# Upravljanje korisničkim upitima u ispitivanju programske podrške u automobilskoj industriji

**Ferić, Franjo**





**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU**
**FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA**

**Sveučilišni studij**

# UPRAVLJANJE KORISNIČKIM UPITIMA U ISPITIVANJU PROGRAMSKE PODRŠKE U AUTOMOBILSKOJ INDUSTRIJI

**Diplomski rad**

**Franjo Ferić**

**Osijek, 2022.**

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

**Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit**

**Osijek, 19.09.2022.**

**Odboru za završne i diplomske ispite**

# Imenovanje Povjerenstva za diplomski ispit

| | |
|---|---|
| **Ime i prezime Pristupnika:** | Franjo Ferić |
| **Studij, smjer:** | Diplomski sveučilišni studij Računarstvo |
| **Mat. br. Pristupnika, godina upisa:** | D-1117R, 13.10.2020. |
| **OIB studenta:** | 14447397753 |
| **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:** | Doc. dr. sc. Denis Vranješ |
| **Naslov diplomskog rada:** | Upravljanje korisničkim upitima u ispitivanju programske podrške u automobilskoj industriji |
| **Znanstvena grana diplomskog rada:** | **Informacijski sustavi (zn. polje računarstvo)** |
| **Zadatak diplomskog rada:** | Kod korištenja programske podrške općenito, a posebno kod one koja se koristi u mrežnom okruženju i predviđena je za upotrebu od strane većeg broja različitih korisnika, a iz razloga povećanja učinkovitosti rada, poželjno je korištenje nekakve vrste ticketing sustava. Takvim načinom rada dokumentirani su svi upiti, odnosno situacije koje su se pojavile tijekom upotrebe sustava, a ujedno je omogućen relativno brz, pregledan i jednostavan kanal komunikacije za sve korisnike sustava. U ovome radu zadatak je integrirati ticketing sustav Jira sa sustavom za provođenje udaljenog ispitivanja programske podrške u automobilskoj industriji. Cilj rada je identificirati neophodan dio Jira primjenskog programskog sučelja (engl. application programming |
| **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<br>Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda<br>Jasnoća pismenog izražavanja: 2 bod/boda<br>Razina samostalnosti: 3 razina |
| **Datum prijedloga ocjene od strane mentora:** | 19.09.2022. |

| | |
|---|---|
| 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, 27.09.2022.**

| | |
|---|---|
| **Ime i prezime studenta:** | Franjo Ferić |
| **Studij**: | Diplomski sveučilišni studij Računarstvo |
| **Mat. br. studenta, godina upisa:** | D-1117R, 13.10.2020. |
| **Turnitin podudaranje [%]:** | 1 |

Ovom izjavom izjavljujem da je rad pod nazivom**: Upravljanje korisničkim upitima u ispitivanju programske podrške u automobilskoj industriji**

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

U današnjem svijetu eksponencijalan napredak tehnologije dovodi do eksponencijalnog porasta količine pogreški (engl. *bug*), zastarijevanja koda, korištenih tehnologija i sl. Globalizacija je omogućila svakom kućanstvu pristup gotovo neograničenom broju različitih tehnoloških proizvoda (igrice, streaming usluge, facebook i sl.), ali ljudima koji stvaraju i održavaju te iste proizvode (programeri, testeri, projekt menadžeri, projekt planeri itd.) ona otežava posao. Rastom broja korisnika, raste broj različitih načina korištenja proizvoda. To otežava predviđanje mogućih ponašanja proizvoda prilikom osmišljavanja i stvaranja njegove strukture. Testiranjem proizvoda nemoguće je pokriti sve moguće pristupe korištenja tih proizvoda. Zbog toga će uvijek biti, nakon stavljanja gotovog proizvoda na tržište, prijavljenih problema koje je potrebno ukloniti. Popravci trebaju biti brzo i djelotvorno odrađeni kako bi se proizvod što prije vratio na tržište i zadržao svoju relevantnost. Zato je poželjno kod ispitivanja programske podrške koristiti nekakav ticketing sustav. Korisnost ticketing sustava je njihova mogućnost dokumentiranja svih upita, odnosno svih prijavljenih problema ili događaja koji su se pojavili tijekom korištenja sustava te upravljanje i praćenje rješavanja istih. Ticketing sustavi omogućuju jednostavniji, brži i pregledniji način komunikacije za sve korisnike sustava. Tema ovog diplomskog rada je istraživanje načina upravljanja korisničkim upitima u ispitivanju programske podrške u automobilskoj industriji. U radu je napravljen kratak i informativan osvrt na neke od ticketing sustava koji se trenutno koriste u IT tvrtkama. Predložen je i način korištenja istih u ispitivanju programske podrške u automobilskoj industriji. Zadatak rada je, uz prikaz informacija o ticketing sustavima, osmisliti i implementirati mogući način integracije ticketing sustava Jira sa sustavom za provođenje udaljenog ispitivanja programske podrške u automobilskoj industriji. [1]

# 2. TICKETING SUSTAVI

Za razvoj programske podrške koriste se različite metodologije izrade istih, ovisno o prirodi projekta. Većina projekata, zbog velikog broja tehnologija koje se neprestano unaprijeđuju i izmjenjuju, zahtijeva stalno održavanje i prepravljanje programske podrške te se iz tog razloga koriste *Waterfall, Iterative, Agile* i *Rapid Application Development* metodologije izrade. Prema autorovim spoznajama tijekom istraživanja, u automobilskoj industriji su najkorištenije metodologije *Lean, Agile, Waterfall, V-model* i *Iterative and incremental. S*vim navedenim metodologijama zajedničko je održavanje i testiranje programske podrške. Zbog toga gotovo sve tvrtke u nekom obliku koriste ticketing sustav za praćenje pogrešaka programske podrške.

Ticketing sustavi pomažu timovima razvojnih programera da učinkovito u obliku *ticket*-a prijavljuju pogreške, opišu ih, analiziraju, uređuju tko je zadužen za njihovo rješavanje i prate cjelokupan proces od nastanka do rješenja prijavljenog problema. Pravilnim korištenjem sustava za praćenje pogrešaka povećava se produktivnost, ostvaruje brži protok informacija i brže rješavanje problema programske podrške. Postoji više sustava koji su usredotočeni samo na praćenje pogrešaka ili uz svoju glavnu uslugu nude i praćenje pogrešaka. Odabir sustava koji će se koristiti za praćenje pogrešaka ovisi o tvrtci, njezinim mogućnostima i potrebama. [2]

Pojam *ticket* predstavlja prijavljeni događaj koji je potrebno istražiti ili zadatak koji je potrebno obaviti. *Ticket* treba sadržavati ključne podatke o problemu ili zadatku koji predstavlja, a to mogu biti: hitnost *ticket*-a, tko ga je prijavio, tko je zadužen za njega, općenito naslov koji definira njegovu namjenu, detaljan opis prijavljenog zadatka, problema i slično. Zbog toga što gotovo svi ticketing sustavi posjeduju svoj naziv za pojam *ticket* (u Jiri *issue,* u ClickUp-u *task,* u Axosoft-u *item*, itd.) u nastavku rada će se za pojam *ticket* koristiti naziv tiket. [2]

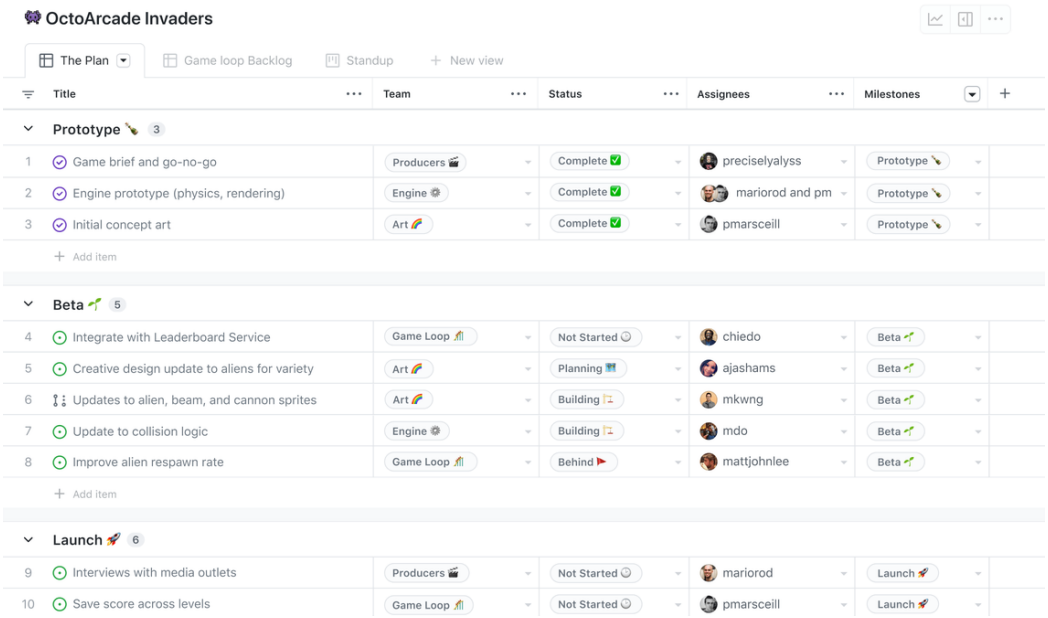
Osnovne funkcionalnosti koje većina ticketing sustava nudi su:

- mogućnost praćenja stanja tiketa od njegova stvaranja do njegova rješavanja
- spremanje svih stvorenih tiketa neovisno o njihovom trenutnom stanju
- mogućnost uređivanja i prilagođavanja forme za stvaranje tiketa
- razvrstavanje tiketa radi bolje preglednosti
- bilježenje tko je podigao tiket, tko je dodijelio i kome je dodijeljen tiket. [3]

Sustavi se međusobno razlikuju u načinu implementacije navedenih i dodatnih ponuda kojima se obično prikazuju kao sustav specijaliziran za određen dio stvaranja projekta koji podržava praćenje pogrešaka. U nastavku su opisani neki od sustava koji podržavaju praćenje tiketa. Za svaki je navedeno što ga razlikuje od ostalih i kako bi mogao biti iskorišten u ispitivanju programske podrške u automobilskoj industriji.

### GitHub Issues

Praćenje izvođenja projekta i rješavanja problema na GitHub-u je, dodatkom novih funkcionalnosti 2022. godine, pojednostavljeno i nešto sličnije ostalim sustavima za praćenje tiketa i projekata. Dodana je funkcionalnost *issue form* gdje su forme svojevrsna nadogradnja do sada dostupnih *issue teplate*-a kojima se određuje izgled odnosno polja za podatke koje će tiket sadržavati. Razlika je u tome što sada postoji .yml tip datoteke koja je zapravo forma s postavkama polja tiketa. Ono što razlikuje GitHub Issues od ostalih načina praćenja je to što se nalazi na istom mjestu kao kod programske podrške te implementacija načina uređivanja tiket forme. Za razliku od ostalih sustava uređivanje tiket forme u GitHub Issues se ne obavlja korištenjem *drag and drop*-a ili odabirom ponuđenih polja s nekakvog izbornika. Uređivanje tiket forme se obavlja korištenjem *YAML* sintakse i omogućava korisniku stvaranje novih polja za podatke o tiketu, primjer *YAML* sintakse nalazi se na slici 2.2. GitHub Issues zajedno s Projects stvara jednostavno okruženje u kojemu programeri mogu pratiti tikete, stvarati, preuzeti na sebe i prijaviti njihovo rješenje usputno, bez dodatnih prijava na posebne sustave za praćenje problema. Na slici 2.1. se nalazi prikaz GitHub tiketa u obliku tablice.




Sl. 2.1. Prikaz tablice tiketa na GitHub Issues korisničkom sučelju. [5]

Iako GitHub Issues i Projects usluge nisu prihvatljiva zamjena sustava specijaliziranih za praćenje projekata i tiketa, one i dalje mogu biti korisne u praćenju korisničkih upita u automobilskoj industriji. Mogu ih koristiti programeri kako bi međusobno lakše i brže komunicirali pri rješavanju prijavljenih tiketa te nakon njihova rješenja samo krajnju izjavu i rješenje objavili na zaseban ticketing sustav koji koristi njihova tvrtka. [4]

```yaml
name: Bug Report
description: File a bug report
title: "[Bug]: "
labels: ["bug", "triage"]
assignees:
  - octocat
body:
  - type: markdown
    attributes:
      value: |
        Thanks for taking the time to fill out this bug report!
  - type: input
    id: contact
    attributes:
      label: Contact Details
      description: How can we get in touch with you if we need more info?
      placeholder: ex. email@example.com
    validations:
      required: false
```

Sl. 2.2. Primjer korištenja *YAML* sintakse za stvaranje *issue form*-e [6]

**Axosoft**

Axosoft je sustav koji se na tržištu pojavio 2002. godine i od tada postao jedan od najkorištenijih sustava za praćenje projekata i tiketa. Tvrtka koja je proizvela Axosoft od 2021. godine preimenovala se u GitKraken prema svom novom vodećem proizvodu, ali Axosoft se i dalje održava i nadograđuje (posljednji update obavljen je 31.5.2022.[7]). Ono što razlikuje Axosoft od ostalih navedenih sustava je to što je to ponajprije sustav za praćenje projekata koji ima dodatnu funkcionalnost praćenja tiketa. Axosoft sa svojim brojnim funkcionalnostima korisniku omogućava mikro upravljanje projektom. To je omogućeno sljedećim funkcionalnostima:

