Razvoj aplikacije u oblaku

Brković, Filip

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Filip Brković

RAZVOJ APLIKACIJE U OBLAKU

ZAVRŠNI RAD

Varaždin, 2024.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Filip Brković

JMBAG: 0135242398

Studij: Primjena informacijske tehnologije u poslovanju

RAZVOJ APLIKACIJE U OBLAKU

ZAVRŠNI RAD

Mentor/Mentorica:

Dr. sc. Darko Andročec

Varaždin, veljača 2024.

Filip Brković

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/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Rad opisuje računarstvo u oblaku. Opisane su prednosti koje poduzeće ostvaruje korištenjem računarstva u oblaku, isto kao i izazovi s kojima se korisnici susreću prilikom implementacije računarstva u oblaku. Rad opisuje načine izvedba računarstva u oblaku i gdje se primjenjuju te su opisani modeli za koje se usluga računarstva u oblaku pruža. Osim međusobne usporedbe modela pružanja usluge, uspoređeni su i pružatelji usluga spomenutih modela.

Za praktični dio rada izrađena je aplikacija za nadmetanje putem interneta koja je dostupna na jednom od pružatelja usluga u oblaku. Opisana je ideja aplikacije te sve funkcionalnosti koje ona pruža. Za opis tehničkog dijela aplikacije naveden je programski jezik i razvojni okviri koji su korišteni s njim, baza podataka koju aplikacija koristi te repozitorij na koji je izvorni kod spremljen. Uz svaki opis tehničkog dijela aplikacije priložen je i programski kod. Na kraju ovog rada temeljen prikupljenih informacija donijeli smo zaključnu misao.

Ključne riječi: aplikacija, web-aplikacija, platforma kao servis, oblak, razvoj aplikacije, računarstvo u oblaku

Sadržaj

1. Uvod

Tema završnog rada je razvoj aplikacije u oblaku. S razvojem tehnologije i širenjem povezanosti putem interneta pojavljuje se sve veća potreba za pristupom našim podacima s udaljenih lokacija. Kao rješenje za navedeni izazov pojavljuje se računarstvo u oblaku.

U prvom dijelu ovog rada opisat ćemo što je to računarstvo u oblaku, što njegova definicija govori. Objasnit ćemo zašto računarstvo u oblaku ima pozitivan učinak na računarstvo te ćemo nabrojati i objasniti prednosti koje računarstvo u oblaku ima nad lokalnom izvedbom. Potom ćemo se osvrnuti na izazove koji se javljaju korištenjem računalstva u oblaku. Opisat ćemo na što bi korisnici trebali obratiti posebnu pažnju prilikom donošenja odluke o korištenju računalstva u oblaku za njihovo poslovanje. U posebnom poglavlju osvrnut ćemo se na ključne sigurnosne prijetnje koje se mogu pojaviti prilikom korištenja računarstva u oblaku. Navest ćemo moguće napade koji se mogu pojaviti kod računarstva u oblaku. Nakon prednosti i izazova opisat ćemo načine izvedba računarstva u oblaku. Svaku izvedbu ćemo opisati i navesti u kojim se situacijama koristi. U nastavku ćemo pažnju posvetiti modelima računarstva u oblaku. Svaki od modela ćemo opisati navesti pružatelje usluga koji u svojoj ponudi imaju opisani model. Kod svakog modela napravit ćemo usporedbu odabranih pružatelja usluga po odabranim kriterijima.

U drugom dijelu ovog rada opisat ćemo izrađenu aplikaciju koja je bila dio praktičnog zadatka. Opisat ćemo ideju iza projektnog zadatka i funkcionalnosti koje aplikacija podržava, zatim ćemo opisati uz pomoć kojih programskih jezika i alata je izrada aplikacije realizirana. U nastavku ćemo navesti pružatelja usluge kojeg smo odabrali za objavu aplikacije na Internet detaljnije ćemo opisati kako smo izradili dizajn, arhitekturu aplikacije te kako smo formirali sheme baze podataka. Opisat ćemo protokol s pomoću kojeg smo aplikaciji dodali posebnu funkcionalnost. Svaki opis poduprijet ćemo programskim kodom.

Na kraju ćemo na temelju prikupljenih informacija zaključiti konačnu misao.

2. Metode i tehnike rada

U završnom radu za razradu teme koristit ćemo se izvorima dostupnim na internetskim stranicama, isto kao i dokumentacijama izrađenim od strane promatranih subjekata. Koristit ćemo se i znanstvenim radovima dostupnim u raznim repozitorijima.

Prikupit ćemo međusobno srodne podatke te ćemo ih međusobno usporediti i prikazati u *Excel* tablicama kako bi usporedba bila preglednija.

Za praktični dio rada koristit ćemo se JavaScript programskim jezikom. *Node.js* [1] izvršno okruženje *i Express.js* [2] razvojno okruženje koristit ćemo za pisanje poslužiteljskog dijela programskog koda, dok za pisanje programskog koda sučelja nećemo koristiti razvojne okvire već osnovni JavaScript. Za pisanje programskog koda koristit ćemo *VSCode* [3] uređivačem programskog koda s proširenjem *Prettier* [36] za automatsko formatiranje programskog koda i alat *ESLint* [37] za statičku analizu programskog koda kako bi se prijevremeno identificirali i ispravili potencijalni problemi. Zbog vrste podataka koje obrađujemo u aplikaciji, za bazu podataka koristit ćemo *MongoDB* [4] ne-relacijsku bazu podatka te *Mongoose* [5] razvojni okvir za olakšanu interakciju.

3. Računarstvo u oblaku

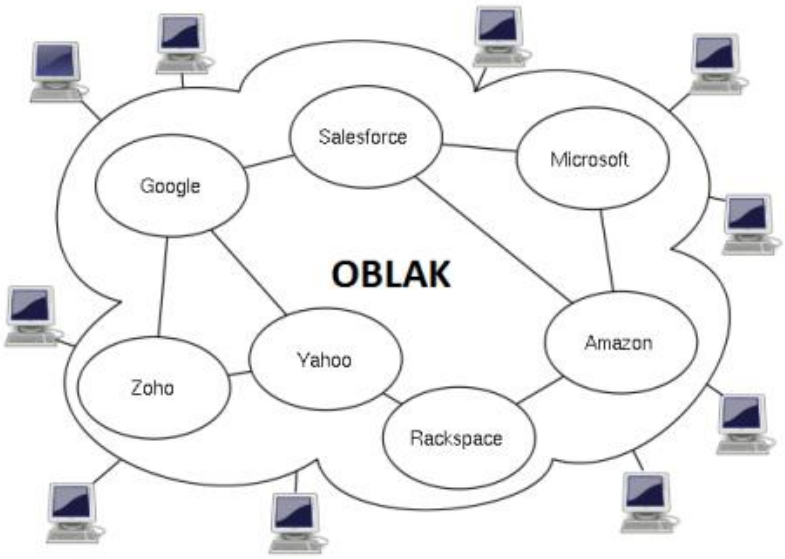
3.1. Računarstvo i računarstvo u oblaku

Kako bi mogli razrađivati temu razvoja aplikacije u oblaku prvo moramo definirati što je to računarstvo u oblaku. Definicija prema Nacionalnom institutu standarda i tehnologije govori da je računarstvo u oblaku model koji omogućuje sveprisutno, praktično i dostupno na zahtjev mrežno pristupanje konfigurabilnim računalnim resursima (npr. mreže, poslužitelji, pohrana, aplikacije i usluge) koji se mogu brzo osigurati i osloboditi uz minimalan napor pružatelja usluge [6].

Prema definiciji je utvrđeno da je računarstvo u oblaku samo dodatak koji nam omogućuje da koristimo razne resurse i usluge uz pomoć interneta koje bi nam u suprotnom bile nedostupne ili dostupne isključivo lokalno [7].

Neke od primjena oblaka su:

- Računarstvo u oblaku i „Big data“
- Internet stvari (eng. *Internet of things* - IoT)
- Razvoj i testiranje softvera
- Pohrana u oblaku
- Mrežne igre



Slika 1 - Ilustracija računarstva u oblaku (https://www.cis.hr/www.edicija/LinkedDocuments/NCERT-PUBDOC-2010-03-293.pdf)

3.1.1. Prednosti računarstva u oblaku

Računarstvo u oblaku na prvi pogled može djelovati kao model koji je previše ovisan o trećoj strani, ali ta tehnologija nosi mnogo prednosti u usporedbi s lokalnim instalacijama.

Jedna od prednosti je skalabilnost. Sama je priroda interneta takva da nas često može uhvatiti nespremne i predstaviti nam nove izazove ili prilike pa isto tako i naše poslovanje može u jednom trenutku zahtijevati puno više resursa nego što ih imamo na raspolaganju. Stoga u slučaju da koristimo računarstvo u oblaku ti uvjeti često mogu biti brzo ispunjeni uz promjenu plana kojeg koristimo za naše poslovanje dok bi taj proces u lokalnoj izvedbi trajao duže [8].

Zajedno uz skalabilnost kao prednost dolazi i fleksibilnost plaćanja. Dok manja poduzeća u svojim počecima nemaju potrebu za vrhunskom opremom i podrškom svoje potrebe mogu ispuniti s povoljnijim planovima koji mogu biti unaprijeđeni u svakom momentu [8].

Proizvod dostupan uz pomoć računarstva u oblaku bit će ažuriran na posljednju inačicu u svakome trenu. Dok kod lokalnih instalacija softvera u poduzećima s velikim brojem računala proces ažuriranja softvera može biti iznimno dug proces, s računarstvom u oblaku problemi takve prirode se zanemaruju [9].

Korisnik kako bi mogao kvalitetno koristiti uslugu i sve njezine pogodnosti ne treba posjedovati znanje o svim dijelovima usluge, već mu je samo potrebno znanje usko vezano uz samu uslugu koju koristi što uvelike olakšava korištenje usluge i omogućuje joj da stekne veći broj korisnika [8], [9], [10].

3.1.2. Izazovi računarstva u oblaku

Tehnologija oblaka nam pruža mnogo prednosti nad lokalnom izvedbom, ali uz sve prednosti se pojavljuju i nedostaci koje bi korisnik trebao imati na umu prilikom potpune migracije na računarstvo u oblaku.

Jedan od tih nedostataka je stalna potreba za internetskom vezom. Kako je svaki korak procesa korištenja računarstva u oblaku izveden na udaljenoj lokaciji uz pomoć interneta, potrebno je osigurati stalnu i kvalitetnu vezu. Međutim, prekid veze je nerijetka pojava i može se pojaviti u bilo kojem trenutku. Također, prekid veze nije vezan samo uz stranu klijenta već se incidenti mogu dogoditi i na strani pružatelja usluge što može uzrokovati smanjenje kvalitete usluge i nezadovoljstvo kod korisnika.

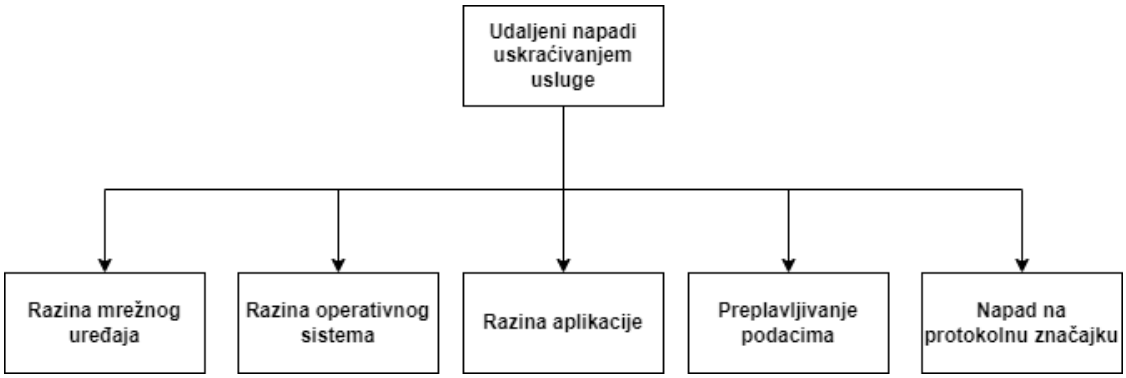
Sigurnost i privatnost su još jedna od ključnih stavaka. Ako je odlučeno da se poslovanje vrši uz pomoć računarstva u oblaku, to znači da je većina informacija vezanih uz naše poslovanje dostupna putem interneta. Nije upitno da pružatelji usluga koriste napredne algoritme za kriptiranje kako bi zaštitili podatke, ali to ne znači da ne postoji mogućnost da naši podaci dođu do zlonamjerne treće strane koja će ih iskoristiti u svoju korist. Stoga je važno da se prije odluke korištenja računarstva u oblaku detaljno istraže sve pogodnosti i mogućnosti koje pružatelji usluge koriste.

U slučaju da poduzeće nije zadovoljno trenutnom uslugom ili trenutna usluga ne ispunjava njegove uvjete, sam proces migracije često je vrlo kompleksan i dugotrajan što dodatno povećava važnost kvalitetne analize svih aspekata poslovanja i pogodnosti koje pružatelj usluge nudi te njihovu kompatibilnost [11].

