



UNIVERZITET U NOVOM
SADU
FAKULTET TEHNIČKIH
NAUKA U NOVOM SADU



Vera Kovačević

Izgradnja skalabilne aplikacije u oblaku korišćenjem AWS servisa

Diplomski rad
- Osnovne akademske studije -

Novi Sad, 2021.

	UNIVERZITET U NOVOM SADU FAKULTET TEHNIČKIH NAUKA 21000 NOVI SAD, Trg Dositeja Obradovića 6	Datum:
	ZADATAK ZA IZRADU DIPLOMSKOG (BACHELOR) RADA	List: 1/1

(Podatke unosi predmetni nastavnik - mentor)

Vrsta studija:	<input type="checkbox"/> Osnovne akademske studije
Studijski program:	Softversko inženjerstvo i informacione tehnologije
Rukovodilac studijskog programa:	prof. dr Miroslav Zarić

Student:	Vera Kovačević	Broj	SW 19/2017
Oblast:	Web programiranje		
Mentor:	doc. dr Siniša Nikolić		
NA OSNOVU PODNETE PRIJAVE, PRILOŽENE DOKUMENTACIJE I ODREDBI STATUTA FAKULTETA IZDAJE SE ZADATAK ZA DIPLOMSKI RAD, SA SLEDECIM ELEMENTIMA: <ul style="list-style-type: none"> - problem – tema rada; - način rešavanja problema i način praktične provere rezultata rada, ako je takva provera neophodna; - literatura 			

NASLOV DIPLOMSKOG (BACHELOR) RADA:

Izgradnja skalabilne aplikacije u oblaku korišćenjem AWS servisa

TEKST ZADATKA:

Istražiti i opisati osnovne AWS servise. Definisati AWS arhitekturu koja omogućava elastičnost i visoku dostupnost. Kreirati demo aplikaciju koja će se postaviti na predloženu arhitekturu. Testirati rješenje. Dokumentovati rješenje.

Rukovodilac studijskog programa:	Mentor rada:

Primerak za: - Studenta; - Mentora

SADRŽAJ

1.	UVOD	1
2.	STANJE U OBLASTI	3
2.1.	Microsoft Azure	3
2.2.	Google Cloud Platform	4
3.	KORIŠĆENE TEHNIKE I TEHNOLOGIJE.....	5
3.1.	AWS.....	5
3.2.	VPC	9
3.3.	EC2.....	16
3.4.	S3, Glacier.....	17
3.5.	RDS, Aurora.....	19
3.6.	CloudWatch.....	22
3.7.	Auto Scaling, ELB	23
3.8.	Elastic Beanstalk	28
3.9.	Spring	28
3.10.	Angular.....	30
4.	SPECIFIKACIJA SISTEMA	33
4.1.	Dijagram klasa	33
4.2.	Dijagram slučajeva korišćenja	34
4.3.	Arhitektura sistema	36
5.	IMPLEMENTACIJA SISTEMA	41
5.1.	Klijentska strana.....	41
5.2.	Serverska strana	42
5.3.	AWS.....	48
6.	PRIKAZ IMPLEMENTIRANOG SISTEMA	59
6.1.	Prikaz funkcionalnosti sistema.....	59
6.2.	Prikaz elastičnosti i dostupnosti sistema	64
7.	ZAKLJUČAK	71
	LITERATURA.....	73
	BIOGRAFIJA	75
	KLJUČNA DOKUMENTACIJSKA INFORMACIJA	77
	KEY WORDS DOCUMENTATION	79

1. UVOD

Amazon Web Services (AWS) je platforma veb servisa koja pruža rješenja za skladištenje, umrežavanje i računarstvo na *cloud-u* (engl. *cloud computing*). Predstavlja jedan od dobavljača servisa na *cloud-u* (engl. *cloud service providers*) [1]. U tehničkom smislu, *cloud* je skup velikog broja povezanih hardverskih resursa koje korisnik vidi kao jedan veliki virtuelni računar. Korisnik može da alocira dijelove *cloud-a* za svoje aplikacije. Veb servisi koje nudi AWS su dostupni putem interneta upotrebom tipičnih veb protokola, kao što je HTTP (*Hypertext Transfer Protocol*) [2]. Njihovim kombinovanjem mogu da se izgrade arhitekture pogodne za hostovanje veb sajtova, pokretanje poslovnih aplikacija i skladištenje ogromnih količina podataka [1,3].

Prednost koju nudi *cloud* jeste mogućnost izgradnje efikasne, pouzdane, skalabilne, elastične i visoko dostupne arhitekture za pojedinačne korisničke aplikacije. *AWS Well-Architected Framework* opisuje ključne koncepte i principe za dizajniranje upravo takve arhitekture na AWS. Oslanja se na pet „stubova“:

- ***Operational Excellence*** – unapredivanje i automatizacija sistema na osnovu praćenja njegovog rada;
- ***Security*** – bezbjednost informacija u sistemu;
- ***Reliability*** – pouzdanost sistema; sistem konzistentno izvršava sve što se od njega očekuje i brzo se oporavlja od otkaza;
- ***Performance Efficiency*** – efikasnost sistema; izbor odgovarajućih resursa u zavisnosti od zahtjeva;
- ***Cost Optimization*** – izbjegavanje nepotrebnih troškova [4].

Ovaj rad je fokusiran na pouzdanost i efikasnost sistema. Da bi se to postiglo, potrebno je voditi računa o konceptima *AWS Well-Architected Framework*-a kao što su elastičnost (engl. *elasticity*) i visoka dostupnost (engl. *high availability*).

Prema tome, tema ovog rada je razvoj elastične i visoko dostupne aplikacije korišćenjem AWS servisa. Elastičnost podrazimjeva sposobnost sistema da automatski dobavlja resurse kada su mu potrebni i pušta ih kada mu više nisu potrebni. Visoka dostupnost podrazumijeva sposobnost sistema da izvrši traženu funkcionalnost u datom trenutku [5].

Cilj rada je prikaz osnovnih AWS servisa i prijedloga rješenja za izgradnju elastične i visoko dostupne arhitekture.

U nastavku rada ukratko će biti opisane druge *cloud* platforme i njihovo poređenje sa AWS, zatim će biti dat opis osnovnih AWS servisa sa

naglaskom na one koji su korišćeni u radu, te će biti prikazan primjer elastične i visoko dostupne arhitekture na koju je postavljena jednostavna aplikacija. Budući da se mnogi servisi naplaćuju, prikazana arhitektura uglavnom će biti ograničena na besplatne servise, a na kraju rada će biti dati prijedlozi za njeno unapređenje korišćenjem pogodnijih servisa koji ne spadaju u *Free Tier* [\[6\]](#).

2. STANJE U OBLASTI

Računarstvo na *cloud*-u predstavlja isporuku računarskih resursa i skladišnih kapaciteta kao uslugu korisniku. Kompanije koje nude takve usluge nazivaju se dobavljači servisa na *cloud*-u. Pored AWS-a, neki od njih su: *Google Cloud Platform* (GCP), *IBM Cloud*, *Oracle Cloud*, *Alibaba Cloud*, *Microsoft Azure*.

Servisi koje nude ove kompanije mogu se podijeliti na:

- **Softver kao usluga, SaaS** (engl. *Software as Service*) – pruža mogućnost upotrebe dostupnih aplikacija koje su pokrenute na infrastrukturi *cloud*-a. Primjeri su: *Amazon WorkSpaces*, *Google Apps for Work*, *Microsoft Office 365*;
- **Platforma kao usluga, PaaS** (engl. *Platform as Service*) - korisnik može razvijati, testirati i distribuirati svoje aplikacije koje se pokreću na infrastrukturi *cloud*-a. Primjeri su: AWS *Elastic Beanstalk*, *Google App Engine*, *Heroku*;
- **Infrastruktura kao usluga, IaaS** (engl. *Infrastructure as Service*) – korisnik može upravljati obradom, skladištenjem, umrežavanjem i drugim osnovnim računarskim resursima. Primjeri su: *Amazon EC2*, *Google Compute Engine*, *Microsoft Azure* [1].