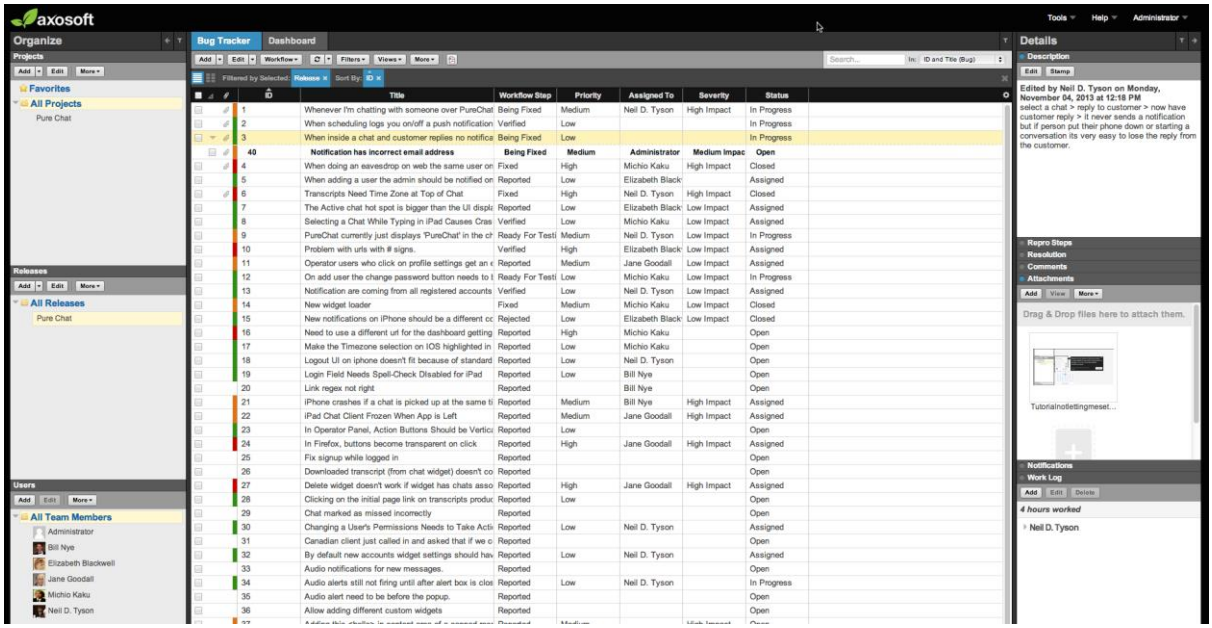
- prikaz sati i postotka slobodnog vremena zaposlenika prilikom dodjele tiketa (ovisno o njegovom radnom vremenu i trenutno dodijeljenim tiketima)

- razrađeno korisničko sučelje s grafičkim prikazom trenutnog stanja projekta (je li trenutnom brzinom rada na projektu moguće zadovoljiti postavljen vremenski cilj, koliki je zaostatak i sl.)
- mogućnost stvaranja *dashboard*-a u kojem korisnik odlučuje koje podatke želi vidjeti i za koje projekte. [8]

Kod stvaranja tiketa formu je moguće urediti dodavanjem dodatnih polja te uklanjanjem postojećih koristeći *drag and drop*. *Drag and drop* način uređivanja funkcionalnosti je sveprisutan u Axosoftu, što je korisno menadžerima jer ne zahtijeva poznavanje posebne sintakse ili kompliciranog načina odabiranja željenih dodataka. Posebno je važna jednostavnost uređivanja funkcionalnosti jer za svaku funkcionalnost Axosofta, uključujući i nenavedene, korisniku je dana mogućnost prilagođavanja njegovim potrebama. To Axosoft čini učinkovitijim i jednostavnijim za korištenje. Axosoft njegove funkcionalnosti čine jednim od najboljih sustava takvog tipa, ali ujedno i jednim od najzahtjevnijih. Njihov raspored, uređenje na web korisničkom sučelju i količina sitnica kojima je moguće upravljati u Axosoftu stvaraju problem pri osposobljavanju korisnika za rad u sustavu. Potrebno je uložiti znatnu količinu vremena da bi korisnik mogao pravilno iskoristiti sve funkcionalnosti koje mu sustav nudi. Izgled web korisničkog sučelja i raspored funkcionalnosti za upravljanje tiketima je unaprijeđen, ali i dalje složen i nedovoljno pregledan. Korisničko sučelje za grafičke prikaze i upravljanje projektima je potpuna suprotnost. Prikaz početnog korisničkog sučelja se nalazi na slici 2.3.

Postoje razni načini filtriranja tiketa i projekata koji, premda nedovoljno, korisniku olakšavaju snalaženje. Korisniku se nudi sloboda i velik broj funkcionalnosti za mikro upravljanje. To sa sobom donosi i povećanu mogućnost nastanaka ljudske greške pri unosu podataka ili upravljanju njima. Svi navedeni nedostaci Axosofta imaju znatno manji utjecaj nakon što je korisnik osposobljen za rad u sustavu i nakon što se funkcionalnosti prilagode tako da se funkcionalnost sustava usredotoči na zadovoljavanje potreba tvrtke ili korisnika koji se njime koriste. Za upravljanje korisničkim upitima u vrijeme ispitivanja programske podrške u automobilskoj industriji, Axosoft sustav bi bio koristan jer posjeduje već implementirane funkcionalnosti poput praćenja zauzetosti zaposlenika, jasan pregled stanja projekta i sl. Mogao bi se koristiti uz već postojeći sustav za provođenje ispitivanja programske podrške ili tako da se korištenjem Axosoft *REST API*-a omogući komunikacija sustava za provođenje ispitivanja programske podrške sa Axosoftom (npr. korisnik može u vlastitom sustavu stvoriti tiket koji će biti podignut na Axosoft).

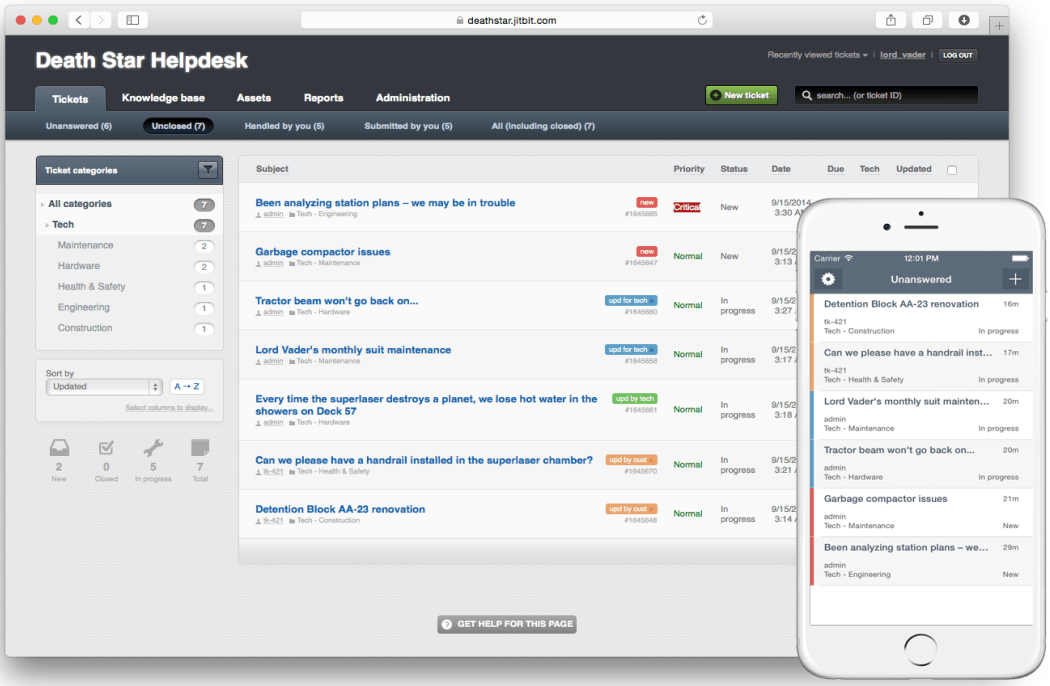
Ako korisnik ili tvrtka traže isključivo ticketing sustav s manjom ili nikakvom potrebom za upravljanje projektima, tada Axosoft ne bi trebao biti prvi izbor jer je složen i ostao bi neiskorišten. [8]



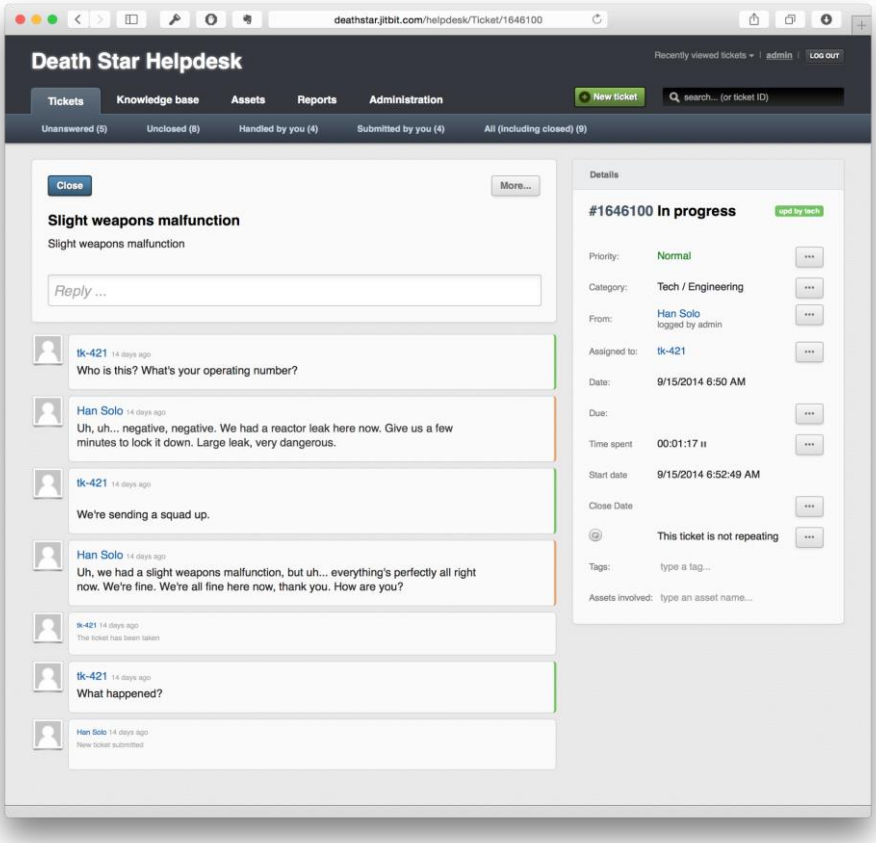
Sl. 2.3. Prikaz početnog korisničkog sučelja Axosoft sustava. [9]

**Jitbit Helpdesk**

JitBit Helpdesk je sustav koji je usredotočen na praćenje i upravljanje tiketima. Tim pristupom je stvoreno jednostavno i pregledno korisničko sučelje prikazano na slici 2.4. koje posjeduje sve funkcionalnosti potrebne za zadovoljavanje potreba praćenja i upravljanja tiketima. Za svaki tiket se pri njegovom otvaranju odbrojava vrijeme provedeno na tiketu i pridodaje ukupnom vremenu provedenom na tiketu. Omogućene su velike količine različitih kategorija tiketa, stvaranje tiketa putem maila, komentari su označeni različitim bojama ovisno o tome jesu li ih napravili korisnici ili članovi tvrtke (slika 2.5.), upravljanje pravima korisnika sustava, postavljanje automatskih događaja u slučaju zadovoljavanja zadanih uvjeta i sl. Usredotočenost sustava na samo upravljanje i praćenje tiketa također olakšava i filtriranje tiketa te njihovo praćenje. Uz mogućnosti vezane isključivo za tikete korisniku je omogućeno prilagođavanje teme sustava, *logo*-a i sl. JitBit nudi *SaaS* i lokalnu poslugu gdje je razlika u sitnicama kao što je način omogućavanja stvaranja tiketa putem maila. JitBit sustav je jednostavan i praktičan alat za upravljanje korisničkim upitima u ispitivanju programske podrške te bi se u automobilskoj industriji mogao koristiti zasebno ili pomoću njegovog *REST API*-a kao priključak već postojećem sustavu ispitivanja programske podrške. U oba slučaja JitBit bi bio koristan dodatak koji bi ubrzao i olakšao upravljanje korisničkim upitima. [10][11][12]




Sl. 2.4. Prikaz glavnog korisničkog sučelja JitBit sustava. [10]

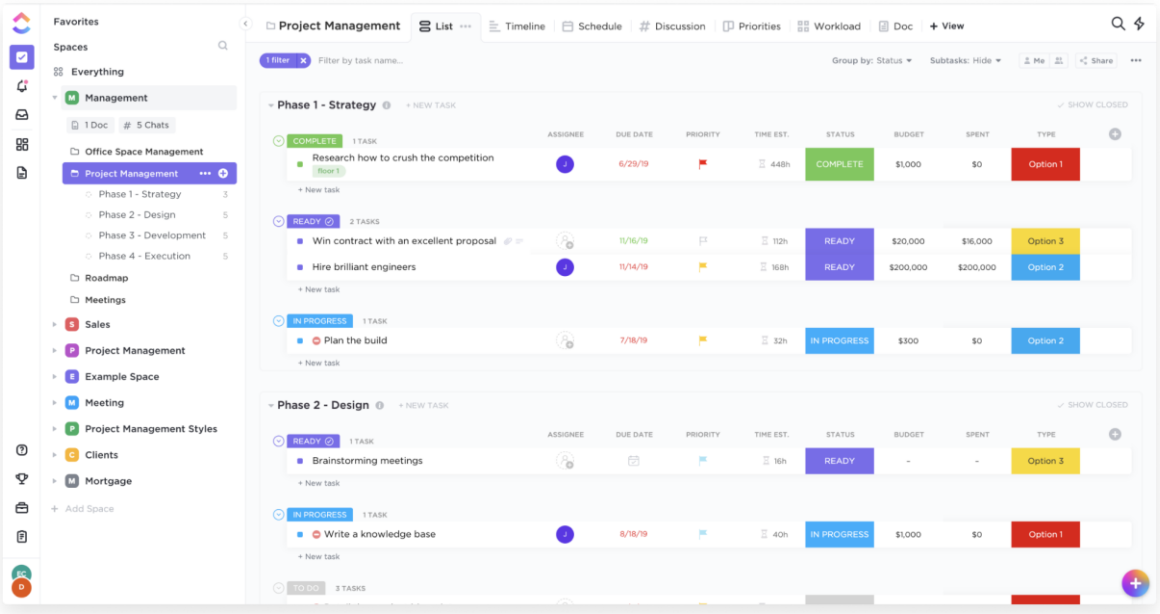



Sl. 2.5. Prikaz podataka jednog tiketa u JitBit sustavu. [13]

