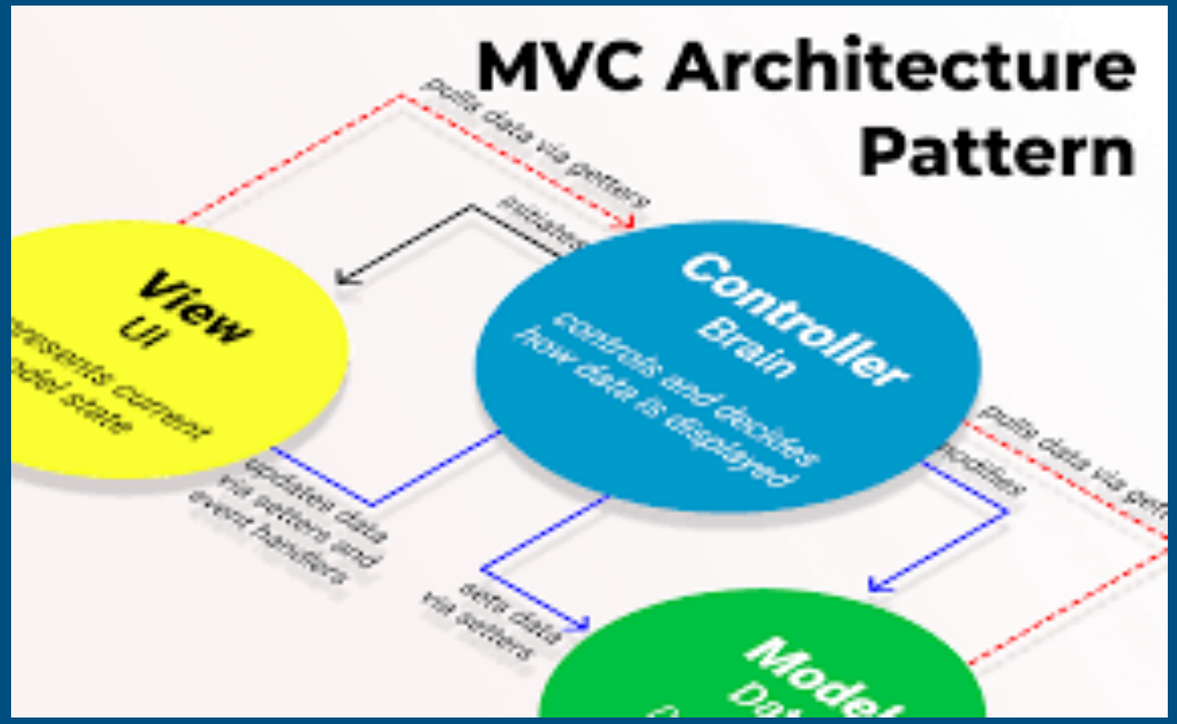
MVC in HAPI



Full Stack Web Development

Hapi Application Patterns

Models

Views

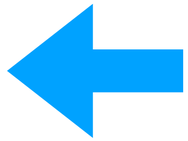
Controllers

Hapi Application

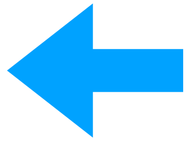
Minimal Hapi Application

src/server.js

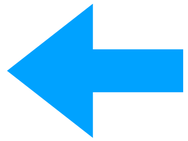
```
import Hapi from "@hapi/hapi";
import path from "path";

import { fileURLToPath } from "url";

const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);

async function init() {
  const server = Hapi.server({
    port: 3000,
    host: "localhost",
  });
  await server.start();
  console.log("Server running on %s", server.info.uri);
}

process.on("unhandledRejection", (err) => {
  console.log(err);
  process.exit(1);
});

init();
```



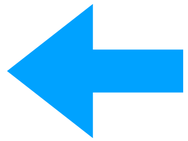
Import Hapi



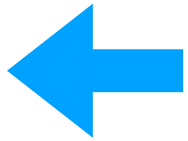
Import utilities to locate folder paths



Initialise a server



Handle failure to start



Start the app

Application Scripts

2 scripts:
“start”
“lint”

package.json

```
{
  "name": "playtime",
  "version": "0.1.0",
  "description": "Playtime: a Hapi/node application for managing Play]
  "main": "src/server.js",
  "type": "module",
  "scripts": {
    "start": "node src/server.js",
    "lint": "./node_modules/.bin/eslint . --ext .js"
  },
```

```
npm run start
```

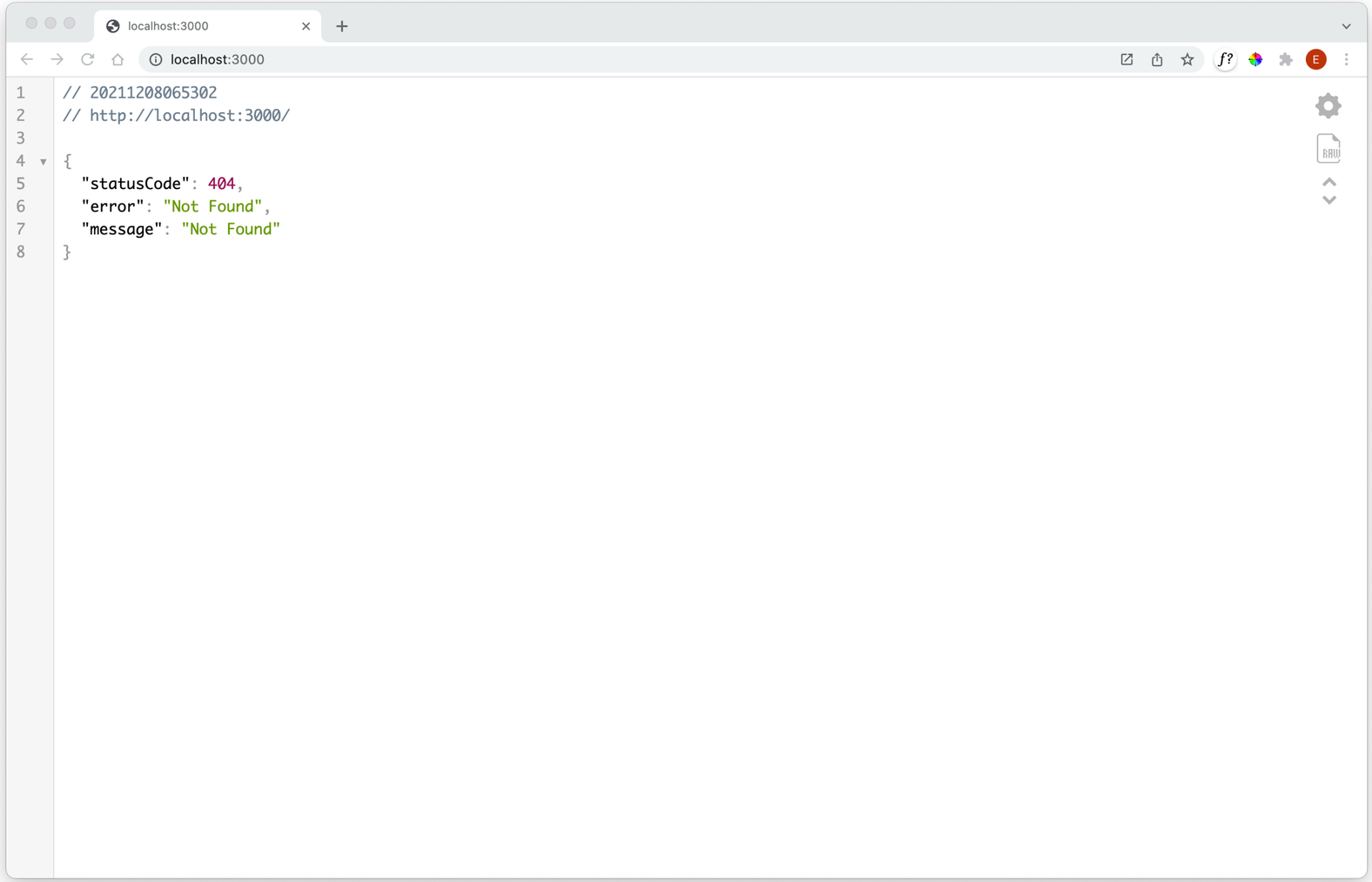
- Can also do ‘npm run lint’…

Running the application

- Start the application …

```
npm run start


> playtime@0.1.0 start
> node src/server.js

Server running on http://localhost:3000
```

