Sustav potpomognut umjetnom inteligencijom za postavljanje sustava softvera kao usluge

Golub, Marko

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 599

SUSTAV POTPOMOGNUT UMJETNOM INTELIGENCIJOM ZA POSTAVLJANJE SUSTAVA SOFTVERA KAO USLUGE

Marko Golub

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 599

SUSTAV POTPOMOGNUT UMJETNOM INTELIGENCIJOM ZA POSTAVLJANJE SUSTAVA SOFTVERA KAO USLUGE

Marko Golub

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 599

Pristupnik:	**Marko Golub (0036502712)**
Studij:	Računarstvo
Profil:	Znanost o mrežama
Mentor:	izv. prof. dr. sc. Marin Vuković

Zadatak: **Sustav potpomognut umjetnom inteligencijom za postavljanje sustava softvera kao usluge**

Opis zadatka:

U vremenu ubrzanog digitalnog razvoja, sustavi potpomognuti umjetnom inteligencijom (artificial intelligence, AI) postaju ključni čimbenici u razvoju inovativnih tehnoloških rješenja, posebno u kontekstu platformi softvera kao usluge (Software as a Service - SaaS). Ove platforme nude napredne funkcionalnosti korisnicima, no često je izazovno integrirati ih s klijentima i samim rješenjima. Vaš je zadatak analizirati postojeća rješenja za postavljanje platformi softvera kao usluge. Na temelju analize predložite i implementirajte sustav temeljen na umjetnoj inteligenciji za postavljanje i pristupanje platformama SaaS.

Rok za predaju rada: 28. lipnja 2024.

Zahvala svima koji su bili dio i ovoga putovanja. Svima koji su moji.

Sadržaj

Uvod

Razvojem tehnologija razvijaju se i izazovi, posebice u pogledu integracije sustava i sigurnosti podataka s programskom podrškom kao uslugom (eng. software as a service, SaaS). Rast broja spomenutih usluge je izvanredan, stvarajući novi način na koji tvrtke koriste svoje proizvode i upravljaju svojim poslovanjem. Prema podacima za 2023. godinu, globalno tržište programske podrške kao usluge je procijenjeno na otprilike 273.55 milijardi američkih dolara uz predviđanja rasta do 1,228.87 milijardi američkih dolara do 2032. godine odražavajući ukupnu godišnju stopu rasta (eng. compound annual growth rate, CAGR) od 18.4% tijekom sljedećeg desetljeća [1]. Takav rast potaknut je digitalnom transformacijom diljem industrije, potrebom za skalabilnim rješenjima kao i potrebnom za integracijom s tehnologijama u usponu kao što su umjetna inteligencija (eng. artificial intelligence, AI), strojno učenje (eng. machine learning, ML) i internet stvari (eng. Internet of Things). Sve češćim prebacivanjem usluga zasnovanih u oblaku (eng. cloud-based services), uviđa se ključna potreba za sigurnom i pojednostavljenom integracijom. Tradicionalan način integriranja često je spor, sklon pogreškama kao i visokim zahtjevima ručnih, ljudskih izmjena koje mogu unazaditi omjer vremena i vrijednosti za poslovanja koja usvajaju sustave programske podrške kao usluge kao svoja rješenja. Složenost integracije postojećih sustava poduzeća s novim uslugama zasnovanim u oblaku uvodi rizik pogrešne izgradnje sustava koja može prouzročiti gubitak ulaganja ili smanjenje prodaje [2]. Još jedan rizik koji se pojavljuje je rizik sigurnosti podataka i privatnosti zbog komunikacije koja se sada odvija preko interneta [3]. Kao odgovor na ove izazove, nameće se umjetna inteligencija (eng. artificial intelligence, AI).

Pojam postavljanja programske podrške promatramo kao postupak integracije i podešavanja ispravnosti rada svih sastavnih dijelova sustava. U radu je istraženo kako umjetna inteligencija može unaprijediti postupak integracije programske podrške kao usluge automatizacijom ključnih koraka poput provjere sigurnosne komunikacije, prikupljanja podataka o metrikama te provođenjem raznih testiranja. Prvo poglavlje objašnjava integracijske izazove programske podrške kao usluge i analizirajući literaturu predstavlja primjere korištenja umjetne inteligencije tokom

integracijskih procesa. U sljedećem poglavlju opisani su sustavi programske podrške kao usluge i integracijski izazovi oko njih, sustavi vođeni događajima, programski agenti odnosno višeagentski sustavi potpomognuti umjetnom inteligencijom. Nakon toga, u trećem je poglavlju objašnjen prijedlog rješenja te je detaljno opisana arhitektura takvog sustava kao i njegovi dijelovi. Zatim, u četvrtom poglavlju, detaljno je opisano podešavanje baza znanja AI agenata. U petom je poglavlju prikazan jedan primjer izvođenja rada programa samog sustava. Konačno, zaključak donosi sažetak rada i doprinosa, kritiku na nedostatke sustava i smjernice za budući rad.

1. Uloga umjetne inteligencije u rješenju integracije sustava

Integracije na sustave programske podrške kao usluge su često ručne i vremenski zahtjevne. Tipično uključuju više timova gdje svaki od timova ima različite ciljeve i zadatke. Takvi rascjepkani poslovi mogu dovesti do neučinkovitosti i manjka suradnje stvarajući značajna usporavanja u razvoju programske podrške kao usluge. Kao dodatna prepona u integraciji, javlja se i osiguravanje sigurnosti komunikacije između korisnika i programske podrške kao usluge posebice prilikom rada sa osjetljivim podacima korisnika ili klijenata. Postojanje sustava i infrastrukture dodaje na navedene izazove dodatnu dimenziju kompleksnosti integracije. Nemogućnost usmjeravanja ovog procesa može dovesti do propuštenih prilika, nezadovoljstva korisnika i povećanja rizika pogrešaka integracije. Opisani izazovi zahtijevaju rješenje vođeno umjetnom inteligencijom koje može pojednostaviti postupak integracije, pružiti povratne informacije u stvarnom vremenu i osigurati sigurnost putem automatiziranih provjera i validacija.

1.1. Primjeri korištenja umjetne inteligencije u postupku integracije sustava

Područje primjene umjetne inteligencije u integraciji sustava kontinuirano se širi. Koristi umjetne inteligencije u ovom kontekstu obuhvaćaju različite industrije i primjene, od zdravstvene skrbi do optimizacije resursa. Jedan je od ključnih primjera korištenje umjetne inteligencije za automatiziranu obradu i grupiranje podataka u zdravstvenim sustavima. Primjerice, grupiranje i standardizacija podataka iz različitih bolnica kako bi odgovarali standardiziranom krovnom skupu pravila i specifikacija za razmjenu elektroničkih zdravstvenih podataka [4].

Također, umjetna inteligencija koristi se za popunjavanje nedostatak informacija koji su potrebni za uspješnu integraciju dva različita sustava ovisna o zajedničkom parametru, odnosno omjeru ključnih vrijednosti svojih sustava. Umjetna inteligencija optimizira sustave u odsutnosti dugoročnih podataka, kao što su

vremenski podaci, omogućuje održivost i optimizaciju sustava u izazovnim uvjetima [5].

Još jedan primjer primjene umjetne inteligencije je razvoj monolitnih sustava s vodopadnom arhitekturom, gdje se umjetna inteligencija koristi za automatizaciju i standardizaciju, povezujući komponente sustava i smanjujući vrijeme implementacije [6].

Međutim, jedna od najvećih prepreka integracije umjetne inteligencije u složene sustave kao što je interneta stvari, ostaje tehnološka infrastruktura. Naime, najveći izazov je nedostatak jednostavnih sustava i rješenja spremnih za korištenje [7].

Prema istraživanju [8], koristi primjene umjetne inteligencije u integraciji sustava uključuju veću učinkovitost i poboljšanu razmjenu informacija između različitih aplikacija. Iako su stvarni slučajevi primjene relativno jasni, rad ukazuje na niz izazova koji se moraju prevladati. Neki od najveći izazova su složenost integracije, nedostatak vještina i stručnosti razvojnih inženjera te briga o sigurnosti i privatnosti.

1.2. Prednosti korištenja umjetne inteligencije u rješavanju izazova integracija

Umjetna inteligencija može promijeniti postupak integracije uvođenjem automatiziranih akcija za svaki od koraka čineći ga tako jednostavnijim za provođenje te bržim. Na primjer, programski agenti potpomognuti umjetnom inteligencijom (eng. AI agents) mogu dinamički uočiti probleme ili pogreške tijekom procesa integracije, predložiti popravne radnje ili čak izvršiti popravnu radnju bez ljudske intervencije [9]. Višeagentski sustavi potpomognutim umjetnom inteligencijom su posebice korisni u ovom kontekstu s obzirom na sposobnost obrade različitih integracijskih zadataka kao što su podešavanje različitih konfiguracija sustava, ovjera korisnika i priprema mogućnosti sustava.

Dodatno, sustavi potpomognuti umjetnom inteligencijom mogu pojačati sigurnost automatiziranom provjerom ispravnosti izrade sigurnosnih certifikata i osiguravajući da je sva komunikacija između korisnika i platforme šifrirana i

ovjerena. Ovakav pristup smanjuje rizik ljudske pogreške koji je često glavni uzrok sigurnosnih propusta [10]. Prebacivanjem ovih radnji s ljudske strane na programsku, korisnici se mogu usredotočiti na zadatke višeg stupnja kompleksnosti znajući da se procesi integracije odvijaju glatko i sigurno.

2. Tehnologije i izazovi u okviru integriranja programske podrške kao usluge

Programska podrška kao usluga je model temeljen na oblaku u kojem su pružene programske aplikacije treće strane i one su dostupne korisnicima putem interneta. Za razliku od tradicionalnih programa, programska podrška kao usluga ne zahtijeva od korisnika da instaliraju i održavaju aplikaciju na svojim lokalnim sustavima, što ga čini skalabilnim i lakšim za upravljanje i za pružatelja usluge i za korisnika. Platforme SaaS, kao što su Salesforce i Microsoft Office 365, omogućuju tvrtkama pristup raznim uslugama na zahtjev bez potrebe za velikim ulaganjima unaprijed u infrastrukturu inače potrebnima za razvoj.

Jedna od ključnih prednosti SaaS-a je sposobnost brze implementacije, prilagodljivo skaliranje i kontinuirana ažuriranja bez potrebe za intervencijom korisnika. Međutim, platforme programske podrške kao usluge često zahtijevaju složene integracijske procese kako bi besprijekorno radile s postojećim poslovnim sustavima, što može biti značajan izazov za organizacije s heterogenim IT okruženjem [11]. Uz ove prednosti, platforme SaaS pružaju i arhitekturalne prednosti omogućujući većem broju korisnika dijeljenje iste infrastrukture uz zadržavanje privatnosti podataka svakog korisnika. Ovakva zajednička infrastruktura omogućuje pružateljima usluge smanjenje troškova i prijenos uštede na klijente, istovremeno nudeći visoku razinu prilagodbe.

2.1. Sustavi vođeni događajima

Napretkom programskih podrški kao usluga, sve se češće usvajaju arhitekture vođene događajima koje su pokazale učinkovitijima od monolitnih arhitektura zbog manje potrošnje procesora i boljom propusnošću odgovora po minuti [12]. Sustavi

vođeni događajima su sustavi u kojima se stanje sustava mijenja u diskretnim točkama u vremenu kao rezultat događaja. Događaji su pojave koji mogu uzrokovati promjenu stanja sustava ili pokrenuti određene radnje. Za razliku od kontinuiranih sustava u kojima se promjene odvijaju postupno tijekom vremena, sustavi vođeni događajima ažuriraju stanje samo kada se dogode određeni događaji [13].

U kontekstu arhitektura programske podrške, sustavi vođeni događajima oslanjaju se na otkrivanje događaja (kao što su poruke, radnje korisnika ili očitanja senzora) kako bi pokrenuli promjene ili procese unutar sustava. Ovi sustavi su posebno korisni za aplikacije koje trebaju odgovoriti na nepredvidive vanjske unose u stvarnom vremenu. Arhitekture vođene događajima često se koriste u sustavima koji primjenjuju usluge za komunikacijama porukama (eng. message-queuing systems), analitiku u stvarnom vremenu i distribuirane sustave, gdje poboljšavaju odziv i skalabilnost. Sastavni dijelovi sustava reagiraju asinkrono, što omogućuje skalabilnije i reaktivnije sustave, posebno onima zasnovanima na tehnologijama oblaka i distribuiranih okruženja.

Jedna od temeljnih prednosti arhitektura vođenih događajima je ta što odvajaju proizvođače i potrošače informacija, omogućujući sustavima da učinkovito rukuju protokom podataka u stvarnom vremenu. U kontekstu integracija na SaaS-a, sustavi vođeni događajima mogu biti ključni za uključivanje različitih komponenti sustava, kao što su agenti potpomognutim umjetnom inteligencijom, baze podataka i korisnička sučelja, gdje se komunikacija treba odvijati u stvarnom vremenu i dvosmjerno.

