Sadržaj

Spisak slika

Skraćenice

API – *Application programming interface*, Interfejs za programiranje aplikacija

URL – *Application programming interface*, Interfejs za programiranje aplikacija

CLI – *Application programming interface*, Interfejs za programiranje aplikacija

YAML – *Application programming interface*, Interfejs za programiranje aplikacija

1. Uvod

U današnje vreme

2. Teorijske osnove

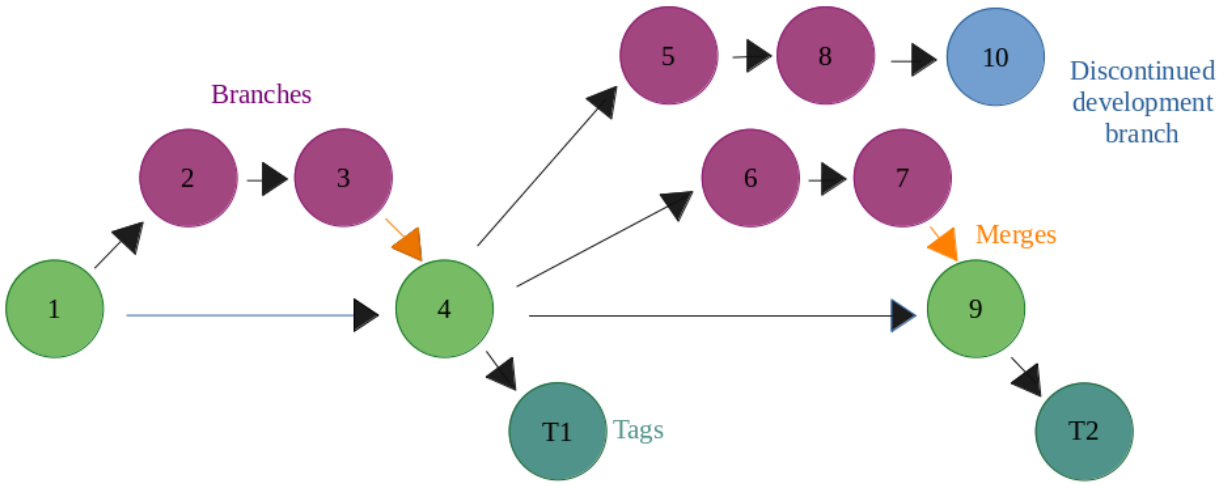
U ovom poglavlju će biti ukratko objašnjeni svi koncepti i alati koji su bili potrebni za izradu projekta o kojem ovaj rad govori. NABROJ

2.1 Open source razvoj

2.2 Sistem za praćenje verzija datoteka

Sistem za praćenje verzija datoteka (eng. *version control system*) je sistem zadužen za praćenje istorije promena u datotekama. Promena koja se pamti se najčešće naziva revizija. Sve promene se uglavnom čuvaju na centralizovanom sistemu za skladištenje podataka i svi korisnici sa potrebnim privilegijama mogu da mu pristupaju, preuzmu najnoviju verziju datoteke ili grupe datoteka, izmene ih i zatim objave novu verziju koja se od tog momenta smatra najnovijom.

Pored najnovije verzije, čuva se i celokupna istorija promena i moguće je vratiti datoteke u bilo koju od prethodnih revizija. U poglegu softvera (eng. *software*), ovo nam omogućava da vratimo u upotrebu bilo koju prethodnu verziju, u slučaju da dođe do greške u nekoj od novijih verzija. Istorija se čuvaju u strukturi tipa grafa:



Slika 2.1: Sistem za praćenje verzija datoteka

Graf se najčešće sastoji od glavne grane(eng. *branch*), na kojoj se nalaze glavne izmene, i pomoćnih grana na kojima se nalaze izmene za koje još nije odlučeno da li će pripadati glavnoj grani. Ako sve izmene na pomoćnim granama budu odgovarajuće, one se spajaju (eng. *merge*) sa glavnom granom i postaju deo njenog sadržaja. U slučaju praćenja verzija softvera, ovakve grane najčešće predstavljaju nove funkcionalnosti (eng. *feature*) koje se dodaju na glavnu granu projekta ili ispravljanje greške (eng. *bugfix*) koja je nastala u nekoj od prethodnih verzija.

Važne izmene se mogu dodatno naglasiti dodavanjem oznake (eng. *tag*). One uglavnom predstavljaju objavljivanje nove verzije softvera (eng. *release*), dodavanje nove funkcionalnosti ili ispravku greške.

2.3 Git

Git [1] je besplatan softver otvorenog koda (eng. *open-source*) za praćenje verzija datoteka u distribuiranom okruženju. Primarno se koristi za razvoj softvera, ali se može koristiti i za praćenje bilo kojih drugih datoteka. Razvoj je započeo 2005. godine kreator Linux-a, Linus Torvalds, za potrebe razvoja jezgra Linux operativnog sistema. Distribuiran je pod GNU General Public License Version 2 licencom. Primarno je napravljen za Linux operativni sistem, ali je naknadno prilagođen i za macOS, BSD, Solaris i Windows.

