Razvoj web aplikacija podržan testiranjem pomoću alata Selenium

Tonkić, Josip

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Josip Tonkić

Razvoj web aplikacija podržan testiranjem pomoću alata Selenium

ZAVRŠNI RAD

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Josip Tonkić

Matični broj:

Studij: Primjena informacijske tehnologije u poslovanju

Razvoj web aplikacija podržan testiranjem pomoću alata Selenium

ZAVRŠNI RAD

Mentor:

prof. dr. sc. Dragutin Kermek

Varaždin, rujan 2020.

Josip Tonkić

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada korištene su etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sadržaj

1. Uvod

Razvijanje web aplikacije ili obične aplikacije zahtijeva puno vremena te je cijeli proces dosta kompleksan. Testiranje aplikacija proces je koji se obavlja prije nego aplikacija koju stvaramo bude u tzv. produkciji, tj. kada proizvod šaljemo do krajnjeg korisnika. Testiranjem se pokušavaju pronaći greške u aplikaciji kako bi aplikacija bila što kvalitetnija. Ipak, uvijek se može dogoditi da postoje greške, zato je potrebno provjeravati i testirati aplikacije. Samim testiranjem podiže se zadovoljstvo korisnika jer u sustavu ili aplikaciji koja se koristi neće biti grešaka te će korisnici moći nesmetano koristiti aplikaciju. Testiranje se može podijeliti na dvije vrste: automatsko i ručno testiranje.

Ručno testiranje je testiranje gdje „tester“ testira aplikaciju ručno te provjerava sve funkcionalnosti aplikacije prema specifikaciji, tj. dokumentaciji aplikacije.
Kod automatskog testiranja radi se o pisanju programskog koda koji sve to obavlja za nas. Napiše se programski kod koji će uz pomoć tzv. lokatora locirati elemente na koje treba kliknuti te zatim sam istestirati određene funkcionalnosti aplikacije. Automatskim testiranjem smanjuje se vrijeme testiranja, smanjuje se mogućnost propusta greške te se time povećava kvaliteta aplikacije.

2. Uvod u web aplikacije

Često se pojmovi kao što su web aplikacija ili web stranica smatraju istima. Web aplikacije su interaktivne te komuniciraju s bazom podataka. Web aplikacije sastoje se od više programskih jezika jer postoji klijentska strana na kojoj se koriste HTML, CSS i JavaScript te također PHP u kojem se pišu programske skripte. One komuniciraju s bazom podataka u kojoj se pišu upiti u SQL-u. U PHP-u postoje dvije metode kao dva osnovna zahtjeva koje korisnik može slati poslužitelju, a to su GET i POST metode.

GET metoda služi kako bi se dobio zahtjev od nekog resursa poslužitelja.
POST metoda služi kako bi se poslao podatak resursu poslužitelja.

„Web aplikacija je softverski program koji se izvršava na web (http) serveru. Za razliku od tradicionalnih, desktop aplikacija koje se pokreću vašim operativnim sustavom, web aplikacije se najčešće pokreću uz pomoć "web browser" programa (Chrome, Firefox, Opera, itd...)." [1] Web aplikacija sastoji se od dviju komponenata: klijenta i poslužitelja. HTML se može koristiti kao kostur aplikacije što znači da je to prezentacijski jezik. HTML se koristi kako bi se posložile web stranice u određene elemente, tj. oznake (eng. *tag*), kako bi se osmislile početne pozicije elemenata na stranice te njihove klase, identifikatori itd. „On služi samo za opis naših hipertekstualnih dokumenata. HTML datoteke zapravo su obične tekstualne datoteke, nastavak im je .html ili .xhtm. Osnovni građevni element svake stranice su oznake koje opisuju kako će se nešto prikazati u web pregledniku. Poveznice unutar HTML dokumenata povezuju dokumente u uređenu hijerarhijsku strukturu i time određuju način na koji posjetitelj doživljava sadržaj stranica." [2]

Osnovna struktura web aplikacije, tj. programskog koda HTML-a sastoji se od elemenata, a svaki element sastoji se od para HTML oznaka. Naravno, HTML dokument mora početi s <!DOCTYPE> oznakom koja označava deklaraciju dokumenta, tj. koja se inačica HTML-a koristi, a to će u ovom slučaju biti HTML 5. Zatim postoji <html> element koji označava početak dokumenta i </html> element koji označava završetak dokumenta. Postoje i četiri njegova glavna elementa: *header, body, header i footer*. Element <header> također ima svoj završni element </header> u kome se nalaze metapodatci web dokumenta kao što je „utf-8". U elementu <body> nalazi se npr. cijela web aplikacija te je u njemu i <header> koji također ima svoj završni element </header> koji predstavlja zaglavlje web aplikacije. Element <footer> sa

završnom oznakom </footer> predstavlja podnožje web aplikacije. Kada je sve to određeno te se napiše programski kod HTML-a, slijedi drugi programski jezik koji će se integrirati u HTML kod, a to je CSS. „**CSS** je kratica izraza *Cascading Style Sheets*. Radi se o *stilskom* jeziku koji se rabi za opis prezentacije dokumenta napisanog pomoću *markup* (HTML) jezika.

Kako se web razvijao, prvotno su u HTML ubacivani elementi za definiciju prezentacije (npr. oznaka <font>), ali je dovoljno brzo uočena potreba za stilskim jezikom koji će HTML osloboditi potrebe prikazivanja sadržaja (što je prvenstvena namjena HTML-a) i njegovog oblikovanja (čemu danas služi CSS). Drugim riječima, stil definira kako prikazati HTML elemente. CSS-om se uređuje sam izgled i raspored stranice.“ [3]

CSS služi za pisanje i izradu dizajna web stranice ili web aplikacije. CSS kod piše se u HTML dokumentu ako se napravi element <style> koji ima završni element </style> ili se piše u drugom dokumentu koji ima nastavak .css te se on zatim u *headeru* implementira u CSS *text/style* dokument. Potrebno je u dokumentu naznačivati elemente kao što su ID ili klasa putem kojih se mogu locirati elementi u CSS-u. Primjerice, ako element ima u sebi , u CSS-u se pristupa elementu tako da se upiše *#meni* te se na taj način pristupi klasi *meni*, a programski kod piše se unutar vitičastih zagrada koje također moraju imati završni par. Ista se odnosi i na id kojim se pristupa pomoću točke, u ovom slučaju *.meni* {„ovdje pišemo CSS programski kod“}.

Nakon što je završen dizajn web aplikacije potrebno je u sve to implementirati PHP programski jezik u kojem se pišu skripte koje će pretvoriti web stranicu u web aplikaciju. „PHP (skraćenica za „Personal Home Page Tools“) jedan je od programskih jezika koji se orijentira po C i PERL sintaksi namijenjen prvenstveno programiranju dinamičnih web stranica. PHP je kao slobodni software. PHP se ističe širokom podrškom raznih baza podataka i internet protokola kao i raspoloživosti brojnih programskih knjižnica.“ [4]

PHP, kao i CSS, može se pisati implementirano u HTML programski jezik ako se postavi oznaka za otvaranje PHP programskog koda „<?php“ i „?>“ kao oznaka za zatvaranje programskog koda. Također, postoje i skripte koje su odvojeno napisane te se pozivaju i imaju nastavak .php. U PHP-u se može pisati i HTML kod te se može spojiti PHP i SQL kako bi se dobivali podatci iz baze podataka.

Primjer kratke PHP skripte:

```
<?php
session_start();
if(!isset($_SESSION["id"])){
     header("Location:non_reg_non.php");
}
include_once("iwa_2019_vz_projekt.php");
$veza=mysqli_connect($server, $korisnik, $lozinka, $baza);
$upit=" SELECT v.naziv, s.sredstva_id, v.slika ,s.iznos,
s.korisnik_id, v.valuta_id FROM sredstva s, valuta v
  WHERE s.valuta_id = v.valuta_id AND s.korisnik_id
='{$_SESSION["id"]}'";
$rezultat = mysqli_query($veza, $upit);

$upit1=" SELECT naziv, valuta_id FROM valuta";
$rezultat1 = mysqli_query($veza, $upit1);

                ?>
```

2.1. Osnove HTML-a

Sada kada je objašnjeno kako funkcionira web aplikacija te od kojih se sve programskih jezika sastoji, u daljnjem tekstu proći će se kroz osnove svakog jezika, njihovu povijest i elemente. Prvi je na popisu HTML kao temelj cijele web aplikacije ili web stranice.

2.1.1. Povijest HTML-a

Tim Barners-Lee je 1989. godine razvio HTML na CERN-u u Švicarskoj. U to vrijeme Barners-Lee nije znao da će napraviti nešto na svjetskoj razini. Ono što je on htio postići je napraviti jedno mjesto na kojemu će se moći udaljeno pristupati dokumentima, tj. jedno mjesto gdje će biti spremljeni svi dokumenti različitih istraživanja. Iako je to bio savjet drugih, on je htio napraviti poveznice u dokumentima kako bi se korisnicima, čitajući jedno istraživanje, klikom na određenu riječ mogao prikazati dio drugog istraživanja kao dodatak. Tako bi se moglo „skakati“ s dokumenta na dokument. 1994. godine nastaje W3C, tj. „World Wide Web

Consertium“ te imaju veliki interes za HTML. Grupa koja izrađuje HTML je u studenom 1995. godine otkrila probleme. Naime, sve o čemu su oni razgovarali u teoriji je odlična ideja, ali praksu je bilo teže izvoditi, stoga se raznim nadogradnjama došlo do HTML 2.0 inačice. Kasnije nastaje inačica 3.0 koju objavljuje W3C i u kojoj je dodatak definiranje tablica.

„Daljnji razvoj ove verzije HTML-a označilo je prihvaćanje "specifičnih" oznaka podržanih u tada najvećim i najprihvaćenijim web preglednicima. Tako su nastale mnoge duplikacije pa je postojalo više oznaka koje su imale istu funkciju. Podebljani tekst, primjerice, bilo je moguće definirati oznakom, ali i oznakama.“ [2] Nakon niza događaja 1997. godine izlazi HTML 3.2 koji objavljuje W3C. Inačica 4.0 pojavljuje se u prosincu 1997. godine, 4.01 u prosincu 1999. godine te XHTML 1.0 u siječnju 2000. godine. Možemo reći da se HTML sastoji od specifikacija od kojih je zadnju 1999. godine izdao W3C. Sada dolazi nova verzija HTML-a koja se temelji na XML jeziku. Zatim se počinje razvijati novi HTML 5 koji ima nove funkcionalnosti, npr. može dodati video, audio, canvas. Mogu se dodati i novi elementi kao što su <main>, <aside>, <nav>, <figure> itd. Njegovim daljnjim razvojem nastala je HTML 5.2 inačica koja je izašla 2017. godine. Može se reći da se HTML proslavio samo zato što se proslavio i web. Ako na početku web stranice vidimo <!DOCTYPE html>, to označava DTD, što znači da se radi o HTML 5 dokumentu. “World Wide Web dokument (stranica) služi za uspostavljanje poveznica među dokumentima (podrazumijeva se da dokument sadrži tekst, sliku, zvuk, grafiku).“ [5]
XHTML se sastoji od elemenata, atributa te entiteta. Elementi su blokovi koji imaju početnu i završnu oznaku. Neki imaju samo početnu, tj. sami se zatvaraju, tipa <input /> ili <img> dok , <footer>, <label> i mnogi drugi moraju imati svoj završni par.

2.1.2. Oznake i atributi

Atributi se dodaju elementima kako bi atributi dobili značenje. Primjerice, ako se elementu <img> pridruži atribut „src“, tada se atributom „src“ pridružuje lokacija slike na ovaj način: src=“slika1.jpg“. Ovim načinom govori se elementu <img> da je lokacija slike apsolutna te se nalazi u datoteci gdje je HTML i da je slika1. ime slike. Nadalje, postoji i relativna putanja gdje se u atribut „src“ zadaje drugačija putanja do slike, kao naprimjer src=“jtonkic/Dokumenti/slika1.jpg“. Zatim se može dodati atribut „alt“ koji govori kako će se zvati slika te koji će se tekst pokazivati na mjestu slike. Primjer navedenog atributa: alt=“Moja

prva slika“. Slici se još može dodati *width* i *height*, kao visina i širina slike, na isti način kao i prije, witdh=“100“ i height=“100“. Svi elementi nemaju iste atribute. Svaki element ima atribute za sebe koje može dodati po određenoj, zadanoj sintaksi.

Naprimjer, element <input>:

```
<input class="korime" name="korime" type="text"/>
```

Ovdje je pridružena klasa po kojoj se kasnije može pozvati kako bi se uredili elementi u CSS-u. Dodano je ime elementu pod *name* s nazivom *korime* te kakav je tip podatka koji se unosi u element, koji je u ovom slučaju *text*. Tip može također biti i *number, time, date* itd.

Primjer web dokumenta

<head> ----------→ Ovdje se pišu metapodatci, naziv web stranice, tko ju je uredio i ostalo te označava početak elementa.

</head> ----------→ Ovdje završava element head.
<body> ----------→ Ovdje počinje tijelo web stranice.
<header> ----------→ Ovdje počinje tijelo zaglavlja stranice (u njemu se treba sadržavati element <style> za CSS).

</header> ----------→ Ovdje završava tijelo web stranice.
</body> ----------→ Ovdje završava tijelo web stranice <footer>.
</footer> ----------→ Ovdje završava tijelo web stranice.
</html> ----------→ Ovdje završava tijelo web stranice.

Moja prva web stranica

```
<head>  <title> Moja prva web stranica  <title>
   </head>
 <body>
<header>
</header>
<p> Ovdje se nalazi nekakav tekst! <p>
</body>
</footer>
</html>
```

Gore napisan kod za HTML predstavlja web stranicu koja će imati ispisan tekst u paragrafu „p“ „Ovdje se nalazi nekakav tekst“. U elementu <title> nalazi se naslov web stranice, a to je „Moja prva web stranica“. Zaglavlje i podnožje ostavljeni su prazni.

2.2. Osnove CSS-a

CSS je kratica izraza *Cascading Style Sheets*. Radi se o stilskom jeziku kojem je cilj oblikovati i pozicionirati elemente HMTL-a. Kako je navedeno, HTML je početak stvaranja web aplikacije. Web aplikaciji ili web stranici se pomoću CSS-a ne pridodaju nikakve funkcionalnosti, već se samo izrađuje stilski list (eng. *style sheet*), tj. dokument u kojem se nalazi CSS stilska uputa. Kasnije će se vidjeti da se CSS može koristiti na više načina, tj. da može biti i implementiran u HTML izvorni kod.

