Google App Engine

Štefančić, Antonio

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Antonio Štefančić

GOOGLE APP ENGINE

ZAVRŠNI RAD

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Antonio Štefančić

Matični broj: 42667/13-IZV

Studij: Informacijski sustavi

GOOGLE APP ENGINE

ZAVRŠNI RAD

Mentor:

Doc. dr. sc. Darko Andročec

Varaždin, svibanj 2022.

Antonio Štefančić

Izjava o izvornosti

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

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

U ovom radu proučava se jedan od najznačajnijih servisa računarstva u oblaku, Googleov servis Google App Engine, skraćeno GAE. Kroz poslovnu perspektivu se tijekom cijelog rada važu prednosti i nedostaci razvoja u oblaku, analiziraju se i uspoređuju različiti modeli pružanja usluga te se daje detaljniji opis platforme kao servisa (PaaS). Glavna srž tematike, Google App Engine, s vremenom se razvio u globalno prepoznati servis sa jako širokim spektrom mogućnosti razvoja, iako i dalje podržava razvoj samo u određenom skupu programskih jezika i okruženja. U praktičnom dijelu rada demonstrira se postavljanje već kreirane aplikacije na App Engine te funkcionalnosti te aplikacije, koja je razvijena u jeziku Python i razvojnom okruženju Flask te koristi DataStore na samoj platformi.

Ključne riječi: računarstvo u oblaku, Google App Engine, PaaS, Python, Flask

SADRŽAJ

1. Uvod

Jedna od novijih paradigmi koja je svoj začetak ali i vrtoglavi uspon doživjela u 21. stoljeću jest paradigma oblaka. Ispravnije bi bilo koristiti termin računarstva u oblaku jer nisu podaci ključni u toj paradigmi, već procesi koji se odvijaju nad tim podacima na udaljenim poslužiteljima koji su prosječnom korisniku lako dostupni.

U ovome radu bit će obrađen jedan od najsofisticiranijih servisa u oblaku, Googleov servis koji i manje i velike organizacije koriste u svojem poslovanju, a vrlo je vjerojatno da će ga u budućem razdoblju početi koristiti i one organizacije koje to još uvijek ne čine. Radi se o Google App Engineu, skraćeno GAE, koji je osmišljen iz razloga da organizacije, ali i pojedince, oslobodi tereta upravljanja sklopovskom i softverskom infrastrukturom.

Iako ovaj servis ima besplatnu inačicu, ona je ograničena resursima koje vam Google daje na raspolaganje. GAE je ograničen utoliko što nameće platformu za razvoj pa tako aplikacije mogu biti pisane u Javi, PHP-u, Pythonu, Rubyju, ali ne mogu biti pisane u programskim jezicima kao što su C#, C++ i drugima.

Osnovna pretpostavka App Enginea nalazi se u tome da korisnik, bilo to organizacija ili pojedinac, može s lakoćom razvijati aplikacije, misleći samo na procese i podatke koje stvara, a brigu i odgovornost za tehnička pitanja prepustiti samom Googleu. Tako korisnik pristupa oblaku sa svojim pristupnim podacima neovisno gdje se u datom trenutku nalazio, računajući da usluga koju koristi je sigurna, stabilna i jednostavna za korištenje.

Kroz teorijsku podlogu koja se prožima većim dijelom ovog rada, steći će se uvid u koncept platforme kao servisa, a kasnije će se u radu na praktičnom primjeru demonstrirati kako se određena aplikacija može prenijeti na oblak, konkretno na Google App Engine, i koristiti na njemu.

Konačni cilj ovog rada bio bi zainteresirati radoznale pojedince o prednostima koje jedan servis kao što je GAE nudi te potaknuti širenje razvoja na oblak kako bi se što učinkovitije koristio dostupni mrežni prostor, sklopovska infrastruktura i smanjila već povelika potrošnja energije i nedovoljna iskoristivost elektroničke opreme u svijetu.

2. Računarstvo u oblaku

Računarstvo u oblaku (engl. *cloud computing)* jedan je od novijih pojmova u posljednjem desetljeću koji je uzeo velikog maha u tehnološkom svijetu. Osnovna ideja računarstva u oblaku je da se računalni resurs može iznajmiti po potrebi. Računalni resursi mogu biti različiti: poslužitelji, usluge, aplikacije ili nešto drugo. U današnje vrijeme, kada trebamo sve više računalnih resursa, a ponajviše zbog pojave vršnog opterećenja te izuzetne važnosti skalabilnosti kod sustava i aplikacija, bitno je da se resursi mogu unajmiti bez potrebe za interakcijom s osobljem i bez velikih ulaganja u infrastrukturu, barem sa strane onoga koji taj resurs planira koristiti po potrebi. Resurs se koristi dok su usluge, aplikacije ili sustavi za koje je resurs namijenjen aktivni, a resurs se otpušta ili prestaje koristiti kada usluge, aplikacije ili sustavi prestanu raditi ili se nadograde na neki novi resurs.

Velika prednost kod ovakvog principa je u tome da se korisnik ne mora brinuti hoće li ga skupo stajati plaćena oprema (on unajmljuje resurs, a za opremu se brine druga strana), tko će ju održavati te hoće li ona biti maksimalno iskorištena s obzirom na njene troškove (korisnik nema ulaganja u kapitalnu infrastrukturnu opremu, tj. poslužitelje, mrežne i strujne kapacitete i posebne poslužiteljske prostorije, već samo snosi operativne troškove razmjerno svojem korištenju konkretnih resursa) [1].

Prema Čorak [1, str. 2], računarstvo u oblaku može se definirati kao „sustav u kojem je moguće iznajmljivanje jednog ili više poslužitelja i pokretanje različitih programa na njima; virtualnog poslužitelja na kojemu korisnici mogu pohranjivati podatke i po volji im pristupati; korištenje programa koji se nalazi na internetu i pohranjuje te štiti podatke za vrijeme pružanja usluge; korištenje mnoštva internet programa uz pomoću kojih je moguće integrirati fotografije, karte, GPS informacije i druge korisne stvari“.

2.1. Značajke računarstva u oblaku

Koncept računarstva u oblaku povlači sa sobom problematiku oblaka. U ovome smislu pojam oblaka koristi se kako bi se opisali procesi koji se odvijaju na nekim podacima na udaljenim poslužiteljima. Ti poslužitelji mogu se nalaziti tisuće kilometara udaljeni od korisnika, a korisnik gotovo trenutačno komunicira s njima uz pomoć svojih uređaja pomoću stabilne i brze internetske veze. Upravo takva internetska veza te globalno povezana mreža računala omogućila je da jedan ovakav koncept „zaživi“ i postane ključan u poslovanjima mnogih današnjih organizacija. Korisnik ili organizacija koji/a koristi usluge računarstva u oblaku ne brine niti održava poslužitelje, a usluge može koristiti kad to želi te jednako tako ih isključiti bez ikakvih dodatnih troškova i u bilo kojem trenutku.

Ključna osobina kod ovog koncepta krije se u činjenici da korisnik (odnosno organizacija) nije ni na koji način izravno povezan sa računalnim sklopovljem (engl. *hardware*), koje mu omogućava usluge računarstva u oblaku te se ne mora brinuti za isti. Kompletnu brigu pritom prepušta pružatelju usluge s kojim je ugovorio uslugu, a on vodi brigu o programima, odnosno aplikacijama na kojima radi i eventualno podacima kojima želi upravljati kroz taj program ili aplikaciju kojeg/ju planira postaviti na oblak [1].

Prije nego bi korisnik (ili organizacija) odlučio staviti svoje programe u oblak, bitno mu je uvjeriti se da će mu usluga koju traži biti sigurna, jednostavna za korištenje, stabilna i uvijek dostupna. Osim toga, postoje i mnogi drugi razlozi zašto bi se odlučio koristiti usluge oblaka, a neki od njih su:

- **ušteda novca i vremena i osoblja**,
- **plaćanje proporcionalno potrošnji**,
- **korištenje zajedničkih resursa**,
- **skalabilnost** i
- **sigurnost**.

Kad korisnik pojedinac ili neka organizacija odluče koristiti usluge računarstva u oblaku, oni uvijek umanjuju svoje troškove rada, vremena i opreme. Tako primjerice organizacija koja odluči razvijati neki svoj informatički proizvod u oblaku ne ulaže u dodatnu računalnu opremu, osoblje koje će provesti određeno vrijeme i uložiti neki trud u održavanje te opreme te smanjuje odgovornost ukoliko bi došlo do nekog kvara ili nekonzistentne usluge koja tada prelazi na pružatelja usluge računarstva u oblaku. Stoga je jedna od najvećih prednosti upravo ušteda novca i vremena, ali i dobivanje na brzini kojom se može isporučiti ili omogućiti određeni proizvod ili usluga.

Druga bitna prednost očituje se u jednostavnoj frazi – „koliko koristiš, toliko i platiš“. Korisnik bira koje će i koliko resursa koristiti i u kojem vremenskom razdoblju te ovisno o tim postavkama, plaća usluge u ovisnosti o vremenu i obujmu u kojem ih koristi. Sustavi u oblaku automatski provjeravaju koji se resursi koriste, optimiziraju njegovo korištenje te ga oslobađaju po potrebi. Svaki korisnik (ili organizacija) može pratiti uporabu svakog svojeg resursa te tako resurse raspodjeljivati, smanjivati ili povećavati po potrebi te ima cjelovit uvid u ono što koristi i plaća za to korištenje.

Resursi (mrežni prostor, procesorska moć, memorija, mrežna propusnost i dr.) nisu ograničeni i ne treba ih se trošiti bespotrebno i nelogično. Upravo zbog toga kod koncepta računarstva u oblaku resursi se udružuju kako bi poslužili što više korisnika i što manje vremena stajali neaktivni. Postoji više modela, a jedan od njih je model s više zakupljenih jedinica (engl. *multi-tenant model*) kod kojeg se različiti fizički i virtualni resursi dinamički zauzimaju i oslobađaju po zahtjevima korisnika [1].

Vrlo važna značajka je i skalabilnost. Broj poslužitelja potreban za posluživanje korisnika smanjuje se ili povećava ovisno o potrebama, a to se većinom odvija gotovo trenutno i u radu se to ne može ni zamijetiti. I prilikom izrazito velikog opterećenja na sustav, sve se odvija fluidno i bez iznimki. Mrežni i računalni kapaciteti uvijek su dostupni i za optimizaciju njihova korištenja brine se isključivo pružatelj usluge računarstva u oblaku. Tako primjerice, ako korisnik (ili organizacija) postavi na sustav svoju aplikaciju i ona u određenom vremenskom trenutku zahtijeva znatno više računalnog ili mrežnog kapaciteta, sustav će promptno reagirati i povećati svoje kapacitete dok ta potražnja traje, ili ih smanjiti ako bi bilo obratno.

Sigurnost je također vrlo bitna značajka jer svaki korisnik ili organizacija očekuju od pružatelja usluga da će stvoriti sve preduvjete da ne dođe do nekih propusta u pružanju usluge. Zaštita podataka, sigurnosni rizici i problemi s održavanjem opreme potencijalne su opasnosti za koje je odgovoran pružatelj usluge i na koje treba računati, a koji je dobio povjerenje svojih korisnika za to.

No postoje i određeni nedostaci koje korisnik treba imati u vidu pri razvoju aplikacija na ovaj način, a neki od njih su da se kod ovakvog razvoja aplikacija ne pamte međustanja (engl. *stateless*), da se oblak često mijenja i nadograđuje pa je potrebno da korisnik učestalo prati promjene i dodaje nova proširenja funkcionalnosti oblaka svome programu, a vjerojatno najintrigantniji nedostatak vezan je s time da korisnik u realnom vremenu ne zna gdje su i na koji način njegovi podaci pohranjeni (što može izazvati problem ako dođe do promjene pojedinih lokalnih zakona o sigurnosti podataka u zemlji gdje je smješten podatkovni centar za određeni oblak kojeg korisnik koristi).

Bitno je ipak naglasiti da, iako se za oblak razvijaju aplikacije koje ne pamte međustanja, da se za tu svrhu upotrebljavaju baze podataka u samom oblaku [1].

2.1.1. Poslovna perspektiva na računarstvo u oblaku

Gledajući na koncept računarstva u oblaku iz perspektive poslovnih korisnika koji namjeravaju svoje poslovanje, odnosno usluge koje pružaju prebaciti u oblak, nailazi se na nekoliko posebno bitnih prednosti i nedostataka za takve korisnike.

