Aplikacija za provođenje ispitivanja programske podrške

Bilonić, Barbara

SVEUČILIŠTE JOSIPA JUSRJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

APLIKACIJA ZA PROVOĐENJE ISPITIVANJA PROGRAMSKE PODRŠKE

Diplomski rad

Barbara Bilonić

Osijek, 2022.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 16.09.2022.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime Pristupnika:	Barbara Bilonić
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D-1104R, 13.10.2020.
OIB studenta:	28445766298
Mentor:	izv. prof. dr.sc. Josip Job
Sumentor:	,
Sumentor iz tvrtke:	Davor Kedačić
Predsjednik Povjerenstva:	Izv.prof.dr.sc. Ratko Grbić
Član Povjerenstva 1:	izv. prof. dr.sc. Josip Job
Član Povjerenstva 2:	Izv. prof. dr. sc. Časlav Livada
Naslov diplomskog rada:	Aplikacija za provođenje ispitivanja programske podrške
Znanstvena grana diplomskog rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak diplomskog rada:	Testiranje je neizostavni dio cjelokupnog procesa razvoja proizvoda ili usluge. U razvoju programske podrške, kojega je ispitivanje sastavni dio, najčešće se koriste komercijalno dostupni sustavi za upravljanje razvojem softvera. Zadatak ovog rada je izraditi jednostavnu aplikaciju za provođenje ispitivanja programske podrške te omogućiti njezinu integraciju sa sustavom za upravljanje razvojem softvera. Tema rezervirana za: Barbara Bilonić Sumentor iz tvrtke: Davor Kedačić (Institut RT-RK)
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	16.09.2022.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 26.09.2022.

Ime i prezime studenta:	Barbara Bilonić
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-1104R, 13.10.2020.
Turnitin podudaranje [%]:	3

Ovom izjavom izjavljujem da je rad pod nazivom**: Aplikacija za provođenje ispitivanja programske podrške**

izrađen pod vodstvom mentora izv. prof. dr.sc. Josip Job

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Testiranje je bitan dio razvoja programske podrške. Prilikom procesa testiranja testni se slučajevi mogu svrstati u testne ciljeve koji se zatim svrstavaju u testne planove. Ovakav način organiziranja testnih slučajeva može olakšati proces testiranja. Za praćenje provođenja postupka ispitivanja programske podrške često se koriste softverski alati koji nisu ograničeni samo na postupak testiranja, nego imaju puno širu funkcionalnost, tj. koriste se alati koji obuhvaćaju cjelokupni proces razvoja programske podrške. Uovom će radu biti dan prijedlog web aplikacije koja će omogućiti upravljanje testnim planovima, ciljevima i slučajevima. Aplikacija treba imati mogućnost integracije s komercijalnim alatom za praćenje razvoja programske podrške iz kojeg će dobivati podatke o projektima, testnim planovima, testnim ciljevima i testnim slučajevima. Aplikacija, također, treba omogućiti praćenje statistike o broju izvedenih testnih slučajeva kao i mogućnosti dodjeljivanja testnih slučajeva testerima na izvršavanje. U ovom radu bit će dan pregled dostupnih rješenja za praćenje razvoja programske podrške s opisom pet različitih alata. Dalje će biti opisana arhitektura i dizajn aplikacije, tj. od čega se aplikacija treba sastojati te s kojim vanjskim izvorima treba vršiti komunikaciju kao i na koji način će ova komunikacija biti ostvarena. U četvrtom će poglavlju biti demonstrirane osnovne funkcionalnosti koje nudi aplikacija. Na kraju bit će dan zaključak o odabranim tehnologijama i postignutim funkcionalnostima.

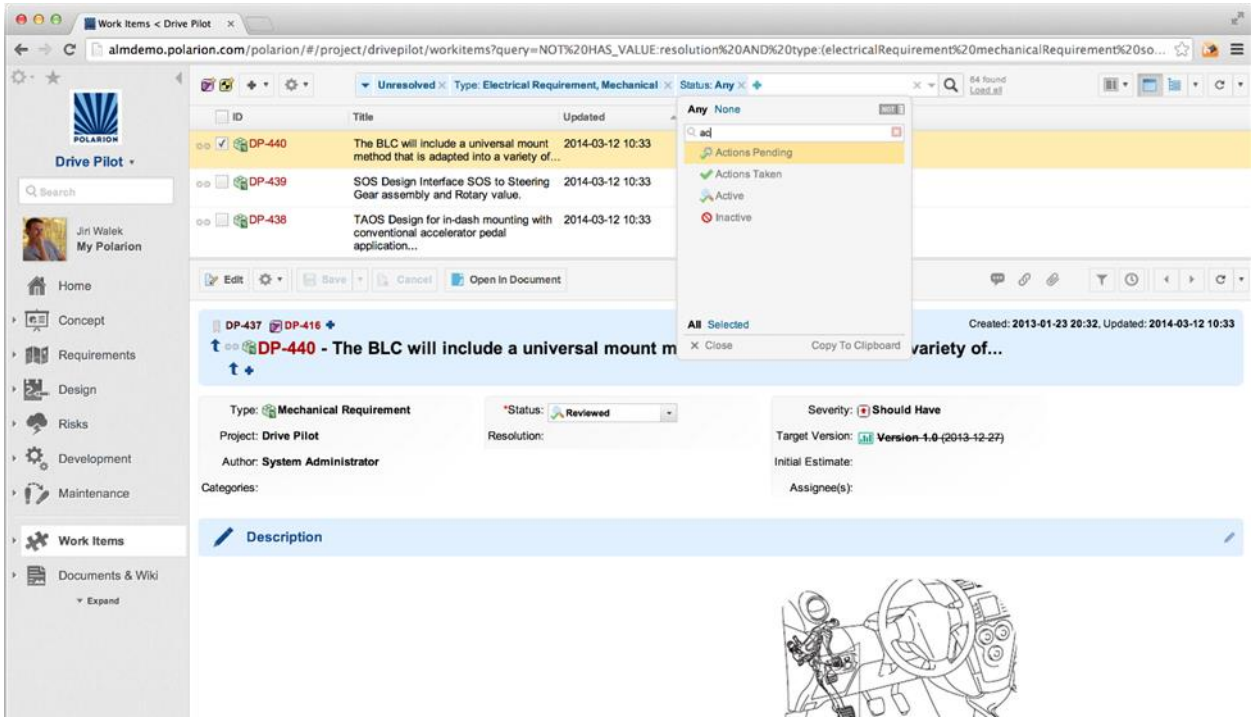
1.1. Zadatak diplomskog rada

Testiranje je neizostavni dio cjelokupnog procesa razvoja proizvoda ili usluge. U razvoju programske podrške, kojega je ispitivanje sastavni dio, najčešće se koriste komercijalno dostupni sustavi za upravljanje razvojem softvera. Zadatak ovog rada je izraditi jednostavnu aplikaciju za provođenje ispitivanja programske podrške te omogućiti njezinu integraciju sa sustavom za upravljanje razvojem softvera.

2. PREGLED DOSTUPNIH RJEŠENJA ZA PRAĆENJE RAZVOJA PROGRAMSKE PODRŠKE

Trenutno, na tržištu se nalazi mnogo dostupnih alata za praćenje razvoja programske podrške. Iako svaki alat pruža različite mogućnosti, svima im je zajedničko omogućavanje praćenja razvoja programske podrške od definiranja zahtjeva pa sve do testiranja i održavanja. Neki od najpoznatijih alata trenutno dostupnih na tržištu su Polarion, Azure DevOps, SpiraTeam, Visure i Windchill.

Polarion je alat za praćenje razvoja programske podrške razvijen od strane tvrtke Siemens. Nudi mnoge mogućnosti za praćenje i sam razvoj programske podrške. Slika 2.1. prikazuje korisničko sučelje Polarion alata.



Slika 2.1. *Korisničko sučelje Polarion alata* [1].

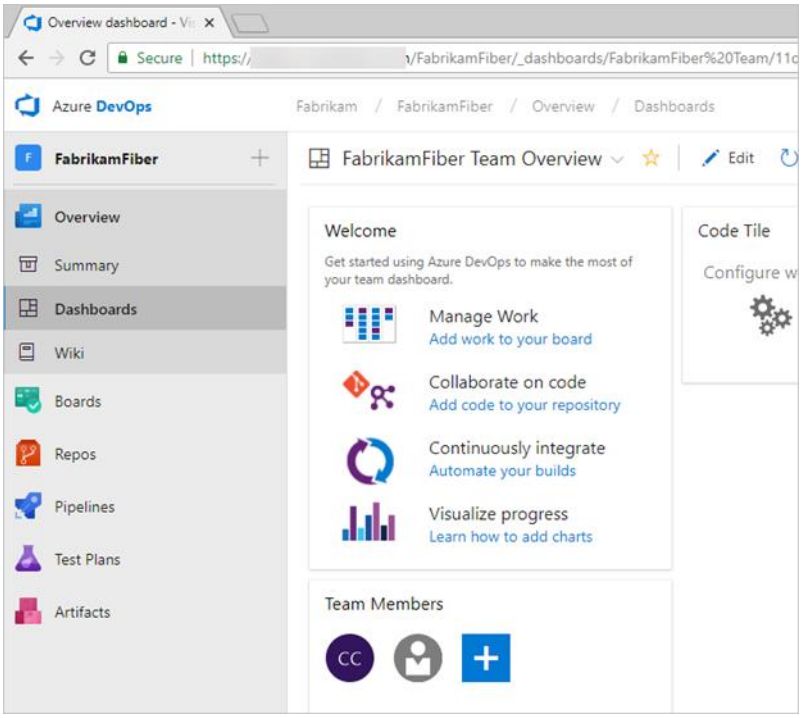
Polarion omogućuje visoku razinu komunikacije između članova projekta preko rasprava, obavijesti, podsjetnika i slično. Također, omogućuje lak rad sa zahtjevima i specifikacijama na programsku podršku, njihovo stvaranje, uvoz i izvoz te njihovu potvrdu pomoću digitalnih potpisa. Polarion olakšava proces upravljanja kvalitetom omogućujući stvaranje testnih slučajeva, njihovog povezivanja sa zahtjevima ili drugim testnim slučajevima, praćenje i kreiranje izvješća o programskim greškama, integraciju s alatima za automatsko izvršavanje testova i slično. Probleme ili greške u programskom rješenju moguće je razvrstavati na kategorije te raditi analize i statistike koji zahtjevi imaju koliki broj problema ili grešaka te kolike su im izmjene potrebne. Također,

omogućava i ponovno korištenje ili dijeljenje zahtjeva, specifikacija, testnih slučajeva i slično između više različitih projekata [1].

Azure DevOps alat razvijen je od strane tvrtke Microsoft. Omogućava rad u oblaku koristeći Azure DevOps Services ili lokalno koristeći Azure DevOps Server. Neke od funkcionalnosti koje nudi ovaj alat su:

- Azure Repos - nudi mogućnost određivanja verzija programske podrške,
- Azure Pipelines - omogućuje laku izgradnju i puštanje servisa u upotrebu,
- Azure Boards - omogućuje planiranje i praćenje pogrešaka u kodu, rada i problema,
- Azure Test Plans - pruža alate za testiranje programske podrške.

