МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»
Інститут комп’ютерних систем
Кафедра інформаційних систем

Звіт

Лабораторна робота № 4

З дисципліни «Проектний підхід до розробки ПЗ»

**Тема: «Основи unit-testing в Angular»**

Виконав:
студент групи УІ-191
Кисельов Д.С.

Перевірив:
Малерик Р.П.

**Одеса – 2021**

**Тема:** Основи unit-testing в Angular.

**Мета лабораторної роботи:**

1) вивчити теоретичні основи модульного тестування;
2) вивчити принципи та підходи до написання модульних тестів;
3) розглянути інструменти для модульного тестування в Angular;
4) реалізувати типові приклади модульних тестів в Angular.

**Структура протоколу до лабораторної роботи:**

1. Завдання на лабораторну роботу.
2. Результати покриття тестами.
3. Повні версії звітів та статистики про покриття тестами.
4. Висновки про виконану роботу.

## 1. Завдання на лабораторну роботу

1. Вивчіть основні принципи написання модульних тестів.

2. Вивчіть приклади тестів для Service і Component.

3. Покрийте тестами 2 існуючі Component і 1 Service.

4. Наведіть у протоколі скріншоти звіту про покриття тестами для кожного тестованого класу.

## 2. Результати покриття тестами

### 2.1 Сервіс AuthService

#### 2.1.1 Код з файлу auth.service.spec.ts

```
import { HttpClient } from '@angular/common/http';
import { TestBed } from '@angular/core/testing';
import { Router } from '@angular/router';
import { AuthService } from '../services/auth.service';
```

```
describe('AuthService', () => {
  let service: AuthService;
  let router: Router;
  let http: HttpClient;

  const token = 'token';

  beforeEach(() => {
    http = jasmine.createSpyObj(['get']);
    router = jasmine.createSpyObj(['navigate']);
    service = new AuthService(http, router);
  });

  // checking for service to be created
  it('should be created', () => {
    expect(service).toBeTruthy();
  });

  // checking for service to redirect to login page after logout
  it('should redirect to login page on logout', () => {
    service.logout();
    expect(router.navigate).toHaveBeenCalledWith(['/login']);
    expect(router.navigate).toHaveBeenCalledTimes(1);
  });

  // checking if the token is returned from localstorage
  it('should return token from localstorage', () => {
    spyOn(localStorage, 'getItem').and.returnValue(token);
    const realToken = service.getToken();
    expect(realToken).toEqual(token);
  });

  // checking if the token is saved in localstorage
```

```
  it('should save token in localstorage', () => {
   spyOn(localStorage, 'setItem');
   service.saveToken(token);
   expect(localStorage.setItem).toHaveBeenCalledWith('access_token', token);
  });
});
```

### 2.1.2 Звіт про покриття тестами AuthService

Під час написання тесту для даного сервісу було виконано перевірку наступного функціоналу:

1) екземпляр сервісу повинен успішно створюватися;

2) сервіс повинен зберігати токен у локальному сховищі;

3) сервіс повинен повертати токен з локального сховища;

4) повинне виконуватись перенаправлення на сторінку з логіному у випадку виходу із сессії;

Такий вигляд має результат тестування у браузері:

```
AuthService
  • should save token in localstorage
  • should be created
  • should return token from localstorage
  • should redirect to login page on logout
```

## 2.2 Компонент LoginComponent

### 2.2.1 Код з файлу login.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { FormBuilder, FormGroup } from '@angular/forms';
import { AuthService } from '../services/auth.service';
```

```
import { LoginComponent } from './login.component';

describe('LoginComponent', () => {
  let login: LoginComponent;
  const fb = new FormBuilder();
  let authService: AuthService;
  let fixture: ComponentFixture<LoginComponent>;
  const token = 'token';

  beforeEach(async () => {
    authService = jasmine.createSpyObj('AuthService', ['getToken']);
    await TestBed.configureTestingModule({
      declarations: [ LoginComponent ],
      providers: [
        { provide: AuthService, useValue: authService },
        { provide: FormBuilder}
      ]
    })
    .compileComponents();
  });

  beforeEach(() => {
    login = new LoginComponent(fb, authService);
    login.ngOnInit();
    fixture = TestBed.createComponent(LoginComponent);
    login = fixture.componentInstance;
    fixture.detectChanges();
  });

