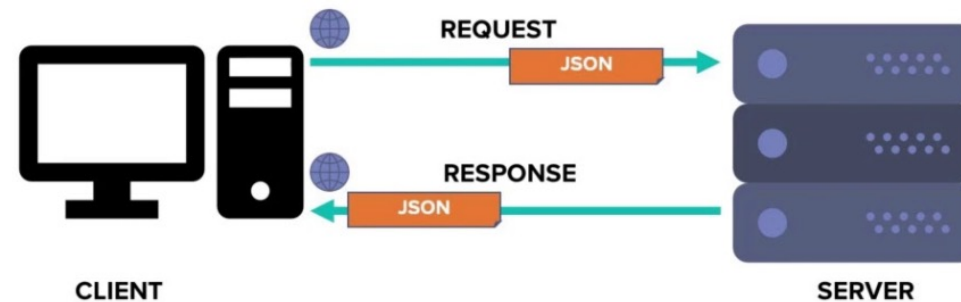


Project Assignment

EzElectronics implementation

EzElectronics

- EzElectronics is a web application composed of:
 - A web server, which exposes HTTP/Rest/JSON APIs
 - A web client which interacts with the server through HTTP/Rest/JSON calls



EzElectronics Client

- Not available for the current assignment
- Will be provided for the second milestone (code implementation)

EzElectronics Server

- Sketched in node.js (with the Express module)
- Available in your repo in the `code/server` folder
- Will provide the implementation of the HTTP/Rest/Json APIs defined in `API.md`
- In the current state it will not work as some functions are not implemented, but that is not necessary for the goal of this first assignment.

EzElectronics Server

- Server source code

```
const cors = require('cors');
import express from 'express';
import initRoutes from './src/routes'
import dotenv from 'dotenv';

dotenv.config();
const app: express.Application = express();

const port: number = 3001;

const corsOptions = {
  origin: 'http://localhost:3000',
  credentials: true,
};

app.use(cors(corsOptions));
initRoutes(app)
if (!module.parent) {
  app.listen(port, () => {
    console.log(`Server listening at http://localhost:${port}`);
  });
}

export { app }
```

npm package manager

- EzElectronics client and server use the npm package manager
- npm is a package manager for the JavaScript programming language. It is the default package manager for the JavaScript runtime environment Node.js
- It consists of a command line client, also called npm, and an online database of public and paid-for private packages, called the npm registry
- If you do not have it yet, please install npm in your local machine!

npm package manager

- When using the npm package manager, two files will be created
 - `package.json`: holds various metadata relevant to the project. It gives information to npm to handle the project's dependencies and some shortcuts to execute complex scripts (e.g., for testing the code). More info at <https://docs.npmjs.com/cli/v6/configuring-npm/package-json>
 - `package-lock.json`: keeps track of the exact version of every package that is installed

EzElectronics server package.json

- The `dependencies` section allows the `npm install` command to install all the `express` dependency
- The `dev-dependencies` section allows the `npm install` command to install all the testing dependencies which will be use for development only
- The `scripts` section defines commands for starting the server (`npm start`), and for launching tests (`npm run test`) .

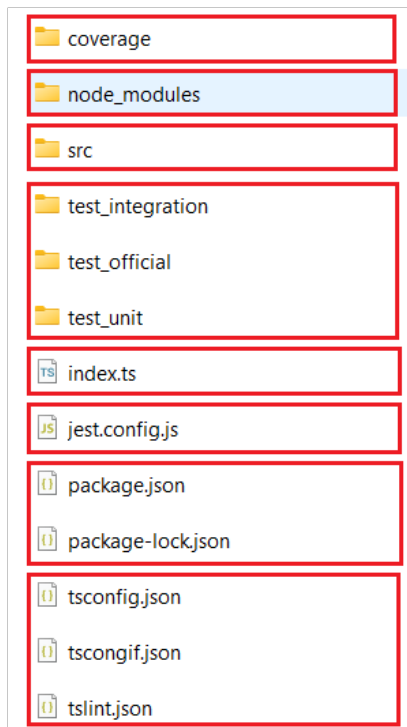
Running EzElectronics server

- Open a terminal
- Open the `code/server` folder
- Type `npm install`
- Type `npm start`
- This message will be shown: `Server listening at http://localhost:3001`
- **Do not close the terminal window**, otherwise the server will shut down