Kako *cloud* predstavlja skup resursa raspoređenih na različite geografske regije na cijeloj planeti, ono što je zajedničko za sve dobavljače jeste mogućnost razvoja aplikacije u različitim regionima, koji se mogu dijeliti na zone dostupnosti (engl. *availability zones*). Zone dostupnosti predstavljaju logičke centre podataka (engl. *data centers*), koji mogu da se sastoje od jednog ili više fizičkih centara podataka. Jedna aplikacija može da se postavi na resurse u okviru više zona dostupnosti. U tom slučaju, ako se u jednoj zoni desi otkaz, aplikacija će nastaviti da radi koristeći resurse iz druge zone. [7].

U nastavku su ukratko opisane najpopularnije alternative AWS-a i navedeni su njihovi najznačajniji servisi.

2.1. Microsoft Azure

Microsoft Azure je platforma razvijena 2010. godine, čija glavna prednost je integrisanost sa *Windows* i *Microsoft* softverom, što znači da se preduzeća koja koriste ove tehnologije često odlučuju za *Azure*.

Kao i svaki *cloud* dobavljač, *Azure* nudi mogućnost razvoja skalabilnih i visoko dostupnih aplikacija upotrebom određenih servisa. Neki od

najznačajnijih servisa su *Azure Virtual Machine* i *Virtual Machine Scale Sets*, koji služe za postavljanje aplikacije i postizanje skalabilnosti. Od servisa zaduženih za skladištenje resursa i podataka najznačajniji su *Blob Storage* i *SQL Database* [8].

2.2. Google Cloud Platform

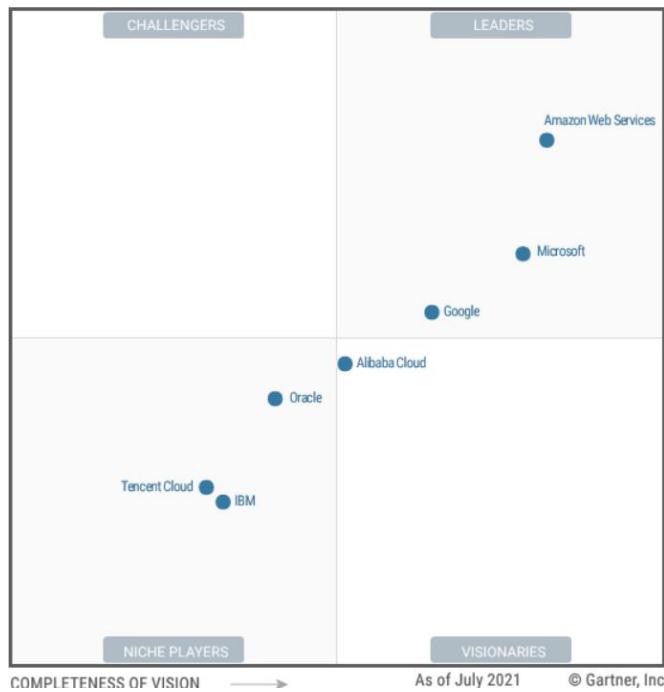
Google Cloud Platform (GPC) predstavlja platformu javno dostupnu od 2011. godine, koja je razvijena na istoj infrastrukturi koju *Google* koristi za svoje proizvode kao što su *Google Search*, *Gmail*, *Youtube*. Nije toliko popularna kod preduzeća kao *Azure* ili *AWS*, ali zahvaljujući brzom razvoju u posljednje vrijeme i napretku u oblasti mašinskog učenja, uspijeva da ostane konkurentna na tržištu. Broj servisa je nešto manji nego kod prethodno navedenih dobavljača. Najznačajniji su *Compute Engine*, *Cloud Storage*, *SQL* [9].

Na slici 2.1 prikazani su ekvivalentni AWS servisi za pomenute *Azure* i GCP servise, a oni će biti detaljno objašnjeni u narednom poglavlju.

	Azure	GCP	AWS
Basic Compute	<i>Virtual Machines</i>	<i>Compute Engine</i>	<i>EC2</i>
App Hosting	<i>App Service</i>	<i>App Engine</i>	<i>Elastic Beanstalk</i>
Object Storage	<i>Blob Storage</i>	<i>Cloud Storage</i>	<i>S3</i>
Relational Database	<i>SQL Database</i>	<i>Cloud SQL</i>	<i>RDS</i>
NoSQL Database	<i>Cosmos DB</i>	<i>Cloud Bigtable</i>	<i>DynamoDB</i>
Cloud Monitoring	<i>Monitor</i>	<i>Stackdriver</i>	<i>CloudWatch</i>

Slika 2.1. Ekvivalentni servisi koje nude *Azure*, GCP i AWS

Na osnovu istraživanja kompanije *Gartner* iz 2021. godine poznato je da danas na tržištu dominira AWS, u velikoj mjeri zahvaljujući najvećem broju servisa i najboljom mreži centara podataka. Na drugom mjestu je *Azure*, zatim *Google* i *Alibaba*. Na slici 2.2 prikazan je raspored kompanija na grafiku koji je nazvan „Magični kvadrant“ (engl. *Magic Quadrant for Cloud Infrastructure and Platform Services*), gdje može da se vidi trenutna pozicija kompanija na tržištu [10].



Slika 2.2. Magični kvadrant [1]

3. KORIŠĆENE TEHNIKE I TEHNOLOGIJE

U ovom poglavlju su opisani najznačajniji servisi koje nudi AWS [11], s naglaskom na servise koji omogućavaju postizanje elastičnosti i visoke dostupnosti i koji su korišćeni u radu, kao i tehnologije korišćene za izradu same aplikacije, *Spring* [12] i *Angular* [13].

3.1. AWS

AWS je platforma za računarstvo na *cloud*-u pokrenuta 2006. godine od strane kompanije *Amazon*. Danas, kao najuspješnija platforma u ovoj oblasti, nudi preko 200 servisa različitih namjena sa mogućnošću plaćanja po sistemu „*pay-as-you-go*“, što znači da korisnik plaća usluge samo onoliko koliko ih koristi. Svaki od servisa ima svoju cijenu koja može biti definisana na osnovu broja minuta ili sati upotrebe, na osnovu saobraćaja (mjeri se gigabajtima ili u brojem zahtjeva) ili na osnovu upotrebe skladišta (mjeri se

kapacitetom ili dužinom upotrebe). AWS takođe nudi i *Free Tier* režim, prema kojem u toku prvih godinu dana korišćenja platforme korisnik može da upotrebljava neke osnovne servise besplatno. [1].

Slično kao i kod ostalih platformi, globalnu arhitekturu AWS-a danas čini 81 zona dostupnosti raspoređena na 25 geografskih regiona, a u planu je još 21 nova zona i 7 novih regiona. Neki od regiona su: *US East (Ohio)*, *Africa (Cape Town)*, *South America (São Paulo)*, *Asia Pacific (Singapore)*, *Europe (Frankfurt)*, *Middle East (Bahrain)* itd, što je prikazano na slici 3.1 [6]. Gdje god da se nalazi, korisnik može da izabere bilo koji region i da rasporedi svoje aplikacije na više zona dostupnosti, što obezbeđuje visoku dostupnost. Takođe, odabirom zone korisnik može da približi aplikaciju korisnicima određenog geografskog područja da bi odziv aplikacije bio brži.



