Testing

EzWh implementation

http://softeng.polito.it

Version 1.0

Type of Testing for EzWh

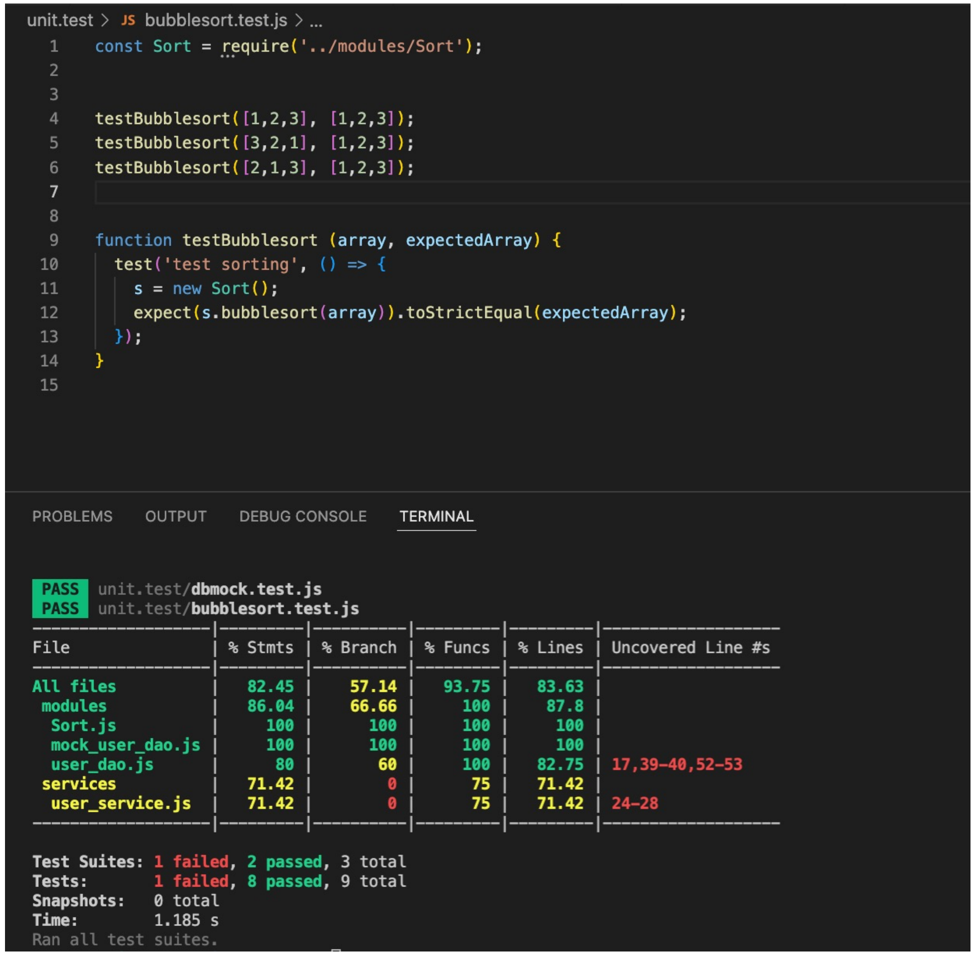
- Unit testing: tests are executed using the class files and not the server
- Integration testing: test are executed using running the `node.js` server and calling the API REST
- E2E testing: the client web app will start working! Also the web client should be tested. However, it is out of the course scope.

Unit Testing

- Using the Jest module
- Adding files named as `module.test.js` inside the `unit_test` folder
- Running them using `npm test`

Unit Testing with Jest

- Create the test
- Run `npm test -- --coverage`
- The command will run all the XXX.test.js file included in the `unit_test` folder



```javascript
const Sort = require('../modules/Sort');


testBubblesort([1,2,3], [1,2,3]);
testBubblesort([3,2,1], [1,2,3]);
testBubblesort([2,1,3], [1,2,3]);


function testBubblesort (array, expectedArray) {
  test('test sorting', () => {
    s = new Sort();
    expect(s.bubblesort(array)).toStrictEqual(expectedArray);
  });
}

```

```
PASS  unit.test/dbmock.test.js
PASS  unit.test/bubblesort.test.js
--------------------|---------|----------|---------|---------|-------------------
File                | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
--------------------|---------|----------|---------|---------|-------------------
All files           |   82.45 |    57.14 |   93.75 |   83.63 |
 modules            |   86.04 |    66.66 |     100 |    87.8 |
  Sort.js           |     100 |      100 |     100 |     100 |
  mock_user_dao.js  |     100 |      100 |     100 |     100 |
  user_dao.js       |      80 |       60 |     100 |   82.75 | 17,39-40,52-53
 services           |   71.42 |        0 |      75 |   71.42 |
  user_service.js   |   71.42 |        0 |      75 |   71.42 | 24-28
--------------------|---------|----------|---------|---------|-------------------

Test Suites: 1 failed, 2 passed, 3 total
Tests:       1 failed, 8 passed, 9 total
Snapshots:   0 total
Time:        1.185 s
Ran all test suites.
```

