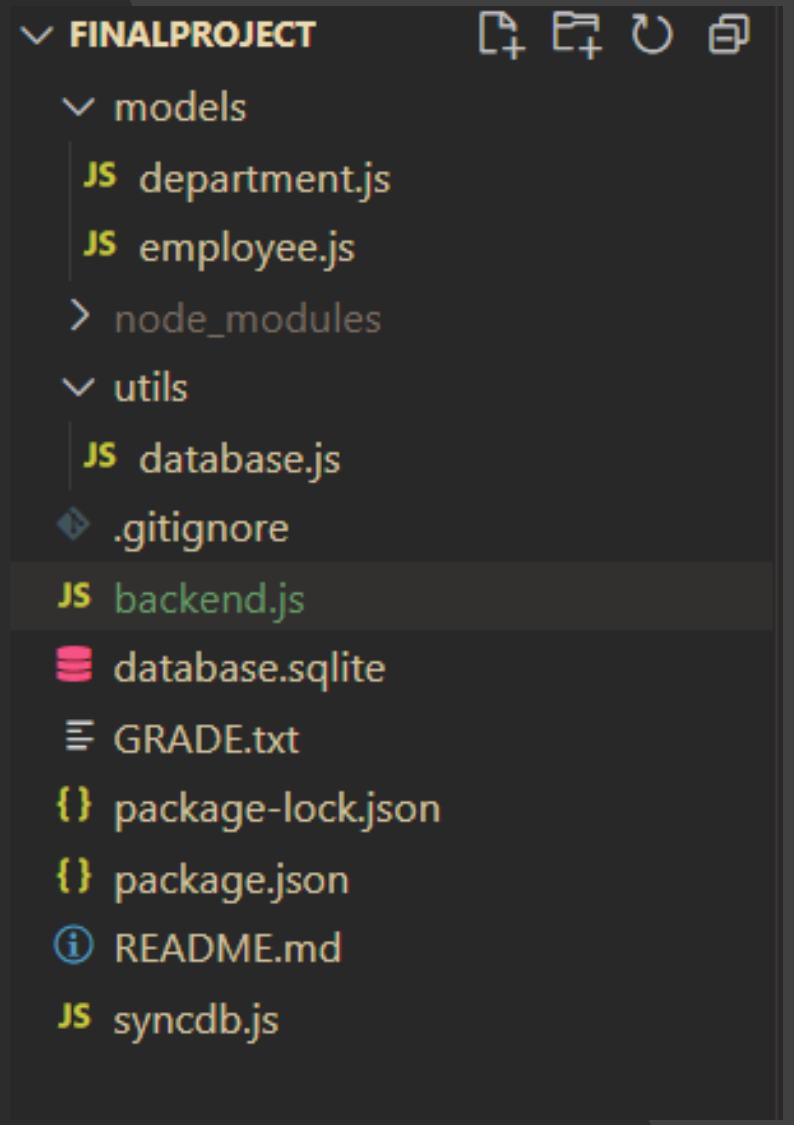
API Service Development

Author: Ozan Topcu

Project Overview

- The main logic behind this project is to provide an API service between a company database and an end user.

- The project consists of a “database.sqlite” file that stores both the user information and department information in its respective tables.



Technical Overview

Used Technologies:

- **Frameworks**
  - **Sequelize**
  - **Express.js**
- **Middleware**
  - **Body-parser**
- **Database**
  - **SQLite**

Project Structure:

-utils

- database.js

-models

- department.js

- employee.js

- syncdb.js

- backend.js

Timetable:	
Setting up the express framework	**30m**
Setting up the database using Sequelize	1h
Creating endpoints and testing them	3h30m

Models

models > JS employee.js > ...

```
1   const Sequelize = require("sequelize");
2   const sequelize = require("../utils/database");
3
4   // Defining a table structure
5   const Employee = sequelize.define('Employee', {
6       employeename: {
7           type: Sequelize.STRING,
8           allowNull: false
9       },
10      departmentname: {
11          type: Sequelize.STRING,
12          allowNull: false
13      },
14  });
15
16  module.exports = Employee;
```

models > JS department.js > ...

```
1   const Sequelize = require("sequelize");
2   const sequelize = require("../utils/database");
3
4   // Defining a table structure
5   const Department = sequelize.define('Department', {
6       departmentname: {
7           type: Sequelize.STRING,
8           allowNull: false
9       },
10      departmentbudget: {
11          type: Sequelize.INTEGER,
12          allowNull: false
13      },
14  });
15
16  module.exports = Department;
```

These model files here define each table structure without being stuck to a single database engine format thanks to Sequelize.

Backend.js

```
// Define a route to update the existing employees
app.post('/employees/:id', async (req, res) => {
    const employeeID = req.params.id;

    try {
        // Find the employee from the database with the primary key which is id
        const employeeToUpdate = await Employee.findByPk(employeeID);

        // If the employee with the given ID doesn't exist, return a 404 Not Found response
        if (!employeeToUpdate) {
            return res.status(404).json({ error: 'Employee not found' });
        }

        // Update the employee's properties with the data from the request body
        await employeeToUpdate.update(req.body);

        // Send the updated employee as a JSON response
        res.json(employeeToUpdate);
    } catch (error) {
        console.error('Error updating employee:', error);
        res.status(500).json({ error: 'Internal Server Error' });
    }
});
```

```
// Setting up the GET method for the employees route of the API
app.get("/employees", async (req, res) => {
    try {
        // Fetch all employees from the Employee table
        const employees = await Employee.findAll();
        // Send the employees as a JSON response
        res.json(employees);
    } catch (error) {
        console.error('Error fetching employees:', error);
        res.status(500).json({ error: 'Internal Server Error' });
    }
});
```

```
// Define a route to create a new employee
app.post('/employees', async (req, res) => {
    try {
        // Create a new employee using data from the request body
        const newEmployee = await Employee.create(req.body);

        // Send the newly created employee as a JSON response
        res.json(newEmployee);
    } catch (error) {
        console.error('Error creating employee:', error);
        res.status(500).json({ error: 'Internal Server Error' });
    }
});
```

Examples of several HTTP requests (get, post, update)

Thank You