3.1.2.1. Ključne sigurnosne prijetnje

Kao što smo to naveli prije, sigurnost igra veliku ulogu kod računarstva u oblaku. Stoga je važno prepoznati koji vrste napada su moguće kako bi se mogle smisliti efektivne solucije za smanjenje broja napada i u krajnjem slučaju prevenciju samih napada. U nastavku ovog poglavlja nabrojat ćemo i opisati ključne vrste napada koji se pojavljuju kod računarstva u oblaku [12].

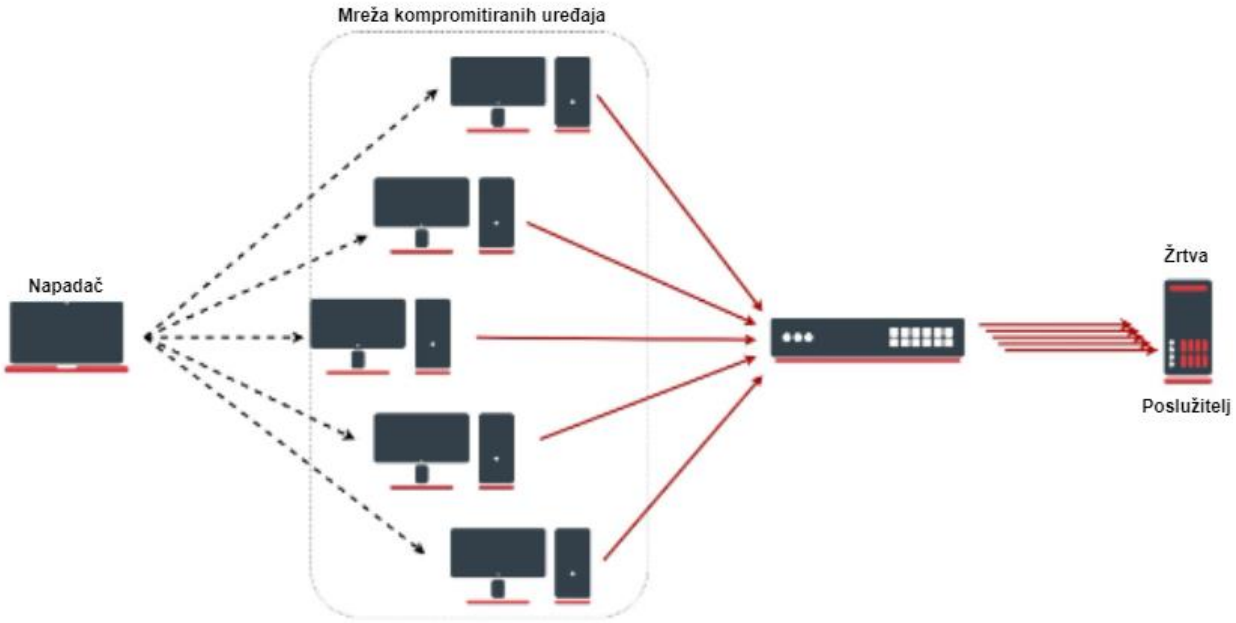
- **Napad omotačem potpisa XML-a** (eng. *Wrapping attack*) - Napad omotačem potpisa XML-a iskorištava slabost u tome što element potpisa ne pruža nikakve informacije o tijelu poruke na koju se odnosi. Time je omogućeno napadaču da promijeni sadržaj tijela poruke i unese zlonamjerni kod, a da pritom ne naruši valjanost potpisa. Drugim riječima, napadač omotava XML potpis oko zlonamjernog koda i prosljeđuje ga kao autentičnu poruku [12].
- **Napadi preplavljivanjem** (eng. *Flooding attacks*) – S obzirom na to da je posao oblaka da poslužuje sve klijente. Ako dođe do povećanja zahtjeva od strane klijenata oblak mora odmah reagirati povećanjem korištenja raspoloživih resursa. Zbog takvog načina rada servera otvara se mogućnost zlonamjernim pojedincima da iskoriste ovu značajku. Napadač izrađuje velik broj lažiranih zahtjeva te zbog toga poslužitelj mora kontrolirati i filtrirati svaki od dolaznih zahtjeva. Kao posljedica poslužitelj je preplavljen lažnim zahtjevima, dok su stvarni zahtjevi odbačeni, što dovodi do distribuiranog napada uskraćivanja usluge (eng. *distributed denial of service*, skr. *DDoS)* [12].



Slika 2 - Vrste napada uskraćivanjem usluge (Izvor: S. El Kafhali, I. El Mir, i M. Hanini, „Security Threats, Defense Mechanisms, Challenges, and Future Directions in Cloud Computing“, Archives of Computational Methods in Engineering)

- **Napad krivotvorenjem meta podataka** (eng. *Metadata Spoofing Attack)* – Kako bi se uspostavila komunikacija putem interneta, klijent mora dohvatiti potrebne informacije za povezivanje na uslugu. Te informacije uključuju internetsku adresu usluge, format poruke, mrežnu lokaciju te sigurnosne zahtjeve. Te informacije klijent prikuplja s pomoću meta podataka. *Web Service Definition Language (*skr. *WSDL)* i *WS-Security-Policy* su poznati dokumenti s meta podacima te zbog činjenice da implementiraju HTTP protokol ili e-poštu, povećava se šansa za napadom krivotvorenja. Sistemi oblaka sadržavaju funkcionalnost WSDL repozitorija te postoji mogućnost povratka WSDL datoteke usluge na dinamičan način i distribucija zlonamjerne WSDL datoteke diljem mreže [12].
- **Nesigurno aplikacijsko programsko sučelje** (eng. *Insecure API*) – Korisnici oblaka obično koriste kolekciju aplikacijskih programskih sučelja (skr. API) kako bi komunicirali i upravljali s uslugama na oblaku. Kako aplikacije s pomoću API-a izvršavaju niz zadataka od pružanja usluga do izvršavanja sigurnosnih funkcija, važno je da su precizno definirani prema strogim politikama koje uklanjanju povoljne prilike za napadače [12].
- **XSS napad** (eng. *Cross-Site Scripting Attack,* skr. *XSS)* – XSS napadi predstavljanju vrstu injekcija, prilikom koje se u poznate i pouzdane internetske stranice ubacuju zlonamjerne skripte s ciljem da se prenesu drugim korisnicima. XSS napadi obično se baziraju na obrasce u pregledniku korisnika te uzrokuju netočnu validaciju korisnikovog unosa što kao rezultat napadači jednostavno mogu iskoristiti [12].

- **Napad SQL injekcijom** (eng. *SQL injection attack*) – Napadi SQL injekcijom obično su posljedica dizajna internetske aplikacije. Napadač u polja s podacima unosi zlonamjeran programski kod sintakse SQL upita. Tako napadač ostvaruje neautorizirani pristup bazi podataka s pomoću kojeg može preuzeti privatne i osjetljive podatke, isto kao što ih može i pokvariti [12].



Slika 3 - Napad uskraćivanja usluge
(https://help.mikrotik.com/docs/pages/viewpage.action?pageId=28606504)

3.2. Modeli provođenja računarstva u oblaku

Kako bi se usluge mogle pružati s pomoću tehnologije oblaka postoje četiri modela izvedbe računarstva u oblaku. Ovisno o specifičnim potrebama svaki model je izveden na različiti način. U nastavku ćemo modele opisati [13].

- **Javni oblak** (eng. *Public Cloud*) – kod ovog modela izvedbe računarstva u oblaku platforma je u vlasništvu poduzeća koje pruža usluge računarstva u oblaku. Pristup oblaku dostupan je svima, neovisno radi li se o pojedincu ili organizaciji. S obzirom na to da je platforma javna posebno se obraća pažnja na sigurnost podataka. Ako je prilikom realizacije javnog oblaka pažnja usmjerena na izvedbu, sigurnost i položaj podataka aplikacije koja se pokreće u oblaku ne bi trebao stvarati probleme prema arhitekturi oblaka i njegovim korisnicima [13].

  Jedna od prednosti javnog oblaka je ta što mogu biti znatno veći nego privatni oblaci te je u ponudi mogućnost da korisnik zakupljeni dio poveća ili smanji ovisno o potrebama poduzeća [13].

  Kao primjere javnih oblaka možemo nabrojati: *Amazon Web Services* [14], *Microsoft Azure* [15], *Google Cloud Platform* [15], *IBM Cloud* [41].

- **Privatni oblak** (eng. *Private Cloud*) – ovaj model pruža infrastrukturu računarstva u oblaku isključivo jednoj organizaciji. Upravljanje infrastrukturom može odrađivati sama organizacija ili se taj zadatak može prepustiti nekome drugom. Poduzeća koriste privatne oblake kada žele ili imaju potrebu za većim nadzorom podataka nego što li je to moguće pomoću javnog oblaka [13].

  Svrha privatnih oblaka je da pružaju najveći mogući nadzor i sigurnost podacima pohranjenim na njih. Infrastruktura privatnih oblaka može biti u vlasništvu poduzeća, ali može biti i raspoređena unutar podatkovnog centra [13].

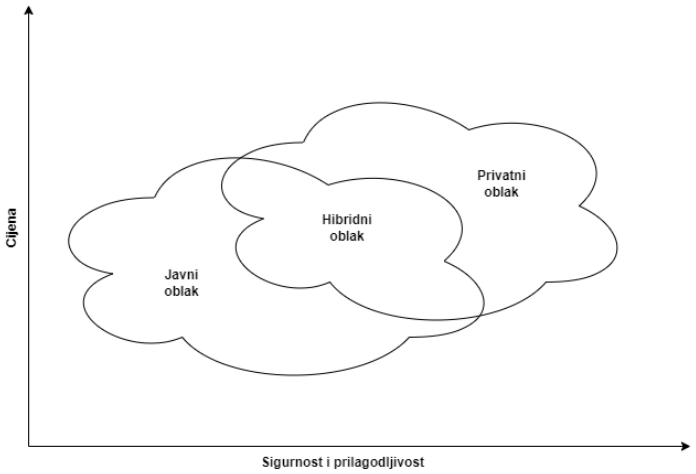
  Za primjer pružatelja usluge privatnih oblaka možemo spomenuti *VMware* [38], *SAP* [39], *OpenStack* [40].

- **Zajednički oblak** (eng. *Community Cloud*) – ako nekoliko organizacija želi dijeliti strukturu oblaka koristi se model zajedničkog oblaka. Infrastruktura je izgrađena da podržava zajedničke potrebe poduzeća. Zajedničkim oblakom mogu upravljati same organizacije isto kao i neka treća strana poput pružatelja usluge [13].

- **Hibridni oblak** (eng. *Hybrid Cloud*) – kod modela hibridnog oblaka dolazi do kombinacije dva ili više različitih oblaka. Svaki oblak ostaje posebni entitet ali je međusobno povezan sa standardiziranim tehnologijama koje omogućuju prijenos podataka ili komunikaciju aplikacija na siguran i efikasan način [13].

  Hibridni model povezuje javni i privatni oblak koji omogućuje rješenje problema proširivanja dostupnih resursa kod privatnih oblaka te isto tako i rješenje veće razine sigurnosti određenih podataka koja kod javnih oblaka nije dostupna. Jedan od izazova hibridnih oblaka je sama raspodjela aplikacija po javnom i privatnom oblaku. Uz problem raspodjele potrebno je uzeti u obzir obrađuju li se resursi ili su dostupni samo za čitanje [13].

  Kao primjer usluge hibridnog oblaka možemo navesti *Amazon*-ovu uslugu *AWS Outposts* [14], te isto tako i *Googleovo* rješenje *Anthos* [15].



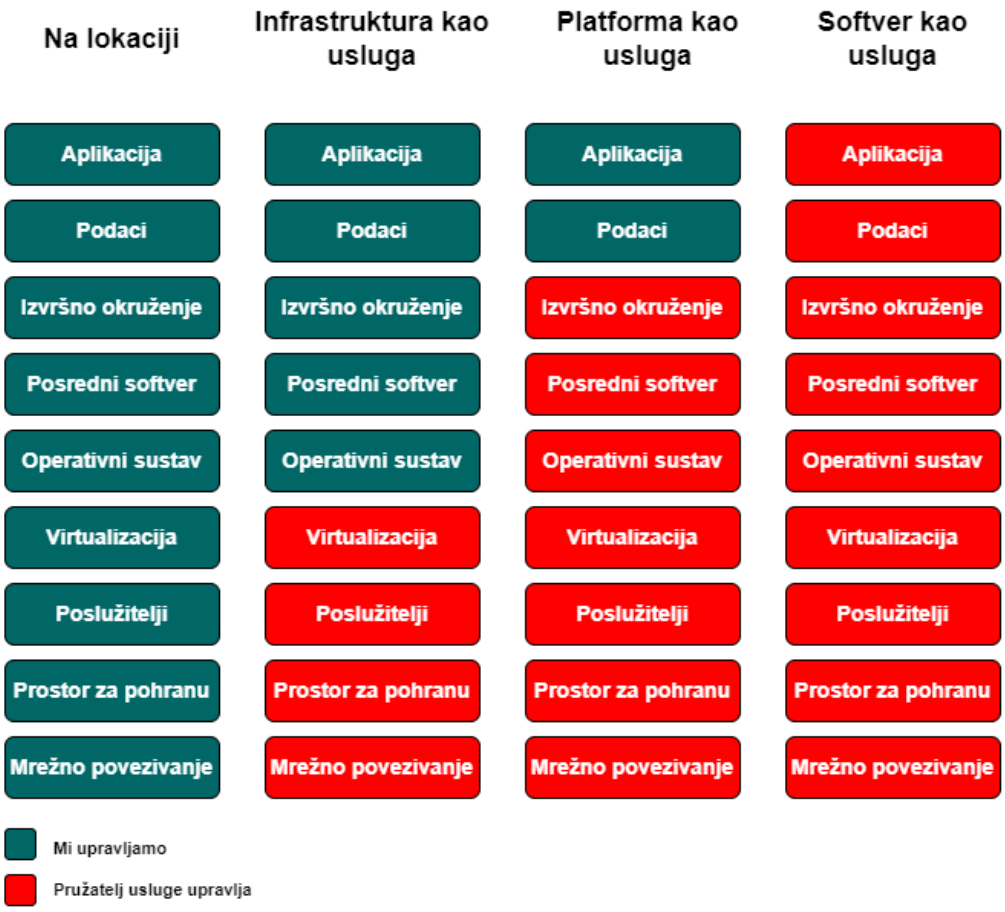
Slika 4 - Prikaz odnosa modela izvedbe računarstva u oblaku (Prema: https://www.businesstechweekly.com/operational-efficiency/cloud-computing/private-cloud-vs-public-cloud/)

3.3. Modeli pružanja usluga kod računarstva u oblaku

Postoje 3 modela računarstva u oblaku:

1. Infrastruktura kao usluga (eng. *Infrastructure as a service - IaaS*)
2. Platforma kao usluga (eng. *Platform as a service - PaaS*)
3. Softver kao usluga (eng. Software *as a service - SaaS*)

U svakom od modela se broj usluga koje su dostupne povećava prema redoslijedu kako su navedeni.



