Projektni zadatak

Testiranje mobilne Android aplikacije

Recipe Helper

Zvonimir Medak, 28.1.2020.

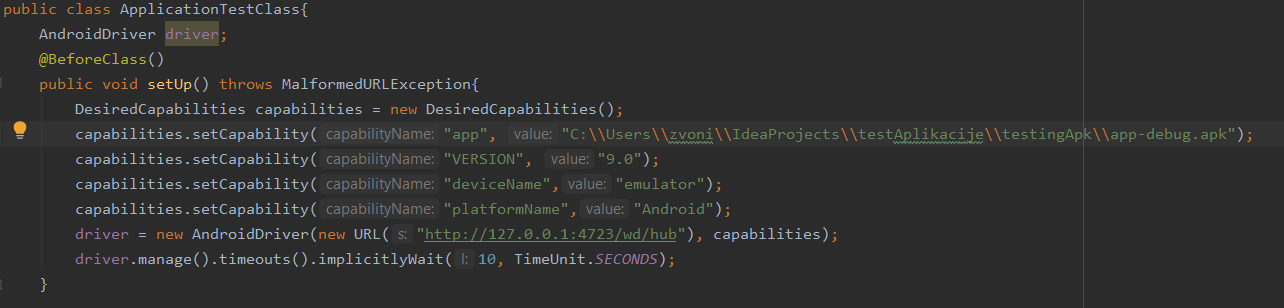
**UVOD**

Zadatak projekta je napraviti automatsko testiranje mobilne Android aplikacije. Zadatak je odrađen uz pomoć navedenih stvari u Readme fileu na gitu. Glavni dio se obavlja preko Selenium drivera i Appiuma koji omogućavaju cijeli postupak upravljanja aplikacijom. Testni slučajevi su pisani u Java programskom jeziku u IDE-u IntelliJ. Testirano je na Pixel 3, API 28 (9.0 Android). Napisano je sedam testnih slučajeva koji se odvrte automatski kada se pokrene projekt.

Aplikacija nudi login/registraciju korisnika kako bi se krenuli dohvaćati recepti sa <https://api.edamam.com>. Nakon toga, aplikacija omogućava pregled dohvaćenih recepata, spremanje recepata u lokalnu bazu i odlazak preko web browsera na web stranicu recepta.

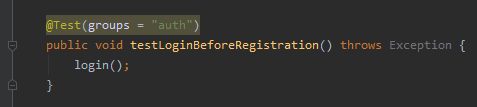
**TESTOVI I POSTAVLJANJE**

Kako bi se mogao pokrenuti cijeli proces testiranja potrebno je prvo namjestiti u *setUp()* postavke za driver. U funkciji se postavljaju capabilities u kojima je potrebno specificirati putanju aplikacije koja se testira, verziju Androida/iOS-a na kojem se testira, naziv emulatora/simulatora te naziv platforme Android/iOS. Ovo je potrebno navoditi je li Android ili iOS jer podržava testiranje za obje platforme. Nakon postavljanja potrebno je inicijalizirati driver.



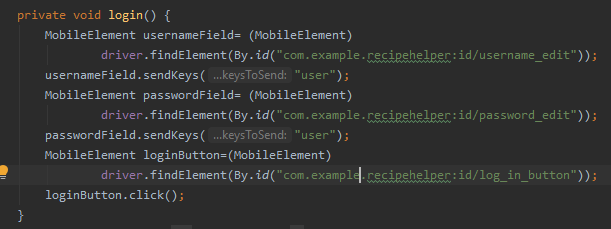
Slika 1, prikaz setUp() funkcije

Nakon postavljanja drivera u *setUp()* funkciji koja ima anotaciju @BeforeClass što označava da se treba pozvati funkcija prije svakog testa. Dolazimo do prvog testa koji se zove *testLoginBeforeRegistration().* Iz imena se može zaključiti da test samo provjerava postoji li takav korisnik već u bazi prije registracije. Očekivano stanje je da ne postoji i da aplikacija ne bi trebala ulogirati korisnika. Test poziva funkciju *login* koja je izvučena kako bi se mogla reusati u još jednom testu koji je identičan samo se odvija nakon registracije korisnika. Iznad svakog testa potrebno je postaviti @Test, ali postoje dodatne oznake koje se mogu postaviti kao groups ili dependsOn ako je potrebno da se neki test prvi izvrši pa onda drugi. Ovisno o tome što nam treba možemo ih postavljati u grupe ili stavljati dependsOn, a defaultno je da idu abecedno.



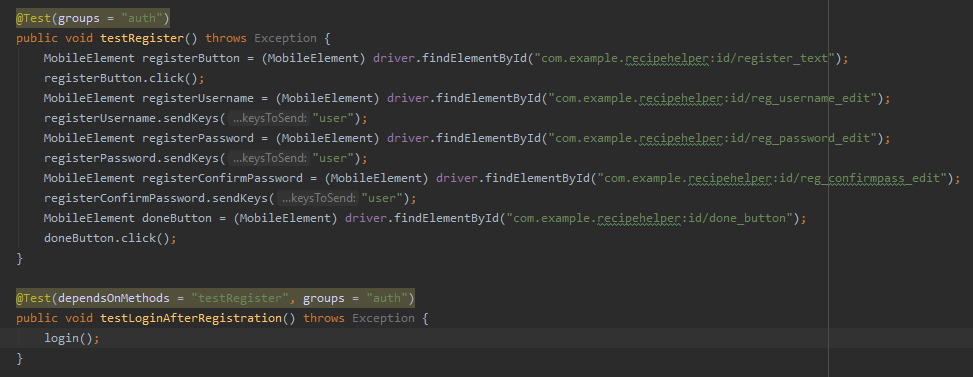
Slika 2, prikaz testLoginBeforeRegistration funkcije

Funkcija *login()* pronalazi username i password textField uz pomoć *driver.findElement(By.id())* funkcije koja uzima id elementa sa screena. Nakon što su pronađeni, preko funkcije *.sendKeys(„“)* se upisuje željeni tekst u textField. Nakon toga se na isti način pronalazi i loginButton nad kojim se poziva funkcija *.click()* kojom se odrađuje tap na button.