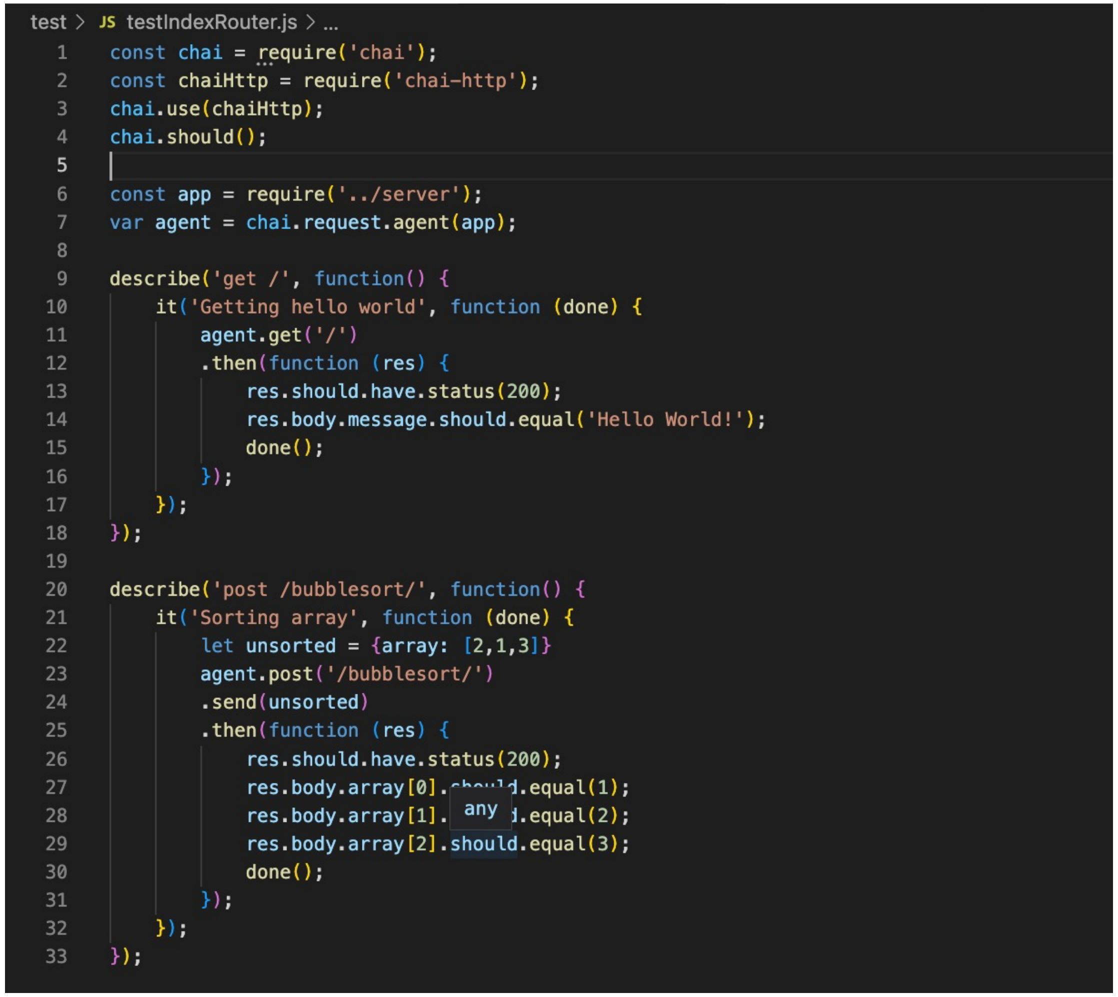
SoftEng http://softeng.polito.it

Integration Testing

- Using the Mocha module
- Adding js files in the `test` folder
- Running them using npm run apiTest

Integration testing with Mocha

- Create the test
- Run `npm run apiTest`
- The command will run all the XXX.js file included in the `test` folder

