Motivarea alegerii framework-ului Selenium

Am ales Selenium pentru testarea automată a aplicațiilor web datorită mai multor avantaje pe care le oferă. În primul rând, este o soluție open-source și gratuită, ceea ce o face accesibilă și rentabilă pentru companii de diferite dimensiuni. Selenium este compatibil cu o varietate de platforme și browsere, permițând testarea aplicațiilor web pe medii diverse și asigurând o acoperire completă.

Suportul pentru mai multe limbaje de programare, cum ar fi Java, Python, și C#, oferă flexibilitate și ușurință în integrarea cu proiectele existente. Selenium Grid permite rularea testelor în paralel pe mai multe mașini, accelerând procesul de testare și îmbunătățind eficiența.

Comparând Selenium cu alte framework-uri de testare automatizată, vom examina câteva alternative populare, fiecare având punctele sale forte și limitări. Printre acestea se numără Cypress, Puppeteer, WebDriverIO și Playwright. Iată o comparație detaliată între Selenium și fiecare dintre aceste framework-uri:

1. Selenium vs. Cypress

- Arhitectură: Selenium interacționează cu paginile web prin browser, folosind drivere specifice fiecărui browser. Cypress rulează în același run-time cu aplicația testată, oferind o performanță superioară și acces mai direct la elementele DOM.
- Suport pentru limbaje: Selenium suportă multiple limbaje de programare, în timp ce Cypress este limitat la JavaScript.
- Suport multi-browser: Selenium are un suport mai extins pentru diferite browsere și versiuni, în timp ce Cypress avea inițial suport limitat doar pentru Chrome, extinzându-se recent la Firefox și Edge.
- Ușurința de utilizare: Cypress este adesea considerat mai ușor de folosit pentru dezvoltatorii JavaScript datorită sintaxei sale concise și suportului pentru testare real-time.
- Capabilități de testare: Cypress nu suportă testarea pe mai multe tab-uri sau fereastra și nu este potrivit pentru aplicații care se extind pe mai multe domenii la fel de bine ca Selenium.

2. Selenium vs. Puppeteer

- Arhitectură: Puppeteer controlează direct browserul Chrome sau Chromium printr-o interfață de programare a aplicațiilor (API) dedicată, ceea ce oferă o mare viteză și control. Selenium poate interacționa cu mai multe browsere.
- Limitări de browser: Puppeteer este limitat la Chrome și Chromium, în timp ce Selenium are un suport mai larg.
- Utilizare: Puppeteer este ideal pentru testarea aplicațiilor web care necesită performanțe înalte în Chrome, automatizări specifice browserului sau capturi de ecran și PDF-uri. Selenium este mai versatil pentru testarea cross-browser.

3. Selenium vs. WebDriverIO

- Ecosistem: WebDriverIO este un wrapper pentru Selenium care oferă o sintaxă mai simplă și suport pentru JavaScript (Node.js). Acesta oferă o integrare mai bună cu sistemele moderne JS și framework-urile de testare.
- Funcționalități: WebDriverIO poate integra direct servicii și extensii pentru a extinde funcționalitatea, cum ar fi testele vizuale sau de integrare.
- Scalabilitate: Ambele suportă testare scalabilă și distribuită, dar WebDriverIO are o curvă de învățare mai mică pentru dezvoltatorii JS.

4. Selenium vs. Playwright

- Suport multi-browser: Playwright suportă nativ Chromium, Firefox și WebKit, oferind o acoperire similară cu Selenium, dar cu o implementare mai consistentă pentru testare cross-browser.
- Funcționalități noi: Playwright permite testarea în contexte mobile, interceptarea rețelei și simularea geolocației.
- Performanță: Playwright oferă o performanță îmbunătățită prin evitarea JSON Wire Protocol folosit de Selenium, reducând astfel latența.

Câteva exemple de teste generate pentru aplicația noastra web care utilizează filtrarea produselor după detalii:

Test 1: Filtrare pe baza unui cuvânt cheie care nu există

Scenariu de Test: Filtrarea produselor folosind un cuvânt cheie inexistent**

1. Precondiții: Utilizatorul trebuie să fie pe pagina cu produse.

2. Pași de Test:

- Navighează către câmpul de căutare 'Detalii' și introdu textul 'unicorn'.
- Apasă pe butonul 'Filtrare'.
- Așteaptă până când sistemul încearcă să localizeze elementele cu clasa 'produs'.