Za razliku od drugih sistema za praćenje verzija datoteka u distribuiranom okruženju, Git nije softver klijent-server arhitekture. Svaki Git repozitorijum na svakom računaru čuva kompletnu istoriju promena i verzija i nije zavisan od pristupa mreži ili centralizovanom serveru. Kao takav, Git je osnova za razvoj mnogih drugih nezavisnih alata za praćenje verzija datoteka.

2.4 GitHub

GitHub [2] je servis otvorenog koda za hostovanje (eng. *hosting*) repozitorijuma sa različitim datotekama, najčešće izvornim kodovima za razna softverska rešenja. Za praćenje verzija datoteka, GitHub koristi Git. GitHub služi kao centralizovano mesto na kojem se čuvaju i razmenjuju sve izmene koje su sačuvane upotrebom Git alata.

U ovom trenutku, GitHub se smatra najpopularnijim alatom ovog tipa i trenutno broji preko 40 miliona korisnika. Stvari koje ovaj servis omogućava su:

1. Upravljanje projektima
2. Upravljanje timom
3. Pregled i komentarisanje koda
4. Integracija
5. Paketi
6. Bezbednost
7. Hostovanje
8. Akcije

Moguće mu je pristupiti preko CLI verzije, desktop verzije, ekstenzije za Visual Studio program za uređivanje datoteka i mobilne aplikacije.

2.4.1 Upravljanje projektima

Upravljanje projektom je proces koji je potreban u bilo kojem većem programerskom projektu. Menadžeri projekta su zaduženi da prate razvoj projekta i upućuju programere u razvoj narednih delova projekata, kao i da upravljaju raspoređivanjem vremena rada na različitim zadacima koje projekat iziskuje.

GitHub nudi mogućnost koridinacije projektom, praćenje razvoja i menjanje statusa projekta na jednom mestu. Kordinacija započinje kreiranjem zadataka (eng. *tasks*) koji traže izmene koda ili unapređenja projekta. Zadaci mogu biti u obliku greške/pitanja (eng. *issue*), komentara (eng. *comment*) ili pull-request-a, što podrazumeva kreiranje celine koja je spremna za spajanje sa glavnom granom na projektu. Zadacima se dodeljuju osobe koje su zadužene za njih. Time će oni

dobijati obaveštenja kada dođe do bilo koje izmene u kodu vezane za njihov deo posla. Zadacima je moguće dodeliti i prioritet i oznake, što dodatno olakšava proces upravljanja projektom.

Sve ove informacije moguće je pratiti i menjati u formi tabele (eng. *board*) ili u repozitorijumu na kojem se projekat nalazi. Svaki zadatak ima unikatan URL na kojem se mogu videti sve potrebne informacije vezane za njega. Čuva se i istorija svih zadataka i izmena na projektu, kojima se stvara uvid u proces razvoja projekta. Po završetku svih zadataka, urpavljanje projektom se završava i projekat se označava kao završen.

2.4.2 Upravljanje timom

Kako u timu imamo različite uloge, koje imaju različite permisije, GitHub nudi mogućnost hijerarhijskog organizovanja učesnika na projektu gde se svakoj grupi učesnika ili pojedincu dodeljuju potrebna prava.

Takođe, moguće je dodati u repozitorijum dokumente koji se tiču projekta, kao što su kodeks ponašanja (eng. *code of conduct*) i licenca (eng. *license*). Ove dokumente nije potrebno svaki put pisati iznova, već postoje unapred predefinisane verzije često korišćenih kodeksa ponašanja i licenci.

2.4.3 Pregled i komentarisanje koda

Komunikacija između učesnika na projektu doprinosi kvalitetu izrade samog projekta. Pored podele zadataka i raspoređivanja vremena za svaki zadatak, javlja se potreba i za diskutovanjem koda koji se dodaje u obliku pull-request-a.

Pri kreiranju pull-request-a, GitHub nudi prikaz svih izmena u odnosu na granu sa koje su izmene krenule i na koju bi se dodale ako je sve u redu. Taj prikaz je dostupan svima koji učestvuju u izradi projekta. Moguće je, dodatno, zamoliti nekoga od učesnika da pregleda izmene pre njihovog spajanja sa odabranom granom (eng. *requests a review*). Na svakoj od izmena, moguće je dodati poruku, koja otvara mogućnost diskusije sa ostalim učesnicima projekta. Po zavšetku, osoba koja je zadužena da pregleda izmene može da odobri ili dobije pull-request, uz mogućnost traženja dodatnih izmena pre odobrenja.

Takođe, pri kreiranju pull-request-a, GitHub automatski pregleda kod u potrazi za bezbednosnim propustima ili konfliktima sa granom sa kojom se trenutna grana spaja. Ako ih ima, GitHub obaveštava korisnika i traži izmenu pre odobrenja i spajanja koda.

2.4.4 Integracija

Pored samog pisanja aplikacije, često se javlja potreba za testiranjem i pracenjem (eng. *monitoring*) aplikacije, automatizovanim postavljanjem u produkciju (eng. *deployment*), kontinualnom integracijom (eng. *continuous integration*), proverom kvaliteta koda (eng. *code quality*) i drugim pomocnim aktivnostima u procesu razvoja softver-a.