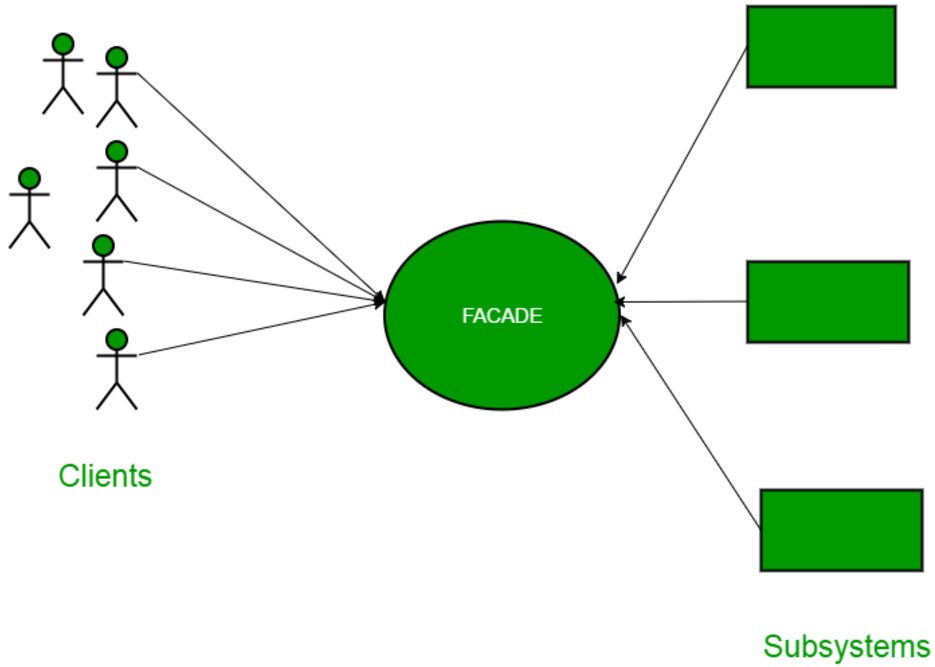
- …no routes defined yet



Patterns

Facade Pattern

- The facade pattern (also spelled façade) is a software-design pattern commonly used in object-oriented programming.
- Analogous to a facade in architecture, a facade is an object that serves as a front-facing interface masking more complex underlying or structural code



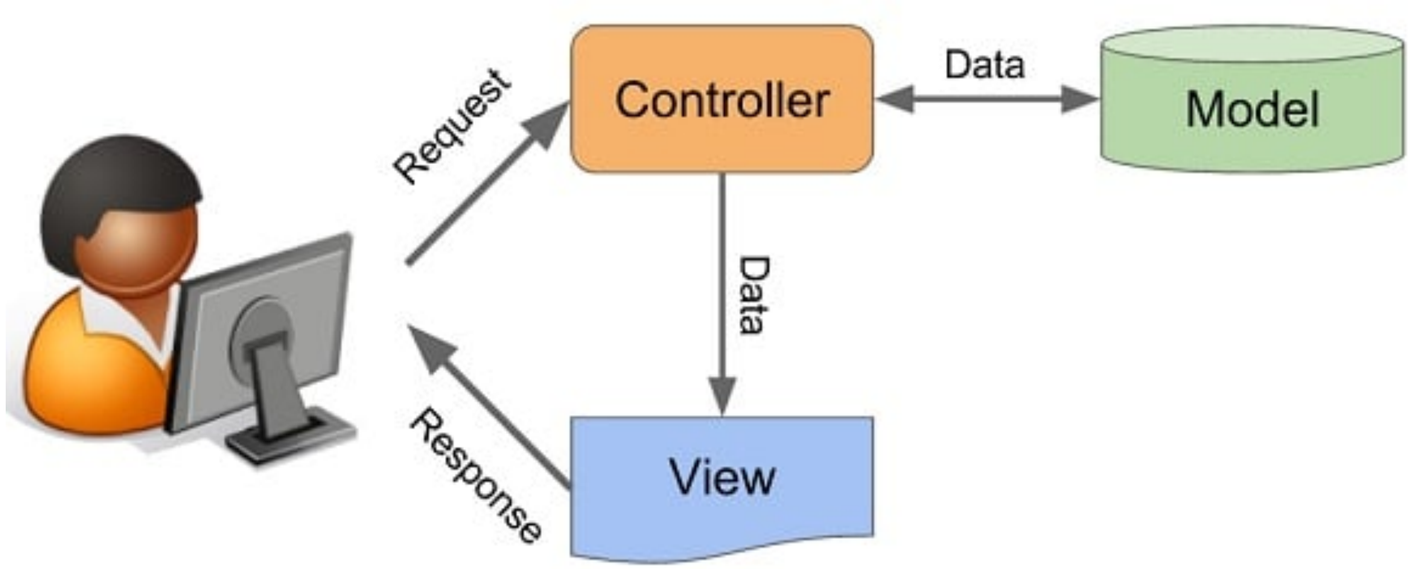
- improve the readability and usability of a software library by masking interaction with more complex components behind a single API
- provide a context-specific interface to more generic functionality
- serve as a launching point for a broader refactor of monolithic or tightly-coupled systems in favour of more loosely-coupled code

https://en.wikipedia.org/wiki/Facade_pattern

MVC Pattern

- Model–view–controller (MVC) is a software design pattern commonly used for developing user interfaces that divide the related program logic into three interconnected elements.

- This is done to separate internal representations of information from the ways information is presented to and accepted from the user.



- Model: The central component of the pattern. It is the application's dynamic data structure, independent of the user interface.[10] It directly manages the data, logic and rules of the application.

- View: Any representation of information such as a chart, diagram or table. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.

- Controller: Accepts input and converts it to commands for the model or view

https://en.wikipedia.org/wiki/Model–view–controller

Models

user-mem-store.js

- **user-mem-store** object maintains and manages a store of user objects
- These users are maintained in a simple in-memory array
- Provides general access methods, largely relying on each user having a unique id

user-mem-store.js

```
import { v4 } from "uuid";

let users = [];

export const userMemStore = {
  async getAllUsers() {
    return users;
  },

  async addUser(user) {
    user._id = v4();
    users.push(user);
    return user;
  },

  async getUserById(id) {
    return users.find((user) => user._id === id);
  },

  async getUserByEmail(email) {
    return users.find((user) => user.email === email);
  },

  async deleteUserById(id) {
    const index = users.findIndex((user) => user._id === id);
    users.splice(index, 1);
  },

  async deleteAll() {
    users = [];
  },
};
```

Encapsulate Database Access

- The **db** object acts as a simplified Facade to the database
- Clients (controllers and other components) access the database via this object.
- Specifically, user and play store databases access via **db**

db.js

```
import { userMemStore } from "./mem/user-mem-store.js";
import { playlistMemStore } from "./mem/playlist-mem-store.js";

export const db = {
  userStore: null,
  playlistStore: null,

  init() {
    this.userStore = userMemStore;
    this.playlistStore = playlistMemStore;
  },
};
```

playlist-mem-store.js

- **playlist-mem-store** object maintains and manages a store of playlists objects
- Playlists are maintained in a simple in-memory array
- Provides general access methods, largely relying on each user having a unique id

playlist-mem-store.js

```
import { v4 } from "uuid";

let playlists = [];

export const playlistMemStore = {
  async getAllPlaylists() {
    return playlists;
  },

  async addPlaylist(playlist) {
    playlist._id = v4();
    playlists.push(playlist);
    return playlist;
  },

  async getPlaylistById(id) {
    return playlists.find((playlist) => playlist._id === id);
  },

  async deletePlaylistById(id) {
    const index = playlists.findIndex((playlist) => playlist._id === id);
    playlists.splice(index, 1);
  },

  async deleteAllPlaylists() {
    playlists = [];
  },
};
```

UUIDs

- A universally unique identifier (UUID) is a 128-bit label used for information in computer systems. The term globally unique identifier (GUID) is also used

- When generated according to the standard methods, UUIDs are, for practical purposes, unique.

