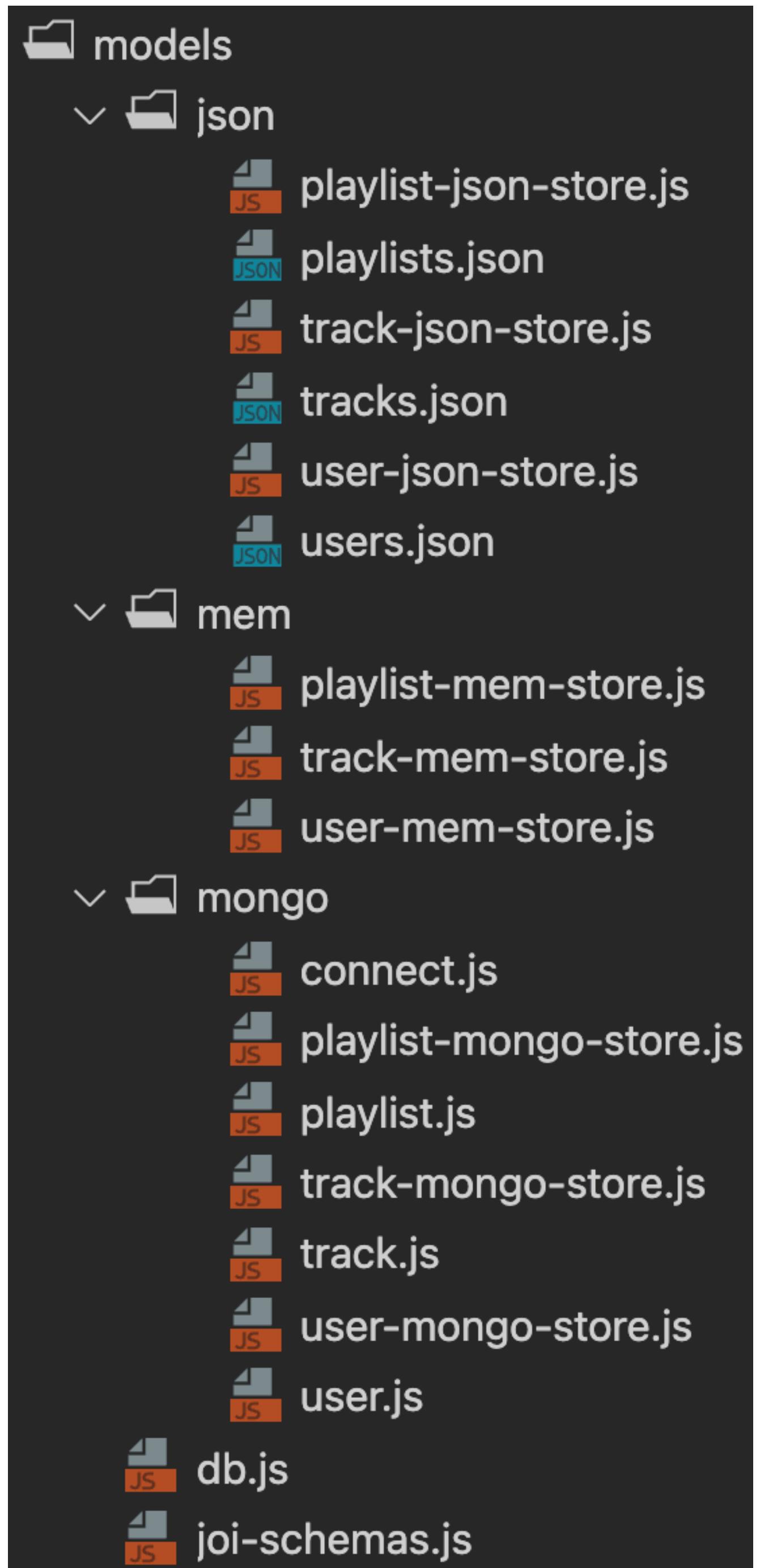
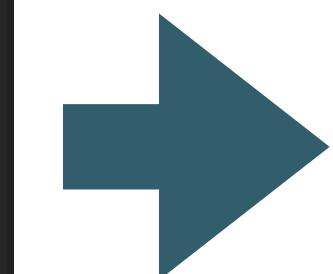
- ✓ create a playlist 126ms
- ✓ delete all playlists 54ms
- ✓ get a playlist - success 67ms
- ✓ delete One Playlist - success 56ms
- ✓ get a playlist - bad params 20ms
- ✓ delete One Playlist - fail 18ms