Korisnici plaćaju onoliko resursa u oblaku koliko im treba umjesto iznajmljivanja uz pretplatu ili rezerviranja resursa na duže vrijeme. Takav je oblik poslovanja posebno prigodan za manje tvrtke koje tek počinju s poslovanjem i žele izbjeći nabavu skupe opreme za izvršavanje određenih poslova. Resursi se dodjeljuju na temelju potreba i prioriteta i korisnici ih mogu otpustiti čim više nemaju potrebu za njima. Često se korisnicima stvara dojam neograničenih računalnih resursa za korištenje, a to se postiže metodama virtualizacije i pametnim dodjeljivanjem resursa [2].

Kao što je sigurnost jedna od mogućih prednosti računarstva u oblaku, tako je sigurnost i veliki potencijalni problem koji takvi korisnici mogu osjetiti u svome poslovanju. Naime, korisnik svoje poslovanje, odnosno usluge koje pruža krajnjim korisnicima, smješta na tuđu infrastrukturu. Ako bi došlo do nekih problema sa strane pružatelja usluga računarstva u oblaku i infrastruktura bi postala nedostupna, to bi korisniku potencijalno izazvalo velike poslovne gubitke. Zato je vrlo bitno kod korisnika imati povjerenja u svojeg izabranog pružatelja da do takvih problema neće doći, ili ako će doći da će ih isti znati u što kraćem roku riješiti. Također je poželjno da korisnik ima osmišljeni plan (rezervnu opciju) što učiniti ako bi do takvog ili sličnog problema došlo, kako krajnji korisnici njegovih usluga ne bi izgubili pristup tim uslugama ili bili onemogućeni ih koristiti.

Postoji još jedna potencijalna opasnost, a tiče se prvenstveno krajnjih korisnika koji koriste konačnu uslugu koja je smještena u oblaku. Vrlo često takve usluge sadrže tajne, odnosno privatne podatke korisnika koji moraju biti zaštićeni od strane trećih osoba koji bi ih mogli zlouporabiti. Ako bi došlo do narušavanja sigurnosti i gubljenja podataka krajnjih korisnika, to bi moglo imati značajne posljedice za krajnjeg korisnika, korisnika koji pruža svoju uslugu u oblaku, ali i samog pružatelja računarstva u oblaku [2].

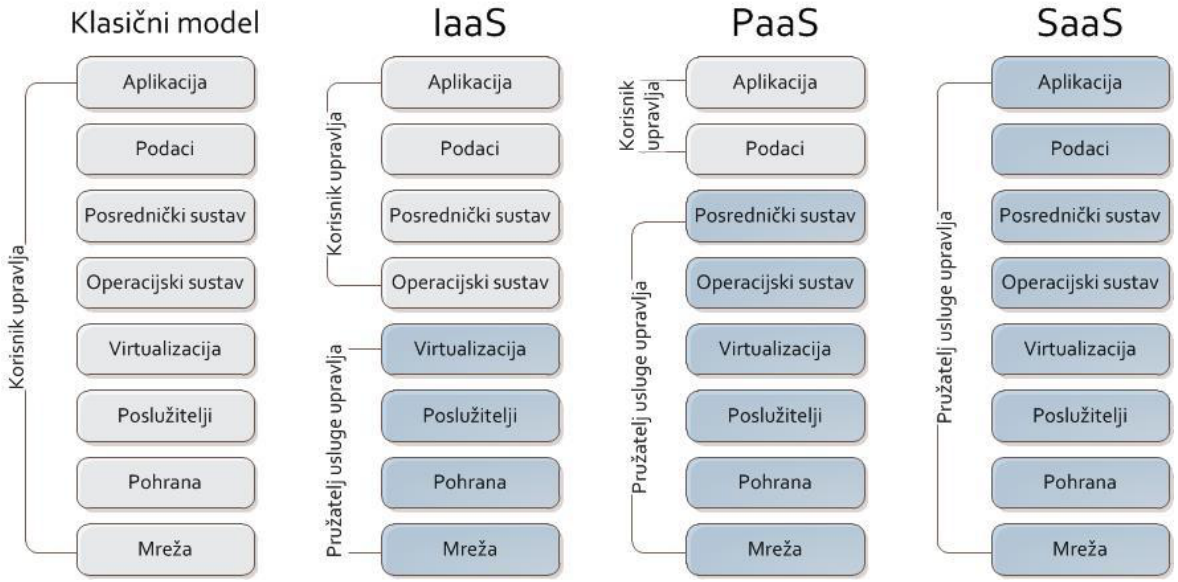
2.2. Modeli pružanja usluge računarstva u oblaku

Računarstvo u oblaku ugrubo se sastoji od tri različite razine usluga koje omogućuju korisnicima pokretanje programa i pohranjivanje podataka. Ovisno o potrebama i zahtjevima, korisnik odabire onu koja mu najviše odgovara, a razlikuju se po razini fleksibilnosti, kontrole korisnika i ponajprije obuhvatnosti same usluge.

Modeli pružanja usluge su:

- softver kao servis (engl. *software as a service – SaaS*),
- platforma kao servis (engl. *platform as a service – PaaS*),
- infrastruktura kao servis (engl. *infrastructure as a service – IaaS*).

Bitno je naglasiti da se računarstvo u oblaku razvija po modelu *sve kao usluga* (engl. *everything-as-a-service, XaaS*) te se stoga i posrednička platforma, virtualni strojevi i aplikacije koriste kao usluge u oblaku. Slika 1 prikazuje čime u određenom modelu korisnik ima pravo upravljati, odnosno koje su usluge dostupne korisniku.



Slika 1. Modeli pružanja usluge računarstva u oblaku [3, str. 3]

2.2.1. Softver kao servis – SaaS

Softver kao servis najpoznatiji je model pružanja usluge te je zaslužan za mnoge promjene koje su se dogodile u organizacijama u protekla dva desetljeća. Ovaj model omogućava uporabu dostupnih aplikacija putem interneta koji se nalaze u infrastrukturi oblaka. To znači da korisnik usluge nema potrebe za ikakvom instalacijom programskih proizvoda na svoje računalo, već aplikaciji koju treba pristupa s različitih klijentskih uređaja uz pomoć klijentskog sučelja kao što su primjerice web ili mobilni preglednik. Pružatelj usluge je pritom

odgovoran za održavanje aplikacije, a ažuriranja aplikacije se odvijaju u samom oblaku što znači da korisnik prilikom svakog pristupanja istoj koristi njenu najnoviju inačicu [1, 3].

S poslovnog aspekta gledajući, organizacije koje u svojem poslovanju koriste *SaaS* model, spremni su na manje financijske rizike jer se najam softvera u oblaku temelji na principu da koliko koristiš, toliko i platiš, odnosno da se najčešće bira opcija koja uključuje sve potrebe dostatne za određenog korisnika. Tako organizacija ne kupuje opremu i licencu kako bi se mogla koristiti nekim uslugama, već uslugu unajmljuje u oblaku za određeni period za koji je ta usluga potrebna [3, str. 5].

Bitna prednost kod ovog modela je da njegove kopije usluga koristi više korisnika (kao npr. elektronička pošta), a potencijalna mana se krije u tome što organizacija kod ovog modela mora predati kontrolu nad svojim podacima i imati povjerenje u pružatelja usluga da su ti podaci sigurni [4, str. 163].

Neki od primjera *SaaS* usluga dati su u tablici ispod (tablica 1), a najpoznatiji pružatelj *SaaS* usluga je Google sa svojim skupom aplikacija u oblaku.

Organizacija	**Aplikacija**	**Svrha i mogućnosti**
Google	*Google Dokumenti*	Kreiranje, obrada. dijeljenje i slanje dokumenata te zajednički pristup i rad više korisnika u stvarnom vremenu
	Google Kalendar	Zakazivanje sastanaka, zapisivanje bilješki, dijeljenje i objavljivanje događaja
	Gmail	Primanje i slanje e-pošte, stvaranje skica, arhiviranje te filtriranje poruka
	Google Stranice	Stvaranje, dijeljenje i objavljivanje web stranica
Microsoft	*Office 365*	Kreiranje, obrada, dijeljenje i slanje dokumenata, prezentacija, tabličnih kalkulacija i slično te zajednički pristup i rad više korisnika u stvarnom vremenu
Dropbox	*Dropbox*	Prenošenje, obrada i dijeljenje datoteka različitih vrsta sa ili bez zaštita s lozinkom
SAP	*Bussiness ByDesign*	Planiranje i organizacija resursa organizacije
IBM	*Lotus*	Suradnja među zaposlenicima unutar organizacije radi lakšeg povezivanja

Tablica 1. Primjeri SaaS aplikacija [4, str. 164]

2.2.2. Platforma kao servis – PaaS

Platforma kao servis razlikuje se od prethodnog modela što omogućuje korisniku postavljanje vlastitih razvijenih aplikacija ili kreiranje aplikacija i upravljanje istima, a pruža mu mogućnost implementacije istih na infrastrukturu davatelja usluge. Tu infrastrukturu, koja se nalazi u pozadini programa, korisnik pritom ne može kontrolirati (tu spadaju operacijski sustav, mreža, itd.), dok ima potpunu kontrolu nad implementiranim programima ili aplikacijama koje je sam razvio i po mogućnosti davatelja usluga – konfiguracijama sustava [4, str. 165].

Pružatelj usluge zadužen je za održavanje svega što čini i sadrži oblak, a to podrazumijeva infrastrukturu oblaka, operacijske sustave na njemu, ali i softver koji se pogodi na toj infrastrukturi. S druge strane, korisnik usluga platforme kao servisa odgovoran je samo za svoj program ili aplikaciju koju je razvio za oblak, odnosno njegovo razvijanje, postavljanje te održavanje na platformi. Korisniku je omogućen pristup resursima, a naplaćuju mu se samo oni resursi koje njegov program (ili aplikacija) koristi. Zbog toga, na korisniku je da na platformi stvori i popuni bazu podataka (ako mu je potrebna, inače ju ne treba koristiti), stvori i učita svoj program ili aplikaciju te omogući korištenje krajnjim korisnicima. Pružatelj usluge bavi se opterećenjima sustava, nadzire rad programa ili aplikacije korisnika te raspodjeljuje resurse kako bi se što efikasnije koristili te ovisno o potrebnoj skalabilnosti, povećava ili smanjuje broj potrebnih mrežnih i računalnih kapaciteta te takvim upravljanjem postiže da program (ili aplikacija) u svakom trenutku bude dostupan sa bilo kojeg mjesta na svijetu [3, str. 4].

S poslovnog aspekta gledajući, korisnik može brže razviti svoju aplikaciju ili organizacija može u kraćem roku razviti svoj program jer dio posla vezan uz platformu odrađuje pružatelj usluge. Pritom su troškovi korištenja ovog modela znatno niži te se smanjuje potreba za dodatnim tehničkim osobljem. To su prednosti kod korištenja ovog modela, no postoje i značajni nedostaci. Jedan od njih se nalazi u smanjuju ovlasti i kontroli korisnika [3, str. 4].

PaaS model usluga donosi sa sobom određena ograničenja:

- platforma je unaprijed određena (stoga korisnici ne mogu kontrolirati i modificirati okružje za rad),
- smanjena je mogućnost migracije (mogućnost prenošenja postojeće aplikacije u oblak je smanjena ili nije moguća ako je aplikacija razvijena u nepodržanom okruženju),
- nije jednostavno promijeniti pružatelja *PaaS* usluge jer migracija nije moguća ili je preskupa za određenog korisnika ili organizaciju [3, str. 5].

Neki od primjera *PaaS* usluga dati su u tablici ispod (tablica 2), a najpoznatiji pružatelj *PaaS* usluga je Google sa svojim platformom *Google App Engine* (skraćeno GAE).

Organizacija	**Platforma**	**Obilježja**
Google	*Google App Engine (GAE)*	Platforma za razvoj, postavljanje i održavanje aplikacija ili programa, nameće platformu za razvoj korisniku i ima besplatnu inačicu
Microsoft	*Azure*	
Amazon Web Services (AWS)	*Elastic BeanStalk*	Platforma za razvoj, postavljanje i održavanje aplikacija ili programa, ne nameće platformu za razvoj korisniku i ima besplatnu inačicu
Salesforce	*Salesforce Platform*	

Tablica 2. Primjeri PaaS platformi [4, str. 165-166]

2.2.3. Infrastruktura kao servis – IaaS

Infrastruktura kao servis nameće se kao potpuno rješenje za korisnik, odnosno kao najsveobuhvatniji model pružanja usluge računarstva u oblaku jer korisniku nudi cjelokupnu računalnu infrastrukturu (virtualne uređaje i druge resurse) na kojoj onda može gotovo neograničeno razvijati svoje aplikacije ili programe [4].