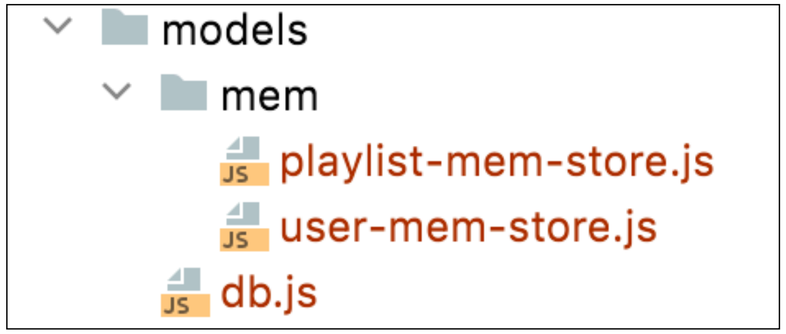
- Their uniqueness does not depend on a central registration authority or coordination between the parties generating them, unlike most other numbering schemes. While the probability that a UUID will be duplicated is not zero, it is close enough to zero to be negligible.

Table of Contents

1. Introduction 2
2. Motivation 3
3. Namespace Registration Template 3
4. Specification 5
4.1. Format. 5
4.1.1. Variant. 6
4.1.2. Layout and Byte Order. 6
4.1.3. Version. 7
4.1.4. Timestamp. 8
4.1.5. Clock Sequence 8
4.1.6. Node 9
4.1.7. Nil UUID 9
4.2. Algorithms for Creating a Time-Based UUID 9
4.2.1. Basic Algorithm. 10
4.2.2. Generation Details 12
4.3. Algorithm for Creating a Name-Based UUID. 13
4.4. Algorithms for Creating a UUID from Truly Random or Pseudo-Random Numbers 14
4.5. Node IDs that Do Not Identify the Host. 15
5. Community Considerations 15
6. Security Considerations 16
7. Acknowledgments 16
8. Normative References 16
A. Appendix A - Sample Implementation 18
B. Appendix B - Sample Output of utest 29
C. Appendix C - Some Name Space IDs 30

https://www.ietf.org/rfc/rfc4122.txt

```
async addUser(user) {
  user._id = v4();
  users.push(user);
  return user;
},
```



npm Search packages Search

uuid DT

8.3.2 • Public • Published a year ago

Readme | Explore BETA | 0 Dependencies | 42,102 Dependents | 34 Versions

uuid

CI passing | Browser passing

For the creation of RFC4122 UUIDs

- **Complete** - Support for RFC4122 version 1, 3, 4, and 5 UUIDs
- **Cross-platform** - Support for ...
  - CommonJS, ECMAScript Modules and CDN builds
  - Node 8, 10, 12, 14
  - Chrome, Safari, Firefox, Edge, IE 11 browsers
  - Webpack and rollup.js module bundlers
  - React Native / Expo
- **Secure** - Cryptographically-strong random values
- **Small** - Zero-dependency, small footprint, plays nice with "tree shaking" packagers
- **CLI** - Includes the `uuid` command line utility

Upgrading from `uuid@3.x`? Your code is probably okay, but check out Upgrading From `uuid@3.x` for details.

Install

```
> npm i uuid
```

Repository
github.com/uuidjs/uuid

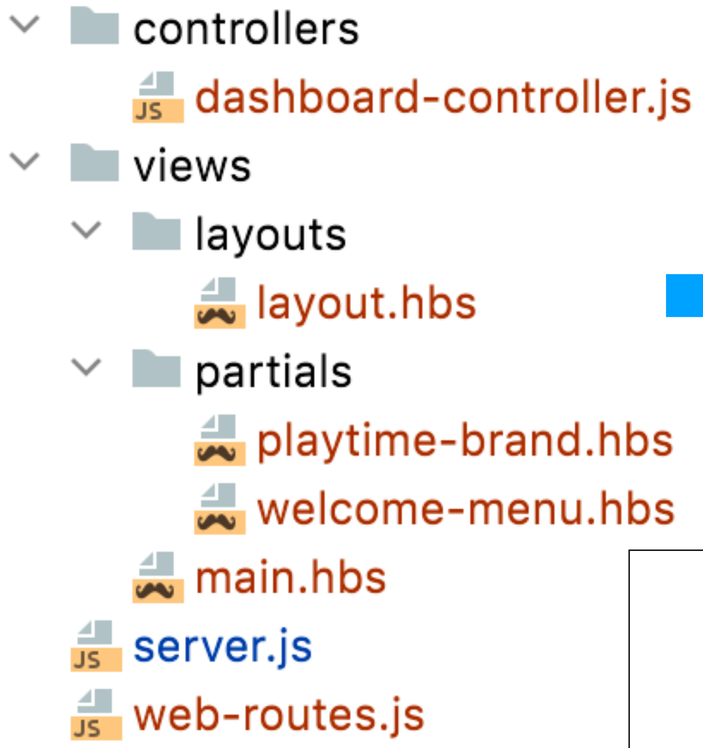
Homepage
github.com/uuidjs/uuid#readme

Weekly Downloads
49,792,478

Version
8.3.2

License
MIT

Views



layout.hbs

```
<!DOCTYPE html>
<html lang="en-IE">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>{{title}}</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bulma@0.9.3/css/bulma
    <script src="https://use.  tawesome.com/releases/v5.15.4/js/all.js"></script>
  </head>
  <body>
    <div class="container">
      {{{content}}}
    </div>
  </body>
</html>
```

Bulma: the **modern** CSS framework that just **works**.

Bulma is a free, open source framework that provides ready-to-use frontend components that you can easily combine to build responsive web interfaces.

✅ No CSS knowledge required.

Font Awesome

Get vector icons and social logos on your website with Font Awesome, the web's most popular icon set and toolkit.

Box
A white **box** to contain other elements

Button
The classic **button**, in different colors, sizes, and states

Content
A single class to handle **WYSIWYG** generated content, where only **HTML tags** are available

Delete
A versatile **delete** cross

Icon
Compatible with all icon font libraries, including **Font Awesome 5**

Image
A container for **responsive images**

Notification
Bold **notification** blocks, to alert your users of something

Progress bars
Native HTML **progress** bars

Table
The inevitable HTML **table**, with special case cells

Tag
Small **tag labels** to insert anywhere

Title
Simple **headings** to add depth to your page

Breadcrumb
A simple **breadcrumb** component to improve your navigation experience

Card
An all-around flexible and composable component

Dropdown
An interactive **dropdown menu** for discoverable content

Menu
A simple **menu**, for any type of vertical navigation

Message
Colored **message** blocks, to emphasize part of your page

Modal
A classic **modal** overlay, in which you can include *any* content you want

Navbar
A responsive horizontal **navbar** that can support images, links, buttons, a dropdowns

Pagination
A responsive, usable, and flexible **pagination**

Panel
A composable **panel**, for compact controls

Tabs
Simple responsive horizontal navigation **tabs**, with different styles

BULMA

Container
A simple **container** to center your content horizontally

Level
A multi-purpose **horizontal level**, which can contain almost any other element

Media Object
The famous **media object** prevalent in social media interfaces, but useful in any context

Hero
An imposing **hero banner** to showcase something

Section
A simple container to divide your page into **sections**, like the one you're currently reading

Footer
A simple responsive **footer** which can include anything: lists, headings, columns, icons, buttons, etc.

