PROGRAMSKO INŽENJERSTVO

# Testiranje softvera

Unit testiranje • Integracijsko testiranje • E2E testiranje

Laboratorijske vježbe

# Vrste testiranja

F **Funkcionalno testiranje**

Provjerava funkcionira li svaka funkcija sustava u skladu sa zahtjevima i specifikacijama.

→ Unit, Integration, Smoke, Regression, UAT

N **Nefunkcionalno testiranje**

Provjerava nefunkcionalne aspekte sustava kao što su performanse, upotrebljivost i pouzdanost.

→ Performance, Load, Stress, Security, E2E

# Funkcionalno testiranje

### Unit Testing

Testiranje pojedine softverske komponente ili modula u izolaciji.

### Integration Testing

Testiranje svih integriranih modula radi provjere kombinirane funkcionalnosti.

### Sanity Testing

Brza provjera radi li nova verzija softvera dovoljno dobro za daljnje testiranje.

### Smoke Testing

Provjera da ne postoji blocker na novom buildu koji onemogućava daljnje testiranje.

### Regression Testing

Testiranje aplikacije u cjelini nakon izmjene bilo kojeg modula ili funkcije.

### User Acceptance Testing

Klijent provjerava radi li sustav prema poslovnim zahtjevima - zadnja faza prije produkcije.

# Nefunkcionalno testiranje

### Performance Testing

Testiranje zadovoljava li sustav performansne zahtjeve pod očekivanim opterećenjem.

### Load Testing

Testiranje maksimalnog opterećenja koje sustav može podnijeti bez degradacije performansi.

### Stress Testing

Testiranje sustava izvan specifikacija kako bi se provjerilo kako i kada zakazuje.

### Volume Testing

Testiranje ponašanja sustava pod jako velikim količinama podataka.

### Security Testing

Testiranje zaštićenosti sustava od unutarnjih i vanjskih prijetnji (pen testovi).

### End-to-End Testing

Testiranje cjelovitog okruženja koje oponaša stvarnu uporabu s bazom, mrežom i integracijama.

# Zašto unit testiranje?

✓ **Rano otkrivanje grešaka**
Ispravlja programske pogreške na početku razvojnog ciklusa kada su popravci najjeftiniji.

✓ **Zaštita od regresije**
Osigurava da buduće promjene ne naruše ponašanje postojećih funkcija.

✓ **Dokumentacija koda**
Testovi služe kao živa dokumentacija koja pokazuje kako se kod koristi.

✓ **Povjerenje pri refaktoriranju**
Razvijate čitljiviji i pouzdaniji kod s povjerenjem tijekom razvoja.

🧪

**Test-Driven Development**

Programer troši više vremena na čitanje nego na pisanje koda

# Given-When-Then obrazac

GIVEN

### Početno stanje

Opisuje kontekst i preduvjete testa. Postavlja početno stanje sustava prije izvođenja akcije.

WHEN

### Akcija

Opisuje akciju ili događaj koji se testira. To je ključna operacija koju želimo provjeriti.

THEN

### Očekivani rezultat

Opisuje očekivani ishod. Provjerava je li sustav u ispravnom stanju nakon akcije.

Ovaj obrazac čini testove čitljivijima i lakšima za održavanje

JAVA

## Spring Boot + JUnit 5

```java
@DisplayName("Discount coupon validation test")
public class DiscountCouponTest {
   private DiscountCouponService service;
   @BeforeEach
   public void setup() {
      service = new DiscountCouponService();
   }
   @Test
   public void testValidCoupon() {
      // Given
      String couponCode = "SPRINGBOOT";
      // When
      boolean isValid = service.validateCoupon(couponCode);
      // Then
      assertTrue(isValid);
   }
}
```

Biblioteke: **org.junit.jupiter** + **spring-boot-starter-test**

JAVA

## REST API testiranje s MockMvc

```
@AutoConfigureMockMvc
@SpringBootTest
class HardwareControllerTest extends BaseControllerTest {
   @Autowired
   private MockMvc mockMvc;
   @Test
   void getAll() throws Exception {
      mockMvc.perform(
         get("/hardware")
         .header(HttpHeaders.AUTHORIZATION, getAdminAuth())
         .contentType(MediaType.APPLICATION_JSON)
     )
     .andExpect(status().isOk())
     .andExpect(content().contentTypeCompatibleWith(JSON))
     .andExpect(jsonPath("$", hasSize(5)));
   }
}
```

