Full Stack Web Development: Backend Project

- Author: Innokentii Kozlov

- Date: 16.12.2023

Project Overview

API was developed as a final project of Full Stack Web Development Backend course. API utilizes data that was used during the course (backend part), ‘emp’ table to be more precise. API allows to implement CRUD operations which are listed further, as well as different techniques which are aslo listed further.

Used Technologies

- NodeJS + JavaScript
- Sqlite3
- Express, Nodemon
- Postman and curl

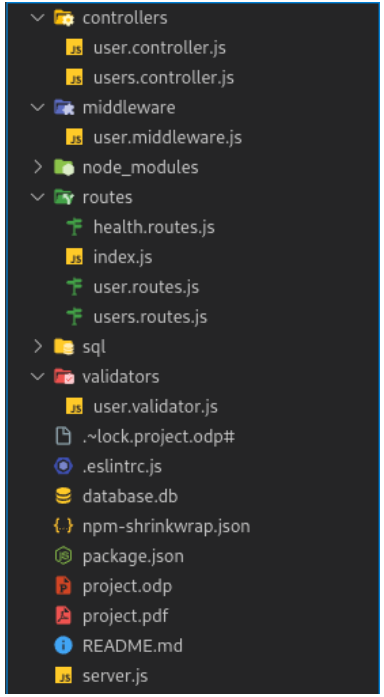
Timetable

Task	Time Spent
SQL Scripts	0.25 hours
API	12 hours
Documentation	0.5 hours
Presentation	2 hours

Code

- Project Structure
- NPM Scripts
- Express Application
- Routes
- Middle-ware
- Controllers
- Data Validation
- SQL and Sqlite

Project Structure



- server.js – entrance point of the NodeJS application
- Routes/ – API available routes
- Controllers/ – controllers
- Middleware/ – middle-ware
- Validators/ – data validation
- Sql/ – sql scripts: create, insert, delete data and tablles

NPM Scripts

```
"scripts": {
  "start:dev": "nodemon server.js",
  "start": "node server.js",
  "eslint": "eslint *.js",
  "fix": "eslint --fix *.js"
},
```

- package.json
- start:dev – starting development server
- start – starting production server
- eslint – check Lint
- fix – fix Lint

Express Application

```
server.js
server.js > ...
var apiRoutes = require('./routes');
var express = require('express');

var app = express();
var port = 8080;

app.use(express.json());
apiRoutes(app);

app.listen(port, () => {
    console.log(`-------------------------------------------------------`)
    console.log(`Example server listening at http://localhost:${port}/api`)
    console.log(`-------------------------------------------------------`)
})
```

- Initialize Express application instance.
- Initialize application port.
- Allow JSON as incoming payload data format.
- Apply API routes to the Express application instance.
- Start Express application instance.

Register Routes

index.js

routes > index.js > registerRoutes > app.use() callback

```javascript
const health = require('./health.routes');
const users = require('./users.routes');
const user = require('./user.routes');


const prefix = "/api"

const registerRoutes = (app) => {
    app.use(`${prefix}/health_`, health());
    app.use(`${prefix}/users`, users());
    app.use(`${prefix}/user`, user());
    app.use((req, res) => {
        res.setHeader('Content-Type', 'application/json');
        res.status(404).json({});
    });
}

module.exports = registerRoutes;
```

- Apply routers to endpoints and assign those to Express application.
- API consists of: ‘health_’, ‘users’, ‘user’ endpoints.
- API returns 404 if route wasn’t found.

Health Check Route

health.routes.js

routes > health.routes.js > ...

```javascript
const express = require('express');

const register = () => {
    const router = express.Router();

    router.get("/", (req, res) => {
      res.json({ status: "OK" });
    });

    return router;
};

module.exports = register;
```

- Return the following to verify API is available:

```
{
    "status": "OK"
}
```

Users Route

users.routes.js

routes > users.routes.js > ...

```
const express = require('express');
const { fetchUsers } = require('../controllers/users.controller');
const { userGetParamMiddleware } = require('../middleware/user.middleware')


const register = () => {
    const router = express.Router();
    router.get('/', userGetParamMiddleware, fetchUsers);
    return router;
};

module.exports = register;
```

- Fetching all users and fetching users by criteria are utilizing the very same route.
- Fetching Users By Criteria requires to verify sent criteria, hence the middle-ware was applied to this route.