Ujedno je ovaj model i najosnovniji za korisnika ili organizaciju i označava skup računalnih, memorijskih i mrežnih resursa koji se mogu koristiti u oblaku. Pružatelji usluge upravljaju čitavom infrastrukturom i korisnik joj nema pristup, ali ima nadzor nad operacijskim sustavom i razvojem aplikacija (ili programa). Pritom korisnik može unajmiti čitava fizička ili virtualna računala te mu se pruža mogućnost pohrane podataka. Kod ovog modela korisnik sam upravlja uravnoteženjem opterećenja pa prema tome odabire i broj fizičkih ili virtualnih računala koje će koristiti, a nakon toga puni bazu podataka sa podacima za rad i na ta ista računala instalira svoju aplikaciju ili program. U ovome najosnovnijem modelu, pružatelj usluge nije odgovoran za nadogradnje sustava i kompletnu administraciju, već se za to brine korisnik, odnosno organizacija koja preko ovog modela razvija ili prenosi svoju aplikaciju ili program [3, str. 10].

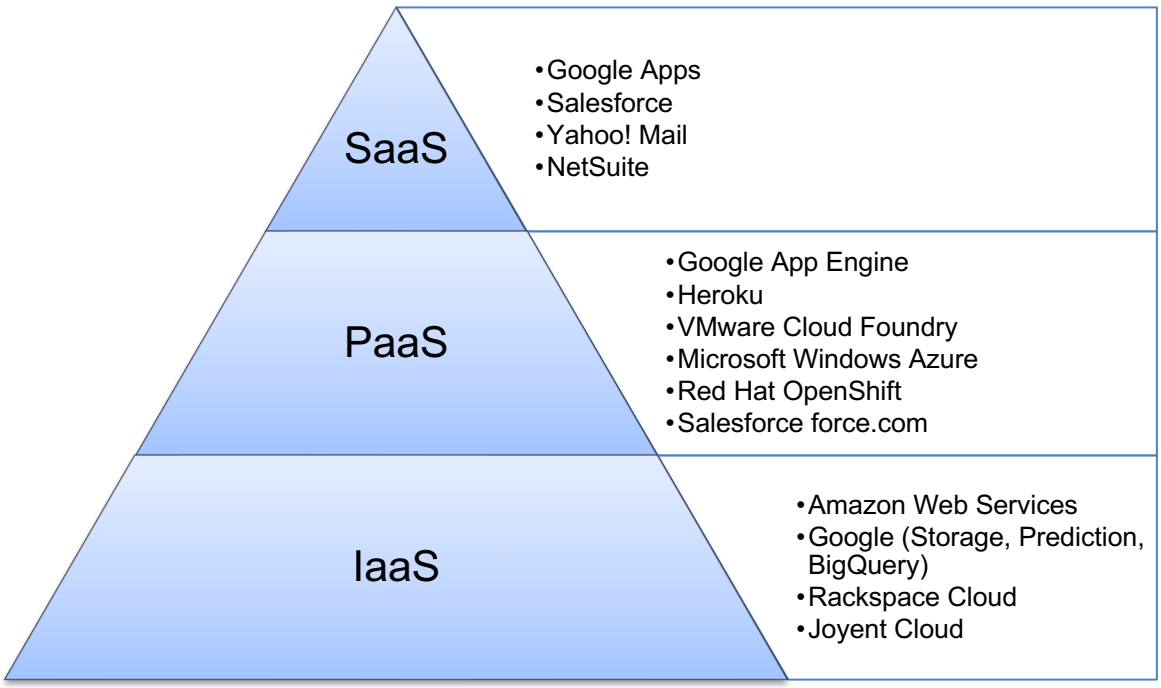
Neki od primjera *IaaS* usluga dati su u tablici ispod (tablica 3).

Organizacija	**Infrastruktura**	**Opis**
Microsoft	*SQL Server*	Baza podataka koja se pokreće unutar virtualnih računala u oblaku
AT&T	*Synaptics Compute as a Service*	Dio platforme za računarstvo u oblaku koja korisnicima omogućuje iznajmljivanje virtualnih

Amazon Web Services (AWS)	*Elastic Compute Cloud*	računala na kojima mogu pokretati vlastite aplikacije ili programe, plaća se po korištenju ili po pretplati

Tablica 3. Primjeri IaaS sustava [4, str. 166]

U ilustraciji ispod vidljivo je u kojem su odnosu (piramidalnom) ova tri modela pružanja usluga računarstva u oblaku.



Slika 2. Modeli pružanja usluga računarstva u oblaku i primjeri

2.3. Platforma kao servis i njen začetak

Platforma kao servis (*PaaS*) sa softverom kao servisom (*SaaS*) donosi velike prednosti razvijatelju programskih proizvoda. Definira se kao platforma koja omogućava razvoj web aplikacija, brzo, jednostavno i bez kompleksnosti nabavke i održavanja potrebnog softvera i infrastrukture jer ova usluga razvijateljima omogućava razvoj aplikacija i usluga putem interneta.

Neke od prednosti PaaS usluge su:

- osjetne uštede jer nema investicija u opremu,
- članovi razvojnog tima mogu biti na raznim lokacijama diljem svijeta i zajedno raditi u oblaku,
- usluga pruža fleksibilnost i kontrolu nad instaliranim alatima za kreiranje platforme koja će biti prilagođena svačijim potrebama,
- brigu za sigurnost podataka i stabilno funkcioniranje prepušta se pružatelju PaaS usluge.

Kod *PaaS* modela, korisniku nije dopušteno upravljanje infrastrukturom oblaka uključujući mrežu, operacijski sustav ili mjesto pohrane podataka jer se za to brine poslužitelj, ali mu je dopušteno upravljanje razvijenim programima (koji su postavljeni na infrastrukturi oblaka) i omogućeno mu je mijenjanje konfiguracije okruženja poslužitelja.

Od ta tri modela (*SaaS*, *PaaS* i *IaaS*), najpoznatiji su *IaaS* i *SaaS*, no situacija se polako krenula mijenjati s obzirom da je lako moguće da je *PaaS* upravo najmoćniji od ostala dva modela. *IaaS* se dobiva besplatno uz *PaaS*, ali *PaaS* obuhvaća širok skup usluga koje su izuzetno skupe i često previše zahtjevne da biste ih sami održavali. Tu se misli na operacijski sustav, baze podataka, licenciranje softvera, umrežavanje i raspoređivanje opterećenja, web i druge poslužitelje, softverske zakrpe i nadogradnje, nadzor, upozoravanja, sigurnosne popravke, administracije sustava itd. Ključna prednost korištenja ovog modela nad samostalnim održavanjem opreme je u tome da neće biti neiskorištenih kapaciteta (engl. *idle capacity*) zbog kupovine znatno jače računarske moći nego što je potrebno prema prethodnim predviđanjima. Za poslovne je korisnike ovo glavna prednost kod *PaaS* modela jer ih ništa ne frustrira više kao skupa investicija za koju znaju da se ne koristi kako i koliko treba [5, str. 608].

2.3.1. Od Amazona, preko Salesforcea do Googlea

Moglo bi se reći da je sve započelo sa Johnom Gageom, 21. zaposlenikom u organizaciji Sun Microsystems, koji je 1984. godine izgovorio jednu frazu koja je glasila „Mreža je računalo“. Ipak, komercijalizacija pristupa računarstvu u oblaku krenula je tek nakon što je Amazon 2006. godine uveo AWS (*Amazon Web Services*), a pokretač za uvođenje bilo je upravo pitanje neiskorištenih kapaciteta pri razvoju. Odgovor je bio jednostavan – iznajmiti cjelokupnu procesorsku moć i kapacitete za skladištenje kao servis.

Organizacija Amazon je bila prva koja je na „velika vrata“ donijela taj novi trend, a nakon njega pridružilo se još nekoliko velikih tehnoloških organizacija kao što su Google, Salesforce, Microsoft, RackSpace, Joyent, VMware i drugi. No, dok su se Amazonovi servisi EC2 i S3 nesumnjivo nalazili na infrastrukturnoj razini (odnosno pripadali *IaaS* modelu), počelo se stvarati tržište za one „igrače“ koji su željeli samo određene usluge za aplikacije ili programe koje razvijaju, odnosno organizacije ili pojedinačne korisnike koji su željeli koristiti usluge *PaaS* modela. To je navelo Salesforce da napravi *force.com* (sadašnji *salesforce.com*), prvi *PaaS* model namijenjen upravo za to tržište. S obzirom na nedostatke koje je imao i činjenicu da nisu svi željeli Salesforce aplikaciju napisanu na još jednom programskom jeziku, Google je razvio općenitiji *PaaS* servis kojeg je pokrenuo 2008. godine, a nazvao ga je App Engine [5, str. 608].

2.4. Google App Engine (GAE)

Google App Engine (skraćeno GAE), jedan je od primjera *PaaS (*engl. *Platform as a Service)* modela pružanja usluga računarstva u oblaku koji omogućava korisnicima postavljanje i pokretanje svojih web aplikacija na Googleovoj infrastrukturi.

Aplikacije se mogu razvijati u programskim jezicima Java, Python ili Go, nakon čega se izvršavaju u izvršnim okruženjima (engl. *runtime environment*) jezika. Korištenje App Engine oblaka naplaćuje se po potrošnji resursa, a ne po vremenu posluživanja. Također, omogućuje se i automatizirano skaliranje broja poslužitelja ako se poveća količina zahtjeva za aplikacijom.

2.4.1. Koraci za korisnika

Korisnik napiše vlastiti aplikaciju, napravi račun na App Engine servisu i pošalje ju na Googleove poslužitelje. Također, korisnik može aplikaciju napraviti i nakon što napravi račun na servisu, u samom radnom okruženju servisa. Usluga korištenja servisa je prvotno besplatna, sve dok se ne iskoristi određena količina resursa koja je na raspolaganju, nakon čega se plaća ovisno o potrebama korisnika ili organizacije. Domenu aplikacije korisnik može sam odrediti ili se prijaviti za Googleovu domenu, koju tada Google dodjeljuje sukladno svojim pravilima. Nakon toga, Google preuzima svu brigu o radu aplikacije. Njihova arhitektura dodjeljuje aplikacijama resurse, brine se za prosljeđivanje zahtjeva prema njima i daje im određeno vrijeme za sastavljanje odgovora. Osim toga, korisnicima su na raspolaganju i dodatne usluge poput memorijskog prostora koje korisničke aplikacije mogu koristiti pri radu [1, str. 10].

2.4.2. Prednosti i nedostaci GAE servisa

Google App Engine osigurava korisničkoj aplikaciji pouzdano izvršavanje pod velikim opterećenjem i s velikim skupovima podataka. Tako servis omogućuje:

- dinamičko posluživanje,
- trajnu pohranu podataka i upite nad podacima (*Google Query Language*) s podržanim transakcijama,
- automatsko skaliranje i raspoređivanje opterećenja,
- API-jeve (engl. *Application Programming Interface*) za autentificiranje korisnika i slanje e-poruka pomoću Googleovih korisničkih računa,
- lokalno razvojno okružje koje simulira GAE,
- redove zadataka (engl. *task queues*) pomoću kojih se aplikacijski kod može izvršavati i izvan HTTP zahtjeva,

- zakazane zadatke (engl. *scheduled tasks*) za okidanje događaja u određenim trenucima ili po definiranim periodičnim intervalima [1, str. 10].

Aplikacije koje se izvršavaju na poslužiteljima ne utječu na aplikacije drugih korisnika, i to iz razloga jer sigurnosno okruženje (engl. *sandbox*) ograničava pristup operacijskom sustavu na kojem se aplikacija izvršava. Ta ograničenja omogućuju App Engineu da distribuira zahtjeve za pristup aplikaciji na više različitih poslužitelja te da po potrebi dodaje i uklanja poslužitelje ovisno o količini zahtjeva za aplikacijom.

Čorak [1, str. 11] navodi neka od ograničenja koje okružje nameće:

- aplikaciji se pristupa isključivo preko HTTP (ili HTTP) zahtjeva, a aplikacija može pristupiti sadržajima na internetu samo pomoću URL ili adrese elektroničke pošte,
- aplikaciji nije omogućeno pisanje po datotekama operacijskog sustava, ali joj je omogućeno čitanje datoteka i to samo onih koje su sastavni dio te aplikacije,
- aplikacija može trajno pohraniti podatke ili ih postaviti u privremenu memoriju (eng. *memcache*) ili čak koristiti neki drugi servis za pohranu podataka,
- aplikacijski kod izvršava se samo kao odgovor na korisničke zahtjeve, redove zadataka i zakazane zadatke,
- kod obrađivanja zahtjeva, dretva ne može pokrenuti novi potproces nakon što se pošalje odgovor na zahtjev.