2.2.1. Povijest CSS-a

Hakon Wium Lie je prvi predložio CSS 10. listopada 1994. godine. Tijekom toga vremena radio je i u CERN-u s Timom Barners-Leeom. Tada je već postojalo nekoliko stilskih jezika, ali niti jedan nije bio s HTML-om. Kasnije ga izdaju u suradnji s Bert Bosom te on postaje suosnivač CSS-a. Kako je popularnost interneta rasla te se njime koristilo sve više korisnika, tako je rasla i sama potreba da se poboljša CSS te stvaraju nove razine CSS-a.

CSS 1

CSS 1 dolazi kao prva specifikacija koja je postala službena preporuka W3C-a, a u produkciji je od 17. prosinca 1996. godine. Hakon Wium Lie i Bert Bos osnivači su CSS-a 1.
CSS 1 imao je mogućnosti poput:

- različitog fonta,
- promjene boje teksta, pozadine i drugih elemenata,
- postavljanja razmaka između riječi, slova i linija u tekstu,
- poravnanja teksta, slika, tablica i svih drugih elementa,
- margine, granice elemenata te pozicioniranja samih elemenata,
- postavljanja id-a ili klasifikacije nekog elemenata.

CSS 2

CSS 2 ima sve što ima i njegov prethodnik, a izdan je u svibnju 1998. godine. Kao nadogradnju u sebi ima mogućnost apsolutnog i relativnog pozicioniranja te z-index. Z-index omogućuje da se elementi postave i po z osi, u smislu da jedan element može biti iznad drugog ili suprotno. Također, moguće je dodavati više medija, poput glazbe itd.

CSS 2.1

Nastavak, tj. dodatak na prošlu CSS razinu je CSS 2.1 koji u sebi ima popravke svih grešaka koje je sadržavao CSS 2. Obrisani su i dodatci koji nisu dobro implementirani ili su slabo razvijeni. Izdan je 7. lipnja 2011. godine.

CSS 3

U ovom nastavku, tj. razini CSS-a postoje specifikacije koje su podijeljene u različite dokumente koji se nazivaju moduli. Svaki modul dodaje novu vrijednost ili obogaćuje prethodnu funkcionalnost. Ova inačica obuhvaća puno pretraživača na svim operacijskim sustavima.

Implementacija CSS-a u kod

Implementiranje CSS-a može se odvijati na više načina. Možemo ga implementirati jednom u dokument, po potrebi u liniji ili u vanjskoj datoteci.

2.2.2. Osnove i sintaksa CSS-a

Smještaj CSS uputa jednom u dokument

Kao što i sam naslov govori, smještanje CSS uputa jednom u dokument znači da će se CSS upute jednom implementirati u HTML, kao što je prikazano u primjeru ispod.
U CSS uputama vidljivo je da je vrijednost atributa *style text/css*, iz čega se može shvatiti da se radi od CSS uputama. To je osnovna vrijednost koja se upisuje.

```
 <html>
<head>
<style type="text/css">
body {
background-color:red;
h1{
color:red;
}
}
 </head>
     <body>
          <header>
<h1> Ovo je moj naslov </h1>

          </header>
     </body>
<footer>
</footer>
</html>
```

Smještaj CSS uputa po potrebi u dokumentu

CSS upute mogu se upisati u dokumentu po potrebi za svaki dio. Naravno, sintaksa CSS jezika identična je kao i za prethodni, ali postoje neke preinake.

```
<html>
<head>
 </head>
     <body style="color:red">
          <header>
<h1 style="color:red" > Ovo je moj naslov </h1>

          </header>
     </body>
<footer>
</footer>
</html>
```

Ovo je linijski način implementacije gdje se upisuju upute samo po potrebi. Također se mogu upisati u druge elemente poput <h2>, , <p>.

Smještaj CSS uputa u vanjsku datoteku

Smještanje CSS uputa u vanjsku datoteku radi tako da u HTML kodu uopće ne postoje ikakve CSS upute, već se one nalaze u zasebnoj datoteci u primjeru „upute.css“ te se datoteka implementira na cijeli HTML dokument.

```
<html>
<head>
<link href="upute.css " rel="stylesheet" type="text/css">

 </head>
     <body >
          <header>
<h1> Ovo je moj naslov </h1>

          </header>
     </body>
<footer>
</footer>
</html>
```

Ovim se načinom HTML kodu pokazalo gdje se nalazi datoteka, koja joj je relacija te koji je tip datoteke.
U upute.css može se nalaziti primjerice ovakav kod:

```
body {
background-color:red;

h1{
color:red;
}
```

Postoji više načina korištenja CSS uputa:

- implicitno
- eksplicitno
- jednoznačno
- po potrebi
- pseudo-klasa

Implicitno korištenje CSS uputa identično je smještaju CSS upute jednom u dokument gdje se u *head* piše <style> </style> te se unutar njega pišu CSS upute.

Eksplicitno korištenje CSS uputa funkcionira tako da se elementima, kao što su naprimjer <p>, , <h1> pridružuje klasa na sljedeći način:

```
<h1 class="naslov" >    </h1>
```

Sada se elementom <h1> može pozvati i korištenje klase „naslov“ tako da se u CSS uputama upiše .naslov jer se pomoću točke može pristupiti klasama.

```
.naslov{
color:red;
}
```

Jednoznačno korištenje CSS uputa

Sintaksa je identična eksplicitnom načinu gdje se dodaje klasa određenom elementu, samo što se kod jednoznačnog korištenja CSS uputa koristi id umjesto klase. Id-u se pristupa pomoću znaka „ljestve“ # , a klasi pomoću „točke“.

```
<h1 id="naslov" >   </h1>
#id{
color:red;
}
```

Korištenje CSS uputa po potrebi

CSS uputa koristi pseudoklase kod elemenata koji imaju unaprijed pripremljene poddijelove, a kojima se mogu promijeniti osnovne osobine koje se odvajaju simbolom :.
Najčešće se koristi oznaka a. [6]

```
<html>
<head>
<style type="text/css">
a:hoover {
color:red
}
</style>
 </head>
     <body >
          <header>
          </header>
     </body>
<footer>
</footer>
```

Korištenje CSS uputa po potrebi

```
<html>
<head>
 </head>
     <body style="color:red">
          <header>
<h1 style="color:red" > Ovo je moj naslov </h1>

          </header>
     </body>
<footer>
</footer>
</html>
```

2.3. Osnove JavaScript-a

„JavaScript je skriptni programski jezik, koji se izvršava u web pregledniku na strani korisnika. Napravljen je da bude sličan Javi, zbog lakšega korištenja, ali nije objektno orijentiran kao Java, već se temelji na prototipu i tu prestaje svaka povezanost s programskim jezikom Java.“ [7] Cilj je kreiranja jezika JavaScript bio da se doda interaktivnost HTML stranicama. JavaScript omogućuje mnogo toga. Temeljna funkcionalnost je reagiranje na događaje, tj. kada se klikne na element, aktivira se događaj. JavaScript ima i mogućnost promijeniti sadržaj nekog HTML dokumenta.

2.3.1. Povijest JavaScript-a

JavaScript je nastao 1996. godine. Razvio ga je Brendan Eich iz Netscapea pod imenom Mocha. Jezik se također zvao LiveScript te je kasnije preimenovan u današnje ime, JavaScript. 1997. godine ECMA objavljuje prvu inačicu nekog standarda za taj jezik. 1998. godine standard je prihvatio međunarodni ISO standard (ISO/IEC 16262). Prvi JavaScript koji je izašao bio je JavaScript 1.0, a zadnji, koji je izašao 2015. godine, bio je ECMA v6. Kao i svaki skriptni jezik, JavaScript se izvršava s interpreterom koji je implementiran u sve preglednike.

2.3.2. Osnove i sintaksa JavaScript-a

Kao što su jezici i do sada imali parove oznaka, tako ih također ima i JavaScript kao početnu oznaku. JS je kao i SQL osjetljiv na velika i mala slova, što znači da imena funkcija, varijable ili bilo što drugo mora biti napisano veličinom slova koja je i definirana. Također, JS radi pomoću blokova koji se označavaju vitičastim zagradama { }. Blokovi se koriste kada se skupina naredbi treba izvršiti ako je ispunjen neki uvjet ili ako je u funkciji. Za određivanje jezika, kao i kod CSS-a, koristi se *type* koji je u ovom slučaju *text/javascript*.

```
<html>
<head>
 </head>
     <body >
<script type="text/javascript">
document.write(„<p>JavaScript program")
</script>
          <header>

          </header>
     </body>
<footer>
</footer>
</html>
```

Također se JS datoteka može uvesti u HTML kod te bi sintaksa izgledala slično kao i kod CSS-a. U ovom će se slučaju implementirati skripta pod nazivom „datoteka.js“ koja u sebi sadrži kod koji je gore naveden: document.write („<p>JavaScript Program“).

```
<html>
<head>
 </head>
     <body >
<script type="text/javascript" src="datoteka.js">
</script>
          <header>

          </header>
     </body>
<footer>
</footer>
</html>
```

2.4. Osnove PHP-a

PHP je skriptni jezik te se izvodi na poslužiteljskoj strani. Koristi se da se naprave interaktivna web mjesta. PHP je besplatan te je otvorenog koda i podržava rad s raznim bazama podataka. MySQL je poslužitelj za bazu podataka koji može raditi s PHP-om neovisno o platformi na kojoj je instaliran. PHP je jedan od najkorištenijih poslužiteljskih alata danas. U projektu je također korišten PHP kao poslužiteljski jezik koji komunicira s bazom podataka te omogućuje da pretvorimo web stranicu u web aplikaciju koja radi s bazom podataka.

2.4.1. Povijest PHP-a

PHP je skriptni jezik nastao iz PHP/FI koji je 1995. godine izradio Rasmus Lerdorf tako da su kombinirane Perl skripte i skripte s njegovih stranica. Skraćenica PHP/FI znači „Personal Home Page Tools/Forms Interpreter“. Ova je inačica bila javna te je bila „otvorenog koda“ tako da su ju svi mogli koristiti i poboljšavati. 1997. godine dolazi novi PHP/FI 2.0 koji je napisan u C-u. „Tada ga je koristila grupa od nekoliko tisuća ljudi širom svijeta na oko 50,000 web stranica, to je otprilike bilo oko 1% internet domena u tom trenutku.“ [8] Iako je bio vrlo veliki projekt, i dalje ga je radio samo Rasmus Lerdorf. Sljedeće godine (1998.) izlazi PHP 3 u kojem više ne sudjeluje samo Rasmus Lerdorf, već mu se pridružuju Andi Gutmans i Zeev Suraski. Najveća promjena bila je u dodanim novim funkcionalnostima.

Bilo je moguće raditi s više baza podataka, protokola i API-a. „Tijekom 1998. godine PHP se širi na desetke tisuća korisnika i stotine tisuća web poslužitelja zauzimajući pri tome udio od oko 10% svih internet poslužitelja.“ [8] PHP 4 dolazi također 1998. godine. Gutmans i Suraski počeli su raditi na poboljšanju jezgre PHP-a. Službeno je tek izdan 2000. godine pod nazivom „Zend Engine“. „PHP 4 je trenutno zadnja verzija PHP-a. Danas PHP koriste stotine tisuća programera na skoro 10 milijuna web poslužitelja, odnosno nalazi se na preko 20% internet domena.“ [8] Nakon nje dolazi potpuno stabilna verzija PHP 5 u kojoj se omogućava objektno orijentirano programiranje, ali mu podrška prestaje 2019. godine te više nije za uporabu. Trenutna inačica PHP-a u produkciji je 7.4.7.

2.4.2. Osnove i sintaksa PHP-a

PHP kao i jezici u prethodnom tekstu mogu se na više načina implementirati u programski kod. PHP jezik može se pisati između oznaka. Oznake mogu biti postavljene bilo gdje u dokumentu neovisno o tome što one izvode. Osnovna pravila PHP-a su da svaka linija koda mora završavati znakom ; te mora biti između zadanih oznaka. Varijable u PHP-u počinju znakom $, npr. $varijabla=5;. Također, postoji još mnogo pravila, primjerice da se varijabla mora pisati malim početnim slovom te ne smije imati $_naziv jer se tako označavaju fiksne globalne varijable. Treba se znati i da naredba *echo* zna razlikovati jednostruke i dvostruke navodnike. Dvostruki navodnici pokazat će vrijednost varijable dok će jednostruki navodnici pokazati samo točno što piše u njima.