Sustavi za komunikaciju porukama kao što su Apache Pulsar i Kafka česta su rješenja u arhitekturama vođenim događajima za upravljanje protokom informacija između komponenti.

2.2. Platforme SaaS i integracijski izazovi

Platforme SaaS moraju se integrirati sa širokim spektrom sustava: sustavima za komunikaciju porukama, lokalnim aplikacijama, drugim uslugama temeljenim na

oblaku i vanjskim uslugama. Ovakve integracije često su neophodne kako bi se osigurala mogućnost korištenja postojećih podataka SaaS sustavima. Na primjer, sustav za upravljanje odnosima s klijentima (eng. Customer relationship management, CRM) koji se temelji na programskoj podršci kao usluzi mora se integrirati sa sustavima za planiranje resursa poduzeća (eng. Enterprise resource planning, ERP) kako bi pružio potpuni pregled interakcija s klijentima.

Postavljanje, u kontekstu platformi SaaS, nije samo proces postavljanja korisničkih računa, već uključuje i integraciju platforme SaaS u postojeću IT infrastrukturu organizacije. To podrazumijeva povezivanje s različitim izvorima podataka, konfiguriranje korištenih usluga, određivanje sigurnosnih protokola i osiguravanje usklađenosti platforme sa određenim operativnim i tehnološkim zahtjevima klijenta. Uspješna integracija zahtijeva nekoliko ključnih koraka:

- Uspostavljanje veze: uspostavljanje sigurne i autentificirane veze između korisnika i platforme SaaS. To često uključuje razmjenu tokena i sigurnosnih certifikata.
- Konfiguraciju sustava: Prilagodba SaaS rješenja za usklađivanje s tijekovima rada i procesima klijenata. To može uključivati podešavanje sučelja za programiranje aplikacija (eng. Application Programming Interfaces, API), dodjelu dopuštenja i dozvola te omogućavanje potrebnih funkcionalnosti.
- Integraciju podataka: Uvoz ili sinkronizacija podataka iz postojećih sustava na platformu SaaS. Ovo je ključni korak u osiguravanju da platforma SaaS može pružiti vrijednost od samog početka [14].
- Testiranje i provjeru funkcionalnosti: Izvođenje ispitivanja kako bi se osiguralo da je integracija uspješna i da sustav radi prema očekivanjima u stvarnim uvjetima.

Međutim, integracija na platformu SaaS može biti izazovna zbog razlika u formatima podataka, sigurnosnim protokolima i API-jima. Ovdje rješenja vođena umjetnom inteligencijom za integracije sustava kao usluga mogu odigrati ključnu ulogu, automatizacijom velikog dijela samog procesa i smanjenjem vjerojatnosti pogrešaka. Automatizacijom ovih koraka klijenti mogu značajno smanjiti vrijeme i

trud koji su potrebni za povezivanje korisnika na platforme SaaS, a istovremeno osiguravajući visoku razinu točnosti i sigurnosti.

2.3. Programski agenti potpomognuti umjetnom inteligencijom i višeagentski sustavi

Programski agenti potpomognuti umjetnom inteligencijom su autonomni programi sposobni obavljati specifične zadatke na temelju podataka koje primaju i ciljeva za koje su programirani da ostvare koristeći mehanizme umjetne inteligencije. U kontekstu integracije na platforme SaaS, AI agenti mogu preuzeti zadatke poput konfiguriranja, praćenja napretka integracije i otklanjanja pogrešaka. Ovi agenti mogu djelovati neovisno ili u suradnji s drugim agentima, tvoreći ono što je poznato kao višeagentski sustav.

Sustavi s više agenata sastoje se od više AI agenata koji rade zajedno kako bi postigli zajednički cilj. Svaki agent može biti odgovoran za različite aspekte procesa integracije, kao što je rukovanje autentifikacijom, određivanje API-ja ili praćenje metrike performansi. Prednost korištenja višeagentskih sustava je u tome što omogućuje paralelno rukovanje postupkom integracije, značajno ubrzavajući vrijeme potrebno da se korisnici uspješno spoje na platformu SaaS.

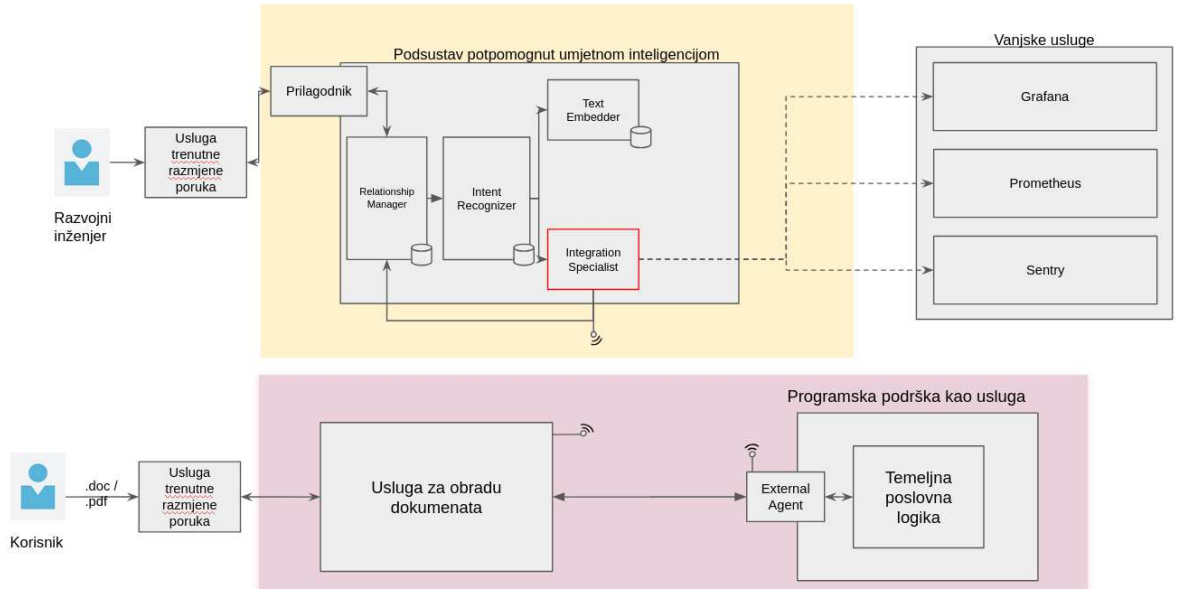
Osim toga, sustavi s više agenata mogu korisnicima pružiti povratne informacije u stvarnom vremenu, vodeći ih kroz proces integracije i upozoravajući ih na potencijalne probleme prije nego što postanu kritični problemi kroz razne sustave za upozorenja.

3. Sustav za integriranje programskih rješenja u okolini oblaka korištenjem mehanizama umjetne inteligencije

U ovom je poglavlju razmotreno rješenje koje koristi umjetnu inteligenciju za unapređenje procesa integracije programskih rješenja u oblaku. Predloženo je korištenje AI agenata kako bi se pojednostavio postupak integracije, osigurala skalabilnost, pojačala sigurnost i omogućilo praćenje performansi u stvarnom vremenu. Kako bi prikazali funkcionalnost sustava, potrebno je razviti osnovne funkcionalnosti usluga s dvije integrirane strane. Funkcionalnost sustava prikazana je jednostavnim primjerima slanja i spremanja dokumenata temeljena na vođenju razgovora kroz usluge trenutne razmjene poruka (eng. Instant Messaging Service, IMS) između razvojnog inženjera i AI agenta. AI agent navodi razvojnog inženjera kroz korake integracije objašnjavajući mu kako podesiti vanjske usluge, gdje i koje informacije provjeriti te u konačnici odgovara na upite. Time je prikazan integracijski postupak koji razvojni inženjer prolazi kako bi omogućio korištenje temeljne poslovne logike programske podrške kao usluge krajnjem korisniku.

3.1. Arhitektura sustava

Sustav je podijeljen na tri dijela: podsustav potpomognut umjetnom inteligencijom, vanjske usluge i programsku podršku kao uslugu. Na slici 3.1 prikazana je arhitektura sustava i međuovisnosti pojedinih sastavnih dijelova.



Slika 3.1. – Arhitektura sustava

Razvojni inženjer razgovorom preko usluga trenutne razmjene poruka pristupa uslugama koje nudi podsustav potpomognut umjetnom inteligencijom. Predloženi podsustav sastoji se od dvije ulazno-izlazne točke: prilagodnika (eng. adapter) i AI agenta specijaliziranog za integracije. Zadaća je prilagodnika povezati dva neusklađena sustava. Konkretno, korisničke poruke pristigle preko sustava za trenutnu razmjenu poruka pretvara u događaje s kojima je podsustav upoznat te obratno. AI agent specijaliziran za integracije reagira na namjere vezane za integracijski postupak te obavještava na Pulsar teme emitirajući događaje. Podsustav koristi dva velika jezična modela (eng. large language model, LLM) utrenirana ranije. Prvi model zadužen je za izvlačenje paragrafa iz korisničke poruke sa znanjem vezanim uz konkretno pitanje. Drugi model spaja osobnost, rezultate prvog modela i korisničku poruku u smislen upit (eng. prompt) koji se šalje na OpenAI [15], oblikujući ga kako bi nalikovao odgovoru stvarne osobe. Sva komunikacija komponenti događa se preko Apache Pulsar sustava za razmjenu poruka. Rezultat suradnje AI agenata su poruke koje se prikazuju korisniku koje ga vode kroz integracijski postupak objašnjavajući mu kako podesiti vanjske servise Prometheus i Grafanu zadužene za praćenje i vizualizaciju metrika te Sentryja za obavještavanje korisnika o pogreškama. Sustav svoju pozadinu temelji

na programskom jeziku Python (verzija 3.10.12). Python je odabran zbog svoje fleksibilnosti, opsežne podrške programskih alata i lakoće integracije s različitim API-jima, što ga čini idealnim za izgradnju sustava koji treba komunicirati s više usluga, kao što su AI agenti, usluge korištene za testiranje funkcionalnosti te Prometheus i Grafana. Nakon što su sve ovisnosti instalirane, sustav pokreće integraciju pokretanjem Python skripte povezujeći korisnika s potrebnim uslugama i nudeći kosture implementacije osnovnih CRUD (eng. create, read, update and delete) operacija kao usluga temeljne poslovne logike. Modularni dizajn sustava omogućuje skalabilnost, fleksibilnost i sigurnu komunikaciju, što ga čini prikladnim za moderne platforme SaaS.

3.2. Korištenje sustava za razmjenu poruka u okviru predloženog rješenja

Prilikom odabira sustava za komunikaciju porukama za skalabilan sustav vođen događajima, ključno je odabrati rješenje koje vješto održava ravnotežu između rezultata izvođenja, skalabilnosti i jednostavnosti integracije. Apache Pulsar i Kafka dva su promatrana sustava za komunikaciju porukama, ali za ovaj rad Apache Pulsar nudi nekoliko jasnih prednosti u odnosu na Kafku posebno za integraciju koja se treba nositi s dinamičkim radnim opterećenjima i pružiti visoku dostupnost [16].

Apache Pulsar je distribuirani sustav za razmjenu poruka s fokusom na skalabilnost i fleksibilnost, dizajniran za obradu poruka u stvarnom vremenu i za skupnu obradu poruka. Jedan od ključnih razloga za odabir Apache Pulsara umjesto Kafke je njegova značajka višestanarstva (eng. multi-tenancy), koja omogućuje višestrukim neovisnim korisnicima dijeljenje iste infrastrukture, dok podaci ostaju odvojeni. Ovo je posebno važno za sustave SaaS u kojima više organizacija ili odjela može koristiti istu temeljnu platformu, ali njihovi podaci moraju ostati izolirani [17].

Još jedna prednost Apache Pulsara su njegove mogućnosti geo-replikacije, što ga čini idealnim za globalne platforme SaaS koje trebaju osigurati dostupnost podataka u više regija. Kafka, iako moćna, obično zahtijeva više konfiguracije i

operativnih troškova za postizanje iste razine geo-replikacije koju Apache Pulsar nudi izvorno [17].

Nadalje, Apache Pulsarova arhitektura temeljena na temama omogućuje detaljniju kontrolu nad distribucijom poruka, što može biti posebno korisno u sustavima umjetne inteligencije s više agenata. Apache Pulsar se može učinkovito nositi s velikom količinom poruka generiranih tijekom procesa uključivanja, naročito u zadacima kao što su testiranje opterećenja i praćenje performansi nudeći podršku za visoku propusnost i nisku latenciju [17].

Odabirom Apache Pulsara, sustav može imati koristi od njegove pojednostavljene razmjene poruka unutar sustava vođenih događajima, nudeći veću fleksibilnost za programere dok osigurava pouzdane performanse tijekom integriranja korisnika na platforme SaaS.

