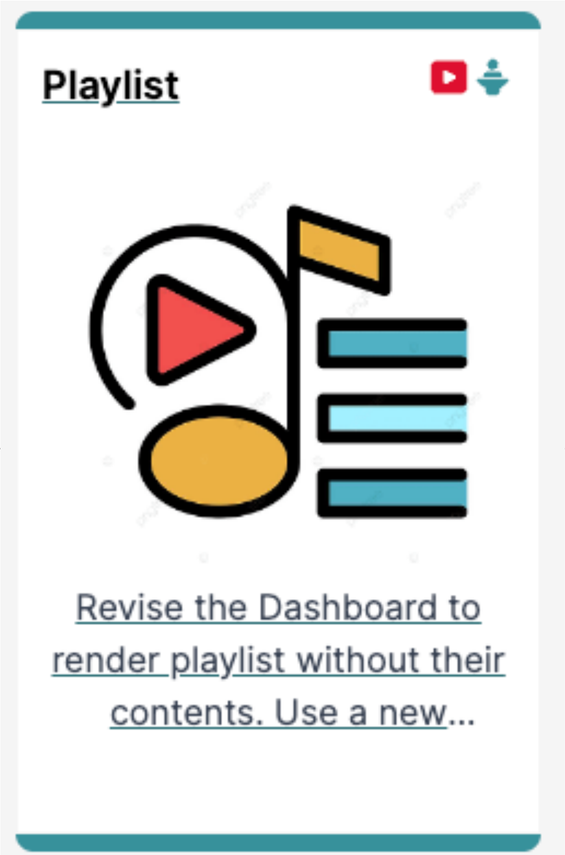
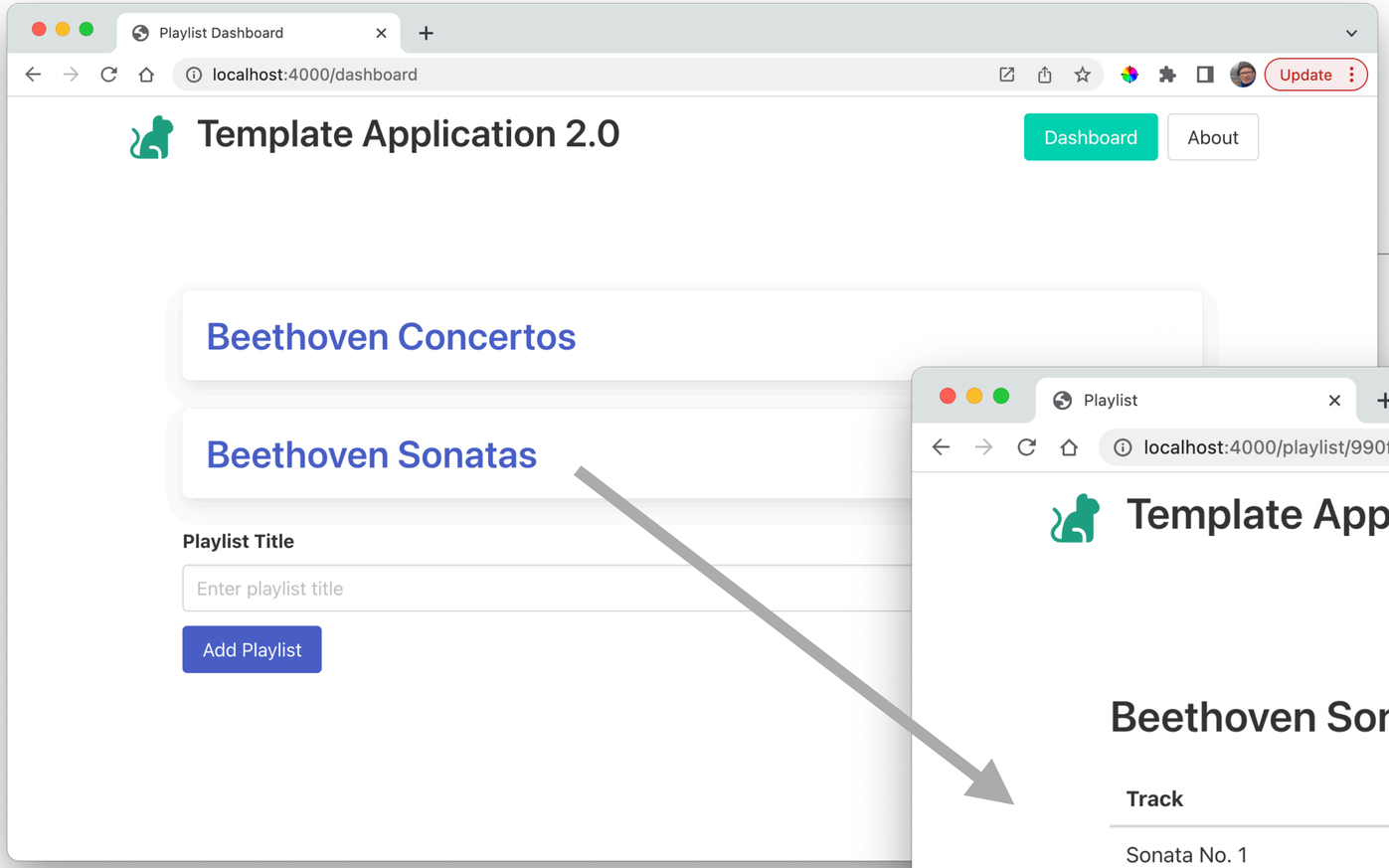