Pošto GitHub nije u mogućnosti da ponudi sve te aktivnosti, omogućio je proces integracije sa servisima koji ih nude. Svi ovi servisi mogu se pronaći u GitHub prodavnici (eng. *marketplace*) [3]. Jedna od novina u GitHub prodavnici su GitHub akcije koje će biti jedna od centralnih tema ovog rada, a bice opisane u narednoj sekciji.

2.4.5 Paketi

Alati poput NPM-a, Docker-a, Maven-a, Gradle-a i drugih omogućavaju kreiranje artifakta koji sadrže sve potrebne alate za pokretanje nekog softverskog rešenja. Većina ovakvih servisa ima mogućnost čuvanja artifakta na njihovim repozitorijumima.

Zbog jednostavnosti korišćenja, GitHub je kreirao alat koji objedinjuje mnoge ovakve repozitorijume u jedan repozitorijum - GitHub Packages.

2.4.6 Bezbednost

Bezbednost u softverskim rešenjima igra veoma bitnu ulogu i najčešće zavisi od mnogo faktora: programera, ljudi koji održavaju kod (eng. *maintainers*), istraživača, timova specijalizovanih za bezbednost i drugih. Da bi se postigao što veći stepen bezbednosti, potrebno je da što više različitih strana učestvuje u testiranju koda na bezbednosne propuste (eng. *vulnerabilities*).

Neke od mogućnosti koje GitHub nudi za poboljšanje bezbednosti koda koji se nalazi na njemu su:

- **Automatsko skeniranje koda** - pri dodavanju koda na repozitorijum, pokreće se automatsko skeniranje koda na poznate bezbednosne propuste i obaveštava se korisnika ako se pronađe neki od njih
- **Definisanje bezbednosnog procesa rada projekata otvorenog koda** - omogućava definisanje bezbednosnih polisa u okviru *SECURITY.MD* datoteke u repozitorijumu u kojoj se navode sve potrebne informacije o tome koje bezbednosne probleme i na koji način treba da prijave korisnici koji ih pronađu

- **Diskutovanje o uticaju bezbednosnih propusta** - kada neki od korisnika prijavi bezbednosni propust ili grešku u kodu, GitHub nudi prostor za komentarisanje i razmenu mišljenja vezanih za postavljeni problem
- **Automatsko skeniranje zavisnosti** - slično skeniranju koda, pokreće se automatsko skeniranje zavisnosti nekog projekta - odnosno svih drugih softverskih rešenja od kojih projekat zavisi. Ako neko od njih ima bezbednosni propust, korisnik se obaveštava o tome. Ovakvi problemi često budu otklonjeni u narednoj verziji rešenja od kojeg projekat zavisi, pa GitHub često nudi promenu verzije kao rešenje ovakvog problema. Ovaj proces se nastavlja za vreme životnog ciklusa projekta, i ako se u bilo kojem momentu pronađe bezbednosni propust, automatski se kreira pull-request sa ispravkom
- **Pretraga kredencijala** - korisnicima se može desiti da slučajno upišu kredencijale za neki od eksternih servisa u kod projekta. Takvi kredencijali imaju specifičan oblik koji GitHub prepoznaje za preko 24 servisa i obaveštava korisnika ako ih pronađe

2.4.7 Hostovanje sajtova

Pored hostovanja repozitorijuma, što je glavna uloga GitHub-a, postoji i mogućnost hostovanja sajta pod GitHub domenom. Potrebno je napraviti repozitorijum sa nazivom *username.github.io* i svi dokumenti u okviru njega će biti statički hostovani pod *username.github.io* domenom.

2.4.8 Akcije

Kao što vidimo iz prethodnih sekcija, GitHub nudi mnoge mogućnosti pored njegove osnovne namene - praćenja verzija datoteka. Ipak, ako i te mogućnosti nisu dovoljne, postoji alat - GitHub Actions koji nudi pisanje tokova posla (eng. *workflow*) i reagovanje na mnoge događaje, tako da korisnik može, u specifičnom formatu, da definiše sve potrebne reakcije na događaje i pokrije slučajeve koje ostali GitHub alati ne pokrivaju. Ove akcije najčešće podrazumevaju automatizaciju, podešavanje i izvršavanje delova softverskih rešenja nakon nekog specifičng događaja.