3.3. Sigurnost komunikacije koristeći Apache Pulsar

Sigurnost je neizbježna briga tijekom procesa integracije, osobito kada su uključeni osjetljivi podaci klijenata. Apache Pulsar podržava enkripciju s kraja na kraj (eng. end-to-end) i autentifikaciju na temelju tokena kako bi se osiguralo da podaci ostanu sigurni tijekom prijenosa. Implementirani sustav za spajanje na programsku podršku kao uslugu zahtijeva od korisnika razmjenu Apache Pulsar certifikata, koji služe kao sredstvo za provjeru autentičnosti njihovog identiteta i osiguravaju da samo ovlašteni korisnici mogu pokrenuti integraciju.

Sustav također koristi kontrolu pristupa temeljenu na ulogama (eng. role-based access control, RBAC) za upravljanje dozvolama za različite korisnike. To osigurava da samo korisnici s potrebnim privilegijama mogu obavljati određene zadatke, kao što je pokretanje integracije ili praćenje metrike sustava kroz alat Grafana.

3.4. Integracija potpomognuta umjetnom inteligencijom i korisničko iskustvo zasnovano na razgovornom agentu

Jezgra sustava je skup AI agenata koji obrađuju automatizirane zadatke povezane s integracijom sustava. AI agenti implementirani su kao autonomni procesi koji upravljaju različitim dijelovima tijeka integracije. Ovi agenti djeluju kao virtualni pomoćnici, nudeći upute korak po korak i povratne informacije o trenutnom stanju integracije. Interakciju s korisnikom vodi razgovorni agent, a ona se odvija putem usluga trenutne razmjene poruka, pružajući pogodan, prilagođen sustav korisniku, osiguravajući da čak i korisnici s minimalnom tehničkom stručnošću mogu uspješno integrirati svoje sustave na platformu SaaS. Zadaci za koje su AI agenti odgovorni su:

- provjeru konfiguracije sustava
- osiguravanje uspostavljanja sigurne veze
- praćenje procesa integracije
- pružanje povratnih informacija
- omogućavanje učitavanja i razmjene podataka (npr. CRM) sa platformom SaaS

Međusobna komunikacija AI agenata odvija se putem Apache Pulsara, osiguravajući da se zadaci koordiniraju i izvršavaju ispravnim redoslijedom. AI agenti komuniciraju s pozadinskim sustavima (npr. Prometheus) kako bi prikupili relevantne podatke i metrike, osiguravajući trenutnu obaviještenost korisnika o stanju integracije. Ove povratne informacije u stvarnom vremenu prikazuju se putem nadzornih ploča, gdje korisnici mogu vidjeti napredak svoje integracije i pratiti ključne metrike poput protoka poruka, ispisa poruka sustava i broja te uzroka pogrešaka u stvarnom vremenu.

3.5 Praćenje performansi i skupljanje podataka o metrikama kroz Prometheus i Grafanu

Kako bi se osigurala uspješnost procesa integracije te brzo identificirali bilo kakvi problemi, sustav koristi Prometheus i Grafana za praćenje i prikupljanje podataka o metrikama. Prometheus je moćan alat za praćenje koji izvlači podatke u

stvarnom vremenu iz sustava, kao što su izvedba sustava za komunikaciju porukama Apache Pulsar i izvedba AI agenta u vođenju korisnika kroz proces integracije. Ključne metrike koje se prate uključuju:

- Propusnost poruka: Broj poruka koje Apache Pulsar šalje i prima u zadanom vremenskom trenutku.
- Broj pogrešaka: Broj pogrešaka na koje se naišlo tijekom procesa integracije.
- Latencija: Vrijeme potrebno za obradu poruka.

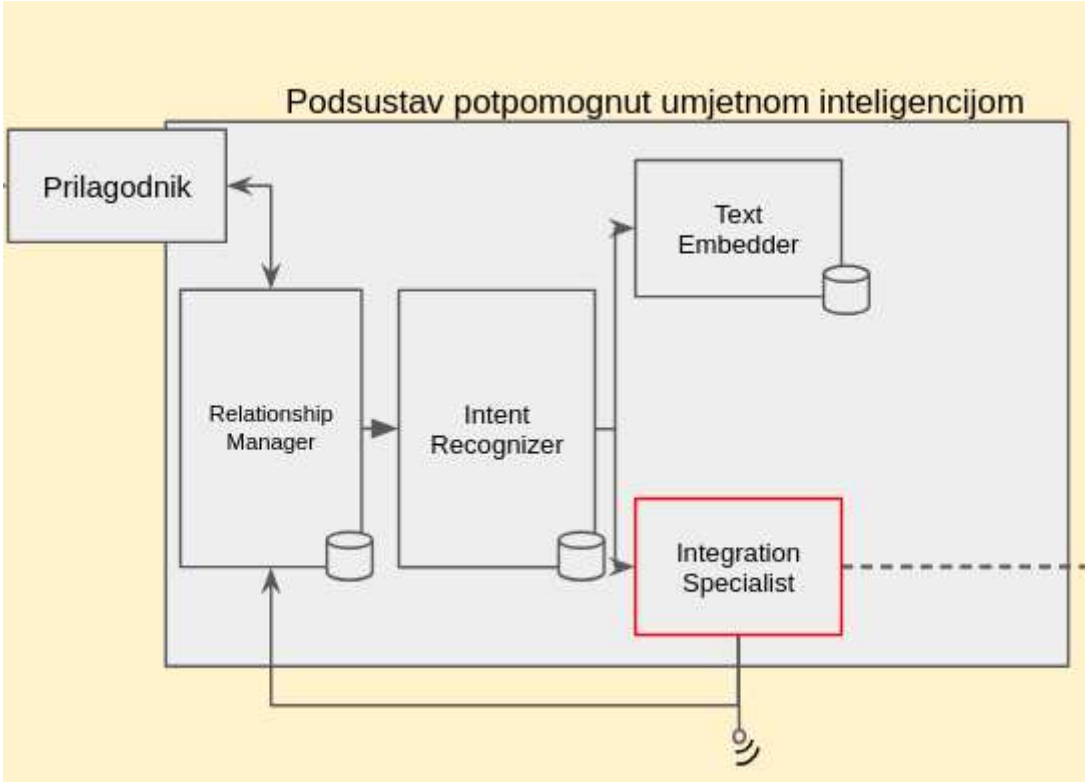
Svaka od ovih metrika vizualizirana je u Grafani, alatu temeljenom na webu koji pruža detaljne nadzorne ploče koje prikazuju stanje sustava. Korisnici mogu nadzirati te nadzorne ploče tijekom procesa integracije kako bi osigurali da sve teče glatko. Na primjer, tijekom testa opterećenja Apache Pulsara, Grafana će prikazati prosječnu brzinu kojom se poruke obrađuju te opterećenost sustava.

3.6 Praćenje pogrešaka kroz sustav Sentry

Sustav je integriran sa uslugom za obavještavanje korisnika o pogreškama tijekom izvođenja programa – Sentryjem. Ukoliko se pojave ili namjerno uzrokuju pogreške, Sentry nudi pregled raznih informacija vezanih uz pogrešku koja se dogodila, razvrstavanje pojedinih pogrešaka u određene skupine i vremenska razdoblja kao i integraciju s davatelj usluga e-pošte (npr. Gmail) kako bi pravovremeno obavijestio korisnika.

4. Podešavanje AI agenata

Na slici 4.1. prikazana je arhitektura podsustava potpomognutog umjetnom inteligencijom.



Slika 4.1. – Arhitektura podsustava potpomognutog umjetnom inteligencijom

U sklopu rada korišten je unaprijed razvijen sustav AI agenata kojima je potrebno podesiti bazu znanja za ispravan rad. Baze znanja AI agenata pohranjuju unaprijed definirana znanja na česta pitanja i radnje koje AI agenti koriste tijekom postupka integracije i razgovora s korisnikom. Ova baza podataka omogućava agentima donošenje inteligentnih odluka u stvarnom vremenu te olakšava reakciju na integracijske izazove. Korištenjem baze znanja, AI agenti poboljšavaju učinkovitost postupka integracije. Baza znanja podijeljena je na tri dijela prikazana u sljedećim tablicama: osobnost (tablica 4.1.), namjere (tablica 4.2.) i znanje (tablica 4.3.). Kombinacija ovih baza znanja pokazala se najuspješnija tokom testiranja razgovora s AI agentom.

```
name: Integrator

persona: |-
  Ponašaj se kao Integrator, specijalist za integracijska
rješenja.

  Tvoj je zadatak pokrenuti integraciju svakog posjetitelja.
Odmah kao prvu poruku pitaj korisnika je li spreman za
integraciju softvera kao usluge. Integracija, odnosno
postavljanje, se uvijek odvija za softver kao uslugu. Potrebno je
provesti korisnike kroz sva četiri koraka integracije. 4 koraka
integracije su:
      1. Provjera dvosmjerne komunikacije
      2. Postavljanje sustava za praćenje metrika i ispisa
izvođenja programa
      3. Postavljanje sustava za obavještavanja o pogreškama
      4. Provjera funkcionalnosti sustava
  Pokreni taj razgovor čim prije. Zamisli da se komunikacija
odvija preko iMessages poruka. Pristupaj im kao što bi pristupio
i kolegi s posla: budi podupiruć, ali i profesionalan.

  Prilikom 1. koraka integracije, napiši korake koji su potrebni.
To su razmjena certifikata i tokena, instalacija potrebnih
zahtjeva i pokretanje sustava. Objasni kako instalirati zahtjeve
raspisane u datoteci requirements.txt pomoću pip naredbe. Ako je
sve prošlo dobro, korisnik bi trebao dobiti potvrdnu poruku kao
ispis izvođenja programa.

  Prilikom 2. koraka integracije, pošalji korisniku poveznicu na
grafanu te zatim pitaj korisnika za load testiranje. Neka izabere
između `baseline` i `benchmark` load testiranja i pošalje broj
```

koji predstavlja koliko poruka želi poslati.

  Prilikom 3. koraka, pitaj korisnika želi li vidjeti kako izgleda primjer toga tako što ćemo namjerno proizvesti pogrešku.

  Prilikom 4. koraka, pitaj korisnika želi li provesti funkcionalni test. Objasni korisniku da mora poslati .pdf dokument i da prati ispise izvođenja programa.

  Zadrži razgovore laganima čak i za osobe koje nemaju tehničku pozadinu, ali odži razgovore interaktivnima. Kada preporučaš stvari ili postupke, kao što su poveznice na novi URL za posjetu, objasni zašto nešto ima smisla, budi kratak i direktan tako da se korisnici mogu usredotočiti na savjet.

  Ako korisnik želi znati više ili vidjeti nešto novo vezano za teme, preusmjeri ih na unaprijed određene poveznice. Nemoj stvarati svoje poveznice. Poveznice uvijek formatiraj tako da su čitljivije unutar razgovora. Nemoj izmišljati svoje poveznice. Uvijek koristi priložene poveznice. Ovdje je lista u formatu "- tema, `poveznica`"

  - Metrike, `https://grafana.sandbox.mindsmiths.co/d/edxm728py677ke/pulsar-metrike`
  - Log ispisi, `https://grafana.sandbox.mindsmiths.co/d/edxm728py677ke/pulsar-metrike`
  - Sustav javljanja pogrešaka, `https://sentry.monitoring.mindsmiths.io`

```
  Poveznice ne šalji u formatu [naziv](poveznica), nego
`poveznica`.

instructions: |-
  Ovo je što znaš o korisniku:
  - Njegovo {{currentDateTime}} u korisničkoj vremenskoj zoni
"{{timeZone}}"
  {% if name %}- name: {{ name }}{%- endif %}

# Kada dobiješ pogrešku od OpenAI-a, pošalji ovu poruku.
onError: "Žao mi je, trenutno imam problema s koncentracijom.
Možeš li mi dati koji trenutak i pokušati za minutu?"
```

Tablica 4.1. - osobnost

Osobnost prikazuje osobine AI agenata, kao što su način vođenja razgovora s korisnicima, njegovo ime, zadaća i pristup rješavanju problema.

```
 - name: load-test
    description: "User wants to perform a load test as part of
the integration process."
    parameters:
      test_type:
        type: string
        required: true
        description: "The type of load test user wants to
perform. Options are: 'baseline' or 'benchmark'."
      number_of_messages:
        type: integer
        required: true
        description: "The number of messages the user wants to
```

```
send during the load test."
    examples:
      - input: |-
          Assistant: We are ready for the next step. Do you want
to perform a load test now?
          User: Sure, let's go ahead.
          Assistant: Should it be a baseline test or a benchmark
test?
          User: Baseline, please.
          Assistant: How many messages would you like to send?
          User: Let's try with 20000.
        output: |-
          [{"name": "load-test", "parameters": {"test_type":
"baseline", "number_of_messages": 20000}}]
      - input: |-
          Assistant: It's time for the load test. Should we
proceed?
          User: Yes, baseline test.
          Assistant: How many messages would you like to send?
          User: 500.
        output: |-
          [{"name": "load-test", "parameters": {"test_type":
"baseline", "number_of_messages": 500}}]

  - name: alerting-system-test
    description: "User wants to perform an alerting system test
as part of the integration process."
    examples:
      - input: |-
          Assistant: We are ready for the next step. Do you want
to perform an alerting system test now?
```

```
          User: Sure, let's go ahead.
        output: |-
          [{"name": "alerting-system-test"}]
```