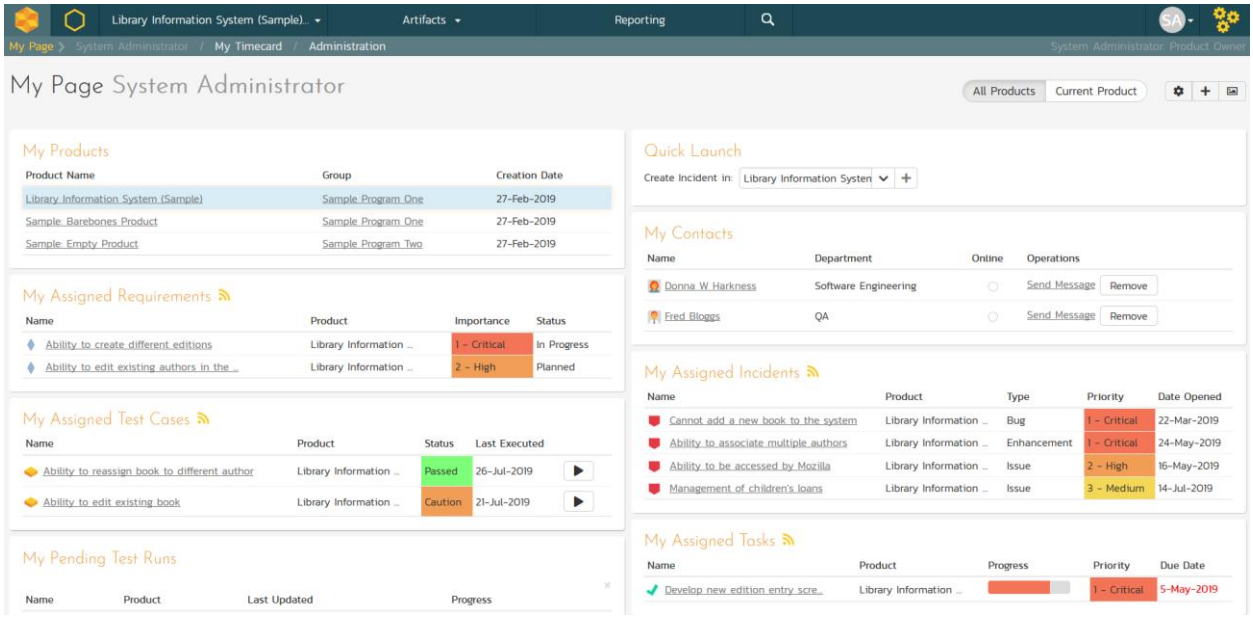
Omogućeno je i praćenje manualnog, istraživačkog i automatskog testiranja te praćenje rezultata testova kao i njihovih analiza i statistika [2]. Slika 2.2. prikazuje korisničko sučelje Azure DevOps alata.



Slika 2.2. *Korisničko sučelje Azure DevOps alata* [2].

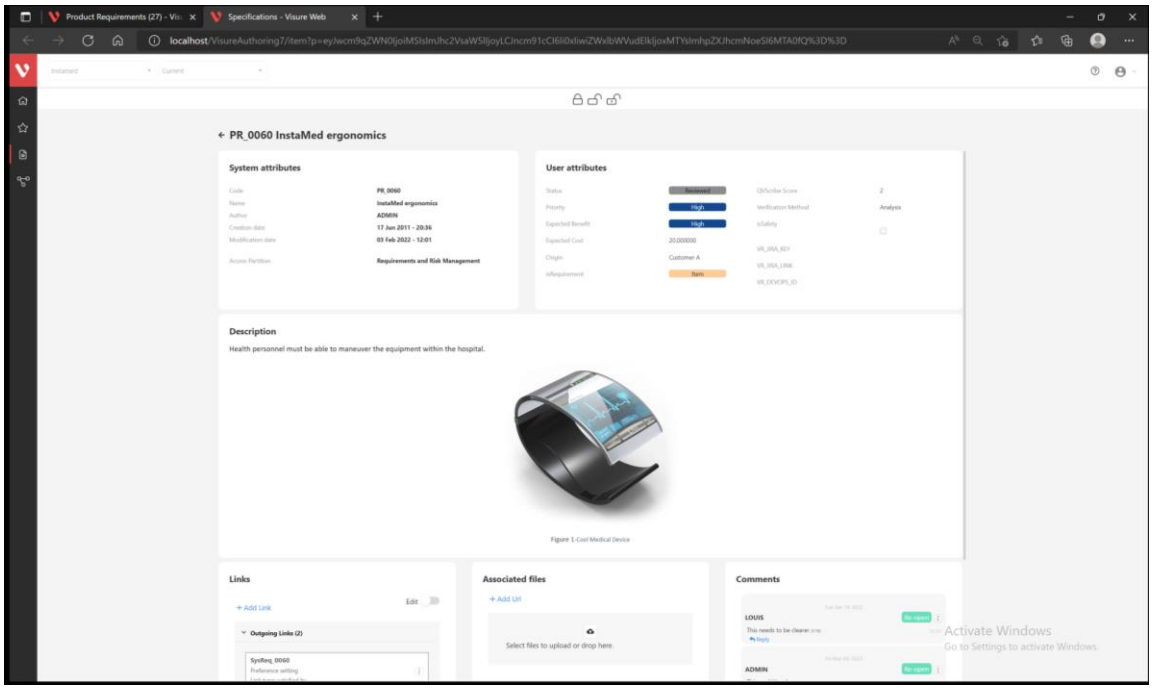
SpiraTeam razvila je tvrtka Inflectra. Ovaj alat omogućuje rad sa zahtjevima na programsku podršku, testovima, planovima, zadatcima, programskim greškama i problemima. Pruža mogućnosti izrade projekata i zahtjeva na programsku podršku, izradu plana izvedbe prema definiranim zahtjevima, kreiranja zadataka te njihovo dodjeljivanje pojedinim članovima tima kao i prikaza trenutno definiranih zadataka i napretka na projektu. Omogućuje kontrolu kvalitete kroz

alate za upravljanje zahtjevima na programsku podršku i testnim slučajevima. Također, podržava praćenje i bilježenje programskih pogrešaka kao i manualno i automatsko testiranje. Omogućava laku izmjenu informacija omogućenu kroz laku izradu i dijeljenje dokumenata kao i mogućnost slanja direktnih poruka te slanja obavijesti na e-mail [3]. Slika 2.3. prikazuje korisničko sučelje SpiraTeam alata.



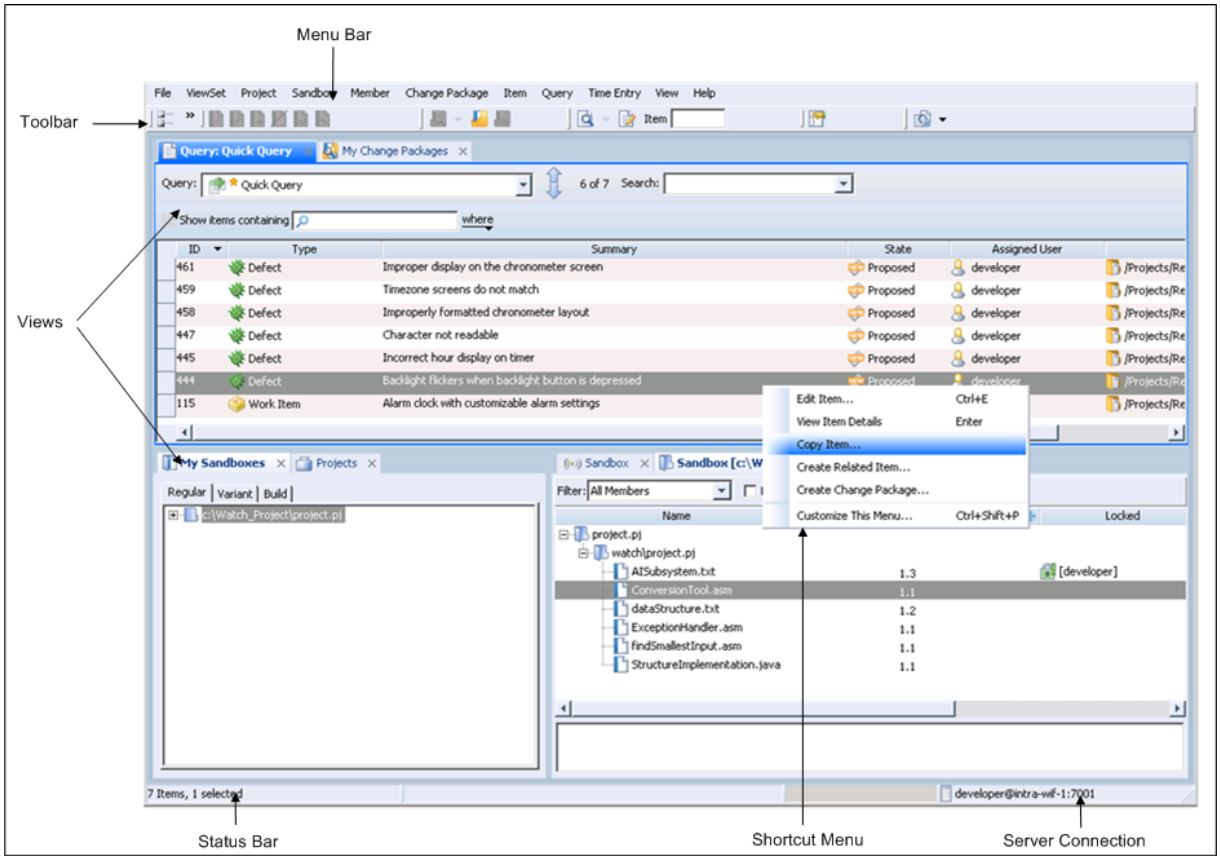
Slika 2.3. *Korisničko sučelje SpiraTeam alata* [3].

Visure je alat za praćenje razvoja programske podrške razvijen od istoimene tvrtke. Omogućuje izradu, rad i upravljanje zahtjevima na programsku podršku. Omogućuje analizu i procjenu rizika prilikom svake faze razvoja programske podrške. Visure ima mogućnost upravljanja testovima te njihovog povezivanja s definiranim zahtjevima. U ovom je alatu moguće pratiti programske greške i probleme. Još je jedna mogućnost uvoz i izvoz podataka iz alata kao što su MS Word i MS Excel. Kako bi olakšao proces provjere, odgovara li programska podrška zadanim standardima, Visure sadržava predloške za neke standarde kao što su SCRUM, IEC 62304, CENELEC EN 50128 i drugi [4]. Slika 2.4. prikazuje korisničko sučelje Visure alata.



Slika 2.4. *Korisničko sučelje Visure alata* [4].

Windchill je alat razvijen od strane tvrtke PTC. Slika 2.5. prikazuje korisničko sučelje Windchill alata.



Slika 2.5. *Korisničko sučelje Windchill alata* [5].

Windchill omogućuje definiranje, rad i upravljanje sa specifikacijama i zahtjevima na programsku podršku. Omogućuje praćenje verzija i povijesti svih dokumenata. Također, omogućuje stvaranje testnih planova i testnih slučajeva te bilježenje njihovih rezultata. Testove je moguće povezivati sa zahtjevima i specifikacijama koje validiraju. Podržava i manualno i automatsko testiranje. Još jedna od ponuđenih funkcionalnosti jest kreiranje izvješća u kojem korisnici imaju mogućnost odabrati tip željenog izvješća i podatke koje izvješće treba sadržavati [5].

Svi ovi alati nude mogućnosti praćenja razvoja programske podrške, no neke funkcionalnosti, prednosti i cijena razlikuju se od alata do alata. Svi alati omogućuju kreiranje zahtjeva na programsku podršku i testnih slučajeva kao i neke forme analize rezultata i samog procesa razvoja programske podrške te integraciju s drugim alatima. Jedna od funkcionalnosti koja izdvaja SpiraTeam alat mogućnost je slanja direktnih poruka. Prednost je Visure alata to što nudi predloške za mnoge postojeće standarde.