Slika 3.1. Postojeći i planirani geografski regioni u AWS [6]

Da bi korisnik mogao da radi sa AWS servisima potrebno je da kreira nalog, nakon čega će mu biti dostupan korisnički interfejs, odnosno AWS *Management Console* [14]. Ovaj korisnički interfejs daje mogućnosti za podešavanje raznih opcija i parametara, kako bi korisnik mogao da prilagodi upotrebu servisa svojim zahtjevima.

U nastavku su navedeni neki od najčešće korišćenih servisa:

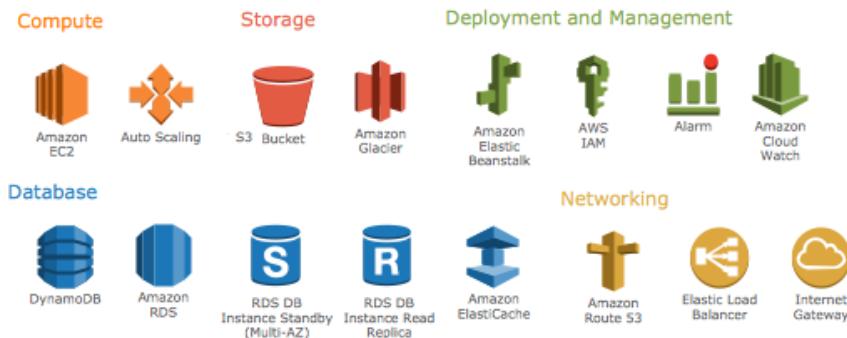
- ***Compute***
 - *Amazon Elastic Compute Cloud (EC2)* – virtuelna mašina na *cloud-u* sa izabranim operativnim sistemom i drugim

parametrima; najčešće se koristi kao server veb aplikacije [11,15];

- *Amazon Lambda* – servis za izvršavanje koda bez servera; kod se izvršava kada se desi određeni događaj [11];
- *Amazon Virtual Private Cloud (VPC)* – logički izolovan dio *cloud-a* koji korisnik alocira za svoje potrebe [11,16];
- *Amazon Elastic Beanstalk* – servis za automatsko postavljanje aplikacije na skalabilnu arhitekturu [11,17];
- *Amazon EC2 Auto Scaling* – servis za definisanje politike skaliranja EC2 instanci [11,18];
- i dr.
- ***Storage***
 - *Amazon Simple Storage Service (S3)* – servis za skladištenje objekata i fajlova [11,19];
 - *Amazon Glacier* – servis za arhiviranje i održavanje rezervne kopije fajlova kojima se ne pristupa često [11,20];
 - i dr.
- ***Database***
 - *Amazon Relational Database (RDS)* – relaciona baza podataka sa mogućnošću skaliranja [11,21];
 - *Amazon DynamoDB* – NoSQL baza podataka za skladištenje dokumenata [11];
 - *Amazon Aurora* – relaciona baza podataka namenjena za *cloud* i kompatibilna sa PostgreSQL i MySQL [11,21];
 - *Amazon ElastiCache* – *in-memory* servis za keširanje [11];
 - i dr.
- ***Networking & Content Delivery***
 - *Amazon CloudFront* – servis za dostavljanje sadržaja preko mreže (engl. *Content Delivery Network, CDN*) [11];
 - *Amazon Route 53* – servis za rutiranje, odnosno *Domain Name System (DNS)* [11,22] na *cloud-u* [11];
 - *Elastic Load Balancing (ELB)* – servis za raspoređivanje saobraćaja na više EC2 instanci [11,23];
 - i dr.
- ***Management Tools***
 - *Amazon CloudWatch* – servis za praćenje rada EC2 instanci i prikupljanje podataka [11,24];
 - i dr.
- ***Security, Identity & Compliance***
 - *Amazon Identity and Access Management (IAM)* – servis za definisanje pravila kontrole pristupa resursima [11,25];

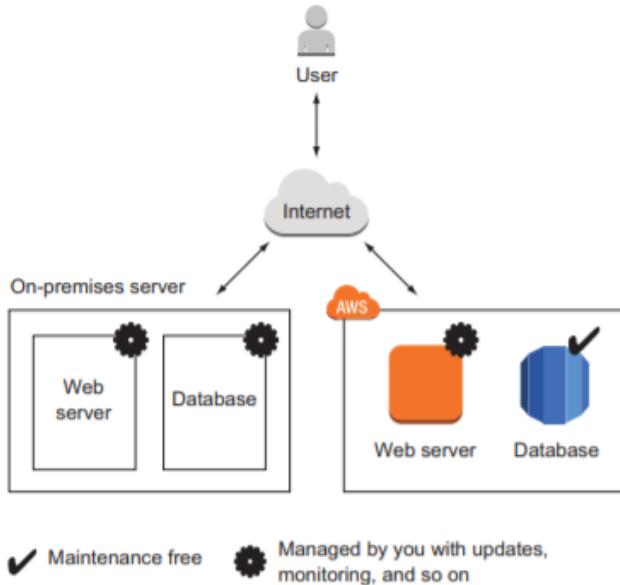
- i dr.
- ***Messaging***
 - *Amazon Simple Notification Service (SNS)* – servis za slanje obavještenja korisniku [11,26];
 - i dr.

Na slici 3.2 prikazane su ikone koje se koriste za predstavljanje nekih od navedenih servisa. Pored njih, AWS nudi i servise najmenjene za rad sa doker kontejnerima (ECR, ECS, Fargate, EKS), migraciju resursa na AWS (*Server Migration Service*, *Database Migration Service*), automatizaciju postavljanja koda na EC2 instance (*CodeDeploy*, *CodeCommit*), mašinsko učenje, robotiku, *Internet of Things* i dr [11].



Slika 3.2. Ikonе za predstavljanje servisa

Na slici 3.3 dat je primjer najjednostavnije arhitekture postavljene na lokalni server i te iste arhitekture postavljene na odgovarajuće AWS servise.



Slika 3.3. Primjer najjednostavnije arhitekture [1]

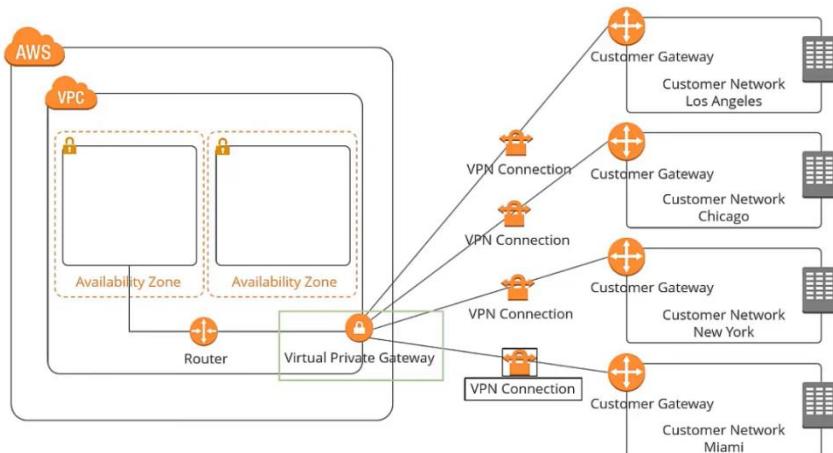
3.2. VPC

Amazon definiše **VPC** (engl. *Virtual Private Cloud*) [16] kao logički izolovan dio *cloud-a* koji omogućava korisniku da pokreće AWS resurse u virtuelnoj mreži koju je definisao. Korisnik sam kontroliše svoje mrežno okruženje, uključujući izbor opsega IP adresa (*Internet Protocol Address*), kreiranje podmreža, konfigurisanje tabela rutiranja i pristupa internetu. Kreiranjem privatnih i javnih podmreža pomoću koncepata kao što su *security grupe* i liste kontrole pristupa na mreži, obezbjeđuje se više slojeva zaštite pristupa instancama servisa na *cloud-u*.

