Универзитет „Св. Кирил и Методиј“ - Скопје
Факултет за информатички науки и компјутерско инженерство
Софтверски квалитет и тестирање

End-to-end, Unit и API тестирање со Cypress, Jest и Supertest на веб страниците Saucedemo и Reqres

Изработиле:
Ана Коловска 221226
Бојана Боцевска 221227

Ментори:
д-р Ѓорѓи Мацаров
д-р Христина Михајлоска

Contents

Вовед

Во современата разработка на софтвер, автоматизираното тестирање претставува клучен елемент за обезбедување на квалитет и стабилност на апликациите. Со зголемување на комплексноста на софтверските системи, рачното тестирање станува неефикасно и подложно на грешки. Затоа се користи комбинација од различни алатки за тестирање кои овозможуваат целосна покриеност на функционалноста на системот.

Оваа работа се фокусира на три клучни технологии за автоматизирано тестирање: Cypress**,** Jest и Supertest**.**

- **Cypress** е алатка за end-to-end (E2E) тестирање која овозможува симулација на кориснички интеракции во веб апликации и проверка на целосниот работен тек. Со него можат да се тестираат интеракциите на корисникот, функционалностите на UI-то и интеграцијата со backend системите.
- **Jest** е framework за unit тестирање кој се користи за верификација на логиката на функциите и компонентите. Тој овозможува тестирање на индивидуални делови од кодот и алгоритмите, обезбедувајќи точност и стабилност во работењето на апликацијата.
- **Supertest** е алатка за тестирање на API-ја која овозможува автоматско испраќање на HTTP барања и верификација на одговорите. Со него се тестираат крајните точки на backend системот, како и нивната соработка со други компоненти.

Целта на оваа работа е да се прикаже логиката на тестирањето и да се анализира графичката покриеност на тестовите преку интеграцијата на овие три технологии. Преку нив се постигнува значително намалување на дефектите, зголемување на продуктивноста и подобрување на стабилноста на системот.

Главен дел

1. Користени алатки и технологии

- Cypress - алатка за end-to-end тестирање со можност за извршување тестови во реално време и добивање визуелен приказ на резултатите.
- Jest – рамка за unit и integration тестирање на JavaScript/Node.js логика.
- Supertest – библиотека за автоматско тестирање на API крајни точки.
- Fixtures (JSON) - користени за чување и манипулирање со тест-податоци
- Mocha - тест рамка кој обезбедува структура (describe, it блокови).
- Chai - библиотека за assertions (should, expect).

2. Cypress и Jest end-to-end и unit тестирање

2.1. Логика на тестирање

Тестирањето е клучно за осигурување на точноста и сигурноста на софтверот. Cypress и Jest имаат дополнителни улоги:

- Cypress е наменет за end-to-end (E2E) и интегрирано тестирање, симулирајќи кориснички интеракции во реален прелистувач. Тој го тестира апликацискиот тек како целина, проверувајќи ги работните процеси и UI однесувањето во реални услови.
- Jest е framework за unit и integration тестирање за JavaScript/Node.js, фокусиран на тестирање на изолирани функции и API ја без потреба од прелистувач.

Комбинирањето на Cypress и Jest овозможува целосно покривање на предната логика и backend функционалноста. На пример, кај типична e-commerce апликација

- Jest тестови ќе проверуваат точност на функции: пресметка на цени, валидација на податоци, одговори од API ја и слично.
- Cypress тестови ќе проверуваат цели текови: најава, додавање производи во кошничка, завршување на нарачка, и верификација на UI то.

За илустрација на примената на Cypress во функционално тестирање, следат конкретни сценарија кои покриваат логирање, работа со продукти, сортирање и процес на нарачка. Овие тестови се имплементирани во кодот подолу, каде се користат custom Cypress команди и Jest функции за верификација на правилноста на корисничките интеракции, износи на производи и сортирање.

- Логирање:
  i. Проверка на присутност и видливост на полиња за корисничко име, лозинка и копче за најавување.
  ii. Тестирање успешно логирање со валидни податоци (standard_user / secret_sauce).
  iii. Тестирање неуспешно логирање со невалидни податоци и појава на порака за грешка.

iv. Логирање со повеќе типови корисници (валиден, блокиран, проблематичен), користејќи fixture.