✓ Track Model tests 175ms

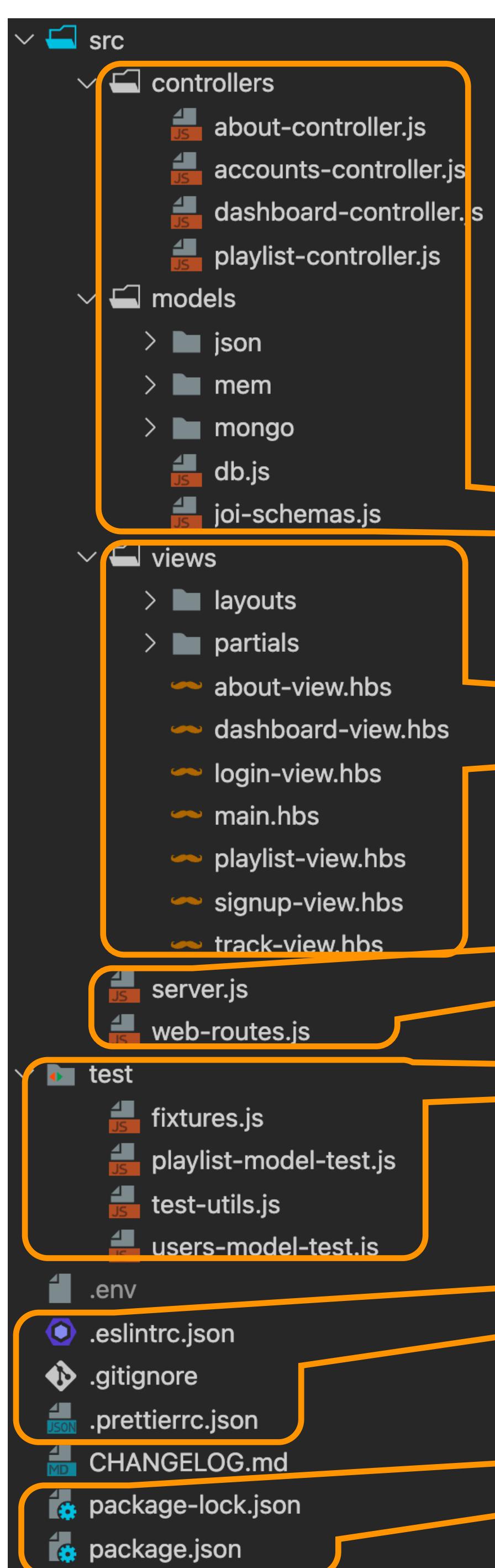
- ✓ create single track 27ms
- ✓ create multiple tracks 24ms
- ✓ delete all tracks 23ms
- ✓ get a track - success 21ms
- ✓ delete One Track - success 19ms
- ✓ get a playlist - bad params 30ms
- ✓ delete One User - fail 31ms