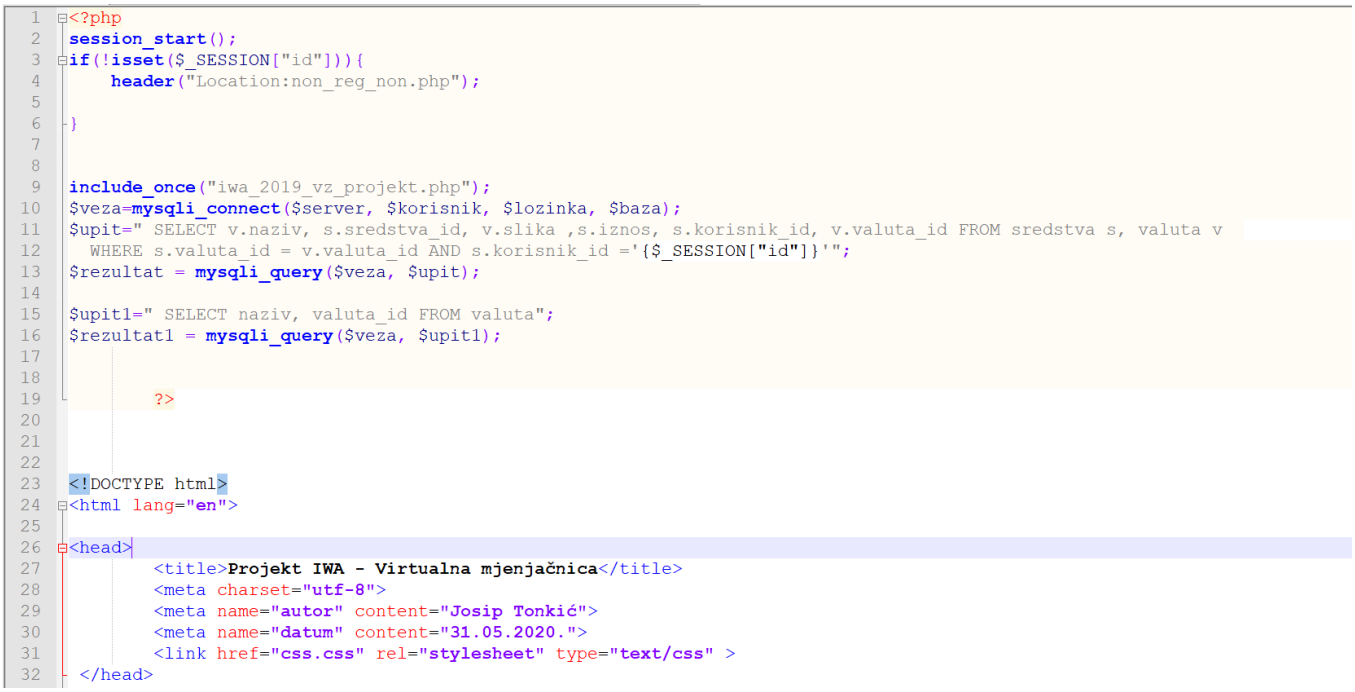
Primjer 1.

```
<?php
$broj1=1;
$broj2=2;

echo „$broj1 i $broj2“;
echo '$broj1 i $broj2';

?>
```

Crveni dio koda ispisat će brojeve 1 i 2.
Plavi dio koda ispisat će doslovno što piše, a to je $broj1 i $broj2.



```php
<?php
session_start();
if(!isset($_SESSION["id"])){
    header("Location:non_reg_non.php");

}


include_once("iwa_2019_vz_projekt.php");
$veza=mysqli_connect($server, $korisnik, $lozinka, $baza);
$upit=" SELECT v.naziv, s.sredstva_id, v.slika ,s.iznos, s.korisnik_id, v.valuta_id FROM sredstva s, valuta v
  WHERE s.valuta_id = v.valuta_id AND s.korisnik_id ='{$_SESSION["id"]}'";
$rezultat = mysqli_query($veza, $upit);

$upit1=" SELECT naziv, valuta_id FROM valuta";
$rezultat1 = mysqli_query($veza, $upit1);


        ?>


<!DOCTYPE html>
<html lang="en">

<head>
        <title>Projekt IWA - Virtualna mjenjačnica</title>
        <meta charset="utf-8">
        <meta name="autor" content="Josip Tonkić">
        <meta name="datum" content="31.05.2020.">
        <link href="css.css" rel="stylesheet" type="text/css" >
</head>
```

Slika 1. Implementacija PHP-a

Slika 1. prikazuje kako je PHP kod implementiran u HTML kod odmah na početku. Korisnik aplikacije to neće vidjeti, ali se to odvija u pozadini. U PHP kodu može se vidjeti kako se radi i sa SQL upitima gdje se iz baze podataka izvlače potrebni podatci. Na početku, session_start(); označava početak rada sa sesijama ako nije postavljena sesija s nazivom *id* da nas „vrati“ natrag na skriptu pod nazivom non_reg_non.php, dok se ispod nalaze SQL upiti koji su potrebni kasnije u HTML kodu. Ovim je samo prikazano da nije bitno gdje se nalaze te gdje su postavljene php skripte.

2.5. Web aplikacija ili web stranica

Prije odluke koja je bolja opcija te što bi i kada bilo najidealnije koristiti, potrebno je naučiti što je web stranica, a što je web aplikacija te koje su suštinske razlike. Kada se govori o web aplikaciji, to je „aplikacija“ koji se izvodi na nekom internetskom pregledniku i koja komunicira s bazom podataka.

Razlika između web stranice i web aplikacije nije jasno definirana, ali se može reći da web aplikacija koristi bazu podataka s kojom komunicira. Za web aplikaciju potrebno je i da ima

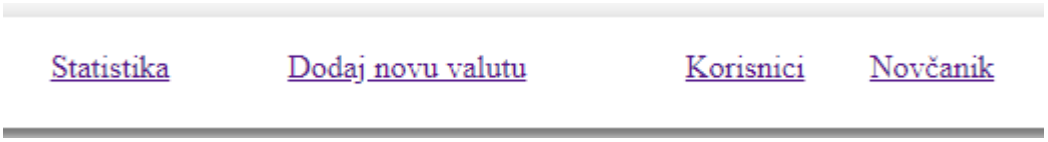
poslužitelja na kojem se ona obavlja. Jedan od poznatih i besplatnih alata je XAMPP poslužitelj (Slika 1.).

Web aplikacije su također napisane u više programskih jezika, kao što su HTML, CSS, JavaScript, te za poslužiteljsku stranu kao što je PHP. Web aplikacija može biti dinamička i statička. Ako se radi o dinamičkoj web aplikaciji, to znači da se ona izvodi na poslužitelju, a statička ne zahtijeva nikakvo procesiranje programskog koda na poslužitelju.
Web aplikacija radi na poslužitelju što predstavlja da poslužitelj upravlja zahtjevima koje je poslao korisnik u aplikaciji. Znači, ako korisnik pritisne tipku „Prijavi se“, u pozadini se odvijaju male programske skripte koje će upitati poslužitelja „Imamo li korisnika u bazi podataka s ovim korisničkim imenom i lozinkom?“ te poslati te podatke na provjeru. To je sve sumirano u 4 osnovna koraka:

- Korisnik šalje zahtjev web poslužitelju putem aplikacije „preglednika“.
- Web poslužitelj prosljeđuje takav zahtjev aplikaciji koja se nalazi na poslužitelju (XAMPP u ovom slučaju).
- Aplikacija izvodi zadatak, daje rezultat te ga šalje natrag web poslužitelju.
- Poslužitelj prosljeđuje rezultat klijentu te ga prikazuje na web pregledniku.

U današnje vrijeme, posebice u 2020. godini kada je nastupila pandemija virusom „COVID-19“, sve se više poduzeća mora osloniti na web stranice i online način rada.
Web stranica je stranica informativnog tipa koja može pružiti dobre informacije o pojedinim uslugama poduzeća.

„Internet stranica je skup stranica (dokumenata kojima se pristupa preko interneta). Internet stranica je ono što vidite na ekranu kada upišete web adresu, kliknete na poveznicu (link) ili navedete upit u neki od pretraživača (npr. Google).“ [9] Kao što definicija web stranice navodi, radi se o skupu dokumenata putem kojih se mogu dobiti informacije. Jedna web stranica može u sebi sadržavati 10-ak pa i puno više drugih web stranica, tj. drugih lokacija kojima se pristupa putem poveznica u navigaciji (Slika 2.).



Slika 2. Navigacijska traka web aplikacije

Ljudi posjećuju web stranice da dobiju potrebne informacije. Web stranice najčešće su pisane u HTML programskom jeziku, iako se mogu napraviti na dosta drugih načina kao što je WordPress. Web stranice pohranjene su na poslužitelju te je potrebno „kupiti“ mjesto na internetu gdje će se nalaziti web stranica, tj. *hosting*. Na web stranici vrlo je bitan i dizajn koji može privući ili odbiti korisnika da pretražuje web stranicu. Dizajn se može raditi od početka ili se može s WordPress stranice preuzeti predložak kako se na dizajn ne bi potrošilo previše vremena. Također je vrlo bitan i SEO, što predstavlja skraćenicu za eng. “Search Engine Optimization“, jer određuje da korisnici mogu lakše doći do web lokacije. Primjerice, ako web stranica ima dobar SEO, onda će biti prikazana visoko na stranici Google kada netko upiše npr. „online Virtualne mjenjačnice“. Ako web stranica ima dobar SEO, tada će biti među prvima. SEO radi pomoću oznaka, ali određuje kvalitetu sadržaja, „promet“ na vašoj web stranici te pretražuju li ju kvalitetni korisnici koji provode dovoljno vremena na vašoj web stranici.

2.5.1. Prednosti i nedostatci web aplikacija

Prednosti web aplikacija:

Za pristup i uporabu potreban je samo web preglednik te svi mogu pristupiti.

Uvijek je ažurirano - nije potrebno skidati nove inačice jer web aplikacija radi na poslužitelju i uvijek se prikažu najnovije promjene.

Pohrana - sve se nalazi na „oblaku“ te nije potrebno da aplikacija zauzima prostor na SSD-u ili tvrdom disku.

Informacije na jednom mjestu - sve informacije nalaze se na poslužitelju te je tako lakše njima pristupati.

Dostupnost - 0/24 sata.

Nedostatci web aplikacija:

Sigurnost - web aplikacije više su izložene napadima jer se nalaze na internetu, tj. poslužitelju, a ne na računalu.

Usporenost - web aplikacije rade na internetu, stoga u slučaju slabe brzine interneta može biti usporen i rad aplikacije.

Kompatibilnost - potrebno je napraviti da web aplikacija radi na više različitih operacijskih sustava i različitih web preglednika.

3. Automatsko testiranje i alati za testiranje

Za razliku od ručnog testiranja, automatsko je testiranje novije na tržištu te ima svoje prednosti i nedostatke. Što se tiče ručnog testiranja, čovjek uvijek može imati propust, da jednostavno ne vidi grešku zbog umora ili manjka koncentracije. Automatsko testiranje pomoću programskog koda, tj.“testnih slučajeva“ prolazi aplikacijom i traži greške. Tako da se može reći da su automatski testovi puno učinkovitiji od ručnog testiranja iako automatsko testiranje, za sada, ne može 100% zamijeniti ručno testiranje, ali može pokriti 60%, u najbolju ruku 80% posla ručnog testiranja, prema slobodnoj procjeni. Također, kada se jednom napiše testni slučaj za određeni scenarij ili funkcionalnost, može se pokrenuti bilo kada i neograničeno puta.

To je velika prednost za automatsko testiranje zbog toga što za taj određeni slučaj nije niti potreban ručni način testiranja koji će provjeriti grešku na aplikaciji jer će se sve odvijati automatski. Naravno, kako računalo nema mogućnost racionalnog te ljudskog razmišljanja, ne može zamijeniti ručno testiranje. Ako se uzme za primjer dizajn, može nastati problem kod automatskog testiranja. Skripta provjerava nalazi li se element, npr. „logo“, na aplikaciji. Programski kod ne nalazi grešku jer se ondje uistinu nalazi logo, ali kada se pogleda aplikacija, element logo može biti prikazan načinom koji nije dizajnerski željen, tj. da nije dobar dizajn prikaza. Zbog toga je potrebno i ručno istestirati aplikaciju. Zamislite da radite za banke te u internetsko bankarstvo dolazi „Biometrija“ - prijavljivanje u aplikaciju putem biometrije. Naravno da će biti lakše i brže istestirati, za početak, sve ručno jer je lakše postaviti svoj prst na zaslon nego pisati programsku skriptu koja će otisak prsta implementirati u bazu i poslati kao da je stisnuto na zaslon i slično. Ipak, za kasnije je najbolje da postoji programski kod koji će provjeravati radi li biometrija na aplikaciji kako ne bi morali stare funkcionalnosti neprestano provjeravati na svakoj novoj inačici aplikacije. Svakako nije dobro da se na aplikaciji rade samo automatski testovi, već je potrebno da se provjerava i ručnim testiranjem jer, kako je već navedeno, automatsko testiranje pokriva najviše oko 80% grešaka u aplikaciji.

Postoji više načina, tj. slučaja automatskog testiranja.

Testiranje API-a

Testiranje API-a (eng. Application Programming Interface) je način testiranja gdje se fokusira na ulaz i izlaz podataka, primjerice u internetskom bankarstvu gdje se može napisati testni

slučaj kada je potrebno pretvoriti dolare u eure. Test će upisati 10 dolara u element i provjerit će tečaj te će ispisati koliko na kartici sada ima eura. Znači da će sustav odmah provjeriti radi li sustav za pretvaranje valute dobro. To se može provjeriti na više načina, ali najčešće se provjera s isElementVisible metodom.

Testiranje web sučelja
Testiranje web sučelja je testiranje same funkcionalnosti aplikacije koja se testira. Ovdje se dolazi do Seleniuma, web drivera i web elemenata. Selenium je alat za testiranje web aplikacija koji se piše u raznim programskim jezicima.

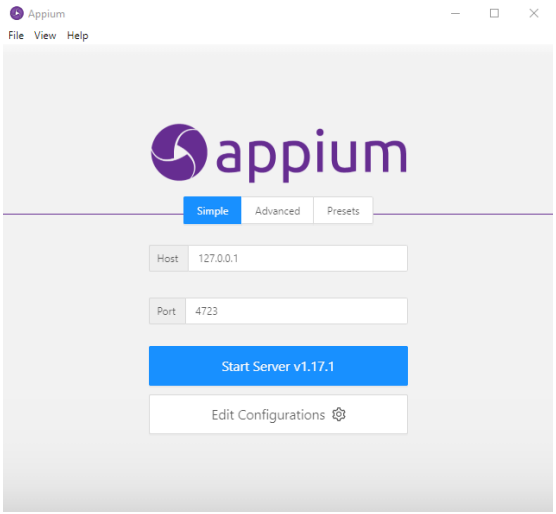
3.1. Alati za automatsko testiranje

Što se tiče alata za automatsko testiranje, postoji široki spektar alata koji se mogu koristiti, a ponekad se koriste jedan s drugim. Postoji i automatsko testiranje mobilnih aplikacija gdje se koristi i Appium za dohvaćanje elemenata. Također se koristi i Cucumber, Node.js, Maven i UIAutomator, Jenkins i brojni drugi, ali Selenium je glavni alat za testiranje. Trenutno je Selenium najpoznatiji proizvod što se tiče aplikacije otvorenog koda za automatsko testiranje web aplikacija. Tu su naravno i drugi konkurenti poput Katalona, Studia, Lambda testa, Test Complete-a. „Automatsko testiranje ima svoja ograničenja. Ono ne zamjenjuje manualno testiranje, već ga nadopunjuje. Iako ne pronalazi istu količinu pogrešaka kao manualno, može značajno povećati kvalitetu i produktivnost testiranja softvera.“ [10]

3.2. Appium

Alat kao što je Appium ne koristi se kod web aplikacija, već se koristi kod mobilnih aplikacija, najviše za dohvaćanje elemenata u aplikaciji. Appium je alat otvorenog koda koji pokreće testne slučajeve i testira aplikacije koristeći web drivere. Najveća je funkcionalnost Appiuma dohvaćanje sadržaja-opisa (eng. *content-desc*), tj. xpathova koji služe da se web driver uputi gdje se nalazi određeni web element (Slika 3.). Appium se može koristiti i s Ranorex-om i sa Seleniumom, i sa svim drugim alatima za automatsko testiranje. Postoji mogućnost pokretanja aplikacije na fizičkom uređaju ili na emulatoru ako se napravi mobilni emulator u alatu Android

Studio. Potrebno je samo Appiumu putem Desired Capabilities reći što raditi i s kojom aplikacijom to želimo (Slika 4.).