Slika 5 - Usporedba modela računarstva u oblaku (Prema: https://www.redhat.com/en/topics/cloud-computing/iaas-vs-paas-vs-saas)

3.3.1.Infrastruktura kao usluga

U modelu infrastruktura kao usluga pružatelj usluge svojim korisnicima osigurava infrastrukturu na kojoj korisnici mogu instalirati svoje virtualne strojeve, operacijske sustave i aplikacije dok se pružatelj obavezuje da isporuči dogovorenu razinu usluge. Pružatelj usluge ima obavezu upravljati s poslužiteljima, osigurati dovoljne količine prostora za pohranu, osigurati kvalitetnu i pouzdanu vezu između servera, te isto tako napraviti virtualizaciju usluge kako bi se korisnik mogao lakše njom koristiti. Također kao karakteristiku infrastrukture kao usluge možemo navesti i varijabilnu cijenu koja je prilagođena količini usluge koju korisnik koristi ([9], [16]).

3.3.1.1. Usporedba pružatelja infrastrukture kao usluge

U ovom poglavlju usporedit ćemo tri popularna pružatelja infrastrukture kao usluge. Pružatelji usluga su *Amazon Web Services* [14]*, Microsoft Azure*[15]*, Google Cloud Platform*[17].

Slika 6 - Uspoređeni pružatelji usluga (https://michelburnett27.medium.com/gcp-vs-aws-vs-azure-6532eb8b3ab0)

Pružatelje usluga usporedit ćemo prema postotku raspoloživosti koja je dogovorena sporazumom o razini usluge, (eng. *Sevice Level Agreement,* skr. *SLA*) ugovorom za uslugu iznajmljivanja virtualnih strojeva, cijeni iznajmljivanja virtualnih strojeva te cijeni spremljenih podataka na ne-relacijsku bazu podataka.

Tablica 1 - Usporedba pružatelja infrastrukture kao usluge

Pružatelji usluga	Postotak raspoloživosti za virtualne strojeve	Cijena virtualnih strojeva prema vremenu korištenja	Cijena ne-relacijske baze podataka prema spremljenim podacima
Amazon Web Services (AWS)	**99.50%**	**€0.1429 / h**	**25 GB / mj - besplatno / €0.285/GB**
Microsoft Azure	**99.50%**	**€0.1695 / h**	**25 GB / mj - besplatno / €0.232/GB**
Google Cloud Platform	**99.90%**	**€0.1860 / h**	**1 GB / dan - besplatno / €0.16/GB**

(Izvor: https://aws.amazon.com/, https://azure.microsoft.com/, https://cloud.google.com/)

Kod usporedbe raspoloživosti usluga velikih kompanija kojima je jedna od djelatnosti pružanje infrastrukture kao usluge, nije iznenađujuće da je postotak raspoloživosti prilično blizu maksimuma. U našem slučaju uspoređivali smo *SLA* ugovorom definiranu raspoloživost za iznajmljivanje virtualnih strojeva za jednu instancu sa standardnim karakteristikama te u tablici 1 možemo vidjeti kako *Google Cloud Platform* [17] jamči najveću raspoloživost. U slučaju da ugovorena raspoloživost nije bila ostvarena, korisnicima je u obliku kredita za uslugu odobren iznos u postotku njihovog mjesečnog računa koji varira od 10 % pa sve do 100 % ukupnog iznosa računa.

Kako bi usporedili cijene korištenja virtualnih strojeva morali smo odabrati specifikacije virtualnih strojeva kako bi usporedba bila što točnija. Odabrali smo da ćemo uspoređivati virtualne strojeve s 4 virtualna procesora, 16 GB RAM-a i Linux operacijskim sustavom te također, za lokaciju našeg poslužitelja odabrali smo Njemačku. Pružatelji usluga naplaćuju korištenje virtualnih strojeva prema definiranoj satnici. Prema podacima iz tablice 1 *Amazon Web Services* [14] je najpovoljniji između tri uspoređena.

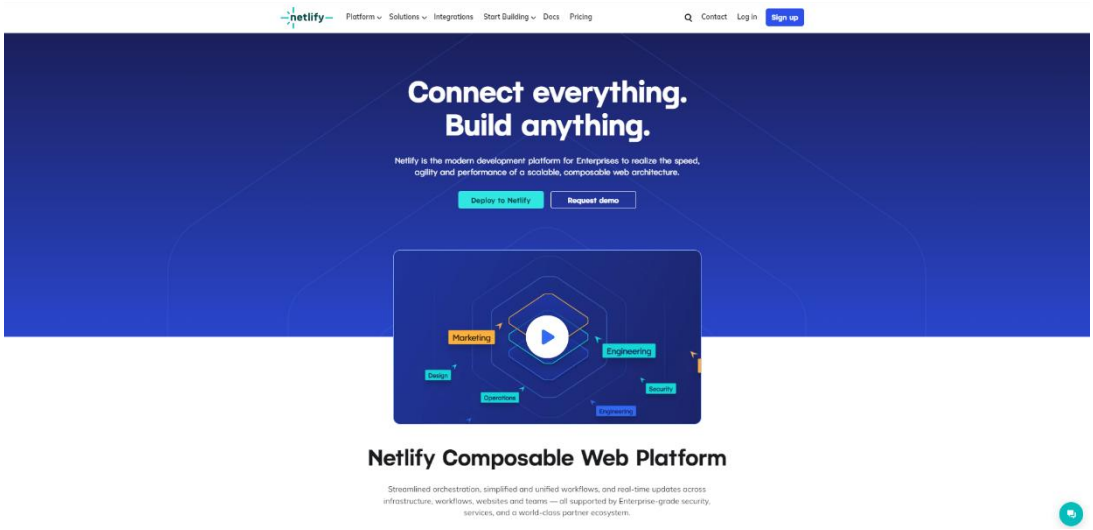
Kao posljednju usporedbu uzeli smo cijene pohrane podataka na ne-relacijskim bazama podataka. Svaki od pružatelja usluga pruža određenu količinu prostora besplatne te naplaćuje svaki GB koji prelazi limit prema definiranoj cijeni. *Amazon Web Services* [14] i *Microsoft Azure* [15] svojim korisnicima pružaju 25 GB mjesečno besplatno dok *Google Cloud Platform* [17] pruža 1 GB besplatnog prostora za podatke, ali isti i kontrolira na dnevnoj bazi. Prema definiranim cijenama *Googleovo* rješenje ima najjeftiniju cijenu, ali zbog manjeg limita i dnevnog kontroliranja vjerojatnost za prekoračenjem je veća, stoga je teško odlučiti koji je najpovoljniji izbor uzimajući u obzir samo cijenu.

3.3.2. Platforma kao usluga

Kod platforme kao usluge korisnik upravlja samo s podacima i aplikacijama dok pružatelj usluge ima obavezu isto kao i kod infrastrukture kao usluge upravljati poslužiteljima, prostorom za pohranu, kvalitetom veze i virtualizacijom. Isto tako i upravlja s i operacijskim sustavom koji usluga koristi, određuje izvršno okruženje u kojemu se izvodi i stvara dodatne usluge koje povećavaju kvalitetu same usluge. Platforma kao usluga koristi se za razvoj aplikacija koje mogu biti za osobne potrebe poduzeća ili za njihovu prodaju na ciljanome tržištu. Hardverom i svim procesima koji su vezani uz njega upravlja isključivo pružatelj usluge stoga klijent ima mogućnost iznajmiti onoliko resursa koliko mu je potrebno čime štedi vrijeme i novac i olakšava cijeli proces razvoja aplikacije ([9], [16], [18]).

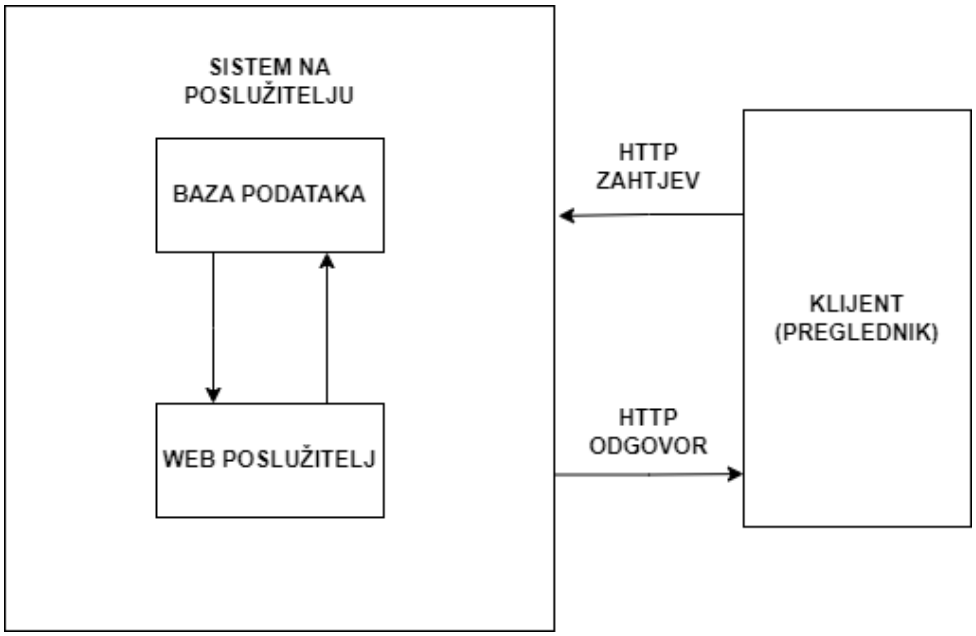
Kao neke od popularnih platforma kao usluga možemo navesti sljedeće:

1. Netlify
2. Heroku
3. Vercel
4. Amazon Web Services (AWS) Elastic Beanstalk
5. Google App Engine



Slika 7 - Početna stranica Netlify.com (izvor: https://www.netlify.com/)

Na spomenute platforme prenosimo poslužiteljski dio koda naše aplikacije koji ima ulogu od odrađivanja kompleksnijih zadataka do samog prijenosa obrađenih podataka klijentu kako bi preglednik mogao dinamički prikazivati stranice. Kao primjer dinamičkih internetskih stranica možemo uzeti stranice profila određenog korisnika, prikaz pretraživanja određenog artikla u internetskoj trgovini i slično.



Slika 8 - HTTP zahtjev/odgovor ciklus (Prema: https://www.geeksforgeeks.org/state-the-core-components-of-an-http-response/)

3.3.2.1. Računarstvo bez poslužitelja

Osim samog hosting-a neki pružatelji usluge također u svojim platformama nude i mogućnost računarstva bez poslužitelja.

Ime „računarstvo bez poslužitelja“ pomalo je zbunjujuće zato što se procesi i dalje odrađuju uz pomoć poslužitelja samo što je koncept taj da se programeri primarno bave kodom sučelja aplikacije dok je poslužiteljski dio koda aplikacije izvršen od strane pružatelja usluge. Na primjer AWS-a usluga koja izvršava funkcije zove se „Lambda“ dok se ta usluga kod Microsoft Azurea naziva: „Serverless functions“ [19].

Također koncept računarstva bez poslužitelja zasniva se na tome da korisnik plaća onoliko koliko je samu uslugu koristio.

