Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Kolaboratívne prototypovanie

[CollabUI]

Metodika testovania

Tím: 24. CollabUI
Vedúci tímu: Ing. Eduard Kuric, PhD.
Členovia tímu: Adrián Nagy, Lukáš Vrba, Ján Kleň, Michal Melúch, Tomáš Mňačko, Peter Písecký, Miloslav Smetana
Študijný program: Inteligentné softvérové systémy
Ročník: 2017/2018

1 Úvod

Táto metodika je určená pre členov tímu číslo 24 v akademickom roku 2016/2017 a obsahuje postupy a pravidlá pre testovanie webovej aplikácie na kolaboratívne prototypovanie. Metodika pozostáva z opisu oboch jazykov, ktoré sú použité v tomto projekte spolu s prikázanými konvenciami a návodmi ako spúšťať a ukladať testy.

2 Skratky a značky

Mock	simulovaný objekt, ktorý napodobňuje správanie skutočného objektu
Fixture	mockovací objekt používaný knižnicou PhpUnit
JS	JavaScript
Commit	zmeny, ktoré vykonal člen tímu v projekte a označil ich komentárom

3 Postupy

Táto kapitola predpisuje spôsob, ktorým sa pri testovaní riadia všetci členovia tímu, konkrétne upresňuje spôsob ukladania testov, ich spúšťanie a pravidlá písania testov. Testy sú vykonávané v dvoch jazykoch a to: Javascript a Php. Každý jazyk má svoje knižnice a pravidlá, rovnako ako aj miesto uloženia. Tieto informácie sú uvedené v podkapitolách nižšie.

3.1 Jazyk PHP

PhpUnit je knižnica, ktorá sa musí používať v prípade ak je potrebné vytvoriť testy pre jazyk PHP. Integračné testy je nutné vytvárať práve pomocou tejto knižnice, rovnako ako testy týkajúce sa databázy.

3.1.1 Konvencie

Na testovanie databázy je vždy potrebné vytvoriť mocky jednotlivých tabuliek pomocou objektov nazývaných Fixture v knižnici PhpUnit. Súbory, ktoré sú použité ako Fixture musia byť nazvané vo formáte NázovtabulkyFixture.php, napríklad UsersFixture.php. Na obrázku 1 je príklad použitia Fixture v jazyku PHP.
Každý test v PhpUnit musí byť definovaný ako funkcia, ktorej názov začína kľúčovým slovom test a potom opisom toho, čo testuje, napríklad *testLogInValid().* Na obrázku 2 je uvedený príklad testu *testLogInValid*.

```php
class UsersFixture extends TestFixture
{
    // Import table definition: specify its name and connection to be used (see app.php -> connections)
    public $import = ['table' => 'users', 'records' => false, 'connection' => 'default'];

    public function init()
    {
        $this->records = [
            [
                'first_name' => 'Test',
                'last_name' => 'Test-Surname',
                'email' => 'testuser@email.com',
                'password' => (new DefaultPasswordHasher)->hash(
                    '12345'
                ),
                'token' => 'password',
                'settings' => null,
                'status' => 2
            ]
        ];
        parent::init();
    }

}
```

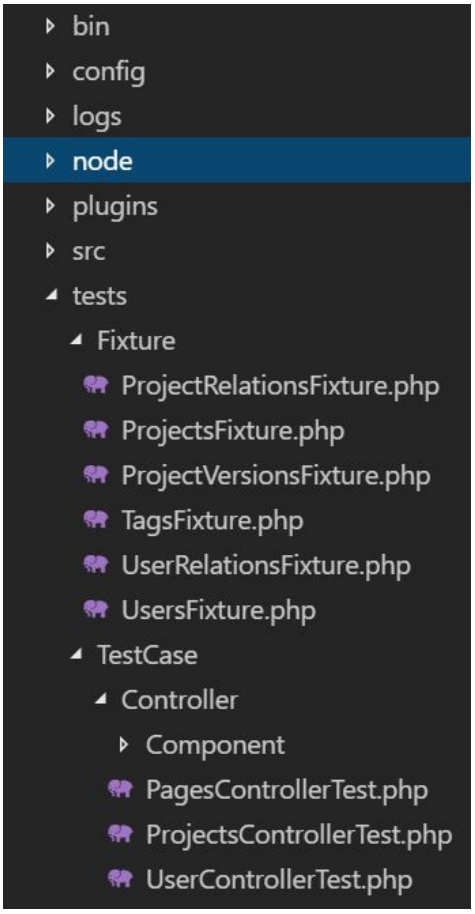
Obr. 1 - príklad mockovania tabuľky Users

```php
public function testLogInValid()
{
     // Load a user from the fixture dummy data
    $user = $this->Users->find('all')->first();
        var_dump($user);
    $data = [
        'email' => 'testuser@email.com',
        'password' => '12345'
    ];
    // Perform the password reset request
    $this->post('/log-in', $data);

    // Look for a 2xx/3xx response code
    $this->assertResponseSuccess();

    // Check the redirect to homepage after a successful request
    $this->assertRedirect('/dashboard');

    $this->get('/log-out');

    $this->assertResponseSuccess();

    $this->assertRedirect('/');
}
```

Obr. 2 - test správneho prihlásenia a presmerovania

3.1.2 Úložisko testov

Testy určené pre jazyk PHP sa musia ukladať do zložky tests\TestCase. A mocky, ktoré reprezentujú tabuľky musia byť umiestnené v zložke tests\Fixture. Na obrázku 3 je ukázaný príklad umiestnenia testov a mockov v jazyku PHP.