Tiles
A **single tile** element to build 2-dimensional Metro-like, Pinterest-like, or whatever-you-like grids

Font Awesome

```
<div class="title p-3 is-flex">
  <i style="font-size: 48px; color: Dodgerblue;" class="fas fa-icons"></i>
  <div class="ml-4">Playtime</div>
</div>
```

Playtime

music

by algolia

27 matching results

"music" Free

music volume-mute play drum play-circle play-circle

volume-off volume-down headphones-alt headphones guitar file-audio

file-audio drum-steelpan podcast volume-up record-vinyl icons

compact-disc spotify soundcloud sliders-h napster microphone-slash

microphone-alt-slash microphone-alt microphone

icons

Solid Style (fas) • f86d • •

Copy HTML

Editors • Updated: Version 5.9.0

playtime-brand.hbs

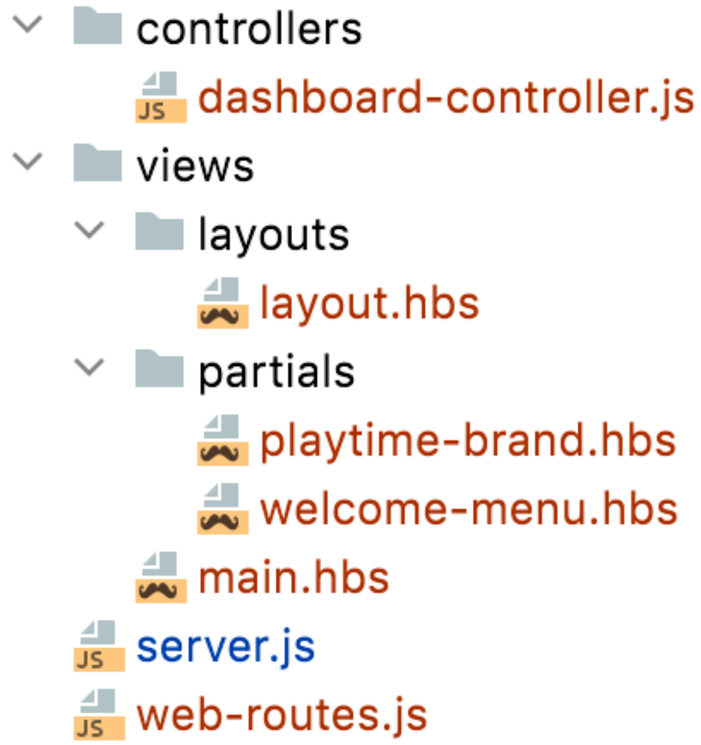
```
<div class="title p-3 is-flex">
  <i style="font-size: 48px; color: Dodgerblue;" class="fas fa-icons"></i>
  <div class="ml-4">Playtime</div>
</div>
```

Playtime

welcome-menu.hbs

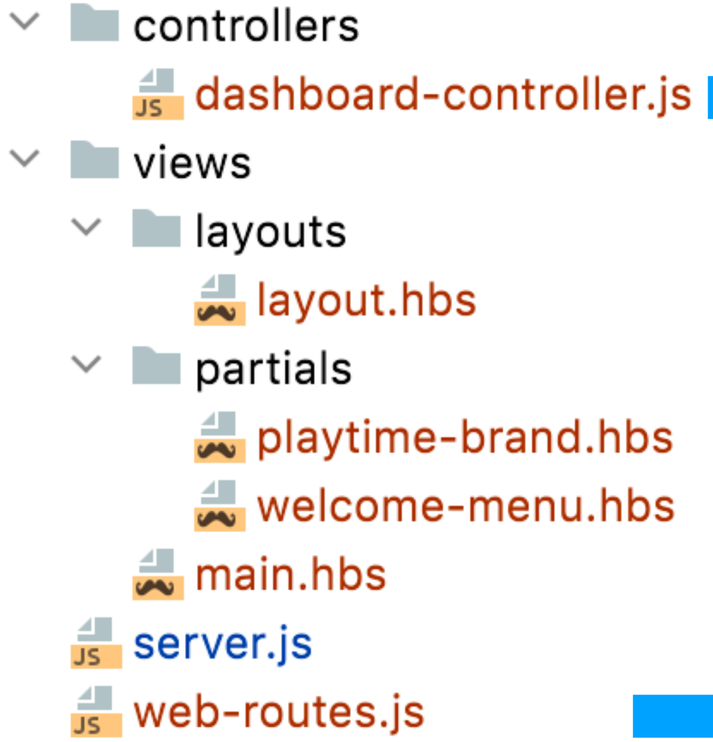
```
<nav class="navbar" role="navigation" aria-label="main navigation">
  <div class="navbar-brand">
    {{> playtime-brand}}
  </div>
  <div id="navbarBasicExample" class="navbar-menu">
    <div class="navbar-end">
      <div class="navbar-item">
        <div class="buttons">
          <a class="button" id="login" href="/login"> Log in </a> <a class="button"
          up </a>
        </div>
      </div>
    </div>
  </div>
</nav>

<script>
  document.getElementById("{{active}}").classList.add("is-primary");
</script>
```

Log in Sign up



main.hbs

```
{{> welcome-menu}}

<section class="section">
  <p> Sign up or Log in... </p>
</section>
```

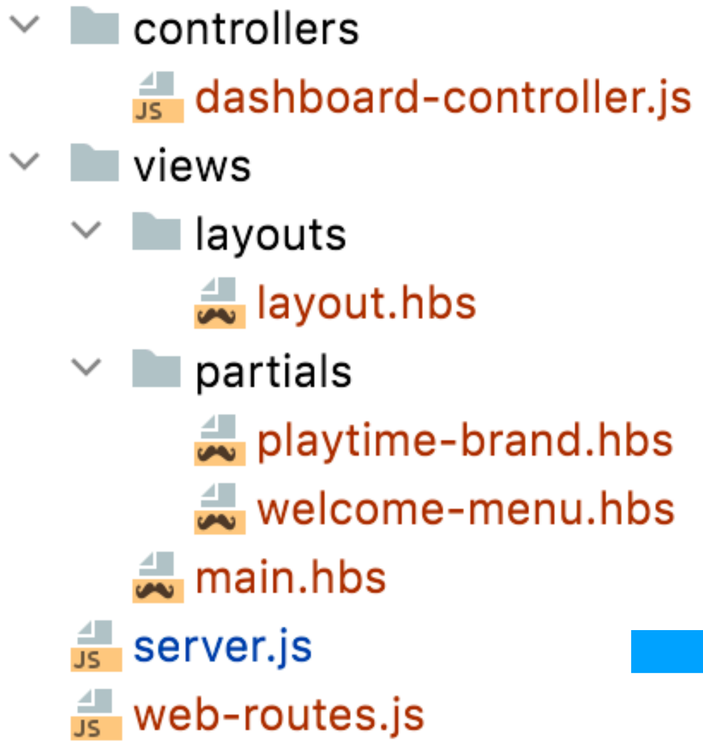


dashboard-controller.js

```
export const dashboardController = {
  index: {
    handler: async function (request, h) {
      return h.view("main");
    },
  },
};
```