2.4.3. Komponente GAE servisa

App Engine ima četiri glavne komponente koje čine cijeli sustav: izvršna okruženja programskih jezika, skalabilnu sklopovsku infrastrukturu, administraciju baziranu na webu i skup alata za razvoj softvera (engl. *Software Development Kit*, SDK).

2.4.3.1. Izvršna okruženja programskih jezika

Izvršna okruženja dostupna u App Enginu su za programske jezike Python, Java i Go. S obzirom da se u ovome radu koristi programski jezik Python, daljnje pisanje odnosit će se samo na taj programski jezik.

Dodatno, bitno je naglasiti da zahvaljujući podršci za Javu, korisnici mogu pisati kod i na jezicima sa odgovarajućim interpreterom koji se može izvršavati u Javinom virtualnom stroju (engl. *Java Virtual Machine*, JVM), kao što su Ruby, PHP, JavaScript, a izvršavaju ga JRuby, Quercus, Rhino i Jython (dodatno su tu i Scala i Groovy). Posebno je zanimljiva kombinacija Pythona sa Jythonom jer korisnici koji već imaju Java pakete u ovoj kombinaciji mogu razvijati potpuno nove projekte na Pythonu, odnosno ako ne žele razvijati nove biblioteke za programski jezik Python, na ovaj način mogu iskoristiti postojeće pakete razvijene u Jythonu [5, str. 610].

2.4.3.2. Sklopovska infrastruktura

Sklopovska infrastruktura je kod ovog modela nešto nalik crnoj kutiji jer korisnici ne znaju puno o bilo kojem dijelu opreme na kojem se izvršava kod. Vjerojatno je da se koristi u te svrhe Linux i to u velikoj mjeri te da se poslužitelji na kojima se nalazi aplikacija ili program korisnika ili organizacije nalazi u centrima podataka priključenim na globalnu mrežu.

Bigtable, sustav nerelacijskih baza podataka koji App Engine koristi za pohranu podataka, jedino je što korisnici kod *PaaS* modela i znaju da GAE sustav koristi. Računarstvo u oblaku korisnike oslobađa potrebe da brinu o infrastrukturi (barem što se tiče *PaaS* modela na kojem se temelji GAE sustav) i svi detalji o infrastrukturi i održavanju iste odvija se u pozadini [5, str. 610].

2.4.3.3. Administracija bazirana na webu

App Engine kroz jednostavno sučelje za administratora aplikacije korisniku pruža uvid u njegovu aplikaciju, promet, podatke, korisnička prijavljivanja, korištenje i slično.

Iako GAE pruža mnoge usluge, API sučelja za razvojne programere i omogućava visoku kontrolu aplikacije koja se postavi na sam sustav, koncepcija samog App Enginea bazirana je na tzv. modelu pješčane kutije (engl. *sandbox*) što znači da aplikacija jednog korisnika dijeli resurse s drugim korisničkim aplikacijama i da se sve aplikacije nalaze u ograničenom okruženju međuovisne jedne o drugima. To znači da se pritom gubi određena razina kontrole svoje aplikacije i da se ne može na sve moguće komplikacije ili novonastale situacije utjecati kao da se razvija i postavlja aplikacija na neku svoju platformu sa apsolutnim ovlastima i mogućnostima, ali u zamjenu za to dobivaju se komponente i mogućnosti koje je inače teško napraviti i održavati te skalabilnost koja GAE čini posebno primamljivom uslugom [5, str. 610-611].

2.4.3.4. Skup alata za razvoj softvera (Software Development Kit, SDK)

Radi ograničenja koja su ili još uvijek u vrijeme pisanja ovog teksta postoje, a među kojima su bili ograničenje za grananjem novih procesa, nedozvoljeni pozivi operacijskog sustava, nemogućnost izrade datoteke na lokalnom disku i slična ograničenja, postoji skup alata za razvoj softvera, tzv. App Engine SDK, koji sadrži API sučelja više razine pomoću kojih se mogu nadoknaditi mogući gubici funkcionalnosti koji su nastali zbog prethodnih ograničenja.

Također, pošto je verzija programskog jezika Python koju GAE izvršava podskup pune distribucije, sa tom verzijom korisniku nije omogućen pristup svim Pythonovim alatima, iako sve verzije nakon verzije 2.7 podržavaju znatno više biblioteka, među kojima su i one poznatije poput NumPy, lxml i PIL. Međutim, moguće je koristiti Pythonove pakete i drugih proizvođača

i dodati ih izvornom kodu, dokle god se radi o „čistom“ Pythonovom kodu (nema izvršnih datoteka poput *.so* ili *.dll* datoteka).

Ukupan broj datoteke koje se mogu postaviti na GAE je ograničen, s time da je i ograničena ukupna veličina tih datoteka (tu se primarno govori o datotekama aplikacije ili drugim statičkim datotekama kao što su HTML, CSS, JavaScript datoteke i slično).

Ako korisnička aplikacija koristi multimedijalne datoteke veličina većih od dozvoljenih po datoteci, korisnik ih može sačuvati pomoću servisa App Engine Blobstore, koji omogućuje čuvanje datoteka proizvoljnih veličina, odnosno ne postoji određeno ograničenje veličine pojedinih datoteka. Ukoliko postoji veći broj *.py* datoteka, moguće ih je zapakirati u *.zip* datoteku, otpremiti tu datoteku u sustav a bez obzira koliko je *.py* datoteka arhivirano, korisnik plaća samo za tu jednu *.zip* datoteku u kojoj su sve pojedinačne *.py* datoteke arhivirane, dokle god *.zip* datoteka nije veće veličine od one dopuštene.

No, kako bi se otklonila nastala ograničenja, Google nudi više svojih rješenja za iste. Tako za komunikaciju s drugim poslužiteljima može se koristiti API URLfetch, za slanje i primanje elektroničke pošte tu je API Email. Također se može koristiti API XMPP (*eXtensible Messaging and Presence Protocol*) za slanje i primanje trenutnih (IM) poruka. Tu je i rješenje za raspodjelu memorijskog prostora (Memcache) te jedno od najbitnijih – rješenje za pristupanje bazi podataka (Datastore) i ostali [5, str. 612-614].

U tablici ispod prikazani su neki od servisa i sučelja dostupni korisnicima Google App Enginea u vrijeme pisanja ovog rada.

Servis/API	**Opis**
App Identity	Potrebno kada aplikacija sadrži kod koji mora identificirati sebe ili druge API-je koji traže određenu informaciju.
Appstats	Okruženje upravljano događajima (engl. *events*) za mjerenje performansi aplikacije.
Blobstore	Potrebno koristiti kad se prenose datoteke većih veličina od dopuštenog ograničenja.
Capabilities	Omogućuje aplikacijama da detektiraju kad su App Engine Datastore ili App Engine Memcache nedostupni kako bi se korisnicima prenijela obavijest o potencijalnom zastoju u radu aplikacije.
Datastore	Distribuirano, skalabilno, ne-relacijsko trajno spremište podataka.
Images	Potrebno za manipulaciju slikovnih podataka (primjerice obrezivanje, mijenjanje veličine slike, rotiranje i slično).

Mail	API koji služi aplikacijama kao posrednik pri slanju i primanju elektroničke pošte.
Memcache	Distribuirana predmemorija podataka između aplikacije i trajnog spremišta.
NDB (new database)	Novije spremište podataka više razine prikladno za aplikacije razvijene u Python programskom jeziku.
OAuth	Omogućuje siguran način pristupa trećih strana podacima sa strane korisnika bez prethodne autorizacije.
Task Queue	Omogućuje korisnicima pokretanje pozadinskih zadataka neovisno o samoj interakciji s korisnikom.
URLfetch	Potrebno za komunikaciju s ostalim aplikacijama na mreži preko HTTP/S zahtjeva i odgovora.
Users	Autentifikacijski servis samog App Enginea koji upravlja s prijavljivanjem korisnika.

Tablica 4. Servisi i API dostupni u App Engineu [5, str. 614-616]

2.4.4. Razvojna okruženja za App Engine

Za razvoj aplikacija koristeći App Engine postoji široki izbor razvojnih okruženja. Početnici će navjerojatnije koristiti webapp ili webapp2 kako bi se najlakše naučili i započeli prve korake u razvoju svoje aplikacije. Bitno je naglasiti da je to razvojno okruženje ono koje se dobiva kao zadano pri korištenju App Enginea. S ovim razvojnim okruženjem mogu se razviti jednostavne i solidne aplikacije u vrlo kratkom roku i s dovoljno osnovnih alata.

Drugo razvojno okruženje koje se najčešće koristi za razvoj u GAE-u je Django. Google App Engine pruža određene verzije Djanga već na svojim poslužiteljima pa nije potrebno da korisnici samostalno instaliraju Django skupa sa svojim aplikacijama i to predstavlja svojevrsnu prednost onim korisnicima koji se odluče za Django razvojno okruženje. U vrijeme pisanja ovog rada, razvojno je okruženje Django za Python programski jezik najpopularnije među razvojnim programerima te se doima da će tako i ostati neko značajnije duže vremensko razdoblje.

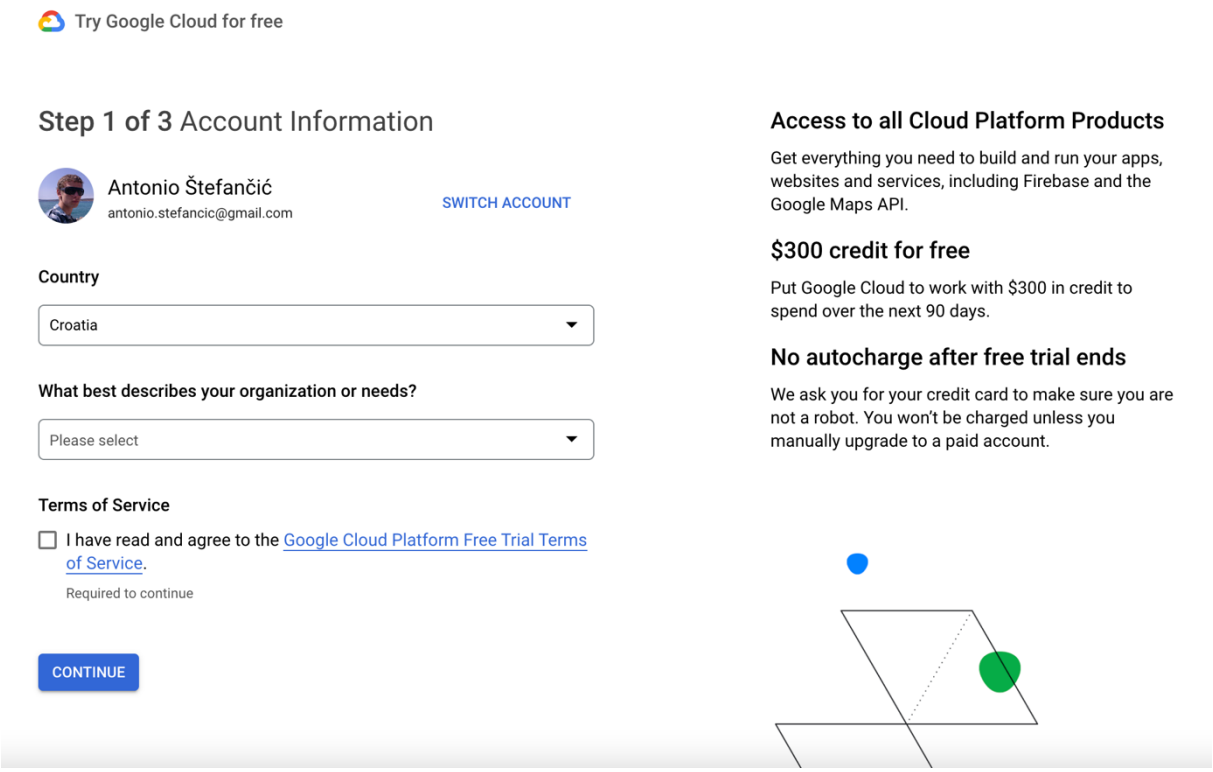
Još jedno razvojno okruženje koje uzima svog zamaha je Flask. Flask je mikro razvojno okruženje koje je specifično zbog izrazite jednostavnosti korištenja, jednostavnosti podešavanja i s kojim se direktno može povezivati na GAE-ov Datastore, odnosno spremište podataka [5, str. 617-619; 6].