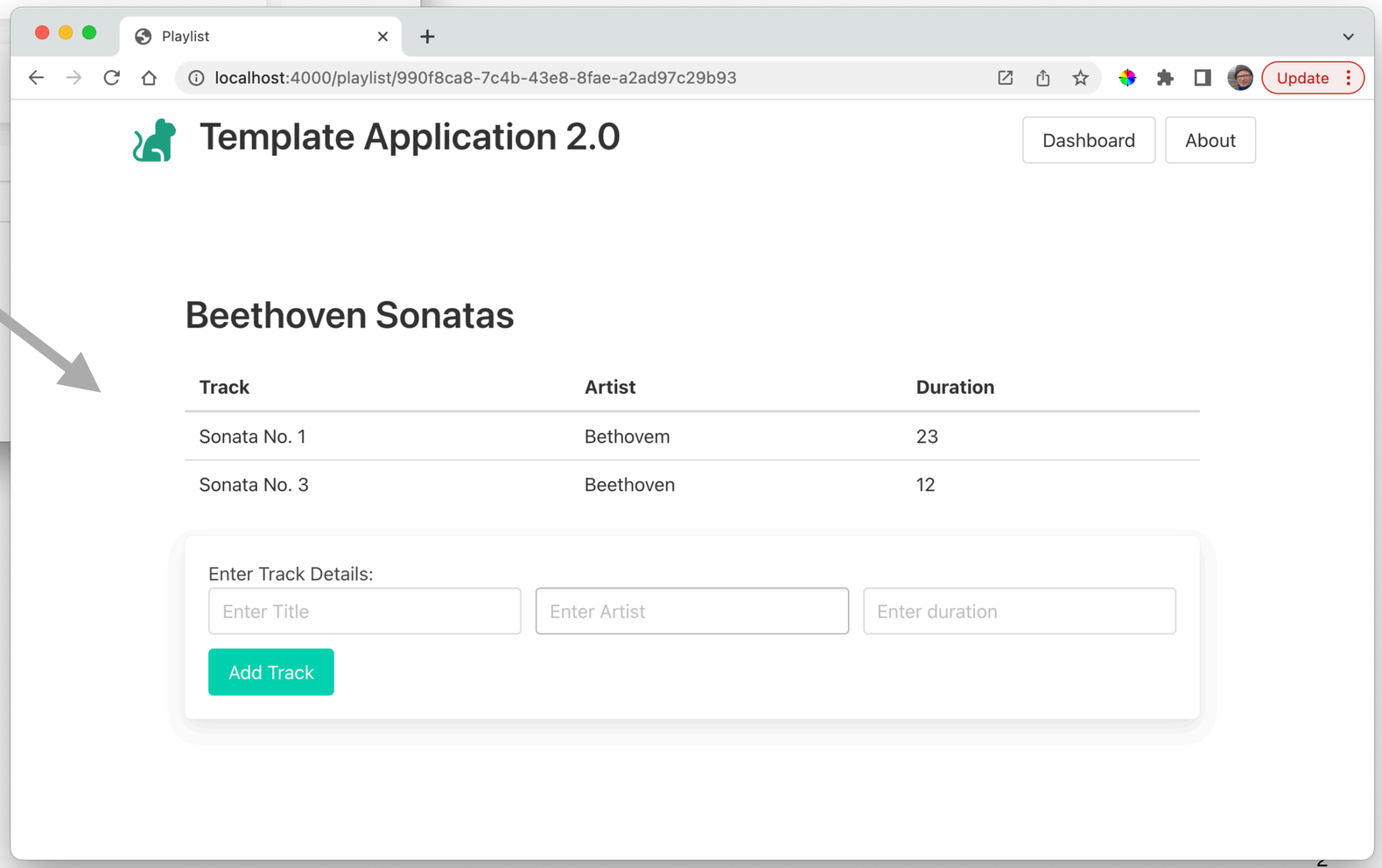
Playlist Controller