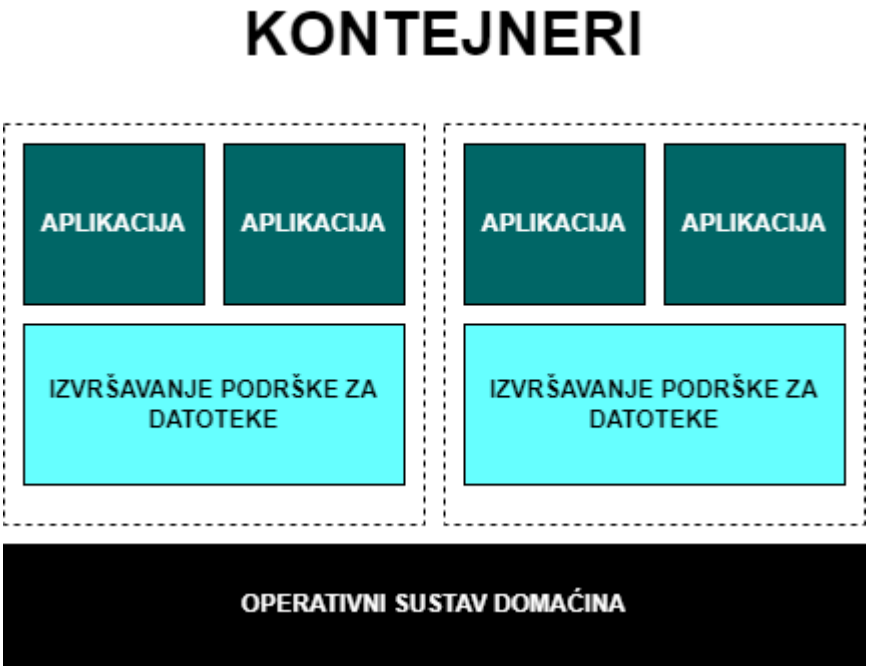
3.3.2.2. Usporedba pružatelja platforme kao usluge

Za usporedbu pružatelja platforme kao usluge uzet ćemo *Heroku* [20], *Netlify* [21] i *Vercel* [22]. Kako su modeli naplate važan čimbenik kod odabira pružatelja usluge, napravit ćemo četiri usporedbe. Prva usporedba fokusirat će se samo na ponuđene modele i njihove cijene, drugom usporedbom obuhvatit ćemo podržane programske jezike, trećom usporedbom fokusirat ćemo se na razvojne okvire koje platforme podržavaju, te ćemo u četvrtoj usporedbi obuhvatiti integracije koje navedene platforme podržavaju.

Tablica 2 - Usporedba modela naplate pružatelja platforme kao usluge

Modeli naplate				
Heroku	*ECO*	*BASIC*	*STANDARD*	*PERFORMANCE*
	$5 / 1000h	**<= $7/mj**	**Standard 1X - <= €23.04/mj** **Standard 2X - <= €46.08/mj**	**Performance M - <= €230.40/mj** **Performance L - <= €460.80/mj**
Netlify	*STARTER*	*PRO*	*ENTERPRISE*	-
	€0 - 1 korisnik	**€17.51 / korisnik**	**Prilagođeno prema upitu**	
Vercel	*HOBBY*	*PRO*	*ENTERPRISE*	-
	€0 - 1 korisnik	**€18.43 / kosrinik**	**Prilagođeno prema upitu**	

Pogledom na tablicu 2 odmah možemo uočiti kako *Heroku* [20] u svojoj ponudi nema besplatan plan dok *Netlify* [21] i *Vercel* [22] nude besplatni plan za jednog korisnika. *Heroku* [20] svoje planove zasniva prema karakteristikama Linux kontejnera koje naziva „dynos“ (slika 9), tako s boljim specifikacijama kontejnera plaćamo veću cijenu. *Netlify* [21] i *Vercel* [22] svoje planove zasnivaju prema broju korisnika. Za 1 osobu korisnik besplatno ima pristup platformama te ako želi svoje projekte odrađivati s timom ljudi mora platiti za *Netlify* [21] 17.51 € ili za *Vercel* [22] 18.43 € po dodatnom korisniku. *Netlify* [21] i *Vercel* [22], osim mogućnosti kolaboracije s timom, kod plaćenih planova korisnicima pružaju niz pogodnosti od kojih su neke: analize internetske stranice, raspoloživost ugovorena *SLA* ugovor od 99.99%, veće brzine, veću količinu funkcija koje računarstvo bez poslužitelja može izvršiti te još unaprijeđenih besplatnih funkcija.

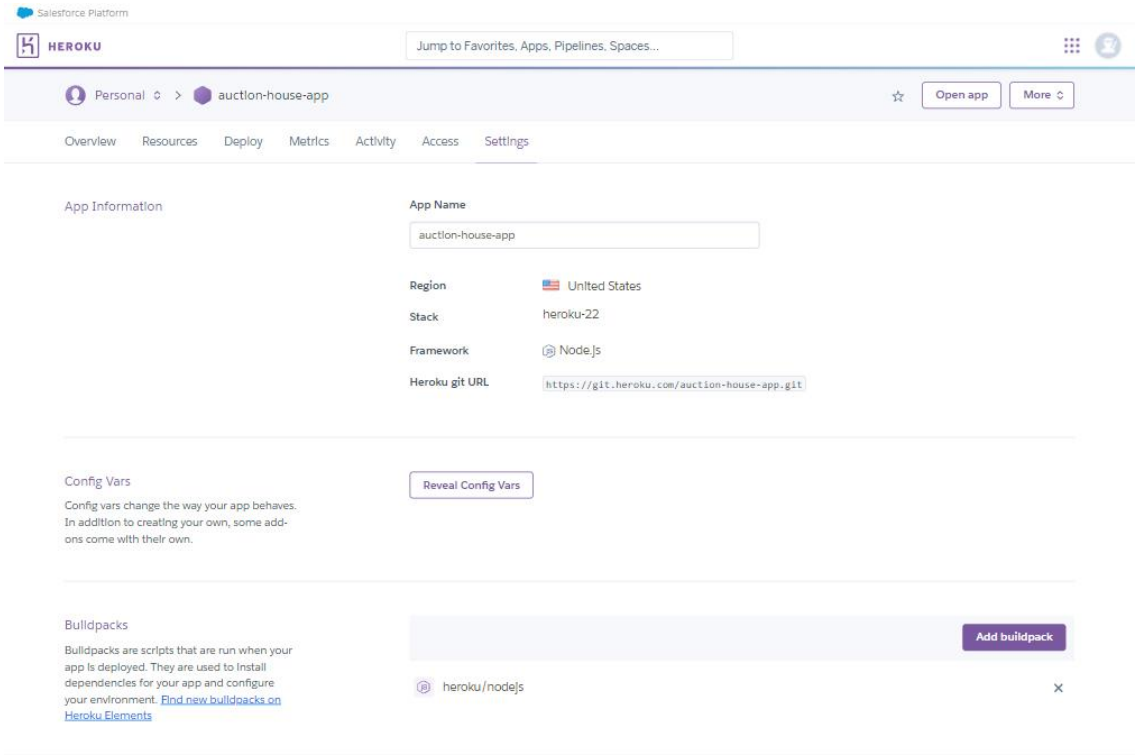


Slika 9 - Prikaz aplikacija u kontejneru (Prema: https://www.redhat.com/en/topics/containers/whats-a-linux-container)

Tablica 3 - Podržani programski jezici

Programski Jezici	Heroku	Vercel	Netlify
JavaScript/Node.js	Da	Da	Da
TypeScript	Da	Da	Da
Python	Da	Da	Da
Ruby	Da	Da	Da
Java	Da	Da	Da
Go	Da	Da	Da
HTML/CSS	Da	Da	Da
PHP	Da	Da	-

Uspoređujući podršku programskih jezika koje odabrane platforme nude možemo primijetiti da je većina jezika podržana. Najveće razlike javljaju se kod PHP i C# programskog jezika. Dok *Netlify* [21] ne podržava aplikacije izrađene u PHP-u *Vercel* [22] i *Heroku* [20] omogućuju svojim korisnicima da postave na Internet aplikacije napisane u PHP programskom jeziku.



Slika 10 - Izgled postavka aplikacije na Heroku platformi (Izvor: https://dashboard.heroku.com)

Tablica 4 - Podržani razvojni okviri na uspoređenim platformama

Podržani razvojni okviri	Heroku	Vercel	Netlify
Express.js	Da	Da	Da
Ruby on Rails	Da	-	-
Django	Da	-	-
Flask	Da	-	-
Next.js	Da	Da	Da
Nuxt.js	Da	Da	Da
Vue.js	Da	Da	Da
Angular	Da	Da	Da
React	Da	Da	Da
Svelte	Da	Da	Da

Usporedbom podržanih razvojnih okvira u tablici 4 možemo primijetiti da *Heroku* [20] podržava sve navedene, što nije slučaj kod *Vercel-a* [22] i *Netlify*-a [21]. Iako Vercel i *Netlify* [21] imaju podršku za programske jezike *Ruby* i *Python*, njihove razvojne okvire *Ruby on Rails* i *Flask* ne podržavaju.

Tablica 5 - Popis podržanih integracija na platformama

Podržane integracije	Heroku	Vercel	Netlify
GitHub / GitLab / Bitbucket	Da	Da	Da
Amazon S3	Da	Da	Da
Firebase	Da	Da	Da
Cloudinary	Da	Da	Da
MongoDB Atlas	Da	Da	-
PostgreSQL	Da	Da	-
MySQL	Da	Da	-
Auth0	Da	Da	-
Google Cloud Storage	Da	-	-

U tablici broj 5 usporedili smo integracije koje platforme podržavaju. Integracije koje smo odabrali često se pojavljuju prilikom razvoja aplikacija stoga smo ih smatrali relevantnima. *Heroku* [20] s pomoću svojih dodataka (eng. *Add-ons*) omogućuje niz različitih integracija pa su tako sve navedene integracije bile podržane. *Vercel* [22] je imao podršku za sve navedene integracije osim za *Google Cloud Storage* [17] za koju postoje alternative poput *Amazon S3* i slično. Kod *Netlify*-a [21] situacija s uspoređenim integracijama je malo lošija u usporedbi s ostalim platformama, ali to ne znači da u ponudi ne postoje druge opcije. *Netlify* [21] u ponudi ima preko 100 različitih podržanih integracija stoga prilikom razvoja aplikacije može ponuditi alternative za neke opcije koje možda nisu podržane.

3.3.3. Softver kao usluga

Kao zadnji od 3 modela imamo softver kao uslugu što, kako i samo ime govori, daje pristup određenom softveru uz pomoć računarstva u oblaku. Isto kao i kod prethodnih modela, pružatelj usluge upravlja sa svim procesima usluge pa čak i s podacima i aplikacijom. U modelu softver kao usluga korisnik dobiva pristup aplikaciji putem interneta u obliku korisničkog sučelja. Zbog tog razloga ne postoji potreba za lokalnom instalacijom i generalnim održavanjem softvera poput ažuriranja i ispravaka grešaka, već se sve to odrađuje u pozadini kod pružatelja usluge.

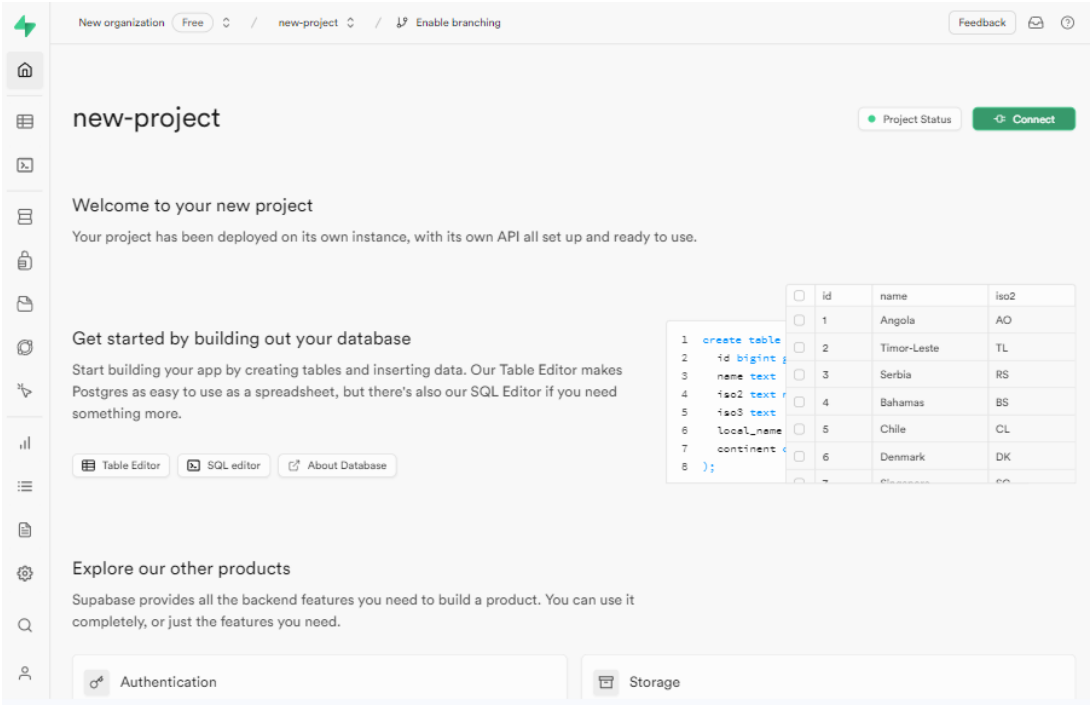
Kod ovog modela računarstva u oblaku često je upotrebljavan model naplate usluga preko pretplate. Pružatelj usluga često ponudi više razina pretplate gdje je često prva razina besplatna, a svaka viša razina korisniku na korištenje daje više mogućnosti koje aplikacija može odraditi.