Razvojna okruženja trenutno dostupna za razvoj za App Engine su sljedeća:

- webapp, webapp2
- bottle
- Django
- Flask
- GAE framework
- Google App Engine Oil (GAEO)
- MVCEngine
- Pyramid
- tipfy
- web2py

2.5. Google App Engine u praksi

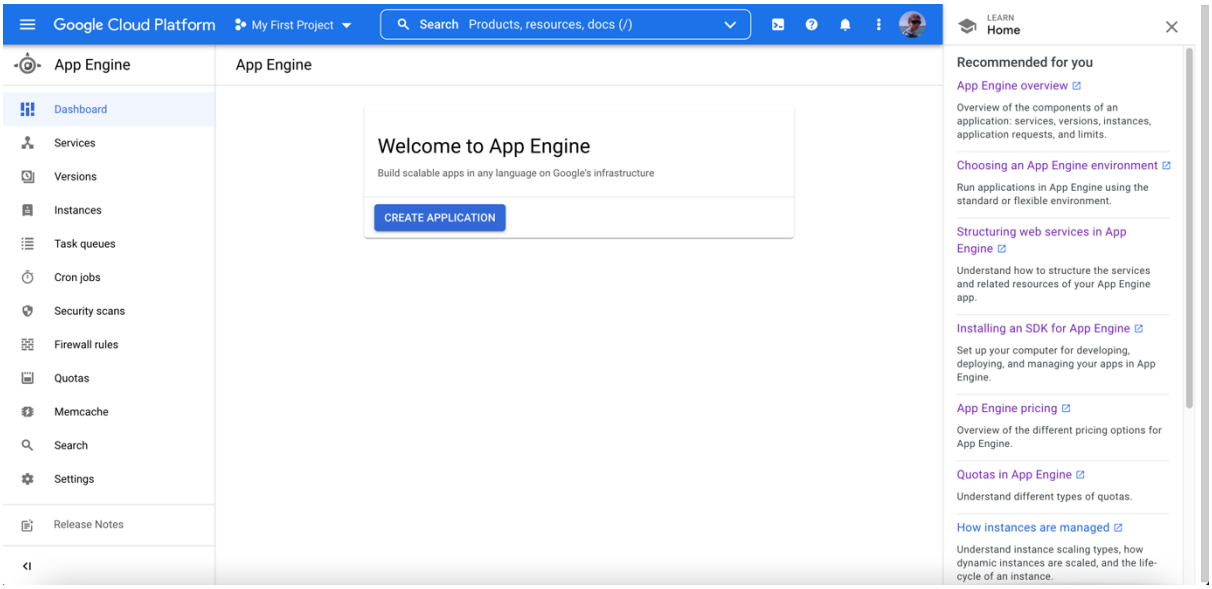
U ovome dijelu rada bit će objašnjeno kako se pristupa Google Cloud uslugama, pronalazi nadzorna ploča Google App Enginea, što se sve na njoj može pronaći i na koji se način kreće u postupak kreiranja aplikacije u sustavu, a bit će date i osnovne informacije prije samog korištenja GAE-a o komponentama svake aplikacije te okruženjima koje postoje.



Slika 3. Izrada računa za Google Cloud usluge

Za početak nužno je izraditi račun za Google Cloud usluge kako bi se moglo koristiti App Engine. Korisnici koji već prethodno imaju Google račun moraju samo ispuniti poneke osnovne podatke i spremni su za sljedeći korak. Korisnici koji nemaju ili ne koriste Google račun, ovdje će ga morati izraditi kako bi koristili ove Googleove usluge. Za izradu besplatnog računa potrebno je unijeti i kartične podatke i autorizirati samu transakciju preko banke, iako sam Google neće ništa naplatiti kroz besplatno probno razdoblje tijekom 90 dana korištenja njihovih usluga. Nakon toga moguća je naplata korištenja usluga ako se uključila opcija mjesečnog automatskog plaćanja. U probnom razdoblju dobiva se i fiktivnih 300 dolara na korisničkom računu kako bi se mogle isprobati određene usluge koje se inače naplaćuju, no tih fiktivnih 300 dolara korisnik mora samostalno aktivirati u samom sučelju Google Cloud usluga [7].

Sljedeći korak je usmjeriti se na nadzornu ploču App Enginea koja izgleda kao na slici ispod, a preko koje se kreiraju nove aplikacije od strane korisnika.



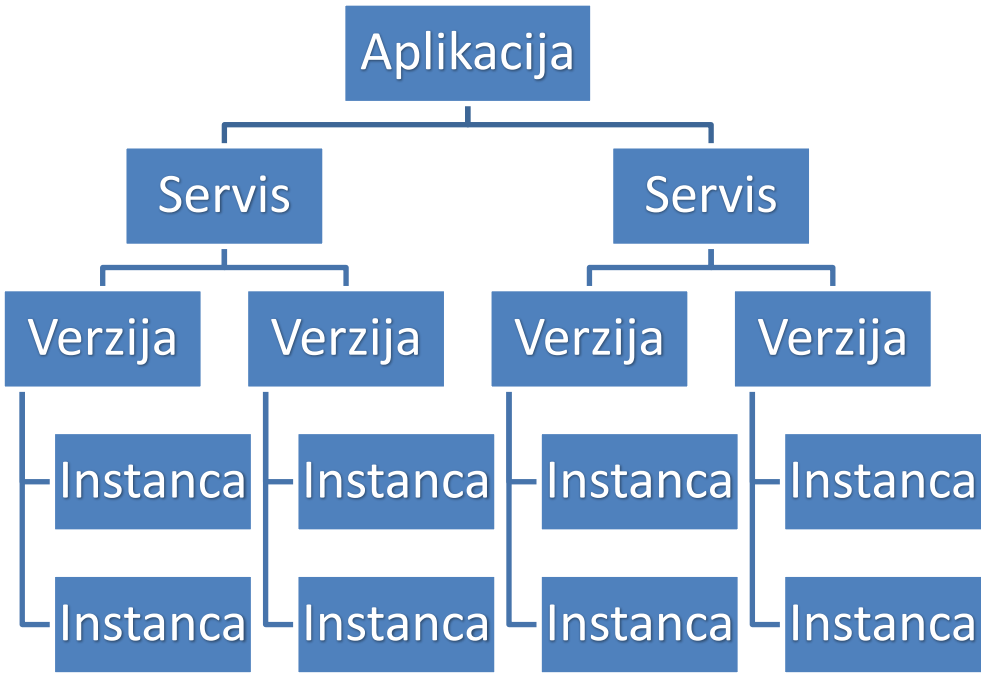
Slika 4. Nadzorna ploča App Enginea

Na nadzornoj ploči sa lijeve strane nalaze se izbornici u kojima se mogu pratiti sve aktivnosti vezane uz aplikaciju koja se razvija, u gornjem dijelu se odabire aplikacija na kojoj se trenutno radi (u ovome primjeru to je aplikacija „My First Project“), u središnjem dijelu nalazi se pozdravna poruka s gumbom za kreiranje nove aplikacije, a s desne strane nadzorne ploče nalazi se poveći broj linkova na određene lekcije i uputstva kako bi se novog korisnika upoznalo s time što je App Engine, čemu služi, što sve može na njemu postići te koja su potencijalna ograničenja, ali i dobre prakse pri započinjanju i daljnjem radu u GAE-u.

2.5.1. Komponente aplikacije na App Engineu

Aplikacija u App Engineu sastoji se od više komponenti. Prva komponenta je aplikacijski resurs koji se kreira prilikom kreiranja same aplikacije. Taj resurs sastoji se od jednog ili više servisa, a svaki servis može biti konfiguriran za korištenje pod različitim vremenima izvođenja s različitim uvjetima performansi. Unutar svakog servisa može se razviti više verzija tog servisa, a svaka verzija se onda pokreće pod jednom ili više instanci, ovisno o tome koliko podatkovnog prometa svaka od instanci može koristiti. Znači, svaka App Engine aplikacija sadrži servise, verzije tih servisa i instanci verzija aplikacije. Prilikom kreiranja aplikacije, resursi se kreiraju u određenom dijelu svijeta, odnosno regiji koju sam korisnik odabere te ovisno o izabranoj regiji razlikuju se i određena ograničenja.

Svaka App Engine aplikacija sadrži barem jedan servis, tzv. zadani servis (engl. *default*), koji onda može sadržavati više verzija tog servisa. U grafičkom prikazu ispod može se vidjeti aplikacija sa dva servisa, od kojih svaki ima dvije verzije, a verzije po dvije instance.



Slika 5. Komponente App Engine aplikacije [7]

Ovakva podjela postoji iz više razloga. Podjela na servise u App Engineu služi tome da se velike aplikacije mogu podijeliti na logične cjeline koje onda na jednostavan i siguran način mogu dijeliti funkcionalnosti App Enginea i komunicirati jedne s drugima, odnosno ponašaju se kao mikroservisi (engl. *microservices*). Naravno, kompletna aplikacija u App Engine može se sastoji od samo jednog servisa, ali je poželjnije da se dizajnira i razvije kroz više servisa koji bi bili poput seta mikroservisa. Na primjer, aplikacija koja upravlja zahtjevima korisnika može sadržavati više servisa zbog toga što postoji više zadataka koje aplikacija mora moći izvesti: jedan bi se servis bavio API zahtjevima s mobilnih uređaja, drugi internim zahtjevima korisnika, a treći analizom zahtjeva na strani poslužitelja. Svaki servis u App Engineu sastoji se od izvornog koda napisanog u programskom jeziku te pripadajućih konfiguracijskih datoteka, a svaki set datoteka koji se prenese određenom servisu predstavlja određenu verziju tog servisa, poput neke vrste nadogradnje samog servisa s niže na više verzije.

Iznimno važna značajka App Enginea vezana je upravo uz verzije servisa. Naime, ako određeni servis ima više verzija, tada je vrlo jednostavno prebacivanje između različitih verzija kako bi se omogućilo testiranje, vraćanje određenih funkcionalnosti na staro ili neke druge radnje, kao što je i moguće preusmjeravanje (ili podjela) podatkovnog prometa prema jednoj ili više određenih verzija aplikacije.

Verzije servisa izvode se na jednoj ili više instanci tih servisa. App Engine po zadanim postavkama samostalno određuje broj instanci skalabilno prometu koji nastaje, a sve u svrhu očuvanja pouzdanih i konzistentnih performansi i smanjenja opterećenja i troškova rada.

Postoje određena ograničenja u pogledu toga koliko servisa aplikacija može imati u besplatnoj verziji, a koliko u plaćenoj verziji korištenja GAE-a. Također, postoje i ograničenja broja verzija po servisu te broja instanci po verziji [7].

U tablici ispod nalaze se spomenuta ograničenja (aktualna u 2022. godini).

Ograničenje	**Besplatna verzija aplikacije**	**Plaćena verzija aplikacije**
Dozvoljeni broj servisa po aplikaciji	5	105
Dozvoljeni broj verzija po aplikaciji	15	210
Dozvoljeni broj instanci po verziji	20	25 (200 za regiju SAD)

Tablica 5. Ograničenja servisa, verzija i instanci u App Engineu [7]

2.5.2. Okruženja u App Engineu

Kako bi se aplikacije u App Engineu mogle pokretati, potrebno ih je postaviti u određeno okruženje koje u GAE sustavu može biti:

- standardno okruženje i
- fleksibilno okruženje.

Moguće je koristiti simultano oba okruženja za aplikaciju pa bi pritom aplikacija mogla iskoristiti prednosti oba okruženja. App Engine je pogodan za aplikacije razvijene na mikroservisnoj arhitekturi koje pritom koriste ova oba okruženja.

No postoje situacije kad je pogodnije razvijati aplikacije u jednom okruženju, a ne u drugome. Primjerice aplikacije koje zahtijevaju brzo skaliranje zbog povećavanja podatkovnog prometa, čije instance se izvode u „pješčanoj kutiji“ (engl. *sandbox*), koje su pisane u novijim verzijama određenog programskog jezika i čiji bi trošak trebao biti neznatan ili gotovo nepostojeći, trebale bi se razvijati u standardnom okruženju. S druge strane, aplikacije koje imaju konzistentnu razinu podatkovnog prometa, pisane su u bilo kojoj verziji (pa i starijoj) određenog programskog jezika, no ovisne su o razvojnom okruženju u kojem je pisan kod te podržavaju niz drugih možebitnih funkcionalnosti (pozadinski procesi i dretve, *SSH debugging*, *WebSockets*), trebale bi se razvijati u fleksibilnom okruženju.