2.4.8.1 Događaji

Neki od događaja na koje je moguće reagovati su:

- `create` - pokreće se izvršavanje svaki put kada se kreira nova granu ili oznaka
- `delete` - pokreće se izvršavanje svaki put kada se obriše granu ili oznaka

- `fork` - pokreće se izvršavanje pri račvanju novog projekta iz trenutnog projekta
- `gollum` - pokreće se izvršavanje svaki put kada se kreira ili izmeni Wiki stranica
- `issues` - pokreće se izvršavanje svaki put kada se desi issue događaj. što podrazumeva otvaranje teme sa pitanjem, njenu izmenu, zatvaranje, brisanje, označavanje, itd.
- `issue_comment` - pokreće se izvršavanje svaki put kada se doda, izmeni ili obriše komentar na otvorenom pitanju
- `label` - pokreće se izvršavanje svaki put kada se doda, izmeni ili obriše oznaka
- `milestone` - pokreće se izvršavanje svaki put kada se doda, izmeni ili obriše prekretnica u projektu
- `project` - pokreće se izvršavanje svaki put kada se desi project događaj, što podrazumeva kreiranje, izmenu, zatvaranje ili brisanje projekta
- `public` - pokreće se izvršavanje svaki put kada privatni repozitorijum postane javni
- `pull_request` - pokreće se izvršavanje svaki put kada se desi pull_request događaj. što podrazumeva otvaranje pull-request-a, izmenu, zatvaranje, brisanje, dodeljivanje, označavanje, sinhronizaciju, traženje pregledanja, itd.
- `pull_request_review` - pokreće se izvršavanje kada se izvrši, izmeni ili obriše pregledanje pull-request-a
- `pull_request_review_comment` - pokreće se izvršavanje kada se kreira, izmeni ili obriše komentar na pregledanju pull-request-a
- `push` - pokreće se izvršavanje svaki put kada se doda novi kod na granu
- `registry_package` - pokreće se izvršavanje svaki put kada se objavi ili izmeni paket
- `release` - pokreće se izvršavanje svaki put kada se desi release događaj, što podrazumeva kreiranje, objavljivanje, izmenu ili brisanje nove verzije projekta

Takođe, moguće je zakazati izvršavanje neke akcije navođenjem tačnog datuma i vremena izvršavanja ili zakazivanje izvršavanja na primer svake nedelje u određeno vreme. Za opisivanje vremena koristi se softverski alat Cron, koji je zadužen za planiranje i zakazivanje poslova na osnovu vremena (eng. *time-based job scheduler*) na Unix operativnim sistemima.

Tokovi poslova se opisuju u YAML formatu. Događaji se opisuju u okviru *on* sekcije, uz dodatak *type* i *branch* podsekcija ako ima potrebe. Na primer:

```
on:
    pull_request:
        branches:
        - master
    release:
        types:
        - created
```

2.4.8.2 Poslovi

Nakon prepoznavanja događaja, pokreće se izvaršavanje jednog ili više poslova (eng. *jobs*), gde se svaki od poslova može sastojati od jednog ili više koraka (eng. *step*).

Za poslove se definiše na kojem operativnom sistemu treba da se izvrše. U ponudi su Windows Server 2019, Ubuntu 20.04, Ubuntu 18.04, Ubuntu 16.04 i macOS Catalina 10.15. Opciono se dodaje naziv posla, dodatni uslov pod kojim se izvršava, promenjive koje se preuimaju iz okruženja (eng. *environment variables*), maksimalno vreme izvršavanja (eng. *timeout*) i drugi parametri.

2.4.8.3 Koraci

Koraci su zaduženi za pokretanje komandi, pokretanje zadataka za podešavanje (eng. *setup tasks*) ili pokretanje GitHub Akcija iz trenutnog repozitorijuma, nekog javnog repozitorijuma ili sa Docker registra (eng. *registry*). Korak ne mora da ima akciju u sebi, ali svaka akcija koja se pokreće mora da se pokreće kao izolovani korak. Svaki korak se pokreće kao zaseban proces u okruženju pokretača (eng. *runner*) i ima pristup svom okruženju i sistemu datoteka na kojem se pokreće. Ako ima potrebe, mogu postojati zasebni koraci koji se pokreću na početku i kraju posla, koji služe da pripreme okruženje pre početka izvršavanja i odrade završne aktivnosti na kraju izvršavanja.

Kao i posao, korak sadrži svoj unikatni id, naziv, uslov izvršavanja ako postoji, ulazne parametre, promenjive koje se preuzimaju iz okruženja, maksimalno vreme izvršavanja itd. Aktivnost koja se izvršava u toku koraka može se definisati na dva načina, upotrebom jednom od sledećih rezervisanih reči:

- `uses` - u slučaju kada se koristi postojeća akcija, na ovom mestu se navodi putanja do akcije

(u trenutnom repozitorijumu, nekom javnom repozitorijumu ili na Docker registru) i verzija akcije u oblliku: `path/name@version`.

- `run` - u slučaju kada se ne koristi postojeća akcija, već se definiše niz komandi u interfejsu komandne linije operativnog sistema. Moguće je imati više `run` komandi u okviru jednog koraka. Tada se svaka komanda pokreće kao odvojeni proces u jezgru operativnog sistema. Ako se pod jednom `run` sekcijom pokrene komanda u više linija, tada se sve komande pokreću u okviru jednog procesa. Komande se mogu pokretati u jednom od sledećih jezika komandne linije: bash, pwsh, python, sh, cmd, powershell.