**MockMvc** simulira HTTP zahtjeve bez pokretanja stvarnog servera

.NET

# C# + xUnit

```
public class DiscountCouponTests
{
   private readonly DiscountCouponService _service;
   public DiscountCouponTests() {
      _service = new DiscountCouponService();
   }
   [Fact]
   public void ValidateCoupon_WithValidCode_ReturnsTrue() {
      // Arrange (Given)
      var couponCode = "DOTNET2024";
      // Act (When)
      var result = _service.ValidateCoupon(couponCode);
      // Assert (Then)
      Assert.True(result);
   }
}
```

**[Fact]** za pojedinačne testove - **[Theory]** + **[InlineData]** za parametriziranje

.NET

## Web API integracijski test

```
public class HardwareApiTests : IClassFixture<WebApplicationFactory<Program>>
{
   private readonly HttpClient _client;
   public HardwareApiTests(WebApplicationFactory<Program> factory) {
      _client = factory.CreateClient();
   }
   [Fact]
   public async Task GetAll_ReturnsSuccessAndCorrectContentType() {
      // Act
      var response = await _client.GetAsync("/api/hardware");
      // Assert
      response.EnsureSuccessStatusCode();
      Assert.Equal("application/json",
      response.Content.Headers.ContentType?.MediaType);
   }
}
```

**WebApplicationFactory** pokreće in-memory test server - NuGet: **Microsoft.AspNetCore.Mvc.Testing**

PYTHON

## pytest - Unit testiranje

```python
import pytest
from services.discount_coupon import DiscountCouponService
class TestDiscountCoupon:
    @pytest.fixture
    def service(self):
        return DiscountCouponService()
    def test_valid_coupon(self, service):
        # Given
        coupon_code = "PYTHON2024"
        # When
        result = service.validate_coupon(coupon_code)
        # Then
        assert result is True
    @pytest.mark.parametrize("invalid_code", ["INVALID","", None])
        def test_invalid_coupon(self, service, invalid_code):
        result = service.validate_coupon(invalid_code)
        assert result is False
```

**@pytest.fixture** za pripremu - **@pytest.mark.parametrize** za parametrizirane testove - **pytest -v**

PYTHON

# FastAPI - Testiranje API-ja

```
from fastapi.testclient import TestClient
from main import app
client = TestClient(app)
def test_get_all_hardware():
   # When
   response = client.get("/api/hardware")
   # Then
   assert response.status_code == 200
   assert response.headers["content-type"] == "application/json"
   assert len(response.json()) == 5
   def test_get_hardware_not_found():
   response = client.get("/api/hardware/999")
   assert response.status_code == 404
def test_create_hardware():
   data = {"name": "GPU", "price": 599.99}
   response = client.post("/api/hardware", json=data)
   assert response.status_code == 201
```

**TestClient** simulira HTTP zahtjeve - slično za Flask: **flask.testing.FlaskClient**

NEXT.JS

## Jest - API Route testiranje

```
import { createMocks } from 'node-mocks-http';
import handler from '@/pages/api/hardware';
describe('Hardware API', () => {
   it('returns all hardware with status 200',async () => {
   // Given
   const { req, res } = createMocks({ method: 'GET' });
   // When
   await handler(req, res);
   // Then
   expect(res._getStatusCode()).toBe(200);
   expect(JSON.parse(res._getData())).toHaveLength(5);
});
   it('creates new hardware item',async () => {
      const { req, res } = createMocks({
         method: 'POST',
         body: { name: 'GPU', price: 599.99 }
   });
   await handler(req, res);
      expect(res._getStatusCode()).toBe(201);
   });
});
```

Paketi: **jest** - **node-mocks-http** - **@testing-library/react**

## Frontend testiranje

**React**

Jest + React Testing Library

`@testing-library/react`

**Angular**

Jasmine + Karma (ugradjeno)

`ng test`

**Vue**

Vitest + Vue Test Utils

`@vue/test-utils`

**Svelte**

Vitest + Svelte Testing Lib

`@testing-library/svelte`

**E2E**

Cypress
Playwright

**Visual**

Chromatic
Percy

**A11y**

jest-axe
pa11y

**Snapshot**

Jest snapshots
Storybook

REACT

## React Testing Library

```
import { render, screen, fireEvent } from '@testing-library/react';
import CouponForm from './CouponForm';
describe('CouponForm', () => {
   it('shows success message for valid coupon',async () => {
   // Given
   render(<CouponForm />);
   // When
   const input = screen.getByPlaceholderText('Unesite kupon');
   const button = screen.getByRole('button', { name: /primijeni/i });
   fireEvent.change(input, { target: { value: 'REACT2024' } });
   fireEvent.click(button);
   // Then
   expect(await screen.findByText(/kupon primijenjen/i))
      .toBeInTheDocument();
   });
});
```