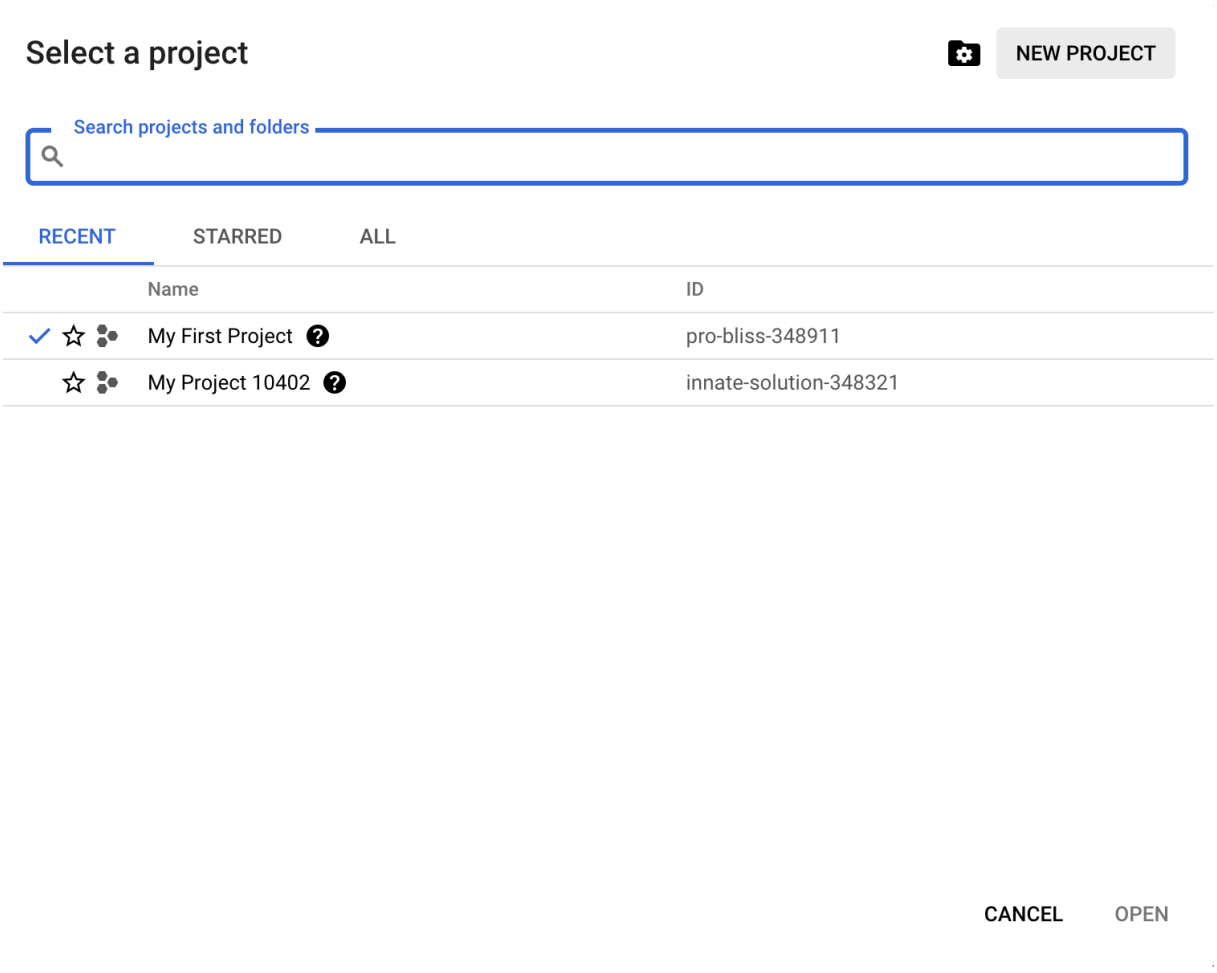
Kod standardnog okruženja plaćanje se vrši s obzirom na broj sati korištenja određene instance, dok se kod fleksibilnog okruženja plaća korištenje virtualnog procesora, memorije i

diskova. Dobro je znati da se aplikacija ili pojedini servisi aplikacije razvijeni u standardnom okruženju mogu prebaciti u fleksibilno okruženje i obratno po potrebi [7].

2.6. Aplikacija preSchedule na App Engineu

U ovome praktičnom dijelu rada bit će prikazano stvaranje projekta, aplikacije unutar samog projekta, instaliranje određenih dodataka za GAE, prenošenje aplikacije preko naredbenog retka na App Engine, moguća podešavanja unutar samog App Enginea te kako je razvijena aplikacija preSchedule i koja joj je svrha.

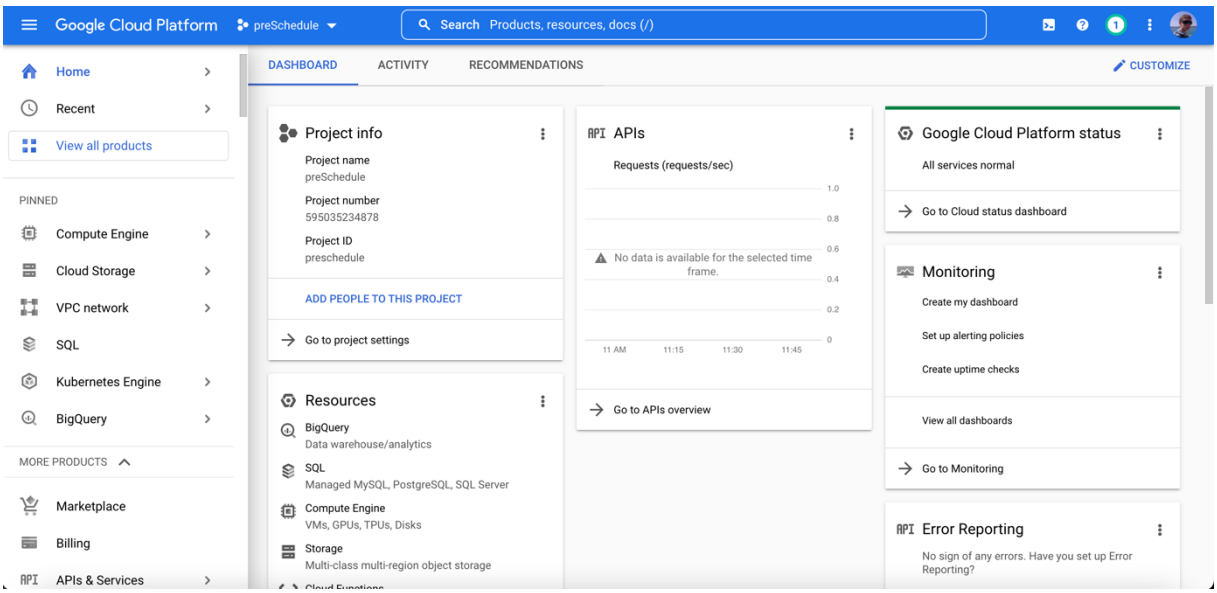
Unutar GAE sustava potrebno je stvoriti novi projekt. Taj projekt mora biti povezan s računom koji ima neku vrstu pretplate, iako u vremenskom razdoblju od 90 dana Google ne naplaćuje ništa za početnu pretplatu, dokle god dana ograničenja u radu nisu prekršena. Na sljedećem prikazu nalazi se dijaloški prozor s popisom prethodno kreiranih projekata i gumbom na gornjoj desnoj strani za kreiranje novog projekta.



Slika 6. Kreiranje novog projekta

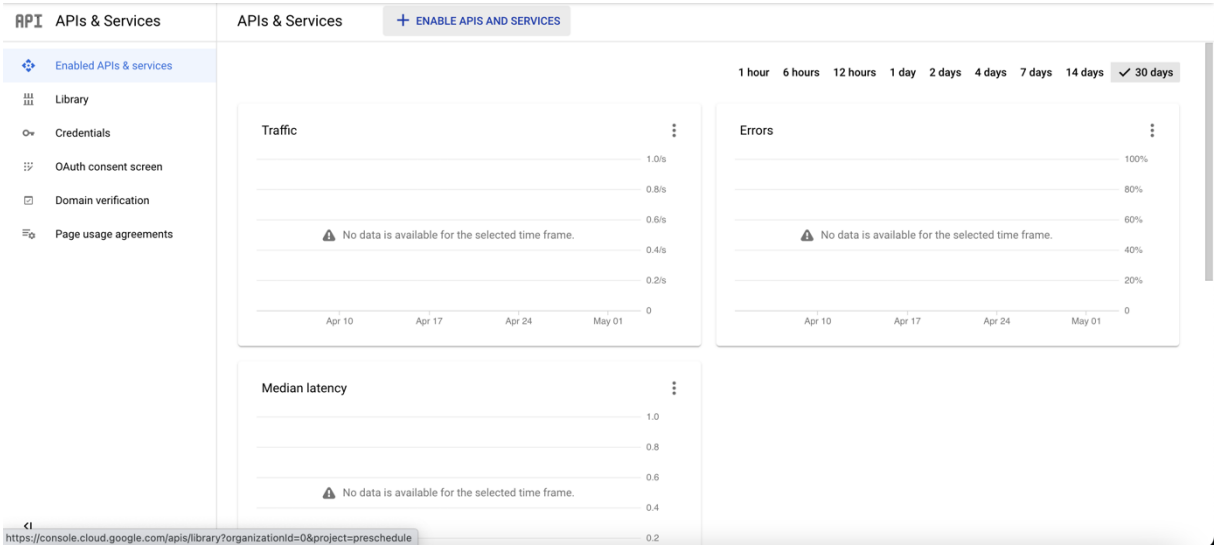
Svaki projekt ima svoje ime i svoj projektni ID. Prilikom kreiranja projekta potrebno je pripaziti na ograničenje broja kreiranih projekata ako se radi o besplatnoj verziji pretplate na App Engine. Nakon toga dolazi se u nadzornu ploču kreiranog projekta s koje se mogu pratiti novonastale promjene, podatkovni promet, zahtjevi za API-jevima te s lijeve strane pronaći

izbornici na sve resurse i opcionalnosti koje nudi GAE. Na slici ispod vidi se nadzorna ploča nakon kreiranja projekta za aplikaciju preSchedule.



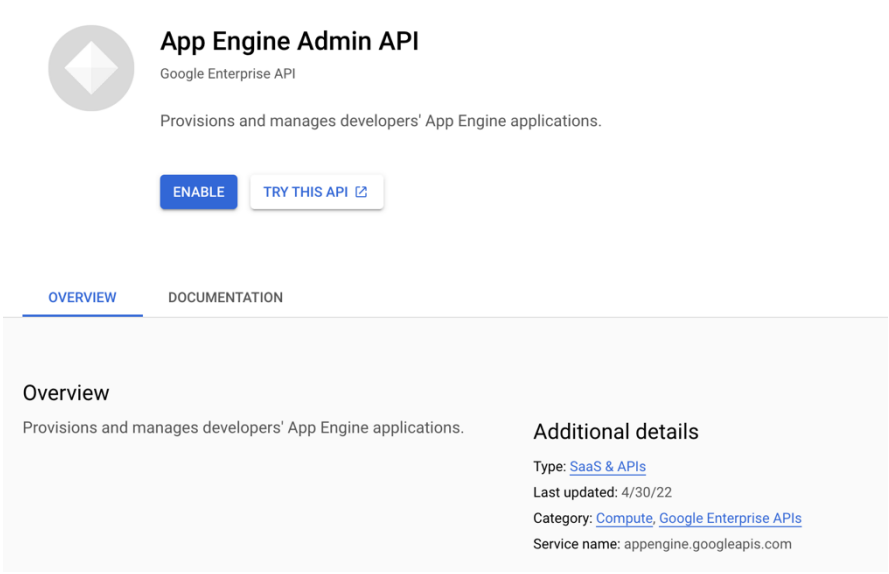
Slika 7. Nadzorna ploča projekta preSchedule

S lijeve strane nalaze se izbornici gdje je potrebno naći kategoriju „API & Services“ gdje se mogu pronaći svi omogućeni API-jevi i servisi. Postoji veliki broj njih koji se mogu koristiti i uključiti i obvezno ih je ovdje prije samog korištenja pronaći i uključiti. Naravno, po potrebi se mogu dodatni servisi i API-jevi uključivati i kasnije. Na slici ispod može se vidjeti kako izgleda prikaz te kategorije za novonastali projekt.