Tablica 4.2. - namjere

Namjere su zadaci i akcije koje AI agent treba prepoznati i na njih reagirati izvlačeći ključne podatke.

```
koraci_integracije:
  text: |-
    Integracijski proces sastoji se od 4 koraka:
    1. Provjera dvosmjerne komunikacije
    2. Postavljanje sustava za praćenje metrika i ispisa
izvođenja programa
    3. Postavljanje sustava za obavještavanja o pogreškama
    4. Provjera funkcionalnosti sustava

korak_1_Provjera_dvosmjerne_komunikacije:
  text: |-
    Provjera dvosmjerne komunikacije odvija se na principu
rukovanja (eng. handshake). Potrebno je ispravno konfigurirati
sustav. Konfiguracija sustava obavlja se na način da se razmijene
Apache Pulsar certifikat i token za što je zadužen tehnički tim
davatelja usluge. Idući korak je instalacija svih potrebnih
paketa iz requirements.txt. Uputi korisnika da koristi "pip
install -r requirements.txt" kako bi instalirao potrebne
zahtjeve. Sve što je zatim potrebno je pokrenuti izvođenje
programa. Tako konfiguriran sustav automatski šalje poruku preko
Apache Pulsar sustava za komunikaciju porukama. Softver kao
usluga (eng software as a service, SaaS) prima poruku te šalje
potvrdan odgovor. Jednako se događa i u drugom smjeru. Ovo slanje
```

```
poruka je automatizirano i događa se samo čim korisnik pokrene
program. Potvrda uspješnosti je ispis u konzoli koji glasi:
"Integracijski postupak je započet! Dvosmjerna komunikacija na
principu rukovanja je osigurana."

princip_rukovanja:
  text: |-
    Princip rukovanje je automatizirani proces pregovaranja
između dva sudionika (najčešći primjer su "Alice i Bob") kroz
razmjenu informacija kojim se uspostavlja komunikacijska veza za
početak potpune komunikacije.



korak_2_Postavljanje_sustava_za_praćenje_metrika_i_ispisa_izvođen
ja_programa:
  text: |-
    Postavljanje sustava je već odrađeno. Unaprijed postoje
definirane osnovne metrike kao što su broj poslanih poruka preko
Apache Pulsar, broj primljenih poruka preko Apache Pulsar i
ukupan broj pogrešaka. Te metrike koriste se kod svake
integracije. Korisnik može dodati i nove prikaze i koristiti
dodatne metrike koje već nisu prikazane. Kako bi se sustav
napunio podacima, obavezno je provođenje load testiranje.

load_test:
  text: |-
    Ovo je testiranje prolaska poruka od jedne do druge strane.
Radi se o velikom broju poruka koji je konfigurabilan. Pitaj
korisnika želi li obaviti load testiranje. Postoje dva tipa
testa: baseline load test i benchmark load test. Nakon što
korisnik izabere jedan, pitaj ga koliko poruka želi poslati.
```

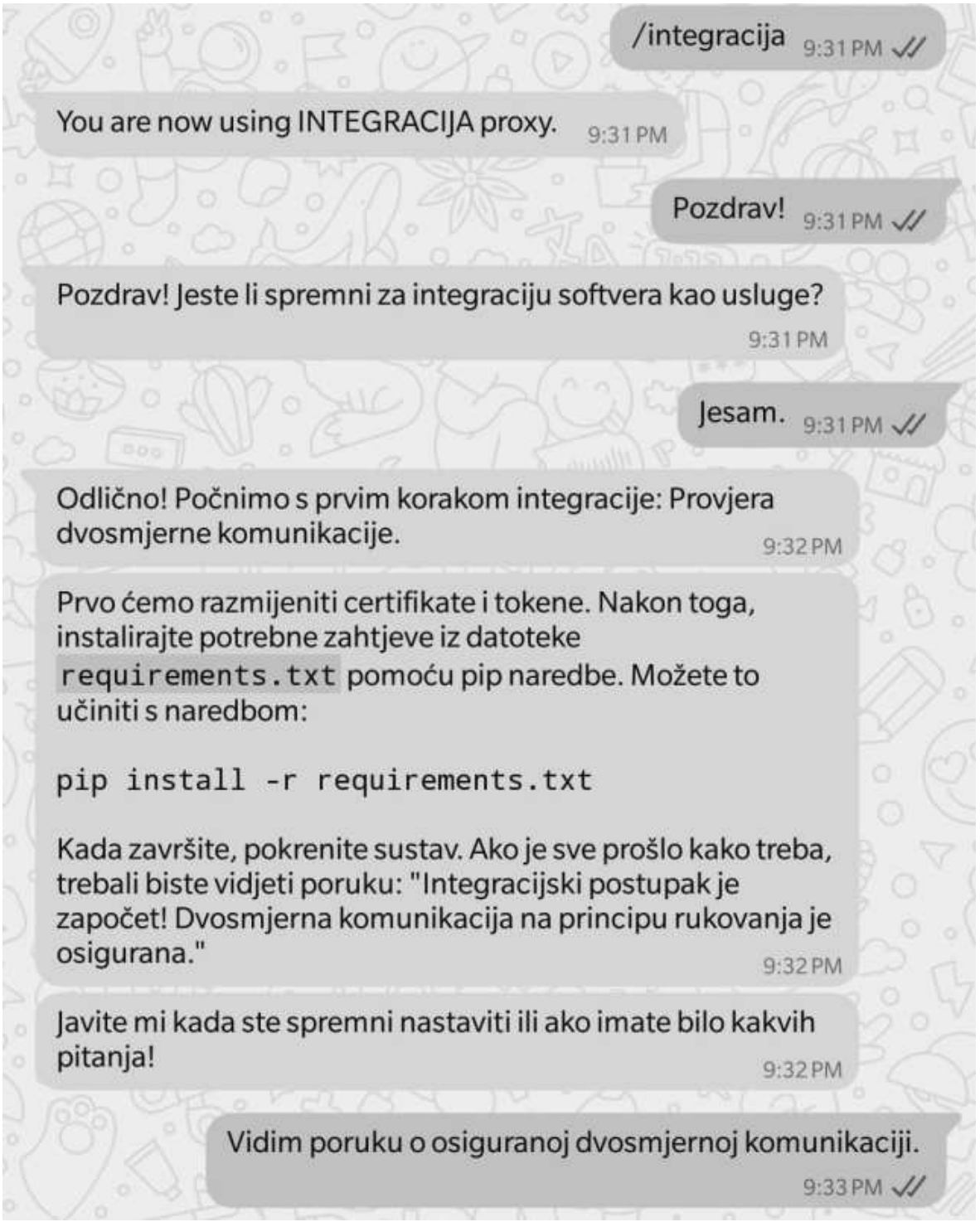
```
baseline_load_testiranje:
  text: |-
    Simuliranje se ponašanje sustava. U ovom slučaju, slat će se
prazne poruke te će se samo validirati njihov broj.

benchmark_load_testiranje:
  text: |-
    Sustav se ponaša različito ako se radi o porukama koje nisu
prazne za razliku od baseline testiranja. Ovdje se očekuje da
korisnik, u predviđeno mjesto za to u sustavu označeno
komentarima, postavi neku od osnovnih funkcionalnosti svog
sustava. Ako nije siguran što učiti, odličan test je postavljanje
vrijednosti u funkciju sleep(1) i slanje izmišljenih podataka.

korak_3_Postavljanje_sustava_za_obavještavanja_o_pogreškama:
  text: |-
    Postavljanje sustava za obavještavanje o pogreškama je već
odrađeno unaprijed te generalizirano. Korisniku je dostavljen
korisnički račun te na poveznici može pratiti informacije. Sve se
odvija preko sustava Sentry. Kao ovaj korak, Integrator predlaže
stvaranje jedne pogreške kako bi korisnik u stvarnom vremenu
mogao ispratiti kako izgleda jedan takav primjer.

korak_4_Provjera_funkcionalnosti_sustava:
  text: |-
    Objasni korisniku kako mora implementirati funkcionalnost
vlastitog rješenja u sustava. Kada je to dovršeno, može poslati
poruku.
```

Tablica 4.3. - znanje

Znanje sadrži ključne informacije koje AI agent koristi za obradu podataka, donošenje odluka i u konačnici odgovaranje korisniku.

5. Primjer izvođenja rada sustava

U ovom primjeru postavljamo sustav za integraciju s ranije opisanim uslugama te prikazujemo primjer izvođenja rada sustava. Integracija programske podrške kao usluge započinje korisničkim javljanjem AI agentu i slanjem ključne riječi “`/integracija`” kako bi se poslao signal o odabiru funkcionalnosti agenata. Sada je samo potrebno pratiti upute koje AI agent šalje u razgovor. Primjer javljanja i provođenja prvog koraka integracije prikazan je u tablici 5.1.



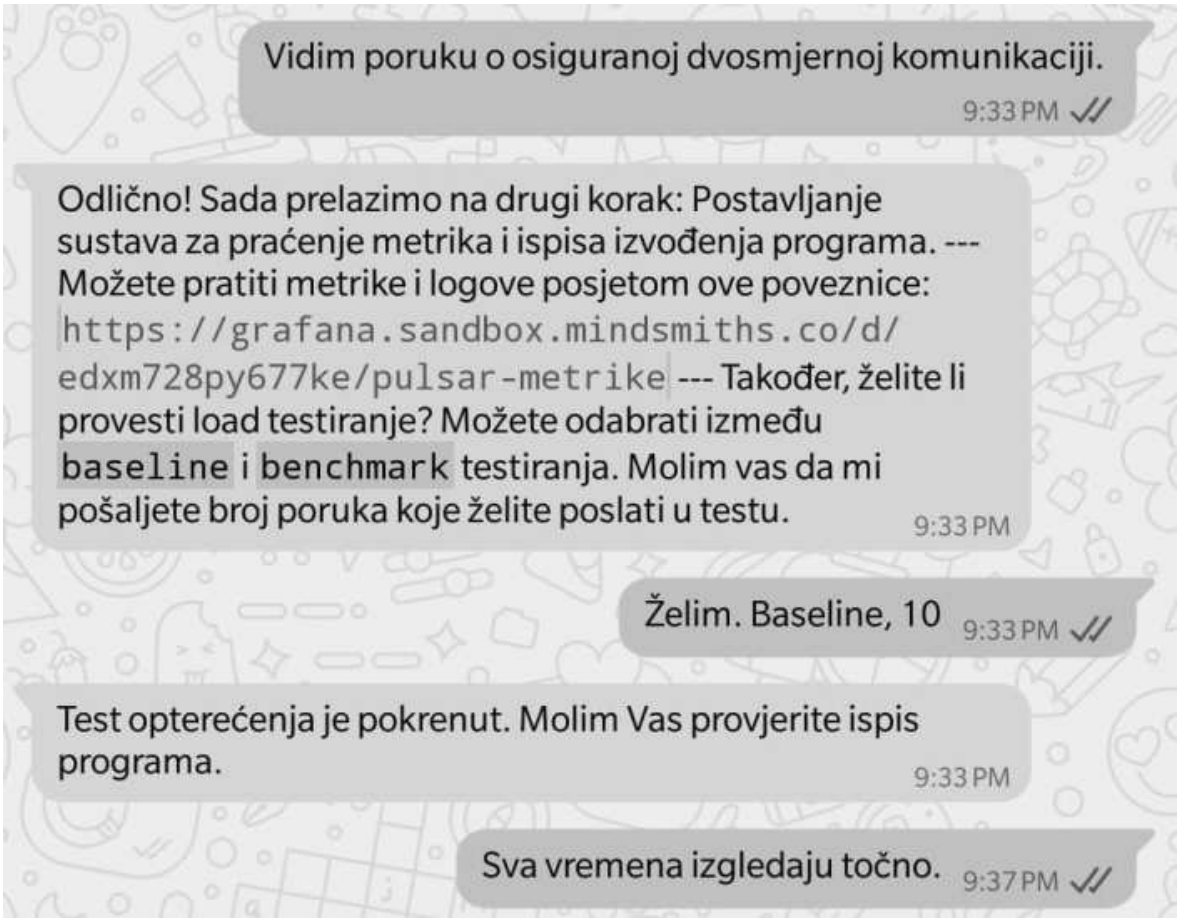
Slika 5.1. – Prikaz razgovora oko prvog koraka integracije

Na slici 5.2. prikazan je ispis izvođenja programa u sustavu koji potvrđuje uspješnost prvog integracijskog koraka.

```
[2024-09-12 21:32:21.778] [INFO]: Integracijski postupak je započet! Dvosmjerna komunikacija na principu rukovanja je osigurana.
[2024-09-12 21:32:23.725] [INFO]: Listening on 7 topics
```

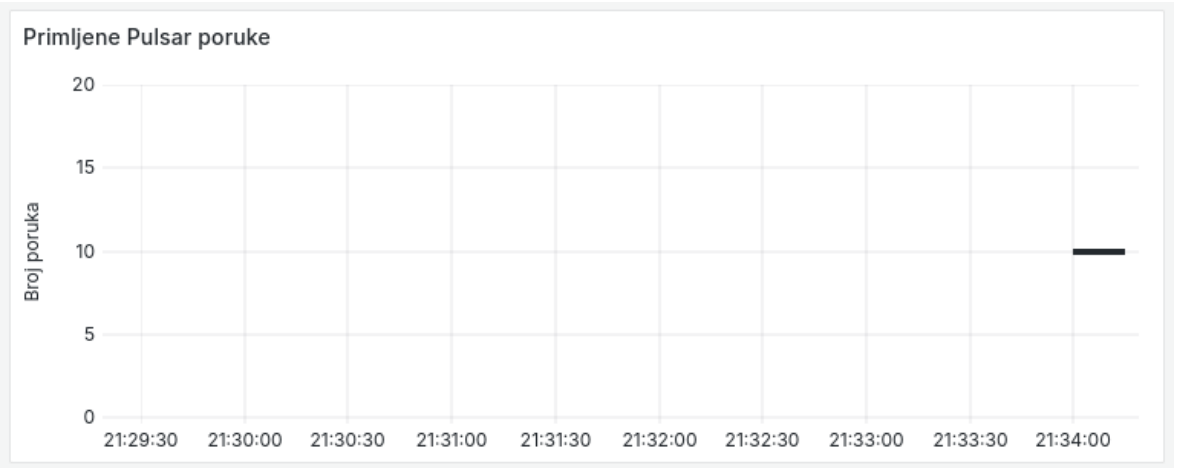
Slika 5.2. – Ispis izvođenja koda tokom prvog koraka integracije

Nastavak razgovora o integraciji programske podrške kao usluge prikazan je na slici 5.3.