3.3.3.1. Baza podataka kao usluga

Jedna od potkategorija softvera kao usluge je baza podataka kao usluga. Korisnik prilikom korištenja usluge kao vrijednost ostvaruje sigurnost i privatnost spremljenih podataka, periodičke izrade sigurnosnih kopija, jednostavnu skalabilnost u slučaju razvoja poslovanja te stalnu dostupnost podataka preko interneta. Baza podataka kao usluga ima inicijalno nisku početnu cijenu što odgovara manjim poduzećima, isto tako su svi zadaci vezani uz administraciju baze podataka automatizirani i odrađeni od strane pružatelja usluge što također ide u korist manjim poduzećima [23].

Kao neke od popularnih baza podataka kao usluga možemo nabrojati:

- Supabase
- MongoDB Atlas
- Amazon DynamoDB
- Oracle MySQL Cloud Service
- Google Cloud Firestore



Slika 11 - Prikaz nadzorne ploče na Supabase.com (izvor: https://supabase.com/dashboard/projects)

3.3.3.2. Usporedba pružatelja softvera kao usluge

Za primjer možemo uzeti *Netflix* [24] koji uz pretplatu pruža uslugu video prijenosa gdje svaki plan povećava kvalitetu video sadržaja i broj korisnika koji u isto vrijeme može koristiti aplikaciju.

Također možemo uzeti *Google Workspace* [25] aplikacije među kojima je globalno poznati *Google Disk* [25] koji korisniku besplatno pruža uslugu pohrane podataka na oblak te isto tako uz pretplatu omogućuje korisniku da proširi prostor za pohranu.

Kako bi mogli kvalitetno usporediti pružatelje softvera kao uslugu potrebno je odabrati softvere koji rješavaju približno sličan problem. U našem slučaju odabrali smo aplikacije koje nam omogućuju suradnju s timovima i upravljanje projektima. Aplikacije koje smo odabrali su: *Microsoft Teams* [26], *Slack* [27], *Zoom* [28] i *Asana* [29].

Tablica 6 - Usporedba komunikacijskih značajki

Značajka	Zoom	Slack	Microsoft Teams	Asana
Slanje poruka	Da	Da	Da	Da
Video konferencije	Da	-	Da	-
Audio konferencije	Da	Da	Da	-
Dijeljenje ekrana	Da	Da	Da	-
Dijeljenje datoteka	Da	Da	Da	Da
Komunikacija putem kanala	-	Da	Da	-
Status prisutnosti	Da	Da	Da	-

Usporedbom komunikacijskih značajki u tablici 6 možemo primijetiti kako *Microsoft Teams* [26] podržava sve navedene značajke, dok *Zoom* [28] i *Slack* [27] podržavaju sve osim jedne značajke. U slučaju *Zooma* [28] podrška komunikacije putem kanala nije podržana, a u slučaju *Slacka* [27] nedostaje mogućnost održavanja video konferencija. Kao četvrti kandidat u usporedbi je *Asana* [29] i rezultati usporedbe komunikacijskih značajku znatno su lošiju u usporedbi s drugim kandidatima, ali te rezultate opravdava činjenica da je *Asana* [29] program primarno namijenjen za organizaciju poslovnih zadataka i kontrolu tijeka rada stoga značajke poput video i audio konferencija, dijeljenja ekrana, komunikacije putem kanala i statusa prisutnosti nisu nužno potrebne.

Tablica 7 - Usporedba značajki za upravljanje projektima

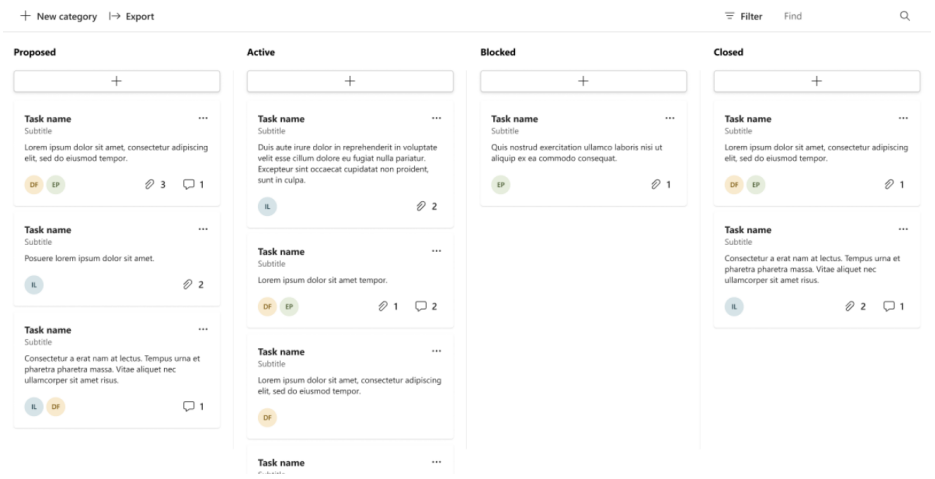
Značajke	Zoom	Slack	Microsoft Teams	Asana
Upravljanje zadacima	-	-	Da	Da
Vremenski okviri projekata	-	-	Da	Da
Kanban ploča	-	-	Da	Da
Integracija kalendara	-	-	Da	Da
Radni prostori i timovi	-	Da	Da	Da
Dodjela zadataka	-	Da	Da	Da
Datumi dospjeća	-	Da	Da	Da

Usporedbom značajki za upravljanje projektima u tablici 7 primjećujemo kako rezultati idu više u korist programima *Microsoft Teams* [26] i *Asana* [29]. Spomenuti programi ispunjavaju sve nabrojene značajke čime omogućuju korisnicima veću razinu kontrole nad projektima. U slučaju *Slacka* [27] značajke poput, izrade radnih prostora i timova, dodjele zadataka te određivanja datuma dospijeća su moguće, dok naprednije mogućnosti upravljanja projektima poput vremenskih okvira projekata, kalendara, kanban ploče nisu prisutne. Aplikacija *Zoom* [28] je prvobitno aplikacija za komunikaciju stoga u svojoj ponudi nema značajke za upravljanje projektima. Ako uzmemo u obzir rezultate iz tablice broj 6 i 7 možemo vidjeti kako *Microsoft Teams* [26] podržava sve spomenute značajke što ga čini i optimalnim izborom za poduzeća.

Modeli naplate				
Zoom	*Basic*	*Pro*	*Business*	*Business Plus / Enerprise*
	Besplatno	**€13.99 / mj / korsinik**	**€20.99 / mj / korisnik**	**Prema upitu**
Slack	*Free*	*Pro*	*Business+*	*Enterprise Grid*
	Besplatno	**€6.69 / mj**	**€11.54 / mj**	**Prema upitu**
Microsoft Teams	*Microsoft Teams (free)*	*Microsoft Teams Essentials*	*Microsoft Teams Essentials with Phone*	-
	Besplatno	**€3.69 / mj / korisnik**	**€7.84 / mj / korisnik**	
Asana	*Personal*	*Starter*	*Advanced*	*Enterprise / Enterprise+*
	Besplatno	**€10.99 / mj / korsinik**	**€24.99 / mj / korsinik**	**Prema upitu**

U tablici 8 možemo vidjeti da sve uspoređene aplikacije u ponudi imaju nekoliko planova gdje je osnovni plan besplatan. Ako želimo omogućiti sve značajke koje aplikacije nude pružatelji usluge omogućili su nam to uz mjesečnu nadoplatu po svakom dodatnom korisniku. Ako planiramo koristiti aplikaciju pružatelja usluga u velikom poduzeću upućeni smo prema prodajnim timovima koje će izraditi ponudu prema našim potrebama. Uspoređujući cijene možemo vidjeti da je *Microsoft Teams* [26] ponovo najpovoljniji.

S podacima dobivenim usporedbama u tablicama 6, 7 i 8 kao najbolji izbor možemo odabrati *Microsoft Teams* [26]. S mnoštvom značajki u komunikacijskom dijelu i dijelu upravljanja projektima te najpovoljnijom cijenom Microsoft Team se ističe od konkurencije kao najisplativiji izbor.



Slika 12 - Kanban ploča u programu Microsoft Teams (izvor: https://learn.microsoft.com/en-us/microsoftteams/platform/concepts/design/design-teams-app-ui-templates)

4. Razvoj aplikacije u oblaku

U ovome poglavlju opisat ćemo praktični dio završnog rada. Prikazat ćemo dijelove koda i opisati njihove zadaće te ćemo prikazati kako oni izgledaju na samoj stranici u pregledniku.

4.1. Opis aplikacije

Aplikacija koja je rezultat ovog završnog rada zamišljena je da radi kao „online“ aukcijska kuća. Korisnik može napraviti svoj račun, postaviti proizvod za koji želi da se ostali korisnici nadmeću te također može se i sam nadmetati za druge proizvode koje su ostali korisnici objavili. Samo nadmetanje se ažurira u stvarnome vremenu kako bi proces nadmetanja bio što realniji. Prilikom isteka vremena za nadmetanje, pobjedniku aukcije pojavljuje se opcija s poveznicom koja ga vodi na stranicu koja prihvaća plaćanja gdje može izvršiti svoju uplatu.

Aplikacija je izrađena u programskom jeziku JavaScript. Programski kod sučelja aplikacije (eng. *Front-end)* pisan je u JavaScriptu bez korištenja razvojnih okvira (eng. *Framework*). Za poslužiteljski dio koda (eng. *Back-end)* koristio se *Node.js* [1] s *Express.js* [2] razvojnim okvirom. Za pohranu podatak koristila se ne-relacijska baza podataka *MongoDB* [5]. Repozitorij sa spremljenim izvornim kodom nalazi se na platformi *GitHub* [30].

4.2. Heroku – odabrani pružatelj platforme kao usluge

Heroku [20] smo odabrali zbog nekoliko razloga:

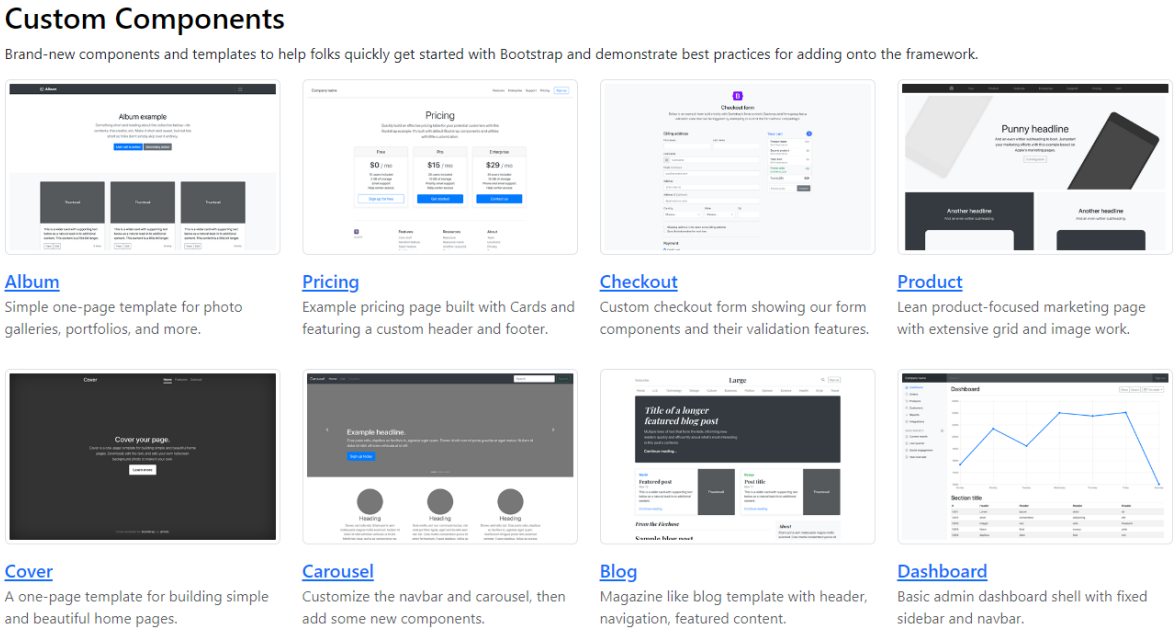
- Nedostatak funkcija za računarstvo bez poslužitelja - S obzirom na to da je poslužiteljska strana koda i kod sučelja aplikacije bio izrađen od strane korisnika te nije bilo potrebe za uslugom računarstva bez poslužitelja
- Integracija s *GitHub-om* [30] – Izvorni kod aplikacije je pohranjen u repozitoriju koji se nalazi na *GitHub-u* [30] koji je izvrsno povezan s *Heroku* [20] platformom što je učinilo implementaciju aplikacije vrlo jednostavnim procesom.
- *GitHub Education* [30] program – program za studente koji pruža razne beneficije studentima među kojima je i vaučer za *Heroku* [20]

4.3. Izrada aplikacije

Kako bi započeli trebali smo prvo odrediti dizajn stranica, arhitekturu aplikacije te način na koji će sama aplikacija primati podatke i obrađivati ih.

4.3.1. Bootstrap biblioteka

Kod izrade dizajna stranice koristili smo se *Bootstrap* [31] bibliotekom. *Bootstrap* [31] je besplatna biblioteka koja korisnicima daje kolekciju gotovih komponenata, stilova i alata koji olakšavaju razvoj modernih internetskih stranica.