  // testing if the component is created
  it('should create', () => {
    expect(login).toBeTruthy();
  });
```

```
  // testing the validity of a form if it is not filled
  it('empty form should be invalid', () => {
    expect(login.form.valid).toBeFalsy();
  });

  // testing the validity of a login field
  it('form with empty login field should be invalid', () => {
    const loginField = login.form.controls.login;
    expect(loginField.valid).toBeFalsy();
  });

  // testing the validity of a password field
  it('form with empty password field should be invalid', () => {
    const passwordField = login.form.controls.password;
    expect(passwordField.valid).toBeFalsy();
  });

 // testing if the title is rendered in a correct tag
  it('should render title in a h1 tag', () => {
    const compiled = fixture.debugElement.nativeElement;
    expect(compiled.querySelector('h1').textContent).toContain('Welcome!');
});
});
```

### 2.2.2 Звіт про покриття тестами LoginComponent

Під час написання тесту для даного компоненту було виконано перевірку наступного функціоналу:

1) екземпляр класу повинен успішно створюватися;

2) порожня форма не повинна бути валідною;

3) форма з порожнім полем логіну не повинна бути валідною;

4) форма з порожнім полем паролю не повинна бути валідною;

5) форма повинна мати компонент із текстом “Welcome”, загорнутим у тег h1 ;

Такий вигляд має результат тестування у браузері:

```
LoginComponent
  • form with empty login field should be invalid
  • should render title in a h1 tag
  • form with empty password field should be invalid
  • should create
  • empty form should be invalid
```

## 2.3 Компонент TableComponent

### 2.3.1 Код з файлу login.component.spec.ts

```
import { HttpClient } from '@angular/common/http';
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { Router } from '@angular/router';
import { Observable, of } from 'rxjs';
import { Order } from '../model/order.model';
import { AuthService } from '../services/auth.service';
import { OrdersService } from '../services/orders.service';
import { TableComponent } from './table.component';

describe('TableComponent', () => {
  let table: TableComponent;
  let ordersService: OrdersService;
  let authService: AuthService;
  let router: Router;
  // let fixture: ComponentFixture<TableComponent>;
  const orders = [
    {name: 'Order_test', category: 'Category_test', price: 22 }
  ];

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ TableComponent ],
```

```
      providers: [
        { provide: HttpClient, useValue: HttpClient }
      ]
    })
    .compileComponents();
  });

  beforeEach(() => {
    ordersService = jasmine.createSpyObj(['getAll']);
    authService = jasmine.createSpyObj(['logout']);
    router = jasmine.createSpyObj(['navigate']);
    table = new TableComponent(ordersService, authService);
    const observable: Observable<Order[]> = of(orders);
    ordersService.getAll = jasmine.createSpy().and.callFake(() => {
      return observable;
    });
    table.ngOnInit();
  });

  // checking if the table component is created
  it('should create', () => {
    expect(table).toBeTruthy();
  });