Dashboard -> Playlist

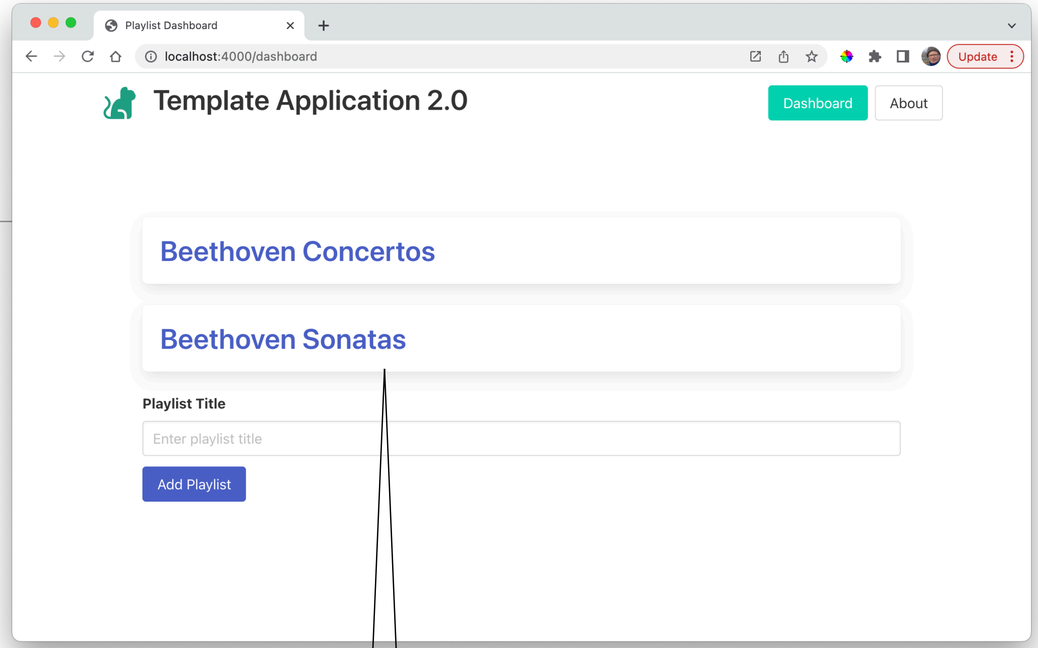




```
import { playlistStore } from "../models/playlist-store.js";

export const dashboardController = {
  async index(request, response) {
    const viewData = {
      title: "Playlist Dashboard",
      playlists: await playlistStore.getAllPlaylists(),
    };
    console.log("dashboard rendering");
    response.render("dashboard-view", viewData);
  },
  …
};
```

