CRUD Demo Project: Express + TypeScript + MySQL + Sequelize

---

1. Project Setup

```
mkdir list-app && cd list-app
npm init -y
npm install express mysql2 sequelize reflect-metadata typescript ts-node-dev
body-parser
npm install -D @types/express @types/node ts-node sequelize-cli
npx tsc --init
```

Update `tsconfig.json`:

```
{
  "compilerOptions": {
    "target": "ES2022",
    "module": "CommonJS",
    "rootDir": "./src",
    "outDir": "./dist",
    "strict": true,
    "esModuleInterop": true,
    "experimentalDecorators": true,
    "emitDecoratorMetadata": true
  }
}
```

---

2. Folder Structure

```
list-app/
│
├─ src/
│   ├─ core/logic/
│   │   ├─ Result.ts
│   │   └─ AppError.ts
│   ├─ infrastructure/database/
│   │   ├─ sequelize.ts
│   │   └─ models/List.ts
│   ├─ modules/list/
│   │   ├─ domain/
│   │   │   ├─ list.ts
│   │   │   └─ listPicture.ts
│   │   ├─ repos/
│   │   │   ├─ interfaces/IListRepo.ts
│   │   │   └─ listRepo.ts
│   │   └─ useCases/
│   │       ├─ createList/CreateListUseCase.ts
│   │       ├─ createList/CreateListController.ts
│   │       ├─ listLists/ListListsUseCase.ts
│   │       ├─ listLists/ListListsController.ts
│   │       ├─ updateList/UpdateListUseCase.ts
│   │       ├─ updateList/UpdateListController.ts
│   │       ├─ deleteList/DeleteListUseCase.ts
│   │       └─ deleteList/DeleteListController.ts
│   └─ server.ts
├─ migrations/
│   └─ <timestamp>-create-lists.js
├─ package.json
└─ tsconfig.json
```

3. Sequelize Setup

src/infrastructure/database/sequelize.ts

```
import { Sequelize } from 'sequelize';
export const sequelize = new Sequelize('list_app', 'root', 'yourpassword', {
  host: 'localhost',
  dialect: 'mysql',
  logging: false,
});
```

src/infrastructure/database/models/List.ts

```
import { DataTypes, Model } from 'sequelize';
import { sequelize } from '../sequelize';

export class List extends Model {
  public id!: number;
  public name!: string;
  public picture!: string;
  public startDateTime!: Date;
  public endDateTime!: Date;
  public groupId!: number;
  public readonly createdAt!: Date;
  public readonly updatedAt!: Date;
}

List.init({
  id: { type: DataTypes.INTEGER.UNSIGNED, autoIncrement: true, primaryKey:
true },
  name: { type: DataTypes.STRING, allowNull: false },
  picture: { type: DataTypes.STRING, allowNull: false },
  startDateTime: { type: DataTypes.DATE, allowNull: false },
  endDateTime: { type: DataTypes.DATE, allowNull: false },
  groupId: { type: DataTypes.INTEGER.UNSIGNED, allowNull: false }
}, { sequelize, modelName: 'List', tableName: 'lists' });
```

---

4. Migration

migrations/20251103-create-lists.js

```
'use strict';
module.exports = {
  async up(queryInterface, Sequelize) {
    await queryInterface.createTable('lists', {
      id: { type: Sequelize.INTEGER.UNSIGNED, autoIncrement: true, primaryKey:
true },
      name: { type: Sequelize.STRING, allowNull: false },
      picture: { type: Sequelize.STRING, allowNull: false },
      startDateTime: { type: Sequelize.DATE, allowNull: false },
      endDateTime: { type: Sequelize.DATE, allowNull: false },
      groupId: { type: Sequelize.INTEGER.UNSIGNED, allowNull: false },
      createdAt: { type: Sequelize.DATE, allowNull: false, defaultValue:
Sequelize.literal('CURRENT_TIMESTAMP') },
      updatedAt: { type: Sequelize.DATE, allowNull: false, defaultValue:
Sequelize.literal('CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP') }
    });
  },
  async down(queryInterface, Sequelize) {
    await queryInterface.dropTable('lists');
  }
};
```

Run migration:

```
npx sequelize-cli db:migrate
```

---

5. Core Logic

src/core/logic/Result.ts

```
export class Result<T> { /* as in previous example */ }
export type Either<L, R> = Left<L, R> | Right<L, R>;
export const left = <L, R>(l: L): Either<L, R> => new Left(l);
export const right = <L, R>(r: R): Either<L, R> => new Right(r);
```

src/core/logic/AppError.ts

```
export namespace AppError { export class UnexpectedError extends Error {
constructor(error: any){ super(error?.message || 'Unexpected error occurred');
this.name = 'UnexpectedError'; } } }
```

---

6. Domain Layer

list.ts and **listPicture.ts** as in previous example

---

7. Repository Layer

IListRepo.ts and **listRepo.ts** as in previous example, with `getAll` method added.

---

8. UseCases & Controllers

Create, List, Update, Delete UseCases & Controllers

- `CreateListUseCase.ts` & `CreateListController.ts` (POST /lists)
- `ListListsUseCase.ts` & `ListListsController.ts` (GET /lists)
- `UpdateListUseCase.ts` & `UpdateListController.ts` (PUT /lists/:id)
- `DeleteListUseCase.ts` & `DeleteListController.ts` (DELETE /lists/:id)

Each UseCase calls repository methods; each Controller handles Express req/res.

Example **UpdateListUseCase.ts**:

```
export class UpdateListUseCase {
  constructor(private listRepo: IListRepo) {}
  async execute(id: number, data: any) {
    const list = await this.listRepo.getById(id);
    if (!list) throw new Error('List not found');
    Object.assign(list, data);
    return await this.listRepo.save(list);
  }
}
```

---

9. Express Server

```
import express from 'express';
import bodyParser from 'body-parser';
import { sequelize } from './infrastructure/database/sequelize';
import { ListRepo } from './modules/list/repos/listRepo';
import { CreateListUseCase } from './modules/list/useCases/createList/
CreateListUseCase';
import { CreateListController } from './modules/list/useCases/createList/
CreateListController';
import { ListListsUseCase } from './modules/list/useCases/listLists/
ListListsUseCase';
import { ListListsController } from './modules/list/useCases/listLists/
ListListsController';

const app = express();
app.use(bodyParser.json());

const listRepo = new ListRepo();

const createListController = new CreateListController(new
CreateListUseCase(listRepo));
const listListsController = new ListListsController(new
ListListsUseCase(listRepo));

app.post('/lists', (req, res) => createListController.handle(req, res));
app.get('/lists', (req, res) => listListsController.handle(req, res));

sequelize.sync().then(() => app.listen(3000, () => console.log('Server running
on http://localhost:3000')));
```

---

10. Example Requests

POST /lists

```json { "name": "Shopping List
```