  // checking if the orders are displayed correctly
  it('should display received orders', () => {
      expect(table.orders).toEqual(orders);
  });
});
```

### 2.3.2 Звіт про покриття тестами TableComponent

Під час написання тесту для даного компоненту було виконано перевірку наступного функціоналу:

1) екземпляр класу повинен успішно створюватися;

2) таблиця у файлі має успішно відтворювати отримані дані.

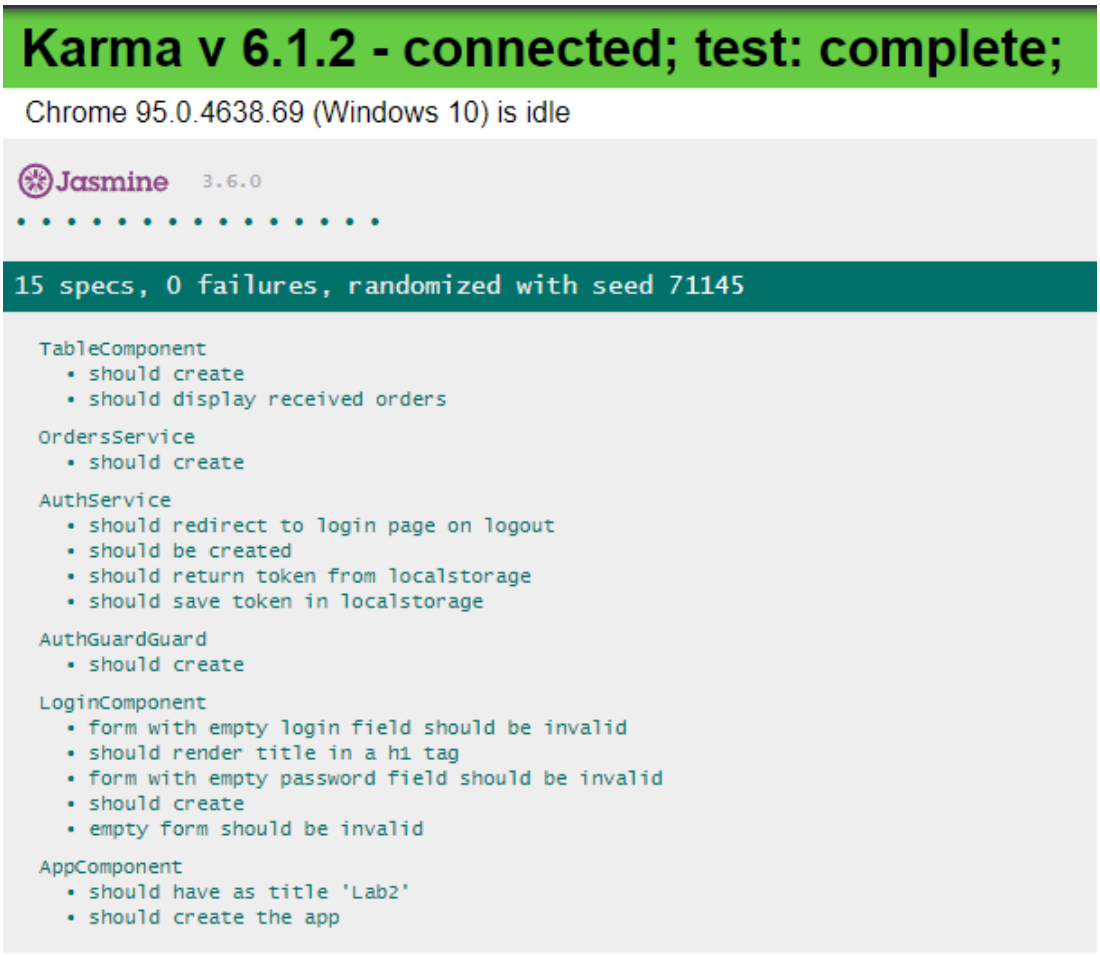
Такий вигляд має результат тестування у браузері:

```
TableComponent
  • should create
  • should display received orders
```

## 3. Повні версії звітів та статистики про покриття тестами

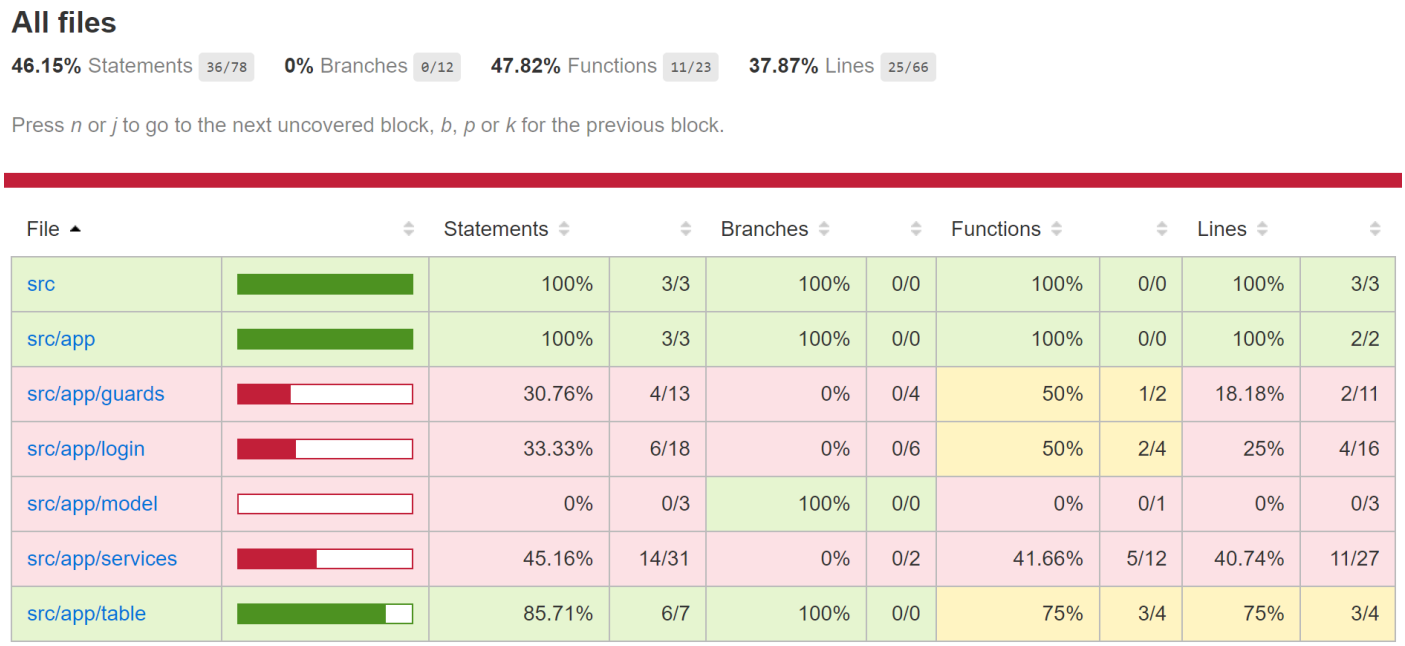
Окрім двох представлених компонентів та сервісу під час виконання роботи були налагоджені стандартні тести для інших компонентів та сервісів проекту (тобто тестування того, що екземпляр класу успішно створюється).

Повний звіт про тести від тест-ранеру Karma у бразуері представлено на скриншоті нижче:



Окрім цього, було переглянуто статистику про покриття тестами сервісів та компонентів у проекті.

Файл із статистикою знаходиться у папці ./coverage/index.html.



**All files**

**46.15%** Statements 36/78 **0%** Branches 0/12 **47.82%** Functions 11/23 **37.87%** Lines 25/66

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

| File | Statements | | Branches | | Functions | | Lines | |
|---|---|---|---|---|---|---|---|---|
| src | 100% | 3/3 | 100% | 0/0 | 100% | 0/0 | 100% | 3/3 |
| src/app | 100% | 3/3 | 100% | 0/0 | 100% | 0/0 | 100% | 2/2 |
| src/app/guards | 30.76% | 4/13 | 0% | 0/4 | 50% | 1/2 | 18.18% | 2/11 |
| src/app/login | 33.33% | 6/18 | 0% | 0/6 | 50% | 2/4 | 25% | 4/16 |
| src/app/model | 0% | 0/3 | 100% | 0/0 | 0% | 0/1 | 0% | 0/3 |
| src/app/services | 45.16% | 14/31 | 0% | 0/2 | 41.66% | 5/12 | 40.74% | 11/27 |
| src/app/table | 85.71% | 6/7 | 100% | 0/0 | 75% | 3/4 | 75% | 3/4 |

## 4. Висновки про виконану роботу

Під час виконання даної лабораторної роботи було проведено дії, пов’язані із практичним засвоєнням основних принципів роботи Unit-тестування у Angular. Після виконання завдань даної лабораторної роботи можна зробити висновок, що головне призначення модульних тестів - перевірка коректності роботи окремих частин програми у ізольованому середовищі незалежно одне від одного. Тобто, дуже важливою особливість модульного тестування полягає є те, що перевірка окремого компоненту повинна проводитися в ізоляції від інших компонентів застосунку. Всі залежності тестованого класу, повинні бути підмінені фейковими реалізаціями (mocks або stubs).

При написанні unit-тестів в даній роботі було використано засоби та інструменти із базового пакету Angular для написання та запуску модульних тестів:

1) Karma – засіб для запуску та прогонки тестів (test-runner);
2) Jasmine – фреймворк, що надає методи перевірок та налаштування фейкових об'єктів;
3) AngularTestBed – засіб для тестування компонентів фреймворку Angular.