Slika 13 - Izbor prilagodljivih komponenata

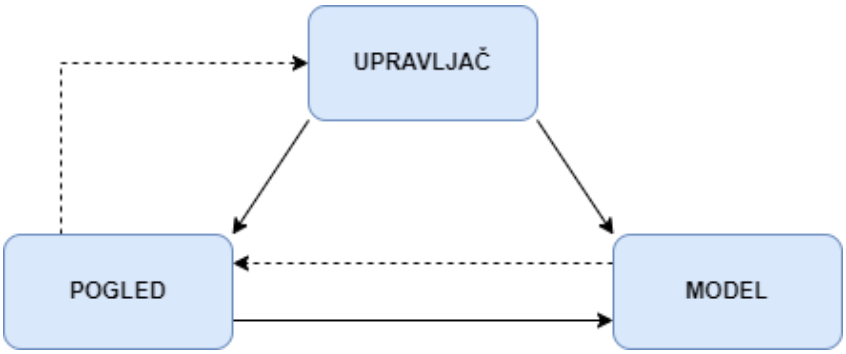
(https://getbootstrap.com/docs/5.3/examples)

4.3.2. MVC arhitektura aplikacije

Za arhitekturu smo koristili MVC arhitekturu (eng. *Model-View-Controller*). MVC obrazac često se koristi za implementaciju logike sučelja, podataka i upravljanja. Naglasak se stavlja na razdvajanje poslovne logike aplikacije i prikaza podataka ([32], [33]).

Tri dijela MVC arhitekture možemo objasniti na sljedeći način

- Model – definira koje podatke bi aplikacija trebala sadržavati te je neovisan o korisničkom sučelju i ponašanjima aplikacije. Kod internetskih aplikacija jedan model obično predstavlja strukturu jedne tablice u bazi podataka [33].
- Pogled (eng. *View)* – predstavlja predložak s predodređenim podacima iz Modela kojima se popunjavaju dijelovi sučelja. Sučelja s kojim korisnik može komunicirati se prikazuju u korisnikovom pregledniku [32].
- Upravljač (eng. *Controller)* – Upravljač služi za interakciju korisnika s aplikacijom. Aplikacijama se često definiraju usmjerivači (eng. *Router*) u kojima se nalaze određene rute. Kada korisnik upiše određenu rutu, usmjerivač poziva funkciju dodijeljenu toj ruti koja je definirana u njenom upravljaču. Upravljač može stvarati, čitati, ažurirati i brisati podatke iz baze podataka. Svaka funkcija upravljača u većini slučajeva završava s prosljeđivanjem obrađenih podataka pogledu [32].



Slika 14 - MVC arhitektura
(https://hr.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller)

4.3.3. Aplikacijsko programsko sučelje

Za našu aplikaciju odlučili smo da će sve interakcije s bazom podataka ići preko aplikacijskog programskog sučelja (eng. *application programming interface,* skr. *API*). Kako bi omogućili da aplikacija može poslati zahtjeve poslužitelju s pomoću HTTP metoda potrebno je bilo izraditi aplikacijsko programsko sučelje i definirati rute na kojima će server izvršavati određene zadatke.

Rute smo definirali na sljedeći način:

```
const app = express();

app.use('/', viewRouter);
app.use('/api/v1/users', userRouter);
app.use('/api/v1/products', productRouter);
app.use('/api/v1/bids', bidRouter);
app.use('/api/v1/checkouts', checkoutRouter);
```

Definirali smo varijablu „app“ kao početnu točku *Express.js* [2] razvojnog okvira. Na „app“ varijabli smo potom definirali zadane rute na koje će klijent prosljeđivati zahtjeve koji će biti dalje proslijeđeni do tražene funkcije od strane definiranih usmjerivača (eng. *router*).

4.3.4. Sheme baze podataka

Za našu aplikaciju koristili smo poznatu ne-relacijsku bazu podataka zvanu *MongoDB* [5]. Ne-relacijske baze podataka sastoje se od kolekcija i dokumenata. Kolekcije su skupina povezanih dokumenata koji dijele strukturu. Sličnost dijele s tablicama u relacijskim bazama podataka, dok dokumenti dijele sličnost s redovima u relacijskim bazama podataka. Naša aplikacija se sastoji od 3 kolekcije: kolekcija za korisnike, proizvode i kolekcija koja zapisuje uspješne transakcije.

Kolekcije koriste sheme kao obrazac prilikom stvaranja novog dokumenta u kolekciji kako bi svi uneseni dokumenti bili istog formata. Također se kod definiranja sheme upisuju validacije koje osiguravaju točnost unesenih podataka.

Kolekcija za korisnike ima sljedeću shemu:

```
const userSchema = new mongoose.Schema({
  name: {
    type: String,
    required: [true, 'User must have a name'],
  },
  email: {
    type: String,
    required: [true, 'User must have and email'],
    trim: true,
    lowercase: true,
    unique: true,
```

```
    validate: [validator.isEmail, 'Please enter valid email'],
  },
  password: {
    type: String,
    required: [true, 'Please enter the password'],
    minlength: 8,
    select: false,
  },
  role: {
    type: String,
    enum: ['user', 'moderator', 'admin'],
    default: 'user',
  },
  passwordConfirm: {
    type: String,
    required: [true, 'Passwords dont match!'],
    validate: {
      validator: function (el) {
        return el === this.password;
      },
      message: 'Passwords are not the same!',
    },
  },
  photo: { type: String, default: 'default.png' },
  passwordChangedAt: {
    type: Date,
  },
  passwordResetToken: String,
  passwordResetExpires: Date,
  active: {
    type: Boolean,
    default: true,
    select: false,
  },
});
```

Kolekcija za proizvode na aukciji ima sljedeću shemu:

```
const productSchema = new mongoose.Schema({
  name: {
    type: String,
    required: [true, 'Product must have a name'],
  },
  description: {
    type: String,
  },
  startingBid: {
    type: Number,
    required: [true, 'Product must have a starting bid'],
  },
  currentBid: {
    type: Number,
    default: function () {
      return this.startingBid;
    },
  },
  seller: {
    type: mongoose.Schema.ObjectId,
    ref: 'User',
    required: [true, 'Product must have a seller'],
  },
  bids: [
    {
      bidder: String,
      amount: Number,
    },
  ],
  endDate: {
    type: Date,
    default: Date.now() + 1000 * 60 * 60 * 24 * 3,
  },
  coverImage: { type: String, default: 'default-no-img.png' },
  photos: { type: [String] },
  emailSent: { type: Boolean, default: false },
});
```

Kolekcija za završetak kupovine ima sljedeću shemu:

```
const checkoutSchema = new mongoose.Schema({
  product: {
    type: mongoose.Schema.ObjectId,
    ref: 'Product',
    required: [true, 'Checkout must belong to a Product!'],
  },
  user: {
    type: mongoose.Schema.ObjectId,
    ref: 'User',
    required: [true, 'Checkout must belong to a User!'],
  },
  price: {
    type: Number,
    require: [true, 'Checkout must have a price for a product.'],
  },
  createdAt: {
    type: Date,
    default: Date.now(),
  },
  paid: {
    type: Boolean,
    default: true,
  },
});
```

4.3.5. Funkcionalnost nadmetanja u stvarnom vremenu

Jedan od posebnosti aplikacije je ta da omogućuje korisniku da se nadmeće za odabrani proizvod u stvarnome vremenu. Spomenutu funkcionalnost ostvarujemo s pomoću *WebSocket* protokola. *WebSocket* protokol je protokol koji podržava dvosmjerni prijenos podataka u isto vrijeme što omogućuje bržu komunikaciju [34].

Kako bi olakšali upotrebu *WebSocket* protokola, koristili smo se bibliotekom *ws* [35] koja nam omogućuje lakšu obradu podataka kod definiranih događaja koji se pojavljuju prilikom korištenja *WebSocket* protokola.

U našem programskom kodu implementacija *WebSocket* protokola odrađena je na sljedeći način:

Kako bi počeli koristiti protokol, potrebno je pokrenuti *websocket* poslužitelja

```
const wss = new WebSocketServer({ server });
```

Potom je potrebno definirati događaje prilikom kojih će se izvršiti određeni dijelovi programskog koda.

Programski kod poslužitelja:

```
wss.on('connection', websocketController.connectionHandler);
wss.on('close', websocketController.closeConnectionHandler);
wss.on('error', websocketController.websocketErrorHandler);
```

Programski kod upravljača:

```
exports.connectionHandler = (ws) => {
  if (!serverState) return;
  if (!serverState[product._id]) return;
  if (serverState[product._id])
    serverState[product._id].clients.add(userData._id);
  activeConnections++;
  ws.send(
    JSON.stringify({
      type: 'initialBids',
      _activeBids: serverState[product._id]._activeBids,
    }),
  );
  if (remainingTime < 0) {
    ws.send(
      JSON.stringify({
        type: 'over',
        message: 'Auction has ended.',
      }),
    );
    ws.close(1000, 'Auction over');
    if (!serverState[product._id].emailSent && product.bids.length >
0) {
      const url = `${protocol}://${host}/products/${product._id}`;
      sendEmailToUser(
        serverState[product._id]._activeBids.at(-1).bidder,
        url,
        'won',
      );
    }
```

```
        updateProductBidsInDB();
      }
      ws.isAlive = true;

      ws.on('message', (data) => {
        const newBid = JSON.parse(data);
        // console.log(newBid);
        if (!serverState[product._id]) return;
        if (remainingTime < 0) {
          ws.send(
            JSON.stringify({
              type: 'over',
              message: 'Auction has ended.',
            }),
          );
          return;
        }
        if (
          product.bids.length > 0 &&
          newBid.amount    <=    serverState[product._id]._activeBids.at(-
1).amount
        ) {
          ws.send(
            JSON.stringify({
              type: 'error',
              message: 'Bid must be a larger amount than the current bid.',
            }),
          );
          return;
        }
        newBid.amount *= 1;
        if (!newBid.amount) {
          ws.send(
            JSON.stringify({
              type: 'error',
              message: 'Bid must be a number!',
            }),
          );
          return;
        }
        serverState[newBid._id]._activeBids.push(newBid);
```

```
    Server.sendNewBids(newBid);
    serverState[product._id]._newBids.push(newBid);
    updateProductOnMessage();
  });
};
```

Kada smo pokrenuli poslužitelja i definirali događaje, aplikacija prilikom registracije određenog događaja obrađuje dobivene podatke te ih šalje klijentu kako bi mogli biti prikazani u pregledniku.

```
userEmail = document.cookie
    .split(';')
    .filter((el) => el.includes('user'))[0]
    .trim()
    .split('=')[1]
    .replace('%40', '@');
  const uri = `wss://${window.location.host.split(':')[0]}:443`;
  const ws = new WebSocket(uri);
  const wsBidding = (formValue) => {
    const id = document.getElementById('product').dataset.id;
    if (!+formValue) {
      showAlert('error', 'Please enter the valid number');
      return;
    }
    const messageData = {
      _id: id,
      amount: formValue,
      bidder: userEmail,
    };
    const message = JSON.stringify(messageData);
    if (userEmail === messageData.bidder) {
      ws.send(message);
    }
  };
  ws.onopen = function open() {
    console.log('connected');
  };
  ws.onclose = function close() {
    ws.close();
    console.log('disconnected');
  };
  ws.onmessage = (event) => {
```

```
          const message = JSON.parse(event.data);
          const biddingState = message._activeBids;
          const newBid = message.bid;

          updateBiddingUI(biddingState, newBid, message,
liveBiddingElement);
        };
```

4.3.6. Primjer obrade zahtjeva u aplikaciji

Kao primjer ćemo prikazati situaciju u kojoj klijent želi prikupiti podatke o određenom proizvodu uz pomoć aplikacijskog programskog sučelja (zbog lakše čitljivosti primjer će se odvijati na lokalnom uređaju):

Klijent u svom pregledniku upisuje rutu: /api/v1/products/12341234 čime na poslužitelja šalje „GET“ zahtjev koji je jedan od HTTP metoda. Poslužitelj zaprima zahtjev od klijenta te poziva usmjerivač definiran na ruti koja se poklapa sa zahtjevom.

```
app.use('/api/v1/products', productRouter);
```

Zbog MVC arhitekture programski kod usmjerivača nalazi se u posebnom modulu koji je izvršen prilikom upisivanja određene rute. U modulu „productRoutes.js“ definiran je „productRouter“ kojega je poslužitelj pozvao. Kako smo u našoj ruti imali definiran i identifikacijski broj samog proizvoda, u usmjerivačkom modulu izvršena je metoda čiji se parametri poklapaju s upisanom rutom.

```
const router = express.Router();
router.route('/:id').get(productController.getProduct);
module.exports = router;
```

U našem slučaju će to biti dio programskog koda koji kao parametre prima „ /:id “ što je način na koji „Express“ razvojni okvir definira varijable unutar ruta i kao drugi parametar zahtjeva metodu „GET“. Kada su prisutna oba parametra, linija koda poziva modul upravljača koji sadrži programski kod koji dohvaća tražene podatke o proizvodu.

```
exports.getOneProduct = async (req, res, next) => {
  try {
    let query = Product.findById(req.params.id);
    const doc = await query;
    if (!doc) {
      return next(new AppError('No documents found with that ID',
404));
```

```
    }
    res.status(200).json({
      staus: 'success',
      requestedAt: req.reqTime,
      data: {
        data: doc,
      },
    });
  } catch (err) {
    console.log(err);
  }
};
```