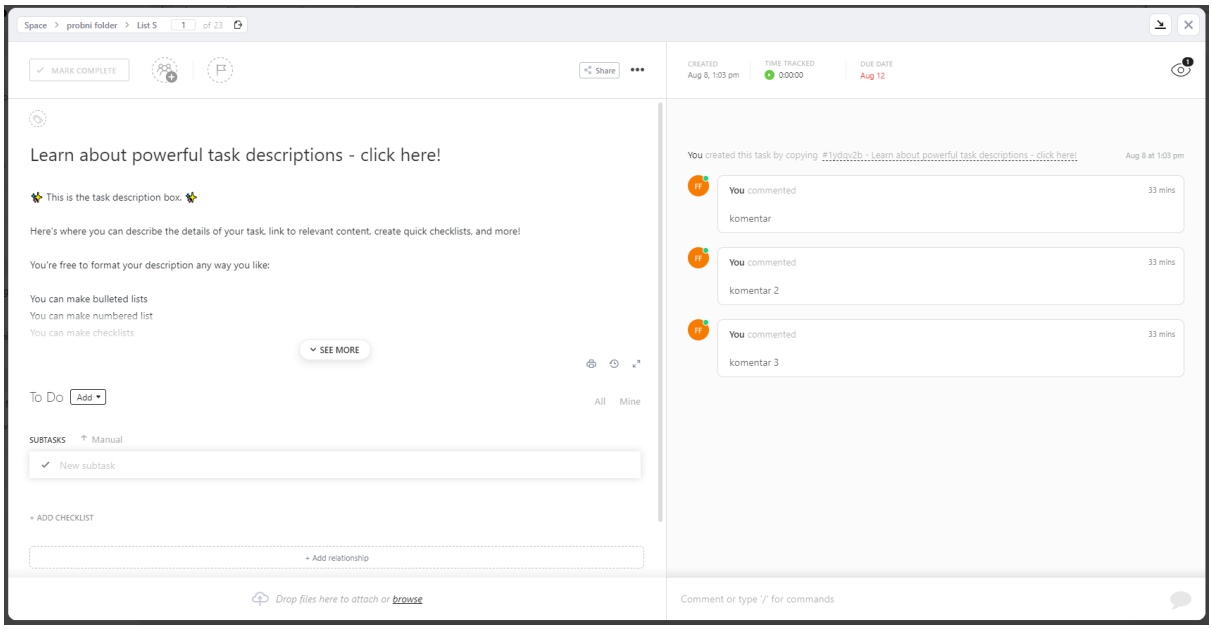
### ClickUp

ClickUp se promovira kao mnogo jeftinija zamjena za sustave kao što su Jira, Axosoft i sl. što njegova cijena i potvrđuje. Postoji besplatna verzija koja sadrži velik broj korisnih funkcionalnosti za praćenje projekata i tiketa, a ostale verzije su jeftinije od konkurenata na tržištu. Svojom funkcionalnošću ClickUp je sličan Axosoftu, ali posjeduje preglednije web korisničko sučelje (primjer prikazan na slici 2.6.), u ponudi ima manje funkcionalnosti vezanih za upravljanje tiketima. ClickUp je i dalje sustav koji omogućava korisnicima praćenje projekta i tiketa, ali samo za pojedince ili tvrtke manje do srednje veličine koje se bave marketingom ili jednostavnom korisničkom podrškom. Razlog tome je što ClickUp sustav još uvijek nije dovoljno razvijen te ima nedostataka: nemogućnost stvaranja različitih tipova tiketa (engl. *issue types*) ili stvaranja njihova prilagođenog oblika, nemogućnost korištenjem *REST API*-a dohvatiti sadržaja polja tiketa u obliku *rich* teksta (može biti velik problem za tvrtke koje zahtijevaju uređivanje opisa tiketa) i slično. Prikaz podataka pojedinog tiketa ClickUp sustava prikazan je na slici 2.7.

ClickUp je najavio da će navedeni nedostaci biti nadograđeni, ali u ovom trenutku to još uvijek nije ostvareno. Iz tih razloga ClickUp nije praktično rješenje za praćenje korisničkih upita u ispitivanju programske podrške u automobilskoj industriji. [14][15]



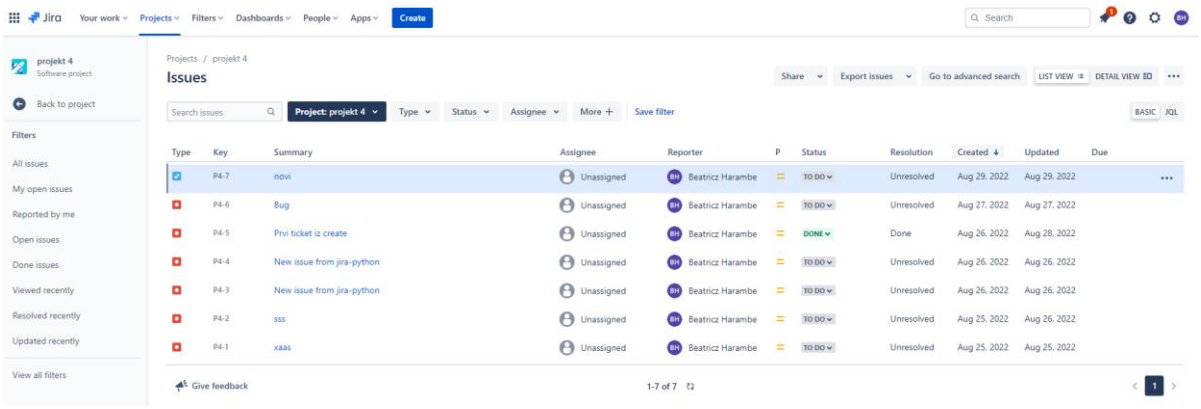

Sl. 2.6. Početno korisničko sučelje ClickUp sustava. [15]




Sl. 2.7. Prikaz podataka pojedinog ClickUp tiketa.

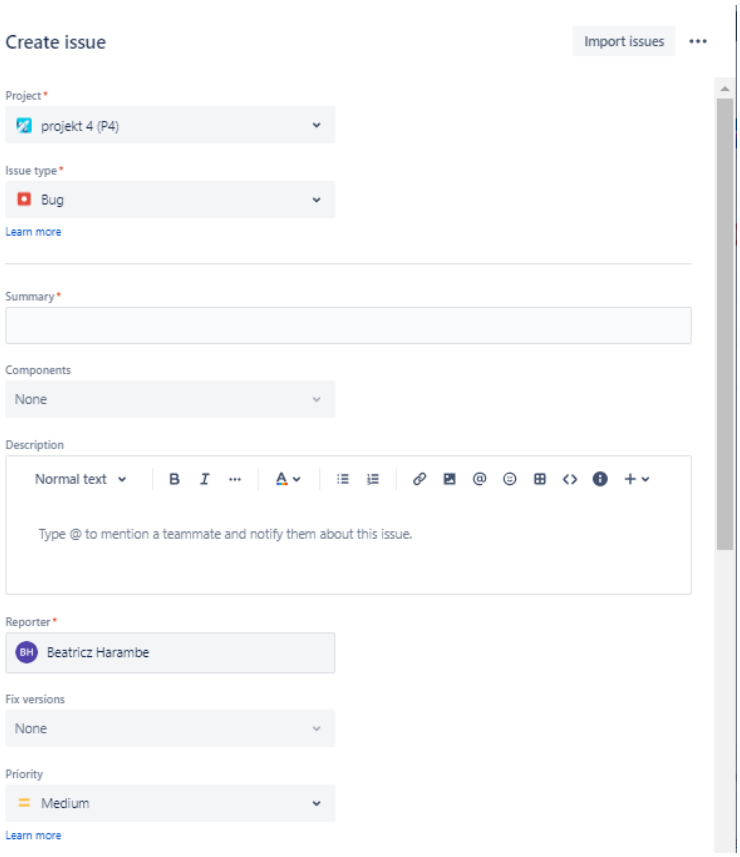
## Jira

Jira je proizvod tvrtke Atlassian za praćenje tiketa i agilno upravljanje projektima. U svojim funkcionalnostima nudi sve potrebne alate koji i više nego zadovoljavaju potrebe kojima je namijenjena. Raspored funkcionalnosti i prikaza na web korisničkom sučelju je pregledan i svrsishodan (slika 2.8.). Jira je poprilično jednostavna za korištenje. Omogućava stvaranje i istovremeno postojanje većeg broja projekata kojima se pristupa preko tražilice, *dropdown* taba ili otvaranjem prozora na kojem se nalazi tablica sa svim projektima i njihovim osnovnim informacijama (naziv, admin projekta i sl.).




Sl. 2.8. Prikaz početnog korisničkog sučelja Jira sustava.

Ovakvo odvajanje prostora za prikaz i odabir projekata ostavlja veći prostor za bolju preglednost tiketa unutar projekta. Stvaranje tiketa je jednostavno zbog točno određenih polja za popunjavanje čije se pojašnjenje može saznati prijelazom pokazivača miša. Prikaz primjera forme za stvaranje tiketa nalazi se na slici 2.9.




Sl. 2.9. Prikaz forme za stvaranje Jira tiketa.

Polja ponuđena pri stvaranju tiketa mogu se uređivati, dodavati nova ili korisnik može stvoriti jedno ili više sebi prilagođenih polja. Također postoje načini filtriranja informacija. Mogu se koristiti osnovni postojeći filteri za Jiru, a korisnik može stvoriti svoj vlastiti filter koristeći *Jira Query Language* i spremiti ga te tako omogućiti drugim korisnicima na projektu da koriste isti. Također su podržani Scrum i Kanban Board prikaz, backlog, agile izvješća, stvaranje Sandbox verzije Jira servera i sl. Sandbox je kopija Jira servera na kojoj se mogu testirati Jira *update*-ovi i aplikacije (engl. *app*) bez da se utječe na stvarni Jira server. To znači da korisnik može stvoriti Sandbox verziju servera kako bi testirao svoju aplikaciju koja komunicira sa Jira serverom bez strahovanja o mogućim posljedicama. Jira ima i nedostatke poput učestalih bug-ova na određenim funkcionalnostima koji se, prema izjavama korisnika, sporo ili uopće ne popravljaju. U nedostake se ubraja i ponekad nejasno napisana dokumentacija za korištenje *REST API*-a ili samog sustava. Usprkos nedostatcima, Jira predstavlja uravnoteženu ponudu alata za upravljanje projektima i tiketima te se nalazi na gotovo svim listama preporučenih sustava za korištenje kod upravljanja projektima i tiketima. Sa svojom uravnoteženom ponudom i jednostavnošću koja pruža mogućnost brzog osposobljavanja korisnika za rad na sustavu, Jira ticketing sustav je među boljim izborima za praćenje upita prilikom ispitivanja programske podrške u automobilskoj industriji. Moguće ju je koristiti zasebno u web pregledniku, ali nudi i razvijen *REST API*, *API* za javascript, *API* za

python i sl. Postojanje navedenih *API*-a omogućava automobilskim tvrtkama da u svoje postojeće sustave za testiranje programske podrške povežu svoj Jira sustav te tako skrate vrijeme potrebno za stvaranje, uređivanje i pregled tiketa. Trenutno u ponudi još uvijek postoje Cloud i Data Center usluga, ali Data Center usluga više nije dostupna za kupovinu te je najavljeno da 15.02.2024. prestaje raditi podrška servera. [16][17][18]

# 3. INTEGRACIJA TICKETING SUSTAVA SA SUSTAVOM ZA PROVOĐENJE UDALJENOG ISPITIVANJA PROGRAMSKE PODRŠKE

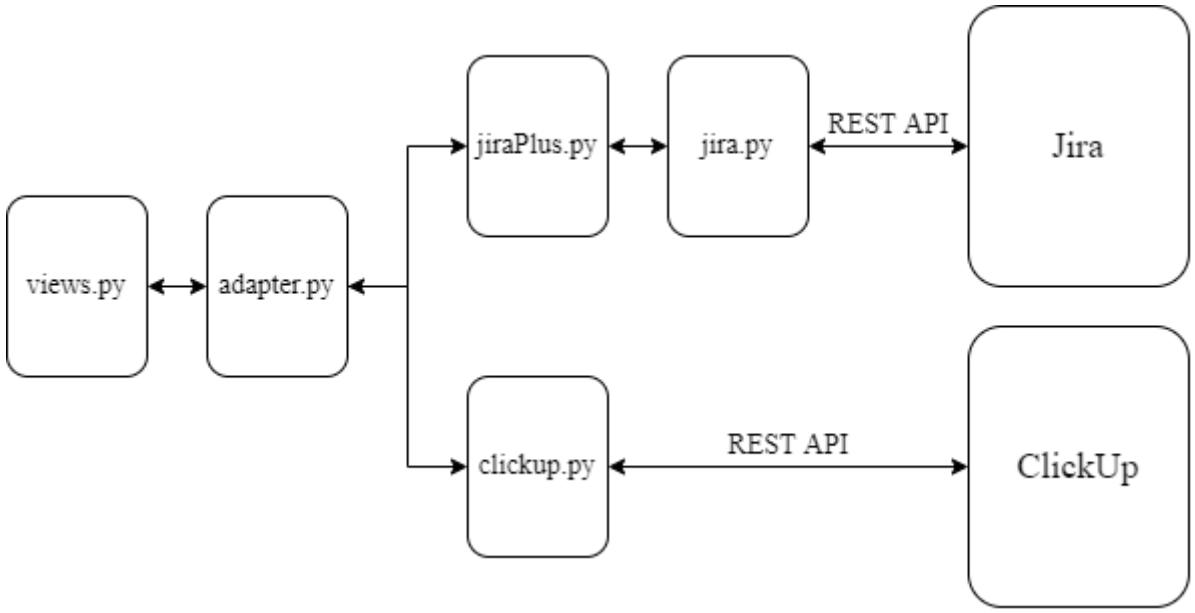
U nastavku rada je prikazan pojednostavljen primjer moguće implementacije integriranja ticketing sustava u sustav za udaljeno ispitivanje programske podrške. Slična rješenja nisu dostupna javnosti jer su sustavi tvrtki, u koje se obavlja integracija ticketing sustava, privatni sustavi za korištenje isključivo unutar tvrtke te se implementiraju ovisno o potrebama i zahtjevima pojedine tvrtke. Jedan od glavnih razloga zašto integrirati ticketing sustav u sustav za ispitivanje programske podrške je objedinjavanje dvaju ili više sustava u jedan. Time je testeru olakšan, ubrzan i prilagođen proces stvaranja, uređivanja i pregleda tiketa jer omogućava da korisnik na istom korisničkom sučelju ima pregled trenutnog rezultata testa, njegovih informacija i svih tiketa koji su do tada podignuti za taj test. Također se uklanja potreba za dodatnom prijavom u ticketing sustav.