web-routes.js

```
import { dashboardController } from "./controllers/dashboard-controller.js";

export const webRoutes = [{ method: "GET", path: "/", config: dashboardController.i
```



server.js

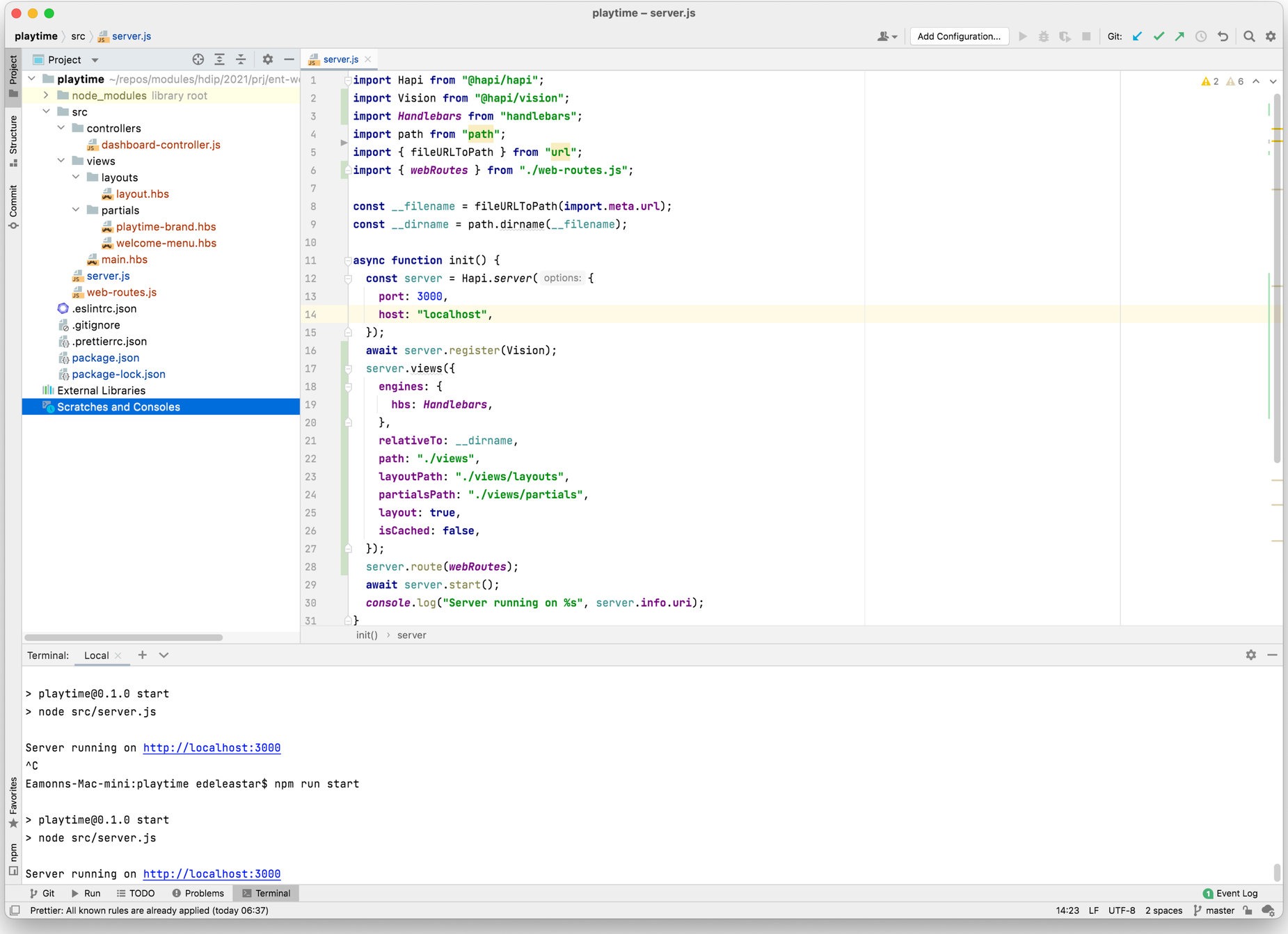
```
import Hapi from "@hapi/hapi";
import Vision from "@hapi/vision";
import Handlebars from "handlebars";
import path from "path";
import { fileURLToPath } from "url";
import { webRoutes } from "./web-routes.js";

const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);

async function init() {
  const server = Hapi.server({
    port: 3000,
    host: "localhost",
  });
  await server.register(Vision);
  server.views({
    engines: {
      hbs: Handlebars,
    },
    relativeTo: __dirname,
    path: "./views",
    layoutPath: "./views/layouts",
    partialsPath: "./views/partials",
    layout: true,
    isCached: false,
  });
  server.route(webRoutes);
  await server.start();
  console.log("Server running on %s", server.info.uri);
}

process.on("unhandledRejection", (err) => {
  console.log(err);
  process.exit(1);
});

init();
```