3. Rezultat așteptat:

- Nu ar trebui să fie afișat niciun produs.
- Sistemul ar trebui să informeze utilizatorul că nu există produse care să corespundă criteriilor de căutare.

Test 2: Filtrare cu cuvânt cheie valid cu caracteristici speciale

Scenariu de Test: Filtrarea produselor folosind un cuvânt cheie cu caractere speciale**

1. Precondiții: Utilizatorul trebuie să fie pe pagina cu produse.

2. Pași de Test:

- Navighează către câmpul de căutare 'Detalii' și introdu textul 'pește@2024'.
- Apasă pe butonul 'Filtrare'.
- Așteaptă până când elementele cu clasa 'produs' sunt localizate.

4. Rezultat așteptat:

- Produsele care conțin cuvântul 'pește@2024' în descrierea lor ar trebui să fie afișate.
- Testul ar trebui să verifice dacă aplicația gestionează corect caracterele speciale.

Test 3: Performanța filtrării la un volum mare de date

Scenariu de Test: Testarea performanței filtrării cu un număr mare de produse**

1. Precondiții: Utilizatorul trebuie să fie pe pagina cu produse care conține un număr mare de intrări (de exemplu, peste 1000 de produse).

2. Pași de Test:

- Navighează către câmpul de căutare 'Detalii' și introdu textul 'fruct'.
- Apasă pe butonul 'Filtrare'.
- Așteaptă până când elementele cu clasa 'produs' sunt localizate.

3. Rezultat așteptat:

- Filtrarea produselor trebuie să fie completată într-un timp rezonabil (de exemplu, sub 5 secunde).
- Testul ar trebui să verifice eficiența filtrării în condiții de sarcină mare.

Aceste teste sunt concepute pentru a verifica diverse aspecte ale funcționalității de filtrare a aplicației tale web, de la gestionarea intrărilor neobișnuite până la performanța sistemului sub sarcină.

În concluzie, alegerea unui framework de testare automatizată depinde de cerințele specifice ale proiectului, experiența echipei, și de necesitățile de compatibilitate și performanță ale aplicațiilor testate. Selenium rămâne o opțiune robustă pentru proiecte care necesită testare extensivă cross-browser și suport pentru mai multe limbaje de programare, în timp ce alte framework-uri pot oferi avantaje specifice pentru cazuri de utilizare mai restrânse sau medii tehnice specifice.

Diagramele Grafului Cauză-Efect

user story 1: utilizatorul doreste sa se duca pe pagina de produse si sa caute un produs care contine peste in ingrediente

Cauze / Conditii intrare:

C1: Utilizatorul se află pe pagina de produse

C2: Funcționalitatea de căutare este activă

C3: Utilizatorul introduce termeni de căutare

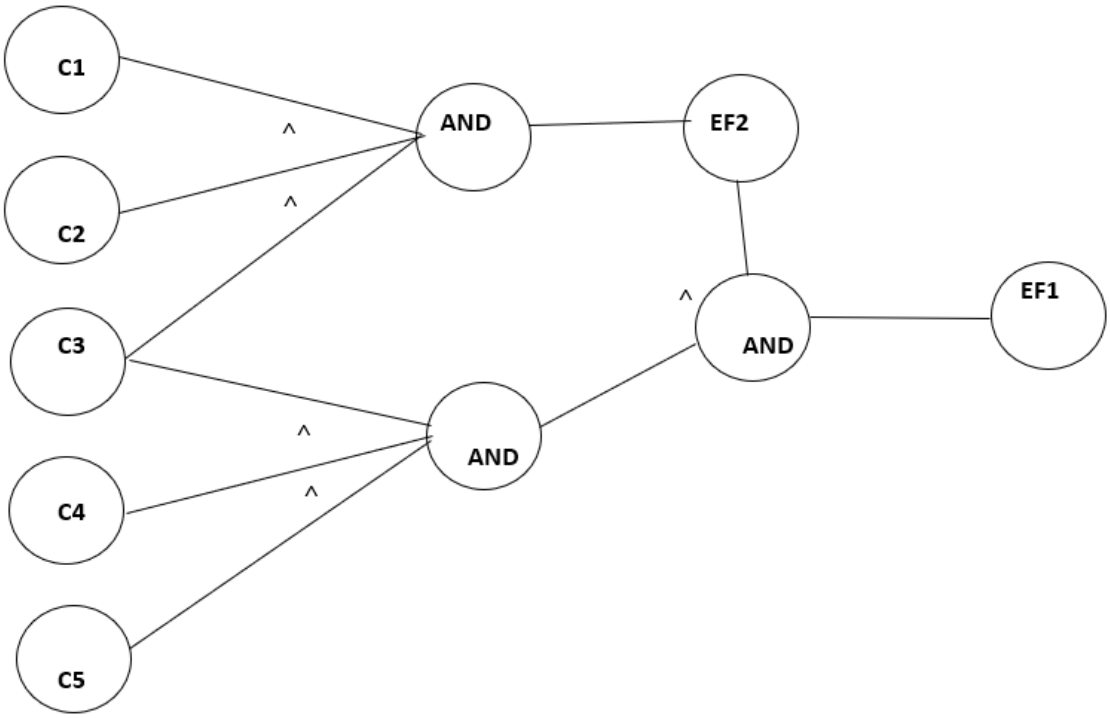
C4: Termenul de căutare specific "pește" este folosit

C5: Utilizatorul apasă butonul de căutare

Efecte:

Ef1: Afișarea rezultatelor care conțin "pește" în ingrediente

Ef2: Răspunsul interfeței la căutare



user story 2: utilizatorul se afla deja pe pagina de produse si doreste sa reseteze cautarile

Cauze / Conditii intrare:

C1: Utilizatorul se află pe pagina de produse - acesta este punctul de pornire al acțiunii.

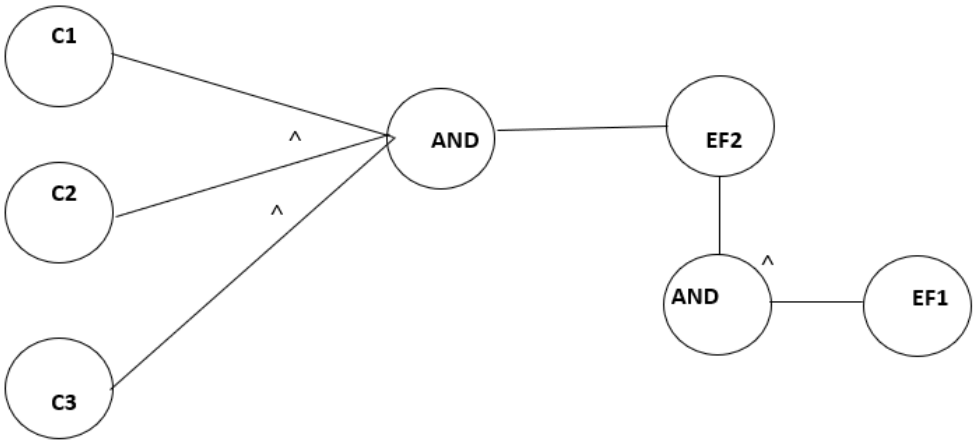
C2: Funcționalitatea de căutare este activă - câmpul de căutare și butonul de reset sunt prezente și accesibile.

C3: Utilizatorul apasă butonul de resetare - acțiunea specifică prin care utilizatorul solicită ștergerea criteriilor de căutare introduse anterior.

Efecte:

Ef1: Criteriile de căutare sunt șterse - câmpul de căutare trebuie să fie gol după acțiune.

Ef2: Pagina de produse este reîncărcată cu toate produsele afișate - arată lista completă de produse fără filtre aplicate.



user story 3: utilizatorul se afla pe pagina de produse si doreste sa afiseze produsele care au pretul mai mare decat cel minim afisat

Cauze / Conditii intrare:

C1: Utilizatorul se află pe pagina de produse - acesta este punctul de pornire al acțiunii.