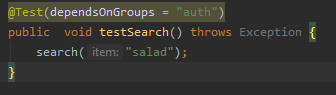
Slika 3, funkcija login

Nakon neuspjelog logiranja poziva se drugi test koji testira registraciju i registrira korisnika. Funkcija na isti način dohvaća elemente kao i u svakom drugom testu preko id-ja. Prvo pronalazi registerButton na početnom screenu gdje poziva klik. Nakon toga pronalazi username, password i confirm password text Fieldove u koje unosi „user“ i stisne button done. Nakon što se se to odradi, kao što je vidljivo iznad funkcije *testLoginAfterRegistration()*, može se pokrenuti login nakon registracije jer je *testLoginAfterRegistration()* ovisio o funkciji *testRegister().* Tu je očekivano stanje da će se korisnik uspjeti ulogirati s novonastalim accountom.



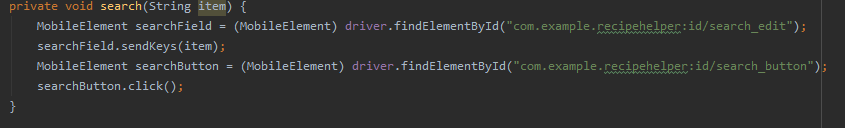
Slika 4, testovi registracije i login nakon registracije

Nakon uspješnog logina korsinika se odvodi na pager je se dohvaćaju recepti i postoji search bar. Prvi test bit će testiranje search bara *testSerach()* koji poziva funkciju *search(String item)*. Kao što je vidljivo na slici ispod, test ovisi o grupi „auth“ što znači da cijela grupa „auth“ se mora izvršiti kako bi ovaj test uopće započeo.



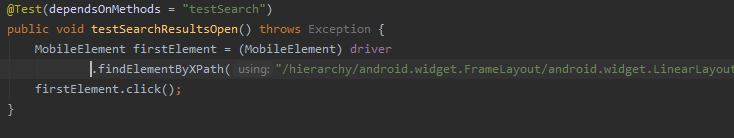
Slika 5, test searcha

Funkcija search pronalazi searchField u koji upisuje predani string koji je dobila kroz parametar. Nakon toga klikne na searchButton.



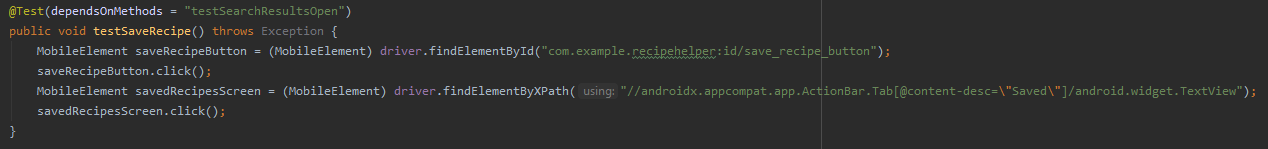
Slika 6, search funkcija

Sljedeći test je otvaranje details screena o receptu. On ovisi o testu *testSearch()* kako bi imali rezultate nakon upisivanja u search bar. Odabire prvi element i otvara ga. Željeno ponašanje je da se otvori details screen recepta.



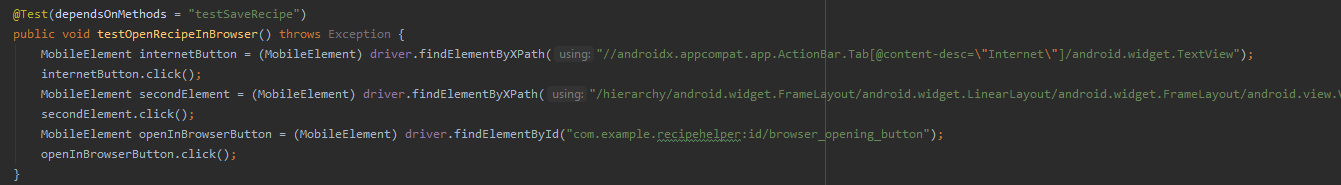
Slika 7, test otvaranja details screena

Nakon njega želimo testirati spremanje recepta u lokalnu bazu podataka. To se odvija u *testSaveRecipe()* testu koji pronalazi saveRecipeButton i klikne na njega čime dolazi na početni pager i odabire savedButton i klikne na njega kako bi se korisnika odvelo na screen gdje se prikazuju spremljeni recepti. Željeno ponašanje je da se pronađe spremljeni recept.



Slika 8, test spremanja recepta u lokalnu bazu

Nakon spremljenog recepta želimo testirati i otvaranje linka recepta u web browseru što se odvija preko testa *testOpenRecipeInBrowser()*. Pronalazi internetButton na pageru gdje odlazi na recepete s interneta i odabire drugi element. Nakon što se otvori details screen odabire button openInBrowser što otvara url na recept u browseru.



Slika 9, test otvaranja recepta u browseru