2.4.8.4 Kontekst okruženja

Na osnovu konteksta okruženja, moguće je odlučivati o tome da li će se neki korak ili posao izvršiti. Za to postoji rezervisana reč `if`. Kontekst se nalazi u nekoliko objekata koji su nosioci različitih informacija i konteksta različitih delova toka posla. Neki od najčešće korišćenih su:

- `github` - informacije o toku posla

  Primer: naziv akcije, putanja do akcije, id trenutnog posla koji se izvršava, koji korisnik je zaslužan za pokretanje akcije, koji korisnik je vlasnik repozitorijuma na kojem je pokrenuta akcija, grana ili oznaka na kojo se desilo pokretanje akcije, itd.

- `env` - sadrži promenjive koje se preuzimaju iz okruženja podešene u okviru toka posla, posla ili koraka.

- `job` - informacije o poslu koji se trenutno izvršava

  Primer: status posla, kontejner u kojem se posao izvršava, servisi koji su potrebni za izvršavanje posla, itd.

- `steps` - informacije o koraku koji se trenutno izvršava

  Primer: status koraka, vrednost izlaznih promenjivih, itd.

- `runner` - informacije o pokretaču trenutnog posla

  Primer: operativni sistem, putanja privremenog (eng. *temporary*) direktorijuma, itd.

- `secrets` - sadrži promenjive koje se čuvaju kao tajne

- `needs` - dozvoljava pristup promenjivama koje su izlazne promenjive prethodnih poslova

2.4.8.5 Pisanje akcija

Neke od unapred definisanih akcija nudi GitHub, dok su druge napisane od strane zajednice koja ih koristi. Svako može da napiše akciju koji koristi za sebe ili da je podeli sa drugim korisnicima. Akcija se može pokretati direktno u okviru virtuelne mašine (eng. *virtual machine*) ili u okviru Docker kontejnera. Mogu se definisati ulazni i izlazni parametri i promenjive koje se preuimaju iz okruženja.

Pri pisanju akcije potrebno je kreirati *action.yml* ili *action.yaml* dotatoku koja definiše sve informacije potrebne za izvršavanje akcije. Akcija može biti u obliku Docker kontejnera, ili opisana u JavaScript programskom jeziku. Takođe, postoje kompozitne akcije (eng. *composite run steps action*) koje omogućavaju kombinovanje nekoliko koraka u jednu akciju.

- **Docker akcije** - Docker kontejneri upakuju celokupno okruženje sa kodom GitHub akcije. Ovo čini dosta pouzdano rešenje, pozto korisnik ne mora da brine o verzijama i alatima. U kontejneru se instalira izabrana verzija operativnog sistema i svi potrebni alati, što ovakav način pisanja akcija čini pogodnim za sisteme koji moraju da se prilagode specifičnom okruženju. Mana ovog pristupa je to što je potrebno dosta vremena da se kreira i pokrene kontejner pa su ove akcije sporije od JavaScript akcija.
- **JavaScript akcije** - Ove akcije se pokreću direktno na računarima pokretačima, što pojednostavljuje pisanje koda akcija i znatno ubrzava njihovo izvršavanje. Da bi se mogle izvršavati na bilo kojem operativnom sistemu, ove akcije moraju biti napisane u čistom JavaScript-u.

Akcije napisane od strane pojedinaca mogu se objaviti u GitHub prodavnici i time postati dostupne svima za korišćenje. Da bi akcija bila spremna za objavljivanje, potrebno je da se ceo kod akcije i datoteke potrebne za njeno izvršavanje nalaze u jednom repozitorijumu. Taj repozitorijum ne sme ništa drugo da sadrži. Takođe, repozitorijum mora biti javan, da bi drugi korisnici mogli da mu pristupe. Potrebno je dodati oznaku na repozitorijum, koja predstavlja verziju akcije u GitHub prodavnici i zatim je objaviti.

2.4.8.6 Hostovanje pokretača

GitHub nudi mašina koje hostuju pokretače GitHub Akcija, ali i daje mogućnost hostovanja akcija na korisničkom hardveru (eng. *hardware*). Hostovanje na korisničkim mašinama daje veću slobodu korisnicima u izboru samih hardverskih komponenti, operativnog sistema i alata koji se instaliraju na tim mašinama. Mogu biti u obliku fizičkih mašina, virtuelnih mašina, kontejnera, ili računari u oblaku (eng. *cloud*).

Korisničke mašine se povezuju sa GitHub-om preko aplikacije koju GitHub nudi. Ako se ne pokrene ni jedna akcija u roku od 30 dana, prekida se veza između korisničkog računara i GitHub-a.

Problem pri hostovanju na korisničkim mašinama može biti to što je korisnik zadužen za ažuriranje verzije operativnog sistema i svih softverskih alata na njemu, dok na GitHub-ovim mašinama to radi sam GitHub. Takođe, Github pokreće novu, čistu instancu za svaki posao koji se izvršava, dok korisnik to ne mora da radi.

