Slovenská technická univerzita v Bratislave Fakulta informatiky a informačných technológií

Ilkovičova 2, 842 16, Bratislava 4

METODIKA TESTOVANIA

BlockPay
Blockchain Busters

Tímový projekt
Tím č. 20
Vedúci: Ing. Kristián Košťál
fiittim201920@gmail.com

Vypracoval: Lenka Koplíková,
Veronika Klaciková

1 Vymedzenie obsahu

Tento dokument popisuje postupy a pravidlá testovania napísaného kódu v tíme Blockchain Busters.

2 Dedikácia metodiky

Metodikou sa riadia členovia tímu Blockchain Busters, v rozsahu predmetu Tímový projekt.

3 Písanie testov pre backend

Pri písaní testov treba brať ohľad na:

- dĺžku vykonania testu (čo najkratšia)
- otestovanie čo najväčšieho množstva možných scenárov
- hraničné prípady

Unit testy by mali testovať práve jednu funkcionalitu. Dáta pri týchto testoch môžu byť mockované. Integračné testy by mali overovať kompatibilitu viacerých komponentov. Tieto testy by mali pracovať s reálnymi dátami.

Písanie testov by nemalo zabrať viac ako 30% času vývoja produktu.

3.1 Unit testy

Na vytváranie unit testov sa používajú frameworky:

- `jest` (dokumentácia)
- `superjest` (dokumentácia)

Sady testov (angl. test suite) sa umiestňujú do priečinka `backend/tests` s príponou `.test.js`. Sada testov môže obsahovať viacero testov, ktoré spolu súvisia.

Unit testy sa píšu vo formáte:

```
describe('<názov sady testov>', () => {
  it('<výstižný opis testu>', () => {
    <očakávania (angl. assertions)>
  })
})
```

Ukážkové príklady:

sample.test.js

```
describe('sample test', () => {
  it('should test that true === true', () => {
    expect(true).toBe(true)
  }),
  it('should test that 1 + 2 is equal to 3', () => {
  expect(1+2).toBe(3);
  });
})
```

registrationAPI.test.js
(testuje API pre registráciu, kód pozri nižšie)

```
describe('registration test', () => {

  afterAll(() => {
    return deleteUser("testuser", db_conf.db_test);
  });

  afterAll(function(done) {
    server.close(done); //zastaví server po testoch, aby zbehli automatizované
testy
  });

  //console.log(typeof db);
  it('should create a new user', async () => {
    const res = await request(server) //dolezite ze tu je server a nie app
      .post('/api/registration')
      .send({
        username: "testuser",
        password: "testpassword",
        dbS: "test",      })
    expect(res.statusCode).toEqual(200)
    expect(res.text).toBe('{\"value\":\"success\"}')
  })

})
```

server.js - API pre registráciu

```
//method to receive data from client
app.route('/api/registration').post((req, res) => {
  console.log('Request of registration accepted!');
  var username = req.body.username;
  var password = req.body.password;

  (async () => {
    // check whether is specific user already in database
    // if he is, return fail for new user registration
    // if he is not, add new user to database and return success
    if (await findUser(username)) {
      console.log("User already exists!");
      res.send(JSON.stringify({
        value: 'fail'
      }));
    } else {
      addUser(username, password);
      // console.log("User added!");
      res.send(JSON.stringify({
        value: 'success'
      }));
    }
  })();
});
```

Každý test musí v sebe obsahovať časť kódu na zastavenie behu servera, inak neprebehnú automatizované testy.

```
afterAll(function(done) {
    server.close(done);
  });
```

Tak isto pri volaní metód sa pri request nepoužíva app, ale server.

4 Písanie testov pre frontend

Pre frontend sa testy nepíšu. Ich funkcionalita je definovaná možnosťou načítania stránky.

5 Spúšťanie testov

Testy sa vykonávajú počas code review, ako je definované v dokumente Metodika code review.

Testy sa spúšťajú príkazom `npm test` v priečinku `backend`. Testy sa taktiež spúšťajú pomocou `CI` pri každom pushnutí na vetvu `develop`.