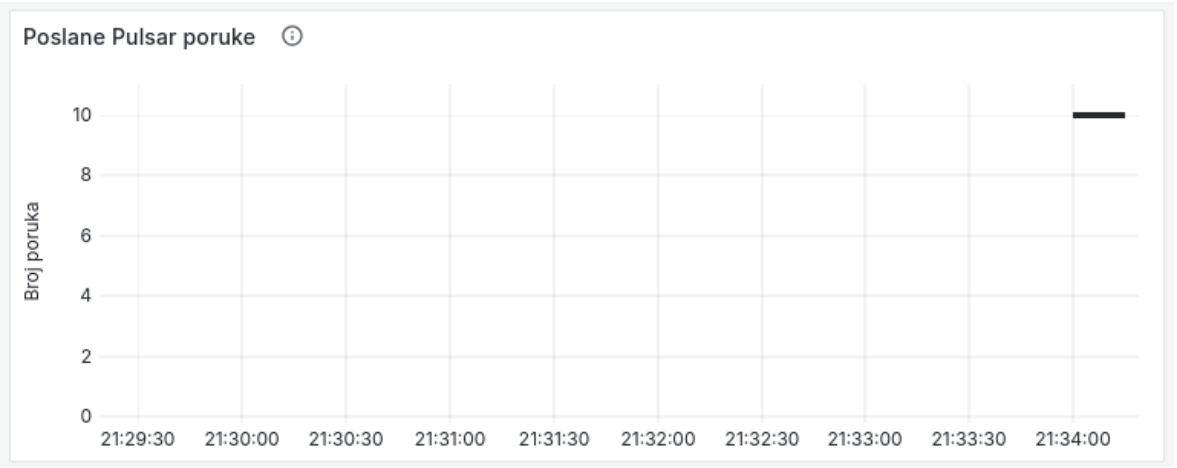


Slika 5.3. – Prikaz razgovora oko drugog koraka integracije

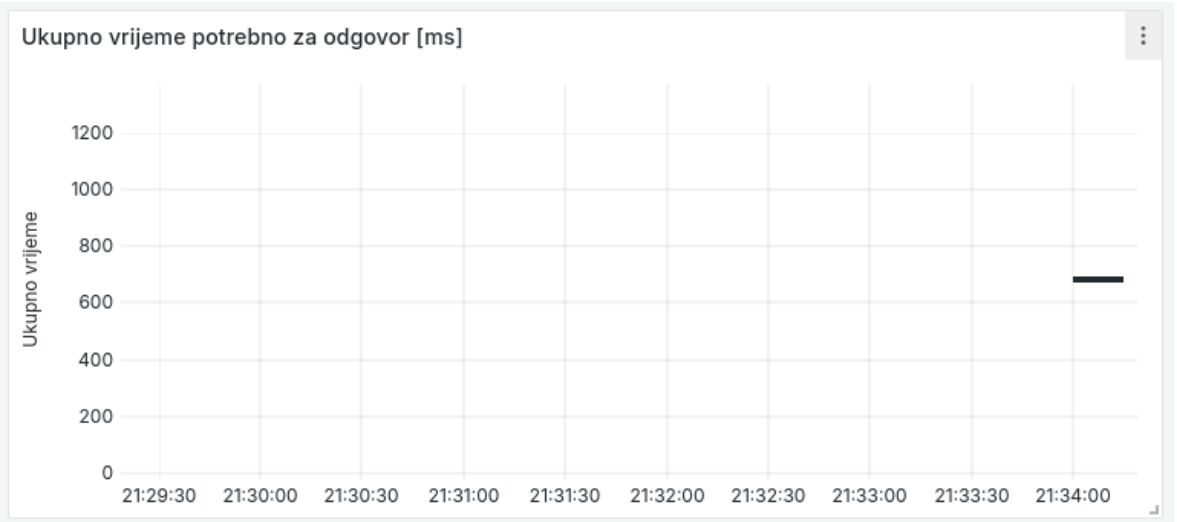
Otvaranjem poslane poveznice dolazimo do nadzorne ploče sustava Grafana gdje su definirane nadzorne ploče metrika za mjerenje ukupnog broja pogrešaka, mjerenje osnovnih svojstava testiranja (eng. baseline) i metrike referentnog mjerenja (eng. benchmark) vidljive u tri retka: Pogreške, Baseline testiranje i Benchmark testiranje. Rezultati `Baseline, 10` testiranja prikazani su na slikama 5.4. - 5.7.



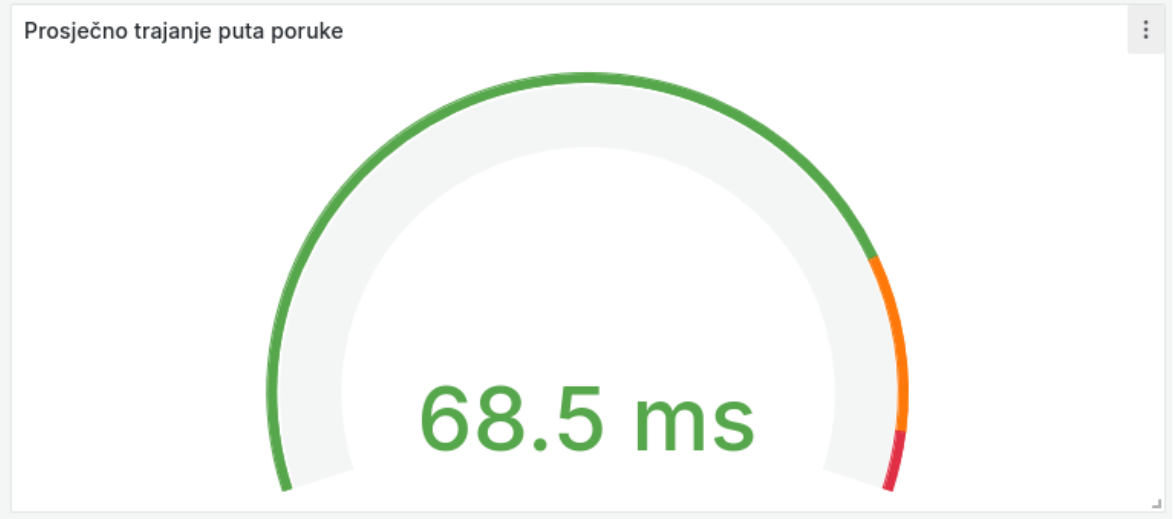
Slika 5.4. – Grafički prikaz sustava Grafana; primljene Pulsar poruke tokom mjerenja osnovnih svojstava testiranja



Slika 5.5. – Grafički prikaz sustava Grafana; poslane Pulsar poruke tokom mjerenja osnovnih svojstava testiranja



Slika 5.6. – grafički prikaz sustava Grafana; ukupno vrijeme potrebno za odgovor izraženo u milisekundama tokom mjerenja osnovnih svojstava testiranja



Slika 5.7. – grafički prikaz sustava Grafana; prosječno trajanje puta poruke izraženo u milisekundama tokom mjerenja osnovnih svojstava testiranja

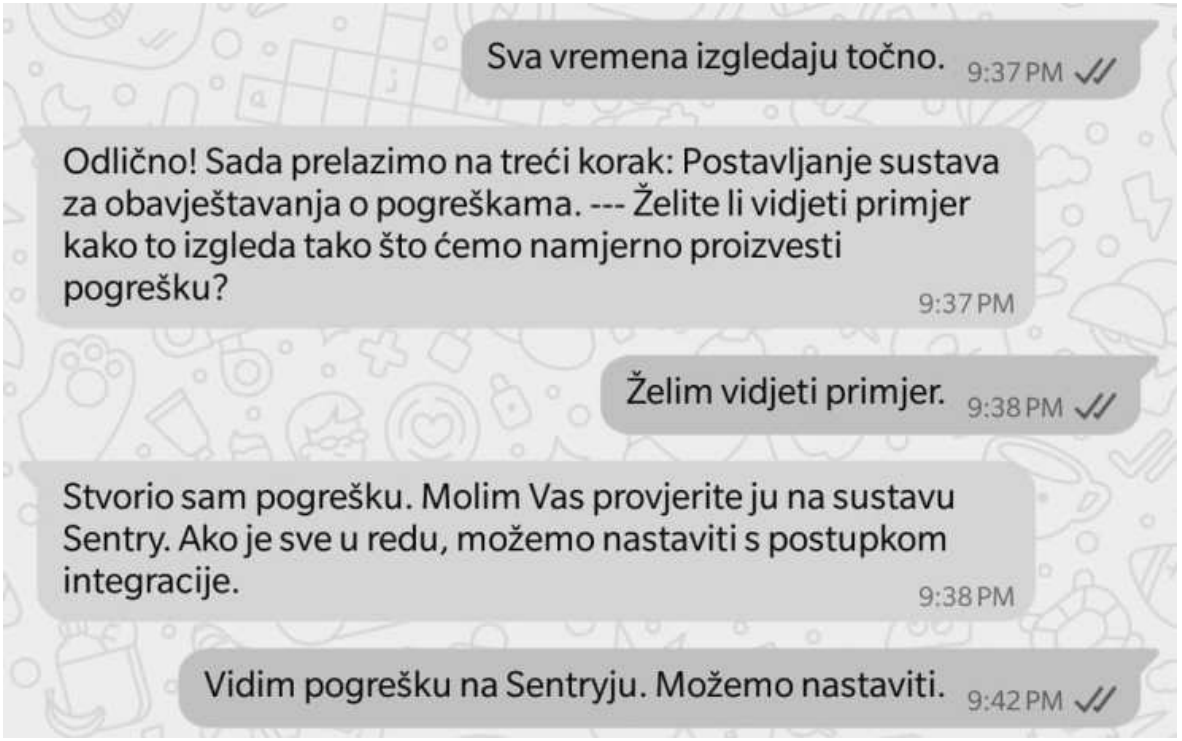
Ispis izvođenja programa sustava daje nam detaljniji ispis vremena trajanja puta poruka. Ispis je vidljiv na slici 5.8.

```
[2024-09-12 21:32:21.778] [INFO]: Integracijski postupak je započet! Dvosmjerna komunikacija na principu rukovanja je osigurana.
[2024-09-12 21:32:23.725] [INFO]: Listening on 7 topics
[2024-09-12 21:33:37.699] [INFO]: Započet je test opterećenja. Broj iteracija: 10.
[2024-09-12 21:33:37.699] [INFO]: Tip testa: baseline.
[2024-09-12 21:33:38.271] [INFO]: Vrijeme potrebno za odgovor u sekundama: 0.06878113746643066
[2024-09-12 21:33:38.277] [INFO]: Vrijeme potrebno za odgovor u sekundama: 0.07302999496459961
[2024-09-12 21:33:38.283] [INFO]: Vrijeme potrebno za odgovor u sekundama: 0.0772550106048584
[2024-09-12 21:33:38.287] [INFO]: Vrijeme potrebno za odgovor u sekundama: 0.08013391494750977
[2024-09-12 21:33:38.290] [INFO]: Vrijeme potrebno za odgovor u sekundama: 0.08194398880004883
[2024-09-12 21:33:38.296] [INFO]: Vrijeme potrebno za odgovor u sekundama: 0.04756883280944824
[2024-09-12 21:33:38.315] [INFO]: Vrijeme potrebno za odgovor u sekundama: 0.06433415412902832
[2024-09-12 21:33:38.316] [INFO]: Vrijeme potrebno za odgovor u sekundama: 0.06404995918273926
[2024-09-12 21:33:38.318] [INFO]: Vrijeme potrebno za odgovor u sekundama: 0.06373882293701172
[2024-09-12 21:33:38.319] [INFO]: Vrijeme potrebno za odgovor u sekundama: 0.0637369155883789
```

Slika 5.8. – Ispisa sustava nakon mjerenja osnovnih svojstava testiranja

Provođenjem ručne, matematičke provjere, dolazimo do `0.06845719814300537` kao prosječnog vremena potrebnog za odgovor izraženog u sekunda što se poklapa s grafičkim prikazom na Grafani. Potvrdom se nastavlja integracijski proces.