2.5 Docker? opet ga svuda ima

2.6 Python

Python [4] je programski jezik opšte namene, spada u grupu programskih jezika visokog nivoa i interpretira se. Dizajniran je tako da bude izrazito čitljiv, što se postiže velikom upotrebom belina (eng. *whitespace character*). Pogodan je za razvoj različitih projekata pošto podržava više programskih paradigmi:

1. Proceduralnu
2. Objektno orijentisanu
3. Funkcionalnu
4. Strukturalnu paradigmu

Takođe, prilagođen je za korišćenje u radličitim okruženjima na različitim operativnim sistemima.

Python se prvi put spominje krajem 1980-tih kao naslednik ABC programskih jezika. Dizajnirao ga je Guido van Rossum i objavio prvu verziju 1991. godine. Python 2.0, koji je izdat 2000. godine je znatno unapređenje prve verzije, koje se do skoro koristilo. Poslednja velika revizija urađena je 2008. godine kada je nastao Python 3.0. Zbog toga što Python 3 nije u potpunosti kompatibilan sa prethodnim verzijama, podrška za Python prestala je u januru 2020. godine.

Umesto da poseduje svu svoju funkcionalnost u jezgru OS-a. Python je dizajniran tako da bude veoma proširiv, što ga je učinilo veoma pogodnim za dodavanje programabilnih interfejsa postojećim aplikacijama.

2.7 Python paketi, PyPI

Python paketi su imenski prostori (eng. *namespaces*) koji u sebi sadrže druge pakete i module - delove softvera koji obavljaju konkretnu funkcionalnost. Svaki paket je direktorijum koji u sebi mora imati *__init__.py* datoteku. Ova datoteka može biti prazna, što označava da je trenutni direktorijum Python paket i da može biti korišćen kao zaseban modul. Inače, *__init__.py* datoteka čuva kod koji služi za inicijaliciju tog paketa.

Da bi se koristile funkcionalnosti koje neki paket nudi, potrebno ga je uvezati (eng. *import*) u projekat korišćenjem rezervisane reči `import`. Da bi se uvezao pojedinačni modul iz nekog paketa, treba naznačiti koji modul se iz kojeg paketa uvezuje korišćenjem sledećeg Python koda:

- `from <package> import <module>`

2.7.1 Distribuiranje Python paketa

Glavna uloga Python paketa je distribuiranje biblioteka za Python programski jezik. Paket može da kreira bilo ko i da ga potom podeli sa ostatkom Python zajednice. Da bi paket bio dostupan, potrebno je kreirati arhivu sa svim potrebnim konfiguracijama za njega i objaviti je na nekom repozitorijumu za Python pakete.

Repozitorijumi su softverski alati za čuvanje i distribuiranje paketa. Postoje javni repozitorijumi, koji omogućavaju razmenu paketa sa bilo kim, i privatni, koji omogućavaju razmenu paketa u ograničenoj grupi korisnika koji imaju specijalne privilegije.

Najčešće korišćen javni repozitorijum je PyPI [5] - The Python Package Index.

2.8 Poetry

Poetry [6] je alat otvorenog koda za kreiranje Python paketa i upravljanje paketima koje Python projekat koristi(eng. *dependency management*). Pomaže korisnicima tako što umesto njih instalira i menja verzije bibliotekama od kojih projekat zavisi, a koje korisnik mora pretkodno da specificira. Poetry je takođe znatno olakšao kreiranje, pakovanje i objavljivanje paketa na PyPI repozitorijumu i time omogućio lakše deljenje Python projekata sa drugim korisnicima Python programskog jezika.

Prva verzija alata objavljena je 28.02.2018. godine. Verzije programskog jezika Python koje su podržane su 2.7 i 3.4+. Ideja je da Poetry radi podjednako dobro na različitim platformama, između ostalog na Linux-u, Windows-u i OSX-u.

2.8.1 Podešavanje Poetry-ja

Za razliku od drugih Python alata za upravljanje paketima od kojih projekat zavisi, umesto pip-a - instalera za Python pakete, Poetry koristi svoj specifičan način za instalaciju. Instalator doda Poetry u korisnički direktorijum, tako da može da se koristi sa bilo kojom verzijom Python-a. Omogućena je i instalacija Poetry-ja korišćenjem pip-a, ali se ne proporučuje iz dva razloga:

1. može dovesti do konflikta sa drugim sistemskim fajlovima
2. otežava održavanje konzistentnosti pre korišćenju različitih verzija Python-a i različitih virtuelnih okruženja (eng. *virtual environments*)

2.8.2 Kreiranje Python projekta koji koristi Poetry

Nakon isntalacije alata, moguće je kreirati projekat koji koristi Poetry za upravljanje paketima ili dodati Poetry u postojeći projekat i time kreirati virtuelno okruženje u kojem će se izvršavati naredne akcije. U oba slučaja će se kreirati *pyproject.toml* datoteka koja čuva sve bitne informacije o projektu. U početku ova datoteka sadrži samo osnovne informacije o projektu:

```
[tool.poetry]
name = "poetry-demo"
version = "0.1.0"
description = ""
authors = ["Name Surname <email>"]

[tool.poetry.dependencies]
python = "*"

[tool.poetry.dev-dependencies]
pytest = "^3.4"
```

Informacije o samom projektu, njegovom autoru, verziji, licenci, opisu, dokumentaciji, itd. nalaze se u prvom segmentu - `[tool.poetry]`. Informacije o paketima koje projekat koristi nalaze se u naredne dve sekcije - `[tool.poetry.dependencies]` i `[tool.poetry.dev-dependencies]`, gde se u prvoj od ove dve sekcije nalaze informacije o paketima koji se koriste u produkcionom okruženju, a u drogoj informacije o paketima koji se

koriste u toku razvoja projekta. Pri instalaciji novog paketa, njegov naziv i verzija će se dopisati u jednu od ove dve sekcije, ili u obe ako je to potrebno.