## 3.1. Implementacija integracije ticketing sustava u postojeći sustav

Za izradu jednostavnog primjera sustava s integriranim ticketing sustavom korišten je *Full-Stack* Python Django okvir [19] te je uz Jiru dodatno odabran ClickUp ticketing sustav. ClickUp ticketing sustav je izabran za korištenje uz Jiru kako bi se prikazao primjer rješenja za ostvarivanje istovremene komunikacije s više ticketing sustava koji se mogu međusobno razlikovati. U primjeru je napravljeno korisničko sučelje za prijavu (engl. *login*) i korisničko sučelje za rad sa tiketima. Napravljen je modul *adapter.py* koji omogućava olakšan rad s oba odabrana ticketing sustava te moduli *jiraPlus.py* i *clickup.py* za komunikaciju s istima. Radi boljeg primjera mogućih načina komunikacije s ticketing sustavima, za komunikaciju sa ClickUp sustavom korišten je njegov *REST API* [20] dok je za komunikaciju s Jira ticketing sustavom korišten postojeći python *API* [21]. Za potrebe uređivanja izgleda korisničkih sučelja korišten je Bootstrap *open source* okvir za razvoj korisničkih sučelja [22].

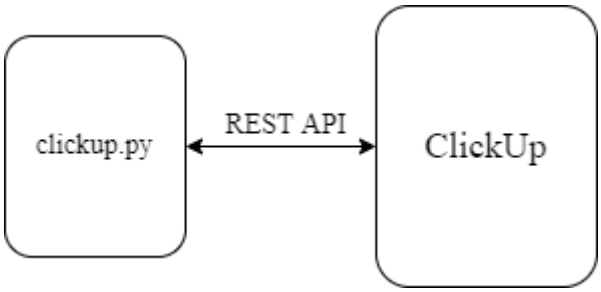
**Adapter** je osmišljeno rješenje u obliku *singleton*[1] klase (engl. *class*) koja omogućava da jedan sustav koji komunicira s više ticketing sustava to čini korištenjem generičkih funkcija koje su implementirane kao metode *Adapter* klase u *adapter.py* modulu. (slika 3.1.)

[1] Pojam *singleton* na hrvatskom jeziku je najbolje je opisan nazivom jedinstveni objekt, ali budući da u hrvatskom jeziku ne postoji jedinstven općeprihvaćen naziv za pojam *singleton* u nastavku ovoga rada koristit će se naziv *singleton* radi razumljivosti teksta.



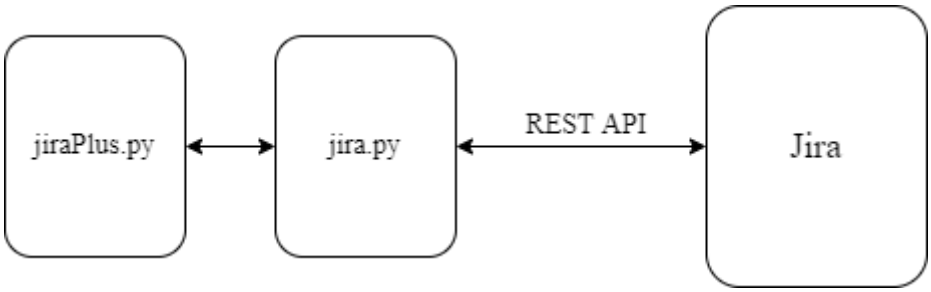

Sl. 3.1. Prikaz komunikacije između pojedinih modula programskog rješenja.

Komunikaciju s **ClickUp** sustavom u pythonu je moguće ostvariti korištenjem *REST API*-a ili korištenjem *clickup-python* biblioteke [23]. U izrađenom sustavu, radi primjera načina mogućeg korištenja *REST API*-a pri komunikaciji s ticketing sustavom, nije korištena gotova biblioteka već je izrađena klasa čije metode pozivaju odgovarajuće krajnje točke ClickUp *REST API*-a. (slika 3.2.)




Sl. 3.2. Prikaz komunikacije sa sustavom ClickUp.

Komunikaciju s **Jirom** moguće je ostvariti korištenjem *REST API*-a ili korištenjem *jira* biblioteke. U izrađenom primjeru koristi se *jira* python biblioteka. Pozivi *jira API*-a obavljaju se unutar metoda *Jira* klase iz *jiraPlus.py* modula gdje su odgovori (engl. *response*) *jira API*-a obrađeni tako da se olakša njihovo daljnje korištenje (slika 3.3.).




Sl. 3.3. Prikaz komunikacije sa sustavom Jira.

### 3.1.1. Priprema modula

Aplikacija je osmišljena tako da se prilikom prijave korisnika odmah postave vrijednosti atributa korištenih klasa modula *adapter.py, jiraPlus.py* i *clickup.py*. Unutar modula *adapter.py* nalazi se klasa *Adapter* čija se instanca stvara prilikom prijave korisnika te kao atribute sadrži instance klase *Jira* iz *jiraPlus.py* modula i klase *ClickUp* iz *clickup.py* modula koje se stvaraju sa svakim pozivom konstruktora *Adapter* klase. U nastavku se nalaze detaljniji opisi stvaranja instanci navedenih klasa te opis radnji koje je potrebno napraviti ako se želi dodati komunikacija s novim ticketing sustavom.

***Adapter***

Kako se ne bi stvarala nova instanca *Adapter* klase za svako pozivanje neke od njezinih metoda, klasa je implementirana kao *singleton*. Kada korisnik unese podatke potrebne za prijavu na jedan ili oba ticketing sustava, poziva se funkcija *createAdapterInstance* iz *views.py* datoteke u kojoj se podaci s korisničkog sučelja za prijavu obrađuju i pojedinačno ili u obliku rječnika predaju u *Adapter* konstruktor. Ako je stvaranje *Adapter* instance prošlo bez nastanka pogreške (engl. *Error*), tada kao povratnu vrijednost *createAdapterInstance* funkcija vraća status 200, u suprotnom se za povratnu vrijednost vraća status 400. U ostalim funkcijama unutar *views.py* datoteke za komunikaciju s Jira ili ClickUp sustavom, kako bi se koristile za to potrebne metode *Adapter* klase, prvo se poziva *static* metoda *getInstance* (iz *Adapter* klase) da bi se dobila instanca *Adapter* klase koja je stvorena prilikom prijave korisnika u sustav te se pomoću dobivene instance poziva potrebna metoda *Adapter* klase.

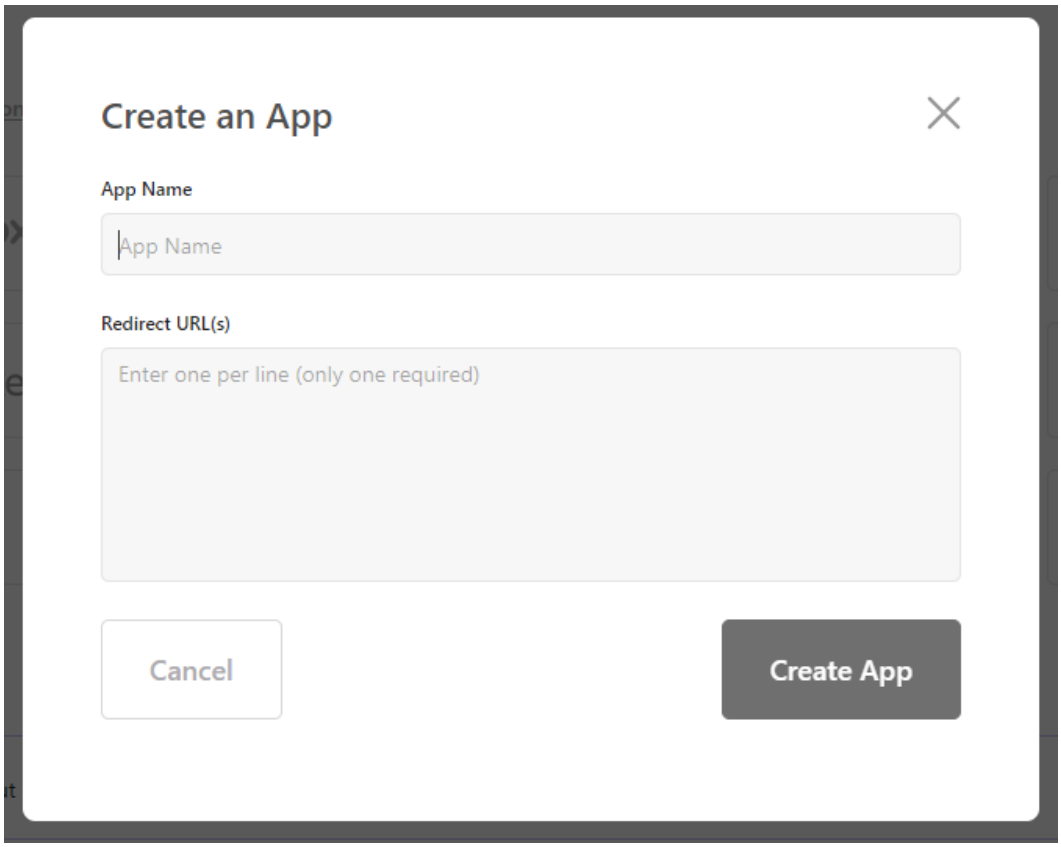
Tab. 3.1. Metode *Adapter* klase za stvaranje instance.

| Metoda | Opis |
|---|---|
| *__init__* | Konstruktor *Adapter* klase. Ako je prvi puta pozvan postavlja vrijednosti *jira, clickup* i *__shared_instance* atributima. U |

| | |
|---|---|
| | suprotnom postavlja vrijednosti samo *jira* i *clickup* atributima. |
| *getInstance* | *Static* metoda *Adapter* klase. Ako postoji kao povratnu vrijednost vraća instancu *Adapter* klase. U suprotnom podiže *Exception* s odgovarajućom porukom. |
| *deleteInstance* | Postavlja *__shared_instance* vrijednost na početnu. |

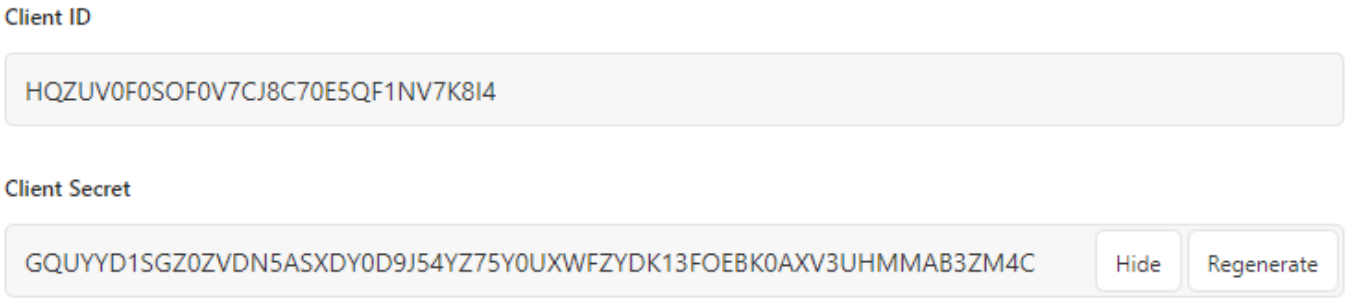
***ClickUp***

Postoje dva načina za komunikaciju s ClickUp sustavom korištenjem *REST API*-a. Jedan od načina je korištenjem korisnikovog osobnog (engl. *personal*) *API token*-a koji korisnik samostalno generira u ClickUp sustavu. Drugi način je da administrator za željeni *workspace* na serveru stvori aplikaciju kojoj se kao *redirect URL(s)* predaju *URL*-ovi sustava s kojim želi omogućiti komunikaciju (slika 3.4.).




Sl. 3.4. Prikaz forme za stvaranje ClickUp aplikacije.

Prilikom stvaranja aplikacije za željeni *workspace*, administrator dobiva *Client ID* i *Client Secret* za tu aplikaciju koje može vidjeti samo administrator *workspace*-a (slika 3.5.).




Sl. 3.5. Prikaz izgleda Client ID i Client Secret informacija.

Da bi se za komunikaciju s ClickUp-om koristili dobiveni Client ID i Client Secret, potrebno je prilikom svake prijave korisnika preusmjeriti (engl. *redirect*) korisnika na ClickUp domenu. Preusmjeravanje korisnika se obavlja putem poveznice (engl. *link*) *https://app.clickup.com/api* kojoj se nadodaje nastavak *?client_id={client_id}&redirect_uri={redirect_uri}* gdje se kao parametri predaju *Client ID* i *URL* servera od sustava za ispitivanje programske podrške. Korisnik zatim na sučelju prikazanom na slici 3.6. odabire jedan ili više željenih *workspace*-ova za rad te je nakon toga vraćen na server od sustava za ispitivanje programske podrške s novodobivenim parametrom *code*. Parametar *code* se mijenja prilikom svakog preusmjeravanja korisnika na ClickUp domenu. Pomoću *Client ID*-a, *Client Secret* i jedinstvenog *code*-a pozivom krajnje točke */oauth/token?client_id={client_id}&client_secret={client_secret}&code={code}* ClickUp *REST API*-a kao odgovor se dobiva jedinstveni *access token* koji kao i osobni *API token* omogućava korištenje ClickUp *REST API*-a. Metode *ClickUp* klase koje se koriste pri stvaranju instance iste opisane su u tablici 3.2.

Tab. 3.2. Opis metoda *ClickUp* klase vezanih za instancu klase.

| Metoda | Opis |
|---|---|
| *__init__* | Konstruktor *ClickUp* klase. Postavlja vrijednosti atributa *ClickUp* klase ovisno o predanim parametrima i trenutnim vrijednostima atributa *__acces_token* i *__code*. Ako nije predan parametar za osobni *API token* poziva metodu *__getAccessToken* i njezinu povratnu vrijednost postavlja za vrijednost atributa *access_token*. U suprotnom vrijednost predanog parametra za osobni *API token* postavlja za vrijednost *access_token* atributa. |
| *__getAccessToken* | Dohvaćanja ClickUp *access token* pozivom *oauth/token?client_id={client_id}&client_secret={client_secret}&code={code}* krajnje točke ClickUp *REST API*-a. |


Sl. 3.6. Prikaz sučelja za odabir sa 2 ponuđena ClickUp *workspace*-a.

***Jira***

Prilikom korištenja *jira* python biblioteke potrebno je prvo napraviti instancu klase *JIRA*. Kao parametre u konstruktor *JIRA* klase predaju se:

- poveznica Jira servera ili oblaka (engl. *cloud*)
- adresa e-pošte (engl. *e-mail*) korisnika
- *API token* korisnika.

U slučaju komunikacije s lokalnim Jira serverom (korisnikov ili tvrtkin) postoji mogućnost da se još uvijek podržava prijava korisničkim imenom i lozinkom te *API token* i e-adresa korisnika nisu potrebne. Instanca klase *JIRA* stvara se i pohranjuje kao atribut *Jira* klase u konstruktoru *Jira* klase pozivanjem metode *getJIRAInstance* i postavljanjem njezine povratne vrijednosti za vrijednost *jira* atributa. U tablici 3.3. nalazi se opis u tekstu navedenih metoda *Jira* klase.