✓ User Model tests 114ms

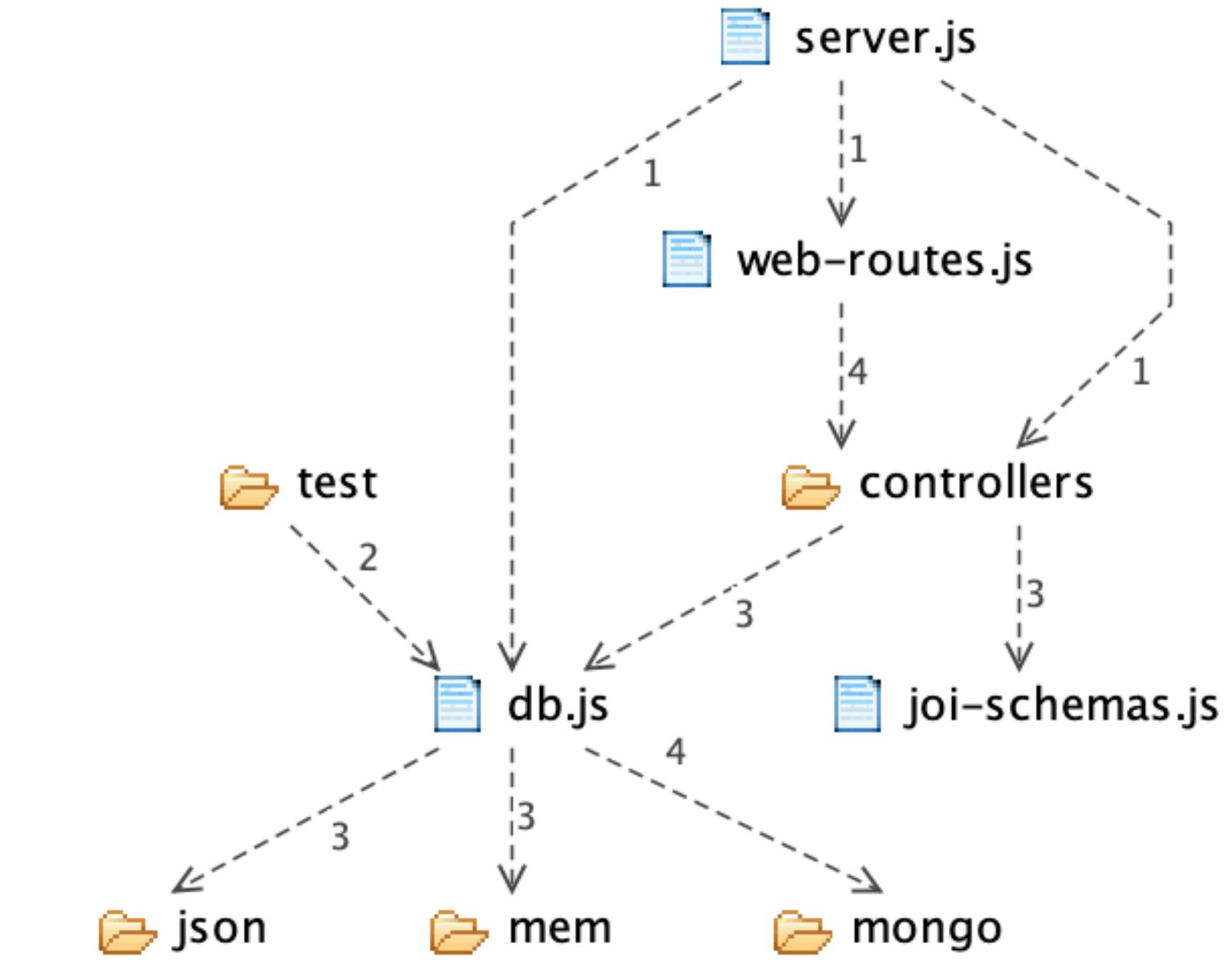
- ✓ create a user 27ms
- ✓ delete all users 19ms
- ✓ get a user - success 18ms
- ✓ delete One User - success 21ms
- ✓ get a user - bad params 15ms
- ✓ delete One User - fail 12ms