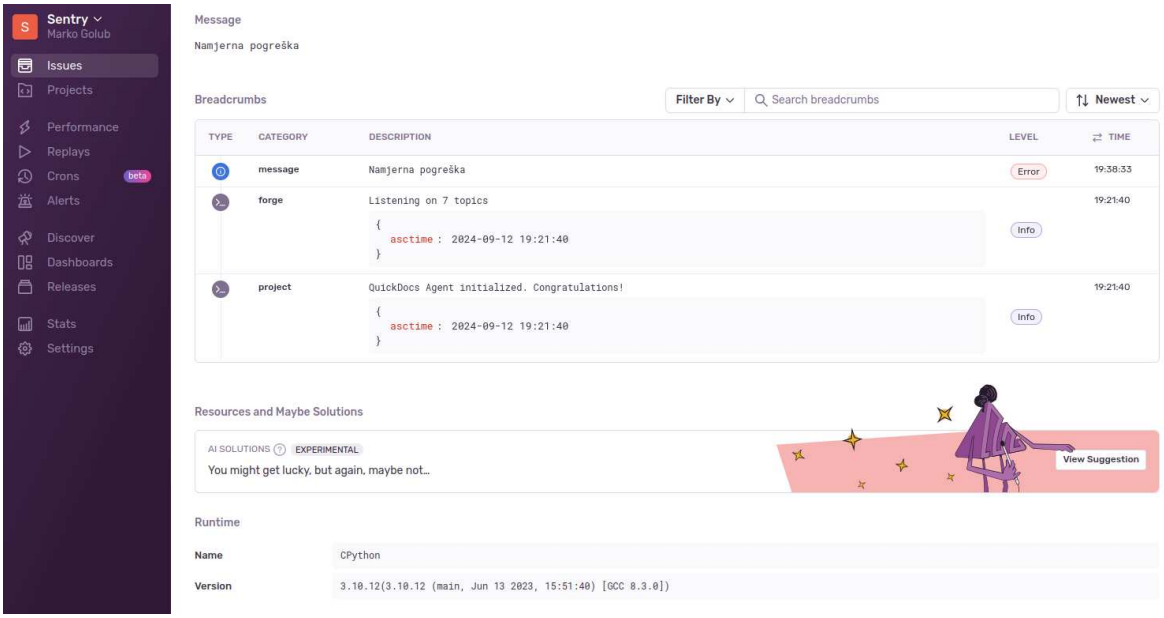
Treći korak integracije kroz koji nas AI agent vodi je podešavanje sustava za obavještavanje o pogreškama. Razgovor je predstavljen u nastavku na slici 5.9.



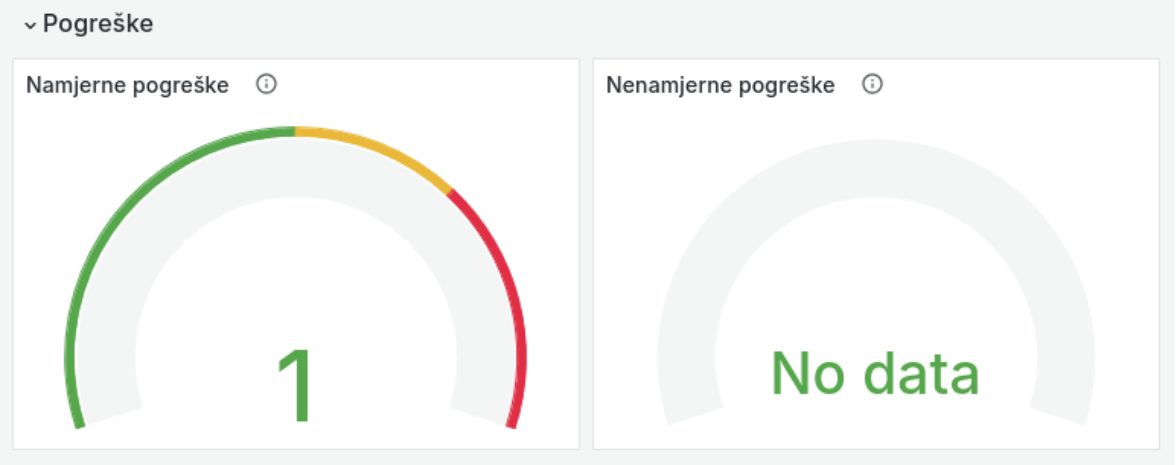
Slika 5.9. – Razgovor oko trećeg koraka integracije

Stvaranjem namjerne pogreške pokreće se automatizirani proces slanja e-pošte korisniku koji obavještava o pogrešci i poveznicom na Sentry radnu površinu, skupljanja informacija o sustavu i samoj pogrešci unutar sustava Sentry, povećanja brojača metrika pogreške i konačno ispisa pogreške unutar samog sustava. Sva četiri događaja prikazana su na slikama označenima brojevima 5.10. - 5.13.

Slika 5.10. – Obavijesti o namjernoj pogrešci kroz e-poštu



Slika 5.11. – Radna površina sustava Sentry koja prikazuje detalje namjerne pogreške



Slika 5.12. – Grafički prikaz sustava Grafana; brojač pogrešaka

```
[2024-09-12 21:38:33.931] [INFO]: Dogodila se pogreška tipa: Namjerna.
[2024-09-12 21:38:33.932] [INFO]: Pogreška: id='rn5qlvmo3ZgtYnXp' timestamp=datetime.datetime(2024, 9, 12, 19, 38, 33, 523971,
```

Slika 5.13. – Prikaz ispisa sustava nakon stvaranja namjerne pogreške

Jednako tako, na sva četiri izvora informacija jasno se vidi kako je pogreška uzrokovana namjerno. Potvrdom očekivane ispravnosti toka ponašanja sustava, dolazimo do zadnje provjere uspješnosti integracijskog procesa - funkcionalnog

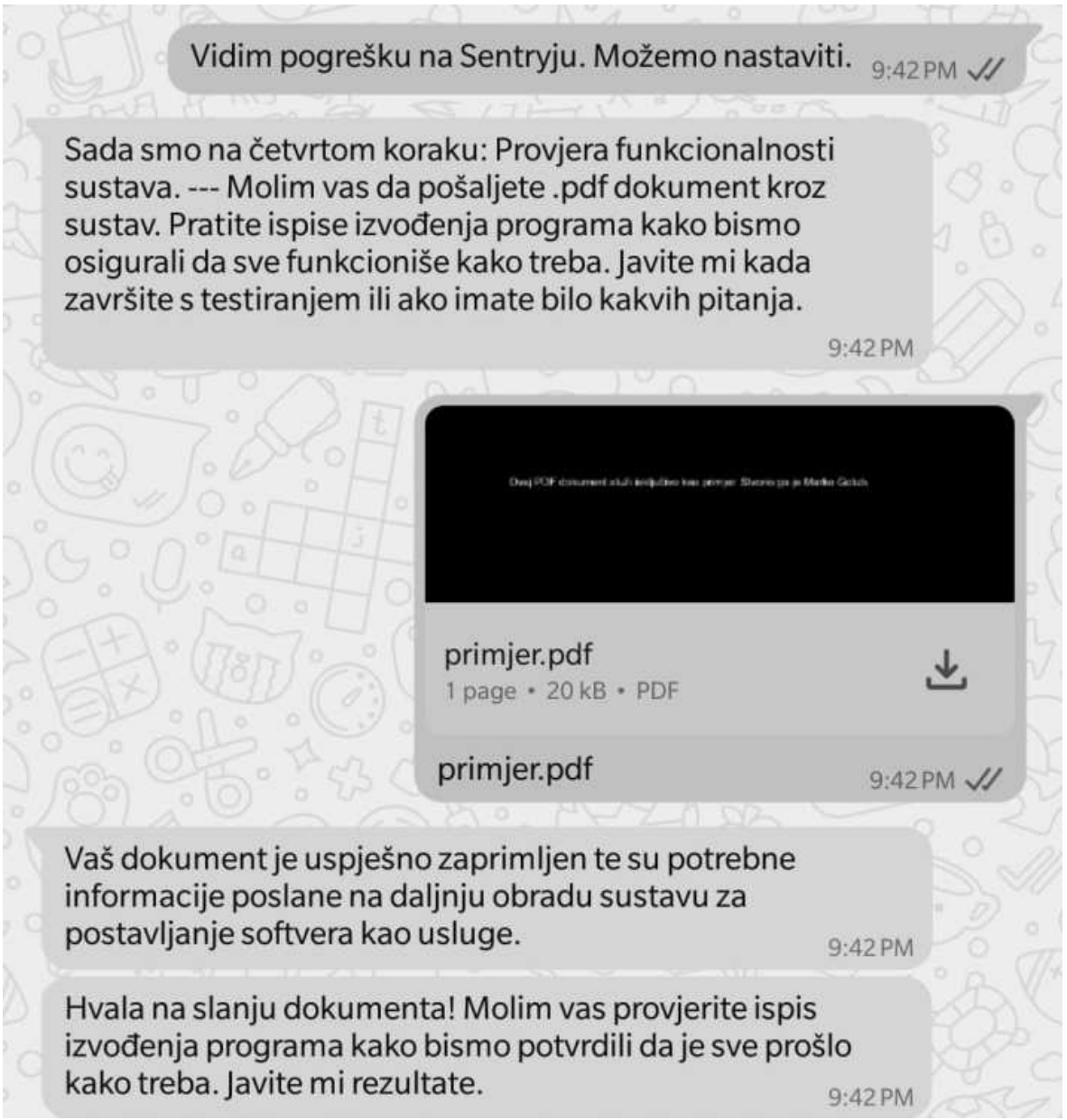
testiranja. U ovom koraku ključno je upotrijebiti neku od osnovnih funkcionalnosti sustava koji se integriraju. U sklopu ovog rada, korišten je primjer sustava za obradu PDF datoteka. Takva funkcionalnost razvijena je unutar samog sustava na komponenti “External Agent” na mjestu označenom za integraciju s vlastitim rješenjem. Primjer se temelji na operacijama programskog alata pyPDF. Sam programski kod, kao i programski komentari korišteni za oznake i upute budućim razvojnim inženjerima prikazani su u tablici 5.1.

```
@on_event(FunctionalityTestEvent)
async def functionality_test(self, event:
FunctionalityTestEvent):
   logger.info(f"Uspješno su zaprimljeni
               podaci za funkcionalno testiranje.")
   data = {}
   # ------------------------------------
   # IMPLEMENTIRAJTE SVOJU LOGIKU OVDJE; API pozivi
   # primjer: data = MojSustavAPI().get_data()
   # za demonstraciju ce se koristiti primjer obrade PDF dokumenta

   import requests
   response = requests.get(event.data['url'])
   if response.status_code == 200:

       file_name = "example.pdf"
       with open(file_name, 'wb') as file:
           file.write(response.content)
       print(f"PDF uspješno preuzet i spremljen kao:
             {file_name}")

       from pypdf import PdfReader
       reader = PdfReader(file_name)
       extracted_text = "".join([page.extract_text() for page in
                                 reader.pages])
       if extracted_text:
           data = {
               "extracted_text": extracted_text
           }
   else:
       print(f"Neuspješno preuzimanje PDF dokumenta. Kod
             pogreške: {response.status_code}")
   # ------------------------------------
   FunctionalityTestDataCreatedEvent(
       data=data,
       rm_id=event.data['rm_id']
```

```
    ).emit()
```

Tablica 5.1. – Programski kod napisan za funkcionalno testiranje

Koraci koje je AI agent poslao prikazani su na slici 5.14.