dashboard-controller.js



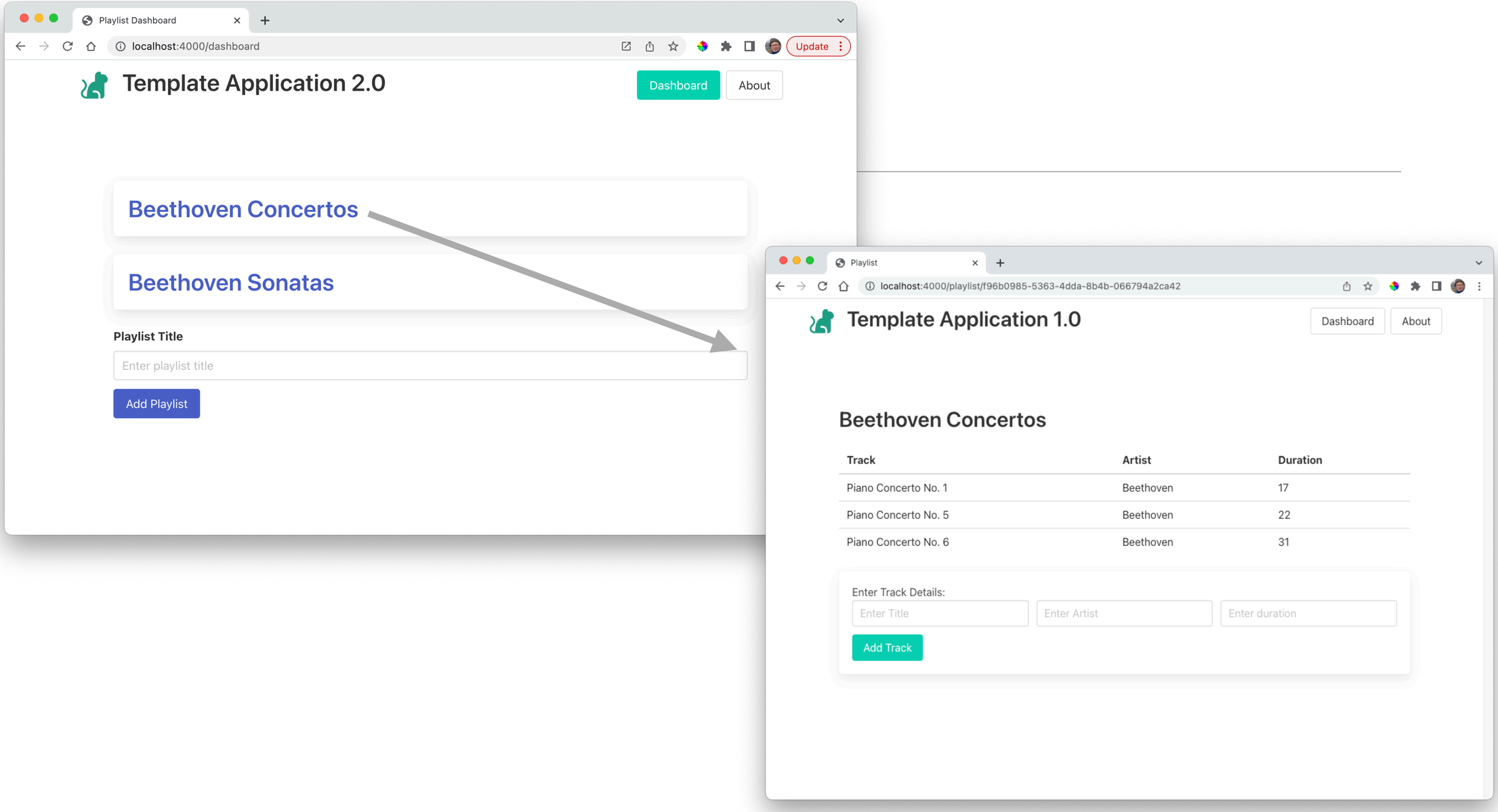

```
{{> menu active="dashboard"}}

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

dashboard-view.hbs

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

list-playlists.hbs



- Full playlist to be displayed when Playlist title selected

Introducing a new view

- Typically need
  - Route
  - Controller
  - View
  - Model

Introducing a new view

routes.js

```
import { playlistController } from "./controllers/playlist-controller.js";
...
...
router.get("/playlist/:id", playlistController.index);
```

playlist-controller.js

```
import { playlistStore } from "../models/playlist-store.js";

export const playlistController = {
  async index(request, response) {
    const playlist = await playlistStore.getPlaylistById(request.params.id);
    const viewData = {
      title: "Playlist",
      playlist: playlist,
    };
    response.render("playlist-view", viewData);
  },
};
```

playlist-view.hbs

```
{{> menu}}

<section class="section">
  <div class="title">
    {{playlist.title}}
  </div>
</section>
```

- Typically need
  - Route
  - Controller
  - View
  - Model

Introducing a new view

- Typically need
  - Route
  - Controller
  - View
  - Model

```
import { v4 } from "uuid";
import { initStore } from "../utils/store-utils.js";

const db = initStore("tracks");

export const trackStore = {
  async getAllTracks() {
    await db.read();
    return db.data.tracks;
  },

  async addTrack(playlistId, track) {
    await db.read();
    track._id = v4();
    track.playlistid = playlistId;
    db.data.tracks.push(track);
    await db.write();
    return track;
  },

  async getTracksByPlaylistId(id) {
    await db.read();
    return db.data.tracks.filter((track) => track.playlistid === id);
  },

  async getTrackById(id) {
    await db.read();
    return db.data.tracks.find((track) => track._id === id);
  },
```