User Routes

```
user.routes.js
routes > user.routes.js > ...
const express = require('express');
const { deleteUser, patchUser, putUser, createUser, fetchUser } = require('../controllers/user.controller');
const { userIdMiddleware, userMiddleware, userPatchParamMiddleware } = require('../middleware/user.middleware');


const register = () => {
    const router = express.Router();

    router.get('/:id', userIdMiddleware, fetchUser);
    router.post('/', userMiddleware, createUser);
    router.put('/:id', [ userIdMiddleware, userMiddleware ], putUser);
    router.patch('/:id', [ userIdMiddleware, userPatchParamMiddleware], patchUser);
    router.delete('/:id', userIdMiddleware, deleteUser);

    return router;
};

module.exports = register;
```

- CRUD operations for User.
- Applying middle-wares to the routes that have ‘id’ as a query parameter (represents ‘empno’), and payload validation middle-wares where payload is being sent.

Middle-ware

```
user.middleware.js
middleware > user.middleware.js > userMiddleware
14  const userIdMiddleware = async (req, res, next) => {
15
16      const id = req.params.id;
17
18      if (id === undefined || isNaN(id)) {
19          res.status(404).json({});
20          return;
21      }
22
23      next();
24
25  }
26
27  const userMiddleware = async (req, res, next) => {
28
29      const payload = req.body;
30
31      if (!isValidUser(payload)) {
32          res.status(400).json({});
33          return;
34      }
35
36      next();
37
38  }
```

- Example of middle-wares: verify sent Id, verify payload for HTTP POST (Create User)
- Middle-ware function uses Request, Response, and NextFunction. If all checks are passed ‘next()’ is called, otherwise the appropriate response is sent.

Controllers: Users

users.controller.js

controllers > users.controller.js > ...

```javascript
let sqlite3 = require("sqlite3").verbose();
let db = new sqlite3.Database("database.db");


const fetchUsers = async (req, res) => {

    const queryParams = req.query;
    const queryValues = Object.values(queryParams).map(value => value.split(',')).flat()

    const sql = Object.keys(queryParams).length < 1
                ? "SELECT * FROM emp"
                : `SELECT * FROM emp WHERE ${
                    Object.keys(queryParams)
                        .map(field => {
                            const values = queryParams[field].split(',');
                            return `( ${values.map(value => `${field} = ?`).join(' OR ')} )`;
                        })
                        .join(' AND ')
                }`

    db.all(sql, queryValues ,(err, rows) => {
        if (err) {
            res.status(500).json({});
            return;
        }
        if (rows.length < 1) {
            res.status(404).json({});
            return;
        }
        res.status(200).json(rows)
    });
}

module.exports = {
    fetchUsers: fetchUsers,
};
```

- Parsing query parameters.
- If query parameters do exist, form sql statement, otherwise keep simple ‘SELECT’ statement.’
- Run statement with parameters.
- If error or no records were found send 500 and 404 responses respectively, otherwise sent found records.

Controllers: User

users.controller.js | user.controller.js

controllers > user.controller.js > createUser

```javascript
let sqlite3 = require("sqlite3").verbose();
let db = new sqlite3.Database("database.db");


const fetchUser = async (req, res) => {
    const id = req.params.id;

    const sql = "SELECT * FROM emp WHERE empno = ?;"

    db.all(sql, [id], (err, row) => {
        if (err || row.length < 1) {
            res.status(404).json({});
        } else {
            res.status(200).json(row)
        }
    });
}

const createUser = async (req, res) => {
    const data = req.body;

    const sql = "INSERT INTO emp VALUES (?, ?, ?, ?, ?, ?, ?, ?);"

    const empno = data.empno
    const ename = data.ename
    const job = data.job
    const mgr = data.mgr
    const hiredate = data.hiredate
    const sal = data.sal
    const comm = data.comm
    const deptno = data.deptno

    db.run(sql, [empno, ename, job, mgr, hiredate, sal, comm, deptno], function (err) {
        if (err) {
            res.status(409).json({});
            return;
        }
        res.status(204).json({});
    })
}
```