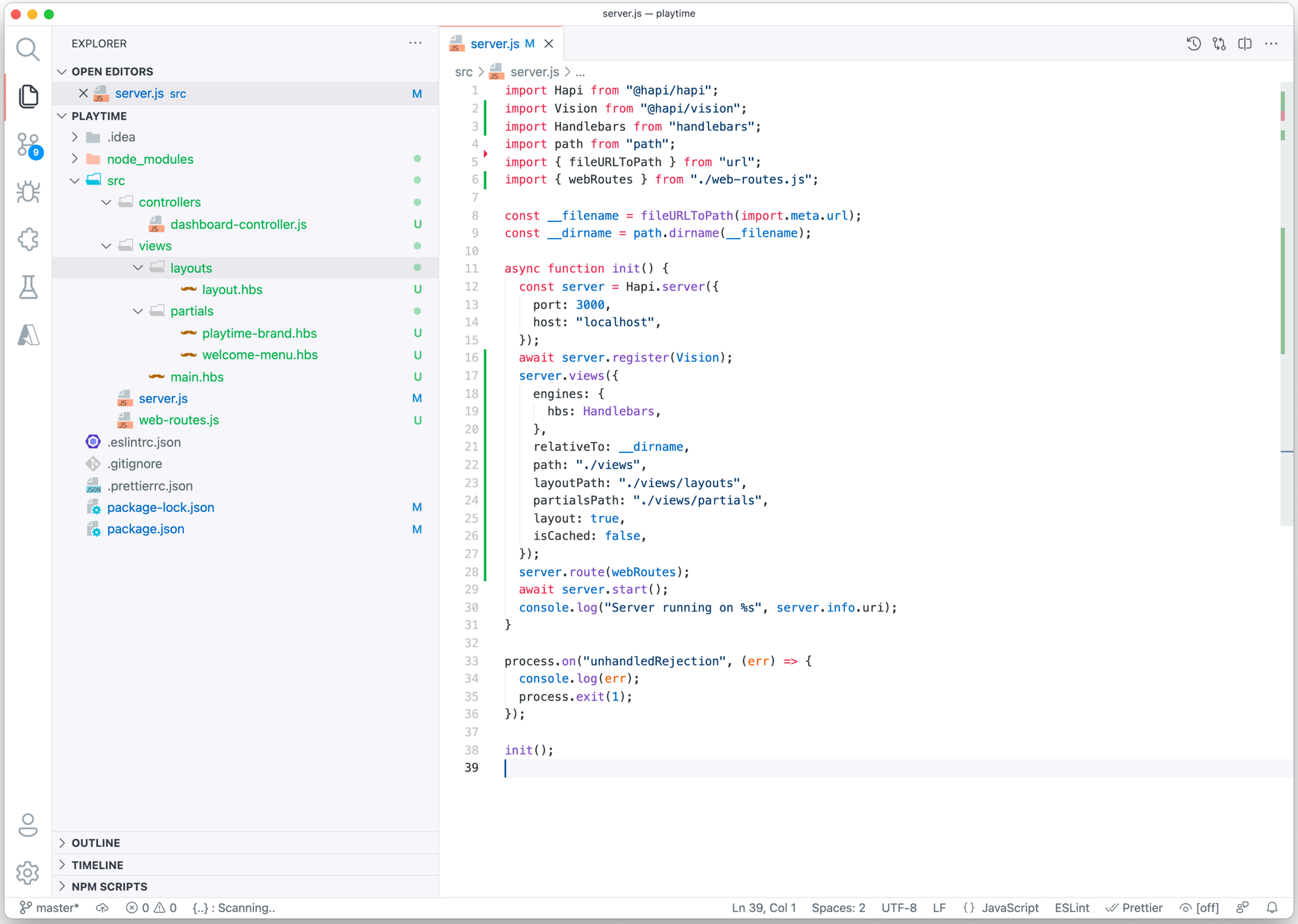
Register plugins

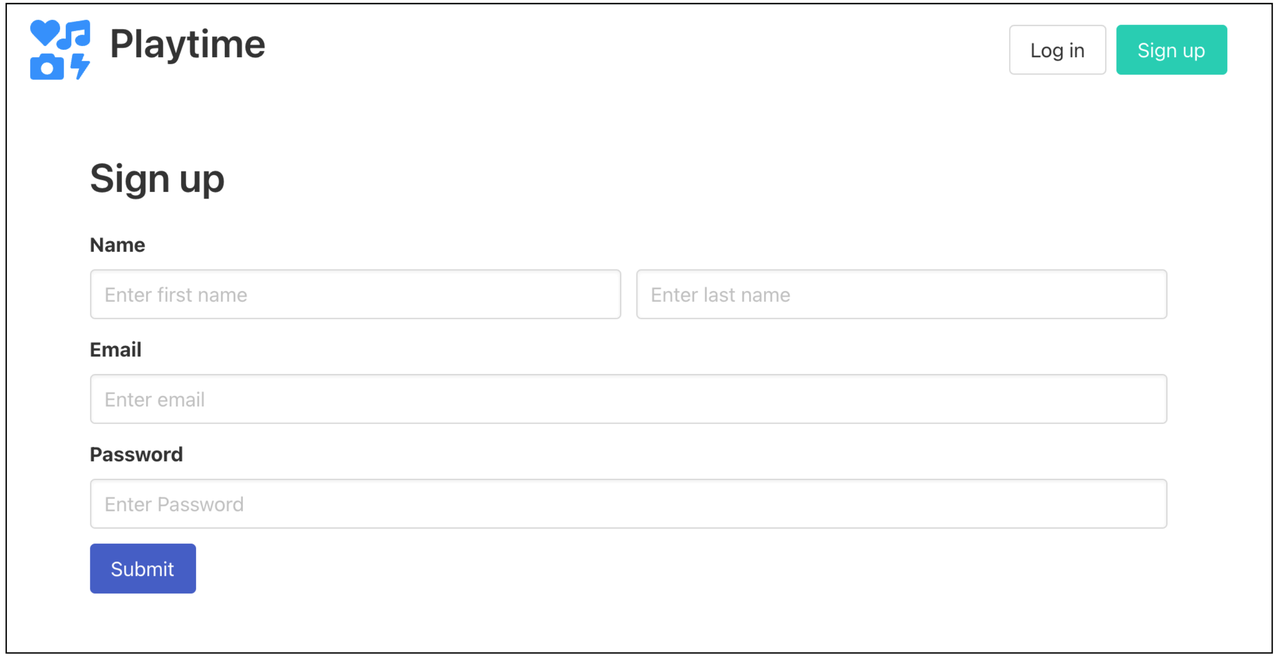
Initialise to use Handlebars engine

Define template locations and cache settings

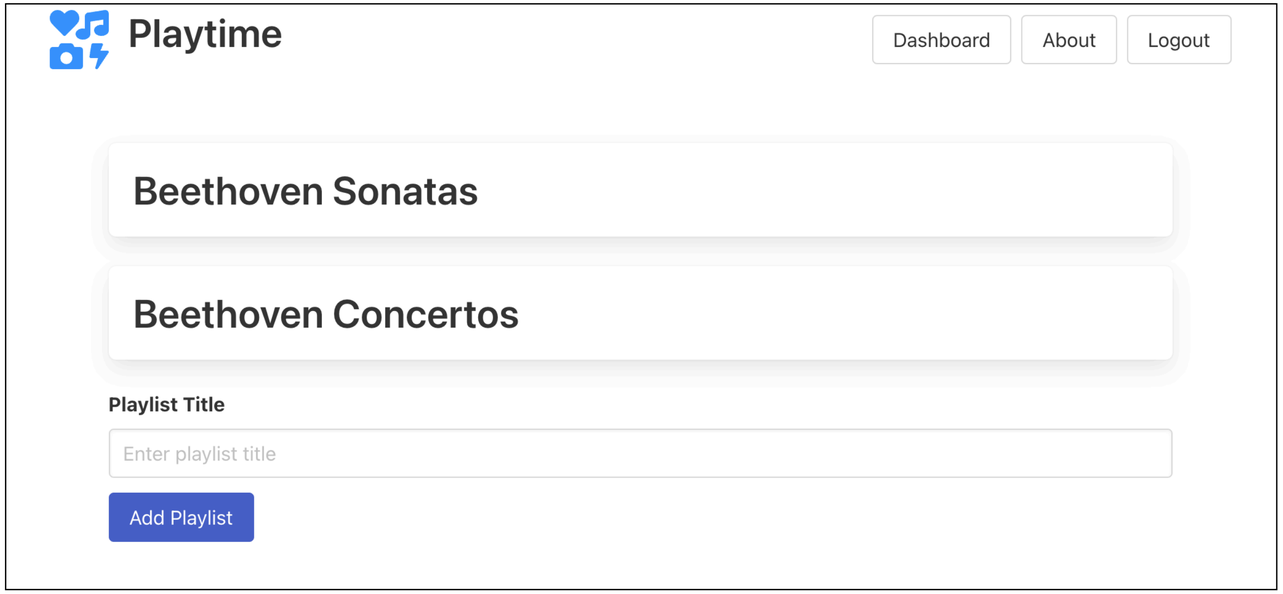
```
async function init() {
  const server = Hapi.server({
    port: 3000,
    host: "localhost",
  });
  await server.register(Vision);
  server.views({
    engines: {
      hbs: Handlebars,
    },
    relativeTo: __dirname,
    path: "./views",
    layoutPath: "./views/layouts",
    partialsPath: "./views/partials",
    layout: true,
    isCached: false,
  });
  server.route(webRoutes);
  await server.start();
  console.log("Server running on %s", server.info.uri);
}
```