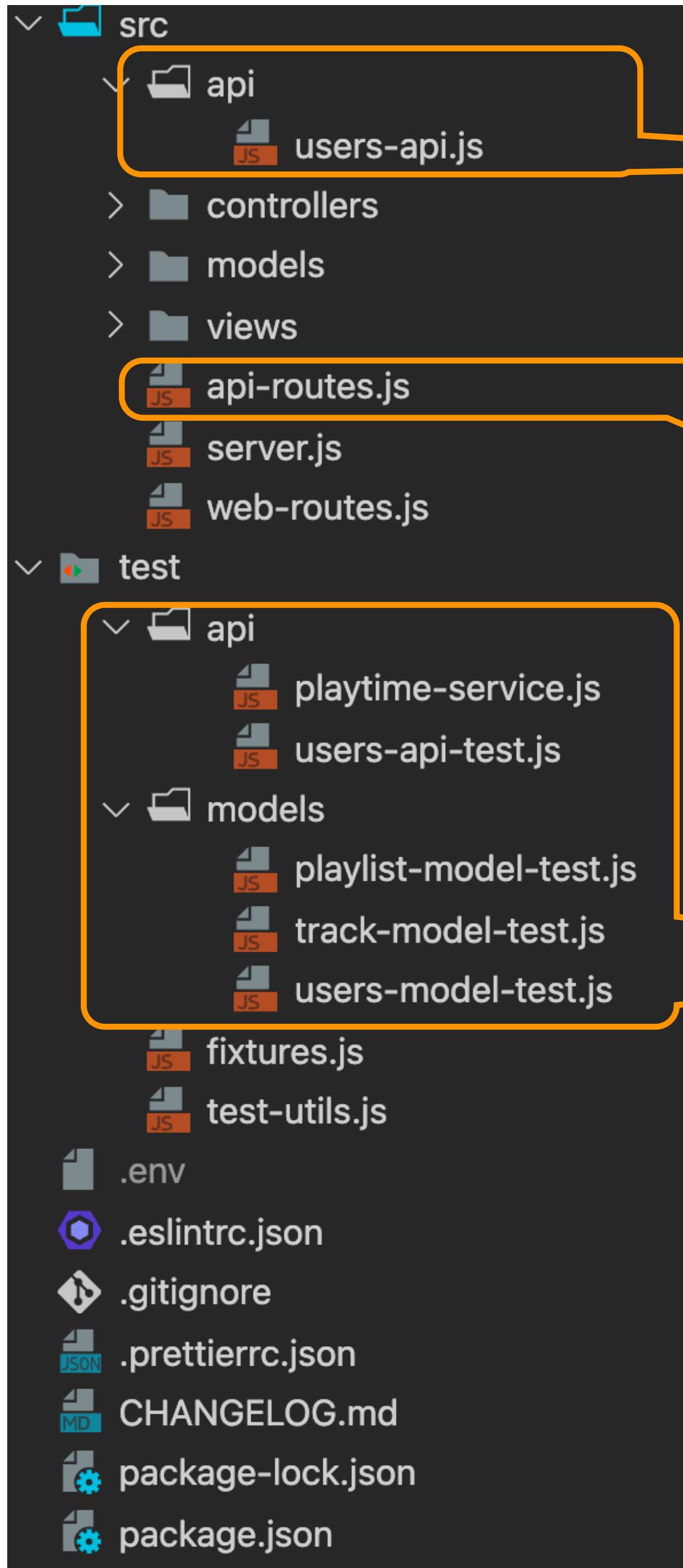


Project Structure



- 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

server.js

```
import { apiRoutes } from "./api-routes.js";  
  
...  
db.init("mongo");  
server.route(webRoutes);  
server.route(apiRoutes);  
await server.start();  
...
```