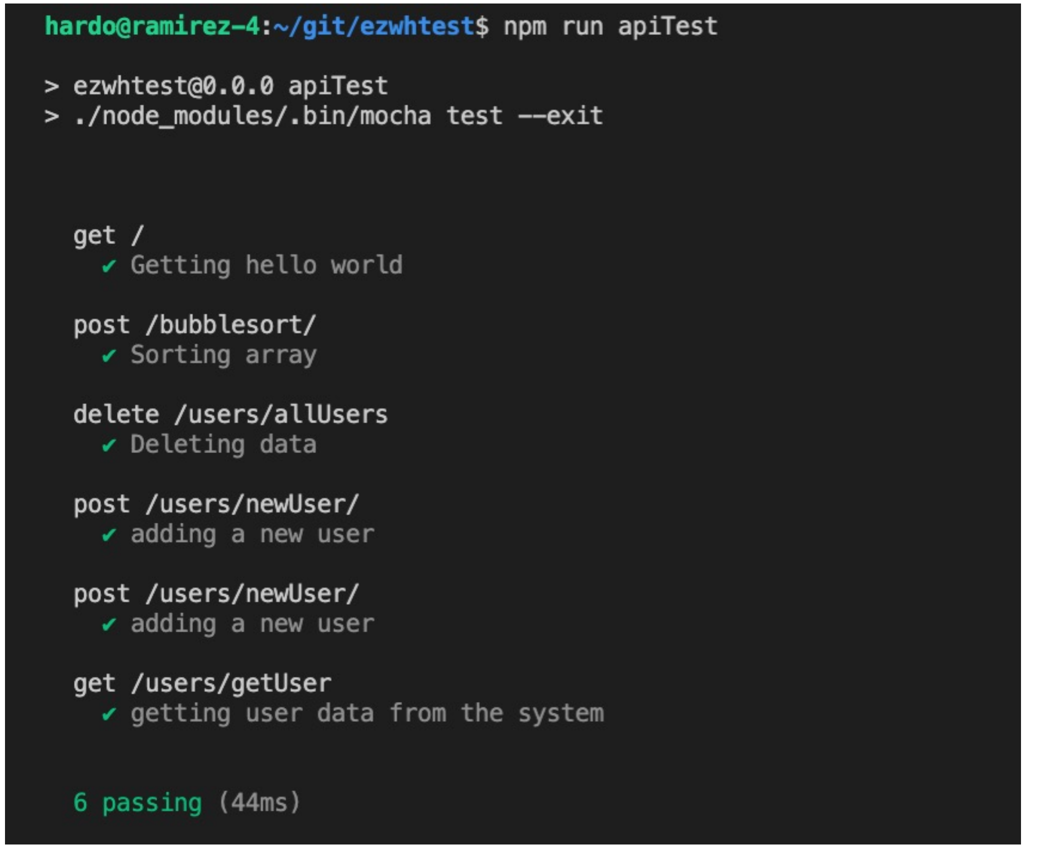


```javascript
const chai = require('chai');
const chaiHttp = require('chai-http');
chai.use(chaiHttp);
chai.should();

const app = require('../server');
var agent = chai.request.agent(app);

describe('get /', function() {
    it('Getting hello world', function (done) {
        agent.get('/')
        .then(function (res) {
            res.should.have.status(200);
            res.body.message.should.equal('Hello World!');
            done();
        });
    });
});

describe('post /bubblesort/', function() {
    it('Sorting array', function (done) {
        let unsorted = {array: [2,1,3]}
        agent.post('/bubblesort/')
        .send(unsorted)
        .then(function (res) {
            res.should.have.status(200);
            res.body.array[0].should.equal(1);
            res.body.array[1].[illegible]d.equal(2);
            res.body.array[2].should.equal(3);
            done();
        });
    });
});
```

SoftEng http://softeng.polito.it

Integration Testing Results

- Mocha will:
  - run the node server
  - call the endpoints (/api/…)
  - Check the results

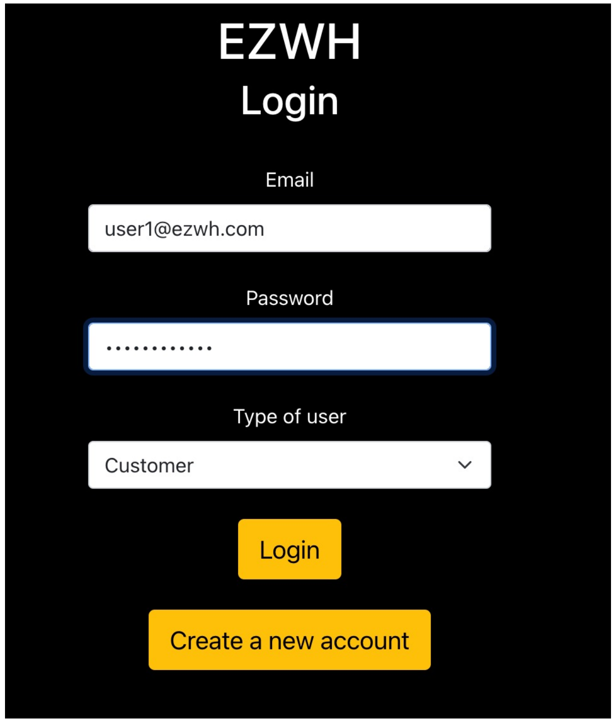


```
hardo@ramirez-4:~/git/ezwhtest$ npm run apiTest

> ezwhtest@0.0.0 apiTest
> ./node_modules/.bin/mocha test --exit



  get /
    ✔ Getting hello world

  post /bubblesort/
    ✔ Sorting array

  delete /users/allUsers
    ✔ Deleting data

  post /users/newUser/
    ✔ adding a new user

  post /users/newUser/
    ✔ adding a new user

  get /users/getUser
    ✔ getting user data from the system


  6 passing (44ms)
```

E2E Testing

- Using the EzWh webapp client
- Once the methods are correctly implemented, the webapp will start working
- **DO NOT** change the API definition, follow the requirement documents