Ovu datoteku je moguće ručno menjati, ali se sa njom najčešće interaguje pozivanjem poetry komandi u konzoli. Neke od tih komandi su:

- `poetry add` - dodaje novi paket u *pyproject.toml* datoteku
- `poetry remove` - briše paket iz projekta

Komande koje takođe interaguju sa *pyproject.toml* datotekom, ali je ne menjaju su:

- `poetry install` - instalira pakete definisane u *pyproject.toml* datoteci
- `poetry update` - ažurira pakete u skladu sa verzijama koje pišu u *pyproject.toml* datoteci
- `poetry lock` - zaključava pakete u prijektu
- `poetry check` - proverava ispravnost *pyproject.toml* datoteke

Za pokretanje komandi u virtuelnom okruženju opsisanom u *pyproject.toml* datoteci koristi se komanda:

- `poetry run`

Za sve ostale detalje najbolje je pogledati dokumentaciju pomoću komande:

- `poetry help`

2.8.3 Kreiranje, pakovanje i objavljivanje Python paketa

Da bi se Python projekat mogao objaviti, potrebno je kreirati arhivu sa svim potrebnim konfiguracijama za njega. To se postiže upotrebom komande:

- `poetry build`

Nakon toga, paket je spreman za objavljivanje. U projektu se pojavi novi direktorijum sa nazivom `dist` i u njemu se nalaze dve datoteke:

1. `poetry-demo-0.1.0-py2.py3-none-any.whl`

2. `poetry-demo-0.1.0.tar.gz`

Repozitorijum na kojem se paket objavljuje se podešava pomoću komande:

- `poetry config`

Paket se objavljuje pozivanjem komande:

- `poetry publish`

U tom momentu, paket postaje dostupan na izabranom repozitorijumu i svako ko ima pristup repozitorijumu može da instalira paket. Najčešće se paketi objavljuju na PyPI repozitorijumu.

2.9 Bash

3. Specifikacija i implementacija projekta

Projekat o kojem ovaj rad govori sastoji se iz grupe alata koje je bilo potrebno napraviti i povezati da bi se dodale nove funkcionalnosti u postojeći projekat - *pjisp-assignment-template* [7]. Pomenuti alat služi za automatizovano kreiranje studentskih zadataka, nihovo puštanje u učionicama u kojima studenti rade zadatke i na kraju pregledanje zadataka. Prvenstveno je namenjen da profesorima i asistentima olakša ceo proces od kreiranja do pregledanja kolokvijuma ili zadataka koje studenti rade.

Zadatak nastavnog osoblja je pre svega da iz repozitorijuma projekta koji je šablonski (eng. *template*) repozitorijum kreira svoj repozitorijum sa zadatkom. Važno je da nastavnik svoj repozitorijum nazove u skladu sa predefinisanim načinom imenovanja. Nakon toga se u repozitorijumu kreiraju svi potrebni materijali koji su potrebni za generisanje studentskog zadatka. Zadatak nastavnika je da pokrenje generisanje zadatka za jedan od predefinisanih testova na predmetu Programski jezici i strukture podataka. Ovaj projekat dodaje mogućnost automatizovanog kreiranja zadatka za određeni test na osnovu naziva repozitorijuma koji nastavnik napravi uz proveru ispravnosti naziva.

Po završetku pisanja zadatka i njegovog testiranja, nastavnik, a najčešće asistent, je dužan da obavesti profesora o tome, da bi profesor mogao da proveri ispravnost zadatka i odobri korišćenje zadatka u nastavi ako je dobar i pogodan za studente. Kao olakšicu, nastavnik može da pokrene alat za testiranje zadatka i proveri ispravnost. Ovaj projekat uvodi isto to testiranje i nakon objavljivanja nove verzije zadatka na GitHub-u. Kao dodatnu proveru, jedan od alata iz ovog projekta proverava da li su izmenjene sve potrebne datoteke, da bi zadatak bio kompletan, odnosno da bi imao korektan tekst zadatka, primer rešenja i testove. Ako svi ovi alati vrate rezultat pozitivnog ishoda, u GitHub repozitorijumu se pojavljuje oznaka da su svi uslovi ispunjeni i da je zadatak spreman za pregledanje. U suprotnom, stoji oznaka da zadatak ne

ispunjava sve potrebne uslove.

Naredna poglavlja opisuju alate koji su korišćeni da bi se postigla opisana unapređenja *pjisp-assignment-template* alata.