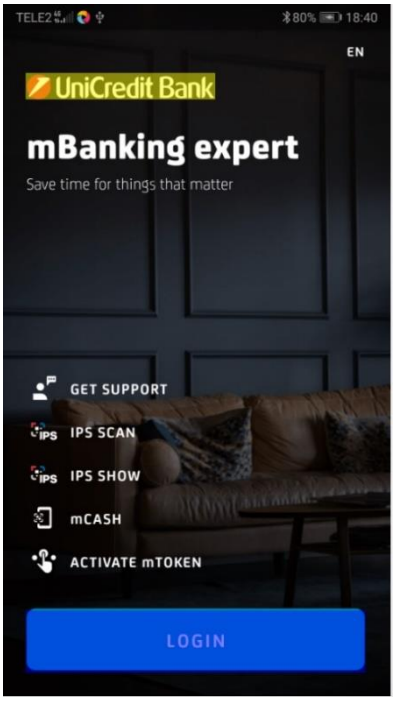
Slika 3. Appium

Potrebno je odrediti *host* i *port* na koji će se spojiti s Appiumom (Slika 3.). Nakon što se to odredi, glavne su postavke Desired Capabilities, odnosno ključevi koji se koriste kako bi pronašli JSON objekte poslane od Appiuma do poslužitelja na kojem je tražen „Hosts“.

```
{
  "app": "C:\\apps\\SK.apk",
  "noReset": true,
  "fullReset": false,
  "platformVersion": "10",
  "platformName": "Android",
  "deviceName": "emulator-5554",
  "appactivity": [redacted],
  "apppackage": [redacted],
  "appWaitPackage": [redacted],
  "appWaitActivity": [redacted],
  "automationName": "UiAutomator2"
}
```

Slika 4. Desired Capabilities

AppActivity, appPackage, appWaitPackage, appWaitActivity su podatci koje se ne smije prikazati radi zaštite podataka, no svaka aplikacija ima svoje te su unikatni za svaku aplikaciju. Njih je potrebno izvući s poslužiteljske strane. Drugi su podatci također bitni, kao što je mjesto gdje se nalazi aplikacija, na kojoj platformi aplikacija radi, kako se zove uređaj na kojem će biti pokrenuta aplikacija. Kao što je navedeno prije, aplikaciju možemo pokrenuti putem emulatora ili pomoću fizičkog uređaja, potrebno je samo u komandnoj liniji (eng. *cmd-u*) upisati naredbu **adb devices** koja će pokazati sve uređaje koji su priključeni te putem kojih se može izvoditi testiranje. Nakon što je sve ispravno upisano moguće je spojiti se na aplikaciju te početi dohvaćati elemente. Odabirom na element „gdje je žuto ili plavo“ s desne strane, „što nije prikazano“ prikazuje se kako su ti elementi prikazani u kodu te preko kojeg elementa ih smijemo dohvatiti. Appium radi s node.js-om te je njega također potrebno instalirati.



Slika 5. Appium elementi

3.3. Cucumber

Cucumber, ili u hrvatskoj inačici „Krastavac“, alat je koji se koristi kao dodatak za automatsko testiranje.

Cucumber se instalira tako da se doda u jednom od okruženja koji se koristi za programiranje: IntelliJ, Eclipse ili brojni drugi. Cucumber koristi jezik zvan „Gherkin“ koji se koristi da se definiraju testni slučajevi. Dizajniran je i osmišljen da ga mogu koristiti i osobe koje ne znaju mnogo o programiranju.

Sintaksa je sljedeća:

Potrebno je napisati ime ili naziv za testne slučajeve iznad blokova naredbi koje će se izvoditi. Ako se napiše iznad blokova naredbi koje se izvode, kod prijave u aplikaciju potrebno je iznad bloka naredbi napisati @, što je znak da počinje naziv scenarija: @Then („^ Logiraj se u aplikaciju $“) (Slika 6.). Zatim bi kasnije u *feature* datoteci mogli pozvati prijavu korisnika tako

da se upiše samo *Then Logiraj se* u aplikaciju te će program započeti prijavu u aplikaciju. Cucumber kod automatskog testiranja ima mnoge prednosti te je jedna od glavnih što ima mogućnosti da se izrade HTML ili JSON izvještaji o prolaznosti testnih slučajeva.

```
# Comment
@tag
Feature: Eating too many cucumbers may not be good for you

  Eating too much of anything may not be good for you.

  Scenario: Eating a few is no problem
    Given Alice is hungry
    When she eats 3 cucumbers
    Then she will be full
```

Slika 6. Krastavac sintaksa

3.3.1. Sintaksa jezika Gherkin

Potrebno je da svaka linija koja se koristi u Cucumber-u počinje sintaksom Gherkin-a. Najvažnije ključne riječi su:

- Given
- When
- Then
- And
- But

Postoji još i nekoliko sekundarnih poput:

- | - podatkovne tablice
- @ - oznake
- # - komentari

Primjer 1:

@Testiranje aplikacije

Feature: Promjena imena
Background:
Given Provjeri je li korisnik prijavljen
Then Otvori aplikaciju
@Promjena imena u aplikaciji
Then Klikni na promjenu imena
And Promijeni ime
Then Izađi iz aplikacije

Rečenice poput „Provjeri je li korisnik prijavljen“ ili „Otvori aplikaciju“ nalaze se u programskom kodu te su označeni blokovi koji se pozivaju kada napišemo određenu rečenicu.

Primjer 2:

@Testiranje aplikacije2

@Promjena lozinke u aplikaciji
Then Klikni na promjenu lozinke
And Promijeni lozinku
Then Izađi iz aplikacije

3.4. Node.js

„Node.js je platforma izrađena na JavaScript programsko-logičkom motoru (eng. *engine*) koja služi za izradu aplikacija za internet. Informatičke kompanije koje temelje svoje aplikacije na radu u pomoćnom alatu Node.js-u su: GoDaddy, IBM, LinkedIn, Microsoft, Netflix, Groupon, Walmart, SAP, Cisco Systems, Paypal i slične usluge.“ [7]

Ryan Dahl je 2009. godine napisao Node.js, otprilike 13 godina nakon što je predstavljen prvi *server-side* Javascript okruženje. [11] Node.js se koristi kako bi se spojile korisničke strane (eng. *front-end* i *back-end*) pomoću jednog programskog jezika. Programiranje se inače obavlja sinkrono. Linija koda se izvršava, sustav čeka rezultat te se zatim rezultat prikazuje.

No, ponekad je taj sistem izvršavanja spor te zahtijeva više vremena. U jezicima poput Jave rješenje za ovaj problem je uvođenje procesorskih jezgri. Još jedan razlog za korištenje Node-a je njegova jednostavnost, a ako se želi izgraditi nešto složenije, uvijek se može ubaciti modul treće strane pomoću Node Package Menager-a. Glavni je cilj Node-a pružiti siguran i lak način za izgradnju aplikacije visoke performanse u JavaScriptu.

3.5. Maven

Apache Maven je alat za upravljanje projektima te može upravljati različitim aspektima životnog ciklusa projekta poput gradnje, izvještavanja, dokumentacije na temelju primarnog izvora informacija. Prije toga, svaki je projekt imao vlastite datoteke za izgradnju koje je trebalo standardizirati u nešto što bi moglo graditi projekte, jasno definirati projekte, olakšati način objavljivanja projekta i način dijeljenja JAR-ova. Taj je problem doveo do stvaranja alata koji bi se mogao koristiti za izgradnju i upravljanje bilo kojim projektom koji radi na Javi. Ovaj je razvoj pojednostavio svakodnevne aktivnosti Java programera. [12]

U datoteci koja ima nastavak .xml nalaze se sve informacije o projektu s nazivom pom.xml. („Project Object Model“). U automatizaciji se Maven koristi kod Java projekata.

4. Selenium

Jason Huggins, inženjer koji je radio za ThoughtWorks, 2004. godine stvorio je Selenium dok je radio na aplikaciji koju je bilo potrebno testirati.
Možemo reći da je Selenium skup alata koji se koriste za automatsko testiranje web sučelja aplikacije. Selenium je prijenosni alat za testiranje web aplikacija te nudi alat za izvođenje i pisanje funkcionalnih testova bez potrebe za učenjem testnog skriptnog jezika (Selenium IDE). Također, nudi testni jezik specifičan za domenu (Selenese) za pisanje testova na mnogim popularnim programskim jezicima, uključujući C #, Groovy, Java, Perl, PHP, Python, Ruby i Scala. Testovi se tada mogu izvoditi na većini modernih web preglednika. Selenium se izvodi na Windows-u, Linuxu i MacOS-u. To je softver otvorenog koda koji je objavljen pod Apache License 2.0. [13]

Selenium je besplatni program otvorenog koda koji možemo koristiti bez dodatnog plaćanja. Također, radi dobro na svim operacijskim sustavima te uz pomoć web drivera izvodi testove na web preglednicima. Jedan je od najprihvaćenijih alata za automatsko testiranje jer u sebi ima puno alata koji se mogu koristiti za testiranje, kao Selenium IDE, Selenium RC, Selenium WebDriver, Selenium Grid (Slika 7.).

Slika 7. Selenium

4.1. Kako testirati u alatu Selenium

U Seleniumu možemo testirati na više načina, to su:

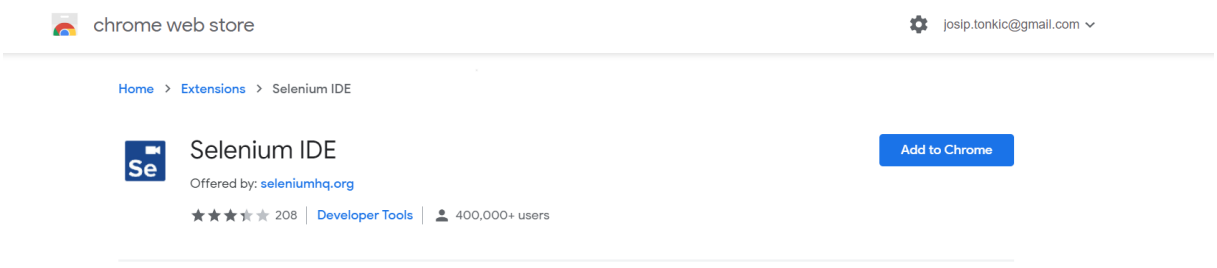
- Selenium WebDriver
- Selenium IDE
- Selenium Grid

Selenium WebDriver objašnjen je u prethodnom tekstu. Selenium WebDriver funkcionira tako da se piše programski kod koji komunicira s web preglednikom te izvodi kod koji je napisan. Ako se aplikacija želi izvoditi na više okruženja ili operacijskih sustava, ovo je najbolji način za to.

```
//SET LANGUAGE
    @iOSXCUITFindBy(accessibility = "Language select")
    @AndroidFindBy(xpath = "//android.widget.TextView[@text='Language select']")
    public WebElement languageTitle;
    public PreLoginPage verifyLanguagePage(){
        languageTitle.isDisplayed();
        return this;
    }
```

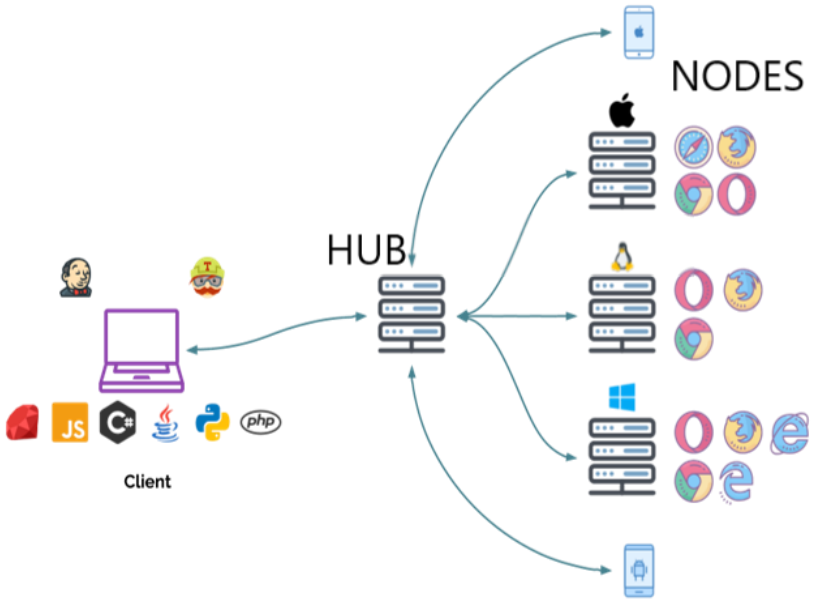
Slika 8. Selenium sintaksa

Selenium IDE je jedan od najjednostavnijih alata koje Selenium ima. Selenium IDE se implementira tako da se preuzme Chrome ili Firefox Add-on koji omogućava snimanje zaslona te editiranje i naših testova. Znači, jednostavno se snimi naš zaslon gdje klikamo, a Selenium IDE će sam napisati programski kod za nas. Tvorac je ovog alata Shinya Kasatani koji je ovaj alat donirao Selenium projektu 2006. godine.



Selenium Grid dio je Selenium paketa koji je specijaliziran za višestruko testiranje u različitim preglednicima, operativnim sustavima i uređajima. To se postiže usmjeravanjem naredbi udaljenih preglednika u kojima poslužitelj djeluje kao čvorište. Korisnik mora konfigurirati udaljeni poslužitelj kako bi izvršio testove. [14]

Ako se testovi žele pokretati na više uređaja u isto vrijeme u više različitih okruženja, tada se koristi Selenium Grid. On također ima opciju da se implementira kod iz Selenium IDE-a tako da se mogu snimiti radnje te ih zatim implementirati. (Slika 9.)



Slika 9. Selenium Grid

4.2. Webdriveri koji se koriste u Seleniumu

Testiranje u Seleniumu podržavaju svi poznati preglednici kao što su Google Chrome, Firefox, Opera preglednik te se također može pisati u brojnim jezicima kao što je Ruby, C#, Phyton, Java itd. Arhitektura webdrivera, tj. način na koji Selenium radi je sljedeći:

U određenom editoru piše se programski kod, tj. testni slučajevi koji se trebaju izvoditi. Nakon što se napiše programski kod i pokrene se test, programski kod pretvara se u JSON format i generirani JSON šalje se na poslužitelja koji je u ovom slučaju driver preglednika.