- Страница со продукти:
  i. Потврда дека сите продукти се прикажуваат правилно по логирањето.
  ii. Додавање производи во кошничката и проверка дека бројачот (badge) се ажурира.
- Кошничка и нарачка:
  i. Додавање производи во кошничката и проверка дека се прикажуваат правилно.
  ii. Валидација дека сумата на сите производи е точна.
  iii. Тестирање на процесот на нарачка со внесување податоци.
  iv. Проверка на грешки ако се остават празни полиња.
- Сортирање на производи:
  i. Проверка дека сите опции за сортирање функционираат правилно (A→Z, Z→A, Price low→high, Price high→low).
  ii. Потврда дека редоследот на производите одговара на очекуваното.

2.2. Графичка покриеност

Во тестирањето, графичката покриеност ја мери количината од однесувањето на апликацијата која е тестирана.

За Cypress:

- Graph coverage се дефинира преку UI текови и транзиции на страници.
- Тестовите следат транзиции помеѓу состојби на апликацијата: login → dashboard → избор на производ → checkout.

Пример:
Cypress тест кој ја посетува /inventory.html, клика на производ и го додава во кошничка, покрива специфични рути и UI состојби, со што ја зголемува графичката покриеност.

За Jest:

- Graph coverage се мери преку извршување на функции и разгранувањата во кодот.
- Jest coverage извештаите покажуваат:
  - Statement coverage
  - Branch coverage
  - Function coverage
  - Line coverage

Пример:

Unit тестови за функцијата sumPrices(prices) ќе проверат повеќе случаи (празен список, валидни цени, невалидни вредности) за да се осигура дека секоја гранка е тестирана.

2.3. Извештаи за покриеност

Комбинирањето на извештаите за покриеност од Cypress и Jest дава целосна мапа на покриеност:

- Jest дава детални функционални покриености за backend логика и unit функции.
- Cypress опфаќа сценарија за кориснички текови и интеграција.

Со интегрирање на двете, развивачите можат да откријат лакови во логиката и недостатоци во UI интеракцијата, обезбедувајќи сигурен и квалитетен софтвер.

2.4. Предности од комбинираното тестирање со Cypress и Jest

- Сеопфатна покриеност: Unit, integriraweинструменти
- и E2E тестирање заедно.
- Побрзо откривање грешки: Unit тестови за логика, E2E тестови за UI/интеграција.
- Подобра одржливост: Јасно одвојување на тестирање на логика и UI.
- Поголема доверба: Графичката покриеност обезбедува дека се тестирани кодни патишта и кориснички текови.

2.5. Методологија

- Чекор 1 – Unit тестирање (Jest): Тестирање на чисти функции, API барања и основна логика.
- Чекор 2 – Integration тестирање (Jest): Тестирање на модули кои работат заедно.
- Чекор 3 – E2E тестирање (Cypress): Симулирање на реални кориснички текови, верификација на UI и интеграција.
- Чекор 4 – Анализа на покриеност: Користење на Jest и Cypress coverage извештаи за целосна покриеност.

Оваа методологија осигурува точност на логиката, функционална сигурност и висока покриеност, што претставува ефикасен тестирачки пристап за современ софтверски развој.

3. API тестирање со Supertest и Jest

3.1. Логика на API тестирање

Supertest претставува библиотека за автоматизирано тестирање на HTTP API-ја во Node.js околина. Таа се користи за верификација на точноста, сигурноста и стабилноста на backend

услугите преку симулација на реални клиентски барања и анализа на нивните одговори. За разлика од рачното тестирање на API-ја, каде барањата се праќаат преку надворешни алатки, Supertest овозможува тестирањето да биде дел од автоматизираната тестна инфраструктура и да се извршува континуирано, заедно со останатите unit и integration тестови.

Логиката во тестирањето со Supertest се темели на:

- Испраќање HTTP барања (GET, POST, PUT, DELETE и други) кон API.
- Верификација на статус кодови (200, 404, 500 и други).
- Потврда на содржината на одговорот (body, headers, JSON структура).
- Потврда на време на одговор и други метрики.

Со ваквиот пристап, Supertest не само што овозможува проверка на функционалната исправност на API-то, туку и тестирање на неговата робустност, стабилност и усогласеност со очекуваните стандарди. Овој вид тестирање е особено важен во системи каде API-то претставува критичен комуникациски слој меѓу frontend и backend деловите на апликацијата, бидејќи секоја грешка на овој слој директно влијае врз корисничкото искуство и интеграцијата со други услуги.