3. PRIJEDLOG RJEŠENJA

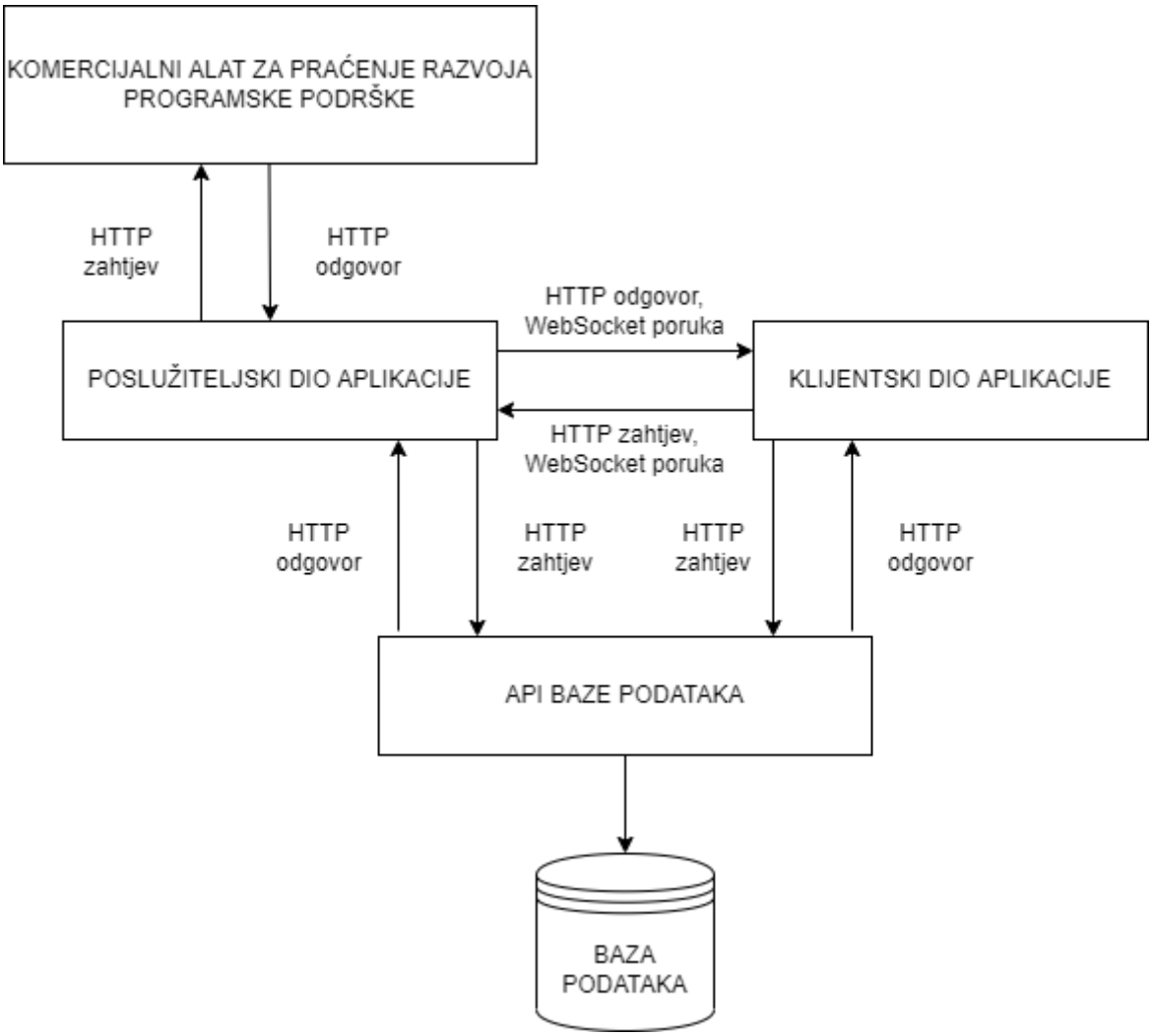
Aplikacija treba omogućiti integraciju s komercijalnim alatom za praćenje razvoja programske podrške. Potrebno je omogućiti kreiranje testnog plana koji će se spremiti u bazu na temelju dostupnih projekata i testnih planova dohvaćenih iz komercijalnog alata za praćenje razvoja programske podrške. Aplikacija treba omogućiti uređivanje prethodno izrađenih testnih planova. Treba sadržavati sučelje za izvođenje testova te sučelje za upravljanje testovima.

Na sučelju namijenjenom za izvođenje testova, tester treba moći vidjeti sve projekte koji sadržavaju testne planove iz kojih su neki testni slučajevi dodijeljeni njemu kao i same testne planove. Za odabrani testni plan tester treba moći vidjeti sve testne ciljeve koji sadržavaju testne slučajeve koji su mu dodijeljeni kao i same testne slučajeve. Testeru je potrebno omogućiti i unošenje rezultata pojedinog izvršenog testa ili unošenje rezultata za više odabranih testnih slučajeva.

Na sučelju za upravljanje testovima potrebno je omogućiti prikaz svih testnih planova koji su prethodno kreirani kao i svih projekata koji sadrže te testne planove. Za odabrani testni plan potrebno je prikazati sve testne ciljeve koje sadrži kao i sve testne slučajeve unutar tih testnih ciljeva. Također, potrebno je omogućiti odabir jednog ili više testnih ciljeva koji će se dodijeliti odabranom testeru.

3.1. Arhitektura i dizajn rješenja

U ovom potpoglavlju bit će prikazani i opisani arhitektura i dizajn rješenja problema. Slika 3.1. prikazuje arhitekturu rješenja.



Slika 3.1. *Arhitektura aplikacije.*

Kao što je vidljivo na slici 3.1. poslužiteljski dio aplikacije komunicira s API-em baze podataka, alatom za praćenje razvoja programske podrške i klijentskim dijelom aplikacije. Klijentski dio komunicira s poslužiteljskim dijelom aplikacije i API-em baze podataka. Iako se podatci o projektima, testnim planovima, ciljevima, slučajevima i sesijama dohvaćaju iz alata za praćenje razvoja programske podrške potrebna je baza podataka za skladištenje dodatnih podataka kako bi se mogle provoditi dodatne analize.

Poslužiteljski dio aplikacije od alata za praćenje razvoja programske podrške dobiva podatke o dostupnim projektima, nakon čega, za svaki projekt, dohvaća testne planove koji se u njemu nalaze. Za svaki testni plan, iz alata za praćenje programske podrške, dohvaća se lista testnih ciljeva koje sadržava te se za svaki testni cilj može dohvatiti popis testnih slučajeva koje sadržava i testnih sesija koje su povezane s njim. Za svaki od tih elemenata, osim što se dohvaćaju podelementi koje sadržava, dohvaćaju se i podatci o njima kao što je naziv, identifikacijski broj, opis i status. Poslužiteljski dio aplikacije u alat za praćenje razvoja programske podrške šalje

podatke o rezultatima izvođenja testnih slučajeva. Komunikacija s bazom podataka odvija se preko API-a baze podataka. Iz baze se podataka dohvaćaju podatci o dostupnim projektima, testnim planovima, testnim ciljevima, testnim slučajevima i testnim sesijama. U bazu se podataka spremaju podatci o rezultatima testova kao i podatci o projektima, testnim planovima, testnim ciljevima, testnim slučajevima i testnim sesijama. Poslužiteljski dio s klijentskim dijelom aplikacije razmjenjuje podatke o projektima te testnim planovima, ciljevima, slučajevima i sesijama koje dohvaća iz alata za praćenje razvoja programske podrške.

Klijentski dio aplikacije komunicira s poslužiteljskim dijelom aplikacije i API-em baze podataka. Od poslužiteljskog dijela aplikacije dohvaćaju se podatci koji se nalaze u alatu za praćenje razvoja programske podrške i šalju mu se podatci koje je potrebno unijeti u taj alat. Također se s poslužiteljskim dijelom aplikacije razmjenjuju podatci o promjenama zbog kojih je potrebno osvježavati podatke u klijentskom dijelu. Iz baze podataka dohvaćaju se potrebni podatci o projektima, testnim planovima, testnim ciljevima, testnim slučajevima i testnim sesijama.

3.1.1. Mehanizmi komunikacije između klijentskog i poslužiteljskog dijela aplikacije

Klijentski dio aplikacije koristi AJAX za komunikaciju s poslužiteljskim dijelom aplikacije. Pomoću AJAX-a pozivaju se URL-ovi poslužiteljskog dijela aplikacije definirani u *Djangu.* U zahtjevima se pomoću AJAX-a šalju potrebni podatci. Primjer AJAX poziva prikazan je na slici 3.1.

```
1:      $.ajax(
2:          {
3:              type: "GET",
4:              url: "/urlPath",
5:              data: {
6:                  'number':123
7:              },
8:              success: function () {
9:                  alert("SUCCESS");
10:             },
11:             error: function(){
12:                 alert("ERROR");
13:             }
14:      });
```

Slika. 3.1. *Primjer koda pravljenja poziva s AJAX-om.*

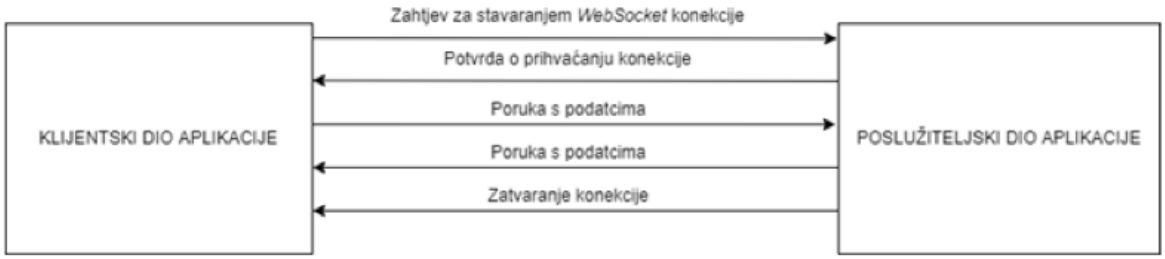
Na primjeru sa slike 3.1. moguće je vidjeti što je potrebno staviti u AJAX zahtjev. Potrebno je staviti tip HTTP zahtjeva (*type*), putanju na koju se zahtjev radi (*url*) i podatke (*data*), ukoliko ih je potrebno poslati. Također, definirano je što će se dogoditi u slučaju uspješnog slanja zahtjeva i

u slučaju da dođe do greške. U ovom primjeru će se samo prikazati prozor upozorenja s odgovarajućom porukom.

Za komunikaciju pri kojoj poslužiteljski dio aplikacije treba poslati podatke ili obavijest klijentskom dijelu bez da je on napravio zahtjev koristi se protokol *WebSocket*. *WebSocket*-i omogućavaju kreiranje otvorenog komunikacijskog kanala između poslužiteljskog i klijentskog dijela. Za implementaciju *WebSocket*-a u poslužiteljskom dijelu korištena je *Django Channels* biblioteka. Ova biblioteka nudi sinkronu i asinkronu korisničku klasu koju je moguće naslijediti kako bi se napravila vlastita korisnička (engl. *consumer*) klasa. Za potrebe ove aplikacije bit će korištena askinkrona korisnička klasa. Klasa nudi funkcije za obradu događaja prilikom rada s *WebSocket*-ima kao što su *websocket_connect*, *websocket_recieve* i *websocket_disconnect*. Funkcija *websocket_connect* prima i šalje poruke za stvaranje konekcije, *websocket_recieve* poziva se kada kroz stvorene kanale dođe poruka, a funkcija *websocket_disconnect* zatvara otvoreni kanal na kraju komunikacije. Prilikom spajanja na *WebSocket* svaki klijent dobije svoju instancu korisničke klase. Ovaj tip komunikacije koristi se za osvježavanje podataka na sučelju za izvođenje testova kada se testeru dodijele testni slučajevi. Za svakog testera kreira se poseban kanal koji kao naziv ima njegov identifikacijski broj te se na taj način određuje kroz koje kanale se šalju poruke. *WebSocket* komunikacija koristi se i za osvježavanje sučelja za upravljanje testovima kako bi se prikazalo ime novo dodijeljenog testera ili uklonilo ime postojećeg.

U klijentskom dijelu potrebno je kreirati *WebSocket* objekt s putanjom definiranom u poslužiteljskom dijelu aplikacije u koju je potrebno dodati identifikacijski broj prijavljenog testera ili menadžera. Na kreirani objekt potrebno je dodati *onMessage* metodu koja će se pozvati svaki put kada se primi poruka putem *WebSocket*-a.

Dijagram koji prikazuje *WebSocket* komunikaciju između poslužiteljskog i klijentskog dijela komunikacije prikazan je na slici 3.2.



Slika 3.2. *Komunikacija između klijentskog i poslužiteljskog dijela aplikacije putem WebSocket protokola.*

Kao što je prikazano na slici 3.2. klijentski dio aplikacije prvo šalje zahtjev za stvaranje *WebSocket* konekcije poslužiteljskom dijelu. Poslužiteljski dio vraća potvrdu o prihvaćenom zahtjevu za konekciju. Kada na klijentskom dijelu aplikacije dođe do odgovarajuće promjene iz njega se poslužiteljskom dijelu aplikacije šalje odgovarajuća poruka. Kada se dogodi promjena zbog koje je potrebno poslati poruku klijentskom dijelu poslužiteljski dio aplikacije putem *WebSocket*-a šalje odgovarajuću poruku. Na kraju komunikacije se zatvara *WebSocket* konekcija.

3.2. Korištene tehnologije

U ovom će dijelu biti opisane tehnologije i razvojni okviri koji će se koristiti za izradu aplikacije.

Praćenje razvoja programske podrške obuhvaća praćenje svih procesa i aktera uključenih u njezin razvoj [6]. Praćenje razvoja može se koristiti uz bilo koju metodu razvoja programske podrške bilo da se radi o vodopadnoj, spiralnoj, agilnoj ili nekoj drugoj metodi razvoja. Ono što praćenje razvoja programske podrške nudi je sljedivost (engl. *traceability*) koda, automatizacija procesa i uvid u napredak razvoja programske podrške. Obuhvaća praćenje svih dijelova razvoja koje možemo podijeliti na upravljanje projektom, arhitekturu, analizu i dizajn, testiranje i upravljanje izdanjima [6]. Praćenje razvoja programske podrške obuhvaća i izradu dokumenata kao što su zahtjevi na programsku podršku, specifikacije, testovi i slično.