E2E Testing


EZWH

Login

Email

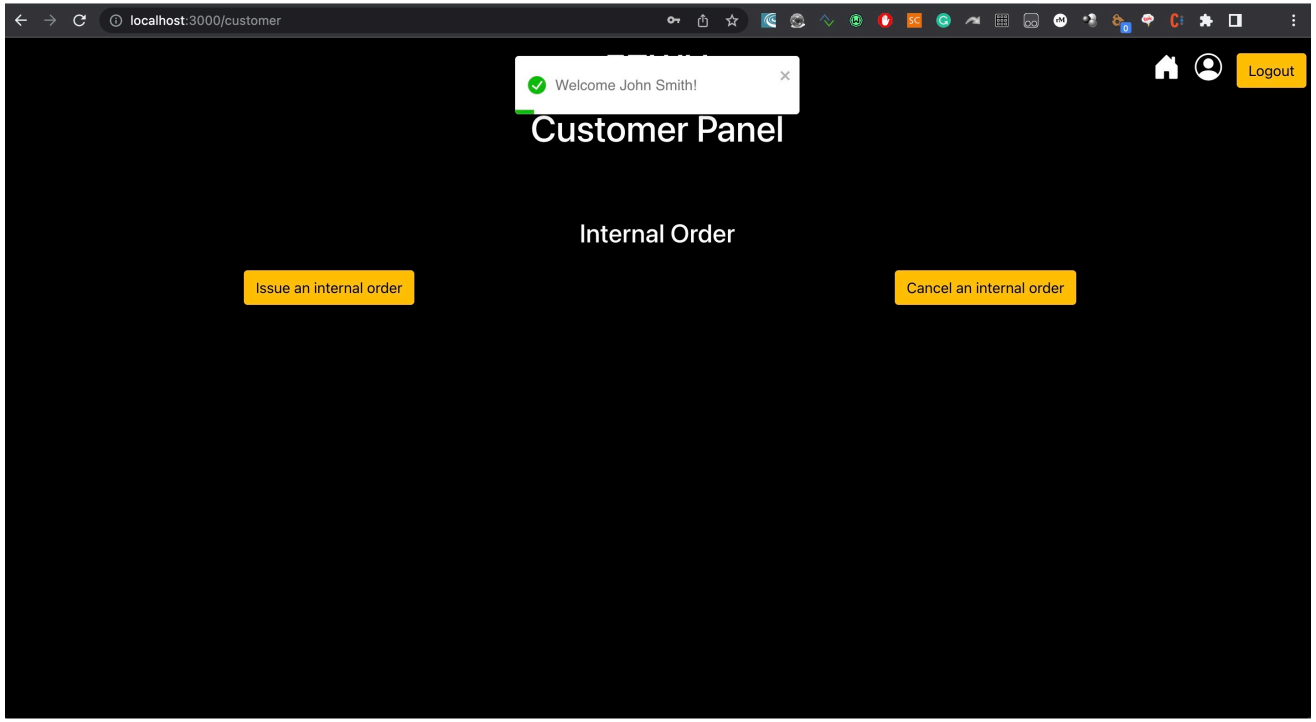
user1@ezwh.com

Password

•••••••••••••

Type of user

Customer

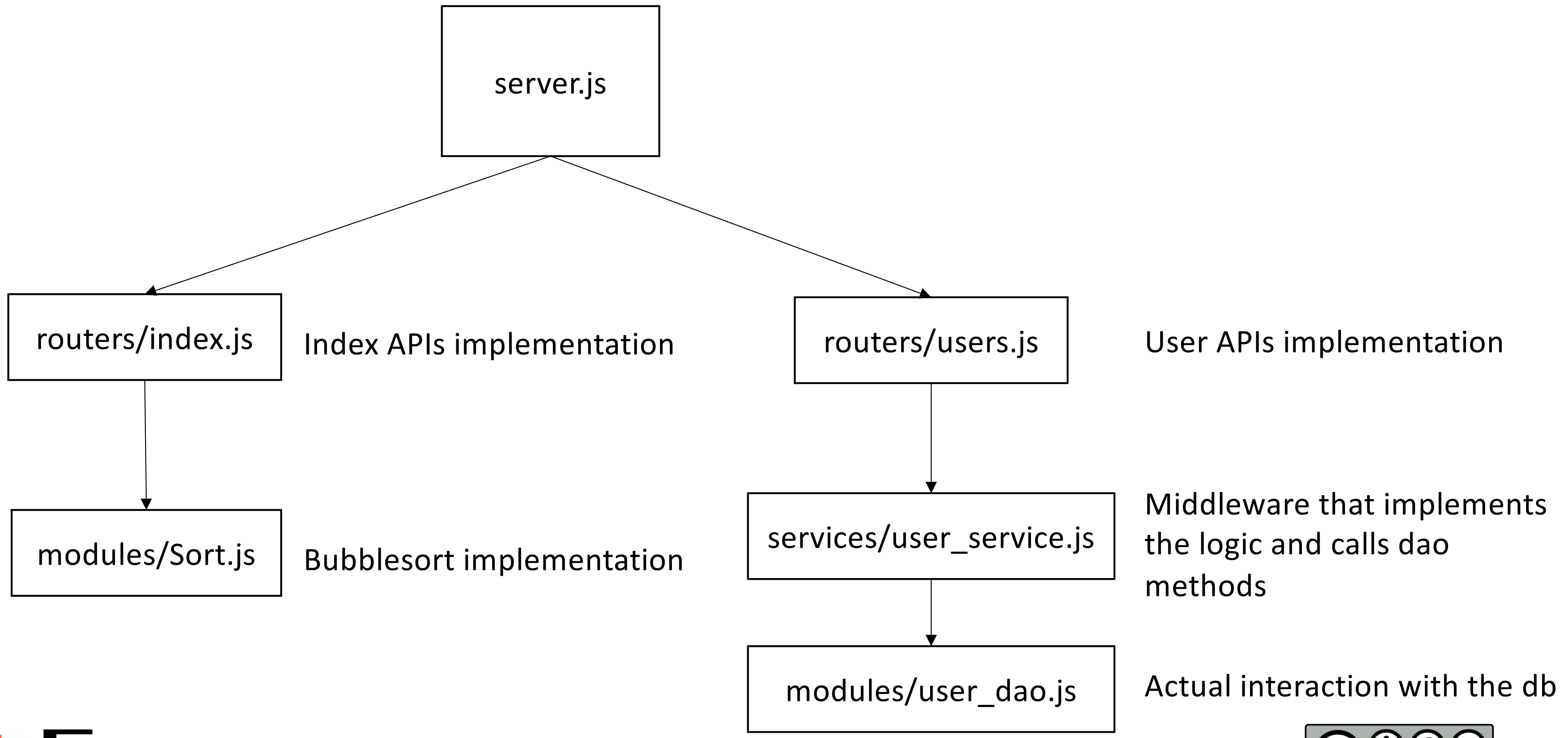

localhost:3000/customer

Welcome John Smith!

Customer Panel

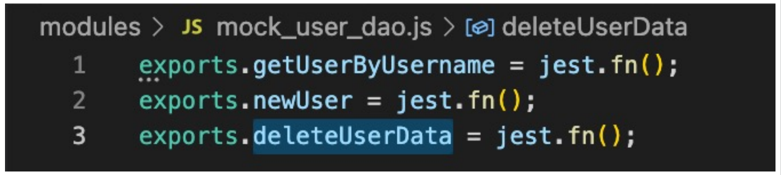
Internal Order

Working Example Software Design


server.js
routers/index.js
Index APIs implementation
routers/users.js
User APIs implementation
modules/Sort.js
Bubblesort implementation
services/user_service.js
Middleware that implements the logic and calls dao methods
modules/user_dao.js
Actual interaction with the db