playtime – server.js
playtime src server.js
Add Configuration...
Git:
Project
playtime ~/repos/modules/hdip/2021/prj/ent-w
node_modules library root
src
controllers
dashboard-controller.js
views
layouts
layout.hbs
partials
playtime-brand.hbs
welcome-menu.hbs
main.hbs
server.js
web-routes.js
.eslintrc.json
.gitignore
.prettierrc.json
package.json
package-lock.json
External Libraries
Scratches and Consoles
server.js
import Hapi from "@hapi/hapi";
import Vision from "@hapi/vision";
import Handlebars from "handlebars";
import path from "path";
import { fileURLToPath } from "url";
import { webRoutes } from "./web-routes.js";
const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
async function init() {
const server = Hapi.server(options: {
port: 3000,
host: "localhost",
});
await server.register(Vision);
server.views({
engines: {
hbs: Handlebars,
},
relativeTo: __dirname,
path: "./views",
layoutPath: "./views/layouts",
partialsPath: "./views/partials",
layout: true,
isCached: false,
});
server.route(webRoutes);
await server.start();
console.log("Server running on %s", server.info.uri);
}
init() server
Terminal: Local
> playtime@0.1.0 start
> node src/server.js
Server running on http://localhost:3000
^C
Eamonns-Mac-mini:playtime edeleastar$ npm run start
> playtime@0.1.0 start
> node src/server.js
Server running on http://localhost:3000
Git Run TODO Problems Terminal
Event Log
Prettier: All known rules are already applied (today 06:37)
14:23 LF UTF-8 2 spaces master


server.js — playtime
EXPLORER
OPEN EDITORS
server.js src M
PLAYTIME
.idea
node_modules
src
controllers
dashboard-controller.js U
views
layouts
layout.hbs U
partials
playtime-brand.hbs U
welcome-menu.hbs U
main.hbs U
server.js M
web-routes.js U
.eslintrc.json
.gitignore
.prettierrc.json
package-lock.json M
package.json M
OUTLINE
TIMELINE
NPM SCRIPTS
server.js M
src > server.js > ...
1 import Hapi from "@hapi/hapi";
2 import Vision from "@hapi/vision";
3 import Handlebars from "handlebars";
4 import path from "path";
5 import { fileURLToPath } from "url";
6 import { webRoutes } from "./web-routes.js";
7
8 const __filename = fileURLToPath(import.meta.url);
9 const __dirname = path.dirname(__filename);
10
11 async function init() {
12 const server = Hapi.server({
13 port: 3000,
14 host: "localhost",
15 });
16 await server.register(Vision);
17 server.views({
18 engines: {
19 hbs: Handlebars,
20 },
21 relativeTo: __dirname,
22 path: "./views",
23 layoutPath: "./views/layouts",
24 partialsPath: "./views/partials",
25 layout: true,
26 isCached: false,
27 });
28 server.route(webRoutes);
29 await server.start();
30 console.log("Server running on %s", server.info.uri);
31 }
32
33 process.on("unhandledRejection", (err) => {
34 console.log(err);
35 process.exit(1);
36 });
37
38 init();
39
master* 0 0 {..} : Scanning..
Ln 39, Col 1 Spaces: 2 UTF-8 LF JavaScript ESLint Prettier [off]


Playtime
Log in
Sign up
Sign up
Name
Enter first name
Enter last name
Email
Enter email
Password
Enter Password
Submit


Playtime
Log in
Sign up
Log in
Email
Enter email
Password
Enter Password
Submit


Playtime
Dashboard
About
Logout
Beethoven Sonatas
Beethoven Concertos
Playlist Title
Enter playlist title
Add Playlist

menu.hbs

```
<nav class="navbar">
  <div class="navbar-brand">
    {{> playtime-brand}}
  </div>
  <div class="navbar-menu" id="navMenu">
    <div class="navbar-end">
      <div class="navbar-item">
        <div class="buttons">
          <a id="dashboard" class="button" href="/dashboard"> Dashboard </a> <a id="about" class="button" href="/
          Logout </a>
        </div>
      </div>
    </div>
  </div>
</nav>
```

welcome-menu.hbs

```
<nav class="navbar" role="navigation" aria-label="main navigation">
  <div class="navbar-brand">
    {{> playtime-brand}}
  </div>
  <div id="navbarBasicExample" class="navbar-menu">
    <div class="navbar-end">
      <div class="navbar-item">
        <div class="buttons">
          <a class="button" id="login" href="/login"> Log in </a> <a class="button" id="signup" href="/signup"> Sign
          up </a>
        </div>
      </div>
    </div>
  </div>
</nav>

<script>
  document.getElementById("{{active}}").classList.add("is-primary");
</script>
```

list-playlist.hbs

```
{{#each playlists}}
  <div class="box box-link-hover-shadow">
    <h2 class="title">
      {{title}}
    </h2>
  </div>
{{/each}}
```

add-playlist.hbs

```
<form action="/dashboard/addplaylist" method="POST">
  <div class="field">
    <label class="label">Playlist Title</label> <input class="input" type="text" placeholder="Enter playlist titl
  </div>
  <button class="button is-link">Add Playlist</button>
</form>
```

dashboard-view.hbs

```
{{> menu active="dashboard"}}

<section class="section">
  {{> list-playlists}}
  {{> add-playlist}}
</section>
```

login-view.hbs

```
{{> welcome-menu active="login"}}

<section class="section">
  <h1 class="title">Log in</h1>
  <form action="/authenticate" method="POST">
    <div class="field">
      <label class="label">Email</label> <input class="input" type="text" placeholder="Enter email" name="email">
    </div>
    <div class="field">
      <label class="label">Password</label> <input class="input" type="password" placeholder="Enter Password" nam
    </div>
    <div class="field is-grouped">
      <button class="button is-link">Submit</button>
    </div>
  </form>
</section>
```

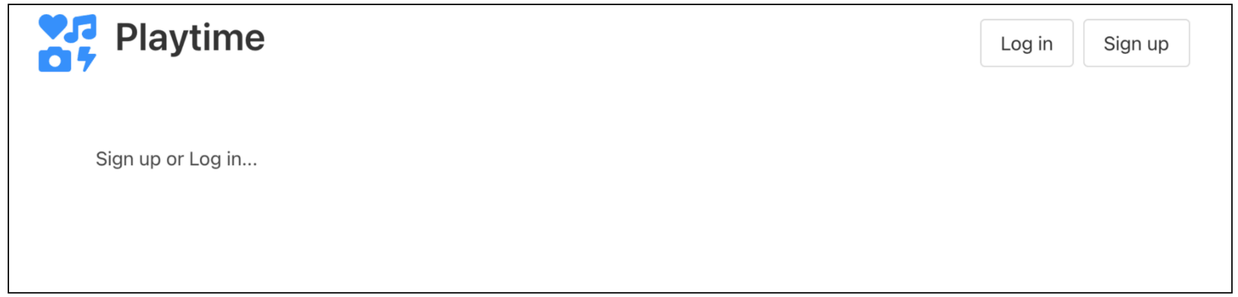
signup-view.hbs

```
{{> welcome-menu active="signup"}}

<section class="section">
  <h1 class="title">Sign up</h1>
  <form action="/register" method="POST">
    <label class="label">Name</label>
    <div class="field is-horizontal">
      <div class="field-body">
        <div class="field">
          <input class="input" type="text" placeholder="Enter first name" name="firstName">
        </div>
        <div class="field">
          <input class="input" type="text" placeholder="Enter last name" name="lastName">
        </div>
      </div>
    </div>
    <div class="field">
      <label class="label">Email</label> <input class="input" type="text" placeholder="Enter email" name="email">
    </div>
    <div class="field">
      <label class="label">Password</label> <input class="input" type="password" placeholder="Enter Password" nam
    </div>
    <div class="field is-grouped">
      <button class="button is-link">Submit</button>
    </div>
  </form>
</section>
```