AWS koristi **CIDR** (*Classless Inter-Domain Routing*) metodu za alociranje IP adresa i rutiranje. Prema ovoj notaciji identifikator IP adrese se sastoji od same adrese i njoj pridružene mrežne maske. Maska predstavlja 32-bitni binarni broj koji govori koje bitove IP adrese treba posmatrati kao bitove adrese mreže. U verziji IPv4, sama adresa se sastoji od četiri broja u opsegu od 0 do 255. Primjer adrese je 192.168.100.0/22. CIDR blok je niz IP adresa grupisanih u jedan zapis u tabeli rutiranja. Npr. IPv4 blok 192.168.100.0/22 predstavlja 1024 adrese od 192.168.100.0 do 192.168.103.255. Svaki Amazon nalog nudi podrazumijevani VPC (engl. *default VPC*), čiji CIDR blok je *subnet* maska 16, što znači da obuhvata 65536 privatnih IP adresa. Koristan je za testiranje servisa, međutim nije

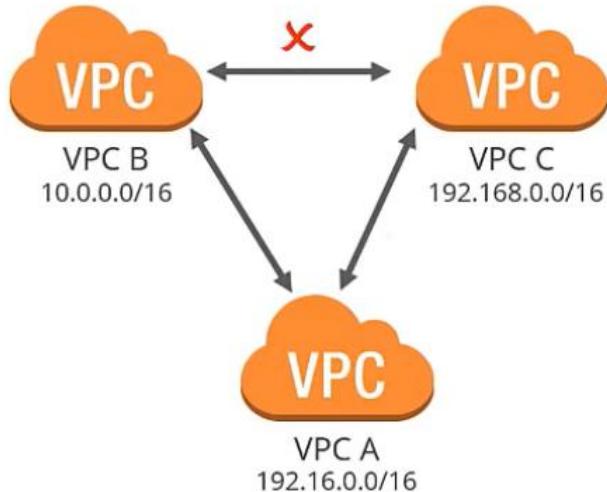
preporučljivo koristiti ga u produkciji. Kreiranje sopstvenog, prilagođenog VPC-a nudi mogućnosti povećanja bezbjednosti kreiranjem sopstvenih podmreža i pravila pristupa za njih.

Podrazumijevano, EC2 instance koje se pokrenu u prilagođenom VPC-u ne mogu da komuniciraju sa udaljenom mrežom. Ta komunikacija može da se ostvari pomoću **VPN** (*Virtual Private Network*) veza. Jedna od njih je *Site-to-site* veza, gdje se sa AWS strane veze postavlja *Virtual Private Gateway*, a na udaljenoj mreži se postavlja konfigurisani fizički uređaj ili aplikacija (*Customer Gateway Device*). VPN tunel za komunikaciju će se aktivirati kada se generše saobraćaj sa udaljene mreže. Na slici 3.4 prikazana je opisana arhitektura.



Slika 3.4. VPN veza [27]

Moguće je i uspostaviti vezu sopstvenog prilagođenog VPC-a sa bilo kojim VPC-om koji se nalazi u istom regionu, bez obzira na to kojem korisničkom nalogu pripada. Ta komunikacija se naziva **VPC Peering** i predstavlja vezu tipa jedan-na-jedan, što znači da veza nije tranzitivna. Da bi instance jednog VPC-a mogle da komuniciraju sa instanicama drugog, oni moraju da budu direktno povezani. Takođe, moraju da imaju različit opseg IP adresa, jer u suprotnom ne bi mogli da se povežu. Na slici 3.5 je prikazan primjer gdje VPC A može da komunicira sa VPC-om B i VPC-om C, ali B i C ne mogu da komuniciraju međusobno.

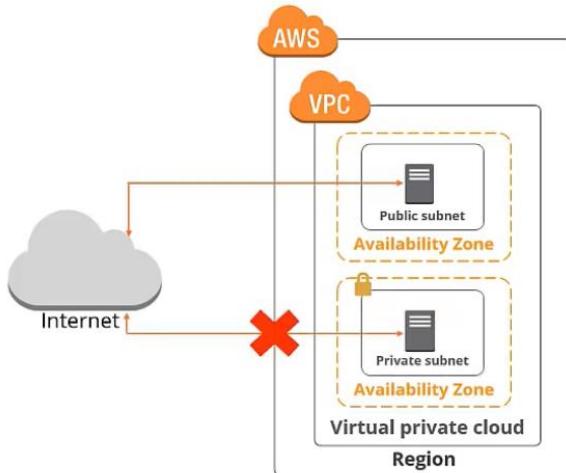


Slika 3.5. VPC Peering [\[27\]](#)

Privatne IP adrese su adrese koja nisu dostupne putem interneta, i koriste se za komunikaciju između instanci u istoj mreži. Prilikom pokretanja, nova instance dobija svoju privatnu IP adresu i interni DNS naziv (engl. *internal DNS hostname*) koji odgovara toj adresi. Da bi se pristupilo instanci putem interneta potrebna je **javna IP adresa**, kojoj se dodjeljuje eksterni DNS naziv (engl. *external DNS hostname*). Javne IP adrese se dodjeljuju iz rezerve IP adresa (*Amazon Pool of Public IP Addresses*). Kada se instance zaustavi ili terminira, adresa se oslobađa, a prilikom novog pokretanja instance dodjeljuje se nova adresa. Ukoliko je neophodno da instance zadrži svoju javnu adresu, koristi se **elastična IP adresa**. Ona predstavlja statičnu, perzistentnu javnu adresu, koju je potrebno prvo alocirati na korisničkom nalogu, a zatim pridružiti instanci.

Opseg IP adresa u korisničkom VPC-u čini **podmrežu** (engl. *subnet*). Da bi se pokrenula instance u nekom VPC-u neophodno je da postoji podmreža. Razlikuju se javne i privatne podmreže, u zavisnosti od toga da li resursi koji se pokreću u njima treba da budu povezani sa internetom. Na primjer, veb serveri uglavnom zahtjevaju vezu sa internetom, dok kod baza podataka to nije neophodno i one mogu da se smjesti u privatnu podmrežu. Svaki VPC sadrži podrazumijevanu podmrežu čija mrežna maska je 20, što znači da nudi 4096 adresa. VPC može da se prostire preko više zona dostupnosti, ali podmreža je uvek mapirana na jednu zonu. Na slici 3.6

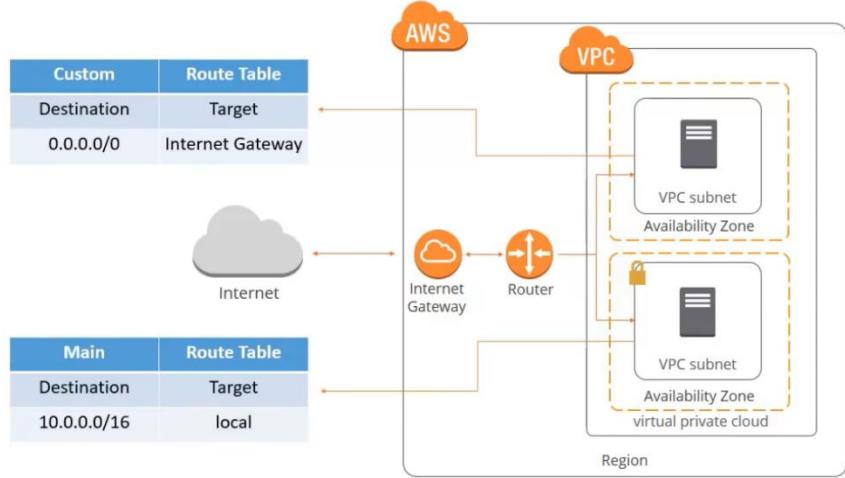
prikazan je VPC sa dvije zone dostupnosti, od kojih se u svakoj nalazi po jedna podmreža.