- Fetching User By Id, or ‘empno’ in database. (HTTP GET)
- Create User with JSON payload. (HTTP POST)

Data Validation

user.validator.js

validators > user.validator.js > <unknown>

```javascript

const isValidUser = (data) => {

    if (data === undefined) {
        return false;
    }

    return (
        typeof data.empno === "number" &&
        typeof data.ename === "string" &&
        typeof data.job === "string" &&
        typeof data.mgr === "number" || data.mgr === null  &&
        typeof data.hiredate === "string" &&
        typeof data.sal === "number" &&
        typeof data.comm === "number" || data.comm === null &&
        typeof data.deptno === "number"
    );
}

module.exports = {
    isValidUser: isValidUser
}
```

Validate the payload being sent to the */api/user* endpoint with HTTP POST method according to database SQL Schema.

Create Tables

create.sql

sql > create.sql

```sql
-- ----------------------------------------------------------------------
--
--          Create Tables
--
-- ----------------------------------------------------------------------

CREATE TABLE dept
(
    deptno                  INTEGER       NOT NULL UNIQUE
    , dname                 VARCHAR(14)
    , loc                   VARCHAR(13)

    , CONSTRAINT dept_primary_key
      PRIMARY KEY (deptno)
);

CREATE UNIQUE INDEX dept_deptno_pk
    ON dept (deptno);

DROP TABLE IF EXISTS EMP;
DROP TABLE IF EXISTS emp;

CREATE TABLE emp
(
    empno                   INTEGER       NOT NULL UNIQUE
    , ename                 VARCHAR(10)   NOT NULL
    , job                   VARCHAR(9)    NOT NULL

    , mgr                   INTEGER
    , hiredate              DATE          NOT NULL
    , sal                   NUMERIC(7,2)  NOT NULL
    , comm                  NUMERIC(7,2)
    , deptno                INTEGER       NOT NULL

    , CONSTRAINT emp_empno_pk
      PRIMARY KEY (empno)

    , CONSTRAINT emp_mgr_fk
      FOREIGN KEY (mgr)
      REFERENCES emp (empno)

    , CONSTRAINT emp_deptno_fk
      FOREIGN KEY (deptno)
      REFERENCES dept (deptno)
);

CREATE UNIQUE INDEX emp_empno_uindex
    ON emp (empno);
```

- Create table ‘dept’, standing for Department (not being used in API).
- Create table ‘emp’, standing for Employee.

Insert Data

insert.sql

sql > insert.sql

```sql
-- ----------------------------------------------------------------------
--
--          Insert DEPT
--
-- ----------------------------------------------------------------------


INSERT INTO dept VALUES (10,'ACCOUNTING','NEW YORK');
INSERT INTO dept VALUES (20,'RESEARCH','DALLAS');
INSERT INTO dept VALUES (30,'SALES','CHICAGO');
INSERT INTO dept VALUES (40,'OPERATIONS','BOSTON');


-- ----------------------------------------------------------------------
--
--          Insert EMP
--
-- ----------------------------------------------------------------------


INSERT INTO emp VALUES (7839,'KING','PRESIDENT',NULL,'1981-11-17',5000,NULL,10);
INSERT INTO emp VALUES (7698,'BLAKE','MANAGER',7839,'1981-05-01',2850,NULL,30);
INSERT INTO emp VALUES (7782,'CLARK','MANAGER',7839,'1981-06-09',2450,NULL,10);
INSERT INTO emp VALUES (7566,'JONES','MANAGER',7839,'1981-04-02',2975,NULL,20);
INSERT INTO emp VALUES (7654,'MARTIN','SALESMAN',7698,'1981-09-28',1250,1400,30);
INSERT INTO emp VALUES (7499,'ALLEN','SALESMAN',7698,'1981-02-20',1600,300,30);
INSERT INTO emp VALUES (7844,'TURNER','SALESMAN',7698,'1981-09-08',1500,0,30);
INSERT INTO emp VALUES (7900,'JAMES','CLERK',7698,'1981-12-03',950,NULL,30);
INSERT INTO emp VALUES (7521,'WARD','SALESMAN',7698,'1981-02-22',1250,500,30);
INSERT INTO emp VALUES (7902,'FORD','ANALYST',7566,'1981-12-03',3000,NULL,20);
INSERT INTO emp VALUES (7369,'SMITH','CLERK',7902,'1980-12-17',800,NULL,20);
INSERT INTO emp VALUES (7788,'SCOTT','ANALYST',7566,'1982-12-09',3000,NULL,20);
INSERT INTO emp VALUES (7876,'ADAMS','CLERK',7788,'1983-01-12',1100,NULL,20);
INSERT INTO emp VALUES (7934,'MILLER','CLERK',7782,'1982-01-23',1300,NULL,10);
```