Za potrebe ove aplikacije dio razvoja životnog ciklusa programske podrške koji je potreban je testiranje. Alat koji se koristi trebao bi imati mogućnost izrade projekata, testnih planova, testnih ciljeva, testnih sesija i testnih slučajeva. Projekt sadrži sve informacije i dokumente o razvoju i planiranju određene programske podrške. Testni plan koristi se kao cjelokupni plan testiranja neke verzije programske podrške. Testni planovi mogu sadržavati testne ciljeve koji se koriste za grupiranje testnih slučajeva u cjeline. Testni slučajevi sadržavaju podatke o testovima koji se koriste za ispitivanje neke određene funkcionalnosti programske podrške. Testne sesije zapravo predstavljaju jedno izvršavanje testnih slučajeva koje sadrže. Alat također treba imati API koji omogućuje dohvaćanje podataka o projektima, testnim planovima, testnim ciljevima i testnim sesijama.

Za izradu poslužiteljskog dijela aplikacije bit će korišten *Django*. To je *Python* web razvojni okvir visokog nivoa [7]. *Django* omogućuje izradu prikaza (engl. *view*). Prikazi kao povratne vrijednosti daju HTTP odgovore koji mogu sadržavati određene tražene podatke ili HTML predloške koje je moguće unaprijed definirati. Kako bi se izvan *Django* aplikacije mogle pozvati određene funkcije iz skripte za prikaz, *views.py*, potrebno je definirati URL putanje koje se pridružuju određenim

funkcijama. Također, omogućuje rad s bazom podataka izradom modela pomoću kojih se mapira baza podataka.

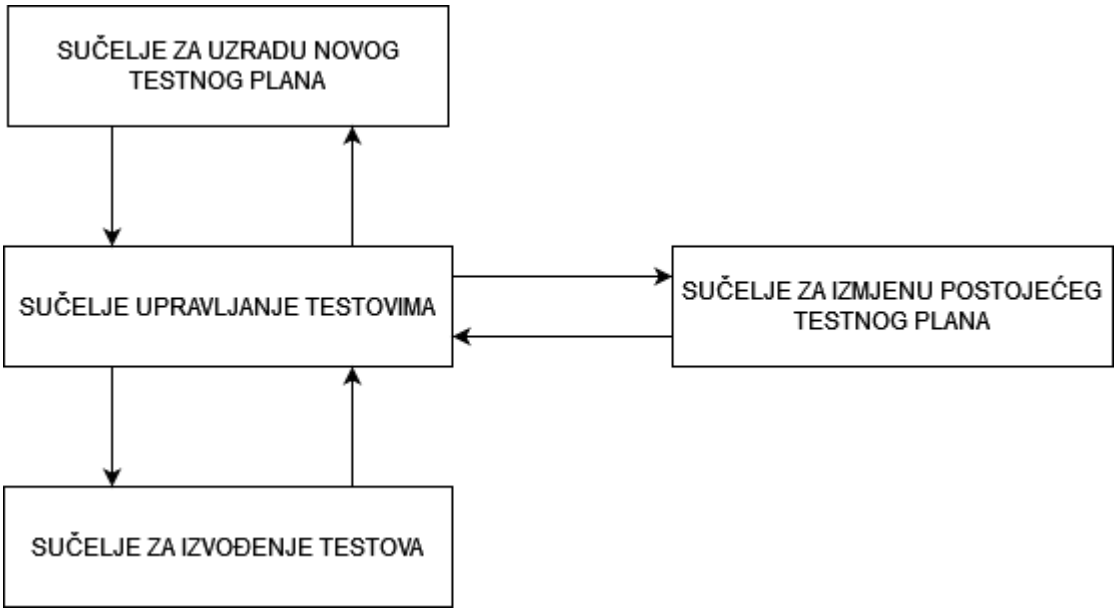
Za izradu klijentskog dijela aplikacije bit će korišteni *JavaScript* i *JavaScript* biblioteka *jQuerry*.

JavaScript programski je jezik visokog nivoa koji se većinom koristi kao skriptni jezik za izradu web stranica. Često se kompajlira tijekom samog izvođenja (engl. *Just-in-time*). Podržava više programskih paradigmi koje su objektno orijentirana, imperativna i funkcijska [8].

jQuerry je JavaScript biblioteka koja nudi mnoge funkcionalnosti. Nudi mogućnosti kao što su dohvaćanje, izrada i manipulacija HTML elementima, upravljanje događajima, animacije i korištenje AJAX-a [9].

3.3. Klijentski dio aplikacije

Klijentski dio aplikacije treba sadržavati i prikazivati četiri sučelja. Web sučelje za izradu novog testnog plana na temelju testnog plana iz alata za praćenje razvoja programske podrške, web sučelje za uređivanje postojećeg testnog plana, web sučelje za izvođenje testova i web sučelje za upravljanje testovima. Slika 3.3. prikazuje način kretanja kroz web sučelja.



Slika 3.3. *Prikaz mogućnosti kretanja kroz web sučelja koja nudi klijentski dio aplikacije.*

Kao što je vidljivo sa slike 3.3. sa sučelja za upravljenje testovima moguće je doći do svih drugih sučelja i vratiti se nazad na njega. Prilikom korištenja aplikacije potrebno je prvo izraditi novi testni plan. Zatim je potrebno njegove testove dodijeliti testerima na sučelju za upravljanje testovima. Tek nakon toga tester će moći vidjeti testove na sučelju za izvođenje testova. Kreirani se testni plan bilo kada može izmijeniti.

3.3.1. Izrada novog testnog plana

Ovo web sučelje treba omogućiti izradu novog testnog plana čiji će se podatci spremati u bazu podataka. Na sučelju treba biti omogućeno učitavanje projekata i testnih planova iz alata za praćenje programske podrške. U *select* HTML elementu potrebno je prikazati sve dostupne projekte te za odabrani projekt sve dostupne testne planove. Za testni je plan potrebno odrediti naziv, datum početka, datum završetka te broj testnih stanica dostupnih za izvođenje testnih slučajeva tog testnog plana. Nakon odabira željenog testnog plana i unošenja svih potrebnih podataka za njega dohvaćaju se dostupni testni ciljevi, testni slučajevi i testne sesije te se spremaju u bazu podataka. U tablici 3.1. mogu se vidjeti sve funkcije potrebne u skripti za izradu novog testnog plana.

Tablica 3.1. *Funkcije potrebne u skripti za izradu testnog plana.*

Funkcija	Opis
Funkcija za dohvaćanje projekata	Funkcija treba poslati AJAX zahtjev na URL definiran u poslužiteljskom dijelu aplikacije kako bi dohvatila listu dostupnih projekata iz alata za praćenje razvoja programske podrške. Ukoliko su podatci uspješno dohvaćeni treba pozvati funkciju za učitavanje projekata.
Funkcija za učitavanje projekata	Funkcija treba učitati dohvaćene projekte u *select* HTML element.
Funkcija za dohvaćanje testnih planova	Funkcija na temelju izabranog projekta treba dohvatiti testne planove koje on sadrži. Funkcija treba napraviti zahtjev pomoću AJAX-a na URL definiran u poslužiteljskom dijelu aplikacije. Ukoliko su podatci uspješno dohvaćeni potrebno je pozvati funkciju za učitavanje testnih planova.
Funkcija za učitavanje testnih planova	Funkcija treba predane testne planove učitati u odgovarajući HTML *select* element.
Funkcija za izradu testnog plana	Funkcija treba dohvatiti podatke iz odgovarajućih HTML polja, napraviti provjeru jesu li uneseni svi podatci i jesu li podatci validni te napraviti zahtjev pomoću AJAX-a na URL definiran u poslužiteljskom dijelu u kojem će poslati potrebne podatke o testnom planu.

3.3.2. Uređivanje postojećeg testnog plana

Ovo web sučelje treba omogućavati izmjenu naziva testnog plana, njegovog planiranog datuma početka i završetka te broja i naziva testnih stanica na kojima se mogu izvršavati njegovi testni slučajevi. Također, treba omogućiti brisanje testnog plana. Web stranici se kao URL parametar treba predati identifikacijski broj testnog plana. Podatci o testnom planu koje je potrebno prikazati

su projekt, testni plan, naziv testnog plana, datum početka, datum završetka i popis testnih stanica. Tablica 3.2. prikazuje funkcije koje trebaju biti sadržane u skripti za uređivanje postojećeg testnog plana.

Tablica 3.2. *Funkcije potrebne u skripti za uređivanje postojećeg testnog plana*.

Funkcija	Opis
Funkcija za dohvaćanje podataka o testnom planu	Funkcija treba napraviti zahtjev prema API-u baze podataka pomoću AJAX-a s identifikacijskim brojem testnog plana kao parametrom kako bi dobila potrebne podatke o testnom planu. Ukoliko su podatci uspješno dohvaćeni potrebno je pozvati funkciju za učitavanje podataka o testnom planu.
Funkcija za učitavanje podataka o testnom planu	Funkcija predane podatke o testnom planu treba učitati u odgovarajuća HTML polja.
Funkcija za dohvaćanje podataka o testnim stanicama	Funkcija treba napraviti zahtjev prema API-u baze podataka kako bi dohvatila potrebne podatke o testnim stanicama. Ako su podatci uspješno dohvaćeni potrebno je pozvati funkciju za učitavanje testnih stanica. Funkcija također podatke o testnim stanicama sprema u globalnu varijablu.
Funkcija za učitavanje testnih stanica	Funkcija za svaku testnu stanicu iz predane liste treba napraviti odgovarajući HTML element te učitati naziv stanice.
Funkcija za brisanje testnog plana	Funkcija treba napraviti poziv na odgovarajuću krajnju točku (engl. *endpoint*) API-a baze podataka kako bi se testni plan obrisao iz baze.
Funkcija za izmjenu testnog plana	Funkcija treba dohvatiti sve podatke koji su podložni izmjenama iz odgovarajućih HTML elementa. Ukoliko su svi potrebni podatci uneseni i validni, funkcija treba napraviti zahtjev na odgovarajuću krajnju točku API-a baze podataka u kojem će poslati sve potrebne podatke za izmjenu testnog plana.
Funkcija za izmjenu podataka o testnim stanicama	Funkcija se poziva svaki put kada se dogodi promjena u broju ili nazivima testnih stanica. Prilikom svake izmjene mijenjaju se podatci u globalnoj varijabli koja sadržava podatke o testnim stanicama.
Funkcija za izmjenu podataka o testnim stanicama u bazi podataka	Funkcija treba napraviti zahtjev na odgovarajuću krajnju točku API-a baze podataka koristeći AJAX u kojem će poslati podatke o testnim stanicama kako bi se podatci izmijenili u bazi podataka.

3.3.3. Izvođenje testova

Na ovom web sučelju potrebno je testeru prikazati sve testne slučajeve koji su njemu dodijeljeni na izvršavanje te informacije o njima i omogućiti mu unos rezultata za izvršene testne slučajeve. Za prikaz testnih slučajeva prvo je potrebno prikazati projekte koji sadrže testne planove s testnim slučajevima koji su dodijeljeni tom testeru. Projekte je potrebno prikazati u HTML *select* elementu. Za odabrani projekt u drugi HTML *select* element potrebno je učitati sve testne planove dostupne za taj projekt koji sadrže testove dodijeljene tom testeru. Za odabrani testni plan prikazuju se testni ciljevi koji sadrže testne slučajeve dodijeljene tom testeru kao i same testne slučajeve. Testne ciljeve i testne slučajeve potrebno je prikazati u obliku strukture stabla. Kao viša razina prikazuju se testni ciljevi čija se lista testnih slučajeva može zatvoriti ili prikazati pritiskom tipke miša na njihove kontejnere. U tablici 3.3. su nabrojane neke od funkcija koje trebaju biti sadržane unutar skripte za izvođenje testova.