Slika 3.6. Primjer VPC-a [27]

Ono što javnu podmrežu čini javnom je njena **tabela rutiranja**, koja pomoću **Internet Gateway**-a šalje i prima mrežni saobraćaj sa interneta kada je to potrebno. Tabela rutiranja određuje kuda je mrežni saobraćaj usmjeren definisanjem skupa pravila koja se nazivaju rute. Svakoj podmreži je pridružena tabela rutiranja koja kontroliše rutiranje za tu podmrežu. Jednoj podmreži ne može biti pridruženo više tabele rutiranja istovremeno, ali tabela rutiranja može biti pridružena različitim mrežama istovremeno. Svaki VPC ima podrazumijevanu tabelu rutiranja. Na slici 3.7 prikazana je proširena prethodno opisana arhitektura, sa glavnom, odnosno podrazumijevanom, i novom tabelom rutiranja koje su pridružene javnoj i provatnoj podmreži.

Internet Gateway je horizontalno skalirana i visoko dostupna VPC komponenta koja obezbeđuje komunikaciju instanci sa internetskom mrežom. Ima ulogu da obezbijedi cilj (engl. *target*) u tabeli rutiranja za rutu koja usmjerava saobraćaj ka internetu. Instance koje komuniciraju sa internetskom mrežom moraju da imaju javnu IP adresu, odnosno da budu u javnoj podmreži. Da bi dobio pristup internetu, VPC-u se pridružuje tačno jedan *Internet Gateway*.

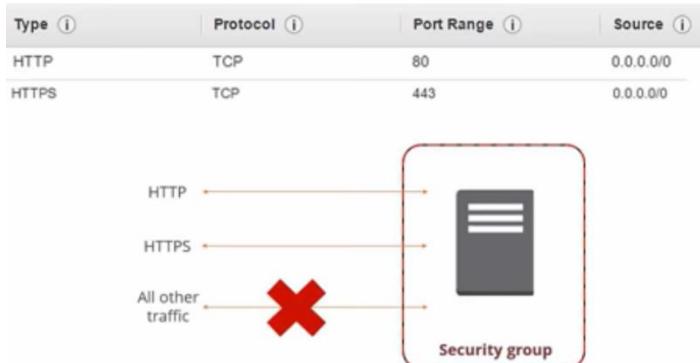
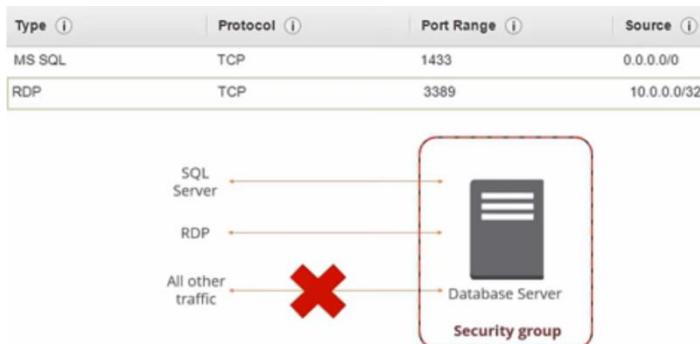


Slika 3.7. Proširen primjer VPC-a [27]

U slučajevima kada je potrebno da instance koje se nalaze u privatnoj podmreži imaju pristup internetu ili drugim AWS resursima, ali da ostane onemogućeno da internet inicira komunikaciju sa njima, koristi se **NAT Device** (engl. *Network Address Translation*). NAT Device prosljeđuje saobraćaj od privatne podmreže ka internetu ili drugim resursima i vraća dobijeni odgovor instanci. Kada se saobraćaj odvija prema internetu, izvorna IP adresa instance je zamijenjena adresom NAT Device-a, a kada se odgovor vraća adresa se ponovo prevodi u privatnu adresu instance. NAT Device mora da bude smješten u javnu podmrežu i da ima pridruženu elastičnu IP adresu. Da bi se sve to omogućilo, potrebno je podesiti tabelu rutiranja privatne podmreže tako da prosljeđuje saobraćaj NAT Device-u. Postoje dva tipa NAT Device-a: **NAT Gateway** i **NAT Instance**. Preporučuje se prvi, jer nudi veću dostupnost i zahtjeva manje održavanja.

Da bi komunikacija instanci sa internetom bila bezbjedna, potrebno je podesiti pravila za pristup mreži tako da dozvoljavaju samo željeni saobraćaj. Koriste se **security grupe** (engl. *security group*) i **liste kontrole pristupa** (engl. *Network Access Control List, Network ACL*).

Security grupe služe kao virtuelni zid koji kontroliše saobraćaj instance. Definišu se pravila koja označavaju dozvoljeni dolazeći ili odlazeći saobraćaj (engl. *inbound rules, outbound rules*) za instance kojima je pridružena grupa. Na slici 3.8 prikazana je security grupa pridružena instanci veb servera koja dozvoljava HTTP i HTTPS saobraćaj sa bilo kojeg izvora, a na slici 3.9 security grupa pridružena instanci servera baze podataka koja dozvoljava SQL i RDS saobraćaj, ali ograničava izvor RDP saobraćaja [28].

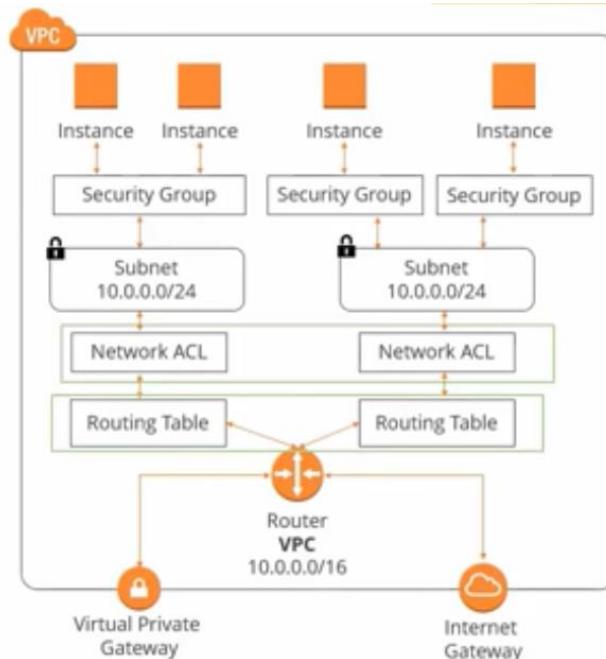
Slika 3.8. Security grupa za web server [\[27\]](#)Slika 3.9. Security grupa za server baze podataka [\[27\]](#)

Network ACL je opcionalni sloj bezbjednosti VPC-a koji služi kao virtuelni zid koji kontroliše saobraćaj podmreže. Definišu se pravila koja označavaju dozvoljeni ili zabranjeni dolazeći ili odčezaći saobraćaj. Na slici 3.10 prikazana su pravila za podrazumijevani ACL koji se automatski dodjeljuje podmreži. Pravila se izvršavaju redom, a posljednje, označeno zvjezdicom, obezbjeđuje da svaki paket koji se ne poklapa ni sa jednim iznad navedenim pravilom bude odbijen. Navedeni slojevi zaštite instanci u VPC prikazani su na slici 3.11.