- Insert data to ‘dept’ table.
- Insert data to ‘emp’ table.

Delete Data

- Delete data.

delete.sql

sql > delete.sql

```sql
-- ----------------------------------------------------------------------
--
--          Delete data
--
-- ----------------------------------------------------------------------

DELETE FROM dept;
DELETE FROM emp;
```

Tables

database.db

database.db

Search tables... Reset Filters Records: 4

Tables (2)

- dept
- emp

	deptno	dname	loc
1	10	ACCOUNTING	NEW YORK
2	20	RESEARCH	DALLAS
3	30	SALES	CHICAGO
4	40	OPERATIONS	BOSTON

database.db

database.db

Search tables... Reset Filters Records: 14

Tables (2)

- dept
- emp

	empno	ename	job	mgr	hiredate	sal	comm	deptno
1	7369	SMITH	CLERK	7902	1980-12-17	800	NULL	20
2	7499	ALLEN	SALESMAN	7698	1981-02-20	1600	300	30
3	7521	WARD	SALESMAN	7698	1981-02-22	1250	500	30
4	7566	JONES	MANAGER	7839	1981-04-02	2975	NULL	20
5	7654	MARTIN	SALESMAN	7698	1981-09-28	1250	1400	30
6	7698	BLAKE	MANAGER	7839	1981-05-01	2850	NULL	30
7	7782	CLARK	MANAGER	7839	1981-06-09	2450	NULL	10
8	7788	SCOTT	ANALYST	7566	1982-12-09	3000	NULL	20
9	7839	KING	PRESIDENT	NULL	1981-11-17	5000	NULL	10
10	7844	TURNER	SALESMAN	7698	1981-09-08	1500	0	30
11	7876	ADAMS	CLERK	7788	1983-01-12	1100	NULL	20
12	7900	JAMES	CLERK	7698	1981-12-03	950	NULL	30
13	7902	FORD	ANALYST	7566	1981-12-03	3000	NULL	20
14	7934	MILLER	CLERK	7782	1982-01-23	1300	NULL	10

Test API Calls

- GET All Users
- GET Users By Criteria
- GET User
- Delete User
- Create New User
- PUT User
- PATCH User

Fetch All Users

innokentii@dsl-trebng12-b048ab-60:~/Documents/Courses/full-stack-web-development/backend/project