Пример:
Тестирање на API за корисници може да вклучува:

- Проверка дали /api/users?page=2 враќа статус код 200 и листа на корисници.
- Проверка дали /api/users/2 враќа корисник со точен JSON формат.

3.2. Логичка структура на тестови

Типичен Supertest тест се состои од:

1. Иницијализација на тестната сесија со request(server).
2. Извршување на HTTP барање со .get(), .post(), .put(), .delete().
3. Поставување на headers или body со .set() и .send().
4. Верификација на одговорот со .expect() или expect(response.body).toMatchObject(...).

Пример во Jest:

```
1. const request = require('supertest');
   const api = request("https://reqres.in");


   describe('reqres.in API тестирање', () => {
       it('GET /api/users?page=2', async () => {
           const response = await api
            2. .get("/api/users?page=2")
            3. .set("x-api-key", "reqres-free-v1");


       4.  expect(response.status).toBe(200);
           expect(response.body.page).toBe(2);
       });});
```

3.3. Графичка покриеност

Графичката покриеност при API тестирање се мери како процент од покриените API крајни точки и сценарија.

Со Supertest можеш да ги тестираш:

- Сите HTTP методи (GET, POST, PUT, DELETE).
- Варијации на податоци (валидни, невалидни, празни, големи).
- Сите статус кодови и грешки.
- Headers и метаподатоци.

Пример:
Ако API има /api/users, /api/users/:id и /api/login, тестирањето треба да опфати сите овие крајни точки и сите важни сценарија (валидни барања, грешки, edge cases).

3.4. Извештаи за покриеност

Supertest во комбинација со Jest дава детални извештаи за покриеност на backend API:

- Statement coverage
- Branch coverage
- Function coverage
- Line coverage

Во контекст на овој проект, преку Supertest се тестираат сите клучни HTTP методи (GET, POST, PUT, PATCH, DELETE), а извештаите овозможуваат јасен преглед на тоа колку од логиката на API-то е проверена. Со нив лесно се идентификуваат делови од кодот што не се извршени, како и сценарија што недостигаат во тековните тестови.

3.5. Предности од тестирањето со Supertest

- Тестирање на цел API без зависност од UI.
- Автоматизација на backend тестови како дел од CI/CD процес.
- Потврда на API барање за да се осигура дека структурата на одговорот останува конзистентна.
- Тестирање на безбедност преку поставување headers, токени и API-клучеви.

3.6. Методологија

- Чекор 1 – Дефинирање на тест сценарија: Определување на сите API крајни точки и очекувани одговори.
- Чекор 2 – Unit и integration тестирање: Тестирање на изолирани модули и интеграција на API крајни точки.
- Чекор 3 – API функционални тестови со Supertest: Испраќање барања и проверка на одговори.
- Чекор 4 – Извештаи за покриеност: Користење на Jest coverage извештаи за покриеност на backend кодот.

Овој пристап овозможува целосна верификација на API-то, гарантирајќи точност и сигурност на backend логиката.

Заклучок

Интеграцијата на Cypress, Jest и Supertest овозможува сеопфатен пристап кон автоматизираното тестирање, кој ги покрива сите критични слоеви на софтверскиот систем. Cypress овозможува тестирање на целосниот кориснички тек, симулирајќи реални интеракции и проверувајќи функционални и визуелни аспекти на апликацијата. Jest обезбедува unit тестирање кое ја верификува логиката, алгоритмите и функциите, овозможувајќи брзо откривање и корекција на грешки во кодот. Supertest ја проверува точноста на API крајните точки и нивната соработка со други системи, овозможувајќи стабилност и квалитет на backend-от.

Овој интегриран пристап овозможува шорока графичка покриеност и рано откривање на дефекти, што значително ги намалува времето и трошоците за нивно корегирање. Со автоматизираното тестирање се зголемува продуктивноста бидејќи овозможува брзо повторно тестирање и континуирана верификација на функционалностите. Со правилна имплементација на овие технологии, се гради стабилен и одржлив тестен екосистем кој овозможува континуирана интеграција и испорака на софтвер со висок квалитет.

Линк до проектот на GitHub -
https://github.com/akolovska/Saucedemo-cypress-testing

Линк до Cypress и Jest тестирањето -

https://github.com/akolovska/Saucedemo-cypress-testing/blob/master/cypress/e2e/test.cy.js

Линк до supertest тестирањето -

https://github.com/akolovska/Saucedemo-cypress-testing/blob/master/supertest/api.test.js