Inbound					
Rule #	Type	Protocol	Port Range	Source	Allow/Deny
100	All traffic	All	All	0.0.0.0/0	ALLOW
*	All traffic	All	All	0.0.0.0/0	DENY

Outbound					
Rule #	Type	Protocol	Port Range	Source	Allow/Deny
100	All traffic	all	all	0.0.0.0/0	ALLOW
*	All traffic	all	all	0.0.0.0/0	DENY

Slika 3.10. Podrazumijevani Network ACL [\[27\]](#)



Slika 3.11. Slojevi zaštite instanci u VPC [\[27\]](#)

3.3. EC2

Amazon Elastic Compute Cloud (EC2) [15] je veb servis koji predstavlja virtualni server i pruža skalabilni računarski kapacitet na *cloud-u*. Eliminiše potrebu za ulaganjem u hardver i omogućava pokretanje onoliko instanci servera koliko je potrebno, podešavanje bezbjednosti, umrežavanje i upravljanje skladištem.

Pri kreiranju virtualnog servera osnovni koraci su izbor softvera i hardvera. *Amazon Machine Image* (AMI) je šablon koji sadrži softversku konfiguraciju (npr. operativni sistem, instalirane aplikacije i sl.). Pokrenuta EC2 instanca predstavlja kopiju AMI-a koja se izvršava na virtuelnom serveru. Instanca se izvršava sve dok se ne zaustavi ili terminira, osnosno prekine. AMI može da bude predefinisan od strane *Amazon-a* ili korisnički definisan. Kriterijumi za izbor AMI-a su region, operativni sistem, arhitektura (32-bitna ili 64-bitna), tip skladišta koje sadrži podatke za pokretanje instance, dozvola za pokretanje instance, itd. Dozvolu za pokretanje mogu da imaju svi AWS nalozi, samo eksplicitno navedeni nalozi ili samo vlasnik instance.

AMI može da se pokrene na različitim tipovima instanci. Tip instance određuje kombinaciju hardverskih resursa, kao što su procesor i memorija, i bira se u zavisnosti od opterećenja. Tipovi instance se mogu podijeliti na četiri grupe:

- računarski optimizovani – biraju se kada aplikacija zahtjeva procesorsku snagu;
- memorijski optimizovani – biraju se kada aplikacija kešira veliku količinu podataka;
- optimizovano skladištenje – biraju se kada aplikacija čuva veliku količinu podataka u lokalnom skladištu;
- za opštu svrhu – balansiraju procesorske i memorijske zahteve.

Pored izbora softvera i hardvera, EC2 omogućava i konfiguriranje instanci, odnosno definisanje koliko instanci je potrebno, u kojoj podmreži se nalaze, kojoj *security* grupi su pridružene, ponašanje instanci pri zaustavljanju i prekidanju, i sl. Moguće je i dodati skladište uz instancu (*root device volume*) gdje se smještaju podaci potrebni za pokretanje instance. Može biti *Elastic Block Store volume* (EBS) ili *instance store volume*. EBS predstavlja virtualni SSD disk koji postoji nezavisno od životnog ciklusa instance, odnosno čuva podatke i kada se instanca prekine, može da se otkači od instance i prikači nekoj drugoj instanci i da se kreira snimak (engl. *snapshot*) kao rezervna kopija podataka. *Instance store volume* čuva podatke samo dok se instanca ne zaustavi ili prekine.

3.4. S3, Glacier

Amazon Simple Storage Service (S3) [19,20] je servis za skladištenje objekata koji nudi veliku skalabilnost, dostupnost, sigurnost, pouzdanost i visoke performanse. Objekat predstavlja datoteku (npr. tekstualni fajl, jar fajl, slika, itd) sa svojim metapodacima. Korisnici mogu da skladište bilo koju količinu podataka veličine i do 5TB, te mnoge kompanije koriste ovaj servis za različite svrhe (veb sajtovi, mobilne aplikacije, arhiviranje podataka, logovanje i sl.). Objektima može lako da se pristupi i kada se jednom sačuvaju vjerovatnoća da se izgube je veoma mala. Takođe, visoka skalabilnost daje mogućnost da se veličina skladišta povećava sa povećanjem količine podataka odmah kada je to potrebno. Nije ograničen na jednu zonu dostupnosti, što znači da je obezbjeđena visoka dostupnost.

Objekti se čuvaju u S3 *bucket-ima*, koji predstavljaju kontejnere za skladištenje podataka kreirane od strane korisnika. Kada se objekat doda u *bucket*, dodjeljuje mu se jedinstveni identifikator na osnovu kojeg mu se može pristupiti. Objekti se mogu *upload-ovati*, *download-ovati*, kopirati i brisati.

Korisnik ima mogućnost da konfiguriše politike pristupa *bucket-u* koji je kreirao. One specificiraju ko može da pristupi *bucket-u*, šta sve može da vidi, da li može samo da čita ili i da piše, itd. Takođe, može da konfiguriše politike životnog ciklusa objekata u *bucket-ima*, npr. da definiše vrijeme kada će objekat automatski da se obriše ili prebaci na neko drugo mjesto. S3 nudi i opcije korisne za developere, kao što su mogućnost za *rollback*, tako da se često koristi i kao rezervni sistem za kontrolu verzija koda.

Svaki S3 *bucket* pripada jednoj od klase koje određuju hardver a bira ih korisnik pri kreiranju:

- *Standard for frequent data access* – koristi se kada se podacima često pristupa, npr. svakodnevno, pa bi trebalo da se obezbijedi brzo preuzimanje;
- *Standard for infrequent data access* – koristi se za podatke kojima se rjeđe pristupa;
- *Amazon Glacier* – servis za arhiviranje podataka na *cloud-u*. Koristi se za skladištenje podataka kojima se rijetko pristupa i nisu neophodne visoke performanse, ali je prednost znatno niža cijena u odnosu na ostale klase;
- *One Zone-IA Storage Class* – podaci se čuvaju u jednoj zoni dostupnosti, što smanjuje dostupnost, ali je i cijena niža;
- *Standard Reduced Redundancy Storage* – koristi se za podatke koji se često kopiraju, pa nije potrebna velika pouzdanost.

Na slici 3.12 prikazane su karakteristike klase.

Storage Class	Durability	Availability	SSL support	First byte latency	Lifecycle Management Policies
STANDARD	99.999999999%	99.99%	Yes	Milliseconds	Yes
STANDARD_IA	99.999999999%	99.99%	Yes	Milliseconds	Yes
ONEZONE_IA	99.999999999%	99.5%	Yes	Milliseconds	Yes
GLACIER	99.999999999%	99.99%	Yes	Minutes or Hours	Yes
RRS	99.99%	99.99%	Yes	Milliseconds	Yes

Slika 3.12. Karakteristike klase S3 *bucket-a* [\[19\]](#)

Primjer primjene različitih karakteristika klasa je premještanje objekta u drugo skladište po potrebi u sklopu upravljanja njegovim životnim ciklusom. Npr. ako objektu koji se nalazi u S3 *Standard* skladištu niko nije pristupio 30 dana, pomoću akcije za transakciju (engl. *Transaction Action*) on se automatski premješta u S3 *Infrequent Access* skladište, a ako mu i dalje niko ne pristupa, premješta se u *Glacier*.