Obr. 3 - miesto uloženia PHP testov

3.1.3 Spúšťanie testov

Testy sa musia spúšťať pred každým commitom do ktorejkoľvek vetvy. Každý člen tímu je zodpovedný za to, aby testy po jeho práci boli úspešné.
PHP testy sa spúšťajú pomocou zadania príkazu *phpunit* do konzoly v root adresári, v prípade, že je potrebné vykonať konkrétny test, ako prvý argument sa zadá cesta ku testu, napríklad *phpunit tests\TestCase\UserControllerTest.php*

3.2 Jazyk JavaScript

Pre jazyk JavaScript boli určené knižnice Mocha na vytváranie a overovanie JS testov a knižnica Sinon, ktorá je použitá na overovanie správnych volaní funkcií a ich argumentov.

3.2.1 Konvencie

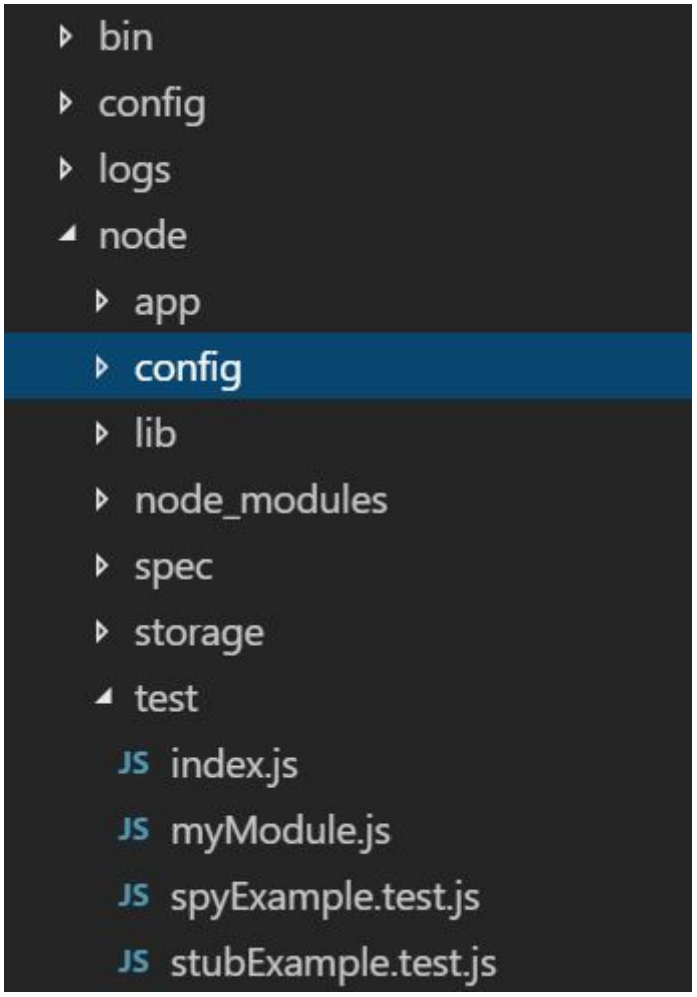
Pri vytváraní testov v jazyku JS sa rozďeluje test na 3 časti a to: given, when, then. V časti given sa inicializujú dáta a premenné, v časti when sa vykonáva logika, ktorá má byť otestovaná, konkrétne sa volajú sa funkcie a v poslednej časti then sa overuje korektné správanie funkcií.
Testy sa vždy nazývajú názovsúboru.test.js v opačnom prípade nebude vedieť skript nájst a vykonať testy.
Príklad testu a konvencií je uvedený na obrázku 4.

```
const sinon = require('sinon');
const myModule = require('./myModule');
const assert = require('assert');

// spy allows us to get information like number of calls,
// what arguments they received and so on
describe('spy example', function () {

it('calls the original function once', function() {
    // given
    var callback = sinon.spy()
    var proxy = myModule.once(callback);

    //when
    proxy();
    proxy();

    //then
    assert(callback.calledOnce);
});
```

Obr. 4 - príklad konvencií JS testov

3.2.2 Úložisko testov JavaScript

Testy pre JS musia byť uložené v zložke node\test a nazvané podľa konvencie uvedenej v časti 3.2.1. Testy, ktoré sú umiestnené v inej zložke nebudú vykonané. Príklad umiestnenia testov je na obrázku 5.



Obr. 5 - príklad umiestnenia JS testov

3.2.3 Spúštanie testov

Testy sa musia spúštať pred každým commitom do ktorejkoľvek vetvy. Každý člen tímu je zodpovedný za to, aby testy po jeho práci boli úspešné.
JS testy sa spúštajú zo zložky node zadaním príkazu *npm run test* do konzoly