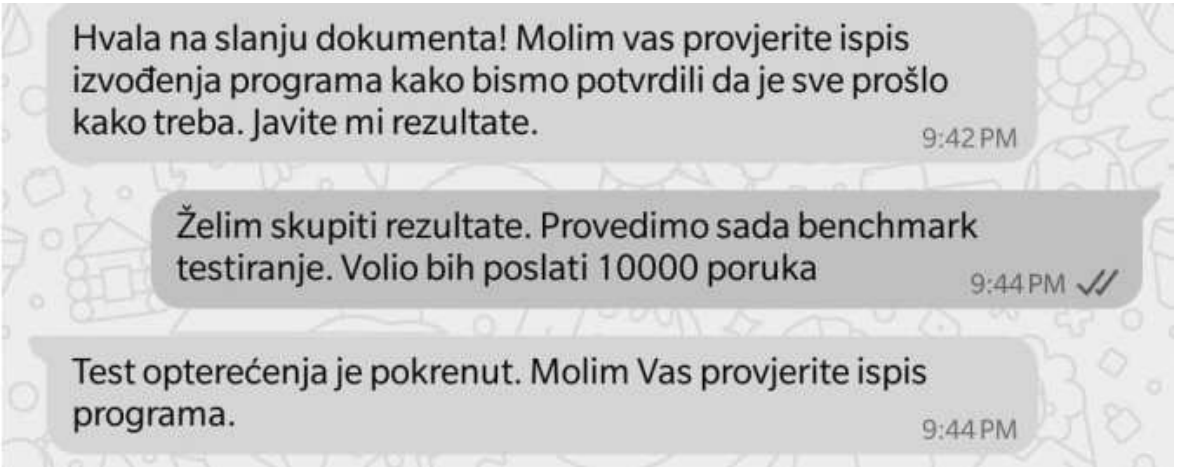
Slika 5.14. – Dio razgovora s AI agentom vezan za funkcionalno testiranje

Uspješno izvođenje zadnjeg koraka integracije programske podrške kao usluge vidi se i unutar ispisa izvođenja programa na slici 5.15.

```
[2024-09-12 21:33:38.319] [INFO]: Vrijeme potrebno za odgovor u sekundama: 0.0637369155883789
[2024-09-12 21:38:33.931] [INFO]: Dogodila se pogreška tipa: Namjerna.
[2024-09-12 21:38:33.932] [INFO]: Pogreška: id='rn5qlvmo3ZgtYnXp' timestamp=datetime.datetime(
[2024-09-12 21:42:57.516] [INFO]: Uspješno su zaprimljeni podaci za funkcionalno testiranje.
PDF uspješno preuzet i spremljen kao: example.pdf
[2024-09-12 21:42:58.713] [INFO]: Uspješno je završen test funkcionalnosti!
```

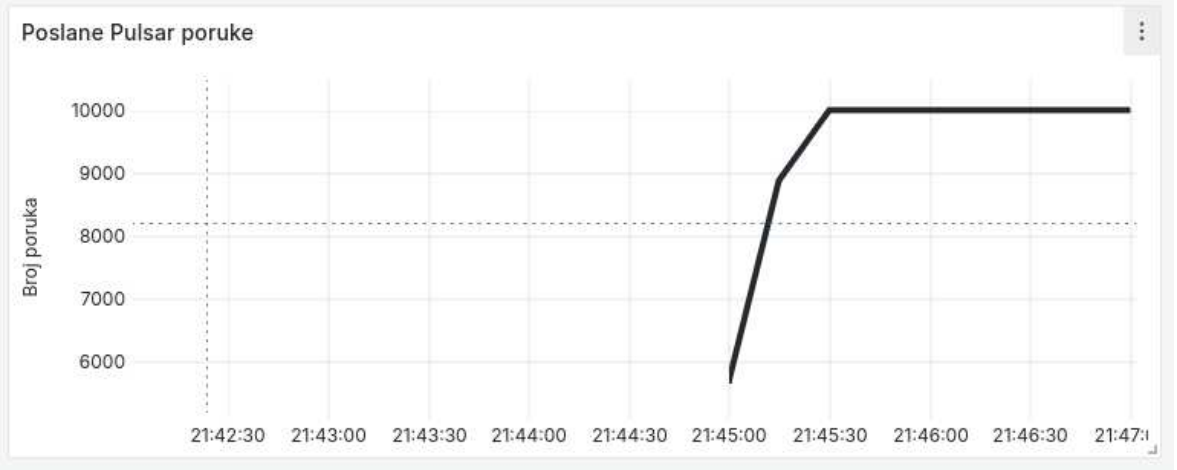
Slika 5.15. – Prikaz ispisa sustava nakon funkcionalnog testiranja

Završetkom integracije programske podrške kao usluge, provedeno je i zadnje testiranje: benchmark. Samo testiranje aktivirano je kroz razgovor s AI agentom koji je prikazan je na slici 5.16.

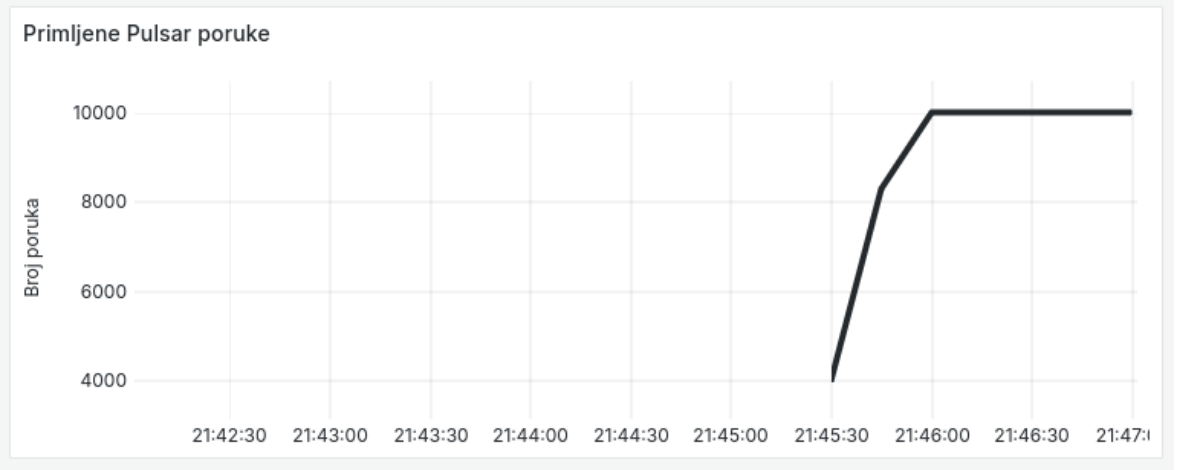


Tablica 5.16. – Pokretanje benchmark testiranja

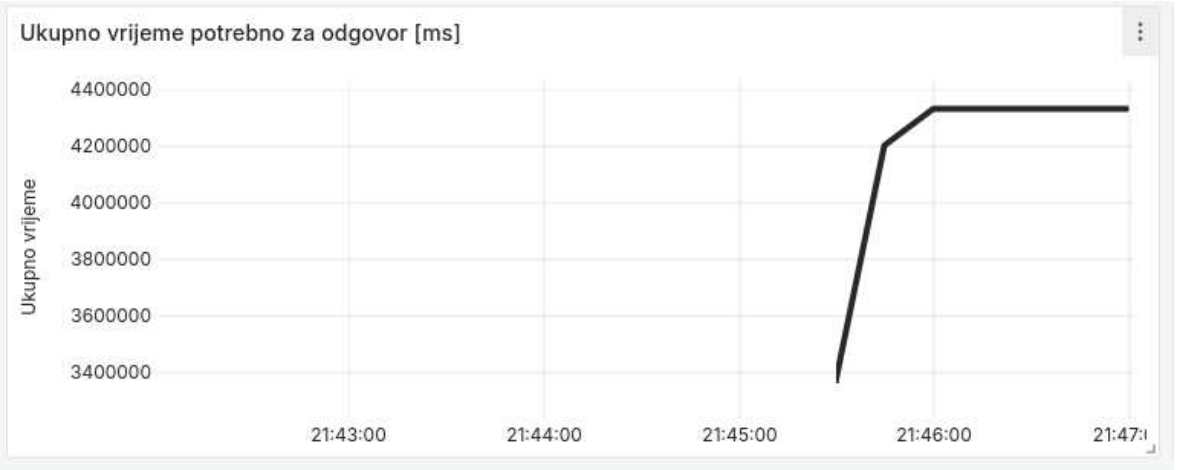
Rezultati benchmark testiranja također je moguće analizirati kroz Grafanu. Slike 5.17. - 5.20. prikazuju podatke skupljene tokom testiranja.



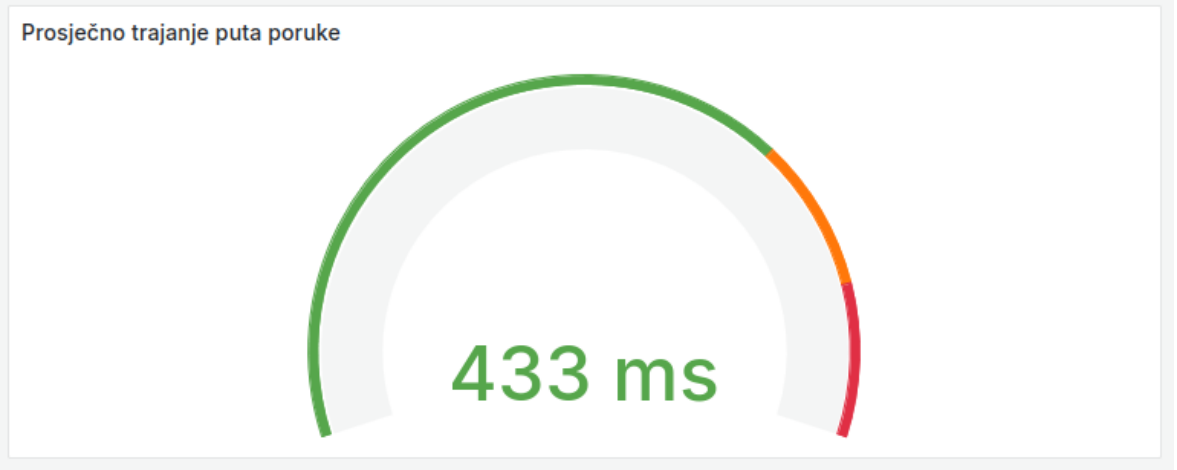
Slika 5.17. – Grafički prikaz sustava Grafana; poslane Pulsar poruke tokom benchmark testiranja



Slika 5.18. – Grafički prikaz sustava Grafana; primljene Pulsar poruke tokom benchmark testiranja



Slika 5.19. – Grafički prikaz sustava Grafana; ukupno vrijeme potrebno za odgovor izraženo u milisekundama tokom benchmark testiranja



Slika 5.20. – Grafički prikaz sustava Grafana; prosječno trajanje puta poruke tokom benchmark testiranja

Zaključak

Integracija umjetne inteligencije u procesu postavljanja programske podrške kao usluge predstavlja značajan korak naprijed u pojednostavljivanju i osiguravanju usvajanja platformi temeljenih na oblaku. Automatiziranjem ključnih radnji ovoga procesa, pružanjem povratnih informacija u stvarnom vremenu i osiguravanjem skalabilnih performansi, ovaj sustav integracije rješava mnoge izazovne točke integracije. Povećanim brojem preseljenja usluga u oblak, rješenja poput ovog postat će sve vrijednija u osiguravanju glatkih, sigurnih i jednostavnih integracija. S daljnjim napretkom u svijetu umjetne inteligencije, potencijal za još veće razine automatizacije i optimizacije je golem. Nastavkom usavršavanja spomenutih područja, sustav za integraciju programske podrške kao usluge potpomognut umjetnom inteligencijom postat će ključan alat u budućnosti integracije platformi SaaS.

Razvoj sustava potpomognutih umjetnom inteligencijom predstavlja učinkovit način rješavanja izazova integracije poslovnih sustava u platforme temeljene na oblaku. Automatizacijom ključnih koraka integracije kao što su osiguravanje sigurne komunikacije, integracija podataka i testiranje performansi, sustav ne samo da smanjuje vrijeme i trud potrebne za integraciju, već i smanjuje rizik od ljudske pogreške.

Iako je sustav automatiziran, automatizacija nije potpuna. Postoje ograničenja koja zahtijevaju ljudsku intervenciju kao i izazovi same obrade teksta i višejezičnosti kada je riječ o umjetnoj inteligenciji. Također, nedostatak ovakvog rješenja je izazov vezan vezan za “izmišljanje” ispravnih odgovora AI agenata na korisničke upite. Jedan od prijedloga za okvir budućeg rada su automatizacija izrade korisničkih računa na vanjskim sustavima (Grafana, Sentry). Drugi je uvođenje kontrole točnosti odgovora samih AI agenata. Konačno, prebacivanje rješenja u oblik samostalne web usluge gdje bi AI agent vodili korisnika na poveznice i usmjeravali mu pažnju na stvari koje žele istaknuti može dodatno olakšati postupak integracije.

Literatura

[1] Fortune Business Insights (2024.), *Software as a Service (SaaS) Market Size, Share & Industry Analysis, By Deployment Type (Public, Private, and Hybrid), By Application (Customer Relationship Management (CRM), Enterprise Resource Planning (ERP), Content, Collaboration & Communication, BI & Analytics, Human Capital Management, and Others), By Enterprise Type (Large Enterprises and SMEs), By Industry (IT & Telecom, BFSI, Retail & Consumer Goods, Healthcare, Education, Manufacturing, and Others), and Regional Forecast, 2024-2032*

[2] Wonil Kim ,Joo Hwan Lee, Chuleui Hong, Changhee Han Hanku Lee, Bongshik Jang (2012.), *An innovative method for data and software integration in SaaS*.