Tablica 3.3. *Funkcije potrebne u skripti za izvođenje testova.*

Funkcija	Opis
Funkcija za dohvaćanje projekata	Funkcija treba napraviti zahtjev prema odgovarajućoj krajnjoj točki API-a baze podataka koristeći AJAX. Kao parametar treba poslati identifikacijski broj testera. Cilj je dobiti popis projekata koji sadrže testne planove čiji su testni slučajevi dodijeljeni testeru. Ako su podatci uspješno dohvaćeni poziva funkciju za dohvaćanje testnih planova i funkciju za učitavanje projekata.
Funkcija za učitavanje projekata	Funkcija učitava projekte iz predane liste u HTML *select* element.
Funkcija za dohvaćanje testnih planova	Funkcija treba napraviti zahtjev prema API-u baze podataka koristeći AJAX kako bi dobila listu testnih planova iz odabranog projekta koji sadržavaju testove dodijeljene testeru. Ukoliko su podatci uspješno dohvaćeni poziva funkciju za učitavanje testnih planova.
Funkcija za učitavanje testnih planova	Funkcija učitava testne planove iz predane liste u HTML *select* element.
Funkcija za izradu *WebSocket* objekta	Funkcija kreira *WebSocket* objekt s URL-om definiranim u poslužiteljskom dijelu aplikacije.
Funkcija za dohvaćanje sadržaja testnog plana	Funkcija treba napraviti zahtjev prema odgovarajućoj krajnjoj točki API-a baze podataka kako bi dobila sadržaj testnog plana koji sadržava popis testnih ciljeva i testnih slučajeva za taj testni plan. Ukoliko su podatci uspješno dohvaćeni poziva funkciju za učitavanje sadržaja testnog plana.

Funkcija za učitavanje sadržaja testnog plana	Funkcija za svaki testni cilj sadržan u podatcima o testnom planu poziva funkciju za njegovo učitavanje
Funkcija za učitavanje testnih ciljeva	Funkcija izrađuje potrebne HTML elemente za prikaz testnog cilja te po završetku poziva funkciju za učitavanje testnih slučajeva sadržanih u tom testnom cilju.
Funkcija za učitavanje testnih slučajeva	Funkcija izrađuje potrebne HTML elemente za prikaz testnog slučaja i u njih stavlja odgovarajuće podatke
Funkcija za dohvaćanje podataka o testnom slučaju	Funkcija radi AJAX zahtjev prema poslužiteljskom dijelu aplikacije kako bi dobila podatke o testnom slučaju iz alata za praćenje razvoja programske podrške. Ukoliko su podatci uspješno dohvaćeni poziva funkciju za učitavanje podataka o testnom slučaju.
Funkcija za učitavanje podataka o testnom slučaju	Funkcija izrađuje potrebne HTML elemente i učitava podatke o testnom slučaju u njih.
Funkcija za osvježavanje rezultata testova	Funkcija treba izmijeniti vrijednost polja koje prikazuje rezultat za pojedini testni slučaj nakon unosa rezultata.
Funkcija za osvježavanje projekata, testnih planova, testnih ciljeva i testnih slučajeva nakon dodijele novih testova	Funkcija se poziva nakon primitka obavijesti putem *Websocket*-a nakon dodijele ili uklanjanja testnih slučajeva testeru. Funkcija treba ponovno učitati podatke o projektima, testnim planovima, testnim ciljevima i testnim slučajevima.

Testne ciljeve i testne slučajeve potrebno je prikazati u obliku stabla gdje testni ciljevi predstavljaju prvu, a testni slučajevi drugu razinu. Kako bi se to postiglo prvo je potrebno za svaki testni cilj napraviti *div* HTML element koji predstavlja kontejner za cijeli testni cilj zajedno s njegovim testnim slučajevima.

```
1:      let testObjectiveContainer=document.createElement("div");
2:      testObjectiveContainer.setAttribute("class","test-objective-outer-
3:      container");
4:      testObjectiveContainer("id","test-objective-outer-container-"+ID);
```

Slika 3.4. *Primjer koda izrade div HTML elementa.*

Na slici 3.4. vidi se primjer izrade *div* HTML elementa. U ovom slučaju kreira se kontejner koji će sadržavati cijeli testni cilj s njegovim testnim slučajevima. Kontejneru se postave klasa i identifikacijski broj (*id*) kako bi se dalje mogao lako dohvatiti. Dalje se izrađuje još jedan *div* HTML element koji će služiti kao kontejner za podatke o testnom cilju kao što su naziv, status, identifikacijski broj i slično. Taj kontejner izrađuje se na isti način kao i prethodni. Na kontejner koji sadrži podatke o testnom cilju također se doda slušatelj događaja (engl. *event listener*) koji će obaviti određene radnje prilikom pritiska tipke miša na taj *div* HTML element.

```
1:      testObjectiveContainer.addEventListener("click",function(){
2:         loadTests(tests);
3:      });
```

Slika 3.5. *Primjer koda dodavanja slušatelja događaja na kontejner testnog cilja*.

Na slici 3.5. prikazan je kod pomoću kojeg se postavlja slušatelj događaja na kontejner testnog cilja. Prilikom pritiska tipke miša na kontejner poziva se funkcija koja će učitati testne slučajeve sadržane u tom testnom cilju. Ta funkcija prvo provjerava prikazuju li se testni slučajevi. Ukoliko se prikazuju uklanja ih, a ukoliko nisu prikazani, prikazat će ih. Svi testni slučajevi smještaju se u već kreirani kontejner za testne slučajeve koji se nalazi unutar vanjskog kontejnera testnog cilja. Za svaki testni slučaj kreira se poseban kontejner koji sadrži podatke o njemu. Taj kontejner je također *div* HTML element i sadrži podatke kao što su status, identifikacijski broj i sl., a umetnut je u kontejner za testne slučajeve. Na svaki kontejner za testni slučaj potrebno je dodati slušatelja događaja koji će slušati za događaj pritiska tipke miša. Pritiskom tipke miša na kontejner testnog slučaja poziva se funkcija koja dohvaća detalje o testnom slučaju.

Još jedna od funkcionalnosti koju je potrebno implementirati odabir je više testnih slučajeva koristeći pritisak tipke miša dok su stisnute tipke CTRL ili SHIFT. Svaki testni slučaj na koji se pritisne tipkom miša dok je pritisnuta tipka CTRL bit će označen. Pritiskom tipke miša dok je pritisnuta tipka SHIFT na testni slučaj označit će se svi testni slučajevi od zadnje odabranog do onog koji je odabran pritiskom tipke miša dok je pritisnuta tipka SHIFT. Te funkcionalnosti mogu se postići na način da se definiraju tri globalne varijable, jedna koja će označavati zadnji označeni test, druga koja će sadržavati polje testova koji su odabrani pritiskom tipke miša dok je pritisnuta tipka SHIFT i treća koja će sadržavati polje odabranih testova. U funkciji definiranoj u slušatelju događaja potrebno je napraviti provjere je li se pritisak tipke miša dogodio dok su bile pritisnute tipke CTRL ili SHIFT.

Ukoliko je tijekom pritiska tipke miša bila pritisnuta tipka CTRL, potrebno je provjeriti je li test već označen i nalazi li se u polju svih označenih testova. Ukoliko jest, potrebno ga je odznačiti uklanjanjem klase *highligt* iz njegovog kontejnera. Klasa je *highlight* klasa koja se dodaje označenim testnim slučajevima te mijenja boju njihovih kontejnera kako bi se ukazalo na to da je test odabran. Test je također potrebno ukloniti iz polja svih odabranih testova te varijablu koja sadržava zadnji označeni test postaviti na vrijednost *null*. Ukoliko test nije već odabran potrebno ga je dodati u polje svih odabranih testova, dodijeliti mu klasu *highlight* te taj kontejner spremiti u varijablu koja sadržava zadnji označeni test.

Ako je na kontejner testnog slučaja pritisnuto tipkom miša dok je bila pritisnuta tipka SHIFT potrebno je testu na koji je pritisnuto tipkom miša dodijeliti klasu *highlight*. Ako je vrijednost varijable koja sadržava zadnji odabrani test različita od *null* potrebno je proći kroz sve kontejnere

testnih slučajeva koji se nalaze između prethodno odabranog kontejnera i onog trenutno odabranog te one koji se ne nalaze u polju testnih slučajeva odabranih sa SHIFT-om dodati u to polje i dodijeliti im klasu *highlight*. Ako se bez otpuštanja SHIFT tipke pritiskom tipke miša pritisne na neki drugi kontejner testnog slučaja potrebno je odznačiti sve testove iz polja koje sadržava sve testove označene sa SHIFT-om i ponoviti prethodno definirani postupak odabira testova. Ukoliko je vrijednost varijable koja sadržava zadnji označeni testni slučaj jednaka *null* tada se pritisnuti kontejner spremi u tu varijablu. Kada se otpusti tipka SHIFT svi testovi iz polja testnih slučajeva odabranih sa SHIFT-om prebacuju se u polje svih odabranih testova te se polje prazni. Za provjeru otpuštanja tipke SHIFT potrebno je na cijeli HTML dokument staviti slušatelj događaja koji će slušati za *keyup* događaj.

Još je jedna potrebna funkcionalnost odznačavanje svih označenih testova pritiskom tipke miša izvan kontejnera koji sadržava prikaz testnih ciljeva i testnih slučajeva. To je moguće postići stavljajući slušatelj događaja na cijeli HTML dokument koji će na pritisak tipke miša odznačiti sve testove. Budući da je na neke elemente potrebno moći pritisnuti tipkom miša bez da se testovi odznače u njihove funkcije definirane u slušatelju događaja za pritisak tipke miša potrebno je dodati *event.stopPropagation()*. U HTML elementima pritisak tipke miša se „prenosi“ od najnižeg elementa preko elemenata koji ga sadržavaju sve do samog HTML dokumenta. Koristeći *stopPropagation()* to se može izbjeći te će se pritisak tipke miša zadržati samo na tom elementu što znači da neće doći do samog dokumenta i testovi se neće odznačiti.

3.3.4. Upravljanje testovima

Sučelje za upravljanje testovima treba omogućiti odabir projekta iz baze podataka, odabir jednog od testnih planova tog projekta te prikaz testnih ciljeva i testnih slučajeva u obliku stabla. Također, treba omogućiti odabir više testova te njihovog dodjeljivanja testeru. Neke od funkcija koje su potrebne u skripti za upravljanje testovima nalaze se u tablici 3.4.

Tablica 3.4. *Funkcije potrebne u skripti za upravljanje testovima.*

Funkcija	Opis
Funkcija za kreiranje *WebSocket* objekta	Funkcija kreira *WebSocket* objekt s URL-om definiranim u poslužiteljskom dijelu aplikacije u koji se doda i identifikacijski broj korisnika.
Funkcija za dohvaćanje projekata	Funkcija radi zahtjev na odgovarajuću krajnju točku API-a baze podataka kako bi dobila sve projekte dostupne u bazi. Ukoliko su podatci uspješno dohvaćeni poziva funkciju za učitavanje projekata.
Funkcija za učitavanje projekata	Funkcija predani popis projekata učitava u odgovarajući HTML *select* element.

Funkcija za dohvaćanje testnih planova	Funkcija radi zahtjev na odgovarajuću krajnju točku API-a baze podataka pomoću AJAX-a kako bi dobila sve testne planove za izabrani projekt. Ukoliko su podatci uspješno dohvaćeni poziva se funkcija za učitavanje testnih planova.
Funkcija za učitavanje testnih planova	Funkcija testne planove iz predanog popisa učitava u odgovarajući HTML *select* element.
Funkcija koja dohvaća podatke o testnom planu	Funkcija radi zahtjev na odgovarajuću krajnju točku API-a baze pomoću AJAX-a kako bi dohvatila detalje o testnom planu uključujući popis njegovih testnih ciljeva i testnih slučajeva. Ukoliko su podatci uspješno dohvaćeni poziva funkciju za učitavanje testnih ciljeva i funkciju za učitavanje detalja o testnom planu.
Funkcija za učitavanje testnih ciljeva	Funkcija kreira sve potrebne kontejnere za svaki testni cilj i dodaje ih u predviđeni kontejner u HTML-u.
Funkcija za učitavanje testnih slučajeva	Funkcija kreira sve potrebne kontejnere za svaki testni slučaj i dodaje ih u predviđeni kontejner u HTML-u.
Funkcija za učitavanje detalja o testnom planu	Funkcija predane detalje o testnom planu učitava u za to predviđene HTML elemente.
Funkcija za osvježavanje prikaza dodijeljenih testova	Funkcija treba na temelju podataka o tome kojem je testeru dodijeljen ili uklonjen pojedini testni slučaj napraviti osvježavanje podataka o dodijeljenom testeru za taj testni slučaj.

Na sučelju za upravljanje testovima potrebno je omogućiti prikaz testnih ciljeva i slučajeva nekog testnog plana u obliku strukture stabla i odabir više testnih slučajeva kao što je objašnjeno u potpoglavlju 3.3.3. Istovremeni odabir više testova vrši se na isti način kao u prethodnom potpoglavlju. Dodatna funkcionalnost koju ovo sučelje omogućuje prilikom odabira više testnih slučajeva je da se mogu odabirati cijeli testni ciljevi i na taj način se odabiru i svi testni slučajevi koje oni sadržavaju.