Principi: Testiraj ponašanje, ne implementaciju - Koristi **getByRole**, **getByText** umjesto selektora

ANGULAR

## Jasmine + Karma

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { CouponFormComponent } from './coupon-form.component';
describe('CouponFormComponent', () => {
   let component: CouponFormComponent;
   let fixture: ComponentFixture<CouponFormComponent>;
   let couponService: jasmine.SpyObj<CouponService>;
   beforeEach(async () => {
      const spy = jasmine.createSpyObj('CouponService', ['validate']);
      await TestBed.configureTestingModule({
         declarations: [CouponFormComponent],
         providers: [{ provide: CouponService, useValue: spy }]
      }).compileComponents();
      fixture = TestBed.createComponent(CouponFormComponent);
      component = fixture.componentInstance;
   });
   it('should validate coupon on submit', () => {
      couponService.validate.and.returnValue(true);
      component.couponCode = 'ANGULAR2025';
      component.onSubmit();
      expect(couponService.validate).toHaveBeenCalledWith('ANGULAR2025');
   });
});
```

**TestBed** za konfiguraciju modula - **jasmine.SpyObj** za mockanje servisa - **ng test**

VUE

## Vitest + Vue Test Utils

```
import { mount } from '@vue/test-utils';
import { describe, it, expect } from 'vitest';
import CouponForm from './CouponForm.vue';
describe('CouponForm', () => {
   it('emits submit event with coupon code',async () => {
      // Given
      const wrapper = mount(CouponForm);
      // When
      await wrapper.find('input').setValue('VUE2024');
      await wrapper.find('form').trigger('submit');
      // Then
      expect(wrapper.emitted('submit')).toBeTruthy();
      expect(wrapper.emitted('submit')[0]).toEqual(['VUE2024']);
   });
   it('displays error message for invalid coupon',async () => {
      const wrapper = mount(CouponForm, {
         props: { error: 'Neispravan kupon' }
      });
      expect(wrapper.find('.error').text()).toBe('Neispravan kupon');
   });
});
```

**mount()** za renderiranje - **wrapper.emitted()** za provjeru dogadaja - **vitest**

THYMELEAF

## Spring MVC View testiranje

```java
@WebMvcTest(CouponController.class)
class CouponControllerTest {
   @Autowired private MockMvc mockMvc;
   @MockBean private CouponService couponService;
   @Test
   void showCouponForm_ShouldRenderTemplate() throws Exception {
      mockMvc.perform(get("/coupons"))
      .andExpect(status().isOk())
      .andExpect(view().name("coupon-form"))
      .andExpect(model().attributeExists("couponDto"));
   }
   @Test
   void validateCoupon_WithValidCode_ShouldShowSuccess() throws Exception {
      when(couponService.validate("SPRING2025")).thenReturn(true);
      mockMvc.perform(post("/coupons/validate")
      .param("code", "SPRING2025"))
      .andExpect(status().isOk())
      .andExpect(content().string(containsString("Kupon primijenjen")));
   }
}
```

**@WebMvcTest** za testiranje samo MVC sloja - **view().name()** provjerava ime templatea

.NET

## Razor Pages testiranje

```
public class CouponPageTests
{
   private readonly Mock<ICouponService> _mockService;
   private readonly CouponModel _pageModel;
   public CouponPageTests() {
      _mockService = new Mock<ICouponService>();
      _pageModel = new CouponModel(_mockService.Object);
   }
   [Fact]
   public void OnGet_ShouldReturnPage() {
      var result = _pageModel.OnGet();
      Assert.IsType<PageResult>(result);
   }
   [Fact]
   public async Task OnPost_WithValidCoupon_ShouldSetSuccessMessage() {
      _mockService.Setup(s => s.ValidateAsync("RAZOR2025"))
         .ReturnsAsync(true);
      _pageModel.CouponCode = "RAZOR2025";
      await _pageModel.OnPostAsync();
      Assert.True(_pageModel.IsValid);
   }
}
```

**Moq** za mockanje servisa - Testira se **PageModel** direktno - NuGet: **Moq**, **xunit**

# E2E testiranje - Cypress i Playwright

## Cypress