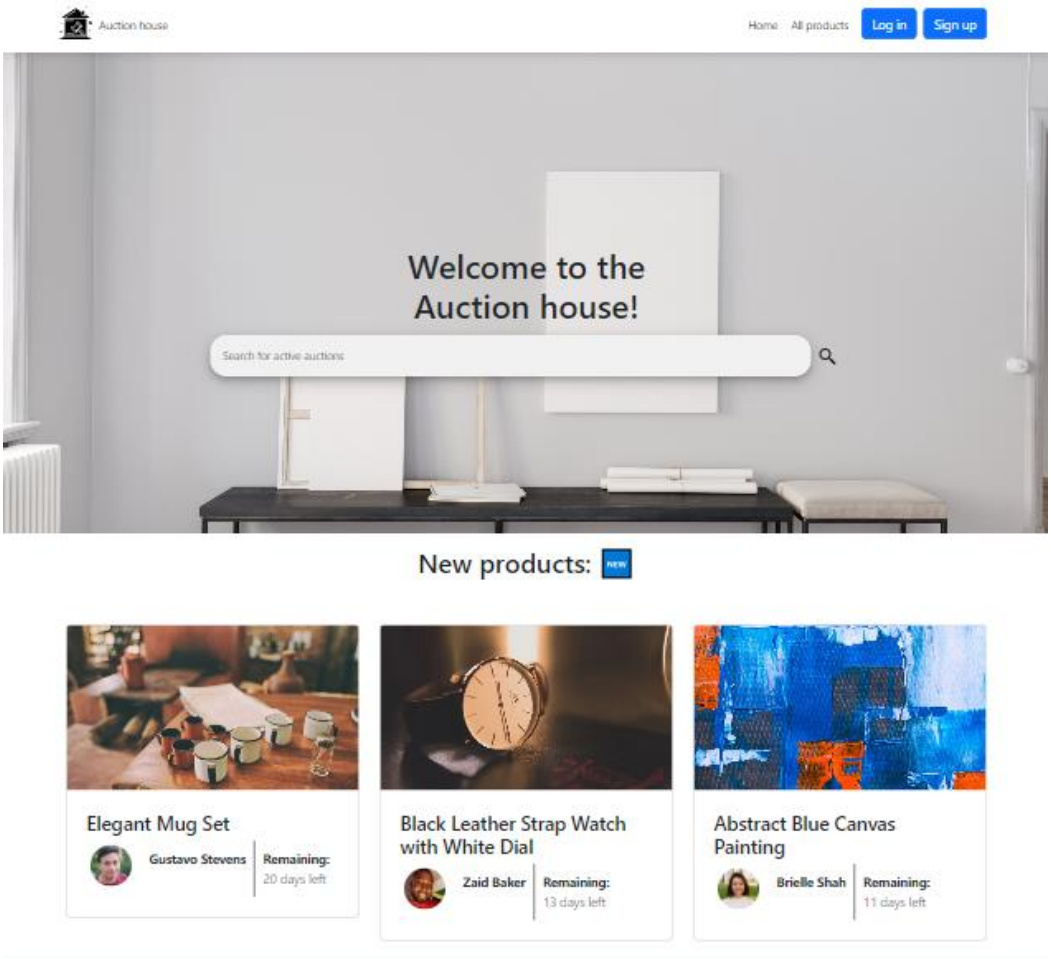
U slučaju da u bazi podataka ne postoji upisani identifikacijski broj i baza podataka ne ispiše rezultat, postavljen je uvjet kojim obavještavamo korisnika o krivom unosu. Ako baza podataka proslijedi rezultat, isti će biti poslan klijentu u formatu standardiziranom za slanje podataka putem HTTP protokola zvanim JSON.

4.4. Prikaz aplikacije u pregledniku

Prenošenjem izrađene aplikacije na oblak odabranog pružatelja usluge dobivamo poveznicu na našu aplikaciju. U našem slučaju poveznica je sljedeća:

https://auction-house-app-e7698736e88b.herokuapp.com/

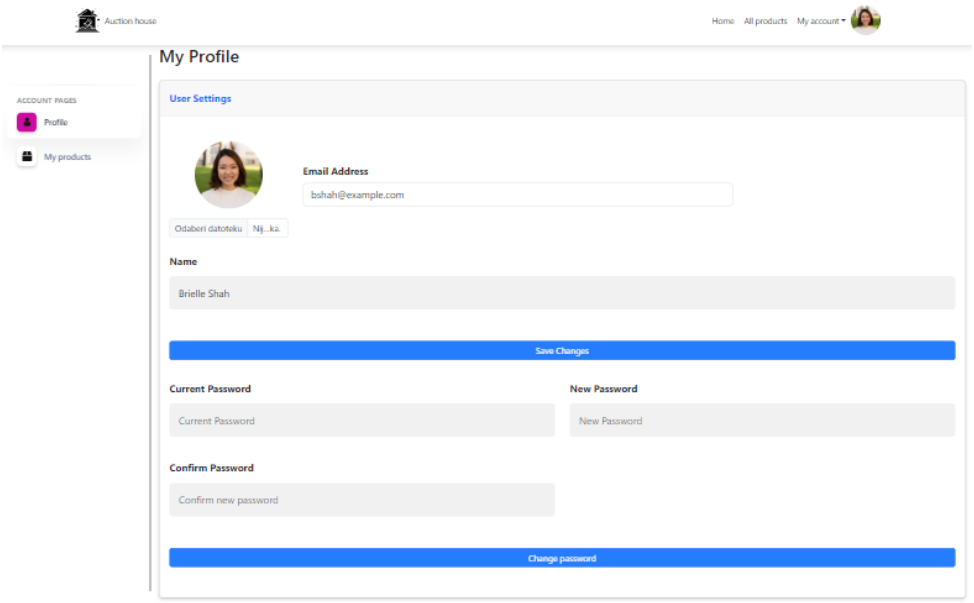
Posjećivanjem navedene poveznice susrećemo se s početnom stranicom aplikacije (Slika 5). Na početnoj stranici možemo vidjeti trenutne aukcije, njihove vlasnike te koliko još vremena one traju.



Slika 15 - Početna strana aplikacije

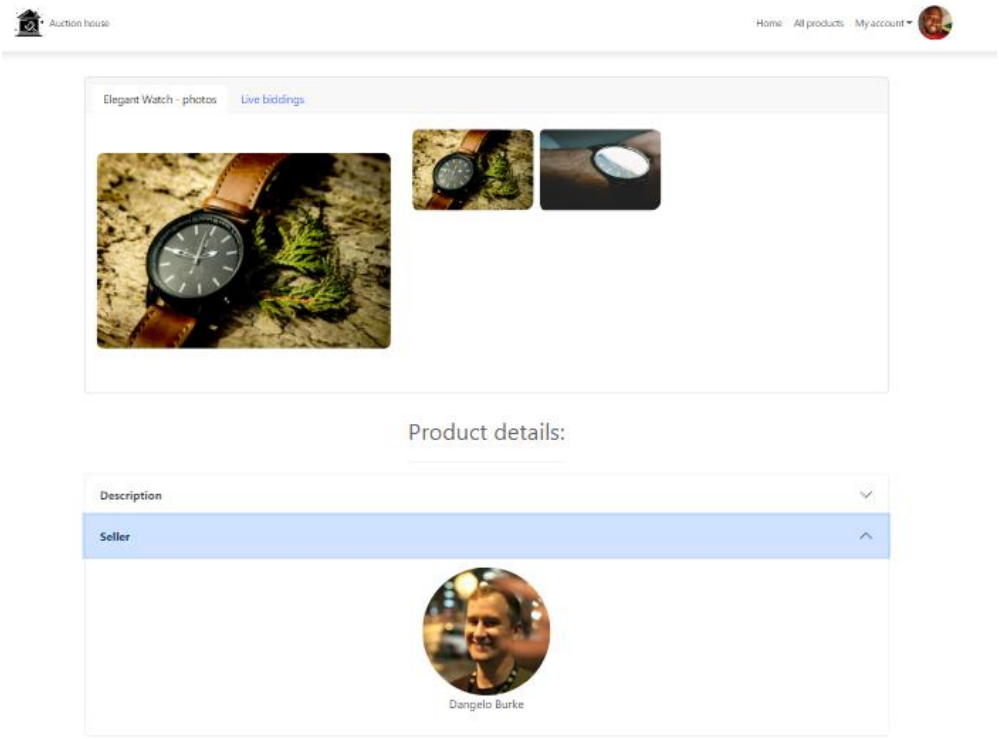
Ukoliko želimo pregledavati ili sudjelovati u aukcijama, potrebno se prvo registrirati ispunjavanjem forme.

Nakon uspješne registracije korisniku je poslana elektronička pošta dobrodošlice i preusmjeren je na stranicu svog profila gdje može promijeniti ime, sliku, adresu e-pošte i svoju lozinku (Slika 6). Također su mu omogućene prethodno ograničene opcije što znači da sada može pregledavati aktivne aukcije i sudjelovati u njima.

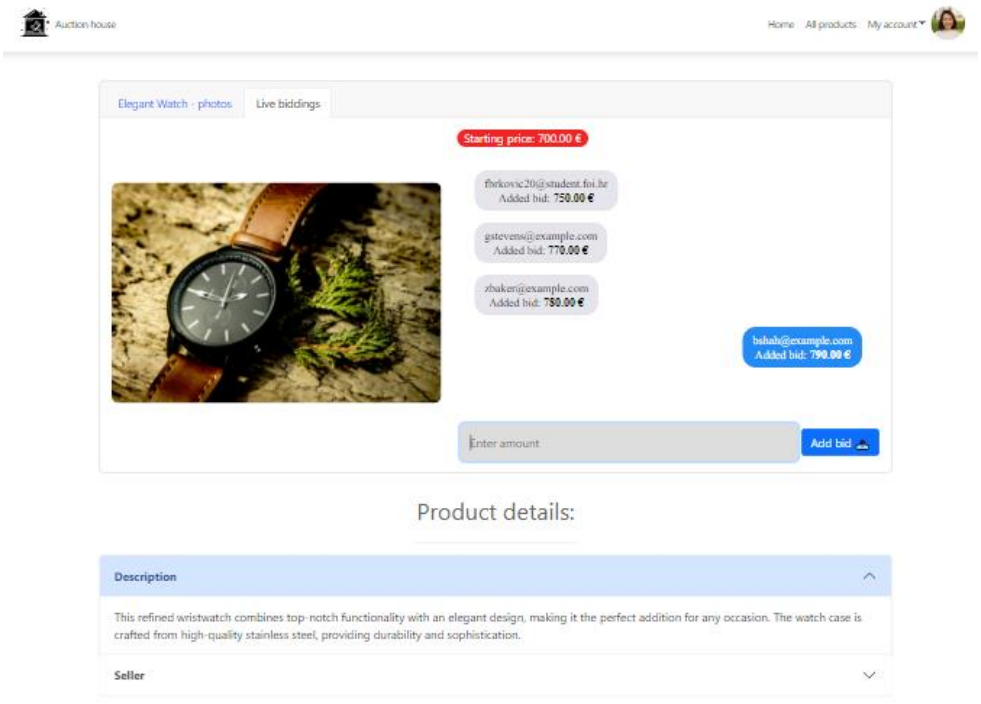


Slika 16 - Izgled stranice profila

Otvaranjem jedne od aukcija korisnika dočekuje jedna ili više slika proizvoda Slika(8), te ako se želi nadmetati klikom na karticu „*Live biddings*“ prikazuje mu se sučelje za nadmetanje (Slika 9).

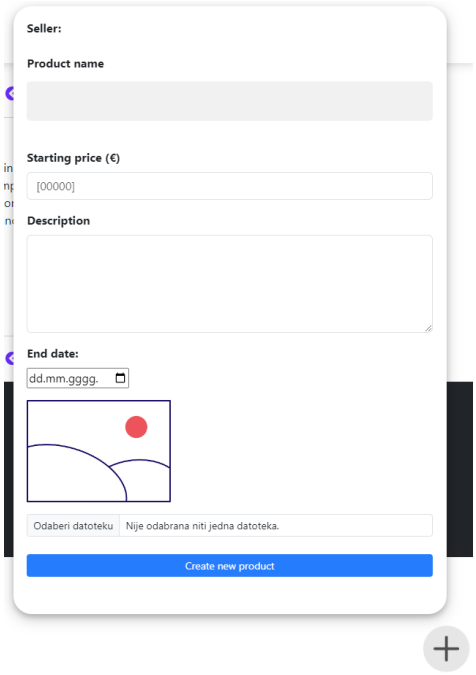


Slika 17 - Stranica proizvoda na aukciji



Slika 18 - Sučelje za nadmetanje

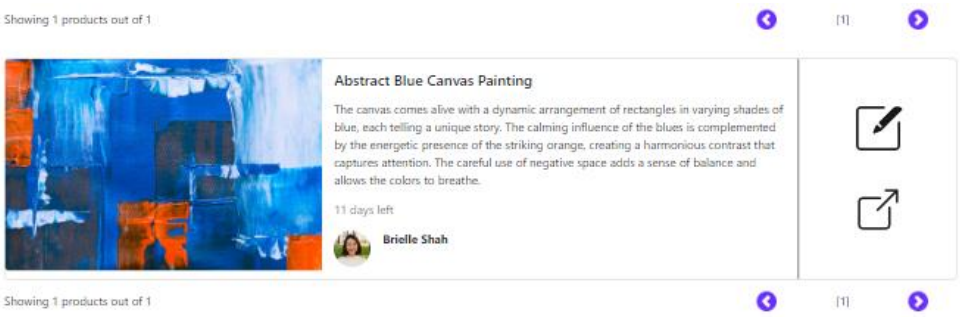
Ako korisnik želi izraditi svoju aukciju to može učiniti navigacijom na stranicu sa svojim proizvodima koja se nalazi u izborniku „My account“ pod opcijom „My products“. Na stranici „My products“ klikom na plavo dugme u donjem desnom dijelu ekrana pojavljuje se obrazac za izradu aukcije za proizvod (Slika 9).