Budući da se pritiskom tipke miša na testni cilj prikazuju i uklanjaju svi njegovi testni slučajevi za funkcionalnost odabira uzeto je da također mora biti pritisnuta tipka CTRL. Ako se na kontejner testnog slučaja pritisne tipkom miša dok je pritisnuta tipka CTRL provjeri se je li testni cilj već odabran tako da se provjeri sadržava li njegov kontejner klasu *highlight*. Ukoliko ne sadrži klasu ona mu se doda te se testni slučajevi iz tog testnog cilja označe na način opisan u prethodnom potpoglavlju. Ako kontejner sadrži klasu *highlight* ona mu se uklanja te se testni slučajevi u tom testnom cilju odznačavaju na način opisan u prethodnom potpoglavlju.

Kako je na ovom sučelju potrebno moći dodijeliti testne slučajeve testeru jedna od potrebnih funkcionalnosti je osvježavanje prikaza testnih slučajeva na način da se izmjeni podatak o

dodijeljenom testeru. Za dobivanje informacija o dodjeljivanju i uklanjanju testova testeru od poslužiteljskog dijela aplikacije koristi se *WebSocket*. Kada se putem *WebSocket*-a primi poruka da je jedan testni slučaj ili više njih uklonjen ili dodan testeru poziva se funkcija koja radi osvježavanje prikaza. Ova funkcija prođe kroz predane podatke koji sadrže identifikacijski broj testa, ime i prezime testera ukoliko je testni slučaj dodijeljen testeru te status koji označava je li test dodan ili uklonjen te osvježi podatke u kontejneru koji sadrži taj testni slučaj. Budući da je među detaljima testnog plana potrebno prikazati i broj dodijeljenih i nedodijeljenih testnih slučajeva u funkciji za osvježavanje prikaza potrebno je pozvati funkciju koja učitava detalje o testnom planu kako bi se ti brojevi ažurirali.

3.4. Poslužiteljski dio aplikacije

U poslužiteljskom dijelu aplikacije potrebno je omogućiti dohvaćanje i slanje podataka u alat za praćenje razvoja programske podrške. Potrebno je kreirati potrebne funkcije za rad u *views* skripti te kreirati URL putanje pomoću kojih će se moći pozivati one funkcije koje su potrebne. Također, potrebno je napraviti posebnu klasu koja će vršiti komunikaciju s alatom za praćenje razvoja programske podrške kao i klasu koja će naslijediti *WebSocket* korisničku klasu i implementirati funkcije za komunikaciju preko *WebSocket*-a.

3.4.1. *Views* skripta

U *views* skripti potrebno je definirati funkcije preko kojih će se moći pozivati metode iz klase za rad s alatom za praćenje razvoja programske podrške. Ove funkcije moguće je pozvati preko URL putanja definiranih u *urls.py* skripti u *Djang*-u. U tablici 3.5. nabrojane su neke funkcije koje treba sadržavati *views* skripta.

Tablica 3.5. *Funkcije potrebne u views skripti.*

Funkcija	Opis
Funkcija za dohvaćanje projekata	Funkcija poziva odgovarajuću funkciju za dohvaćanje popisa projekata iz alata za praćenje razvoja programske podrške i vraća listu projekta u obliku HTTP odgovora.
Funkcija za slanje novo kreiranog testnog plana u bazu podataka	Funkcija primi podatke o novom testnom planu te pozove odgovarajuću funkciju koja će dohvatiti ostatak podataka iz alata za praćenje razvoja programske podrške te zatim pozvati funkciju za slanje testnog plana u bazu podataka.
Funkcija za dohvaćanje liste testnih planova	Funkcija treba pozvati odgovarajuću funkciju iz klase za rad s alatom za praćenje razvoja programske podrške kako bi dobila listu testnih planova koji pripadaju određenom projektu te ju vratiti u obliku HTTP odgovora

Funkcija za dohvaćanje detalja o testnom slučaju	Funkcija treba pozvati odgovarajuću metodu iz klase za rad s alatom za praćenje razvoja programske podrške kako bi dobila detalje o testnom slučaju koje će se vratiti u obliku HTTP odgovora.

3.4.2. Klasa za rad s alatom za praćenje razvoja programske podrške

Klasa za rad s alatom za praćenje razvoja programske podrške ima funkciju komunikacije s tim alatom putem njegovog API-a. Klasa sadrži metode za dodavanje i dohvaćanje podataka iz alata za praćenje razvoja programske podrške. Kao atribute sadrži putanju do API-a alata kao i putanju na API baze podataka. API zahtjevi rade se pomoću *Requests Python* biblioteke. Na slici 3.6. dan je primjer pravljenja zahtjeva pomoću *Requests* biblioteke.

```
1:      apiPath="https://host.com/api"
2:      url=apiPath+"/details"
3:      response=requests.get(url)
```

Slika 3.6. *Primjer koda pravljenja zahtjeva na API pomoću Requests biblioteke.*

Na slici 3.6. vidljivo je da je potrebno znati URL putanju API-a. Ovdje je putanja spremljena u *apiPath* varijablu. Prije pravljenja zahtjeva na putanju je potrebno dodati željenu krajnju točku. U ovom slučaju to je napravljeno te je cijela putanja spremljena u *url* varijablu. Na kraju je potrebno napraviti zahtjev pomoću *requests* modula tako da mu se preda željena URL putanja. U ovom slučaju napravljen je zahtjev tipa *get*.

U tablici 3.6. nabrojane su neke od metoda potrebnih u klasi za rad s alatom za praćenje razvoja programske podrške.

Tablica 3.6. *Metode potrebne u klasi za rad s alatom za praćenje razvoja programske podrške.*

Funkcija	Opis
Funkcija za dohvaćanje projekata	Funkcija radi zahtjev prema API-u alata za praćenje razvoja programske podrške kako bi dobila listu dostupnih projekata.
Funkcija za dohvaćanje liste testnih planova	Funkcija radi zahtjev prema API-u alata za praćenje razvoja programske podrške kako bi dobila listu testnih planova za određeni projekt.
Funkcija za dohvaćanje detalja o testnom slučaju	Funkcija radi zahtjev prema API-u alata za praćenje razvoja programske podrške kako bi dobila potrebne detalje o testnom slučaju. Neke detalje potrebno je dodatno parsirati kako bi se izvukle odgovarajuće informacije.

Funkcija za dohvaćanje kompletnog testnog plana	Funkcija dohvaća testni plan te sve njegove pripadajuće testne ciljeve, testne slučajeve i testne sesije.
Funkcija za slanje testnog plana u bazu podataka	Funkcija radi zahtjev prema API-u baze podataka u kojem šalje podatke o testnom planu koji se zatim spremaju u bazi.
Funkcija za spremanje rezultata testnog slučaja u alat za praćenje razvoja programske podrške	Funkcija radi zahtjev prema API-u alata za praćenje razvoja programske podrške u kojem šalje sve podatke koji su potrebni kako bi se spremio rezultat testnog slučaja.

3.4.3. Klasa za rad s *WebSocket*-ima

Klasa za rad s *WebSocket*-ima treba naslijediti korisničku klasu iz *Django Channels* biblioteke. U njoj su sadržane funkcije za primanje i slanje poruka putem *WebSocket*-a. Neke funkcije su već definirane u *Django Channels* biblioteci, a neke je potrebno samostalno implementirati te ih pozvati negdje u kodu. U tablici 3.7. nabrojane su metode koje klasa za rad s *WebSocket*-ima treba sadržavati.

Tablica 3.7. *Metode potrebne u klasi za rad s Websocket-ima.*

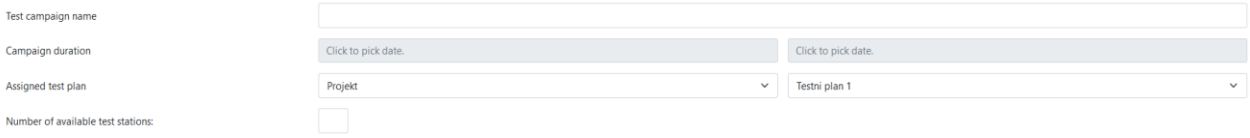
Metoda	Opis
Metoda za otvaranje konekcije	U ovoj metodi stvara se *WebSocket* konekcija. Korisnika se doda u sve potrebne grupe kako bi mogao primati sve poruke koje trebaju biti namijenjene njemu. Na kraju se šalje poruka s potvrdom o prihvaćanju konekcije.
Metoda za zatvaranje konekcije	Korisnika se ukloni iz svih grupa u kojima se nalazi.
Metoda za primanje poruka	Metoda obrađuje primljene poruke te ovisno o njihovom sadržaju poziva druge metode.
Metoda za slanje poruke o osvježavanju podataka	Metoda šalje poruku koja označava da je potrebno osvježiti podatke kao i potrebne podatke.

4. DEMONSTRACIJA FUNKCIONALNOSTI

U ovom poglavlju bit će demonstrirane osnovne mogućnosti koje aplikacija treba imati. Bit će prikazani primjeri korisničkog sučelja za izradu novog testnog plana, uređivanje postojećeg testnog plana, izvršavanje testova i upravljanje testovima.

4.1. Izrada novog testnog plana

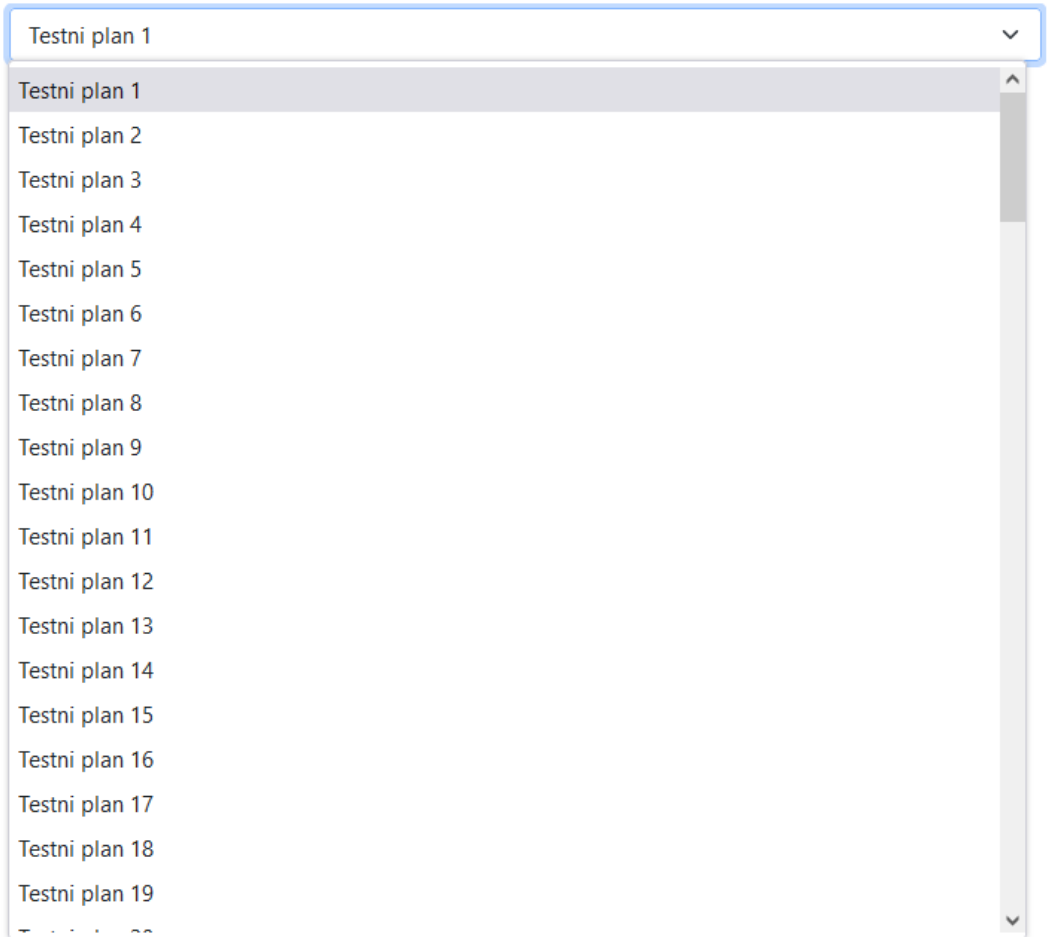
Prilikom otvaranja sučelja za izradu novog testnog plana prikažu se polja kao što je prikazano na slici 4.1.