```
describe('Coupon Form', () => {
   it('applies coupon', () => {
   cy.visit('/checkout');
   cy.get('[data-cy=coupon]')
      .type('SAVE20');
   cy.get('[data-cy=apply]')
      .click();
   cy.contains('20% popusta')
      .should('be.visible');
   });
});
```

Pokretanje: **npx cypress open**

## Playwright

```
import { test, expect }
from '@playwright/test';
test('applies coupon',
   async ({ page }) => {
      await page.goto('/checkout');
      await page.fill('#coupon', 'SAVE20');
      await page.click('#apply-btn');
      await expect(page.locator('.discount'))
         .toContainText('20%');
});
```

Pokretanje: **npx playwright test**

E2E testovi simuliraju korisnicko iskustvo - Koristi **data-cy** ili **data-testid** atribute za stabilne selektore

## Mjerenje pokrivenosti koda

Code Coverage - koliko je koda pokriveno testovima?

**Line Coverage**

Postotak linija koda koje su izvrsene tijekom testiranja

**Branch Coverage**

Postotak grananja (if/else) koje su testirane

**Function Coverage**

Postotak funkcija/metoda koje su pozvane

### Alati po tehnologijama

| | | | | | |
|---|---|---|---|---|---|
| JAVA | JaCoCo | JS | Istanbul / nyc / Jest | NG | Karma + Istanbul |
| .NET | Coverlet | REACT | Jest --coverage | CY | @cypress/code-coverage |
| PY | pytest-cov / coverage.py | VUE | Vitest --coverage | PW | Playwright + Istanbul |

# Pokrivenost koda testovima

**Klase (Classes)**
Postotak testiranih klasa u projektu

**Metode (Methods)**
Postotak metoda izvršenih tijekom testova

**Linije (Lines)**
Postotak izvršenih linija koda

**Grananja (Branches)**
Postotak testiranih if/else grana

**80%**

**Minimalna pokrivenost**

Cilj je postići minimalno 80% u sve četiri kategorije za kvalitetan softver

JAVA

## JaCoCo - Code Coverage

### Maven konfiguracija (pom.xml)

```
<plugin>
<groupId>org.jacoco</groupId>
<artifactId>jacoco-maven-plugin</artifactId>
<version>0.8.14</version>
<executions>
<execution>
<goals><goal>prepare-agent</goal></goals>
</execution>
<execution>
<id>report</id>
<phase>test</phase>
<goals><goal>report</goal></goals>
</execution>
</executions>
</plugin>
```

### Pokretanje

```
mvn clean test
# Report: target/site/jacoco/index.html
```

### Minimalna pokrivenost

```
<check>
<rules><rule>
<element>BUNDLE</element>
<limits><limit>
<counter>LINE</counter>
<value>COVEREDRATIO</value>
<minimum>0.80</minimum>
</limit></limits>
</rule></rules>
</check>
```

JaCoCo generira HTML, XML i CSV izvjestaje - integrira se s **SonarQube**, **Jenkins**, **GitHub Actions**

.NET

## Coverlet - Code Coverage

### Instalacija i pokretanje

```
# Dodaj NuGet paket u test projekt

dotnet add package coverlet.collector
dotnet add package coverlet.msbuild

# Pokreni testove s coverage

dotnet test --collect:"XPlat Code Coverage"

# Ili s MSBuild integracijom

dotnet test /p:CollectCoverage=true
/p:CoverletOutputFormat=cobertura
```

### HTML izvjestaj s ReportGenerator

```
# Instaliraj ReportGenerator

dotnet tool install -g
dotnet-reportgenerator-globaltool

# Generiraj HTML report

reportgenerator
-reports:"**/coverage.cobertura.xml"
-targetdir:"coveragereport"
```

### Minimalna pokrivenost (.csproj)

```
<PropertyGroup>
<Threshold>80</Threshold>
<ThresholdType>line,branch</ThresholdType>
</PropertyGroup>
```

Formati: **cobertura**, **opencover**, **lcov**, **json** - Integracija s **Azure DevOps**, **SonarQube**

PYTHON

## pytest-cov - Code Coverage

### Instalacija i pokretanje

```
# Instaliraj pytest-cov
pip install pytest-cov

# Pokreni s coverage izvjestajem
pytest --cov=src --cov-report=html

# Terminal + HTML + XML izvjestaj
pytest --cov=src \
--cov-report=term-missing \
--cov-report=html \
--cov-report=xml
```

### pyproject.toml konfiguracija