Ako se koristi Chrome, driver preglednika je „Chrome driver“ ili ako se koristi Firefox, onda je „Gecko driver“. Kod se od klijenta do poslužitelja prenosi u JSON formatu putem HTTP protokola. Nakon što kod komunicira s driverom web preglednika, dalje komunicira s određenim preglednikom. Ako koristimo Chrome driver, on zna interpretirati JSON format web pregledniku Chrome kao Selenium kod te zna izvršavati željene radnje na njemu. Kada web preglednik obradi željeni kod, on putem HTTP poslužitelja vraća webdriveru odgovor gdje ga webdriver pretvara natrag u JSON format i šalje ga natrag klijentu. Ovo je samo koncept na koji način Selenium funkcionira u pozadini te to nije potrebno znati kako bi se napisao programski kod u Seleniumu, ali je bitno znati da programski kod u Seleniumu ne komunicira direktno s web preglednikom. (Slika 10.)

Slika 10. Selenium pozadinski procesi

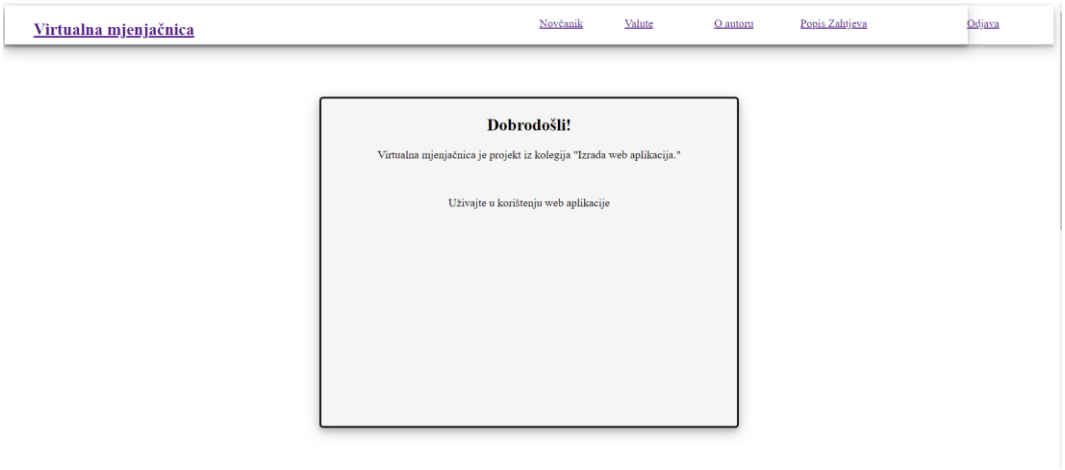
Webdriveri preglednika:

- Mozilla Firefox Driver
- Internet Explorer WebDriver
- Opera Browser WebDriver
- Ghost WebDriver
- Google Chrome WebDriver
- Safari WebDriver

Nakon odluke s kojim se preglednikom želi raditi potrebno je s interneta preuzeti odgovarajući webdriver te ga implementirati u programski kod.

5. Opis aplikacije Virtualna mjenjačnica

Aplikacija Virtualna mjenjačnica nalazi se na lokalnom web poslužitelju koji se pokreće preko modula Apache. Web aplikacija je pisana u HTML-u, CSS-u i PHP-u te se koristio SQL za komuniciranje s bazom podataka. Aplikacija ima brojne funkcionalnosti kao što su prodavanje i kupovanje valuta, editiranje valuta, pregled informacija o valutama itd. Za web aplikaciju postoji automatski test koji će provjeriti logiranje korisnika, moderatora i administratora, dodavanje valute i ostale funkcionalnosti.

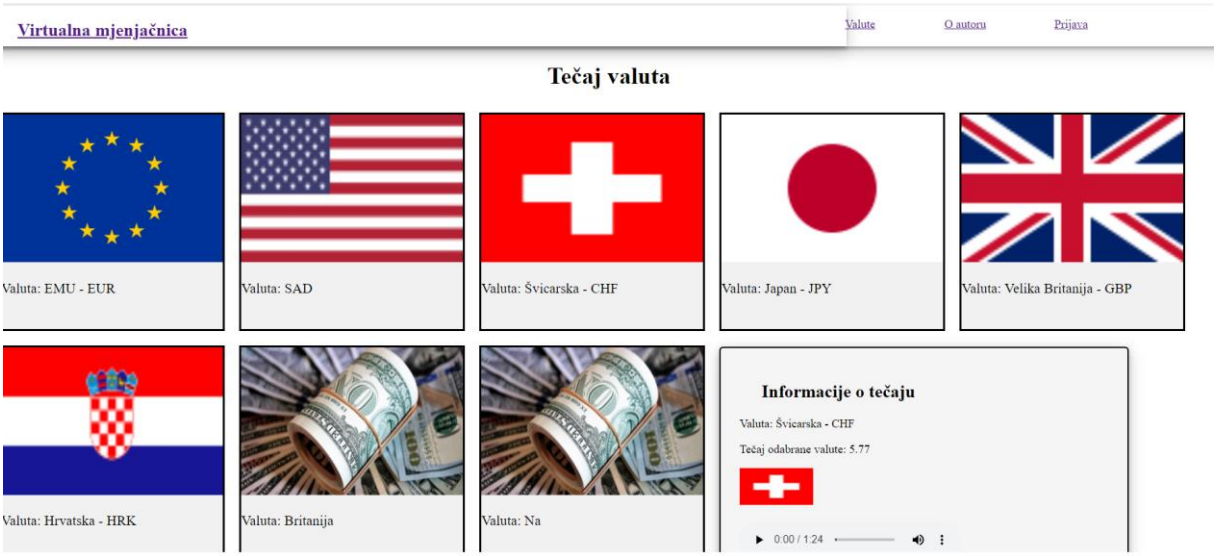


Slika 11. Početna stranica web aplikacije

5.1. Korisnici i uloge korisnika

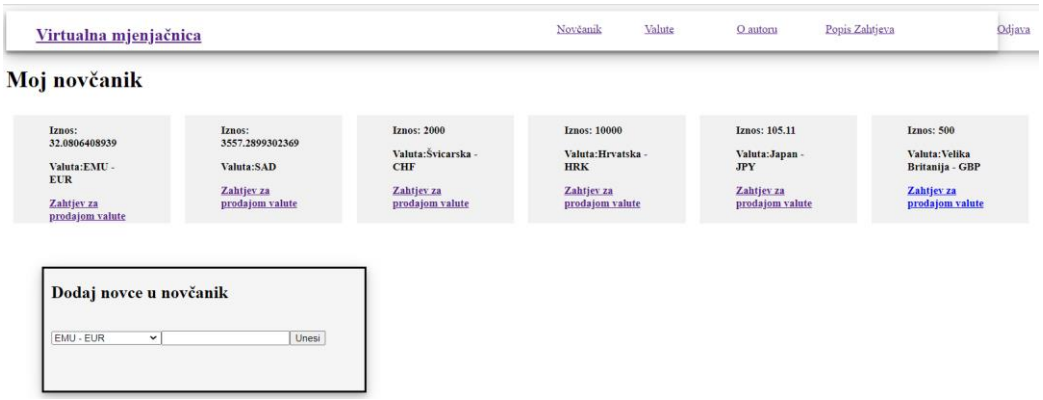
Postoje četiri razine uloga korisnika u web aplikaciji, a to su: neregistrirani korisnik, registrirani korisnik, moderator i administrator. Svaka od pojedinih uloga ima svoju zadaću u aplikaciji koju mora izvršavati. Korisnik ima najmanje ovlasti, administrator ima najveće ovlasti, a moderatora možemo zamišljati kao voditelja Virtualne mjenjačnice.

Neregistrirani korisnik ima mogućnost pregleda svih valuta iz sustava koji je predstavljen kao galerija slika gdje klikom na zastavu valute dobiva osnovne informacije o valuti, tečaj valute, himnu valute (ako je ona postavljena).



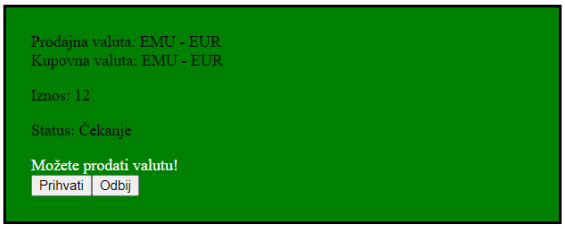
Slika 12. Neregistirani korisnik sučelje

Registrirani korisnik ima sve mogućnosti kao i neregistrirani korisnik te može u svoj novčanik dodati valute po izboru kako bi ih mogao dalje prodavati. Također, korisnik može prodati, tj. poslati zahtjev za prodajom valute koju posjeduje, a ima i mogućnost pogledati status svih svojih poslanih zahtjeva, koji mogu biti na čekanju, prihvaćeni ili odbijeni.



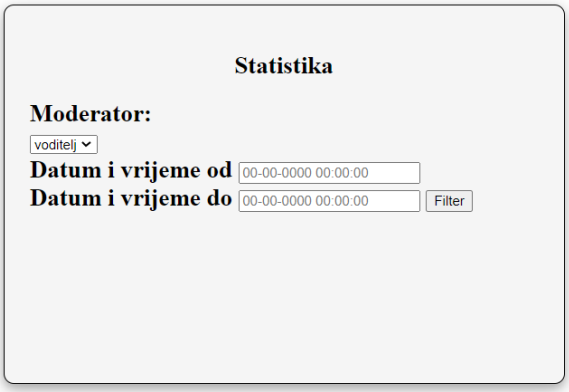
Slika 13. Novčanik registriranog korisnika

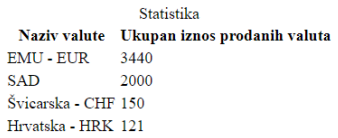
Moderator/voditelj također ima sve mogućnosti kao i prethodne dvije razine korisnika, a dodatno može jednom dnevno ažurirati vrijeme kada se pojedina valuta prodaje. Svaki moderator zadužen je za određenu valutu u odnosu da moderator može imati, tj. biti zadužen za više valuta, a jedna valuta može imati jednog moderatora. Moderator je zadužen i za prihvaćanje ili odbijanje zahtjeva koji su poslani za prodaju valute te može prihvatiti ili odbiti valutu ako je u granicama vremena prodaje te određene valute. (Slika 14.)



Slika 14. Zahtjev za prodaju valute

Administrator, uz sve mogućnosti prethodnih uloga, može unositi, ažurirati i pregledavati korisnike sustava te postavljati uloge korisnicima da postanu administratori, moderatori ili korisnici. Također ima mogućnost uvođenja novih valuta, može ažurirati valute i njihove tečajeve te postavljati moderatore koji su zaduženi za valutu. Administrator može vidjeti ukupni iznos prodanih valuta pod dijelom web aplikacije Statistika te ima mogućnost pregleda svih poslanih zahtjeva prodaje valuta.





Slika 15. Statistika kao funkcionalnost administratora

5.3. Baza podataka aplikacije

Za bazu podataka web aplikacije korišten je PhpMyAdmin, što je alat za komuniciranje s MySQL bazom podataka. Bazu podataka napisao je profesor na FOI-u za kolegij „Izgradnja web aplikacija“. Na PhpMyAdmin pristupamo tako da se treba u XAMPP-u uključiti MySQL modul kako bi se omogućio pristup lokalnom poslužitelju na kojem je baza podataka na poveznici http://localhost/phpmyadmin/.

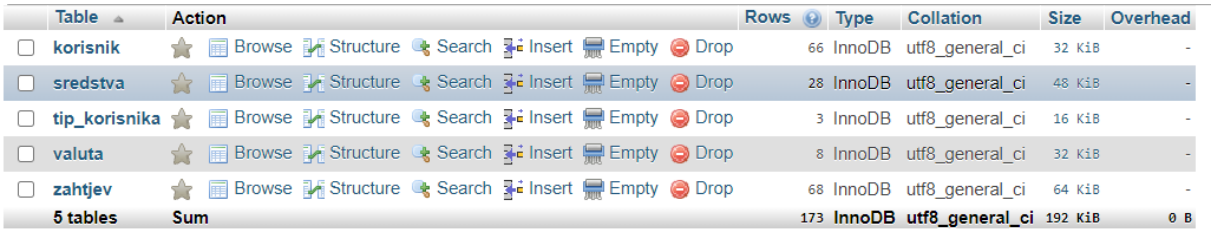


	Table	Action	Rows	Type	Collation	Size	Overhead
☐	korisnik	Browse Structure Search Insert Empty Drop	66	InnoDB	utf8_general_ci	32 KiB	-
☐	sredstva	Browse Structure Search Insert Empty Drop	28	InnoDB	utf8_general_ci	48 KiB	-
☐	tip_korisnika	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8_general_ci	16 KiB	-
☐	valuta	Browse Structure Search Insert Empty Drop	8	InnoDB	utf8_general_ci	32 KiB	-
☐	zahtjev	Browse Structure Search Insert Empty Drop	68	InnoDB	utf8_general_ci	64 KiB	-
	5 tables	Sum	173	InnoDB	utf8_general_ci	192 KiB	0 B

Slika 16. Baza podataka web aplikacije

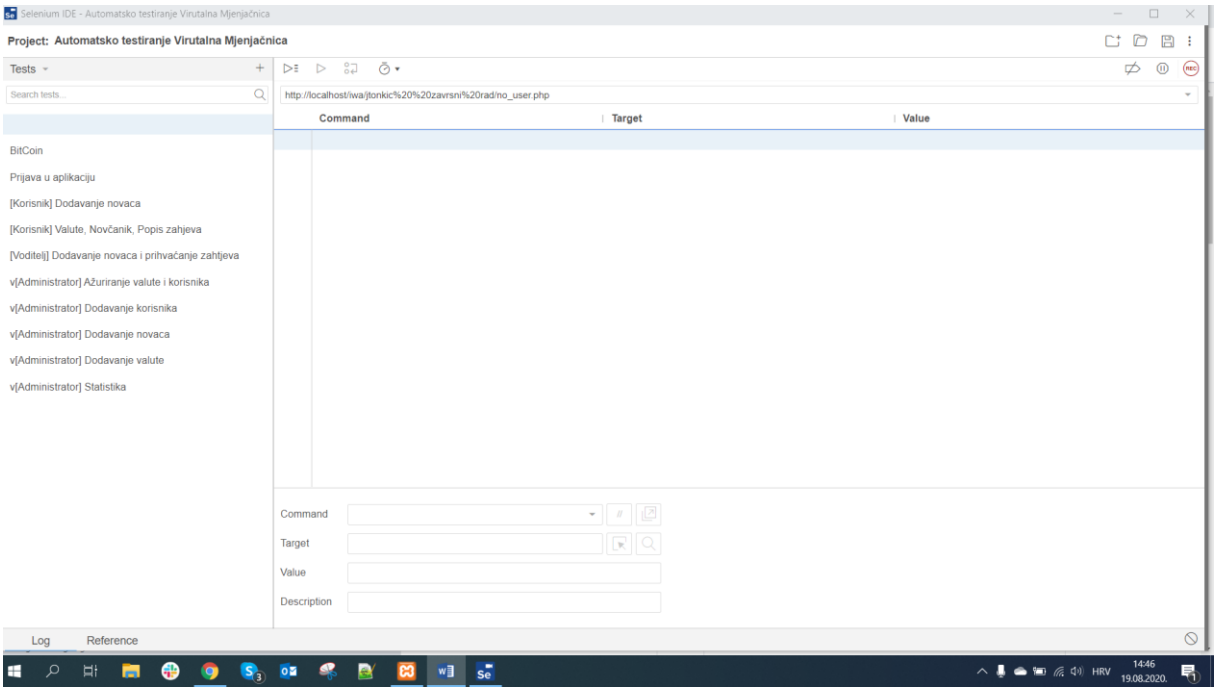
Baza podataka sastoji se od pet tablica, to su:

- Korisnik
- Sredstva
- Tip_korisnika
- Valuta
- Zahtjev

U tablici korisnik administrator ima mogućnost unositi korisnike u bazu podataka. Sve te tablice povezane su jedna s drugom pomoću primarnih i vanjskih ključeva. Za primjer, u tablici Korisnik nalazi se i Tip_korisnika, a to je tablica koja govori koju ulogu korisnik ima u sustavu: korisnik, administrator, voditelj. Također, postoji i tablica Valuta u kojoj se nalaze sve valute iz našeg sustava, tablica zahtjeva sa svih zahtjevima koji su u sustavu te Sredstva kao tablica koja govori koliko svaki pojedini korisnik ima novca, tj. koliko posjeduje sredstava pojedinih valuta. (Slika 16.)