Ukoliko je potrebno ograničiti pristup *bucket-u* koriste se IAM politike (engl. *Identity and Access Management*), koje omogućavaju da se pomoću skripte definiše koji korisnici, uloge ili aplikacije imaju pravo pristupa *bucket-u*. Uloga je skup permisija koji se dodjeljuje nekom entitetu (npr. EC2 instanca) i daje mu privremene kredencijale za pristup nekom resursu. Još jedna metoda zaštite podataka je enkripcija podataka, koja sprečava korisnike bez odgovarajućeg ključa da pročitaju sadržaj.

S obzirom na to da radi sa statičkim podacima, posebna mogućnost koju nudi S3 je hostovanje statičkog veb sajta. Kada se *bucket* konfiguriše tako da dozvoljava ovu opciju, veb sajt postaje dostupan na određenom *endpoint-u* koji pored imena *bucket-a* sadrži i ime regiona u kojem se nalazi.

3.5. RDS, Aurora

Amazon Relation Database Service (RDS) [21] je servis koji omogućava rad sa relacionim bazama podataka na *cloud-u*. Automatizuje podešavanje i upravljanje bazom podataka i pruža promjenljiv kapacitet, skalabilnost i pouzdanost. Vodi računa o rezervnim kopijama podataka, detekciji otkaza i oporavku od otkaza. Daje izbor od nekoliko različitih baza podataka: *Amazon Aurora*, *PostgreSQL*, *MySQL*, *MariaDB*, *Oracle Database* i *SQL Server*.

Instanca baze podataka predstavlja izolovano okruženje na *cloud-u* kojem može da se pristupi na isti način kao i samostalnoj bazi podataka. Procesorski i memorijski kapacitet baze podataka određeni su klasom instance baze podataka, te se promjenom klase može izvršiti vertikalno skaliranje baze podataka. Klase su podijeljene u tri grupe:

- *standard* – balansirana optimizacija memorije i procesora;
- *memory optimized* – optimizacija memorije;
- *burstable performance* – veći kapacitet procesora.

RDS instance koriste EBS kao skladište (engl. *storage volume*). Postoje tri tipa skladišta, a razlikuju se u performansama i cijeni:

- *General Purpose SSD* – ekonomično skladište koje može da se koristi za aplikacije sa opterećenjem bilo koje vrste;
- *Provisioned IOPS SSD* – koristi se za aplikacije koje zahtjevaju brze ulazno-izlazne operacije;
- *Magnetic* – magnetno skladište; ima slabije performanse od SSD skladišta.

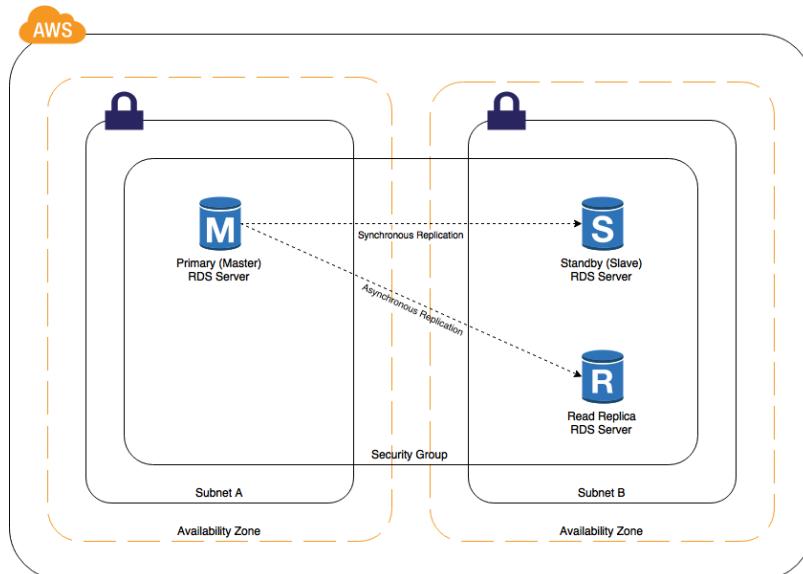
Da bi se obezbijedila sigurnost baze podataka instance se pokreću unutar VPC-a, u podmreži koja je zaštićena *security* grupom i/ili ACL listom. Amazon vodi računa o zaštiti podataka u bazi, i daje tri opcije za autentikaciju korisnika baze: autentikacija lozinkom, autentikacija pomoću IAM servisa i *Kerberos* autentikacija. *Kerberos* je protokol za autentikaciju koji koristi enkripciju sa simetričnim ključem i eliminiše potrebu slanja lozinki putem interneta.

RDS pruža i mogućnost automatskog kreiranja snimka kao rezervne kopije baze podataka (engl. *snapshot*). Snimak se čuva u S3 *bucket-u*, a korisnik može da specificira period u toku dana kada će se kreirati kopija (engl. *backup window*) i period između dva uzastopna kreiranja kopije (engl. *backup retention period*). Takođe, od snimka može da se kreira instance baze podataka, odnosno da se obnovi baza od koje je nastao snimak.

Jedna od najvećih prednosti RDS servisa je visoka dostupnost i podrška za oporavak od otkaza, što se postiže opcijom *Multi-AZ Deployment*, odnosno postavljanjem instance baze podataka na više zona dostupnosti.

Ukoliko je odabrana ova opcija, RDS će automatski kreirati *standby* repliku primarne baze u nekoj drugoj zoni dostupnosti. *Standby* replika se održava automatski sinhronom replikacijom primarne baze, što znači da se podaci upisuju u primarnu bazu i u repliku istovremeno i one su uvijek sinhronizovane. U slučaju prekida rada primarne baze RDS automatski prelazi na *standby* repliku, i taj period oporavke od otkaza (engl. *failover*) traje oko 60-120 sekundi.

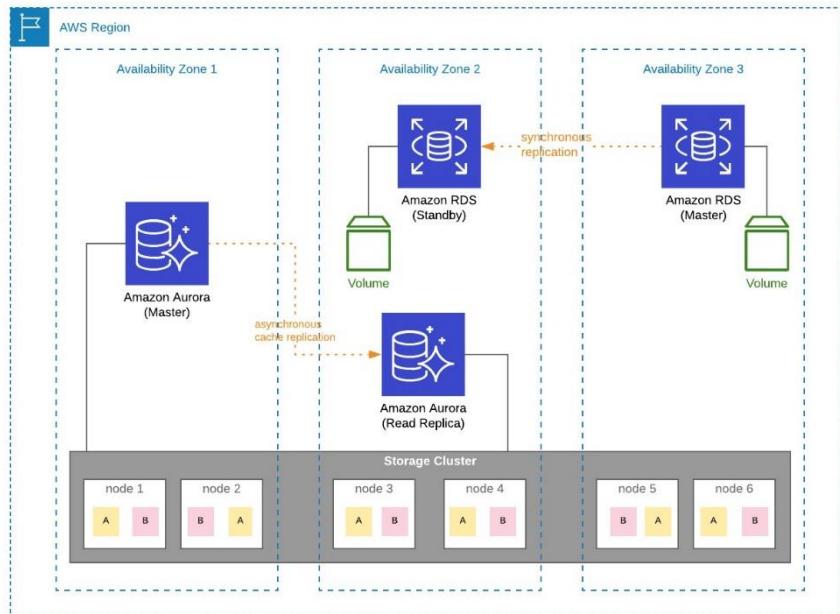
Kako bi se smanjilo opterećenje primarne baze podataka koristi se horizontalno skaliranje, odnosno kreiranje *read* replike. Ona predstavlja kopiju primarne baze koja se automatski ažurira asinhronom replikacijom, odnosno podaci se ažuriraju tek nakon što se upišu u primarnu bazu. Smanjuje opterećenje primarne baze tako što preuzima i izvršava upite za čitanje. Moguće je kreirati više *read* replika za jednu instancu baze i unaprediti *read* repliku tako da i sama postane instanca. Na slici 3.13 prikazana je primarna baza sa svojom *standby* replikom i *read* replikom.