```
[tool.coverage.run]
source = ["src"]
branch = true

[tool.coverage.report]
fail_under = 80
show_missing = true
```

### Primjer terminal outputa

```
Name                 Stmts   Miss   Cover
-----------------------------------
src/__init__.py 2 0 100%
src/coupon.py 25 3  88%
src/service.py 42 8  81%
-----------------------------------
TOTAL 69 11  84%
```

HTML report: **htmlcov/index.html** - XML za CI/CD: **coverage.xml** (Cobertura format)

JS/TS

# Jest / Vitest - Code Coverage

### Jest (React, Next.js, Node)

```
# Pokreni s coverage
npm test -- --coverage

# jest.config.js
{
collectCoverage: true,
coverageThreshold: {
global: {
branches: 80,
functions: 80,
lines: 80,
statements: 80
}
}
}
```

### Vitest (Vue, Vite projekti)

```
# Instaliraj coverage provider
npm i -D @vitest/coverage-v8

# vite.config.ts
test: {
coverage: {
provider: 'v8',
reporter: ['text', 'html'],
thresholds: {
lines: 80, branches: 80,
functions: 80
}
}
}
```

Report: **coverage/lcov-report/index.html** - Angular koristi **karma-coverage** s istim Istanbul backendom

## E2E Coverage - Cypress i Playwright

### Cypress Code Coverage

```
# Instaliraj plugin

npm i -D @cypress/code-coverage
npm i -D babel-plugin-istanbul

# cypress/support/e2e.js

import '@cypress/code-coverage/support'

# cypress.config.js

setupNodeEvents(on, config) {
require('@cypress/code-coverage/task')
(on, config);
}
```

### Playwright Coverage

```
# V8 native coverage

import { chromium } from 'playwright';

const browser = await chromium.launch();
const page = await browser.newPage();

# Pokreni coverage

await page.coverage
.startJSCoverage();
# ... testovi ...

const coverage = await page
.coverage.stopJSCoverage();
```

E2E coverage mjeri koje dijelove frontend koda korisnik stvarno koristi - kombinirati s unit test coverageom za potpunu sliku

## CI/CD integracija - Coverage u pipelineu

### GitHub Actions primjer

```
name: Test Coverage
on: [push, pull_request]
jobs:
test:
runs-on: ubuntu-latest
steps:
uses: actions/checkout@v4
run: npm ci
run: npm test --coverage
uses: codecov/codecov-action@v3
with:
fail_ci_if_error: true
```

### Coverage servisi

**Codecov**
PR komentari, badgevi, trendovi

**Coveralls**
Besplatno za open source

**SonarQube**
Quality gates, security

**Azure DevOps**
Ugradjena integracija

### Coverage badge u README

```
![Coverage]
(https://codecov.io/gh/user/repo/badge)
```

| | | |
|---|---|---|
| coverage 85% | coverage 72% | coverage 45% |

Preporuka: Postavite **minimalni prag** u CI - build pada ako coverage padne ispod 80%

# Projektni zadatak

## Zadatak

Na projektnom zadatku napišite unit testove (koji koriste „mockove" za ostale slojeve aplikacije) i integration testove (koji ne koriste „mockove" za ostale dijelove aplikacije) za svaku klasu na **frontend** i **backend** dijelu aplikacije (I8 – 1 bod). Uključite pokretanje testova u sklopu CI/CD *pipelinea* (I3 – 1 bod).

### 📊 Zahtjevi za pokrivenost koda:

| ≥80% | ≥80% | ≥80% | ≥80% |
|---|---|---|---|
| Klase | Metode | Linije koda | Grananja |

- ✓ Testirajte servise i kontrolere
- ✓ Testirajte UI komponente
- ✓ Dokumentirajte test izvještaje

# Pregled alata za testiranje

## BACKEND

**Java/Spring**
JUnit 5, MockMvc, Mockito

**.NET/C#**
xUnit, NUnit, Moq

**Python**
pytest, unittest

**Node.js**
Jest, Mocha, Supertest

## FRONTEND

**React**
RTL, Jest, Vitest

**Angular**
Jasmine, Karma, TestBed

**Vue**
Vue Test Utils, Vitest

**Server-side**
Thymeleaf, Razor, MVC

## E2E & INTEGRACIJA

- Cypress - E2E testiranje
- Playwright - Cross-browser E2E
- Selenium - Browser automation
- Postman/Newman - API testiranje

# Hvala na pažnji!

Pitanja?

🧪 Testirajte rano i često

📊 Cilj: 80% pokrivenost

🚀 Automatizirajte testove