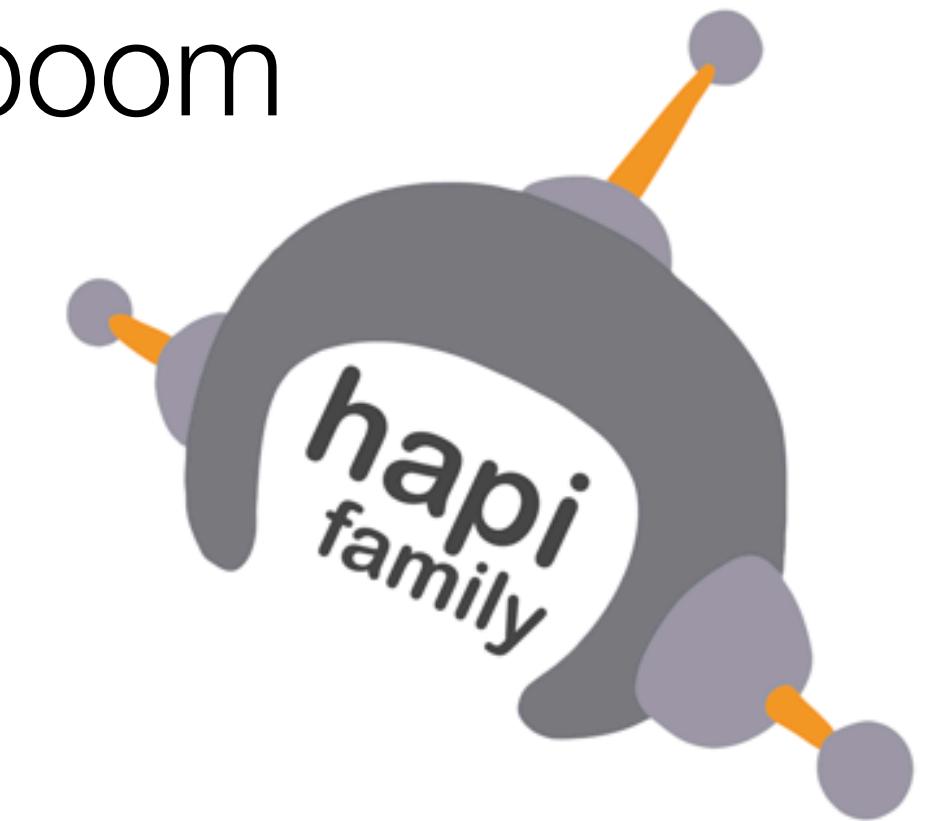
api-routes.js

Get Users Endpoint

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

@hapi/boom

<https://github.com/hapijs/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.

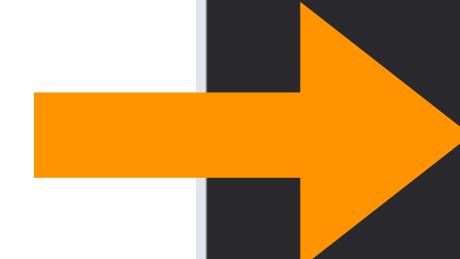
Visit the [hapi.dev Developer Portal](#) for tutorials, documentation, and support

🔗 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

```
// 20220131091122
// http://localhost:3000/api/users

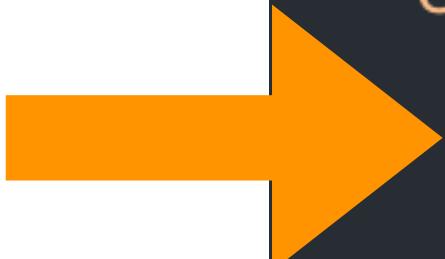
[
  {
    "_id": "61f4097496857dd77e19917e",
    "firstName": "Bart",
    "lastName": "Simpson",
    "email": "bart@simpson.com",
    "password": "secret",
    "__v": 0
  },
  {
    "_id": "61f40974f34d08eef165502b",
    "firstName": "Marge",
    "lastName": "Simpson",
    "email": "marge@simpson.com",
    "password": "secret",
    "__v": 0
  },
  {
    "_id": "61f40974f34d08eef165502e",
    "firstName": "Bart",
    "lastName": "Simpson",
    "email": "bart@simpson.com",
    "password": "secret",
    "__v": 0
  }
]
```