6. Automatsko testiranje Virtualne mjenjačnice

Automatsko testiranje web aplikacije Virtualna mjenjačnica napravljeno je pomoću Selenium IDE alata (Slika 17.).



Slika 17. Selenium IDE

Selenium IDE, kao što je objašnjeno ranije u tekstu, funkcionira tako da zaslon snima što korisnik radi te na temelju toga piše testne slučajeve. Virtualna mjenjačnica sastoji se od 10 testnih slučajeva:

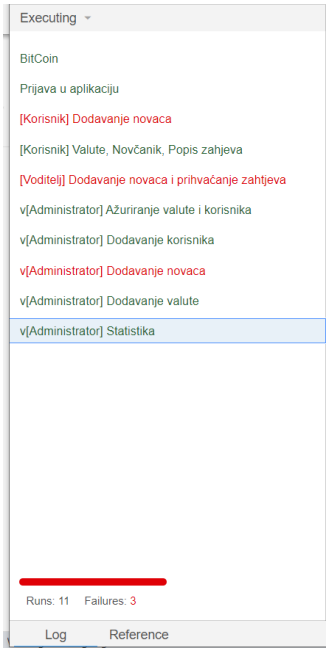
- Bitcoin
- Prijava u aplikaciju
- [Korisnik] Dodavanje novca
- [Korisnik] Valute, novčanik, popis zahtjeva
- [Voditelj] Dodavanje novca i prihvaćanje zahtjeva
- [Administrator] Ažuriranje valute i korisnika
- [Administrator] Dodavanje korisnika
- [Administrator] Dodavanje novca
- [Administrator] Dodavanje valute
- [Administrator] Statistika

Svaki testni slučaj ima svoj naziv te se u njemu nalaze koraci koje Selenium izvodi prilikom testiranja aplikacije. Primjerice, u testnom slučaju Bitcoin prvi je korak otvoriti zadanu poveznicu na kojoj se nalazi aplikacija, zatim povećati prozor aplikacije, kliknuti na tekst gdje piše Prijava, kliknuti na element koji se naziva *korime*, unijeti *pkos* u taj element, kliknuti na element lozinka, zatim unijeti lozinku te kliknuti na prijavu (Slika 18.).

1	*open*	http://localhost/iwa/jtonkic%20%20zavrsni%20rad/no_user.php	
2	*set window size*	1936x1056	
3	*click*	linkText=Prijava	
4	*click*	name=korime	
5	*type*	name=korime	pkos
6	*type*	name=lozinka	123456
7	*send keys*	name=lozinka	${KEY_ENTER}

Slika 18. Testni koraci

Na temelju ovih testnih koraka Selenium zna na koje elemente treba kliknuti te može sam automatski testirati aplikaciju. Na ovaj način, s napisanim testnim koracima, ne treba se ručno testirati npr. radi li „Prijava“ u aplikaciji, već postoje napisani testni koraci za to. Ako test ne bude uspješan, dobije se izvještaj da test nije prošao te se može vidjeti gdje se nalazi greška. (Slika 19.)

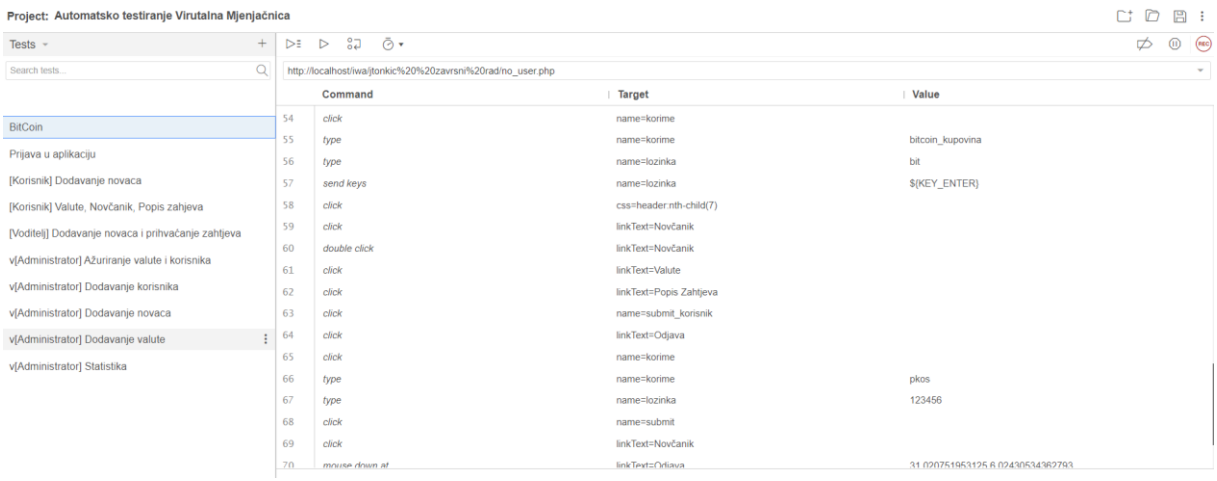


Slika 19. Izvještaj Selenium IDE

6.1. Testni scenarij Bitcoin

U ovome se testnom slučaju prvo testiraju koraci gdje se kupuje valuta bitcoin-a, a prodaje hrvatska kuna. Testira se radi li SQL upit te postavlja li kriptovalutu bitcoin u novčanik prijavljenog korisnika. Sada kada postoji kriptovaluta bitcoin, mora se testirati može li se dodati kriptovaluta bitcoin u novčanik te se dodaje „100“ bitcoin-a u novčanik.

Zatim je također potrebno testirati prodaju i kupovinu kriptovaluta. Prvi je dio testnog slučaja testiranje funkcionalnosti prodaje valute euro za kriptovalutu. Kada se kupuje kriptovaluta, sustav provjerava novi tečaj kriptovalute te prodaje 100 eura u zamjenu za kriptovalutu bitcoin. Zatim se isprobava i prodaje kriptovaluta bitcoin. Za prihvaćanje zahtjeva prodaje kriptovalute potrebna je prijava s posebnim moderatorom koji je zadužen samo za kriptovalutu. (Slika 20.) Test se prijavljuje kao korisnik voditelj da provjeri jesu li promijenjeni iznosi za bitcoin i euro. U ovom testnom slučaju testira se najviše funkcionalnosti rada sa zahtjevima, prodajom, kupovinom kriptovaluta te prihvaćanje i odbijanje zahtjeva.



Slika 20. Testni koraci za scenarij Bitcoin

6.2. Testni scenarij Prijava u aplikaciju

Testni scenarij Prijava u aplikaciju izvršava se kao što je objašnjeno u naslovu - prijavom u aplikaciju. Suština je ovog testnog slučaja provjera radi li sustav za prijavu u aplikaciju. Testni scenarij sastoji se od niza koraka u kojima se testira prijavljivanje svih korisnika u sustav. Prvo što testni scenarij odrađuje prijava je u sustav s krivim korisničkim imenom i lozinkom. Kada su upisani krivi podatci, tada se dobije poruka da su podatci netočni i ako se nije otvorio novi prozor u aplikaciji, može se zaključiti da baza podataka i web aplikacija dobro komuniciraju. Sljedeći je korak ovog testnog scenarija prijavljivanje i odjavljivanje s ispravnim podatcima iz baze podataka za sve tipove korisnika, a to su administrator, voditelj i korisnik.
Na ovaj način provjerava se komunikacija s bazom podataka i aplikacijom te radi li u web aplikaciji sustav za prijavljivanje korisnika.

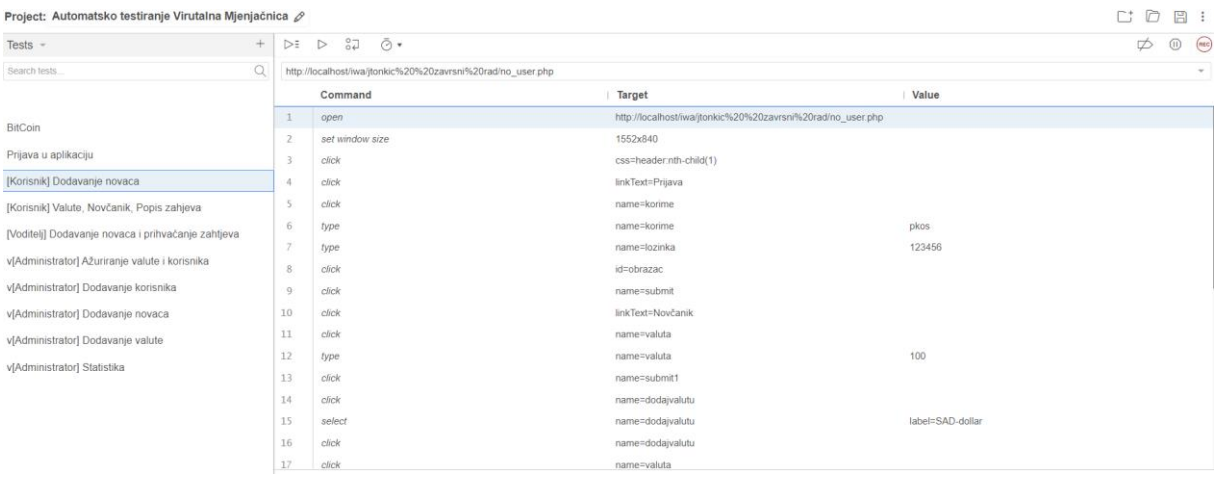
U testnim koracima 20 i 21 naredba koju automatski test izvodi je unos (eng. *type*), a oznaka gdje treba unositi ima naziv *korime* i lozinku. U koraku 20 unosi se *pkos* kao korisničko ime, a u koraku 21 unosi se *123* kao lozinka korisnika. U testnom koraku 22. naredba koja se izvršava zove se *send keys* te je vrijednost koja se šalje tipkom *enter*. (Slika 21.)

	Command	Target	Value
1	*open*	/iwa/jtonkic%20%20zavrsni%20rad/no_user.php	
2	*set window size*	1552x840	
3	*click*	linkText=Prijava	
4	*click*	linkText=O autoru	
5	*click*	linkText=Početna stranica	
6	*mouse down at*	linkText=Valute	29,12
7	*mouse move at*	linkText=Valute	29,12
8	*mouse up at*	linkText=Valute	29,12
9	*click*	linkText=Valute	
10	*click*	linkText=Valute	
11	*click*	linkText=Virtualna mjenjačnica	
12	*click*	linkText=Prijava	
13	*click*	name=submit	
14	*click*	name=korime	
15	*type*	name=korime	admin
16	*click*	name=lozinka	
17	*type*	name=lozinka	1

Slika 21. Testni koraci za scenarij Prijava u aplikaciju

6.3. Testni scenarij Dodavanje novca

Testni scenarij Dodavanje novca testira rade li sve funkcionalnosti dodavanja valuta u novčanik prijavljenog korisnika. Testni scenarij testira prvo mogućnost prijavljivanja u sustav. (Slika 22.) Kada je taj dio uspješan, dolazi se do testiranja funkcionalnosti dodavanje novca.
Potrebno je istestirati slučajeve kada već postoji valuta u novčaniku te dodaje li valute koje trenutno korisnik ne posjeduje u novčaniku. Prvi dio testnog scenarija prolazi kroz sve valute koje korisnik ima, testira valute koje korisnik ne posjeduje te doda sve valute još jednom. Na ovaj se način pokrivaju svi scenariji vezani za dodavanje novca u novčanik korisnika. Kada se ovaj dio testa odradi za korisnika, tada će testni scenarij ponoviti to sve i za moderatora i za administratora web aplikacije.



