

**Софтверски квалитет и тестирање**

Проектна задача на тема –

**Selenium,  
Spring MVC, Mutation,  
Input space coverage и  
Load тестирање на  
Spring Boot веб  
апликација**

Изработил: Благој Петков 191097

# 1. Вовед

Целта на оваа проектна задача е тестирање на Spring Boot апликација, со помош на Selenium Framework, Spring MVC Test Framework, Mutation testing, Input Space Coverage како и Load testing со помош на JMeter.

# 2. Код кој што се тестира

Линк до GitHub репозиториум каде што е поставен целиот код од апликацијата како и тестовите:

[https://github.com/blagoipetkov/skit\\_proekt\\_phalaenopsis](https://github.com/blagoipetkov/skit_proekt_phalaenopsis)

# 3. Алатки

Алатки кои што се користат во проектната задача:

## Selenium Framework

Се користи за целосно тестирање на сите погледи кои што ги нуди апликацијата, целиот интерфејс, регистрација и најава на корисник, добивање и преземање на сертификат за покажаното знаење во апликацијата како и тестирање на внесување на точни како и погрешни податоци во полињата за внес на податоци.

## Spring MVC Test Framework

Се користи за интеграциско тестирање на контролерите, на сите погледи кои што ги враќа апликацијата, дали се вистинските погледи кои што треба да бидат вратени, со соодветниот статус код како и дали погледите се вратени со вистинските атрибути. Во случај доколку корисникот се пренасочува кон друг метод се тестира дали пренасочувањето е кон вистинскиот метод.

## PIT Mutation Testing System, Mockito

Се користи за мутациско тестирање на сервисната логика на апликацијата, поточно на трите сервиси кои што се содржани во апликацијата, CertificateServiceImpl, TopicServiceImpl и UserServiceImpl. При тестирањето исто така е користен и Mockito за креирање на mock објекти.

## Input space coverage with Junit

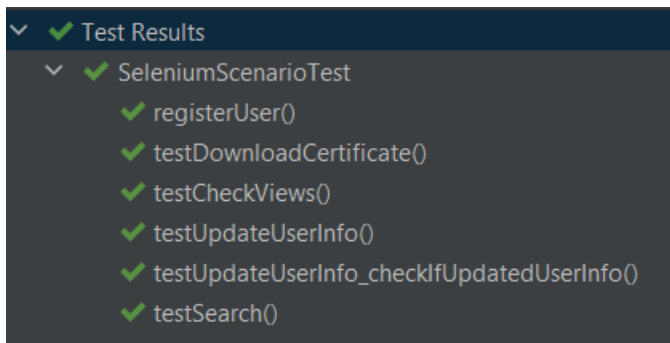
Се користи за тестирање со помош на Base Choice Coverage на методот UserService.create() кој што креира нов корисник во апликацијата.

## Load testing with JMeter

Се користи за тестирање на функционалноста на апликацијата при load од повеќе корисници кои испраќаат барања за содржината на апликацијата.

## 4. Резултати од тестирањето

### 1. Selenium Framework



Со селениум успешно поминаа сите тестови кои што ги тестираат сите погледи на апликацијата, како и сите рути кои што може да ги помине корисникот, со внесување на погрешни како и точни информации во полињата.

### 2. Spring MVC Test Framework

Test Results	1 sec 358 ms
PhalaenopsisTests	1 sec 358 ms
getLoginPage()	546 ms
loginUser()	157 ms
registerUser()	156 ms
getCertificatePage()	125 ms
getYellowRootsPage()	15 ms
getRootsPage()	16 ms
getStemPage()	16 ms
registerUserFailTest()	15 ms
getBlackRootsPage()	16 ms
getLeavesPage()	15 ms
updateUser()	141 ms
getRegisterPage()	16 ms
getGreenRootsPage()	15 ms
getWhiteRootsPage()	16 ms
testSearchTopics()	31 ms
getHomePage()	15 ms
getBrownRootsPage()	16 ms
getAirRootsPage()	16 ms
getFlowersPage()	15 ms

Со помош на Spring MVC Test Framework беше истестирана контролерската логика на апликацијата, погледите што се враќаат кон корисникот, статус кодовите како и атрибутите кои што се праќаат.

### 3. PIT Mutation Testing System, Mockito

Со мутациското тестирање беше постигнато целосно покривање на кодот како и убивање на сите мутанти.