Tab. 3.3. Metode *Jira* klase za stvaranje instance.

| Metoda | Opis |
|---|---|
| *__init__* | Konstruktor *Jira* klase. Poziva metodu *getJIRAInstance* i postavlja dobivenu povratnu vrijednost kao vrijednost *jira* atributa. Atribut *jira* koristi se u *Jira* klasi za pozivanje metoda *JIRA* klase. |
| *getJIRAInstance* | Postavlja dane parametre (poveznica na Jira server, e-adresa korisnika, *API token*) u oblik za predaju *JIRA* konstruktoru. Kao povratnu vrijednost vraća instancu *JIRA* klase. |

**Dodavanje dodatnog ticketing sustava**

U slučaju dodavanja novog ticketing sustava uz Jira i ClickUp sustav, za potrebe komunikacije s istim, prvo je potrebno proučiti moguće načine komunikacije s odabranim ticketing sustavom. Ako ne postoji već napravljena python biblioteka za komunikaciju s ticketing sustavom, potrebno je stvoriti klasu te u nju implementirati konstruktor koji će kao vrijednost atributa klase postaviti podatak koji se koristi za dobivanje odobrenja korištenja *REST API*-a odabranog ticketing sustava. Zatim je potrebno unutar konstruktora *Adapter* klase (tablica 3.1.) postaviti parametre kojima će biti predane vrijednosti potrebne za poziv ranije implementiranog konstruktora klase za odabrani ticketing sustav. Posljednji korak je unutar konstruktora *Adapter* klase (tablica 3.2.) postaviti uvjet za poziv konstruktora klase odabranog ticketing sustava te dobivenu instancu spremiti kao atribut *Adapter* klase za daljnje pozivanje njenih metoda.

### 3.1.2. Hijerarhija sustava

Kod različitih ticketing sustava vrlo je vjerojatno da imaju međusobno različite nazive za iste pojmove, kao u slučaju tiketa, ali nerijetko se događa da imaju i različite hijerarhije za razvrstavanje tiketa. Različiti nazivi za iste pojmove nemaju velik utjecaj na težinu implementacije komunikacije s različitim sustavima. Budući da oba naziva predstavljaju isti pojam prilikom

implementacije ih se generalizira jednim nazivom. U slučaju različitih hijerarhija potrebno je znati hijerarhiju svih ticketing sustava koji se koriste i odrediti gdje se hijerarhije poklapaju te ih na tim razinama generalizirati, a za ostale razine hijerarhije stvoriti zasebne načine dohvaćanja. Moguće je i postaviti stalne vrijednosti za razine hijerarhije koje se razlikuju sve do prvih koje se poklapaju, ali tako je omogućena komunikacija s ticketing sustavima samo do određene razine. Što znači da će korisnik imati mogućnost stvaranja tiketa samo na strogo određenom mjestu ticketing sustava. U rješenju koje nudi ovaj rad primijenjen je prvi navedeni pristup rješavanja problema nepodudarnih hijerarhija. Nastavak rada opisuje metode *Adapter, Jira* i *ClickUp* klasa koje se koriste prilikom dohvaćanja elemenata hijerarhije pojedinog ticketing sustava te nudi objašnjenje kako dodati dohvaćanje elemenata novododanog ticketing sustava.

***Adapter***

U *Adapter* klasi se nalaze dvije metode za dohvaćanje podudarajućih elemenata hijerarhije Jira i ClickUp sustava, *get_Lists_or_Projects* i *getTickets*. One generaliziraju metode za dohvaćanje tiketa s oba sustava te metode za dohvaćanje Jira projekata i ClickUp *list*-i budući da su to dvije podudarne razine hijerarhije Jira i ClickUp sustava. Naziv i opis funkcija nalazi se u tablici 3.4.

Tab. 3.4. Metode *Adapter* klase za dohvaćanje elemenata hijerarhije.

| Metoda | Opis |
|---|---|
| *get_Lists_or_Projects* | Koristi *getListsFromFolder* metodu *ClickUp* klase i *get_Projects* metodu *Jira* klase. Dohvaća sve Jira projekte ili ClickUp *list*-e, ovisno koji je sustav zadan. Kao povratnu vrijednost vraća listu elemenata u kojoj svaki element predstavlja jedan Jira projekt ili ClickUp *list*-u i sadrži njegove vrijednosti identifikatora i ime. (tablice 3.5. i 3.6.) |
| *getTickets* | Koristi *getTasksFromList* metodu *ClickUp* klase i *get_IssueKeys* metodu *Jira* klase. Dohvaća sve tikete iz zadanog sustava. Kao povratnu vrijednost vraća listu elemenata u kojoj svaki element predstavlja jedan tiket te sadrži vrijednost njegovog identifikatora i ime. (tablice 3.5. i 3.6.) |

***ClickUp***

U hijerarhiji ClickUp sustava na najvišoj razini je *workspace*, zatim redom slijede: *space, folder, list, task* (tiket) i *subtask*. Pri korištenju ClickUp sustava putem web preglednika nije potrebno stvarati *folder*-e u koje će se zatim stvarati *list*-e u kojima se tada mogu stvarati tiketi. Moguće je stvoriti *list*-e odmah u željenom *space*-u, ali to bi otežalo komunikaciju sa ClickUp sustavom putem *REST API*-a u slučaju da se *folder*-i nasumično koriste ili ne koriste. Zato je potrebno postaviti određena pravila o načinu stvaranja hijerarhije u *workspace*-u. Za potrebe diplomskog rada odlučeno je uvijek koristiti *folder*. U napravljenoj *ClickUp* klasi su, zbog potrebe prolaska kroz cijelu ClickUp hijerarhiju prilikom traženja željenog tiketa, napravljene metode kojima se dohvaća timove (*workspace*-ove), *folder*-e i *list*-e koje korisnik može vidjeti i na ClickUp serveru. U tablici 3.5. se nalaze nazivi i opisi tih metoda.

Tab. 3.5. Metode *ClickUp* klase za dohvaćanje elemenata hijerarhije.

| Metoda | Opis |
|---|---|
| *getTeams* | Dohvaća sve *workspace*-ove pod uvjetom da korisnik ima pristup istima i da ih je odabrao prilikom prijave (u slučaju korištenja *Client ID*-a i *Client Secret*-a za prijavu). Kao povratnu vrijednost vraća listu elemenata u kojoj svaki element predstavlja jedan *workspace* i sadrži njegove vrijednosti (ime, identifikator, članovi i sl.). |
| *getSpaces* | Dohvaća sve *space*-ove iz zadanog *workspace*-a. Kao povratnu vrijednost vraća listu elemenata u kojoj svaki element predstavlja jedan *space* i sadrži njegove vrijednosti (ime, identifikator i sl.). |
| *getFolders* | Dohvaća sve *folder*-e iz zadanog *space*-a. Kao povratnu vrijednost vraća listu elemenata u kojoj svaki element predstavlja jedan *folder* i sadrži njegove vrijednosti (ime, identifikator i sl.). |
| *getListsFromFolder* | Dohvaća sve *list*-e iz zadanog *folder*-a. Kao povratnu vrijednost vraća listu elemenata u kojoj svaki element predstavlja jedan *list* i sadrži njegove vrijednosti (ime, identifikator i sl.). |
| *getTasksFromList* | Dohvaća sve tikete iz zadane *list*-e. Kao povratnu vrijednost |

| | |
|---|---|
| | vraća listu elemenata u kojoj svaki element predstavlja jedan tiket i sadrži njegove vrijednosti (ime, identifikator i sl.). |

***Jira***

U Jiri postoje samo projekti u kojima se odmah stvaraju tiketi. Korisnik može imati pristup više projekata i u svakom od njih stvarati tikete. Ono po čemu se Jira razlikuje od ClickUp-a je to što je potrebno odabrati tip tiketa koji se stvara. Svaki Jira projekt ima svoje tipove tiketa koje je moguće u njemu stvoriti. Iz tog razloga je u *Jira* klasi implementirana metoda za dohvaćanje tipova tiketa u zadanom projektu kako bi se mogli prikazati i odabrati prilikom stvaranja tiketa u sustavu za ispitivanje programske podrške. Naziv i opis metode nalaze se u tablici 3.6.

Tab. 3.6. Metode *Jira* klase za dohvaćanje elemenata hijerarhije.

| Metoda | Opis |
|---|---|
| *get_Projects* | Dohvaća sve projekte za koje korisnik ima pravo pristupa. Kao povratnu vrijednost vraća listu elemenata u kojoj svaki element predstavlja jedan projekt i sadrži njegove vrijednosti identifikatora i ime. |
| *get_IssueKeys* | Dohvaća sve tikete iz zadanog projekta. Kao povratnu vrijednost vraća listu elemenata u kojoj svaki element predstavlja jedan tiket iz odabranog Jira projekta te sadrži vrijednosti njegovih identifikatora (*issue key* i *issue id*). |
| *get_IssueTypes* | Dohvaća sve tipove tikete iz zadanog projekta. Kao povratnu vrijednost vraća listu elemenata u kojoj svaki element predstavlja jedan tip tiketa i sadrži njegovu vrijednost identifikatora i ime. |

**Dodavanje dodatnog ticketing sustava**

Za najjednostavniji način dodavanja dohvaćanja elemenata hijerarhije ticketing sustava, uz Jira i ClickUp sustav, potrebno je proučiti njegovu hijerarhiju te odrediti koja se njegova razina podudara sa Jirinom *project* razinom i ClickUp-ovom *list* razinom. Zatim unutar *Adapter* klase u metodi *get_Lists_or_Projects,* opisanoj u tablici 3.4., postaviti uvjet za pozivanje funkcije kojom

se dohvaćaju elementi podudarajuće razine hijerarhije sustava te povratnu vrijednost funkcije, ako je to potrebno, obraditi kako bi bila proslijeđena u istom obliku primjera koji je prikazan na slici 3.7.

```
lists_or_projects=[
                    {'id': list_project_id_1, 'name': list_project_name_1},
                    {'id': list_project_id_2, 'name': list_project_name_2},
                    {'id': list_project_id_3, 'name': list_project_name_3},
                                                ...
                    {'id': list_project_id_n, 'name': list_project_name_n}
                  ]
```

Sl. 3.7. Primjer oblika povratne vrijednosti *get_Lists_or_Projects* metode.

Nakon što je ostvarena mogućnost dohvaćanja elemenata hijerarhije na razini koja se podudara s Jira *project* i ClickUp *list* razinom, potrebno je isti postupak ponoviti za dohvaćanje svih tiketa novog ticketing sustava koji se nalaze unutar jednog od ranije dobivenih elemenata hijerarhije. Dakle, unutar *Adapter* klase u metodi *getTickets,* opisanoj u tablici 3.4., postaviti uvjet za pozivanje funkcije kojom se dohvaćaju svi tiketi novododanog ticketing sustava iz određenog elementa iz liste ranije dobivenih elemenata hijerarhije te povratnu vrijednost funkcije, ako je to potrebno, obraditi kako bi bila proslijeđena u istom obliku primjera koji je prikazan na slici 3.8.

```
tickets=[
        {'id': ticket_id_1, 'name': ticket_name_1},
        {'id': ticket_id_2, 'name': ticket_name_2},
        {'id': ticket_id_3, 'name': ticket_name_3},
                              ...
        {'id': ticket_id_n, 'name': ticket_name_n}
       ]
```

Sl. 3.8. Primjer liste tiketa.

### 3.1.3. Stvaranje tiketa

Kako bi u sustavu za ispitivanje programske podrške bilo moguće stvoriti tiket, potrebno je proučiti koji su osnovni podaci potrebni za stvaranje tiketa u pojedinom sustavu te zatim implementirati funkcije koje će podatke primati, obraditi te predati pri stvaranju tiketa. U nastavku rada su opisane metode *Adapter, Jira* i *ClickUp* klasa koje se koriste prilikom stvaranja tiketa te objašnjenje kako dodati stvaranje tiketa za novododani sustav.

***Adapter***

Metoda *createTicket* u *Adapter* klasi, ovisno o primljenom *ticket_sys* parametru, poziva *create_Issue* metodu *Jira* klase ili *createTask* metodu *ClickUp* klase i predaje im primljene

parametre (*project_or_list_id, ticket_name, ticket_description, issuetype*) u obliku koji pojedina metoda zahtjeva. Kao povratnu vrijednost vraća identifikator novonastalog tiketa ili podiže *ValueError*. Zahvaljujući *createTicket* metodi *Adapter* klase, u *views.py* istoimenoj funkciji *createTicket* je potrebno samo koristeći postojeću instancu *Adapter* klase pozvati *createTicket* metodu *Adapter* klase i kao parametre joj predati podatke primljene u zahtjevu (engl. *request*). Za stvaranje tiketa *createTask* metodom primljeni parametri se metodi predaju pojedinačno, a za stvaranje tiketa *create_Issue* metodom primljeni parametri se predaju u obliku rječnika koji je prikazan na slici 3.9.

Tab. 3.7. Metoda *Adapter* klase za stvaranje tiketa.

| Metoda | Opis |
|---|---|
| *createTicket* | Koristi *createTask* metodu *ClickUp* klase i *create_Issue* metodu *Jira* klase. Stvara tiket u odabranom Jira projektu ili ClickUp *list*-i, ovisno koji su sustav i projekt/*lista* zadani. Kao povratnu vrijednost vraća identifikator novostvorenog tiketa. (tablice 3.8. i 3.9.) |

```
field_contents = {
                'project': {'id': project_id},
                'summary': ticket_name,
                'description': ticket_description,
                'issuetype': {'id': issuetype_id}
            }
```

Sl. 3.9. Primjer *field_contents* rječnika u metodi *createTicket*.

***ClickUp***

Korištenjem ClickUp *REST API*-a stvaranje ticketa se obavlja korištenjem krajnje točke */list/{list_id}/task*. Krajnja točka *REST API*-a za stvaranje tiketa poziva se u *createTask* metodi *ClickUp* klase. Za stvaranje novog tiketa u ClickUp sustavu potrebno je predati identifikator liste u kojoj se tiket želi stvoriti, korisnikov osobni *API token* ili *access token* te rječnik u obliku prikazanom na slici 3.10.

```
data = {
        "name": task_name,
        "description": task_description
       }
```

Sl. 3.10. Primjer oblika *data* rječnika za uređivanje i dodavanje tiketa.