```
  async deleteTrack(id) {
    await db.read();
    const index = db.data.tracks.findIndex((track) => track._id === id);
    db.data.tracks.splice(index, 1);
    await db.write();
  },

  async deleteAllTracks() {
    db.data.tracks = [];
    await db.write();
  },

  async updateTrack(track, updatedTrack) {
    track.title = updatedTrack.title;
    track.artist = updatedTrack.artist;
    track.duration = updatedTrack.duration;
    await db.write();
  },
};
```

- Ultimately these tracks are maintained in a Json file called "tracks.json".

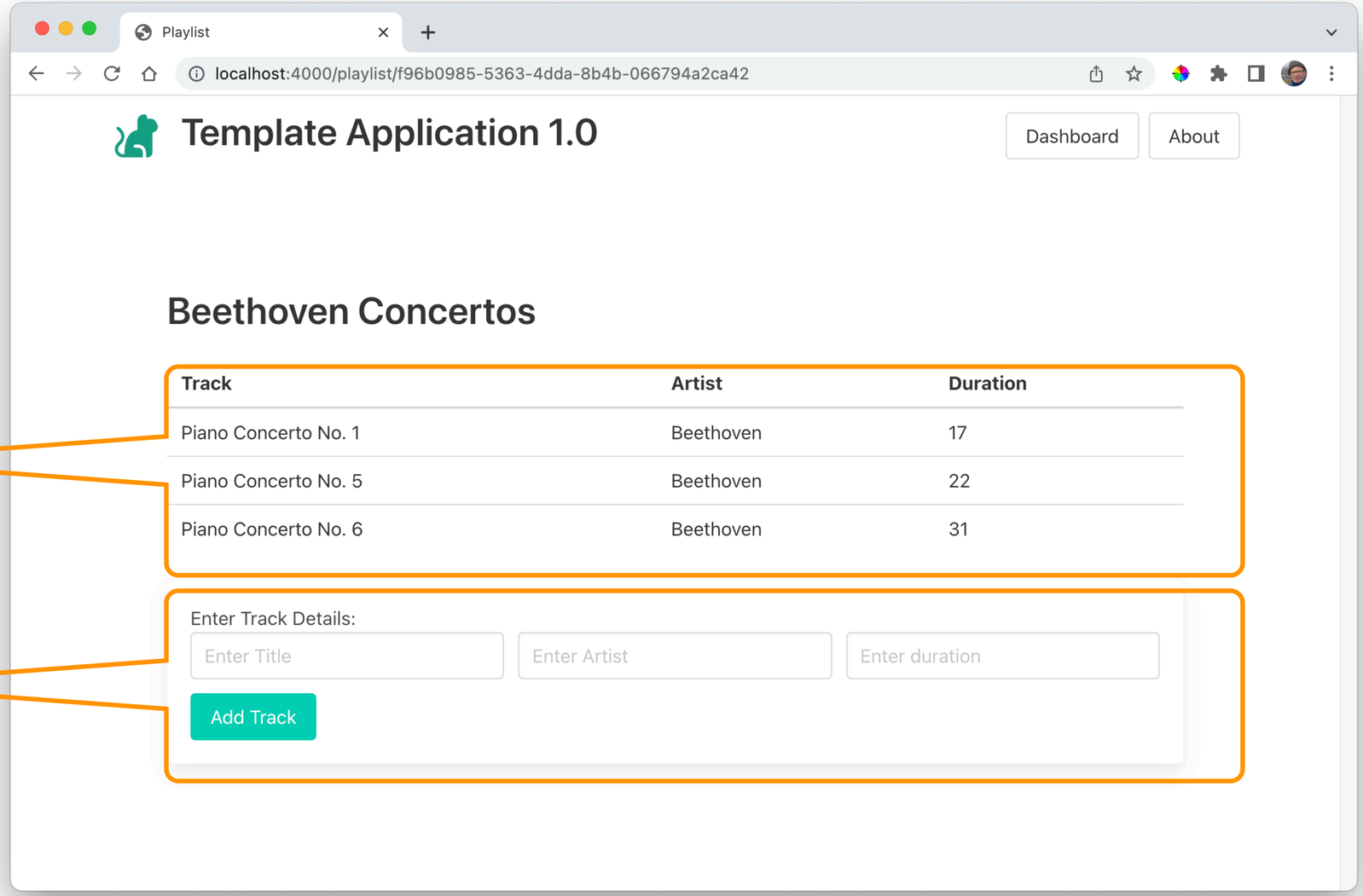
- We do not need to go into the details of how this works for the moment.

- It provides a service whereby we add, remove or update tracks via the methods listed above.

List / Add Tracks

List Tracks

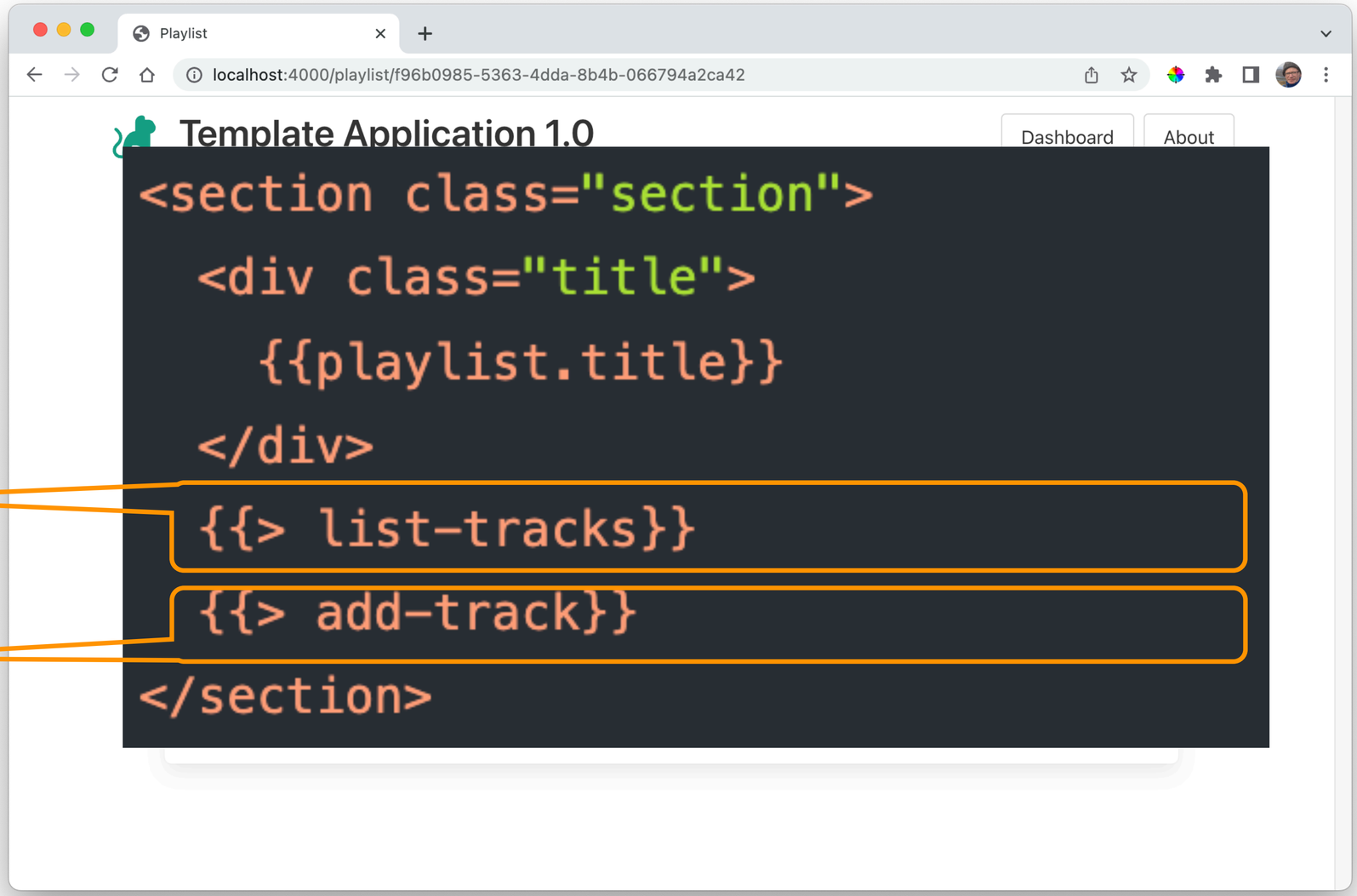
Add Track



List / Add Tracks

List Tracks

Add Track



List Tracks

