E2E testiranje portala MSNR

Opis sistema

Seminarski rad u okviru kursa
Verifikacija softvera
Matematički fakultet

Anđela Križan 1083/2020
Ivana Cvetkoski 1111/2021
Jelena Jeremić 1099/2021

Sadržaj

1 Opis problema

Cilj rada je bio izvršiti **end-to-end testiranje** (skraćeno e2e) portala posvećenog predmetu *„Metodologija stručnog i naučnog rada"* (skraćeno MSNR), otkriti potencijalne propuste i greške u implementaciji ovog portala i ponuditi rešenje za njih. Dodatno, neophodno je bilo istražiti biblioteke i alate pogodne za korišćenje kako bi se ova vrsta testova mogla implementirati.

2 Opis arhitekture sistema

Portal MSNR je zamišljen kao veb aplikacija za praćenje aktivnosti i ispunjavanja obaveza studenata tokom kursa Metodologija stručnog i naučnog rada. Arhitektura portala MSNR predstavlja tipičan primer troslojne arhitekture. Na klijentskoj strani se nalazi korisnički interfejs koji je implementiran u jeziku **Elm**, središnji sloj predstavlja aplikacioni veb interfejs napisan u jeziku **Elixir**, a treći sloj čini relaciona baza podataka. Odabrani sistem za upravljanje bazom je **PostgreSQL**.

2.1 Serverski deo portala

Programski jezik Elixir, kao jezik opšte namene, nema ugrađenu podršku za razvoj veb aplikacija i za tu svrhu se najčešće koristi razvojni okvir **Phoenix**, u kom je i implementiran aplikacioni veb interfejs ovog portala. Razvojni okvir Phoenix napisan je na programskom jeziku Elixir. Instalacijom Elixir-a instalira se i osnovni alat za rad sa Elixir projektima — **Mix**, koji se koristi za kreiranje, kompajliranje i testiranje projekta.
Projekat ima konfiguracionu datoteku **mix.exs**, direktorijum **lib** koji sadrži osnovni kod i direktorijum **test** sa testovima. Datoteka mix.exs sadrži osnovne informacije o projektu — kako se kompajlira, pokreće i listu zavisnih paketa koji se koriste u projektu. Razvojni okvir Phoenix dodaje i konfiguracije vezane za okruženje u kom se aplikacija izvršava. Podržana okruženja su razvojno (dev.exs), testno (test.exs) i produkciono (prod.exs) i konfiguracije su smeštene u direktorijumu **config**, zajedno sa glavnom konfiguracijom (config.exs) koja se odnosi na sva okruženja. Pored navedenih u direktorijumu config nalazi se i runtime.exs konfiguracija zadužena za učitavanje šifri (eng. secrets) i drugih konfiguracionih vrednosti iz promenljivih okruženja.

2.2 Klijentski deo portala

Implementacija korisničkog interfejsa portala izvršena je u programskom jeziku Elm. Elm je statički tipiziran, čisto funkcionalni programski jezik i namenjen je isključivo za kreiranje veb aplikacija. Takođe, Elm nije samo programski jezik već i platforma za razvoj aplikacija.
Osnovna podela aplikacije jeste na: stranice koje se koriste za prijavljivanje i registraciju korisnika, studentsku stranicu i profesorske stranice.

3 Opis rešenja problema

3.1 Podešavanja konfiguracionih datoteka projekta

Za pogodnu biblioteku koja nam omogućava automatizaciju pretaživača i pisanje e2e testova, odabrana je Elixir-ova biblioteka **Hound**. Hound komunicira sa odabranim webdriver-om i omogućava nam da pristupamo elementima grafičkog korisničkog interfejsa. Unutar direktorijuma sa testovima dodat je direktorijum **e2e** u kome su smešteni end-to-end testovi. Kako bi se ovi testovi mogli pokrenuti, unutar datoteke *mix.exs* bilo je potrebno dodati biblioteku Hound u listu zavisnih paketa:

```
{:hound, "~> 1.0"}
```

U datoteci test.exs, vrednost promenljive server moramo postaviti na true kako bismo osigurali da je server pokrenut prilikom pokretanja testova. Nakon toga, u direktorijumu test, u prvoj liniji datoteke *test_helper.exs*, potrebno je dodati narednu liniju:

```
Application.ensure_all_started(:hound)
```

Alat Mix će učitati ovu datoteku pre nego što pokrene testove.
Kako bismo mogli ostvariti komunikaciju sa pretraživačem u kom vršimo testiranje neophodan nam je **webdriver**. Iz tog razloga se u konfiguracionim datotekama test.exs i config.exs dodaje sledeća linija u kojoj navodimo koji ćemo webdriver koristiti:

```
config :hound, driver: "odgovarajući webdriver"
```

3.2 Testiranje

Kako profesorska stranica sadrži veći broj podstranica, a time i veći broj funkcionalnosti za testiranje, više testova je napisano za profesorske stranice nego za studentske.

3.2.1 Testiranje profesorskih stranica

Prilikom testiranja profesorskih stranica, uočeno je da na podrstranici za pravljenje tema (*url: /professor/topics*), prilikom pokušaja brisanja bilo koje postojeće teme na stranici, iskočila bi greška da ta tema ne postoji u bazi.
Problem je bio taj da putanja za brisanje teme nije bila ispravna i prilikom pritiska dugmeta za brisanje teme, nije se moglo pristupiti traženoj temi kako bi se izbrisala iz baze. Da bi se greška uklonila, bilo je potrebno izmeniti dve datoteke. Unutar datoteke **Api.elm** je dodata putanja koja će direktno pristupiti listi svih tema. U datoteci **TopicsPage.elm** je prepravljena putanja koja se postavlja prilikom poziva funkcije za brisanje teme tako što se na putanju do liste svih tema dodavao identifikator teme kojoj želimo pristupiti kako bismo je obrisali.

3.2.2 Testiranje studentskih stranica

Prilikom testiranja studentskih stranica nisu uočene greške koje bi bilo potrebno razrešiti.

3.3 Ideje za unapređenje testova

Većina testova je morala vršiti neku izmenu u bazi kako bi se određena funkcionalnost testirala što je značilo da svaku izmenu koju bi test napravio u bazi bi posle trebalo obrisati i vratiti na stanje pre pokretanja testa. Jedno od rešenja bi bilo pokušaj pravljenja privremene baze koja bi postojala samo tokom izvršavanja testa i koja bi se nakon završetka testa brisala sa diska, čime bismo sprečili bilo kakve izmene u pravoj bazi prilikom celog procesa testiranja.