U rječniku se mogu nalaziti i vrijednosti za polja koja nisu obavezna za stvaranje tiketa. Na slici 3.11. je prikazan primjer slanja zahtjeva na krajnju točku */list/{list_id}/task* ClickUp *REST API*-a i obrada vrijednosti u odgovoru iste unutar *createTask* metode (tablica 3.8.). U primjeru se predaje *data* rječnik u obliku prikazanom na slici 3.10., *self.access_token* sadrži vrijednost osobnog *API tokena* ili *access tokena* ClickUp sustava i parametar *list_id* sadrži vrijednost *list* identifikatora ClickUp sustava. Odgovor je u *json* obliku te se iz tog razloga prvo parsira kako bi se lakše dohvaćale željene vrijednosti, u slučaju primjera dohvaća se vrijednost identifikatora novonastalog tiketa.

Tab. 3.8. Metoda *ClickUp* klase za stvaranje tiketa.

| Metoda | Opis |
|---|---|
| *createTask* | Stvara tiket u odabranoj ClickUp *list*-i. Kao povratnu vrijednost vraća identifikator novostvorenog tiketa. (slika 3.10.) |

```
headers = {
    'Authorization': self.access_token,
}
response = requests.request(
    'POST',
    url= 'https://api.clickup.com/api/v2/list/'+ str(list_id) +'/task',
    data=data,
    headers=headers,
    )
taskInfo = json.loads(response.content)
return taskInfo['id']
```

Sl. 3.11. Prikaz primjera slanja zahtjeva na krajnju točku */list/{list_id}/task*.

***Jira***

Korištenjem Jira python *API*-a moguće je stvoriti tiket korištenjem *JIRA* metode *create_issue*. Prilikom korištenja *JIRA create_issue* metode kao parametre je moguće predati

vrijednost svakog polja zasebno ili u obliku rječnika. Osnovni podaci tiketa čije se vrijednosti obavezno mora predati prilikom stvaranja tiketa opisani su u tablici 3.9. te se mogu predati pojedinačno ili u obliku rječnika prikazanog na slici 3.9.

Tab. 3.9. Obvezni podaci pri stvaranju Jita tiketa

| Podatak | Opis |
|---|---|
| Identifikator Jira projekta | Jira projekti posjeduju dva identifikatora, a to su *project ID* i *project key*. Prilikom stvaranja tiketa korištenjem *create_issue* metode klase *JIRA* potrebno je predati jedan od navedenih identifikatora kako bi tiket bio stvoren u željenom projektu. Ukoliko niti jedan od identifikatora nije predan, nije moguće stvoriti tiket. |
| Identifikator tipa tiketa | Svaki Jira projekt nudi određen broj tipova tiketa koje je moguće unutar njega stvoriti. Svaki tip tiketa posjeduje svoj vlastiti identifikator (*issue type ID*) i ime (*issue type name*). Prilikom stvaranja tiketa korištenjem *create_issue* metode klase *JIRA* potrebno je predati jedno od dvaju navedenih vrijednosti. U suprotnom nije moguće stvoriti tiket. |
| Vrijednosti obveznih polja | Za svaki tip tiketa postoje osnovni podaci tiketa koje je, uz identifikator projekta i tipa tiketa potrebno, potrebno predati kako bi se mogao stvoriti tiket. Isti tipovi u različitim projektima mogu imati različite osnovne podatke koje traže. No, u većini slučajeva obvezni su podaci samo za polja *summary* (kratak opis tiketa) i *description* (detaljan opis tiketa). |

U metodi *Jira* klase *create_Issue* (tablica 3.10.) poziva se *JIRA* metoda *create_issue* kojoj se kao parametar predaje rječnik *fields* koji je oblika prikazanog na slici 3.9. i sadrži vrijednosti za ranije navedene parametre i *prefetch* parametar. Parametrom *prefetch* se upravlja hoće li se ili neće odmah osvježiti stvoreni *issue Resource* kako bi svi njegovi podaci bili prisutni u povratnoj vrijednosti *create_issue* metode. Prilikom stvaranja tiketa na Jira lokalnom serveru, navedeni proces može uzrokovati nastanak pogreške te se iz tog razloga *prefetch* parametru predaje *bool* vrijednost *False.* Koriste se *try* i *except* blokovi za dohvaćanje mogućih *JIRAError* pogrešaka pri stvaranju tiketa. Ako je tiket uspješno napravljen/podignut na Jira server ili oblak, *Jira* metoda

*create_Issue* vraća vrijednost tiket ključa (engl. *issue key*), u suprotnom vraća tekst uhvaćene pogreške. Prikaz opisane implementacije *create_Issue* metode moguće je, radi lakše predodžbe, pogledati na slici 3.12.

Tab. 3.10. Metoda *Jira* klase za stvaranje tiketa.

| Metoda | Opis |
|---|---|
| *create_Issue* | Stvara tiket u odabranom Jira projektu. Kao povratnu vrijednost vraća identifikator novostvorenog tiketa. (slika 3.12.) |

```
def create_Issue(self, fields):

    try:
        issue = self.jira.create_issue(fields= fields, prefetch= False)
    except JIRAError as e:
        return e.text

    return issue.key
```

Sl. 3.12. Prikaz implementacije *create_Issue* metode.

**Dodavanje dodatnog ticketing sustava**

U slučaju dodavanja novog ticketing sustava uz Jira i ClickUp sustav, za potrebe stvaranja tiketa u novom ticketing sustavu prvo je potrebno implementirati funkciju za stvaranje tiketa u novom ticketing sustavu. Nakon toga je potrebno unutar *createTicket* (tablica 3.7.) metode *Adapter* klase implementirati uvjet koji je potrebno ostvariti kako bi se pozvala funkcija za stvaranje tiketa na novom ticketing sustavu. Ako je potrebno, obraditi parametre koje prima *createTicket* metoda kako bi odgovarali obliku potrebnom za poziv funkcije za stvaranje tiketa na novom ticketing sustavu.

### 3.1.4. Dohvaćanje tiket polja

Da bi u sustavu za ispitivanje programske podrške bilo moguće vidjeti željena polja tiketa i njihove vrijednosti, potrebno je znati kako dohvatiti podatke tiket polja te u kakvom obliku sustav vraća iste kako bi ih bilo moguće obraditi. U slučaju Jira i ClickUp sustava, oba sustava vrijednosti tiket polja vraćaju u obliku *JSON*. Nakon što je poznata povratna vrijednost sustava može se implementirati funkcije za dohvaćanje, obradu i prosljeđivanje istih. U nastavku rada su opisane

metode *Adapter, Jira* i *ClickUp* klasa koje se koriste prilikom dohvaćanja podataka tiketa te objašnjenje kako dodati mogućnost dohvaćanja tiketa za novododani sustav.

***Adapter***

Metoda *getTicketFields* u *Adapter* klasi, ovisno o primljenom *ticket_sys* parametru, poziva *getTask* i *getTaskComments* metode *ClickUp* klase (opisane u tablici 3.12.) ili *get_IssueFields* i *get_IssueComments* metode *Jira* klase (opisane u tablica 3.13.) i predaje im primljeni *ticket_id* parametar čija je vrijednost identifikator tiketa. Nakon poziva metoda (*Jira* ili *ClickUp* klase), njihova povratna vrijednost se obrađuje i predaje u uređeni *ticket_fields* rječnik, oblika prikazanog na slici 3.13., koji funkcija *getTicketFields* vraća kao povratnu vrijednost.

```
ticket_fields=[
                'name': ticket_name,
                'status': ticket_name,
                'description': ticket_name,
                'comments': [
                             {'id': comment_id_1, 'text': comment_text_1},
                                                  ...
                             {'id': comment_id_n, 'text': comment_text_n}
                            ]
                ]
```

Sl. 3.13. Primjer *ticket_fields* rječnika.

Rječnik *ticket_fields* sadrži dobivene nazive i vrijednosti određenih polja zadanog tiketa, njegov sadržaj može se razlikovati od primjera na slici 3.13. ovisno o tome iz kojeg sustava je tiket. U slučaju loše zadanog *ticket_sys* parametra podiže *ValueError*. Zahvaljujući *getTicketFields* metodi *Adapter* klase, u *views.py* istoimenoj funkciji *getTicketFields* je potrebno samo koristeći postojeću instancu *Adapter* klase pozvati *getTicketFields* metodu *Adapter* klase i kao parametre joj predati podatke primljene u zahtjevu.

Tab. 3.11. Metoda *Adapter* klase za dohvaćanje tiket polja.

| Metoda | Opis |
|---|---|
| *getTicketFields* | Koristi *getTask* i *getTaskComments* metode *ClickUp* klase i *get_IssueFields* i *get_IssueComments* metode *Jira* klase. Dohvaća polja odabranog tiketa i njihove vrijednosti. Kao povratnu vrijednost vraća *ticket_fields* rječnik s vrijednostima tiket polja. (slika 3.13., tablice 3.12. i 3.13.) |

***ClickUp***

ClickUp *REST API* sadrži krajnju točku */task/{task_id}* za dohvaćanje podataka određenog tiketa i krajnju točku */task/{task_id}/comment* za dohvaćanje komentara za određeni tiket. U *ClickUp* klasi su implementirane metode *getTask* i *getTaskComments,* koje su opisane u tablici 3.12., za dohvaćanje vrijednosti vrijednosti svih tiket polja i komentara.

Tab. 3.12. Metode *ClickUp* klase za dohvaćanje tiket polja.

| Metoda | Opis |
|---|---|
| *getTask* | Stvara i kao povratnu vrijednost vraća rječnik koji sadrži naziv i vrijednost za svako od određenih ticket polja (*summary, issuetype, priority, status, description*). Koristi se za dohvaćanje naziva i vrijednosti tiketa polja. |
| *getTaskComments* | Stvara GET poziv na krajnju točku ClickUp REST API-a /task/{task_id}/comment i vraća listu elemenata u kojoj svaki element predstavlja jedan komentar i sadrži njegove podatke. |

***Jira***

Korištenjem Jira python *API*-a moguće je dohvatiti *Issue* objekt za svaki tiket pojedinačno korištenjem *JIRA* metode *issue* te je moguće dohvatiti vrijednosti pojedinog komentara za pojedini tiket korištenjem *JIRA* metode *comment.* Prilikom korištenja *JIRA issue* metode kao parametar se predaje neki od identifikatora tiketa (*issue key* ili *issue id*) te se za odgovor dobiva objekt klase *Issue* koji posjeduje sve podatke o tiketu čiji je identifikator predan. *JIRA* metoda *issue* koristi se u *Jira* metodama *get_IssueFields* i *get_IssueComments*. Uz pomoć vrijednosti *fields* atributa *Issue* klase moguće je dohvatiti vrijednosti za svako polje i dohvatiti listu identifikatora komentara koja se u *get_IssueComments* metodi koristi za dohvaćanje vrijednosti svakog komentar i stvaranje stvaranje *renderedComments* liste u kojoj svaki element liste predstavlja jedan komentar tiketa te sadrži njegov identifikator i vrijednost u *HTML* obliku.

Tab. 3.13. Metode *Jira* klase za dohvaćanje tiket polja.

| Metoda | Opis |
|---|---|
| *get_IssueFields* | Stvara GET poziv na krajnju točku ClickUp REST API-a |

| | |
|---|---|
| | /task/{task_id} i vraća rječnik u kojem se nalaze sve vrijednosti odabranog tiketa. |
| *get_IssueComments* | Stvara i kao povratnu vrijednost vraća listu u kojoj svaki element liste predstavlja jedan komentar tiketa te sadrži njegov identifikator i vrijednost u *HTML* obliku. Koristi se za dohvaćanje vrijednosti svih komentara tiketa u *HTML* obliku. |

**Dodavanje dodatnog ticketing sustava**

U slučaju dodavanja novog ticketing sustava uz Jira i ClickUp sustav, za potrebe dohvaćanja polja tiketa sa novog ticketing sustava prvo je potrebno implementirati funkcije potrebne za dohvaćanje podataka polja tiketa sa novog ticketing sustava. Nakon toga je potrebno unutar *getTicketFields* (tablica 3.11.) metode *Adapter* klase implementirati uvjet koji je potrebno ostvariti kako bi se pozvale funkcije za dohvaćanje podataka polja tiketa sa novog ticketing sustava. Također je potrebno povratne vrijednosti funkcija obraditi i razvrstati ih u *ticket_fields* rječnik u obliku prema slici 3.13.

### 3.1.5. Uređivanje postojećeg tiketa

Kako bi u sustavu za ispitivanje programske podrške bilo moguće urediti vrijednosti željenih polja tiketa, potrebno proučiti za koja tiket polja pojedini ticketing sustav dopušta uređivanje te implementirati funkcije za dohvaćanje vrijednosti za uređivanje polja,obradu i njihovo postavljanje na ticketing sustav. Za dohvaćanje vrijednosti polja se koriste već spomenute metode opisane u tablicama 3.11., 3.12. i 3.13., u nastavku rada su navedene i opisane metode za uređivanje polja tiketa.

***Adapter***

U *editTicket* metodi *Adapter* klase je implementirano uređivanje zadanog tiketa iz zadanog ticketing sustava. Metoda *editTicket* u *Adapter* klasi, ovisno o primljenom *ticket_sys* parametru, poziva *editTask* metodu (Tablica 3.15.) *ClickUp* klase ili *edit_Issue* metodu (tablica 3.16.) *Jira* klase i predaje im primljene parametre (*ticket_id*, *ticket_name*, *ticket_description*, *comment*). Metodi *editTask* parametri se predaju pojedinačno, dok se metodi *edit_Issue* parametri predaju unutar *field_contents* rječnika čiji je primjer prikazan na slici 3.14., a parametar *comment* je predan zasebno.

```
field_contents = {
                    'summary': ticket_name,
                    'description': ticket_description_text
                }
```

Sl. 3.14. Primjer oblika *field_contents* rječnika za uređivanje tiketa Jira sustava.

Kao povratnu vrijednost, ovisno o uspješnosti uređivanja tiketa, *editTicket* metoda vraća status stanja 200 ili tekst nastalog problema. Zahvaljujući *editTicket* metodi *Adapter* klase, u *views.py* istoimenoj funkciji *editTicket* je potrebno samo koristeći postojeću instancu *Adapter* klase pozvati *editTicket* metodu *Adapter* klase i kao parametre joj predati podatke primljene u zahtjevu.