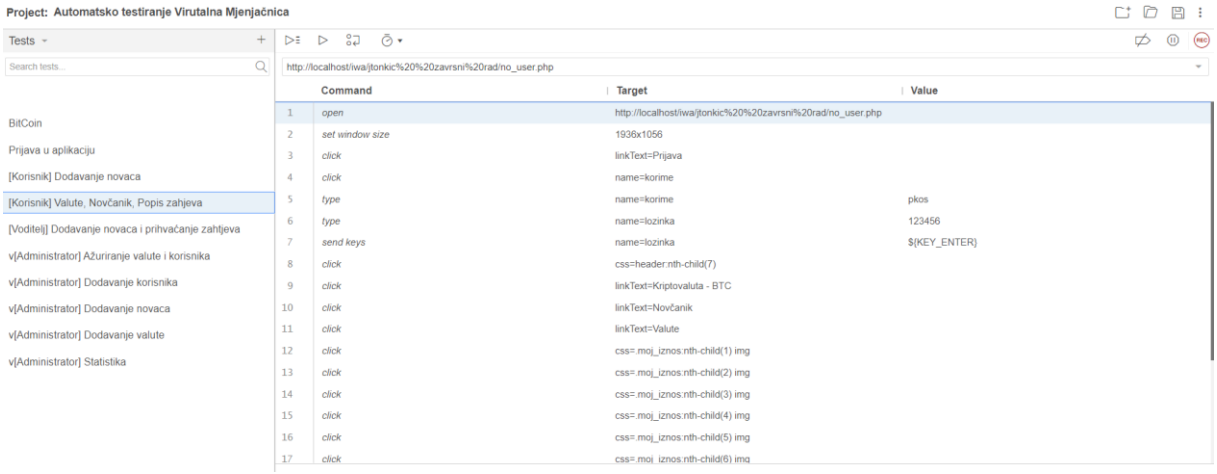
Slika 22. Testni koraci za scenarij dodavanje novaca

6.4. Testni scenarij Valute, novčanik i popis zahtjeva

Web aplikacija u sebi ima različite funkcionalnosti za različite tipove korisnika. Običan korisnik ima ovlasti samo da pogleda sve valute u galeriji valuta te se dobiju osnovne informacije o njima kao što su tečaj, slika, naziv valute i himna, ako je ona postavljena. Moderator ima, uz sve mogućnosti kao i korisnik, mogućnost ažuriranja valute jednom u danu, stoga ovaj testni slučaj mora provjeriti sve testne scenarije. Svaki moderator može ažurirati tečaj valute za koju je on zadužen te može prihvatiti ili odbiti zahtjev za svoju valutu tako da se

testira i taj dio web aplikacije. Prvi dio koji testni scenarij odrađuje testiranje je galerije kao korisnik. Testira se može li se kliknuti i dobiti informacije za svaku valutu koja se nalazi u galeriji, zatim se ponavlja proces, ali s moderatorom gdje će se pokušati kliknuti i na valute za koje moderator nije zadužen kako bi se izvršila provjera mogu li se valute ažurirati. (Slika 23.)

Nakon toga potrebno je provjeriti može li moderator dva puta ažurirati određenu valutu jer je dozvoljeno ažuriranje jednom dnevno. Ako se na ekranu pojavi poruka „*Nažalost ova valuta je danas ažurirana*“, onda testni scenarij prolazi, a ako ne, slučaj će izbaciti grešku. Nakon toga testira se mogućnost dodavanja novca u novčanik za svaku valutu koju korisnik posjeduje te prodaju valuta, tj. slanje zahtjeva zaduženim moderatorima. Potrebno je prodati svaku valutu te provjeriti status valute. Kada se pošalje zahtjev moderatoru, potrebno je provjeriti je li status na čekanju.



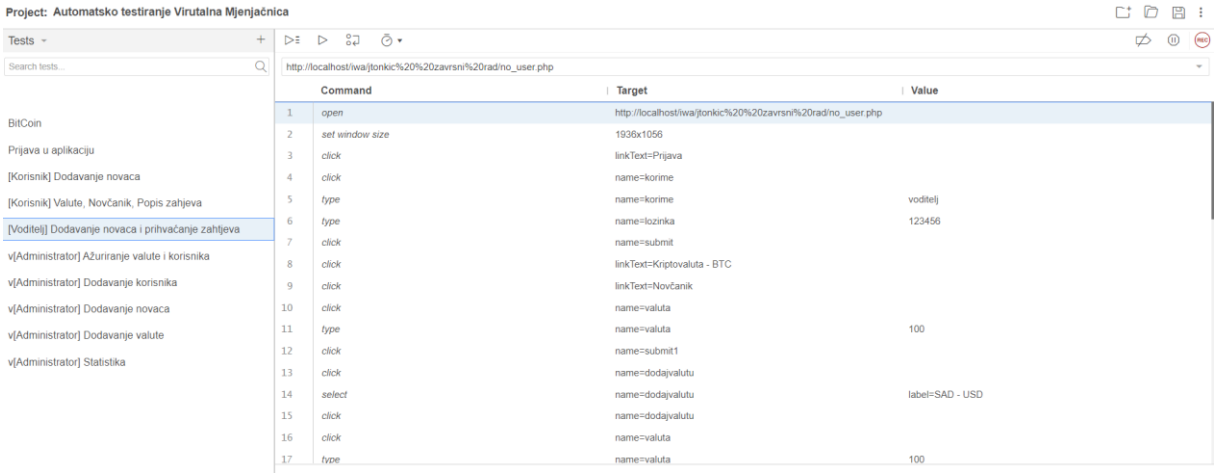
Slika 23. Testni koraci za scenarij valute, novčanik i popis zahtjeva

6.5. Testni scenarij Dodavanje novca i prihvaćanje zahtjeva

Kod ovoga se scenarija testira s korisnikom moderatorom. Potrebno je također testirati mogućnost dodavanja novca za valutu koju posjeduje prijavljeni korisnik i valute koje ne posjeduje, da se provjeri radi li funkcija za dodavanje sredstava u novčanik za tipove korisnika moderatora ispravno.
Nakon toga testira se može li se dodati novac za svaku valutu koju korisnik posjeduje te prodaju valuta, tj. slanje zahtjeva zaduženim moderatorima. Prvo se prodaje valuta USD za koju je zadužen moderator *kdunst*, ali je korisnik prijavljen s moderatorom voditeljem. (Slika 25.)

Ako moderator voditelj vidi zahtjev za prodaju valute USD, testni je scenarij neuspješan i ispisuje se greška, a ako ne, test se nastavlja. Zatim se prodaje valuta EUR te se s prijavljenim korisnikom koji je zadužen za tu valutu provjerava može li je vidjeti i prihvatiti. Ako može, test će biti uspješan. Ako se dogodi da test ne bude uspješan, dobije se odgovarajuća greška. (Slika 24.)

Slika 24. Pad testnog slučaja

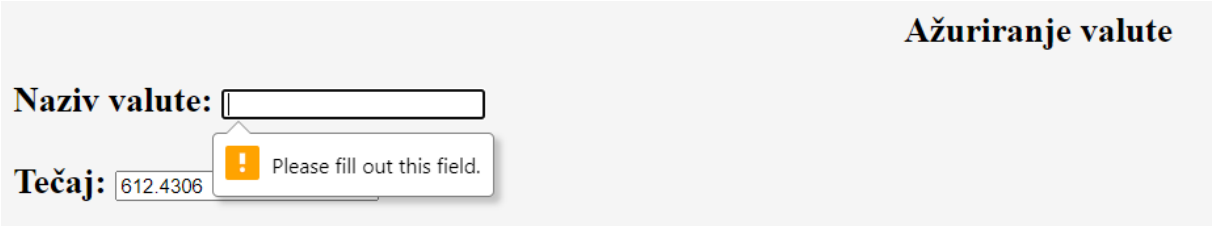


Slika 25. Testni koraci za scenarij Dodavanje novca i prihvaćanje zahtjeva

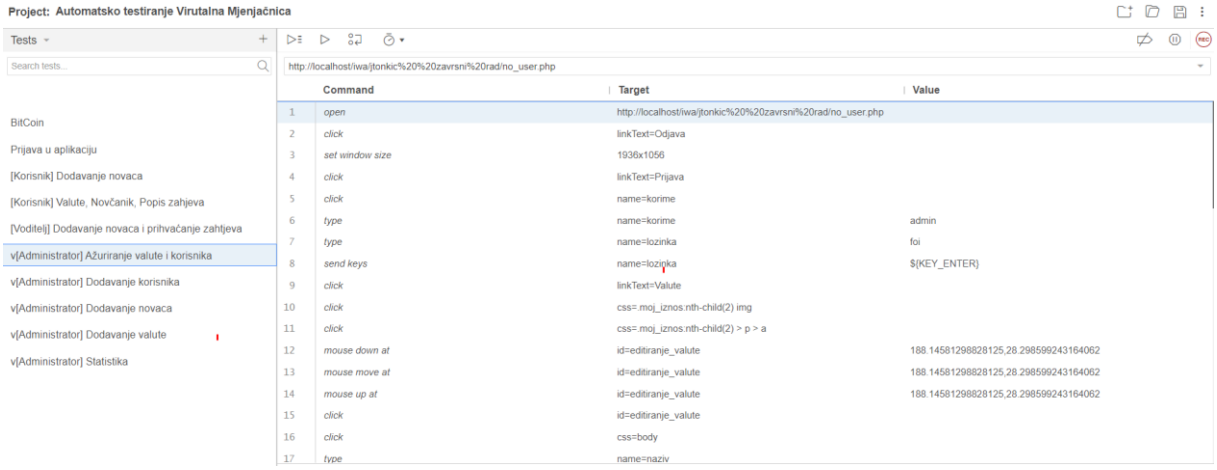
6.6. Testni scenarij Ažuriranje valute i korisnika

Kod ovoga testnog scenarija provjerava se ažuriranje valute i korisnika administratora. Administrator ima mogućnost ažuriranja svih valuta koje se nalaze u galeriji, ali i dalje je moguće ažurirati valutu jednom dnevno. U prošlim testnim scenarijima već se testiralo ažuriranje valute te valute koja je danas ažurirana tako da se prvo ažuriraju te valute, što je ujedno i prvi korak. Prvi je korak ažuriranje valuta koje je danas u prošlim testovima već ažurirao moderator. Zatim se testira mijenjaju li se podatci kada ih administrator ažurira. Kada se klikne na poveznicu Ažuriraj, potrebno je dobiti novu stranicu koja ima naslov „*Ažuriranje valute*“. Postoji i element naziva: *naziv valute, tečaj moderator zadužen za valutu, slika URL, URL zvuka, vrijeme prodaje od i vrijeme prodaje do.*

Drugi je korak izmjena podataka odabrane valute te promjena naziva valute. (Slika 27.) Test provjerava je li promijenjen naziv valute te ako jest, ponovno se ažurira ista valuta. Zatim se provjerava slanje obrasca, ali s praznim podatcima jer polja valuta trebaju biti obvezna za unos. Ako sustav ne odlazi dalje na drugi dio skripte, test će biti uspješan. Zatim se dolazi do druge funkcionalnosti, a to je ažuriranje korisnika sustava. Ažuriranje korisnika sustava radi istim principom kao i ažuriranje valute. Prvo je potrebno promijeniti vrijednosti korisnika, zatim provjeriti je li došlo do promjene u tablici. Drugi je dio testiranje ako se ne ispune obvezna polja unosa. (Slika 26.)



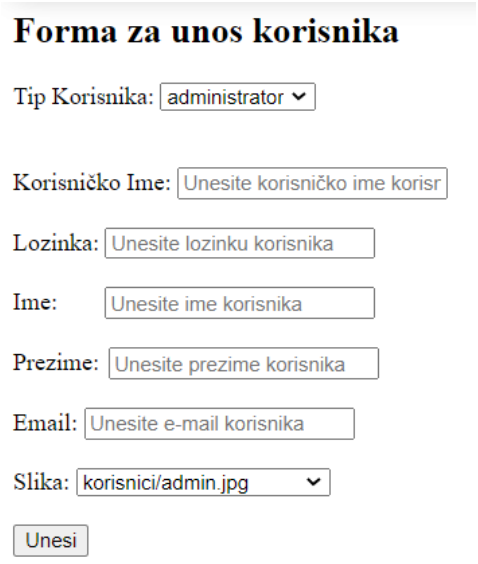
Slika 26. Prazno obvezno polje



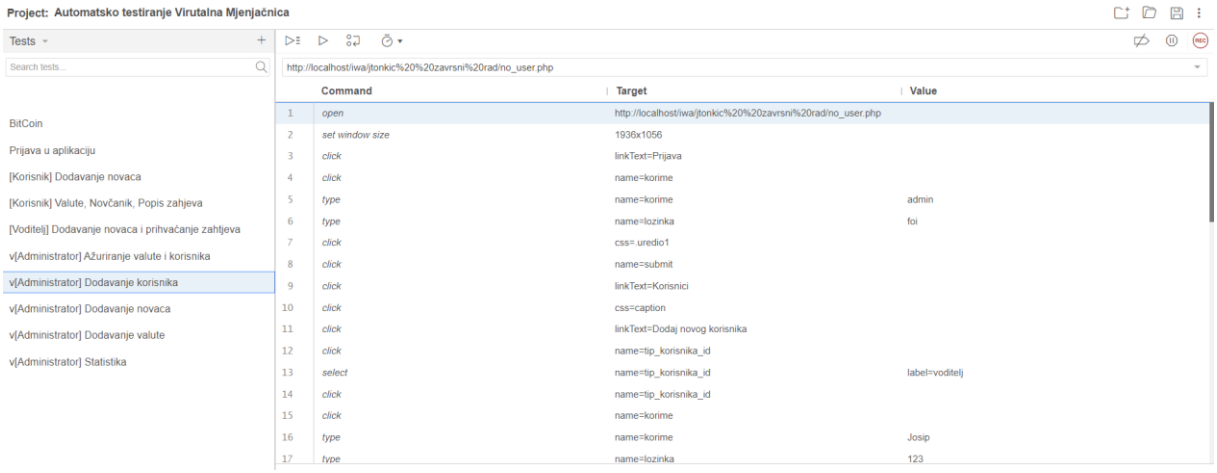
Slika 27. Testni koraci za scenarij Ažuriranje valute i korisnika

6.7. Testni scenarij Dodavanje korisnika

Administrator kao tip korisnika ima mogućnost dodavanja novih korisnika u sustav. U scenariju se testira dodavanje korisnika, njihovo prijavljivanje te brisanje novonastalih korisnika. Prvi je korak testiranje kreiranja korisnika. (Slika 28.) Kako i kod valuta korisnici trebaju imati obvezna polja unosa, u prvom dijelu testiranja ne upisuje se ime korisnika, koje je obvezno polje, te se šalje obrazac. Ako web aplikacija ne doda korisnika, test nastavlja dalje tako da unese sve obvezne podatke korisnika. Zatim testira je li taj korisnik upisan u bazu podataka. Potrebno je napraviti odjavu s aplikacije te probati upisati korisničko ime i lozinku testnog korisnika da se testira radi li dodavanje korisnika u sustav. (Slika 29.) Ako radi, test nastavlja dalje te se odjavljuje iz web aplikacije i prijavljuje se dalje kao administrator sustava. Potom se testira radi li mogućnost brisanja postojećih korisnika u sustavu. Sljedeći je korak brisanje novonastalog korisnika, odjava iz aplikacije i pokušaj prijavljivanja sa starim, obrisanim korisnikom. Ako se ne uspije prijaviti u aplikaciju, test će biti uspješan.