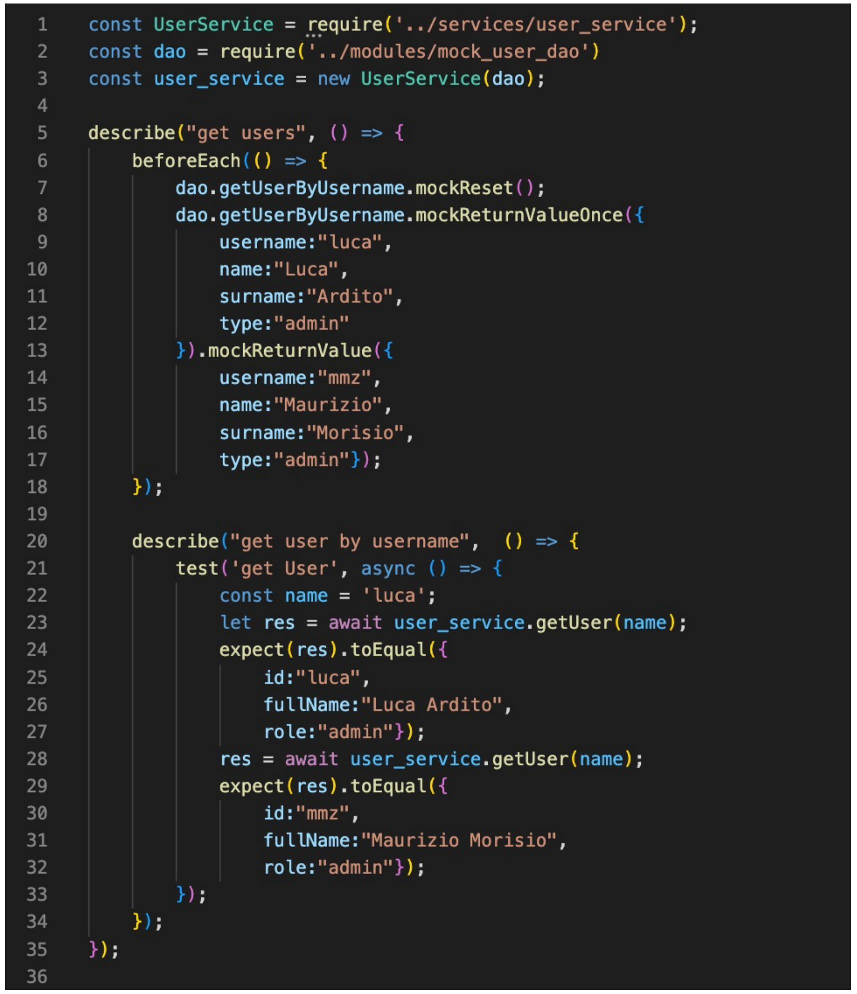
Mocking the DB in Unit Test

- Jest is able to mock the db for us



modules > JS mock_user_dao.js > deleteUserData

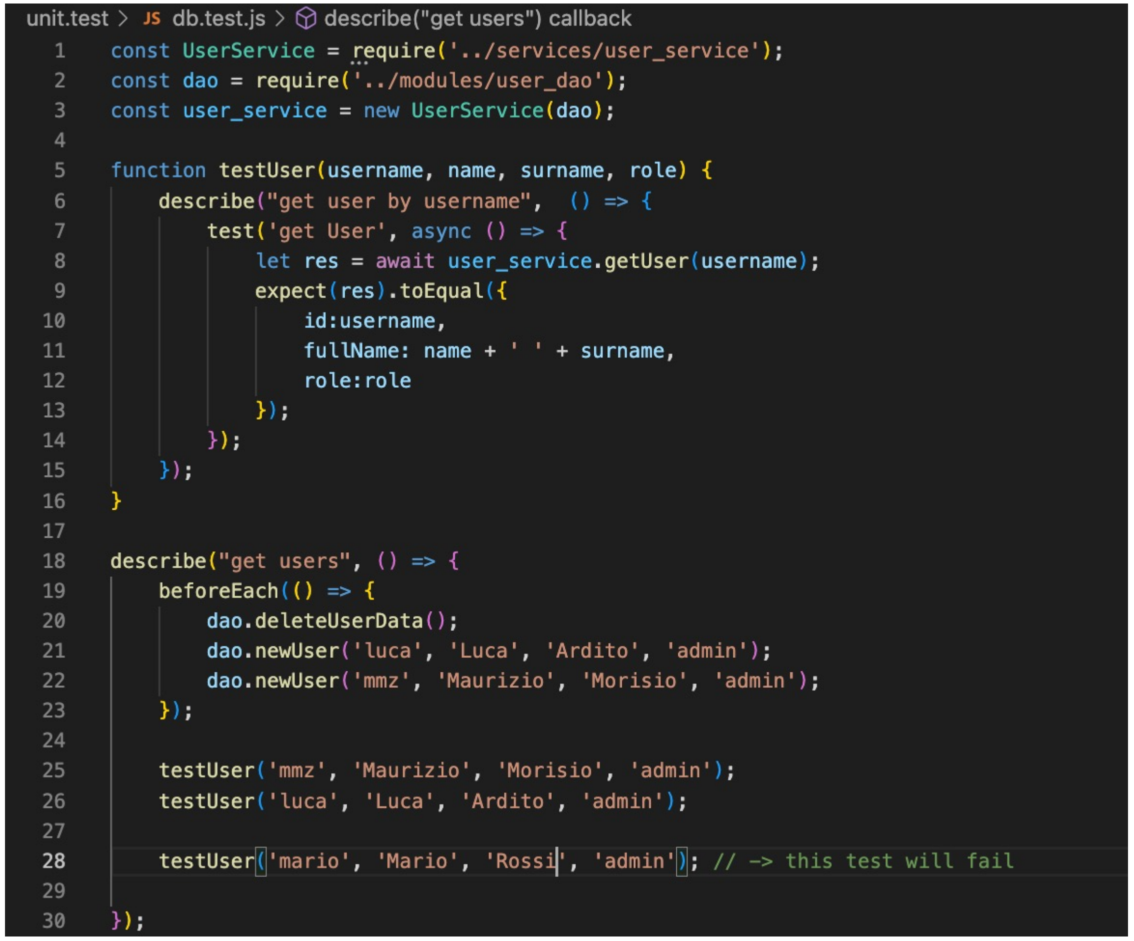
```
exports.getUserByUsername = jest.fn();
exports.newUser = jest.fn();
exports.deleteUserData = jest.fn();
```



```
const UserService = require('../services/user_service');
const dao = require('../modules/mock_user_dao')
const user_service = new UserService(dao);

describe("get users", () => {
    beforeEach(() => {
        dao.getUserByUsername.mockReset();
        dao.getUserByUsername.mockReturnValueOnce({
            username:"luca",
            name:"Luca",
            surname:"Ardito",
            type:"admin"
        }).mockReturnValue({
            username:"mmz",
            name:"Maurizio",
            surname:"Morisio",
            type:"admin"});
    });

    describe("get user by username",  () => {
        test('get User', async () => {
            const name = 'luca';
            let res = await user_service.getUser(name);
            expect(res).toEqual({
                id:"luca",
                fullName:"Luca Ardito",
                role:"admin"});
            res = await user_service.getUser(name);
            expect(res).toEqual({
                id:"mmz",
                fullName:"Maurizio Morisio",
                role:"admin"});
        });
    });
});
```