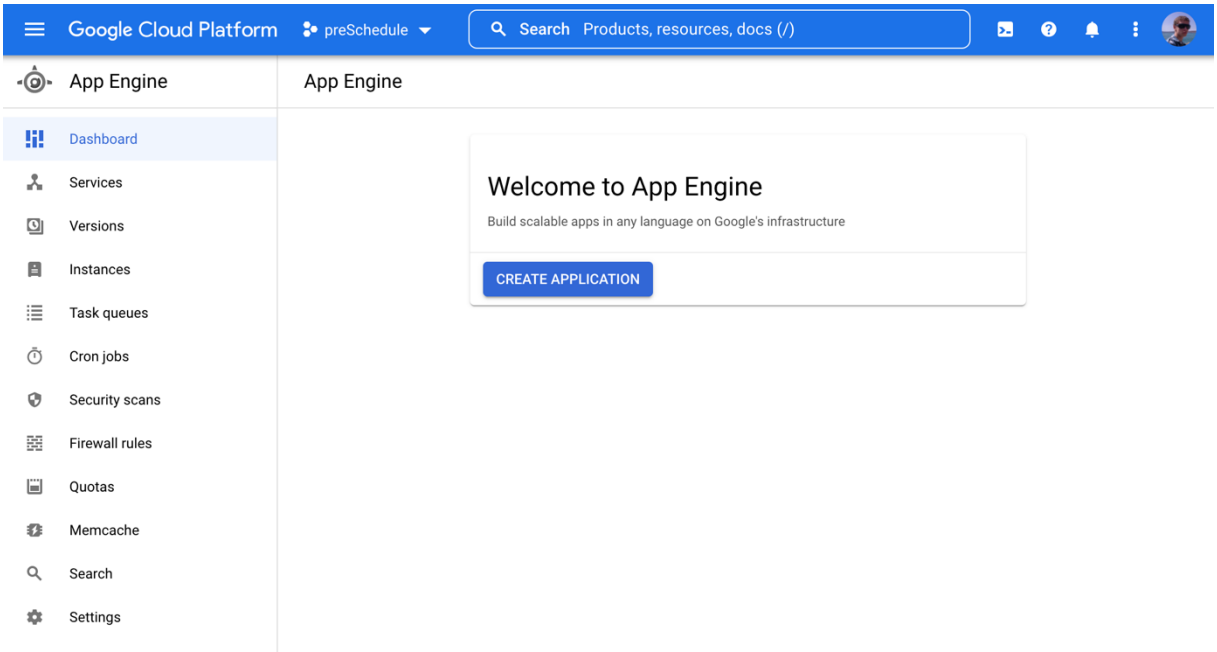
Slika 8. Sučelje za uključivanje i pregled API-ja i servisa

Osnovne informacije o svakom API-ju dane su u dijalogu prije samog uključivanja istog. Tako, primjerice, može se na slici desno provjeriti koja je uloga *Admin API* prije samog uključivanja te se može pronaći i dodatna dokumentacija istog.



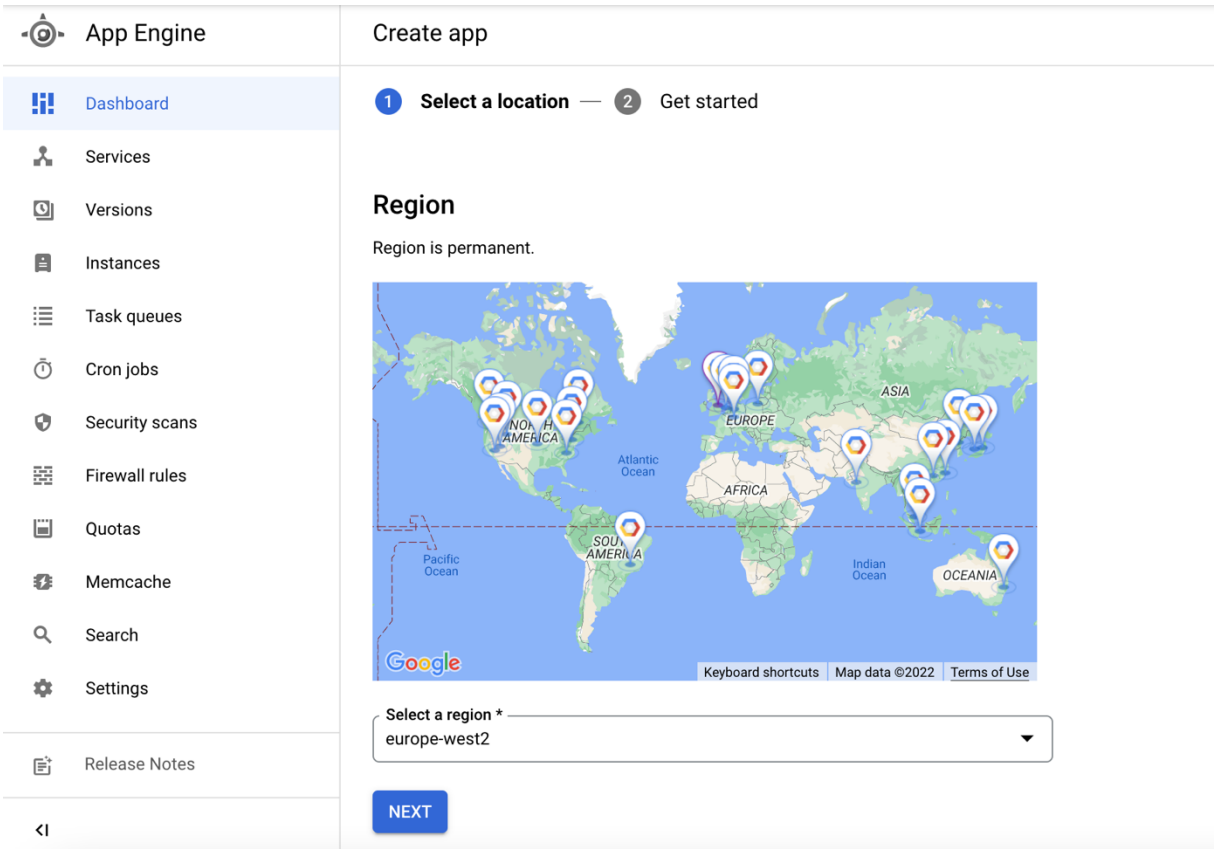
Slika 9. Dijalog za uključivanje Admin API-ja

Nakon toga potrebno je vratiti se na nadzornu ploču App Enginea (App Engine se nalazi također u izborniku s lijeva) te kreirati novu aplikaciju. Bitno je provjeriti da se nalazi u projektu u kojem se želi kreirati aplikacija, a to se može vidjeti u gornjem izborniku, odnosno gornjoj traci. Nova aplikacija se može kreirati preko jednostavnog intuitivnog sučelja, a može se kreirati i preko ugradbene aplikacije naredbenog retka, koja se može uključiti također iz gornjeg izbornika pri desnome rubu. U slici ispod prikazano je kako kreirati novu aplikaciju preko sučelja.



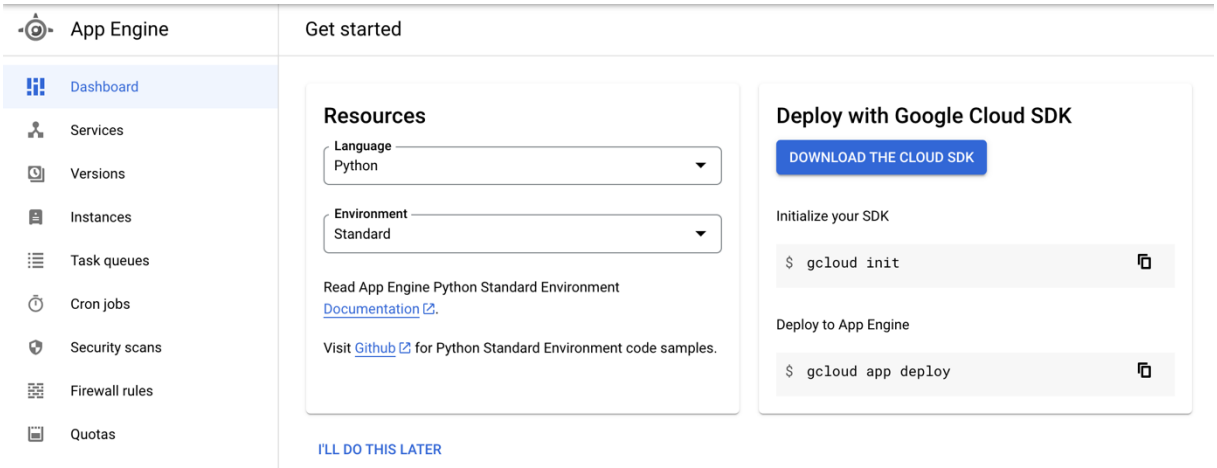
Slika 10. Kreiranje nove aplikacije u projektu preSchedule

Nakon odabira „Create Application“ potrebno je odabrati par stavki kako bi se aplikacija mogla kreirati ispravno. Nužno je izabrati regiju u kojoj će se pokretati nova aplikacija i poželjno je da je ta regija ona najbliža našoj fizičkoj lokaciji. U primjeru na slici ispod odabrana je regija „europe-west2“ (moguće je odabrati i „europe-central“, no zbog jednog postojećeg ograničenja ista nije odabrana).



Slika 11. Odabir regije za kreiranje nove aplikacije

Osim odabira regije, potrebno je odabrati i okruženje u kojem će raditi nova aplikacija, a to može biti standardno ili fleksibilno. Fleksibilno ima svojih prednosti kad se u projektu



Slika 12. Odabir okruženja i preuzimanje Google Cloud SDK

koriste virtualni strojevi, a standardno ima prednosti kod projekata, odnosno aplikacija koje zahtijevaju visoku razinu skalabilnosti u radu. Nakon toga, potrebno je preuzeti i instalirati Google Cloud SDK koji Google u ovome dijelu nudi, kako bi se s alatom Google Cloud CLI projekt sa računala mogao prenijeti na App Engine u oblaku.

Google Cloud CLI je dio skupa alata Google Cloud SDK koji služi za povezivanje sa Google Cloud računom korisnika i prebacivanje svih datoteka aplikacije na App Engine (konfiguracijskih, app.yaml, datoteka izvornog koda itd.).

Prvo je potrebno otvoriti naredbeni redak operacijskog sustava i inicijalizirati Google SDK sa naredbom:

```
./google-cloud-sdk/bin/gcloud init
```

Nakon toga dolazi pozdravna poruka sa kratkim objašnjenjima što se konfigurira i upitom želi li se nastaviti, a nakon toga otvara se preglednik u kojem se potrebno prijaviti sa svojim Google Cloud korisničkim podacima u svoj račun. Nakon uspješne prijave, u naredbenom retku bira se projekt na koji se treba pozicionirati (u ovome primjeru to je projekt „preschedule“):

```
Pick cloud project to use:
 [1] innate-solution-348321
 [2] preschedule
 [3] pro-bliss-348911
 [4] Enter a project ID
 [5] Create a new project
Please enter numeric choice or text value (must exactly match list item):
2

Your current project has been set to: [preschedule].
```

Sljedeća naredba kojom se pokreće prijenos aplikacijskih datoteka na App Engine je:

```
./google-cloud-sdk/bin/gcloud app deploy ~/preschedule/
```

Nakon toga slijedi prijenos koji može potrajati do nekoliko minuta, a nakon njega dolaze poruke o završetku prijenosa i poveznica na stranicu na kojoj se nalazi aplikacija koja se prenosila.