Controllers

accountsController

```
import { db } from "../models/db.js";

export const accountsController = {
  index: {
    handler: function (request, h) {
      return h.view("main", { title: "Welcome to Playlist" });
    },
  },
  showSignup: {
    handler: function (request, h) {
      return h.view("signup-view", { title: "Sign up for Playlist" });
    },
  },
  signup: {
    handler: async function (request, h) {
      const user = request.payload;
      await db.userStore.addUser(user);
      return h.redirect("/");
    },
  },
```



accountsController

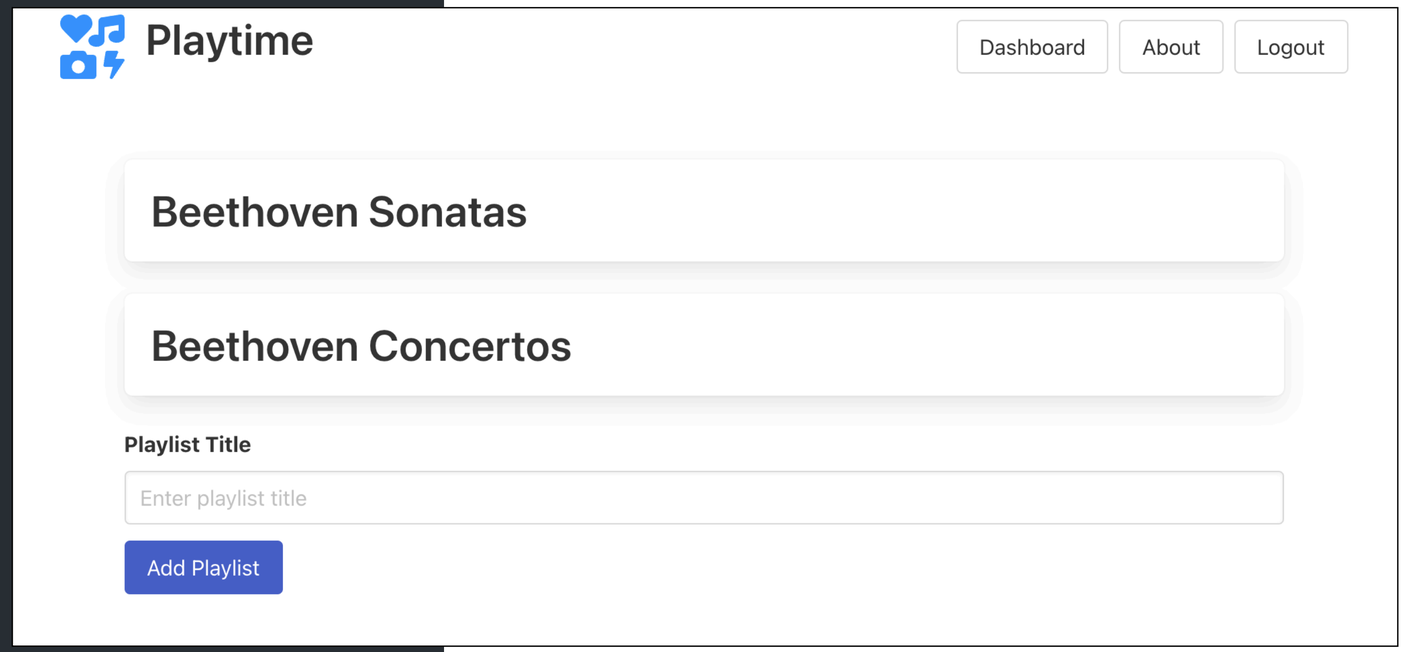
```
showLogin: {
  handler: function (request, h) {
    return h.view("login-view", { title: "Login to Playlist" });
  },
},
login: {
  handler: async function (request, h) {
    const { email, password } = request.payload;
    const user = await db.userStore.getUserByEmail(email);
    if (!user || user.password !== password) {
      return h.redirect("/");
    }
    return h.redirect("/dashboard");
  },
},
```

dashboardController

```
import { db } from "../models/db.js";

export const dashboardController = {
  index: {
    handler: async function (request, h) {
      const playlists = await db.playlistStore.getAllPlaylists();
      const viewData = {
        title: "Playtime Dashboard",
        playlists: playlists,
      };
      return h.view("dashboard-view", viewData);
    },
  },

  addPlaylist: {
    handler: async function (request, h) {
      const newPlayList = {
        title: request.payload.title,
      };
      await db.playlistStore.addPlaylist(newPlayList);
      return h.redirect("/dashboard");
    },
  },
};
```

dashboard-view.hbs

```
{{> menu active="dashboard"}}

<section class="section">
  {{> list-playlists}}
  {{> add-playlist}}
</section>
```



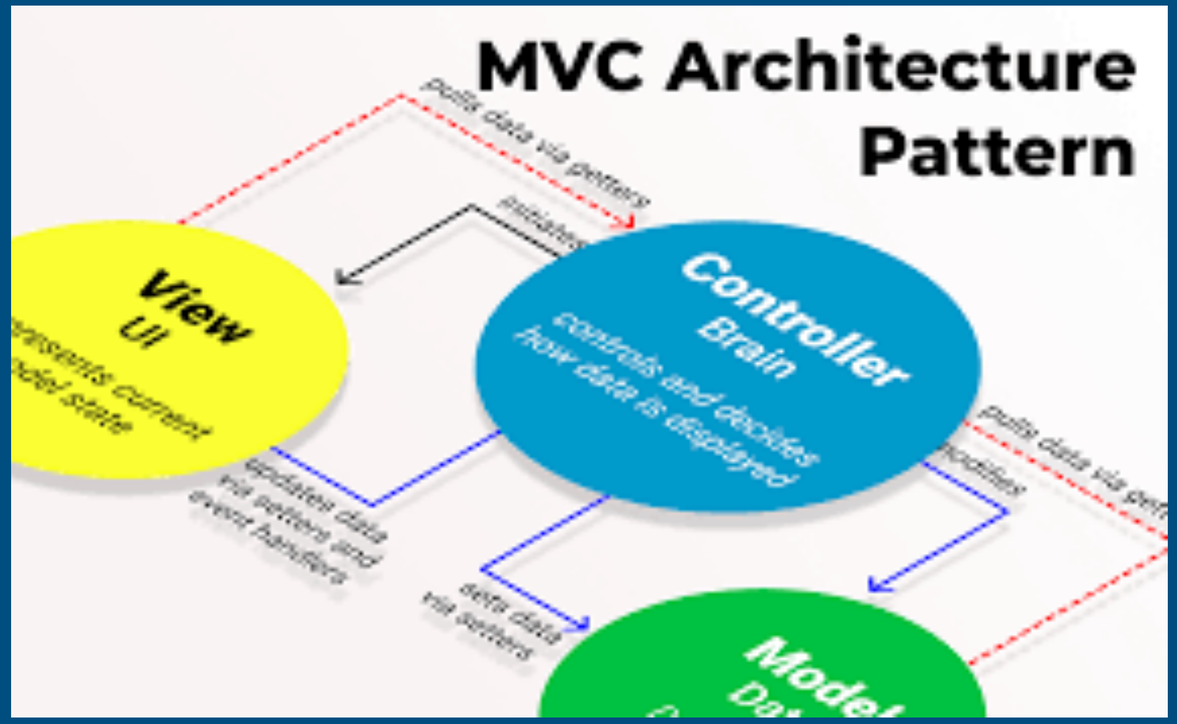
web-routes.js

```
import { accountsController } from "./controllers/accounts-controller.js";
import { dashboardController } from "./controllers/dashboard-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: "/dashboard", config: dashboardController.index },
  { method: "POST", path: "/dashboard/addplaylist", config: dashboardController.addPlaylist },
];
```

Server

```
async function init() {
  const server = Hapi.server({
    port: 3000,
    host: "localhost",
  });
  await server.register(Vision);
  server.views({
    engines: {
      hbs: Handlebars,
    },
    relativeTo: __dirname,
    path: "./views",
    layoutPath: "./views/layouts",
    partialsPath: "./views/partials",
    layout: true,
    isCached: false,
  });
  db.init();
  server.route(webRoutes);
  await server.start();
  console.log("Server running on %s", server.info.uri);
}
```

MVC in HAPI



Full Stack Web Development