3.1 pjisp-template-name

Alat *pjisp-template-name* [8] je konzolna aplikacija kola služi za proveru ispravnosti naziva repozitorijuma. Napisana je u obliku konzolne aplikacije otvorenog koda u programskom jeziku Python.

3.1.1 Način imenovanja repozitorijuma

Za imenovanje repozitorijuma koristi se predefinisani šablon, koji izgleda ovako:

```
pjisp-{SCHOOL_YEAR}-{COURSE_ID}-{TEST_ID}-{GROUP_ID}
```

gde su:

- `SCHOOL_YEAR` - školska godina, na primer 2019
- `COURSE_ID` - identifikator kursa (E214 za zimski semestar ili E111 za letnji)
- `TEST_ID` - identifikator testa (T12, T34 ili SOV)
- `GROUP_ID` - identifikator grupe studenata, na primer G10

Alat pre svega proverava da li je dužina naziva ispravna, odnosno da li naziv ima sve celine odvojene znakom -. Ako nema, ispisuje se poruke *"Repository name length not valid"* i program izlazi sa izlaznim kodom *1*. U suprotnom, program nastavlja sa izvršavanjem.

Nakon provere dužine, proverava se ispravnost ostalih tokena, prema malopre opisanom načinu davanja imena. Ako bilo gde dođe do greške, ispisuje se poruka *"Repository name not valid. Error on <token>"* i izlase se sa izlaznim kodom *1*. Ako naziv u potpunosti ispunjava šablon, kao izlazna vrednost vraća se identifikator testa, da bi na osnovu toga mogle da se generišu datoteke potrebne za kreiranje tog testa.

3.1.2 Instalacija i pokretanje

Ovaj alat se instalira kao bilo koji drugi Python paket. Potrebno je u konzolnoj liniji pokrenuti komandu:

```
pip install pjisp-template-name
```

i nakon toga je alat spreman za upotrebu. Pokreće se pozivanjem komande:

```
pjisp_template_name <template_name>
```

3.2 pjisp-diff

Alat *pjisp-diff* [9] je konzolna aplikacija kola služi pomoćni alat nastavnicima koji proverava da li su izmenili sve potrebne datoteke pri kreiranju zadataka za studente. Napisana je u obliku konzolne aplikacije otvorenog koda u programskom jeziku Python.

Pored provere da li su promenjene sve potrebne datoteke, ovaj alat proverava i da li su ostale nepromenjene sve pomoćne datoteke koje nije potrebno menjati. Datoteke koje je potrebno menjati zavise od ulaznog parametra, šablona spram kojeg se kreira zadatak. Datoteke koje je potrebno uvek menjati su:

- `assignment_solution.c`
- `assignment.rst`

Ako je vrednost ulaznog parametra jednaka “T12”, potrebno je menjati:

- `fixtures/stdio-numbers.yaml`

Ako je šablon vrednost ulaznog parametra jednaka “T34” ili “SOV”, potrebno je menjati:

- `fixtures/file-error-input-not-readable.yaml`
- `fixtures/file-error-output-not-writable.yaml`
- `fixtures/fixtures/file-text.yaml`

Datoteka koju nije dozvoljeno menjati je:

- `assignment_notes.rst`

Ako su po završetku programa izmenjene sve potrebne datiteke i nisu izmenjene one koje ne smeju da se menjaju, alat vraća *0* kao vrednost izlaznog koda. U suprotnom, alat vraća vrednost *1* uz jednu ili više poruka u obliku: *“Please change the <filename> file.”* ili *“Please do not change the <filename> file.”*

3.2.1 Instalacija i pokretanje

Ovaj alat se instalira kao bilo koji drugi Python paket. Potrebno je u konzolnoj liniji pokrenuti komandu:

```
pip install pjisp-diff
```

i nakon toga je alat spreman za upotrebu. Pokreće se pozivanjem komande:

```
pjisp_diff <template>
```

gde `<template>` može biti T12, T34 ili SOV.

3.3 poetry-publish action

oba prethodna projekta su objavljena na pypi pomocu ovoga...

3.4 Error code u smoke-test-u - zasto je potreban?

3.5 pjisp-assignment-template

pisi izmene

4. Primeri korišćenja

5. Diskusija i zakljuci

5.1 Da li su github akcije spremne za produkciju?

6. Literatura

[1] Git. *Git*. URL: https://git-scm.com.

[2] Inc GitHub. *Github*. URL: https://github.com.

[3] Inc GitHub. *Github marketplace*. URL: https://github.com/marketplace.

[4] Python Software Foundation. *Python documentation*. URL: https://www.python.org.

[5] Python Software Foundation. *The Python Package Index*. URL: https://pypi.org.

[6] Poetry. *Poetry documentation*. URL: https://python-poetry.org.

[7] Petar Marić. *pjisp-assignment-template*. URL: https://github.com/petarmaric/pjisp-assignment-template.

[8] Jelena Dokić. *pjisp-template-name*. URL: https://github.com/JRubics/pjisp-template-name.

[9] Jelena Dokić. *pjisp-diff*. URL: https://github.com/JRubics/pjisp-diff.