Persistence Unit Test

- It is required to wait for the data retrieval by using the keyword await



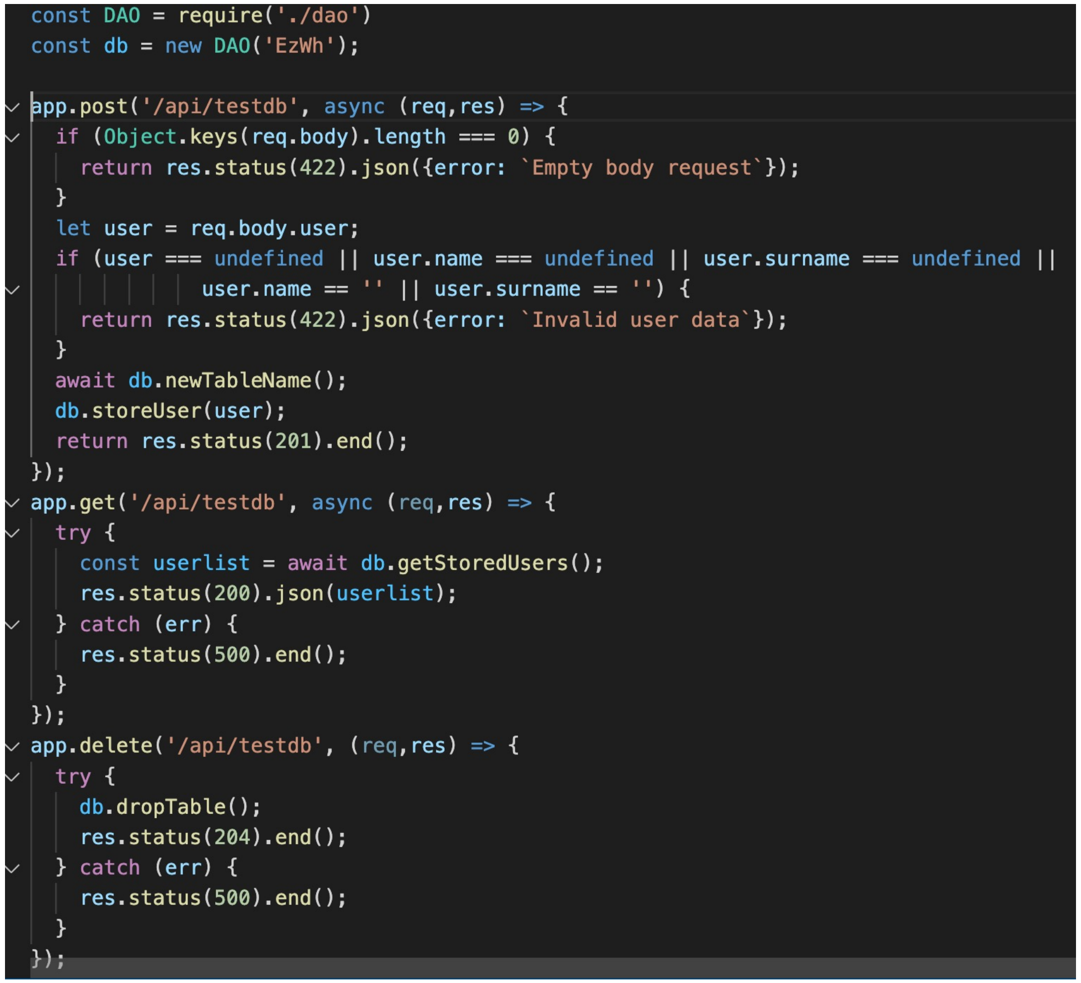
unit.test > db.test.js > describe("get users") callback

```javascript
const UserService = require('../services/user_service');
const dao = require('../modules/user_dao');
const user_service = new UserService(dao);

function testUser(username, name, surname, role) {
    describe("get user by username",  () => {
        test('get User', async () => {
            let res = await user_service.getUser(username);
            expect(res).toEqual({
                id:username,
                fullName: name + ' ' + surname,
                role:role
            });
        });
    });
}

describe("get users", () => {
    beforeEach(() => {
        dao.deleteUserData();
        dao.newUser('luca', 'Luca', 'Ardito', 'admin');
        dao.newUser('mmz', 'Maurizio', 'Morisio', 'admin');
    });

    testUser('mmz', 'Maurizio', 'Morisio', 'admin');
    testUser('luca', 'Luca', 'Ardito', 'admin');

    testUser('mario', 'Mario', 'Rossi', 'admin'); // -> this test will fail

});
```

SoftEng http://softeng.polito.it

Adding the DAO to the node server

- We will create three APIs:
  - GET /users/getUser to read data
  - POST /users/newUser to add data
  - DELETE /users/allUsers to delete all the data stored in the db

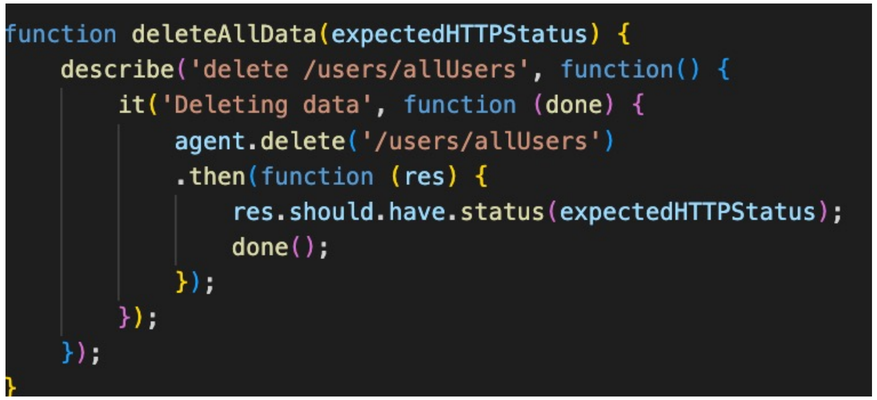


```
const DAO = require('./dao')
const db = new DAO('EzWh');

app.post('/api/testdb', async (req,res) => {
  if (Object.keys(req.body).length === 0) {
    return res.status(422).json({error: `Empty body request`});
  }
  let user = req.body.user;
  if (user === undefined || user.name === undefined || user.surname === undefined ||
          user.name == '' || user.surname == '') {
    return res.status(422).json({error: `Invalid user data`});
  }
  await db.newTableName();
  db.storeUser(user);
  return res.status(201).end();
});
app.get('/api/testdb', async (req,res) => {
  try {
    const userlist = await db.getStoredUsers();
    res.status(200).json(userlist);
  } catch (err) {
    res.status(500).end();
  }
});
app.delete('/api/testdb', (req,res) => {
  try {
    db.dropTable();
    res.status(204).end();
  } catch (err) {
    res.status(500).end();
  }
});
```