Slika 4.1. *Polja na sučelju za kreiranje testnog plana.*

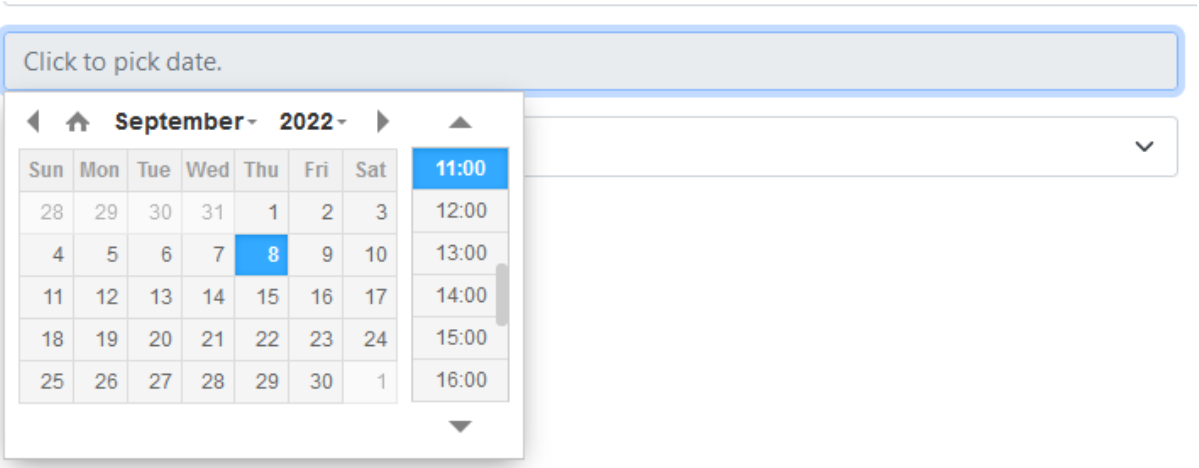
Na sučelju se nalaze - polje za unos naziva testnog plana, polja za odabir datuma završetka i početka, *select* polje za odabir projekta, *select* polje za odabir testnog plana i polje za unos teksta gdje se unosi broj testnih stanica. Polje za unos broja testnih stanica ograničeno je tako da se mogu unositi samo znamenke.

Za izradu novog testnog plana potrebno je prvo odabrati željeni projekt iz odgovarajućeg *select* polja prilikom čega se u drugom *select* polju učita lista dostupnih testnih planova za taj projekt. Na slici 4.2. prikazano je *select* polje za odabir testnog plana.



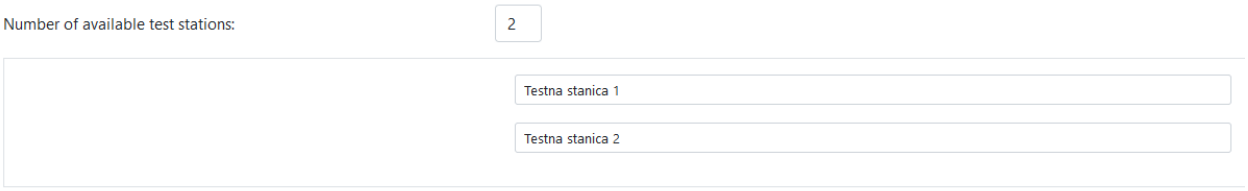
Slika 4.2. *Select polje za odabir testnog plana.*

Nakon što se odabere željeni testni plan njegov naziv automatski se učita u polje za definiranje naziva testnog plana, no moguće ga je naknadno izmijeniti. Za testni plan potrebno je definirati i datum početka i završetka. Na slici 4.3 prikazano je polje za unos datuma.



Slika 4.3. *Polje za unos datuma u sučelju za izradu novog testnog plana.*

Nakon unosa planiranog datuma početka i završetka provedbe testnog plana potrebno je unijeti broj testnih stanica na kojima će se izvoditi testni slučajevi koji mu pripadaju. Kreiranim testnim stanicama moguće je definirati naziv. Na slici 4.4. prikazano je polje za unos broja testnih stanica kao i polja za uređivanje njihovih imena.

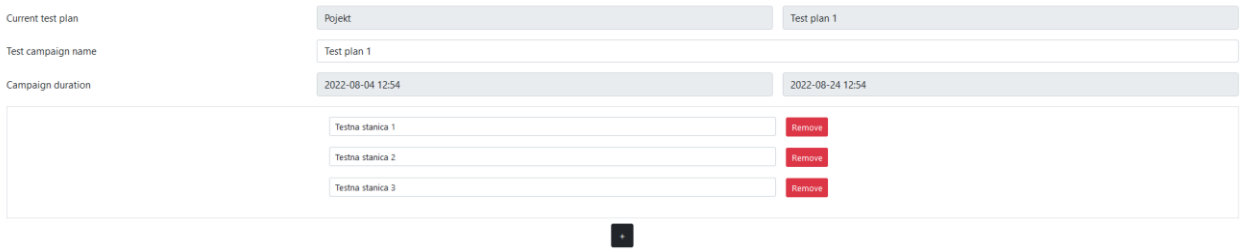


Slika 4.4. *Polja za unos broja i naziva testnih stanica na sučelju za izradu novog testnog plana.*

Nakon što su svi podatci uneseni testni se plan izrađuje pritiskom na gumb *Create*. Ukoliko su sva polja ispravno popunjena testni će plan se spremiti u bazu podataka, a ukoliko ne, prikazat će se odgovarajuća poruka.

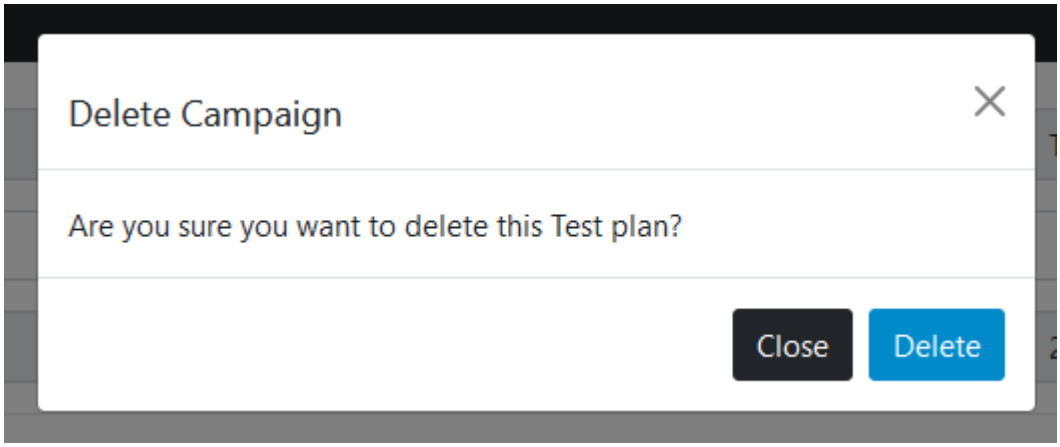
4.2. Uređivanje postojećeg testnog plana

Sučelje za uređivanje postojećeg testnog plana treba omogućiti izmjenu naziva testnog plana kao i njegov planirani datum početka i završetka te uklanjanje i dodavanje testnih stanica kao i izmjenu njihovog naziva. Također, treba omogućiti brisanje testnog plana. Slika 4.5. prikazuje polja na sučelju za izmjenu postojećeg testnog plana.



Slika 4.5. *Polja na sučelju za izmjenu postojećeg testnog plana.*

Pored svake testne stanice nalazi se gumb za njezino uklanjanje koji briše tu testnu stanicu. Također, moguće je dodati novu testnu stanicu pritiskom na gumb „+“. Kada su svi željeni podatci izmijenjeni promjene se mogu spremiti pritiskom na gumb *Edit*. Ukoliko su sva polja popunjena i svi podatci su validni promjene će se spremiti, ukoliko nisu prikazat će se odgovarajuća poruka. Pritiskom na gumb *Delete* pojavljuje se skočni prozor kao na slici 4.6. na kojem je potrebno potvrditi brisanje testnog plana.



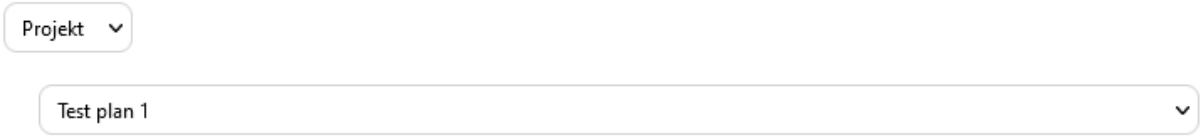
Slika 4.6. *Skočni prozor za potvrđivanje brisanja testnog plana.*

Nakon što se na skočnom prozoru za brisanje testnog plana pritisne gumb *Delete* testni će plan biti obrisan. Pritiskom na gumb *Cancel* radnja će biti obustavljena.

4.3. Izvođenje testova

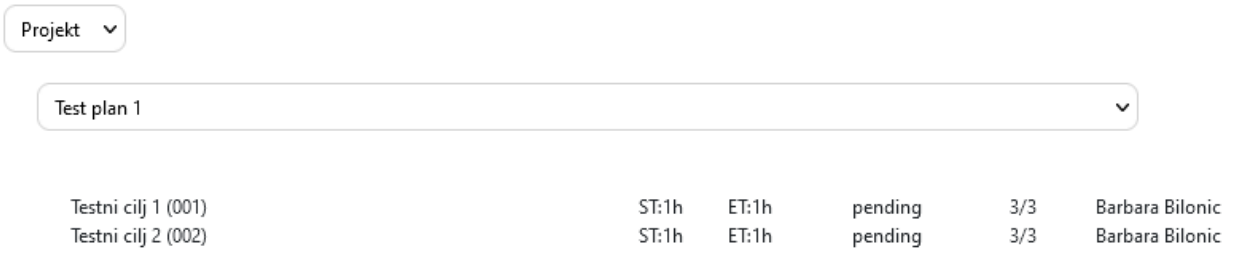
Sučelje za izvođenje testova omogućuje odabir željenog projekta i testnog plana te prikaz njegovih testnih ciljeva i slučajeva u obliku strukture stabla. Na njemu je moguće označavanje više testova te unos rezultata za više ili samo jedan testni slučaj.

Prvo je potrebno odabrati željeni projekt i testni plan iz odgovarajućih *select* polja prikazanih na slici 4.7.



Slika 4.7. *Select polja za odabir projekta i testnog plana na sučelju za izvođenje testova.*

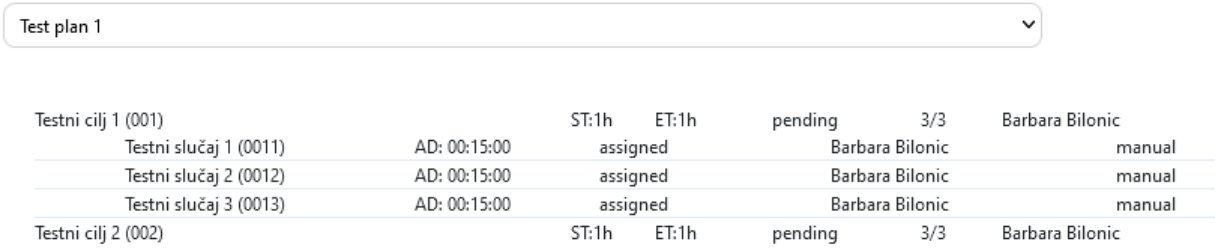
Nakon što su odabrani projekt i testni plan prikazuje se popis testnih ciljeva kao što je prikazano na slici 4.8.



Slika 4.8. *Prikaz popisa testnih ciljeva na sučelju za izvođenje testova.*

Za svaki testni cilj prikazuje se njegov identifikacijski broj, vrijeme trajanja, status, broj testova koji su izvršeni i ime i prezime dodijeljenih testera.

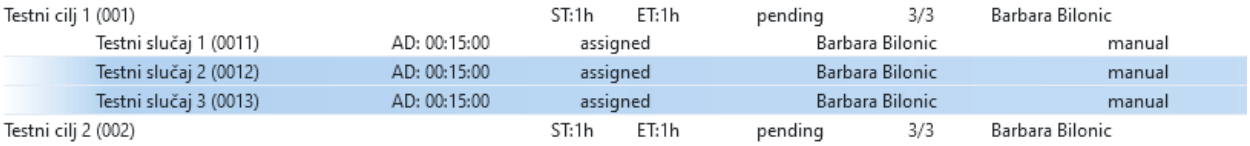
Pritiskom miša na testni cilj prikazuje se popis testnih slučajeva koje testni cilj sadržava kao što je prikazano na slici 4.9.



Slika 4.9. Prikaz popisa testnih slučajeva testnog cilja na sučelju za izvođenje testova.