innokentii@dsl-trebng12-b048ab-60:~/Documents/Courses/full-stack-web-developmen... | innokentii@dsl-trebng12-b048ab-60:~/Documents/Courses/full-stack-web-developmen...

```
innokentii@fedora ~/Documents/Courses/full-stack-web-development/backend/project (main) ❯ curl --silent --include  http://localhost:80
80/api/users
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 1630
ETag: W/"65e-N0jJ/LYNZz6qt53dV8ENgZi0yEM"
Date: Sat, 16 Dec 2023 12:33:25 GMT
Connection: keep-alive
Keep-Alive: timeout=5

[{"empno":7369,"ename":"SMITH","job":"CLERK","mgr":7902,"hiredate":"1980-12-17","sal":800,"comm":null,"deptno":20},{"empno":7499,"enam
e":"ALLEN","job":"SALESMAN","mgr":7698,"hiredate":"1981-02-20","sal":1600,"comm":300,"deptno":30},{"empno":7521,"ename":"WARD","job":"
SALESMAN","mgr":7698,"hiredate":"1981-02-22","sal":1250,"comm":500,"deptno":30},{"empno":7566,"ename":"JONES","job":"MANAGER","mgr":78
39,"hiredate":"1981-04-02","sal":2975,"comm":null,"deptno":20},{"empno":7654,"ename":"MARTIN","job":"SALESMAN","mgr":7698,"hiredate":"
1981-09-28","sal":1250,"comm":1400,"deptno":30},{"empno":7698,"ename":"BLAKE","job":"MANAGER","mgr":7839,"hiredate":"1981-05-01","sal"
:2850,"comm":null,"deptno":30},{"empno":7782,"ename":"CLARK","job":"MANAGER","mgr":7839,"hiredate":"1981-06-09","sal":2450,"comm":null
,"deptno":10},{"empno":7788,"ename":"SCOTT","job":"ANALYST","mgr":7566,"hiredate":"1982-12-09","sal":3000,"comm":null,"deptno":20},{"e
mpno":7839,"ename":"KING","job":"PRESIDENT","mgr":null,"hiredate":"1981-11-17","sal":5000,"comm":null,"deptno":10},{"empno":7844,"enam
e":"TURNER","job":"SALESMAN","mgr":7698,"hiredate":"1981-09-08","sal":1500,"comm":0,"deptno":30},{"empno":7876,"ename":"ADAMS","job":"
CLERK","mgr":7788,"hiredate":"1983-01-12","sal":1100,"comm":null,"deptno":20},{"empno":7900,"ename":"JAMES","job":"CLERK","mgr":7698,"
hiredate":"1981-12-03","sal":950,"comm":null,"deptno":30},{"empno":7902,"ename":"FORD","job":"ANALYST","mgr":7566,"hiredate":"1981-12-
03","sal":3000,"comm":null,"deptno":20},{"empno":7934,"ename":"MILLER","job":"CLERK","mgr":7782,"hiredate":"1982-01-23","sal":1300,"co
mm":null,"deptno":10}]innokentii@fedora ~/Documents/Courses/full-stack-web-development/backend/project (main) ❯
```

Fetch Users By Criteria

```
innokentii@dsl-trebng12-b048ab-60:~/Documents/Courses/full-stack-web-development/backend/project
innokentii@dsl-trebng12-b048ab-60:~/Documents/Courses/full-stack-web-developmen...   innokentii@dsl-trebng12-b048ab-60:~/Documents/Courses/full-stack-web-developmen...
innokentii@fedora ~/Documents/Courses/full-stack-web-development/backend/project (main) > curl --silent --include  http://localhost:80
80/api/users?ename=SMITH%2CJOHN&deptno=10%2C20
[1] 282596
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 115
ETag: W/"73-rqtw0W3ONYbH4zELUUuemRJXNf0"
Date: Sat, 16 Dec 2023 12:32:34 GMT
Connection: keep-alive
Keep-Alive: timeout=5

[{"empno":7369,"ename":"SMITH","job":"CLERK","mgr":7902,"hiredate":"1980-12-17","sal":800,"comm":null,"deptno":20}][1]+  Done
          curl --silent --include http://localhost:8080/api/users?ename=SMITH%2CJOHN
innokentii@fedora ~/Documents/Courses/full-stack-web-development/backend/project (main) >
```

