Тестирање на Full-stack апликација со помош на Junit, RestAssured, JMeter и CYPress

Линк до GitHub репозиториум: https://github.com/ManuelTrajcev/SQAT_Project

Септември 2025

Изработил:

Мануел Трајчев 221045

Содржина

1. Вовед

Тестирањето е клучен аспект во развојот на софтверски апликации, бидејќи помага да се осигура дека кодот функционира правилно и дека системот е стабилен. Во оваа семинарска работа ќе ги разгледаме различните типови тестови кои се користат за тестирање на бекенд апликации, како и конкретни примери за тестови користејќи алатки како **Junit**, **RestAssured, JMeter** и **Cypress**.

2. Тестирање

2.1 Unit Тестови

Unit тестовите се користат за тестирање на поединечни компоненти или функции на системот. Овие тестови обично се пишуваат за секој дел од кодот и осигуруваат дека сите делови на апликацијата работат како што се очекува.

Junit

JUnit е најпознатата Java библиотека за пишување и извршување unit тестови. Таа овозможува лесно дефинирање на тест случаи, проверка на резултати преку assertions и автоматско извршување на тестови. Со JUnit програмерите можат брзо да откријат грешки и да осигурат дека новите промени не влијаат негативно врз постоечката функционалност.

WorkspacesApplicationTests.java

- **Цел:**
  Главен Spring Boot тест кој проверува дали апликацискиот контекст се вчитува успешно.

unitTests/repositoryTests/

- **UserRepositoryTests.java:**
  Овие тестови се насочени кон верификација на методите за пребарување и други операции на репозиторумот за корисници.

- **WorkspaceRepositoryTests.java:**
  Овие тестови се фокусираат на CRUD операции и специфични упити на

репозиторумот за workspaces.

unitTests/servicesTests/

- **UserApplicationServiceImplTest.java:**
  Овие тестови ја проверуваат деловната логика на корисничкиот сервис, вклучувајќи регистрација и валидација на лозинки.

- **WorkspaceServiceImplTest.java:**
  Овие тестови ги проверуваат методите за управување со workspaces во сервис слојот.

2.2 API тестовите

RestAssured

RestAssured е Java библиотека која се користи за тестирање на RESTful API-ја. Таа ви овозможува да пишувате читливи и одржливи тестови кои симулираат HTTP барања и ги верификуваат одговорите. RestAssured се користи најчесто во интеграциони и API тестови за да се провери дали крајните точки се однесуваат како што се очекува.

Клучни карактеристики:

- Поддржува HTTP методи (GET, POST, PUT, DELETE и сл.)
- Овозможува лесно поставување на хедери на барања, параметри и тела
- Обезбедува fluent тврдења за статус на одговор, хедери и содржина на телото
- Добро се интегрира со JUnit и други тест фрејмворци
-

Зошто да го користите RestAssured?

- Ја поедноставува тестирањето на API во Java проекти
- Ги прави тестовите поизразни и полесни за пишување во споредба со рачниот код за HTTP клиент
- Помага да се осигура дека вашите бекенд крајни точки се сигурни и враќаат точни податоци

UserControllerTest.java

1. **Регистрирање на корисник (POST /api/user/register)**
    - **Цел:** Тестирање на регистрација на корисник со валидни податоци.
    - **Што се тестира:**
        - Дали нов корисник може да биде креиран.
        - Дали одговорот содржи статус на успех и информации за корисникот.
        - Обработка на грешки за дупликатни или невалидни податоци.
2. **Најава (POST /api/user/login)**
    - **Цел:** Тестирање на најава на корисник со валидни кредeнцијали.
    - **Што се тестира:**
        - Дали корисникот добива валиден токен за автентикација.
        - Обработка на грешки за погрешни кредeнцијали.
3. **Бришење на корисник (DELETE /api/user/delete/{username})**
    - **Цел:** Тестирање на бришење на корисник.
    - **Што се тестира:**
        - Дали корисникот може да биде избришан по корисничко име.
        - Правилен одговор за неекзистирачки корисници.
4. **Одјава (GET /api/user/logout)**
    - **Цел:** Тестирање на одјава на корисник.
    - **Што се тестира:**
        - Дали сесијата или токенот се невалидирани.
        - Правилен статус на одговорот.
5. **Негативна регистрација/најава/бришење**
    - **Цел:** Тестирање на сценарија за грешки (нпр. невалидни податоци, дупликатна регистрација, погрешна лозинка).
    - **Што се тестира:**

        - Дали API враќа соодветни пораки за грешки и статус кодови.