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

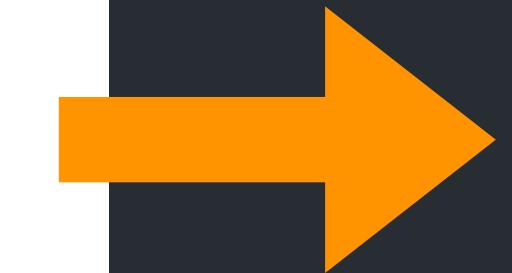
Exception creating user
object

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

The screenshot shows the Postman application interface. On the left, there's a sidebar with options: Home, Workspaces, API Network, Reports, Explore, My Workspace (selected), New, Import, Overview, Collections, APIs (highlighted in orange), Environments, Mock Servers, Monitors, Flows, and History. A central panel displays a message: "No APIs yet" with a small character icon holding a book. Below it, a note says: "APIs define related collections and environments under a consistent schema." A "Create an API" button is visible. The main workspace shows a request card for "GET http://localhost:3000/api/users". The "Params" tab is selected, showing a table with one row: "Key" and "Value". The "Body" tab is selected, showing a JSON response with three user objects:

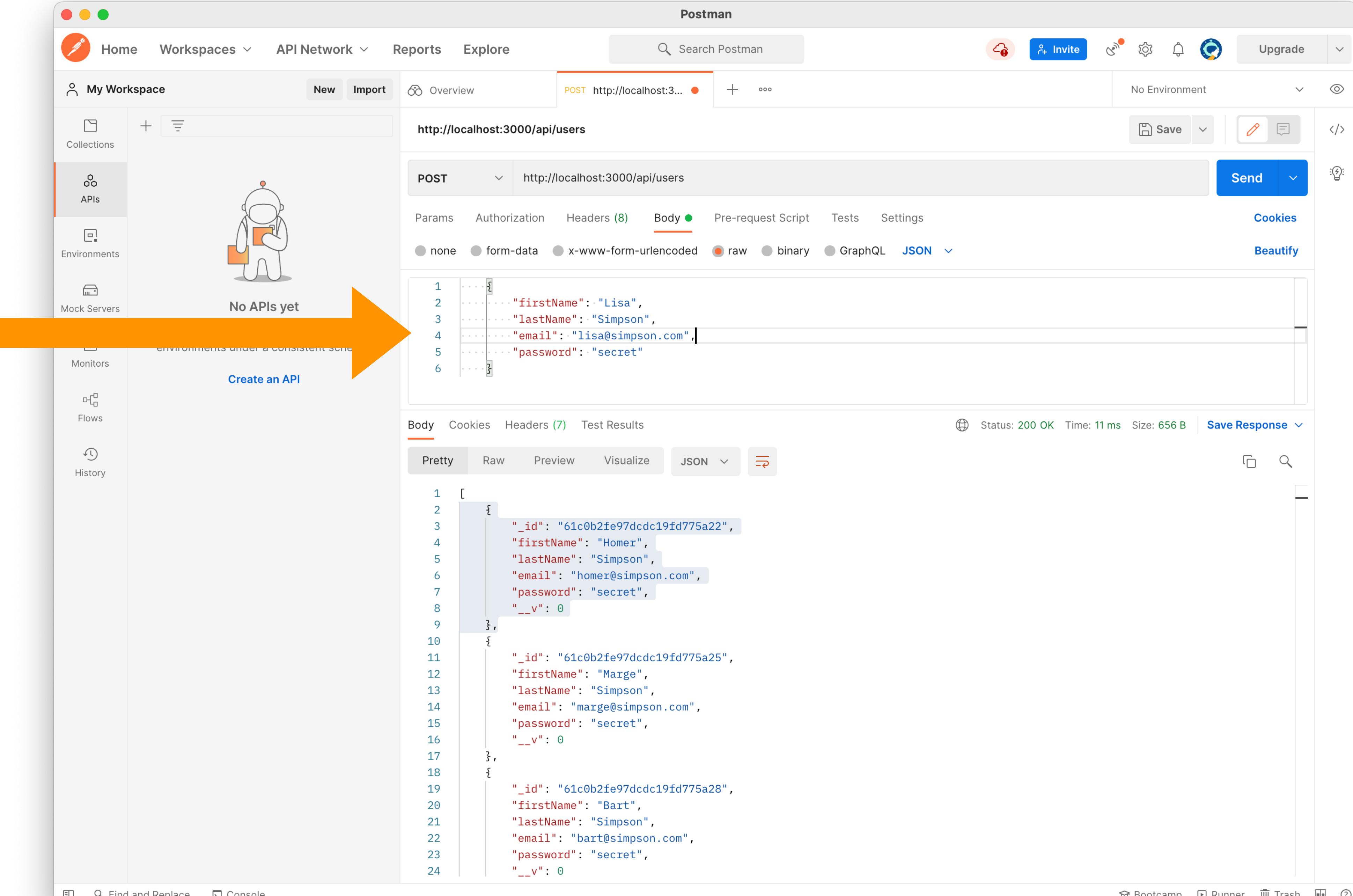
```

1  {
2   "id": "61c0b2fe97dc19fd775a22",
3   "firstName": "Homer",
4   "lastName": "Simpson",
5   "email": "homer@simpson.com",
6   "password": "secret",
7   "__v": 0
8 },
9 {
10  "id": "61c0b2fe97dc19fd775a25",
11  "firstName": "Marge",
12  "lastName": "Simpson",
13  "email": "marge@simpson.com",
14  "password": "secret",
15  "__v": 0
16 },
17 {
18  "id": "61c0b2fe97dc19fd775a28",
19  "firstName": "Bart",
20  "lastName": "Simpson",
21  "email": "bart@simpson.com",
22  "password": "secret",
23  "__v": 0
24 }
25
26