C2: Funcționalitatea de filtrare după preț este disponibilă - existența unui câmp de filtrare sau unui slider pentru ajustarea prețului minim.

C3: Prețul minim afișat este vizibil pentru utilizator - utilizatorul poate vedea prețul minim actual pe pagina.

C4: Utilizatorul setează un preț minim de filtrare - utilizatorul introduce sau selectează un nou prag de preț minim care este mai mare decât cel afișat inițial.

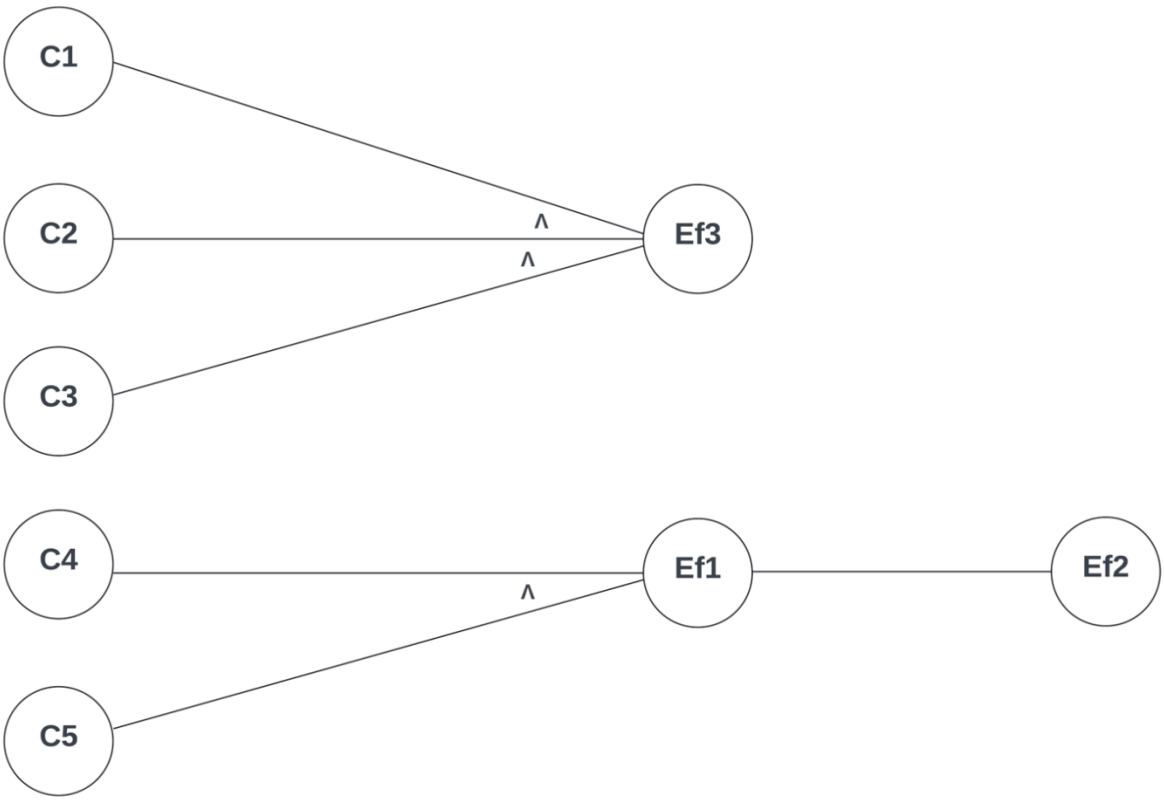
C5: Utilizatorul apasă butonul de aplicare a filtrului - acțiunea specifică prin care filtrul este activat.

Efecte:

Ef1: Lista produselor filtrate este afișată - numai produsele cu preț mai mare decât valoarea minimă setată sunt afișate.

Ef2: Produsele cu preț sub minimul setat nu sunt afișate - filtrarea exclude produsele cu prețuri mai mici.

Ef3: Interfața confirmă aplicarea filtrului - opțional, poate fi afișat un mesaj sau un alt indicator vizual care confirmă că filtrul a fost aplicat.


C1
C2
C3
C4
C5
Ef3
Ef1
Ef2