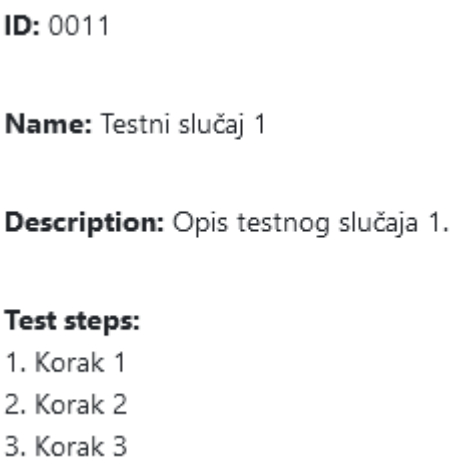
Za svaki test prikazuje se identifikacijski broj, vrijeme izvođenja, status, ime dodijeljenog testera, i tip izvođenja.

Kada se testni slučaj označi pritiskom tipke miša dok su pritisnute tipke SHIFT ili CTRL boja njegovog kontejnera se promjeni kako bi se pokazalo da je test odabran. Prikaz odabranih testnih slučajeva vidljiv je na slici 4.10.



Slika 4.10. *Prikaz odabranih testnih slučajeva na sučelju za izvođenje testova.*

Dvostrukim pritiskom tipke miša na neki testni slučaj otvaraju se detalji o njemu kako je prikazano na slici 4.11.



Slika 4.11. *Prikaz detalja testnog slučaja na sučelju za izvođenje testova.*

Nakon što se otvore pripadajući detalji o testnom slučaju, moguće je unijeti rezultate provedenog testa. Postupak unosa rezultata svodi se na odabir željenog ishoda testa i pritiska na tipku Save (Slika 4.12).

Slika 4.12. *Odabir rezultata testnog slučaja na sučelju za izvođenje testova.*

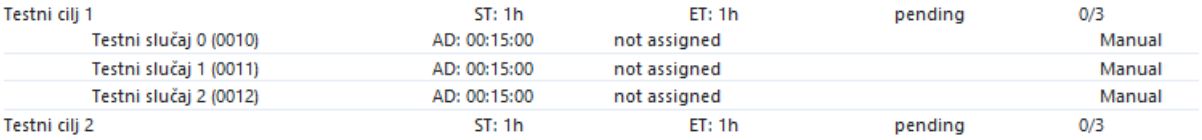
4.4. Upravljanje testovima

Na sučelju za upravljanje testovima omogućen je odabir željenog projekta i testnog plana. Za odabrani testni plan prikazuju se testni ciljevi i slučajevi u obliku prikaza stabla. Omogućen je i odabir jednog ili više testova. Odabirom testnog plana prikazuju se njegovi detalji, odnosno, prikazuju se testni ciljevi. Slika 4.13. prikazuje prikaz testnih ciljeva odabranog testnog plana.

Slika 4.13. *Prikaz testnih ciljeva za odabrani testni plan na sučelju za upravljanje testovima.*

Za svaki testni cilj prikazuju se predviđena vremena izvršavanja, status, ime i prezime dodijeljenih testera ukoliko su neki testni slučajevi iz tog testnog cilja dodijeljeni i broj izvršenih testova od ukupnog broja testnih slučajeva.

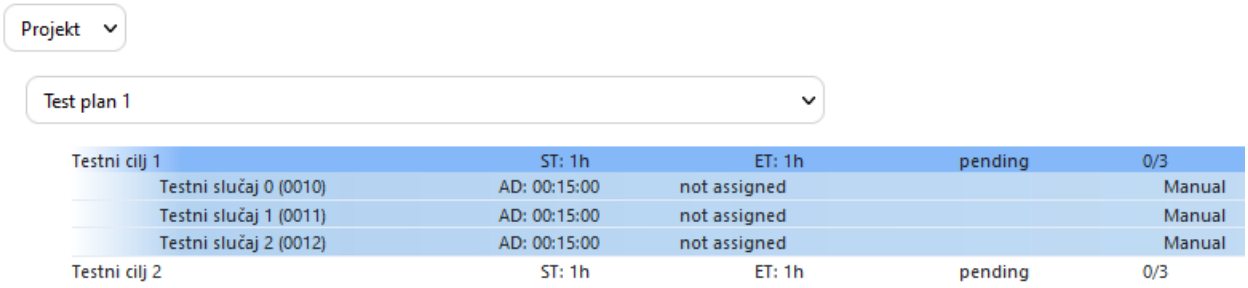
Pritiskom tipke miša na testni cilj otvara se popis njegovih testnih slučajeva kao što je prikazano na slici 4.14.



Slika 4.14. *Prikaz popisa testnih slučajeva testnog cilja na sučelju za upravljanje testovima.*

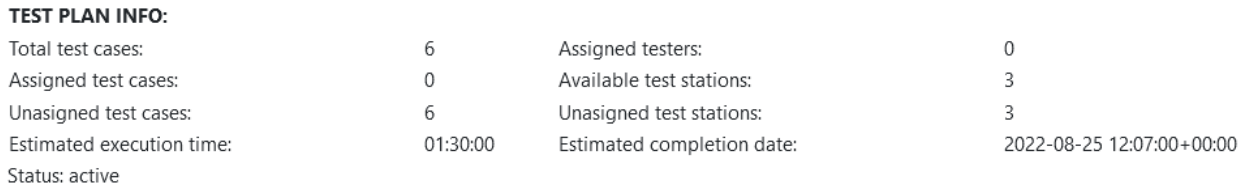
Za svaki testni slučaj prikazuje se njegov naziv, identifikacijski broj, predviđeno vrijeme izvođenja, status, ime dodijeljenog testera ukoliko je dodijeljen i način izvođenja.

Pritiskom tipke miša na testni slučaj dok je pritisnuta tipka CTRL ili SHIFT označava se jedan ili više testnih slučajeva. Pritiskom tipke miša na testni cilj dok je pritisnuta tipka CTRL označavaju se svi testni slučajevi sadržani u tom testnom cilju kao što je prikazano na slici 4.15.



Slika 4.15. *Označavanje svih testnih slučajeva testnog cilja na sučelju za upravljanje testovima.*

Na ovom sučelju također se prikazuju detalji odabranog testnog plana kao što je prikazano na slici 4.16.



Slika 4.16. *Prikaz detalja odabranog testnog plana na sučelju za upravljanje testovima.*

Za odabrani testni plan prikazuje se ukupan broj testnih slučajeva, broj testnih slučajeva koji su dodijeljeni testerima, broj nedodijeljenih testnih slučajeva, procijenjeno vrijeme potrebno za izvršavanje, broj testera kojima su dodijeljeni testni slučajevi, broj testnih stanica, broj slobodnih testnih stanica, procijenjeni datum završetka i status.

5. ZAKLJUČAK

U okviru rada opisana je i izrađena web aplikacija koja omogućava provedbu i administriranje testova. Web aplikacija ima mogućnost integracije s bazom podataka i komercijalnim alatom za praćenje programske podrške te se sastoji od klijentskog i poslužiteljskog dijela.

Poslužiteljski dio aplikacije vrši komunikaciju s API-em alata za praćenje razvoja programske podrške i API-em baze podataka putem HTTP zahtjeva dok se komunikacija s klijentskim dijelom aplikacije odvija preko URL putanja definiranih u poslužiteljskom dijelu i *WebSocket* protokola. Klijentski dio aplikacije vrši komunikaciju s poslužiteljskim dijelom aplikacije i API-em baze podataka putem HTTP zahtjeva dok s poslužiteljskim dijelom aplikacije još dodatno komunicira putem *WebSocket* protokola.

Alat za praćenje razvoja programske podrške treba omogućiti razvrstavanje testnih slučajeva u testne ciljeve, testnih ciljeva u testne planove i testnih planova u projekte. Također, mora imati mogućnost spremanja rezultata testnih slučajeva. Svim navedenim funkcionalnostima moguće je pristupiti preko API-a.

Baza podataka omogućuje spremanje testnih planova, ciljeva, slučajeva i sesija kao i potrebnih podataka o njima. Također, ima funkcionalnost dohvaćanja i izmjenjivanja spremljenih podataka. Tim je funkcionalnostima moguće pristupiti preko API-a baze podataka.

Poslužiteljski dio aplikacije izrađen je u *Django* razvojnom okviru koristeći *Python* programski jezik. On putem definiranih URL putanja pruža mogućnost pozivanja funkcija koje dohvaćaju i upisuju podatke u alat za praćenje razvoja programske podrške i bazu podataka.

Klijentski dio aplikacije izrađen je koristeći *JavaScript* programski jezik i *jQuerry JavaScript* biblioteku. Omogućuje prikaz četiri web sučelja, sučelje za izradu novog testnog plana, sučelje za uređivanje postojećeg testnog plana, sučelje za izvođenje testova i sučelje za upravljanje testovima.

Za izradu ove aplikacije mogle su biti odabrane neke druge tehnologije i razvojni okviri za izradu web aplikacija. Prednost je *Django* razvojnog okvira mogućnost korištenja *Python* programskog jezika koji nudi laku obradu podataka. Korištenjem opširnije baze podataka moguće je izbaciti korištenje alata za praćenje razvoja programske podrške i na taj način pohranjivati sve podatke na jednom mjestu.

LITERATURA

[1] Polarion ALM,Siemens, dostupno na: https://polarion.plm.automation.siemens.com/products/polarion-alm [17. 8. 2022.]

[2] Azure Devops, Microsoft, dostupno na: https://docs.microsoft.com/en-us/azure/devops/get-started/?view=azure-devops [17. 8. 2022.].

[3] SpiraTeam, Inflectra,dostupno na: https://www.inflectra.com/SpiraTeam [17. 8. 2022.]

[4] Visure Solutions,Visure, dostupno na: https://visuresolutions.com [18. 8. 2022.]

[5] Windchill RV&S 12.3.0.0 Help Center, PTC, dostupno na: https://support.ptc.com/help/windchillrvs/r12.3.0.0/en/ [18. 8. 2022.]

[6] R. J., Beginning Application Lifecycle Management,Apress, New York, 2014.

[7] Django, Django Software Fundation, dostupno na: https://www.djangoproject.com/ [20. 8. 2022.]

[8] MDN Web Docks: JavaScript, Mozilla, dostupno na: https://developer.mozilla.org/en-US/docs/Web/JavaScript [25. 8. 2022.]

[9] jQuerry,OpenJS Foundation, 2022., dostupno na: https://jquery.com/ [25. 8. 2022.]

SAŽETAK

Testiranje je neizostavan dio razvoja programske podrške. Kako bi se testni slučajevi bolje organizirali moguće ih je podijeliti u projekte, testne planove i ciljeve. Za lakše praćenje testiranja mogu se koristiti alati za praćenje razvoja programske podrške. U ovom radu predstavljen je prijedlog arhitekture i dizajna aplikacije koja omogućava administriranje provedbe testova kao i integraciju s komercijalnim alatom za praćenje razvoja programske podrške. Izrađena je i opisana Web aplikacija koja se sastoji od klijentskog i poslužiteljskog dijela, a vrši komunikaciju s bazom podataka te komercijalnim alatom za praćenje razvoja programske podrške.

Ključne riječi: web aplikacija, alat za praćenje razvoja programske podrške, administriranje provedbe testova, *Django* razvojni okvir, *JavaScript*

ABSTRACT

An application for software testing

Testing is an indispensable part of software development. In order to better organize test cases, it is possible to divide them into projects, test plans, and test objectives. For easier management of the testing process, it is possible to use application lifecycle management tools. This paper presents a proposal of the architecture and design of a web application that enables the administration of tests as well as the possibility to integrate the application lifecycle management tool. Furthermore, a web-application consisting of a client and a server part which carries out communication with database and commercial application lifecycle management tool has been constructed and subsequently presented in this paper.

Key words: web application, application lifecycle management tool, test execution administration, *Django* framework, *JavaScript*

ŽIVOTOPIS

Barbara Bilonić rođena je 10. kolovoza 1998. u Osijeku. Godine 2013. završila je osnovnu školu „OŠ Tin Ujević“ u Osijeku. Godine 2017. završila je III. gimnaziju Osijek i iste godine upisala Fakultet elektrotehnike, računarstva i informacijskih tehnologija na kojem je 2020. završila preddiplomski sveučilišni studij računarstva. Nakon toga upisala je diplomski sveučilišni studij računarstva na Fakultetu Elektrotehnike, računarstva i informacijskih tehnologija u Osijeku na kojem trenutno pohađa drugu godinu.

_____

Potpis autora