## Pit Test Coverage Report

### Package Summary

com.phalaenopsis.phalaenopsis.service.impl

Number of Classes	Line Coverage	Mutation Coverage	Test Strength
3	100% 46/46	100% 31/31	100% 31/31

### Breakdown by Class

Name	Line Coverage	Mutation Coverage	Test Strength
<a href="#">CertificateServiceImpl.java</a>	100% 13/13	100% 9/9	100% 9/9
<a href="#">TopicServiceImpl.java</a>	100% 8/8	100% 5/5	100% 5/5
<a href="#">UserServiceImpl.java</a>	100% 25/25	100% 17/17	100% 17/17

Report generated by [PIT](#) 1.7.4

### 4. Input space coverage with Junit

Истестиран беше методот UserService.create() со користење на base choice coverage. Секоја карактеристика беше поделена на два блока True и False.

```
@Override
public User create(String username, String password, String repeatPassword, String role, String name, String surname) {
    if(!Objects.equals(username, "") && password.equals(repeatPassword) && findByUsername(username) == null) {
        User user = new User(username, passwordEncoder.encode(password), role, name, surname);
        return userRepository.save(user);
    }
    else throw new RuntimeException();
}
```

```
//Characteristics:

//username is null
//username is empty string

//password is null
//password is empty string

//repeatPassword is null
//repeatPassword is empty string

//role is null
//role is empty string

//name is null
//name is empty string

//surname is null
//surname is empty string

//each characteristic is partitioned into True and False blocks
```

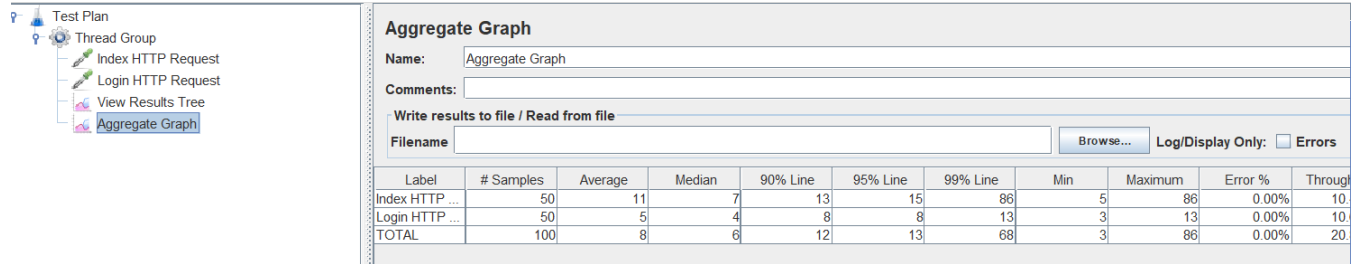
✓ Test Results

✓ ISPTest

- ✓ FFFFFFFFFTF\_Test()
- ✓ FFTFFFFFFFF\_Test()
- ✓ FFFFFFFFFF\_Test()
- ✓ FFFFTFFFFFFFF\_Test()
- ✓ FFFFFFFFFFT\_Test()
- ✓ FFFFFFFFFTFFF\_Test()
- ✓ FFFFFFFTFFFF\_Test()
- ✓ FTFFFFFFFFF\_Test()
- ✓ FFTFFFFFFFF\_Test()
- ✓ FFFFFFFTFFF\_Test()
- ✓ TFFFFFFFFF\_Test()
- ✓ FFFFTFFFFFFFF\_Test()
- ✓ FFFFFFFFFTFF\_Test()

## 5. Load testing with JMeter

Apache JMeter е апликација за load тестирање со чија помош беше направен тест план и тестирање на апликацијата при повеќе HTTP Get барања за нејзината содржина од страна на корисниците.



The screenshot displays the Apache JMeter interface. On the left, the 'Test Plan' tree shows a 'Thread Group' containing 'Index HTTP Request', 'Login HTTP Request', 'View Results Tree', and 'Aggregate Graph'. The 'Aggregate Graph' is selected, and its configuration is shown on the right. The 'Name' is 'Aggregate Graph'. Below the configuration, a table displays the aggregated results for the test plan.

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput
Index HTTP ...	50	11	7	13	15	86	5	86	0.00%	10.
Login HTTP ...	50	5	4	8	8	13	3	13	0.00%	10.
TOTAL	100	8	6	12	13	68	3	86	0.00%	20.