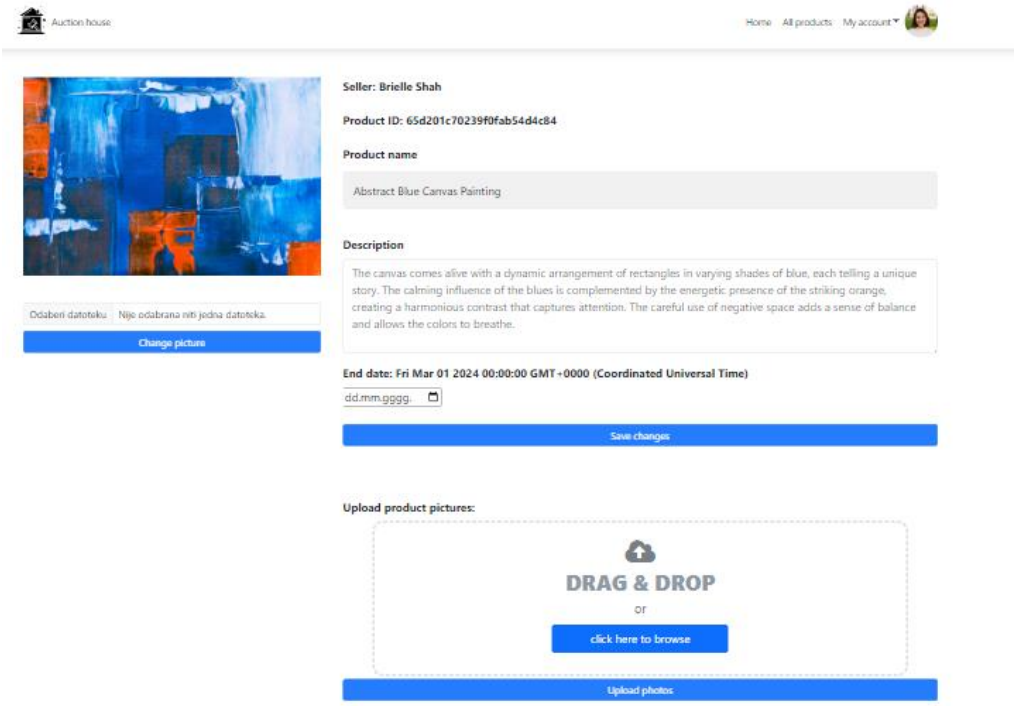
Slika 19 - Obrazac za izradu aukcije

Nakon definiranja imena, početne cijene, opisa artikla, vremena trajanja aukcije i ako želimo slike artikla, izrađuje se nova aukcija u bazi podataka i omogućuje se svim registriranim korisnicima da sudjeluju u njoj.

Ako korisnik želi raditi izmjene na proizvodu, na stranici „My products“ klikom na ikonu koja predlaže uređivanje proizvoda (Slika 10), preusmjereni smo na stranicu koja nam daje iste mogućnosti (Slika 11). Ako želimo možemo također dodati još slika proizvoda koje će biti vidljive na aktivnoj aukciji.



Slika 20 - Prikaz aukcije na stranici s korisnikovim aukcijama



Slika 21 - Uredi proizvod

5. Zaključak

Uz zapanjujuće brz razvoj novih tehnologija potreba za računarstvom u oblaku postaje sve veća što rezultira većom ponudom pružatelja usluga. Pristup aplikacijama putem aktivacijskog ključa mijenja se uslugom dostupnom preko interneta u bilo koje vrijeme s bilo kojeg uređaja.

Prilikom razvoja aplikacije susretali smo se s raznim preprekama od dizajna same stranice do logike koju koristi WebSocket protokol. Za svaku funkcionalnost postojao je barem jedan razvojni okvir koji je proces pisanja programskog koda znatno olakšao te isto tako i za svaki problem u programskom kodu postojalo je rješenje ili smjernica prema uzroku problema. Postavljanje aplikacije na poslužitelja pružatelja usluge također je vrlo jasno i detaljno dokumentiran proces koji je učinio dijeljenje naše aplikacije iznimno jednostavnim. Zbog brojnih apstrakcija u pisanju programskog koda razvoj aplikacija postaje brži.

Dok je proces razvoja aplikacije iznimno olakšan zbog zajedničkih napora zajednice programera koji nesebično dijele svoje izvorne programske kodove, to ne znači da u njega ne treba uložiti truda. Razvoj kvalitetne aplikacije zahtjeva dosta vremena i znanja kako bi ta aplikacija bila ugodna za korištenje i naravno sigurna.

Popis literature

[1] „Node.js“. Pristupljeno: 20. veljača 2024. [Na internetu]. Dostupno na: https://nodejs.org/en

[2] „Express - Node.js web application framework“. Pristupljeno: 20. veljača 2024. [Na internetu]. Dostupno na: https://expressjs.com/

[3] „Mongoose ODM v8.1.3“. Pristupljeno: 20. veljača 2024. [Na internetu]. Dostupno na: https://mongoosejs.com/

[4] „Visual Studio Code - Code Editing. Redefined“. Pristupljeno: 20. veljača 2024. [Na internetu]. Dostupno na: https://code.visualstudio.com/

[5] „MongoDB: The Developer Data Platform | MongoDB“. Pristupljeno: 20. veljača 2024. [Na internetu]. Dostupno na: https://www.mongodb.com/

[6] P. M. Mell i T. Grance, „The NIST definition of cloud computing“, Gaithersburg, MD, 2011. doi: 10.6028/NIST.SP.800-145.

[7] J. Surbiryala i C. Rong, „Cloud computing: History and overview“, u *Proceedings - 2019 3rd IEEE International Conference on Cloud and Fog Computing Technologies and Applications, Cloud Summit 2019*, Institute of Electrical and Electronics Engineers Inc., kol. 2019, str. 1–7. doi: 10.1109/CloudSummit47114.2019.00007.

[8] „What is SaaS (Software as a Service)? Everything You Need to Know“. Pristupljeno: 21. kolovoz 2023. [Na internetu]. Dostupno na: https://www.techtarget.com/searchcloudcomputing/definition/Software-as-a-Service

[9] C. Mustafa Mohammed i S. R. M Zeebaree, „Sufficient Comparison Among Cloud Computing Services: IaaS, PaaS, and SaaS: A Review“, 2021, doi: 10.5281/zenodo.4450129.

[10] E. Gorelik, „L? CRA RIES“, 2013.

[11] „Pros and Cons of Cloud Storage | Secure Storage“. Pristupljeno: 22. kolovoz 2023. [Na internetu]. Dostupno na: https://www.securestorageservices.co.uk/article/11/pros-and-cons-of-cloud-storage

[12] S. El Kafhali, I. El Mir, i M. Hanini, „Security Threats, Defense Mechanisms, Challenges, and Future Directions in Cloud Computing“, *Archives of Computational Methods in Engineering*, sv. 29, str. 223–246, sij. 2022, doi: 10.1007/s11831-021-09573-y.

[13] N. L. CERT suradnji, „NCERT-PUBDOC-2010-03-293“. [Na internetu]. Dostupno na: www.LSS.hr

[14] „IaaS vs. PaaS vs. SaaS“. Pristupljeno: 12. kolovoz 2023. [Na internetu]. Dostupno na: https://www.redhat.com/en/topics/cloud-computing/iaas-vs-paas-vs-saas?sc_cid=7013a000002pgRQAAY&gclid=CjwKCAjw29ymBhAKEiwAHJbJ8v0ODKanCRO1pQJIXO8viISl2I2D7xVSlWCoI9gdAzy2x8kf6QExHBoCDl4QAvD_BwE&gclsrc=aw.ds

[15] „Cloud Computing Services - Amazon Web Services (AWS)“. Pristupljeno: 24. veljača 2024. [Na internetu]. Dostupno na: https://aws.amazon.com

[16] „Cloud Computing Services | Microsoft Azure“. Pristupljeno: 24. veljača 2024. [Na internetu]. Dostupno na: https://azure.microsoft.com

[17] „Cloud Computing Services | Google Cloud“. Pristupljeno: 24. veljača 2024. [Na internetu]. Dostupno na: https://cloud.google.com

[18] „What Is PaaS - Advantages and Disadvantages | Cloud Computing | CompTIA“. Pristupljeno: 21. kolovoz 2023. [Na internetu]. Dostupno na: https://www.comptia.org/content/articles/what-is-paas

[19] „What is serverless computing? | Serverless definition | Cloudflare“. Pristupljeno: 03. veljača 2024. [Na internetu]. Dostupno na: https://www.cloudflare.com/learning/serverless/what-is-serverless/
[20] „Cloud Application Platform | Heroku“. Pristupljeno: 20. veljača 2024. [Na internetu]. Dostupno na: https://www.heroku.com/home
[21] „Scale & Ship Faster with a Composable Web Architecture | Netlify“. Pristupljeno: 25. veljača 2024. [Na internetu]. Dostupno na: https://www.netlify.com/
[22] „Vercel: Build and deploy the best Web experiences with The Frontend Cloud – Vercel“. Pristupljeno: 25. veljača 2024. [Na internetu]. Dostupno na: https://vercel.com/
[23] „Best Database as a Service (DBaaS) Provider in 2024“. Pristupljeno: 25. veljača 2024. [Na internetu]. Dostupno na: https://www.g2.com/categories/database-as-a-service-dbaas
[24] „Netflix“. Pristupljeno: 20. veljača 2024. [Na internetu]. Dostupno na: https://www.netflix.com
[25] „Google Workspace: Secure Online Productivity & Collaboration Tools“. Pristupljeno: 20. veljača 2024. [Na internetu]. Dostupno na: https://workspace.google.com/
[26] „Microsoft – oblak, računala i aplikacije“. Pristupljeno: 26. veljača 2024. [Na internetu]. Dostupno na: https://www.microsoft.com/hr-hr/
[27] „Slack is your productivity platform | Slack“. Pristupljeno: 26. veljača 2024. [Na internetu]. Dostupno na: https://slack.com/
[28] „One platform to connect | Zoom“. Pristupljeno: 26. veljača 2024. [Na internetu]. Dostupno na: https://zoom.us/
[29] „Manage your team’s work, projects, & tasks online • Asana • Asana“. Pristupljeno: 26. veljača 2024. [Na internetu]. Dostupno na: https://asana.com/
[30] „GitHub“. Pristupljeno: 20. veljača 2024. [Na internetu]. Dostupno na: https://github.com/
[31] „Bootstrap · The most popular HTML, CSS, and JS library in the world.“ Pristupljeno: 20. veljača 2024. [Na internetu]. Dostupno na: https://getbootstrap.com/
[32] „Model–view–controller – Wikipedija“. Pristupljeno: 05. veljača 2024. [Na internetu]. Dostupno na: https://hr.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
[33] „Models | Django documentation | Django“. Pristupljeno: 05. veljača 2024. [Na internetu]. Dostupno na: https://docs.djangoproject.com/en/5.0/topics/db/models/
[34] R. Tangen, „Real-Time Web with WebSocket Master’s thesis“, 2015.
[35] „ws - npm“. Pristupljeno: 20. veljača 2024. [Na internetu]. Dostupno na: https://www.npmjs.com/package/ws
[36] „Prettier · Opinionated Code Formatter“. Pristupljeno: 27. veljača 2024. [Na internetu]. Dostupno na: https://prettier.io/
[37] „Find and fix problems in your JavaScript code - ESLint - Pluggable JavaScript Linter“. Pristupljeno: 27. veljača 2024. [Na internetu]. Dostupno na: https://eslint.org/
[38] „VMware - Delivering a Digital Foundation For Businesses“. Pristupljeno: 27. veljača 2024. [Na internetu]. Dostupno na: https://www.vmware.com/
[39] „Poslovni programi za mala i srednja poduzeća | SAP“. Pristupljeno: 27. veljača 2024. [Na internetu]. Dostupno na: https://www.sap.com/croatia/index.html?url_id=auto_hp_redirect_croatia
[40] „Open Source Cloud Computing Infrastructure - OpenStack“. Pristupljeno: 27. veljača 2024. [Na internetu]. Dostupno na: https://www.openstack.org/
[41] „IBM Cloud“. Pristupljeno: 27. veljača 2024. [Na internetu]. Dostupno na: https://www.ibm.com/cloud

Popis slika

Popis tablica