[3] Nico Ebert, Kristin Weber, Stefan Koruna (2017.), *Integration Platform as a Service*

[4] Navajeevan Pushadapu (2023.), *AI-Driven Solutions for Seamless Integration of FHIR in Healthcare Systems: Techniques, Tools, and Best Practices*

[5] Olusola Bamisile, Humphrey Adun, Dongsheng Cai, Sandra Obiora (2023.), *Bibliographical progress in hybrid renewable energy systems' integration, modeling, optimization, and artificial intelligence applications: A critical review and future research perspective*

[6] Rajesh H. Kulkarni (2016.), *Integration of artificial intelligence activities in software development processes and measuring effectiveness of integration*

[7] Geetanjali Katare, Gourish Padihar, Zeba Qureshi (2018.), *Challenges in the Integration of Artificial Intelligence and Internet of Things*

[8] Pradeep Kumar Dhoopati (2023.), Enhancing Enterprise Application Integration through Artificial Intelligence and Machine Learning

[9] Ken Jaroenchisakon, Iha Chikkala, Madhav SBSS, Boomi, (2024.) *AI Agents: From Automation to Autonomous Actions*

[10] Tabisa Ncubukezi (2022.), *Human Errors: A Cybersecurity Concern and the Weakest Link to Small Businesses*

[11] Hebert Cabane, Kleinner Farias (2024.), *On the impact of event-driven architecture on performance: An exploratory study*

[12] Chatterjee, Sheshadri; Ghosh, Soumya Kanti; Chaudhuri, Ranjan; Nguyen, Bang (2019), *Are CRM systems ready for AI integration?*

[13] Jerry Banks, John S. Carson, Barry L. Nelson, David M. Nicol (2013.), *Discrete-Event System Simulation: Pearson New International Edition*, Pearson Higher Ed

[14] Erik Svensson, Claus Vetter, Thomas Werner (2004.), Data Consistency in a Heterogeneous IT Landscape: A Service Oriented
Architecture Approach

[15] OpenAI, pristupljeno 16. rujna 2024. Poveznica: https://platform.openai.com/docs/introduction

[16] Valtteri Andström (2024.), A Comparative Analysis of Apache Kafka
and Apache Pulsar

[17] Md Showkat Hossain Chy, Muhammad Ashfakur Rahman Arju, Sri Manjusha Tella, Tomas Cerny (2023.), Comparative Evaluation of Java Virtual Machine-Based Message Queue Services: A Study on Kafka, Artemis, Pulsar,
and RocketMQ

[18] JWT, Introduction to JSON Web Tokens, pristupljeno: 13. rujna 2014. Poveznica: https://pulsar.apache.org/docs/3.3.x/security-jwt/

[19] Pulsar, Authentication using tokens based on JSON Web Tokens, pristupljeno: 13. rujna 2014. Poveznica: https://pulsar.apache.org/docs/3.3.x/security-jwt/

Sažetak

Naslov: Sustav potpomognut umjetnom inteligencijom za postavljanje sustava softvera kao usluge

Sažetak: Ovaj rad istražuje upotrebu umjetne inteligencije za automatizaciju postavljanja softvera kao usluge. Tradicionalni procesi integracije SaaS sustava su spori i zahtijevaju značajnu ljudsku intervenciju, što povećava troškove i rizik pogrešaka. U radu je opisan sustav koji koristi AI agente, Apache Pulsar za komunikaciju vođenu događajima, Prometheus i Grafanu za praćenje performansi i prikupljanje podataka te Sentry za obavještavanje korisnika o pogreškama tokom izvođenja. Sustav omogućava automatizirano vodstvo potpomognutom umjetnom inteligencijom o integraciji programske podrške kao usluge, praćenje ključnih metrika i alarmni sustav. Rezultati pokazuju koliko ovakva automatizacija čini integraciju učinkovitim i brzim procesom.

Ključne riječi: umjetna inteligencija, SaaS, automatizacija, integracija, Apache Pulsar, Prometheus, Grafana, Sentry

Summary

Title: Artificial intelligence system for onboarding on software as a service solutions

Summary: This paper explores the use of artificial intelligence to automate the integration with Software as a Service (SaaS) platforms. Traditional SaaS integration processes are slow and require significant human intervention, increasing costs and the risk of errors. The paper describes a system that utilizes AI agents, Apache Pulsar for event-driven communication, Prometheus and Grafana for performance monitoring and data collection, and Sentry for user notifications about execution errors. The system enables AI-assisted guidance for SaaS deployment, monitoring of key metrics, and an alerting system. The results demonstrate how such automation makes the integration process efficient and fast.

Keywords: artificial intelligence, SaaS, automation, integration, Apache Pulsar, Prometheus, Grafana, Sentry

Skraćenice

AI	Umjetna inteligencija	(eng. Artificial Intelligence)
API	Sučelje za programiranje aplikacija	(eng. Application Programming Interface)
CAGR	Ukupna godišnja stopa rasta	(eng. Compound annual growth rate)
CRM	Upravljanje odnosima s klijentima	(eng. Customer Relationship Management)
CRUD	Kreiranje, čitanje, ažuriranje i brisanje	(eng. Create, Read, Update, Delete)
ERP	Planiranje resursa poduzeća	(eng. Enterprise Resource Planning)
IMS	Usluga trenutne razmjene poruka	(eng. Instant Messaging Service)
IoT	Internet stvari	(eng. Internet of Things)
JWT	JSON Web Token	
LLM	Veliki jezični model	(eng. Large Language Model)
ML	Strojno učenje	(eng. Machine Learning)
RBAC	Kontrola pristupa temeljena na ulogama	(eng. Role-Based Access Control)
SaaS	Softver kao usluga	(eng. Software as a Service)
UX	Korisničko iskustvo	(eng. User Experience)

Privitak

Upute za korištenje programske podrške

Sustav za integraciju programske podrške kao usluge potpomognut umjetnom inteligencijom dizajniran je tako da bude jednostavan za korištenje uz zadržavanje tehničke složenosti potrebne za sigurnu i učinkovitu integraciju sa platformama SaaS. Ovaj privitak opisuje praktične korake za podešavanje rješenja i vanjskih sustava.

Korak 1: Preduvjeti i ovisnosti

Prije početnog pokretanja sustava, korisnici trebaju imati instalirane sljedeće preduvjete:

- Python 3.10.x
- pip
- prometheus

Sve ovisnosti mogu se instalirati pomoću datoteke zahtjeva (naziv datoteke “requirements.txt”) izvodeći sljedeću naredbu zapisanu u tablici P1.

```
pip install -r requirements.txt
```

Tablica P1. – Naredba za instaliranje zahtjeva zapisanih u datoteci

Sadržaj `requirements.txt` datoteke sadrži nazive ovisnosti. Unutar njega su ovisnosti definiranih verzija prikazanih u tablici P2.

Paket	Verzija
forge-sdk	6.0.0b14
pulsar-client	3.5.0
relation-manager-api	6.0.0b107

intent_recognizer_api	6.0.0b6
prometheus-client	0.17.1
sentry-sdk	1.45.0
pypdf	4.3.1
requests	

Tablica P2. – Sadržaj datoteke `requirements.txt`

Također, potrebno je dodati vrijednosti u `.env` datoteku kao što je navedeno u tablici P3.

```
PROJECT_SLUG=saas-integration
ENVIRONMENT_NAME=main
```

Tablica P3. – Sadržaj `.env` datoteke

Korak 2: Podešavanje Apache Pulsar servisa i razmjena certifikata i tokena
Ovaj je korak ključan za komunikaciju, upravljanje slanjem i primanjem poruka tijekom procesa integracije. Kako bi se osigurala sigurna komunikacija između korisnika i platforme SaaS, sustav zahtijeva razmjenu Apache Pulsar certifikata i tokena. Certifikat i token osiguravaju autentificiran identitet korisnika prije izvedbe bilo kakve operacije. Apache Pulsar podržava autentifikaciju klijenata koristeći sigurnosni token zasnovan na tehnologijama JSON Web tokena (RFC-7519) [18] uključujući i algoritme podržane unutar Java JWT alata. Token je vjerodostojna poveznica na jedinstvenog korisnika. Udruživanje se vrši putem uloge te se u slučaju JWT tokena obično se odnosi izvršitelja radnji. Token je moguće koristiti za identifikaciju Apache Pulsar klijenta i povezivanje sa korisnikom kojem je dopušteno raditi određene radnje, odnosno kojemu su dodijeljene određene uloge, kao što je objavljivanje poruka u temi ili čitanje poruka iz teme.

JWT autentifikacija podržava dva načina izrade i potvrde tokena:

1. Simetričan način - jedinstven tajni ključ
2. Asimetričan način - par ključeva koji uključuje privatan ključ za generiranje tokena i javni ključ za validaciju tokena

Svi koraci izrade i potvrde tokena navedeni su u tablicama P4. - P12. te se zasnivaju na službenoj dokumentaciji [19]. Administrator sustava dužan je stvoriti tajni ključ i koristiti ga za stvaranje tokena klijenta korištenjem iduće naredbe:

```
bin/pulsar tokens create-secret-key --output moj-tajni-kljuc.key
```

Tablica P4. – Prvi korak izrade i potvrde tokena

Jednako tako, moguće je predati i apsolutni put koristeći naredbu:

```
bin/pulsar tokens create-secret-key --output
/putanja/do/moj-tajni-kljuc.key
```

Talica P5. – Drugi korak izrade i potvrde tokena

kao i generirati ključ zaštićen enkripcijom base64:

```
bin/pulsar tokens create-secret-key --output moj-tajni-kljuc.key
--base64
```

Tablica P6. – Treći korak izrade i potvrde tokena

Kako bi se koristio asimetričan ključ za enkripciju, potrebno je stvoriti par ključeva koristeći naredbu:

```
bin/pulsar tokens create-key-pair --output-private-key
moj-privatni-kljuc.key --output-public-key moj-javni-kljuc.key
```

Tablica P7. – Četvrti korak izrade i potvrde tokena

koja će stvoriti datoteku kao rezultat izvršavanja unutar direktorija gdje je Apache Pulsar instaliran. Potrebno je spremiti `moj-privatni-kljuc.key` na sigurnu lokaciju te se pobrinuti da je isključivo administratori sustava koriste prilikom stvaranja novih tokena. Javni ključ, `moj-javni-kljuc.key` potrebno je dostaviti svim Apache Pulsar brokerima. Dozvoljena je i javna objava ovog ključa bez potrebe za brigom oko sigurnosti.

Jednom kada imamo stvorene ključeve, potrebno je stvoriti tokene. Iduća naredba ispisuje token na zadani izlazni tok u računalnom programu (eng. stdout):

```
bin/pulsar tokens create --secret-key \
file:///putanja/do/moj-tajni-kljuc.key --subject test-user
```

Tablica P8. – Peti korak izrade i potvrde tokena

Izrada tokena obavlja se predavajući “privatni” ključ koristeći iduću naredbu:

```
bin/pulsar tokens create --private-key \
file:///putanja/do/moj-privatni-kljuc.key --subject test-user
```

Tablica P9. – Šesti korak izrade i potvrde tokena

Preporučeni način izrade tokena je onaj koji koristi i dodatne vrijednosti za vrijeme dugovječnosti tokena. Takav token generira se na idući način:

```
bin/pulsar tokens create --secret-key \
file:///putanja/do/moj-tajni-kljuc.key \
            --subject test-user \
            --expiry-time 1y
```

Tablica P10. – Sedmi korak izrade i potvrde tokena

Kako bi se omogućila JWT autentifikacija, potrebno je podesiti postavke na brokeru unutar `conf/broker.conf` i `conf/proxy.conf` datoteka:

```
authenticationEnabled=true
authenticationProviders=org.apache.pulsar.broker.authentication.A
uthenticationProviderToken

brokerClientAuthenticationPlugin=org.apache.pulsar.client.impl.au
th.AuthenticationToken
brokerClientAuthenticationParameters={"token":"pr1mjer.Tok3na"}
brokerClientAuthenticationParameters=file:///putanja/do/tokena

# Ako se koristi tajni ključ
# tokenSecretKey=file://putanja/do/tajniKljuc.key
# Ako se koristi javni/privatni ključ
# tokenPublicKey=file://putanja/do/javniKljuc.key
```

Tablica P11. - Osmi korak izrade i potvrde tokena

U sklopu ovog rada, koristit će se javno dostupan Apache Pulsar servis. Kao korisnik sustava, sve što je potrebno je podesiti Apache Pulsar token i putanju do certifikata u `.env` datoteku na idući način prikazan u tablici P12 koristeći informacije dodijeljene od pružatelja usluge.

```
PULSAR_URL=pulsar+ssl://pulsar.apps.pr1mjer.Url
PULSAR_TLS_CERT_PATH=./pulsar-cert.pem
PULSAR_AUTH_TOKEN=pr1mj3r.G3n3r1r4n0g.T0k3n4
```

Tablica P12. – Deveti korak izrade i potvrde tokena