S ovom naredbom može se i u naredbenom retku pokrenuti izvođenje aplikacije:

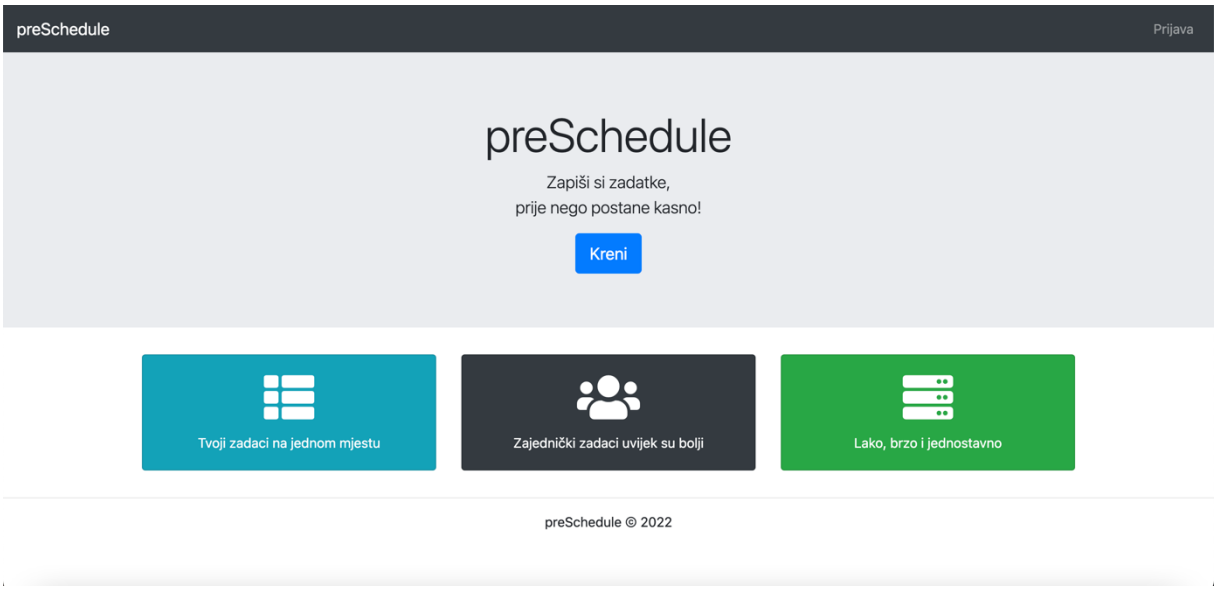
```
gcloud app browse
```

2.6.1. Funkcionalnosti aplikacije preSchedule

Za potrebe ovog rada, osmišljena je i napravljena aplikacija preSchedule kojoj je svrha da poveže zadatke više osoba iz jednog tima s ciljem da tim bude spremniji i upućeniji prije i za vrijeme odvijanja poslovnih sastanaka. Tako primjerice jedna osoba može po potrebi, u određenim vremenskim periodima postavljati svoje zadatke na sustav i to isto mogu raditi i ostale osobe u nekom timu. Nakon nekog vremena u sustavu će biti veći broj zadataka u čiji uvid će imati jednako sve osobe u timu i postati upućene u to što netko drugi radi i može li se to odraziti i na ono što ta osoba radi.

Cilj ove aplikacije je pomoći osobama u nekom timu da se lakše pripreme za poslovne sastanke i povežu određene zadatke u veće zadatke ili kooperativno pristupe rješavanju određenih zadataka koji su po nekim osobinama slični.

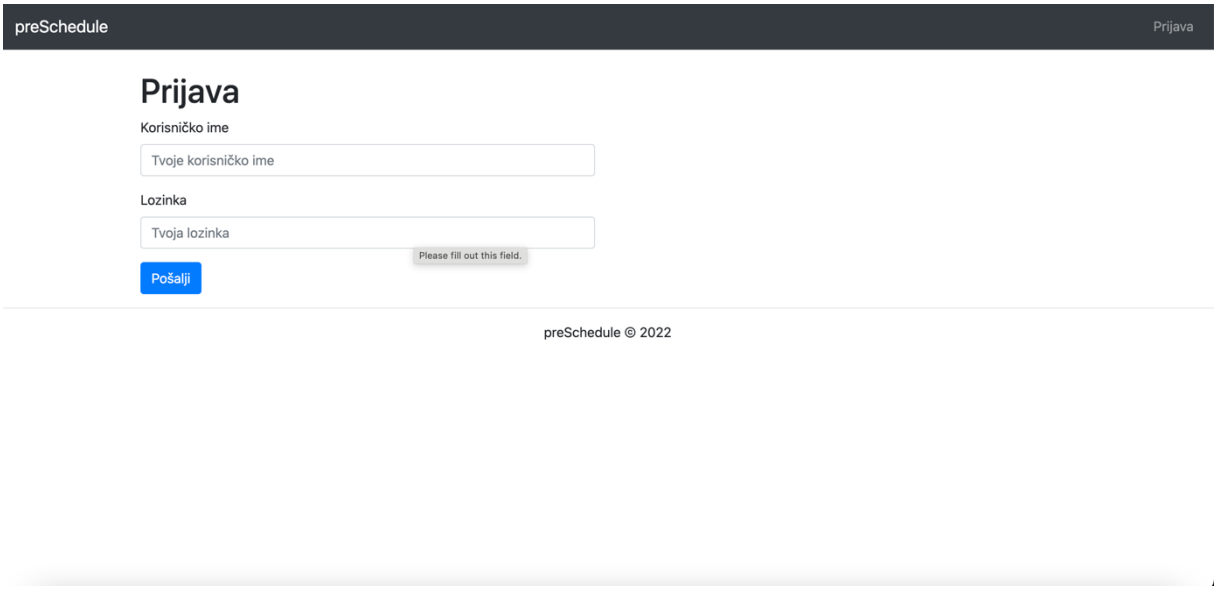
Na slici ispod može se vidjeti kako izgleda početnu sučelje aplikacije koje korisnika informira o tome kako mu aplikacija može služiti i što od nje može očekivati.



Slika 13. Početno sučelje aplikacije preSchedule

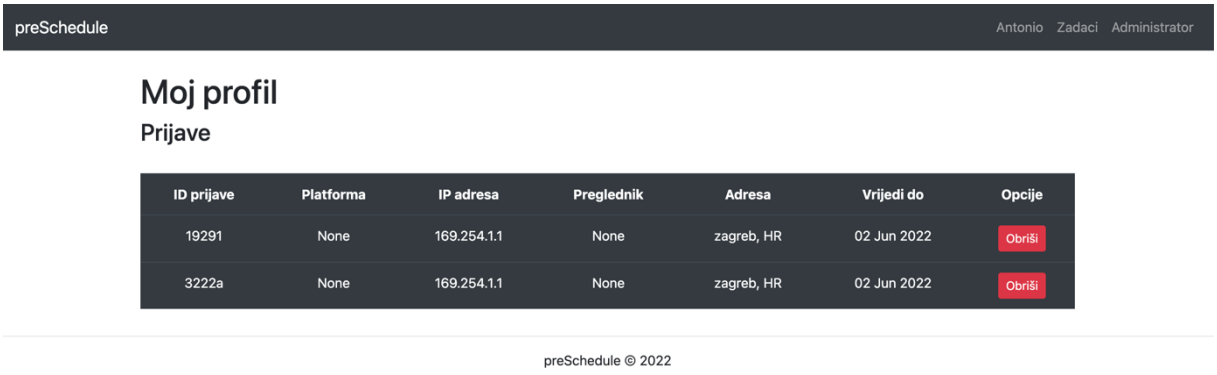
Aplikacija preSchedule sastoji se od nekoliko stranica, a to su:

- naslovna stranica,
- registracija (samo za administratora).
- prijava,
- zadaci,
- nadzorna ploča administratora i
- profil.



Slika 15. Stranica prijave korisnika

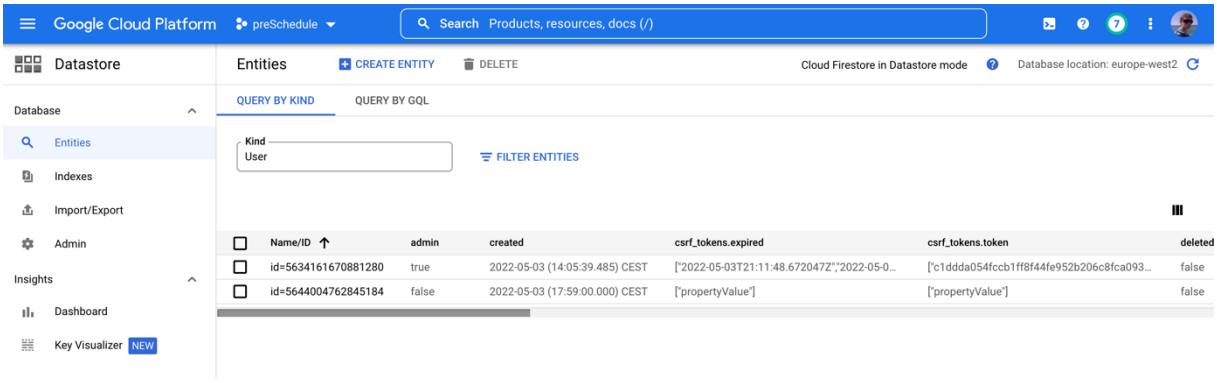
Na slici iznad može se vidjeti prikaz stranice za prijavu korisnika, dok se na slici ispod može vidjeti stranica profila za korisnika (stranica može imati različit prikaz ovisno o tome pristupa li joj administrator ili korisnik).



Slika 14. Stranica profila korisnika

Aplikacija preSchedule napisana je u Python programskom jeziku (najnovija inačica 3.7), razvojnom okruženju Flask koji na pojednostavljeni i intuitivni način omogućava korištenje specifičnih funkcionalnosti, uz dijelove kode napisane u HTML-u, CSS-u, JavaScriptu i jQueryju. Izrađena je u cilju korištenja u Google App Engineu te koristi web servise i Datastore upogonjene na samom App Engineu. Aplikacija je izrađena u razvojnom okruženju i testirana kroz rad lokalno (koristeći lokalni Datastore Viewer, koji se instalira na računalo, mogu se vidjeti podaci koji se lokalno spremaju na Datastore Emulator), no postoji i drugi način na koji se aplikacija može prenijeti na sustav GAE, a to je kloniranjem verzije koja se putem sustava za verzioniranje koda prenijela na neku od platformi za čuvanje i verzioniranje koda, kao što je to primjerice GitHub.

Na slici ispod može se vidjeti prikaz entiteta u Datastore-u aplikacije.



Slika 16. Entiteti u Datastore-u

3. Zaključak

Google App Engine je primjer dobro implementirane *PaaS* usluge. Pojedincu omogućuje širok spektar alata i mogućnosti koje može iskoristiti ako se odluči za implementaciju svojeg rješenja u oblaku. Organizacijama nudi jednostavan i pristupačan način kako da svoje sporedne, a ponekad i ključne usluge ponude u oblaku, kroz Googleov servis.

No iako ovakav pristup sa sobom nosi mnoge prednosti, nedostaci se kriju u tome što osobe ili organizacije svoj rad i trud prepuštaju nekome drugome na održavanje i time snose veliki rizik u slučajevima kada servis ne bi radio, odnosno kada bi zbog tehničkih ili korisničkih poteškoća njihova usluga pala. Tu je na svakome da procijeni koje su mu mogućnosti i koju razinu povjerenja ima u neki tip ovakvih usluga računarstva u oblaku.

Ipak, Google App Engine je kroz duže vremensko razdoblje konstantno nadograđivan i puno je truda i vremena uloženo da takvi servisi danas pružaju visoku razinu usluge koja se može pronaći. Stoga se može zaključiti da je *Paas* model usluga prikladno rješenje u slučajevima za koje je određeno okružje namijenjeno, no i dalje se manje koristi od *IaaS* modela jer nije toliko prikladan za široku upotrebu kao što su to *SaaS* i *IaaS*.

U ovome radu koristio se programski jezik Python te razvojno okruženje Flask koji u kombinaciji odlično funkcioniraju te su često korišteni kod razvoja aplikacija za Google App Engine. Konačna aplikacija preSchedule prikazuje samo najosnovnije mogućnosti koje servis kao što je GAE nudi i za sve zainteresirane postoji širok spektar alata i prikladne dokumentacije za razvoj svojih proizvoda u Google App Engineu.

Popis literature

[1.] M. Čorak, *Sustav za planiranje sastanaka za Google App Engine oblak računala* [diplomski rad]. Elektrotehnički fakultet, Sveučilište Josipa Jurja Strossmayera u Osijeku, 2012. Dostupno: https://www.bib.irb.hr/679144 [pristupano 12.04.2021.]

[2.] N. Ćubić, *Osmišljavanje računalnog oblaka* [diplomski rad]. Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, 2011. Dostupno: https://www.bib.irb.hr/517602 [pristupano 12.04.2021.]

[3.] R. Tomac, Tehno-ekonomska analiza usluga zasnovanih na računarstvu u oblaku [diplomski rad]. Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, 2013. Dostupno: https://www.bib.irb.hr/640871 [pristupano 20.04.2021.]

[4.] A. Tikvica, D. Andročec, „Primjena interoperabilnosti računalnog oblaka u pohrani podataka“, *Zbornik radova veleučilišta u Šibeniku*, str. 161-175, 2018. [stručni rad]. Dostupno: https://www.bib.irb.hr/983570 [pristupano 25.04.2021.]

[5.] W. J. Chun, *Core Python Applications Programming* (treće izdanje), Pearson Education, 2012.

[6.] D. Sanderson, *Programming Google App Engine with Python* (prvo izdanje), O'Reilly Media, 2015.

[7.] Google Cloud [web mjesto]. Dostupno: https://cloud.google.com [pristupano 12.10.2021.]

Popis slika

Popis tablica