Tab. 3.14. Metoda *Adapter* klase za uređivanje tiketa.

| Metoda | Opis |
|---|---|
| *editTicket* | Koristi *editTask* metodu *ClickUp* klase i *edit_Issue* metodu *Jira* klase. Uređuje vrijednosti polja odabranog tiketa. Kao povratnu vrijednost vraća status 200 ili tekst nastalog problema. (Tablice 3.15. i 3.16.) |

***ClickUp***

Polja postojećeg ClickUp tiketa uređuju se korištenjem *REST API /task/{task_id}* krajnje točke, a komentari se dodaju korištenjem */task/{task_id}/comment* krajnje točke. U klasi *ClickUp* metoda *editTiask* stvara *PUT* poziv na krajnju točku ClickUp *REST API*-a */task/{task_id}* i u rječniku *data*, čiji je primjer oblika prikazan na slici 3.10. predaje vrijednosti za tiket polja koja se uređuje. Metoda vraća šifru stanja ili tekst pogreške koje vraća ClickUp sustav ovisno o uspješnosti napravljenog poziva.

Metoda *postTaskComment* iz *ClickUp* klase, stvara *POST* poziv na krajnju točku ClickUp *REST API*-a */task/{task_id}/comment* i u rječniku *data*, prikazanom na slici 3.15. predaje vrijednost za *comment_text*. Metoda vraća šifru stanja poziva dobivenu od ClickUp sustava.

```
data = { "comment_text": str(comment_text)}
```

Sl. 3.15. Prikaz *data* rječnika u *postTaskComment* metodi.

Tab. 3.15. Metode *ClickUp* klase za uređivanje tiketa.

| Metoda | Opis |
|---|---|
| *editTask* | Uređuje zadana tiket polja na ClickUp sustavu. |
| *postTaskComment* | Dodaje komentar na zadani tiket na ClickUp sustavu. |

***Jira***

U *Jira* klasi metoda *edit_Issue* poziva *JIRA* metodu *issue* kojoj predaje *issueKey* čija je vrijednost identifikator tiketa. Nakon dohvaćanja *Issue* objekta željenog tiketa poziva *update* metodu *Issue* klase kojoj kao parametar predaje rječnik čiji je primjer prikazan na slici 3.14. i koji sadrži vrijednosti za tiket polja čija se vrijednost želi urediti. Nakon toga se poziva *JIRA* metoda *add_comment* kojoj se kao parametri predaju *issueKey* i *comment* čija je vrijednost tekst komentara. Koriste se *try* i *except* blokovi za dohvaćanje mogućih *JIRAError* pogrešaka pri uređivanju tiketa. Kao povratnu vrijednost za uspješno uređivanje tiketa vraća vrijednost 200, u suprotnom vraća tekst nastalog problema.

Tab. 3.16. Metoda *Jira* klase za uređivanje tiketa.

| Metoda | Opis |
|---|---|
| *edit_Issue* | Uređuje zadana tiket polja i dodaje komentar na zadani tiket na Jira sustavu. |

**Dodavanje dodatnog ticketing sustava**

U slučaju dodavanja novog ticketing sustava uz Jira i ClickUp sustav, za potrebe uređivanja polja tiketa s novog ticketing sustava, prvo je potrebno implementirati funkcije potrebne za dohvaćanje i uređivanje podataka polja te za dodavanje komentara na tiket s novog ticketing sustava. Nakon toga potrebno je unutar *editTicket* (tablica 3.14.) metode *Adapter* klase implementirati uvjet koji je potrebno ostvariti kako bi se pozvale funkcije za uređivanje polja tiketa i dodavanje komentara na isti na novododanom ticketing sustavu. Po mogućnosti kao povratnu

vrijednost funkcija vratiti status 200 ako je uređivanje uspjelo te u suprotnom vratiti poruku nastalog problema.

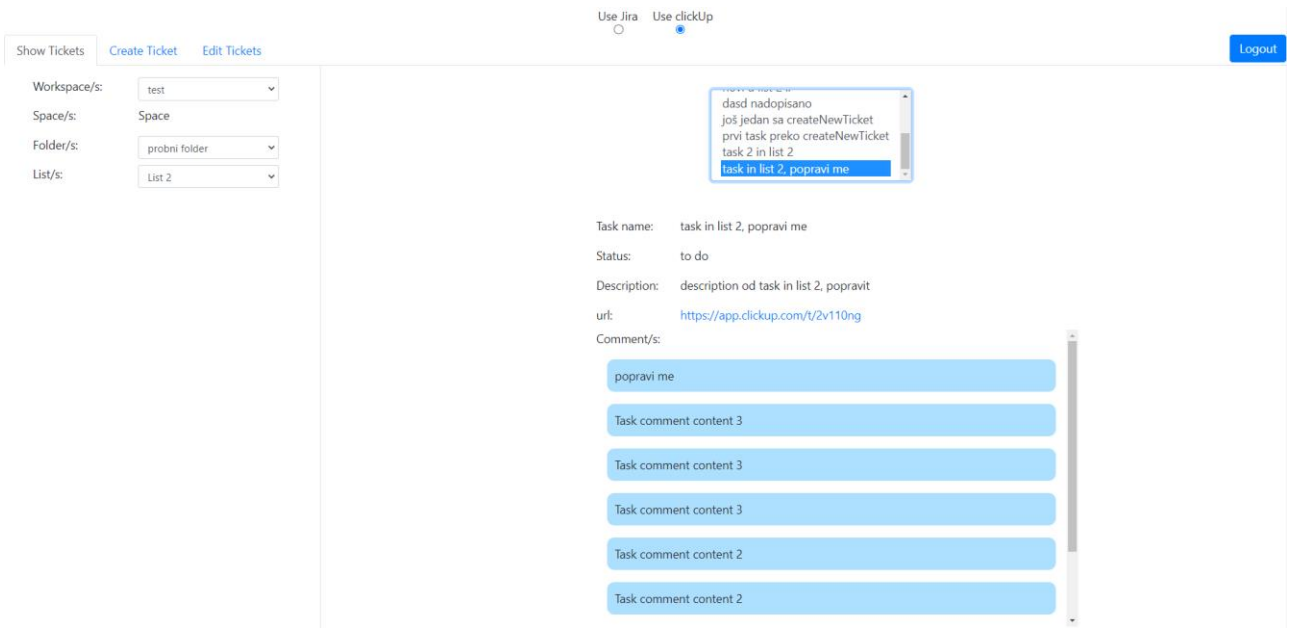
## 3.2. Korisničko sučelje za prijavu

U korisničkom sučelju za prijavu korisnik može unijeti podatke za prijavu na jedan ili oba ticketing sustava. Mogućnost komunikacije sustava s Jira ticketing sustavom korisnik ostvaruje popunjavanjem *Server link, E-mail* i *API token* polja. Mogućnost komunikacije sustava s ClickUp ticketing sustavom može biti ostvarena na dva načina. Prvi način je popunjavanjem *Client ID* i *Client Secret* polja gdje je nakon toga potrebno kliknuti na *Get ClickUp code* gumb (engl. *button*) koji preusmjerava korisnika na njegov ClickUp server da odabere *workspace* u kojem želi raditi. Nakon odabira *workspace*-a korisnik je vraćen na korisničko sučelje za prijavu gdje može nastaviti s prijavom. Drugi način je popunjavanje *Personal API token* polja, nakon čega nije potrebno koristiti *Client ID* i *Client Secret* polja te samim time ne treba koristiti *Get ClickUp code* gumb. (slika 3.16.)

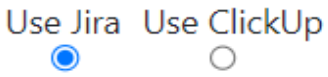

Sl. 3.16. Korisničko sučelje za prijavu: a) bez podataka; b) s unesenim podacima.

## 3.3. Korisničko sučelje za rad s tiketima

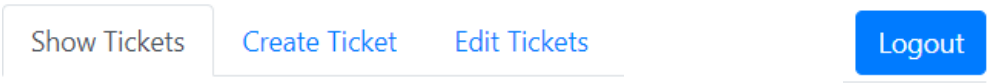
Nakon prijave korisnik dolazi na korisničko sučelje za rad s tiketima prikazano na slici 3.17. Korisnik u ovom sučelju može raditi s tiketima iz ticketing sustava za koje je u sučelju za prijavu predao podatke za prijavu. Na samom vrhu korisničkog sučelja nalazi se izbornik prikazan na slici 3.18. pomoću kojeg korisnik odabire ticketing sustav u kojem će raditi. Odmah ispod izbornika ticketing sustava nalazi se navigacijska traka koja sadrži izbore za tri moguća načina rada s tiketima: prikaz tiketa (engl. *show tickets*), stvaranje tiketa (engl. *create ticket*) i uređivanje tiketa (engl. *edit tickets*). Na traci se nalazi i gumb s nazivom *Logout* prikazan na slici 3.19. b) koji se koristi za odjavu sa sustava. Izbori za način rada s tiketima su prikazani na slici 3.19. a).




Sl. 3.17. Korisničko sučelje - prikaz tiketa



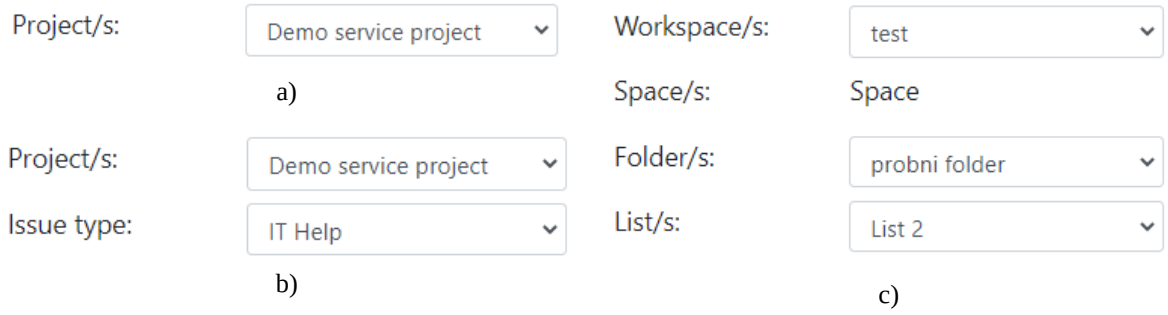

Sl. 3.18. Izbornik ticketing sustava.




a) b)

Sl. 3.19. Elementi izborne trake: a) izbori načina rada s tiketima; b) gumb za odjavu sa sustava.

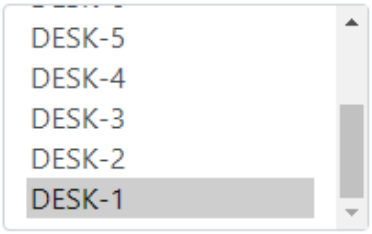
Na lijevom dijelu sučelja nalaze se nazivi za svaku hijerarhiju pojedinog ticketing sustava te pokraj svakog naziva njihov odgovarajući padajući izbornik (engl. *dropdown select*) što je i prikazano na slikama 3.20. a), b) i c).



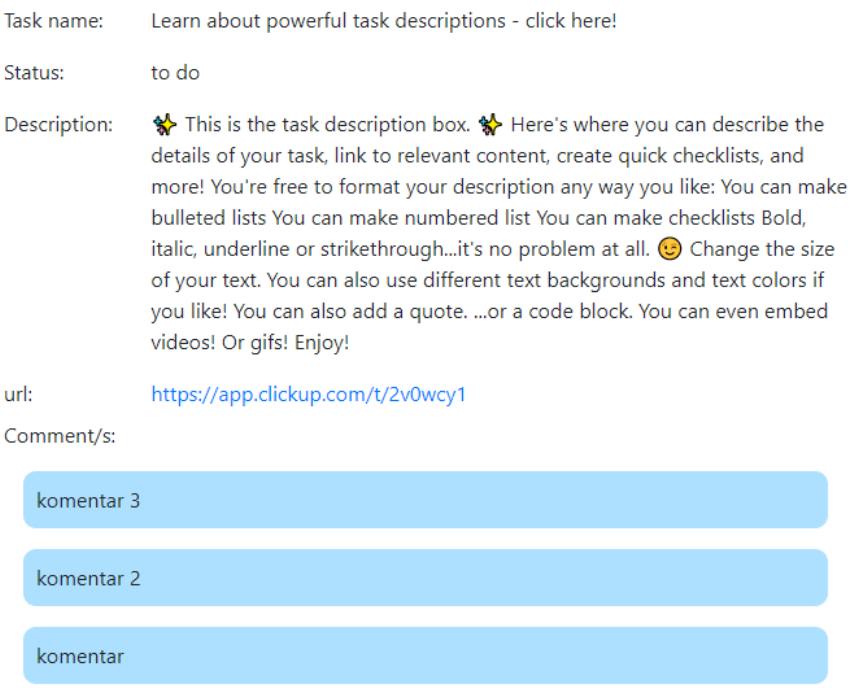

a) b) c)

Sl. 3.20. a) Hijerarhija sustava Jira za opcije prikaza i uređivanja tiketa, b) Hijerarhija sustava Jira za opciju stvaranje tiketa; c) Hijerarhija sustava ClickUp za sve opcije rada s tiketima

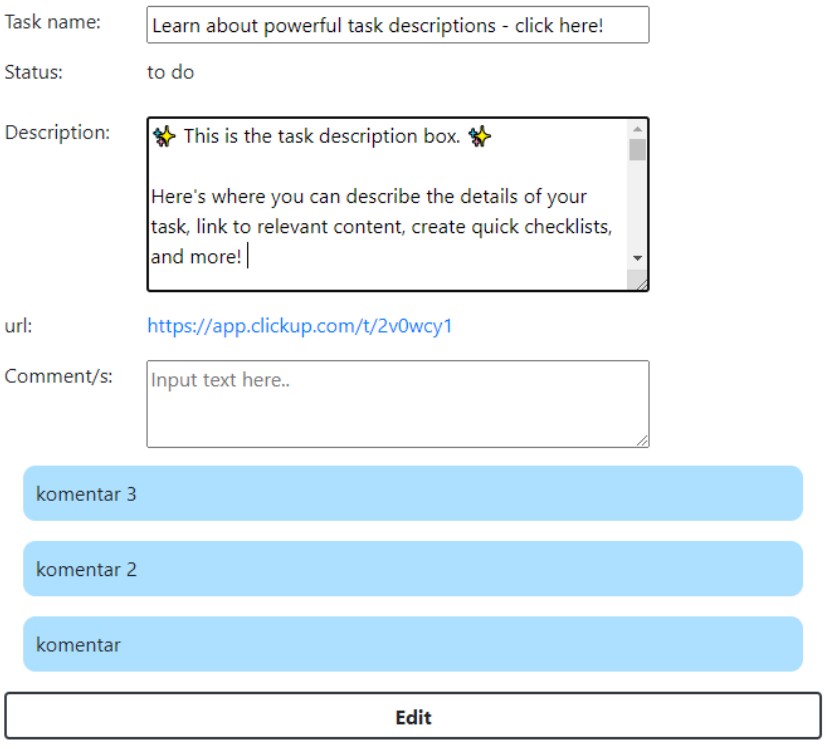
U sredini sučelja, ako je odabran prikaz tiketa ili uređivanje tiketa, redom od vrha prema dolje nalaze se elementi za odabir željenog tiketa i prikaz podataka odabranog tiketa (slike 3.21., 3.22. i 3.23.). Podatke postojećeg tiketa je moguće uređivati samo ako je odabrana opcija uređivanja tiketa (slika 3.23.).