Delete User

```
innokentii@dsl-trebng12-b048ab-60:~/Documents/Courses/full-stack-web-development/backend/project
innokentii@dsl-trebng12-b048ab-60:~/Documents/Courses/full-stack-web-developmen...   innokentii@dsl-trebng12-b048ab-60:~/Documents/Courses/full-stack-web-developmen...
innokentii@fedora ~/Documents/Courses/full-stack-web-development/backend/project (main) ❯ curl --silent --include -X DELETE  http://lo
calhost:8080/api/user/7369
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 2
ETag: W/"2-vyGp6PvFo4RvsFtPoIWeCReyIC8"
Date: Sat, 16 Dec 2023 12:34:14 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{}innokentii@fedora ~/Documents/Courses/full-stack-web-development/backend/project (main) ❯
```

Create User

```
innokentii@dsl-trebng12-b048ab-60:~/Documents/Courses/full-stack-web-development/backend/project
innokentii@dsl-trebng12-b048ab-60:~/Documents/Courses/full-stack-web-developmen...   innokentii@dsl-trebng12-b048ab-60:~/Documents/Courses/full-stack-web-developmen...
innokentii@fedora ~/Documents/Courses/full-stack-web-development/backend/project (main) ❯ curl --silent --include -X POST  http://loca
lhost:8080/api/user --header 'Content-Type: application/json' \
--data '{
    "empno": 7369,
    "ename": "SMITH",
    "job": "CLERK",
    "mgr": 7902,
    "hiredate": "1980-12-17",
    "sal": 800,
    "comm": null,
    "deptno": 20
}'
HTTP/1.1 204 No Content
X-Powered-By: Express
ETag: W/"2-vyGp6PvFo4RvsFtPoIWeCReyIC8"
Date: Sat, 16 Dec 2023 12:35:29 GMT
Connection: keep-alive
Keep-Alive: timeout=5

innokentii@fedora ~/Documents/Courses/full-stack-web-development/backend/project (main) ❯
```

Update User (PUT)

innokentii@dsl-trebng12-b048ab-60:~/Documents/Courses/full-stack-web-development/backend/project

innokentii@dsl-trebng12-b048ab-60:~/Documents/Courses/full-stack-web-developmen... | innokentii@dsl-trebng12-b048ab-60:~/Documents/Courses/full-stack-web-developmen...

```
innokentii@fedora ~/Documents/Courses/full-stack-web-development/backend/project (main) ❯ curl --silent --include -X PUT http://localh
ost:8080/api/user/7369 --header 'Content-Type: application/json' \
--data '{
    "empno": 7369,
    "ename": "JOHN",
    "job": "CLERK",
    "mgr": 7902,
    "hiredate": "1980-12-17",
    "sal": 800,
    "comm": null,
    "deptno": 20
}'
HTTP/1.1 204 No Content
X-Powered-By: Express
ETag: W/"2-vyGp6PvFo4RvsFtPoIWeCReyIC8"
Date: Sat, 16 Dec 2023 12:36:24 GMT
Connection: keep-alive
Keep-Alive: timeout=5

innokentii@fedora ~/Documents/Courses/full-stack-web-development/backend/project (main) ❯
```

Update User (PATCH)

```
innokentii@dsl-trebng12-b048ab-60:~/Documents/Courses/full-stack-web-development/backend/project
innokentii@dsl-trebng12-b048ab-60:~/Documents/Courses/full-stack-web-developmen...   innokentii@dsl-trebng12-b048ab-60:~/Documents/Courses/full-stack-web-developmen...
innokentii@fedora ~/Documents/Courses/full-stack-web-development/backend/project (main) ❯ curl --silent --include -X PATCH http://loca
lhost:8080/api/user/7369 --header 'Content-Type: application/json' \
--data '{
    "ename": "SMITH"
}'
HTTP/1.1 204 No Content
X-Powered-By: Express
ETag: W/"2-vyGp6PvFo4RvsFtPoIWeCReyIC8"
Date: Sat, 16 Dec 2023 12:37:27 GMT
Connection: keep-alive
Keep-Alive: timeout=5

innokentii@fedora ~/Documents/Courses/full-stack-web-development/backend/project (main) ❯
```

Summary. Challenges and Solutions

- API is capable of doing CRUD operations with User model. API was developed with applying different techniques. API consists of 7 endpoints.
- No major or significant challenges were faced during the development process.