Slika 28. Dodavanje korisnika



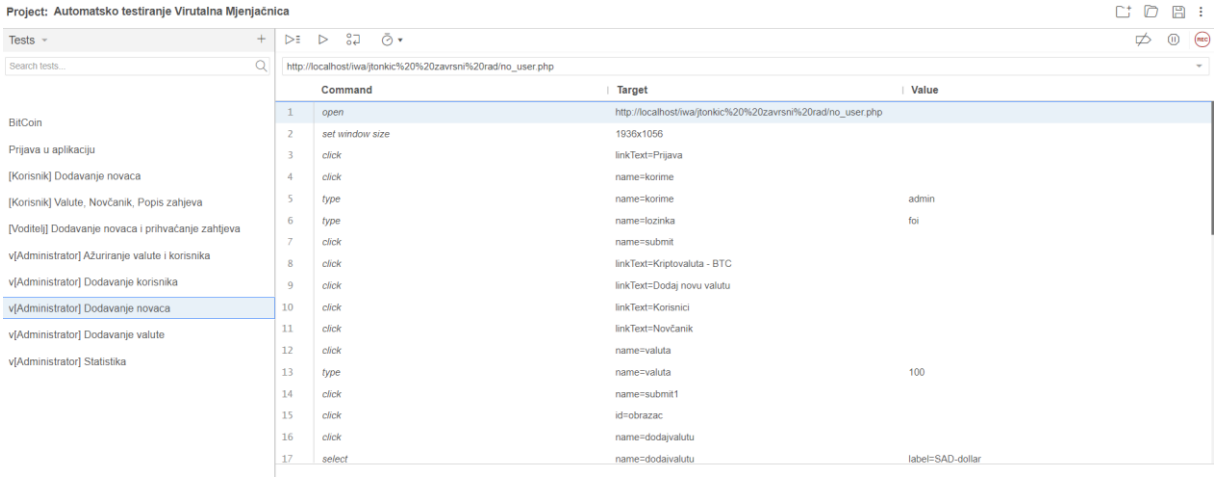
Slika 29. Testni koraci za scenarij Dodavanje korisnika

6.8. Testni scenarij Dodavanje novca i dodavanje valute

U ovom testnom slučaju provjerava se može li korisnik administrator dodati novac u novčanik te stvoriti novu valutu. Prvi je korak dodavanje svih valuta koje korisnik ima ili nema, provjerava se jesu li vrijednosti i valute dodane te ima li administrator mogućnost dodavanja nove valute. Kako se prije već spominjalo, određena polja imaju obvezan unos podataka tako da se još jednom testira taj dio. (Slika 30.)

Prvi je dio stvaranje nove valute gdje se unosi sve osim imena te se pokuša poslati obrazac. Ako se obrazac ne pošalje, tada test uspješno nastavlja dalje. Zatim je potrebno (jer je sada unesen novi tečaj) testirati može li se taj tečaj ažurirati jer je danas već jednom ažuriran. Ako je i taj korak uspješan, tada se vraća nazad na novčanik te se dodaje novac za novu valutu i šalje se zahtjev prodaje nove valute da se testiraju zahtjevi za prodaju nove valute. (Slika 31.) Zatim se aplikacija odjavi te prijavi kao moderator koji je zadužen za novu valutu. Ako moderator vidi valutu, test se završava te je uspješan. Ako ne vidi, sustav će izbaciti grešku.

Slika 30. Dodavanje nove valute

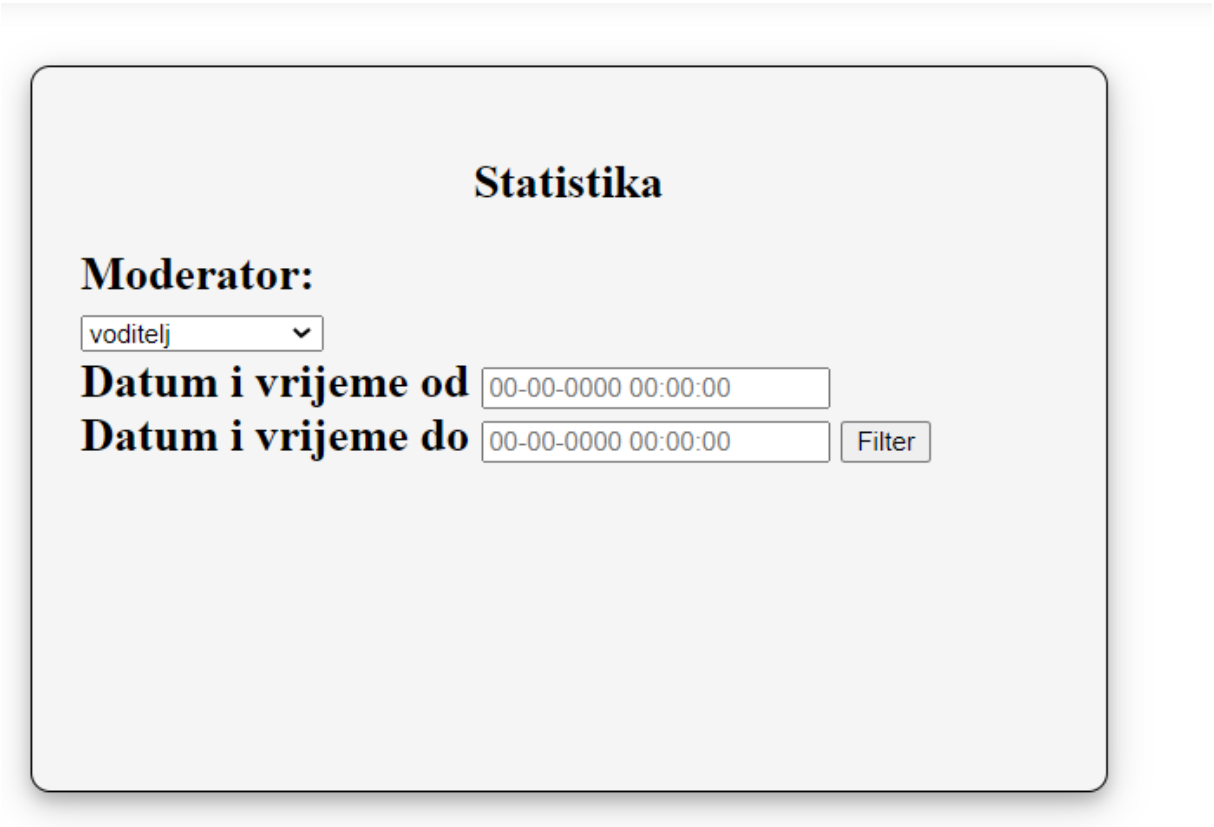


Slika 31. Testni koraci za scenarij Dodavanje novca

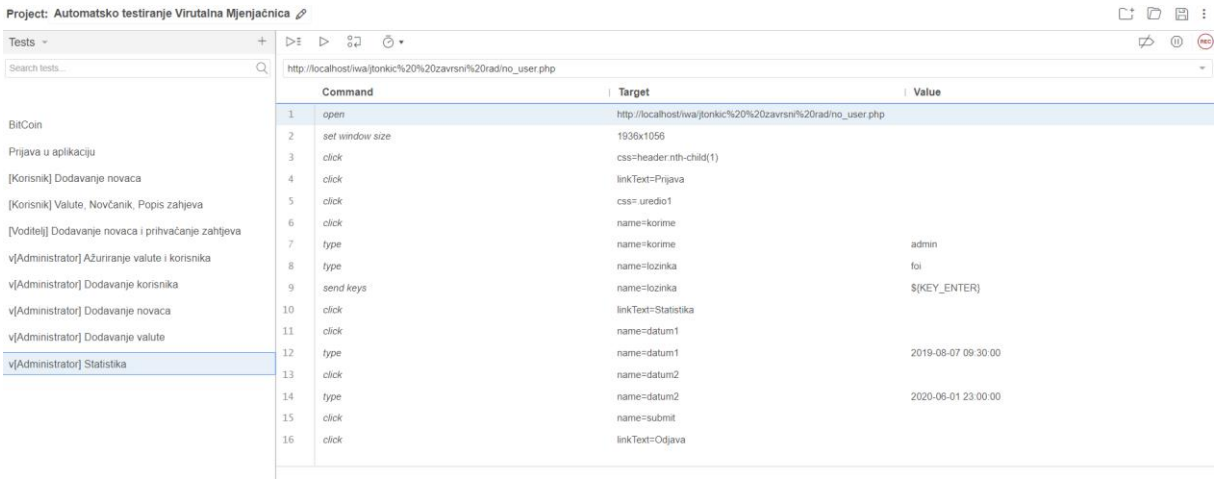
6.9. Testni scenarij Statistika

Zadnji testni scenarij koji je ostao je Statistika, što je ujedno i funkcionalnost administratora. Prvi je korak prijava u aplikaciju kao administrator te otvaranje poveznice Statistika. (Slika 33.) Test će prvo pokušati poslati zahtjev bez vrijednosti u poljima da se testira obveza unosa vrijednosti. (Slika 32.)

Ako sustav ne dobije ništa, tada će test uspješno nastaviti dalje, u suprotnom će ispisati grešku u liniji. Zatim je potrebno u polja *Datum i vrijeme od i do* upisati ispravne vrijednosti, a to je vrijeme kada je počeo prvi test koji se pokrenuo i vrijeme kada je počeo ovaj testni scenarij. Tada će se kao rezultat dobiti vrijednost svih zahtjeva koji su testirani u ovim testnim scenarijima. Ako se dobiju ispravni podatci, tada sustav ispisuje uspješnost testiranja testnih slučajeva, u suprotnom će test biti neuspješan.



Slika 32. Statistika



Slika 33. Testni koraci za scenarij Statistika

6.9.1. Uspješnost i neuspješnost testnog slučaja

Često se može dogoditi da postoji mogućnost greške na testu, ali se uopće ne radi o pogrešci u funkcionalnosti aplikacije. Testovi rade na principu oznaka (eng. *target*) gdje test cilja da treba obaviti određenu akciju nad elementom koji ima naziv npr. *name=korime* te se nekada može dogoditi da programer na strani poslužitelja u aplikaciji promijeni naziv, tj. oznaku elementa, stoga ju testni scenarij više ne vidi pod nazivom *name=korime* i ispisuje se greška. Također se može dogoditi pad testnog scenarija zbog spore brzine interneta ili nekih drugih vanjskih varijabli koje mogu utjecati na testiranje sustava.

Izvještaj je zato bitan dio automatskog testiranja. U testnim scenarijima u tekstu prije ako postoje greške, nastavlja se rad testnog scenarija, ali pocrveni linija koda u kojoj postoji greška. Ako web aplikacija bude uspješno testirana, tada će sustav pokazati zelenu boju na testnom scenariju kako bi se pokazalo da je scenarij uspješno testiran. Potrebno je uvijek paziti i zadati omjer uspješnosti koji je potrebno imati na dnevnom testiranju kako bi se moglo na vrijeme reagirati s izvođenjem testova.
Ako je testni slučaj pronašao grešku, zacrvenit će liniju koda kod koje je pronašao grešku te će se saznati gdje je u testnom scenariju nastala greška.

7. Zaključak

Automatsko testiranje aplikacije u alatu Selenium puno je bolji izbor od ručnog testiranja aplikacije iako ručno testiranje nikad neće biti u potpunosti zamjenjivo automatskim, već može biti zamijenjeno u određenim testnim slučajevima. Selenium kao alat za automatsko testiranje najbolja je opcija zbog toga što je besplatan, u sebi sadrži brojne alate koji se mogu koristiti za automatsko testiranje te može raditi na svim operacijskim sustavima i web preglednicima. Ručno testiranje nikada neće biti u potpunosti zamijenjeno, ali automatsko testiranje može poduzeću donijeti puno više dobiti jer se smanjuje vrijeme izvođenja testova te se testovi mogu u pozadini odvijati cijelo vrijeme. Najveći problem koji se pojavljuje kod automatskog testiranja postotak je uspješnosti testova jer ponekad on ne prikazuje realnu sliku situacije. Često sam tijekom rada na automatskom testiranju imao problema s prijevodima naziva elemenata. Kako Selenium zna element po nazivu, kada se on promijeni sa strane poslužitelja, tada Selenium ispisuje grešku, a funkcionalnost i dalje radi kako treba. Često se može dogoditi da test bude neuspješan jer je aplikacija u određenom trenutku radila usporeno ili je uređaj na kojem se testira aplikacija radio usporeno u danom trenutku. Problemi se ne događaju često, ali se može zaključiti da su izvještaji automatskog testiranja potrebni te je potrebno izvoditi testove svaki dan u određeno vrijeme i jednom tjedno testirati sve funkcionalnosti aplikacije kako bi se dobila bolja i šira slika rada aplikacije. Automatsko se testiranje u nekim trenutcima može činiti kao dosta zahtjevno te da je nekada jednostavno bolje testirati ručno i to je istina, ali jednom kada se automatski testovi postave te se uklone sve varijable okoline koje mogu smetati pri izvođenju automatskih testova, tada će automatsko testiranje pokazati svoj puni potencijal te će biti povećan postotak pronađenih grešaka u aplikaciji i aplikacija će biti 100% kvalitetno istestirana. Automatsko će testiranje biti budućnost testiranja te će u jednom trenutku u budućnosti, kako se tehnologija bude razvijala, u potpunosti zamijeniti čovjeka kao provjeru kvalitete na aplikacijama.

8. Literatura

[1] http://www.mef-lab.com/praktikum-2018/book/Web%20aplikacije.html

[2] https://hr.wikipedia.org/wiki/CSS

[3] https://hr.wikipedia.org/wiki/HTML

[4] https://hr.wikipedia.org/wiki/PHP

[5] http://www.skolasvilajnac.edu.rs/mreze/osnove_html.htm

[6] https://elf.foi.hr/course/view.php?id=267 Prezentacije s kolegija „Izgradnja web aplikacija“

[7] https://hr.wikipedia.org/wiki/JavaScript

[8] https://www.aubx.com/faks/ergonomija/php-povijest.html

[9] https://softwise.hr/hr_HR/blog/internet-stranica

[10] (Fewster, Graham, 1999)

[11] https://en.wikipedia.org/wiki/Node.js

[12] https://hr.photo-555.com/5472240-what-is-maven

[13] https://en.wikipedia.org/wiki/Selenium_(software)

[14] https://www.guru99.com/introduction-to-selenium-grid.html#:~:text=Selenium%20Grid%20is%20a%20part,server%20acts%20as%20a%20hub

Ostatak literature:

- Knjiga Jon Duckett – JavaScript & Jquery
  John Wiley & Sons,Inc.
  Indianapolis 2011
- Knjiga Jon Duckett – HTML & CSS
  John Wiley & Sons,Inc.
  Indianapolis 2011

9. Popis slika