Sl. 3.21. Lista ponuđenih tiketa za odabir



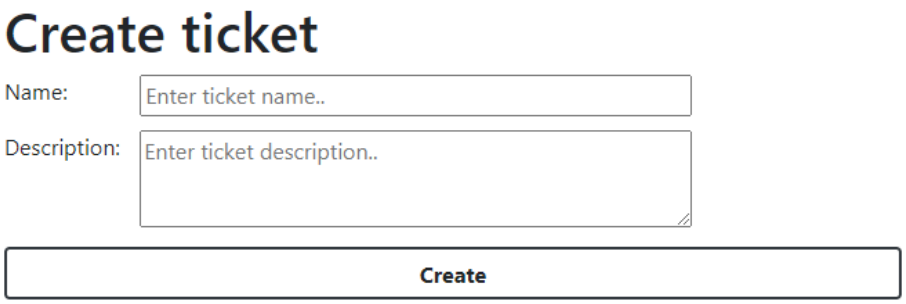

Sl. 3.22. Prikaz podataka odabranog tiketa - opcija prikaz tiketa.




Sl. 3.23. Prikaz podataka odabranog tiketa - opcija uređivanje tiketa.

Ako je odabran način rada za stvaranje tiketa, za stvaranje unutar ClickUp sustava hijerarhija ostaje ista kao kod prikaza tiketa i uređivanja tiketa (slika 3.20. c) dok je za stvaranje tiketa u Jiri dodan

padajući izbornik za tipove tiketa (slika 3.20. b)). U sredini sučelja više se ne nalazi element za odabir tiketa, nego samo forma za stvaranje tiketa (slika 3.24.). Nakon pritiska na gumb za odjavu korisnik je vraćen na korisničko sučelje za prijavu (slika 3.16.).




Sl. 3.24. Prikaz forme za stvaranje tiketa.

# 4. ZAKLJUČAK

Diplomski rad objašnjava što su to sustavi za praćenje tiketa, zašto su oni važni za razvoj i testiranje programske podrške i koje su prednosti njihovog svakodnevnog korištenja. U radu je navedeno pet primjera sustava koji su usredotočeni na praćenje tiketa ili uz svoju glavnu uslugu pružaju mogućnost praćenja tiketa. Navedeni sustavi su opisani s ciljem isticanja njihovih prednosti i nedostataka te je za svaki sustav navedeno kako bi se mogao koristiti ili zašto se ne preporučava njegovo korištenje u automobilskoj industriji.

U glavnom dijelu rada zadatak je bio osmisliti način implementacije integracije ticketing sustava Jira sa sustavom za provođenje udaljenog ispitivanja programske podrške u automobilskoj industriji. Osmišljen je način integracije dva ticketing sustava na jedan sustav za programsku podršku. Implementirana je *Adapter* singleton klasa koja omogućava, u budućnosti, lakšu integraciju dodatnih ticketing sustava sa sustavom za provođenje ispitivanja i njihovo istovremeno korištenje. Implementirano je i jednostavno korisničko sučelje za prijavu i za rad s tiketima sa željenog sustava kako bi se mogla predočiti komunikacija s ticketing sustavima korištenjem implementiranog rješenja.

Da bi se implementirano rješenje moglo integrirati i koristiti u postojećem sustavu za provođenje ispitivanja, potrebno je napraviti izmjene u *Adapter* klasi i u sustavu povezati postojeću prijavu korisnika s prijavom u rješenju. Kako bi se omogućilo da *Adapter* klasu koristi više korisnika istovremeno, bez da se podaci međusobno prepisuju sa svakom prijavom novog korisnika, potrebno postaviti atribut *Adapter* klase koji će biti lista u kojoj se za svakog korisnika na njegovu identifikacijsku vrijednost povezuje njegova instanca *Adapter* klase. Zatim se u konstruktoru *Adapter* klase, umjesto provjere da li je *__shared_instance* vrijednost jednaka početnoj, postavlja uvjet u kojem se provjerava postoji li za predani identifikator korisnika pripadajuća instanca klase. Također će biti potrebno *static* metodi *getInstance* kao parametar predati identifikator korisnika kako bi se kao povratna vrijednost mogla vratiti pripadajuća instanca za dobiveni identifikator. Za povezivanje korisničkih sučelja za prijavu najjednostavnije bi bilo da za svakog korisnika, u bazi podataka sustava za provođenje ispitivanja programske podrške ili općoj bazi podataka tvrtke, postoje zapisani potrebni podaci za prijavu na Jira i ClickUp sustav. Na taj način je moguće nakon prijave korisnika u sustav za provođenje ispitivanja programske podrške iz baze uzeti i iskoristiti te podatke za stvaranje instance *Adapter* klase. Za korištenje korisničkog sučelja za rad s tiketima na osnovnoj razini, prijedlog je implementacija sučelja za rad s tiketima u obliku posebnog prozora kojeg je moguće otvoriti unutar dijela sustava gdje korisnici imaju pregled testova i njihovih

informacija. Razlog takve implementacije je olakšan unos podataka prilikom stvaranja i uređivanja tiketa te olakšano uspoređivanje nastalog problema prilikom izvođenja testa s već postojećim prijavljenim problemima na ticketing sustavu.

Primjenom predloženog oblika rješenja uz navedene nadogradnje ostvarila bi se mogućnost dinamičnog korištenja više različitih ticketing sustava u jednom sustavu za provođenje udaljenog ispitivanja programske podrške u automobilskoj industriji.

# LITERATURA


[1] „The Economic Impacts of Inadequate Infrastructure for Software Testing, National institute of standards and technology, dostupno na: https://www.nist.gov/system/files/documents/director/planning/report02-3.pdf [pristupljeno: 14. rujan 2022.]

[2] M. Ortu, G. Destefanis, M. Kassab, and M. Marchesi, „Measuring and Understanding the Effectiveness of JIRA Developers Communities," May 2015, doi: 10.1109/WETSoM.2015.10. [pristupljeno: 14. rujan 2022.].

[3] „Issue tracking system," *Wikipedia*, Aug. 23, 2022, The Wikimedia Foundation, Inc, dostupno na : https://en.wikipedia.org/w/index.php?title=Issue_tracking_system&oldid=1106232167 [pristupljeno: 14. rujan 2022.].

[4] „GitHub Issues" *GitHub Docs*, GitHub, Inc, dostupno na: https://ghdocs-prod.azurewebsites.net/en/issues [pristupljeno: 14. rujan 2022.].

[5] „GitHub Issues · Project planning for developers" *GitHub*, GitHub, Inc, dostupno na: https://github.com/features/issues [pristupljeno: 14. rujan 2022.].

[6] „Syntax for issue forms" *GitHub Docs*, GitHub, Inc, dostupno na: https://ghdocs-prod.azurewebsites.net/en/communities/using-templates-to-encourage-useful-issues-and-pull-requests/syntax-for-issue-forms [pristupljeno: 14. rujan 2022.].

[7] „Download Axosoft Installed | Axosoft" *Axosoft.com*, Axosoft, LLC, DBA GitKraken, dostupno na: https://www.axosoft.com/downloads [pristupljeno: 14. rujan 2022.].

[8] „Axosoft Support Home | Axosoft Documentation" *support.axosoft.com*, Axosoft, LLC, DBA GitKraken, dostupno na: https://support.axosoft.com/ [pristupljeno: 14. rujan 2022.].

[9] „The G2 on Axosoft" *G2*, G2.com, Inc, dostupno na: https://www.g2.com/products/axosoft/reviews [pristupljeno: 14. rujan 2022.].

[10] „The G2 on Jitbit Helpdesk" *G2*, G2.com, Inc, dostupno na: https://www.g2.com/products/jitbit-helpdesk/reviews [pristupljeno: 14. rujan 2022.].

[11] „Support - Jitbit (authors of Jitbit Helpdesk)", Jitbit, dostupno na: https://www.jitbit.com/support/ [pristupljeno: 14. rujan 2022.].

[12] „Help desk software API", Jitbit, dostupno na: https://www.jitbit.com/helpdesk/helpdesk-api/#/ [pristupljeno: 14. rujan 2022.].

[13] „The G2 on Jitbit Helpdesk” *G2*, G2.com, Inc, dostupno na: https://www.g2.com/products/jitbit-helpdesk/reviews [pristupljeno: 14. rujan 2022.].

[14] „ClickUp Help”, ClickUp - Mango Technologies, Inc, dostupno na: https://help.clickup.com/hc/en-us [pristupljeno: 14. rujan 2022.].

[15] „The G2 on ClickUp” *G2*, G2.com, Inc, dostupno na: https://www.g2.com/products/clickup/reviews [pristupljeno: 14. rujan 2022.].

[16] „Jira Software Cloud support | Jira Software Cloud” *Atlassian Support*, Atlassian, Inc, dostupno na: https://support.atlassian.com/jira-software-cloud/ [pristupljeno: 14. rujan 2022.].

[17] „Jira Software Data Center and Server support | Jira Software - latest” *Atlassian Support*, Atlassian, Inc, dostupno na: https://support.atlassian.com/jira-software-server/ [pristupljeno: 14. rujan 2022.].

[18] Atlassian, „Jira Pricing - Monthly and Annual Subscription Cost per User” *Atlassian*, Atlassian, Inc, dostupno na: https://www.atlassian.com/software/jira/pricing [pristupljeno: 14. rujan 2022.].

[19] „The web framework for perfectionists with deadlines | Django” , Django Software Foundation, dostupno na: https://www.djangoproject.com/ [pristupljeno: 14. rujan 2022.].

[20] „ClickUp™ | API Docs” , Mango Technologies, Inc, dostupno na: https://clickup.com/api [pristupljeno: 14. rujan 2022.].

[21] „jira 3.4.1.dev23+g7846ac3 documentation” , Atlassian, Inc, dostupno na: https://jira.readthedocs.io/ [pristupljeno: 14. rujan 2022.].

[22] M. O. contributors Jacob Thornton, and Bootstrap, „Bootstrap” , dostupno na: https://getbootstrap.com/ [pristupljeno: 14. rujan 2022.].

[23] J. Petrucciani, „pyclickup: A python wrapper for the ClickUp API”, dostupno na: https://github.com/jpetrucciani/pyclickup.git [pristupljeno: 14. rujan 2022.].

# SAŽETAK

Zadatak rada je integracija ticketing sustava Jira sa sustavom za provođenje udaljenog ispitivanja programske podrške u automobilskoj industriji. Integracija ticketing sustava u postojeći sustava za ispitivanje programske podrške je učestala pojava kod tvrtki čija je glavna usluga testiranje programske podrške. Razlog integracije ticketing sustava je dobitak na učinkovitosti te poboljšana mogućnost praćenja vremena koje pojedini zaposlenik provede postavljajući okolinu potrebnu za izvršavanje nekog testa, koliko vremena se izvodi test te koliko je vremena provedeno za stvaranje tiketa ako je rezultat testa bio pad ili preskakanje testa. Budući da je moguće da neka tvrtka koristi više od jednog ticketing sustava, u radu je opisano pet ticketing sutava te su navedene njihove prednosti, nedostatci i mogući načini korištenja istih uz sustave za udaljeno ispitivanje programske podrške. U glavnom dijelu rada opisana je implementacija adaptera koji je osmišljen kao rješenje za integraciju većeg broja različitih ticketing sustava s jednim sustavom za ispitivanje programske podrške. Uz opis implementacije integracije Jira i ClickUp sustava preko adaptera, opisan je i način nadogradnje adaptera u slučaju da se uz Jira i ClickUp sustave želi dodati komunikacija sa još ticketing sustava. Također je prikazan i opisan sadržaj implementiranog korisničkog sučelja koje služi kao primjer onoga što bi sučelje za komunikaciju s ticketing sustavima trebalo sadržavati i kako ostvariti njegovu najbolju preglednost.

**Ključne riječi:** adapter, programska podrška, testiranje, ticketing sustav

## ABSTRACT

**Title:** Ticketing System for Automotive Software Testing

The aim of this paper is to integrate the Jira ticketing system with a remote testing system used in the automotive industry. The integration of the ticketing system into the existing software testing system is a frequent occurrence in companies whose main service is software testing. The reason for this integration is to gain efficiency and the improved ability of monitoring exactly how much time an individual employee spends setting up the environment necessary for test execution, as well as how much time the test is being performed and the time that was spent creating a ticket if the result of the test was a fail or a skip. Since it is possible for a company to use more than one ticketing system, the paper describes five ticketing systems, listing their advantages, disadvantages, and possible ways of using them in addition to systems for remote software testing. The main part of the paper describes the implementation of the adapter, which was designed as a solution for integrating a number of different ticketing systems with one system for software testing. In addition to describing the implementation of integration of Jira and ClickUp systems via the adapter, this paper also presents a description on how to upgrade the adapter if, in addition to Jira and ClickUp systems, a communication with an aditional ticketing systems is wanted. Furthermore, the content of the implemented user interface is also shown and described, which serves as an example of what the interface for communication with ticketing systems should contain and how to achieve the best interface transparency.



**Key words:** adapter, software, testing, ticketing system

# ŽIVOTOPIS

Franjo Ferić rođen je 23. veljače 1999. godine u Slavonskom Brodu. Osnovnu školu pohađao je od 2005. do 2013. godine u OŠ „Dr. Stjepan Ilijašević“, Oriovac, PŠ Slavonski Kobaš. Gimnaziju „ Matija Mesić“, opći smjer, u Slavonskom Brod pohađao je do 2017. godine kada je maturirao. Odmah nakon mature upisuje i od 2017. do 2020. godine pohađa preddiplomski sveučilišni studij Računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku te stječe zvanje sveučilišnog prvostupnika (lat. *baccalaureus*) inženjera računarstva. Odmah 2020. godine na istom fakultetu upisuje diplomski studij računarstva, smjer robotika i umjetna inteligencija.

_____

Potpis autora

# IZJAVA O LEKTURI

Franjo Ferić

***Upravljanje korisničkim upitima u ispitivanju programske podrške u automobilskoj industriji***

**Rad je lektoriran prema pravilima hrvatskog jezika**

Datum:

26. rujna 2022.

Vizjak

Ana Vizjak, prof.