WorkspaceControllerTest.java

1. **Преземање на сите workspaces (GET /api/workspace)**
    - **Цел:** Тестирање на преземање на сите workspaces.
    - **Што се тестира:**
        - Дали крајната точка враќа листа со workspaces.
        - Точна структура на податоците и статус код.
2. **Преземање workspace по ID (GET /api/workspace/{id})**
    - **Цел:** Тестирање на преземање конкретен workspace.
    - **Што се тестира:**
        - Дали се враќа точен workspace за валидно ID.
        - Обработка на грешки за невалидни/неекзистирачки ID.
3. **Креирање workspace (POST /api/workspace)**
    - **Цел:** Тестирање на креирање workspace.
    - **Што се тестира:**
        - Дали workspace може да се креира со валидни податоци.
        - Обработка на грешки за недостасувачки или невалидни полиња.
4. **Уредување workspace (POST /api/workspace/edit/{id})**
    - **Цел:** Тестирање на уредување постоечки workspace.
    - **Што се тестира:**
        - Дали деталите на workspace можат да се ажурираат.
        - Обработка на грешки за невалиден ID или податоци.
5. **Преземање на моите workspaces (GET /api/workspace/my-workspaces)**
    - **Цел:** Тестирање на преземање на workspaces што му припаѓаат на автентификуваниот корисник.
    - **Што се тестира:**

        - Дали се враќаат само workspaces на корисникот.
        - Правилно спроведување на автентикација.

6. **Негативен пристап/уредување на workspace**
    - **Цел:** Тестирање на сценарија за грешки (нпр. неовластен пристап, уредување на неекзистирачки workspace).
    - **Што се тестира:**
        - Дали API враќа соодветни пораки за грешки и статус кодови.

Секое API барање е дизајнирано да ја потврди како **среќната патека** (правилна употреба) така и **обработката на грешки** (невалидна или неовластена употреба) за вашите бекенд крајни точки. Ова осигурува дека вашата апликација е стабилна, сигурна и пријателска за корисниците.

Резиме:

Овие API тестови обезбедуваат сигурност дека HTTP крајните точки на вашата апликација работат како што се очекува, ги обработуваат граничните случаи и враќаат соодветни одговори. Тие помагаат да се откријат проблеми со интеграцијата на рана фаза и да се осигура сигурно искуство за корисниците кои комуницираат со вашиот бекенд.

2.3 UI End-To-End тестирање

Cypress

Cypress е алатка за автоматизирано тестирање на веб-апликации која е специјализирана за функционално тестирање. Таа овозможува пишување на тестови кои симулираат корисничко однесување и проверуваат дали апликацијата функционира како што се очекува.

1. app.cy.js

- **beforeEach хук:**
  Хукот beforeEach во Cypress се користи за чистење на колачиња и локалната меморија пред секој тест, како и за логирање на корисникот, обезбедувајќи свежа сесија за секој тест.

- **Тест за HomePage:**
  Овој тест ја посетува почетната страница и проверува дали се појавува насловот на апликацијата „Workspaces Management System“ и добредојдната порака „Welcome“.

- **Тестови за WorkspacesPage:**
  Првиот тест проверува дали табовите „All Workspaces“ и „My Workspaces“ се видливи, додека вториот тест проверува дали при префрлањето на табот „My Workspaces“ се појавува мрежата со workspaces.

2. login.cy.js

- Овој тест го посетува сајтот за најава и проверува дали се пренасочува на почетната страница по успешна најава со користење на корисничко име и лозинка.

3. logout.cy.js

- Тестот за одјава проверува дали, по кликањето на копчето за одјава, корисникот е пренасочен на страницата за најава.

4. accessWorkspaceMyWorkspacesTab.cy.js

Овој тест го проверува функционирањето на popup прозорецот за workspace и потврдува дека е затворен по кликнување на копчето „Cancel“.

2.4 Performance/Load тестови

JMeter

JMeter е алатка која се користи за тестирање на перформансите и оптоварувањето на веб апликации, симулирајќи голем број на корисници кои истовремено вршат операции на API.

JMeter Тест за оптоварување

Тестот симулира различни кориснички интеракции со бекенд API, кој работи на localhost:8080. Тој користи еден поток (корисник) и извршува низа HTTP барања за тестирање на различни крајни точки поврзани со workspaces и корисничко управување.

Клучни компоненти:

- **Група на потоци (Thread Group):**
  Тестот е конфигуриран со 100 корисници и 1 итерација, што значи дека секое барање ќе се изврши еднаш по ред.

- **HTTP Барања:**
  Тест планот вклучува барања за преземање, уредување и бришење workspaces и корисници. Освен позитивните тестови, се вклучени и негативни тестови за невалидни барања и грешки.

- **Тврдења и резултати:**
  Секое барање е проследено со слушателот „View Results Tree“ за да се прикажат одговорите, статусите и временските информации.

Цел на тестот:

Тестот има за цел да провери како системот се справува со различни операции и како функционира при повисоко оптоварување. Се проверуваат функционалните и перформансни карактеристики на системот.

3. Заклучок

Оваа семинарска работа ги опишува различните алатки и техники кои се користат за тестирање на веб апликации. Преку користење на Cypress за функционално тестирање на frontend, JMeter за тестирање на перформанси и Unit тестови за проверка на поединечни компоненти и API тестови за проверка на крајните точки дотапни од backend, се осигурува дека апликацијата е стабилна и функционална. Тестирањето е критично за обезбедување на квалитетот на софтверот и за спречување на грешки во продуктивната средина.