SoftEng http://softeng.polito.it

Persistence Integration testing

- It is required to call the three APIs and check if the provided results are those expected

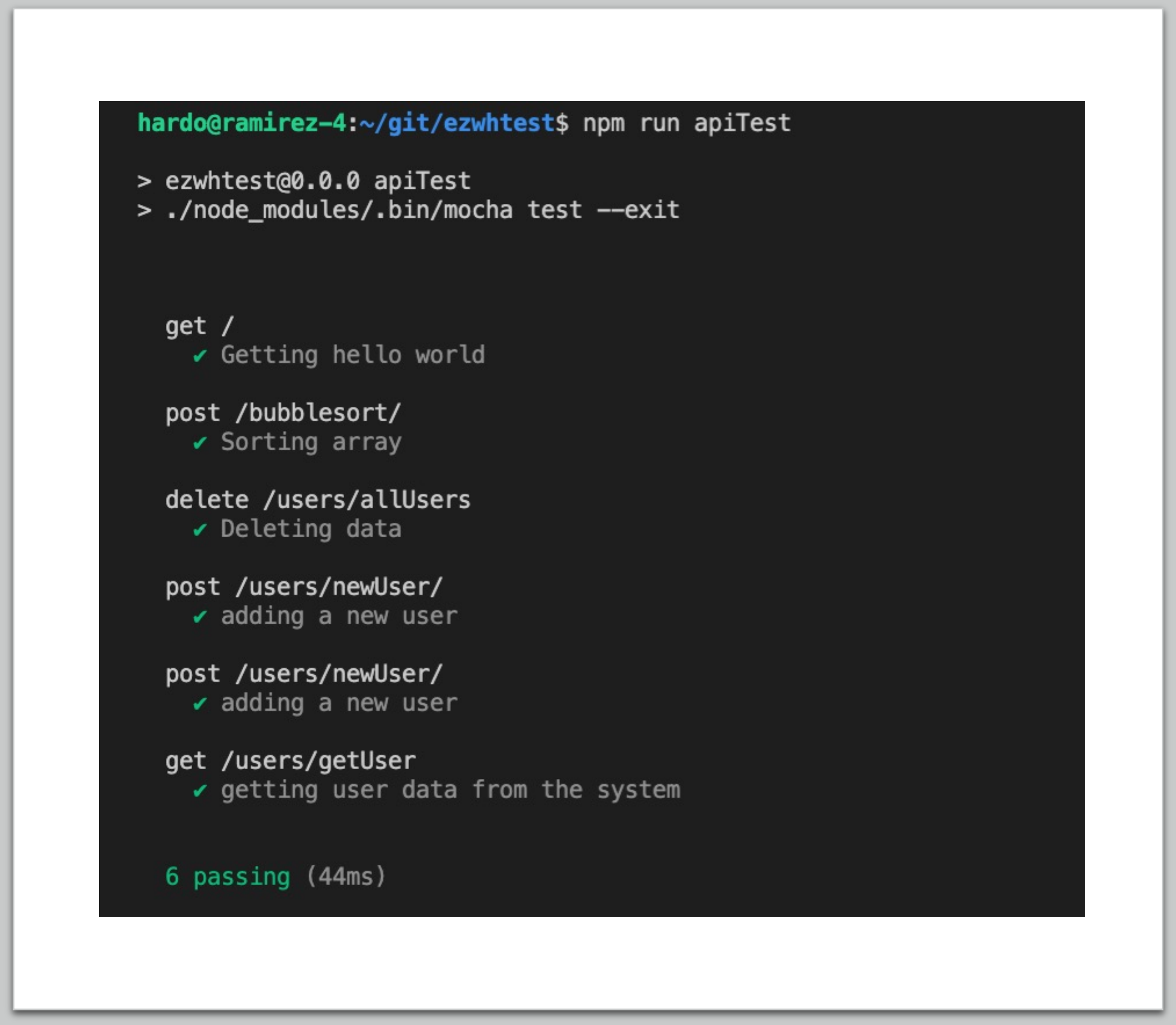


```
function deleteAllData(expectedHTTPStatus) {
    describe('delete /users/allUsers', function() {
        it('Deleting data', function (done) {
            agent.delete('/users/allUsers')
            .then(function (res) {
                res.should.have.status(expectedHTTPStatus);
                done();
            });
        });
    });
}
```

http://softeng.polito.it

Persistence Integration testing results

- `npm run apiTest`



```
hardo@ramirez-4:~/git/ezwhtest$ npm run apiTest

> ezwhtest@0.0.0 apiTest
> ./node_modules/.bin/mocha test --exit



  get /
    ✔ Getting hello world

  post /bubblesort/
    ✔ Sorting array

  delete /users/allUsers
    ✔ Deleting data

  post /users/newUser/
    ✔ adding a new user

  post /users/newUser/
    ✔ adding a new user

  get /users/getUser
    ✔ getting user data from the system


  6 passing (44ms)
```

SoftEng http://softeng.polito.it

Playground Repository

The code presented in these slides is available at this link:

- Starting codehttps://git-softeng.polito.it/d023270/testingjs (master branch)
- Presented code: https://git-softeng.polito.it/d023270/testingjs (testing branch)