```
<table class="table is-fullwidth">
  <thead>
    <tr>
      <th>Track</th>
      <th>Artist</th>
      <th>Duration</th>
    </tr>
  </thead>
  <tbody>
    {{#each playlist.tracks}}
    <tr>
      <td>
        {{title}}
      </td>
      <td>
        {{artist}}
      </td>
      <td>
        {{duration}}
      </td>
    </tr>
    {{/each}}
  </tbody>
</table>
```

list-tracks.hbs

```
export const playlistController = {
  async index(request, response) {
    const playlist = await playlistStore.getPlaylistById(request.params.id);
    const viewData = {
      title: "Playlist",
      playlist: playlist,
    };
    response.render("playlist-view", viewData);
  },
```

- Retrieve playlist tracks from store
- Send to the view to be displayed

Track	Artist	Duration
Piano Concerto No. 1	Beethoven	17
Piano Concerto No. 5	Beethoven	22
Piano Concerto No. 6	Beethoven	31

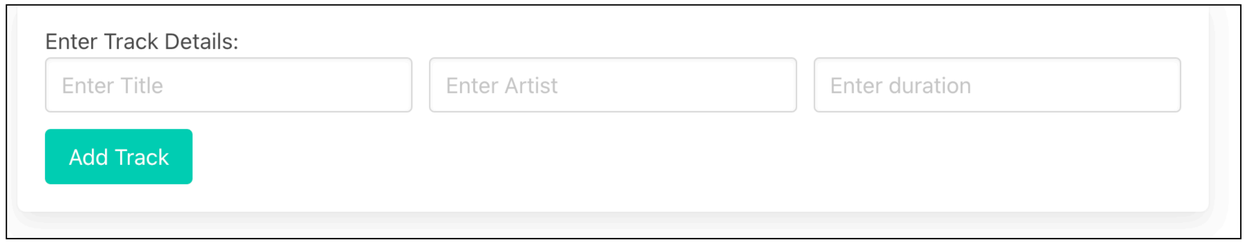
Add Tracks

routes.js

```
router.post("/playlist/:id/addtrack", playlistController.addTrack);
```

add-track.hbs

```
<form class="box" action="/playlist/{{playlist._id}}/addtrack" method="POST">
  <label>Enter Track Details:</label>
  <div class="field is-horizontal">
    <div class="field-body">
      <div class="field">
        <input class="input" type="text" placeholder="Enter Title" name="title">
      </div>
      <div class="field">
        <input class="input" type="text" placeholder="Enter Artist" name="artist">
      </div>
      <div class="field">
        <input class="input" type="text" placeholder="Enter duration" name="duration">
      </div>
    </div>
  </div>
  <button class="button is-primary">Add Track</button>
</form>
```



Add Tracks

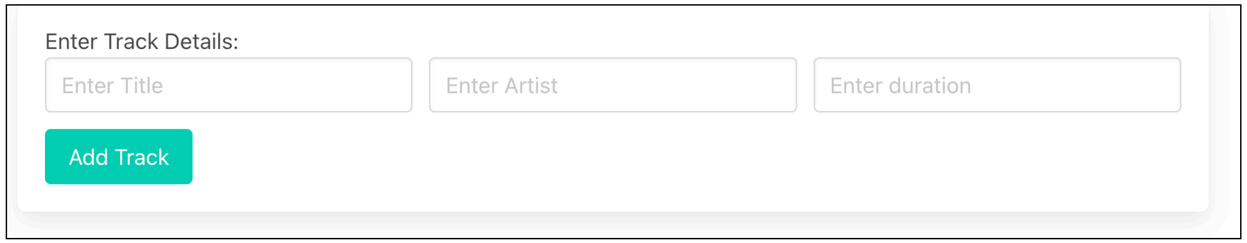
routes.js

```
router.post("/playlist/:id/addtrack", playlistController.addTrack);
```

addTrack controller / action

```
...

  async addTrack(request, response) {
    const playlist = await playlistStore.getPlaylistById(request.params.id);
    const newTrack = {
      title: request.body.title,
      artist: request.body.artist,
      duration: Number(request.body.duration),
    };
    console.log(`adding track ${newTrack.title}`);
    await trackStore.addTrack(playlist._id, newTrack);
    response.redirect("/playlist/" + playlist._id);
  },
```



Playlist Controller