Server folder structure










Files – code/server

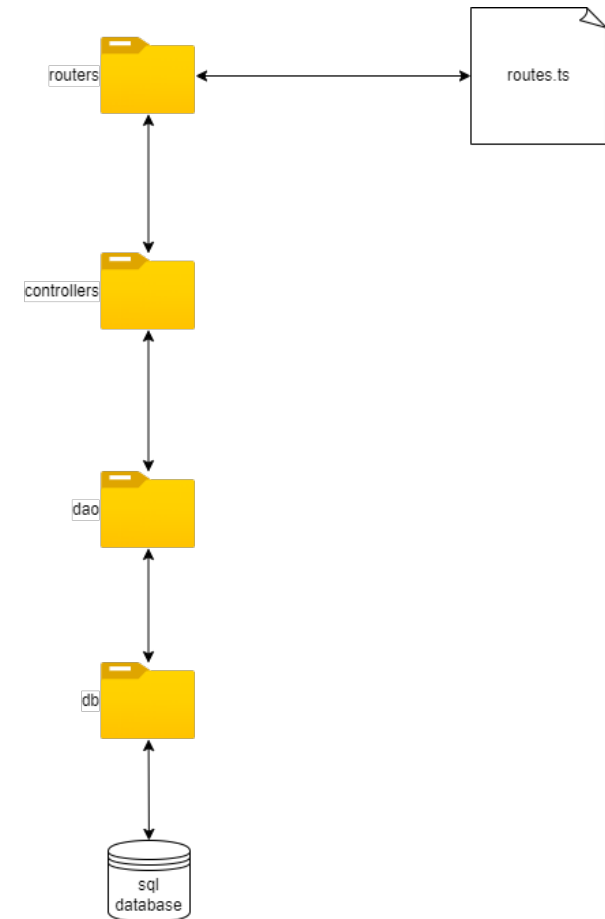
- In your repo you will find these files:



- Information on coverage obtained with tests
- Node dependencies installed after running *npm install*
- Full server implementation
- Tests, separated by category
- Server starting file
- Test configuration file
- Node dependencies files
- Typescript configuration files

Files – code/server/src

 components	• Defines the classes used by the app
 controllers	• Bridges routes and database interaction methods
 dao	• Defines methods that interact with the database
 db	• Connects the app to the database
 errors	• Defines error objects
 routers	• Defines the different routes of the app
 helper.ts	• Defines error handling middlewares
 routes.ts	• Defines the different route path roots
 utilities.ts	• Defines utility middlewares



Project Milestones

The project milestones

- Step 1 (4 weeks from now)
 - Step1-1
 - Reverse documentation (requirements) of given app V1
 - Step 1-2
 - Invent evolution to V2 (requirements v2)
- Step 2 (6 weeks from may 5)
 - Implement evolution to V2

Step 1-1

- Input (from teachers)
 - Code and API of the application v1
 - (will be soon in your git repo)
- Output
 - Requirement document v1
 - GUI prototype v1
 - Estimation document v1

Step 1-1

- Is a reverse documentation case
 - From code to requirements

Reverse Engineering

- The server structure shown in the earlier images is incomplete and not working in the current implementation (interaction with the database is missing).
- Your goal will be to reverse engineer the requirements document from which the server was implemented.
- Coding is not required for this assignment, the actual implementation will be done after the first delivery (additional files and the client will be provided).

Step 1-2

- Proposal of new functionalities
- Output
 - Requirement document – v2
 - GUI prototype –v2
 - Estimation document – v2

Step 1-2

- Is an evolution case
 - from an existing application to a new (improved) version

Deadlines

Deliverables	Duration	Deadline
Requirements doc GUI prototype Estimation V1	4 weeks	May 5
Requirements doc GUI prototype Estimation V2	4 weeks	May 5