```

The status bar at the bottom indicates: Status: 200 OK, Time: 23 ms, Size: 613 B, Save Response.

- POST Request
- New User Details



The screenshot shows the Postman application interface. On the left, there's a sidebar with options like Home, Workspaces, API Network, Reports, Explore, My Workspace, Collections, APIs (which is selected), Environments, Mock Servers, Monitors, Flows, and History. The main workspace has a central area with a small robot icon and the text "No APIs yet". Below it, there's a button "Create an API". A large orange arrow points from the left edge of the slide towards this central area.

The main workspace shows a POST request to `http://localhost:3000/api/users`. The "Body" tab is selected, showing a JSON payload:

```

1  {
2   ...
3   "firstName": "Lisa",
4   "lastName": "Simpson",
5   "email": "lisa@simpson.com",
6   "password": "secret"
7 }

```

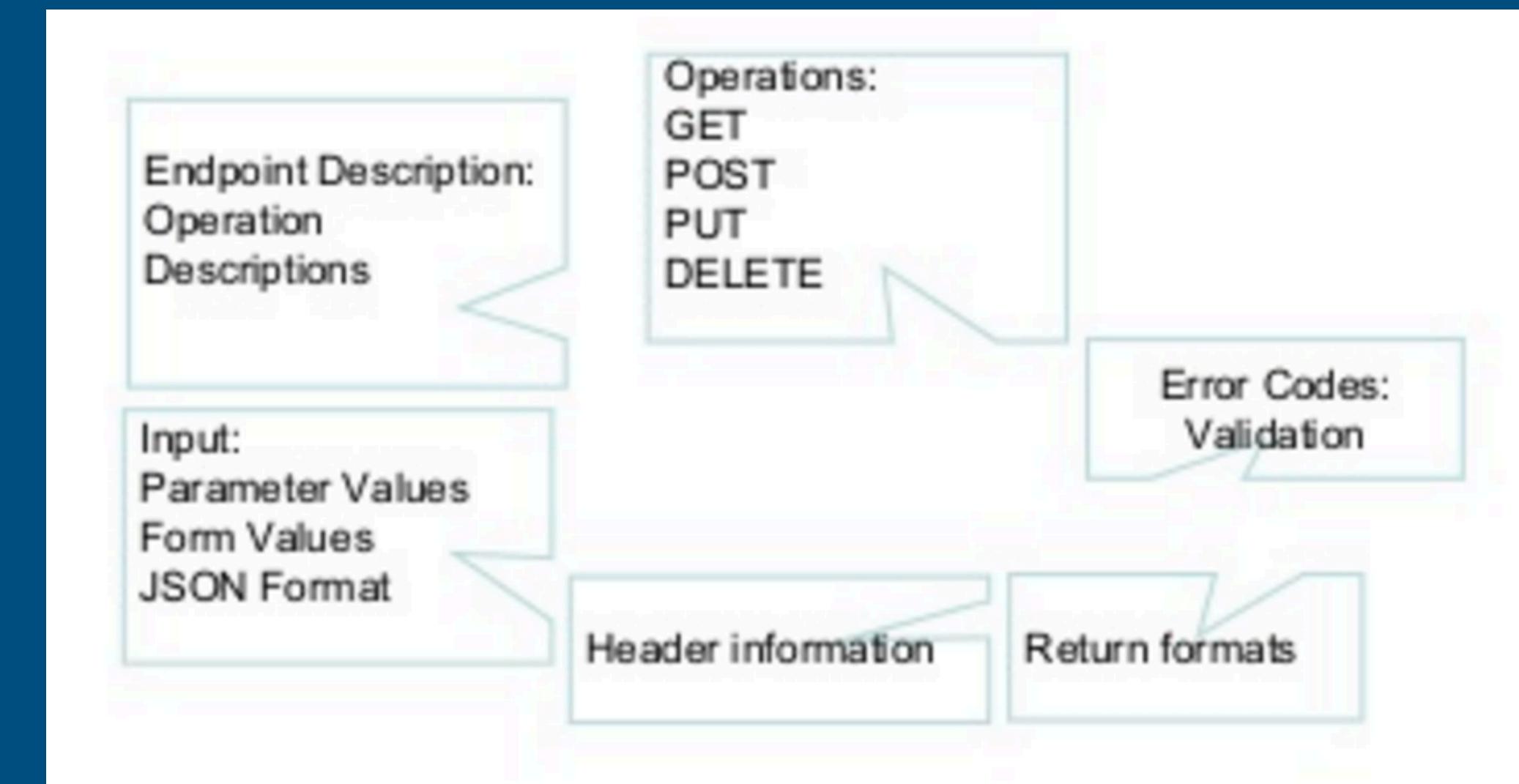
Below the request, the response status is shown as 200 OK with a time of 11 ms and a size of 656 B. The response body is also displayed in JSON format:

```

1 [
2   {
3     "_id": "61c0b2fe97dc19fd775a22",
4     "firstName": "Homer",
5     "lastName": "Simpson",
6     "email": "homer@simpson.com",
7     "password": "secret",
8     "__v": 0
9   },
10  {
11    "_id": "61c0b2fe97dc19fd775a25",
12    "firstName": "Marge",
13    "lastName": "Simpson",
14    "email": "marge@simpson.com",
15    "password": "secret",
16    "__v": 0
17  },
18  {
19    "_id": "61c0b2fe97dc19fd775a28",
20    "firstName": "Bart",
21    "lastName": "Simpson",
22    "email": "bart@simpson.com",
23    "password": "secret",
24    "__v": 0
25  }
]

```

End Points



Full Stack Web Development