Slika 3.13. Primarna baza, *standby* replika i *read* replika [21]

Amazon Aurora je relaciona baza podataka namijenjena isključivo za korišćenje na *cloud-u* i podržana od strane RDS-a. Kompatibilna je sa *MySQL* i *PostgreSQL* bazama, ali nudi značajno bolje performanse od njih (brža je 3-5 puta) i više mogućnosti za postizanje skalabilnosti i dostupnosti. Novina koju uvodi *Aurora* je korišćenje klastera baze podataka (engl.

database cluster) umjesto obične instance baze podataka. Klaster podrazumijeva jednu ili više instanci baze koje mogu da budu primarne instance ili *Aurora read* replike, zajedno sa klasterom skladišta (engl. *cluster database storage volume*). Primarna instanca može da izvršava upite za čitanje i pisanje, a *read* replike su kopije koje izvršavaju samo upite za čitanje i na taj način smanjuju opterećenje. *Read* replike su raspoređene u različite zone dostupnosti i moguće je kreirati maksimalno 15 replika. Takođe, Aurora nudi mogućnost automatskog dodavanja i uklanjanja *read* replika u zavisnosti od opterećenja i definisanja minimalnog i maksimalnog broja replika, što je prednost u odnosu na ostale RDS baze. Još jedna značajna prednost je automatsko unapređenje *read* replike u primarnu bazu, dok kod RDS baza unapređenje *read* replike mora da se odradi ručno. Klaster skladišta predstavlja izdvojeno skladište baze podataka koje je raspoređeno na više zona dostupnosti i kojem pristupaju sve instance i replike. Zahvaljujući tome postiže se visoka dostupnost podataka i značajno ubrzanje sinhronizacije podataka i upita u odnosu na RDS bazu, gdje svaka instanca ima svoje skladište. Klaster skladišta sadrži šest čvorova, odnosno šest kopija podataka u tri različite zone dostupnosti.



Slika 3.14. Poređenje dostupnosti arhitekture RDS i Aurora baze podataka

[\[21\]](#)

3.6. CloudWatch

Amazon CloudWatch [24] je servis za nadgledanje AWS resursa i aplikacija u realnom vremenu. Koristi se za prikupljanje i praćenje vrijednosti metrika, tj. promjenljivih koje mogu da se mjeru. Postoje različite metrike za AWS resurse (npr. procenat iskorišćenosti procesora, količina dolazećeg i odlazećeg saobraćaja EC2 instance, itd.) i za nadgledanje korišćenja AWS servisa (npr. broj operacija koje su izvršene na jednom AWS nalogu), i sl. Moguće je definisati i sopstvene metrike. Na osnovu prikupljenih podataka formiraju se statistički izvještaji, aktiviraju se odgovarajući alarmi i šalju se obavještenja korisniku.

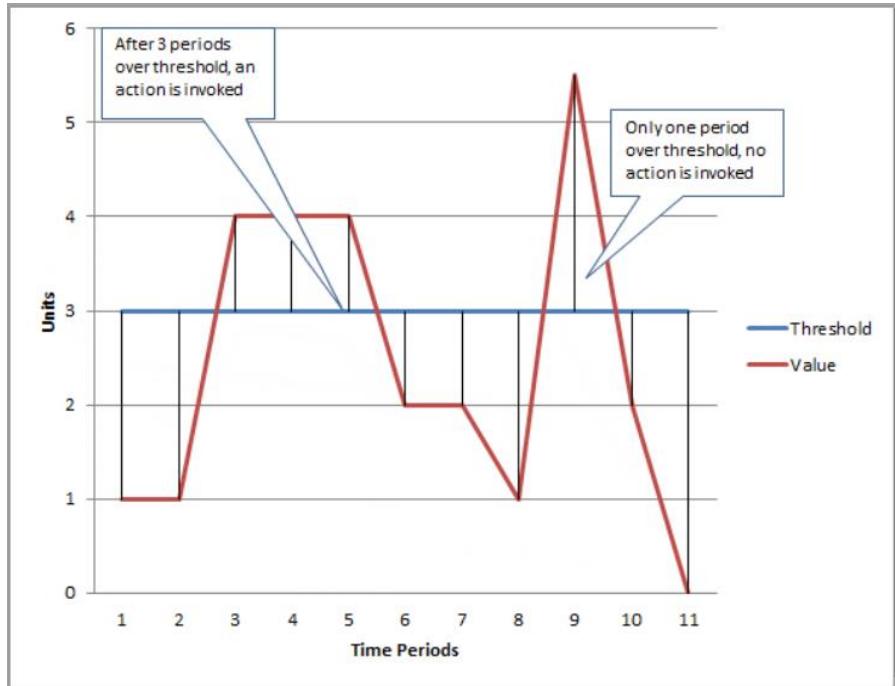
Metrike su definisane imenom, imenskim prostorom i dimenzijama. Imenski prostori služe kao kontejneri u kojima se čuvaju metrike, a dimenzije su kategorije u koje se svrstavaju pojedinačne karakteristike metrike i mogu da se koriste za filtriranje. Vrijednost metrike u određenom vremenskom periodu nazva se *data point*. Svaki *data point* ima svoju vremensku oznaku i (opciono) mjernu jedinicu. Vremenski period *data point-a*, odnosno vremenski interval između dva uzastopna mjerjenja, može da bude jedan minut (standardna rezolucija) ili jedan sekund (visoka rezolucija). Često se koriste i umnošci ovih vrijednosti, najčešće pet minuta ili jedan sat. Što je manja granularnost podaci se duže čuvaju u skladištu, najduže do 15 mjeseci.

Da bi se pratile vrijednosti metrika kreiraju se alarmi. Alarm se aktivira kada vrijednost metrike ili neki izraz koji sadrži tu vrijednost pređe definisani prag. Može da se nalazi u tri različita stanja: OK (nije aktiviran), ALARM (aktiviran) i INSUFFICIENT_DATA (nema dovoljno podataka da se odredi stanje alarma). Moguće je promijeniti način na koji *CloudWatch* posmatra slučaj kada nema dovoljno podataka, tako da se ne koristi posebno stanje za njega. Opcije su: podaci su „dobri“ (ne prelaze prag), podaci su „loši“ (prelaze prag) i podaci se ignorisu (zadržava se prethodno stanje).

Procjena stanja alarma naziva se evaluacija alarma. Pri definisanju alarma potrebno je odrediti:

- period *data point-a* – vremenski interval u kojem se izvršava jedno mjerjenje;
- period evaluacije – koliko uzastopnih perioda *data point-a* se posmatra u jednoj evaluaciji alarma;
- broj *data point-ova* u jednom periodu evaluacije koji moraju da pređu prag kako bi se alarm aktivirao.

Na slici 3.15 prikazana je aktivnost alarma čiji period evaluacije je 3 jedinice vremena (npr. sekunde), a broj *data point-ova* koji moraju da pređu prag da bi se alarm aktivirao je takođe 3. Prema tome, alarm sa slike će se aktivirati samo jednom.



Slika 3.15. Primjer evaluacije alarma [\[24\]](#)

CloudWatch pruža mogućnost specificiranja određenih akcija koje se izvršavaju pri promjeni stanja alarma. Postoje četiri tipa akcija:

- slanje notifikacije na mejl korisnika pomoću SNS servisa (engl. *Simple Notification Service*);
- kreiranje obavještenja za korisnika kada se desi greška u radu AWS servisa na njegovom nalogu;
- zaustavljanje, terminiranje ili ponovno pokretanje EC2 instance;
- skaliranje *Auto Scaling* grupe.

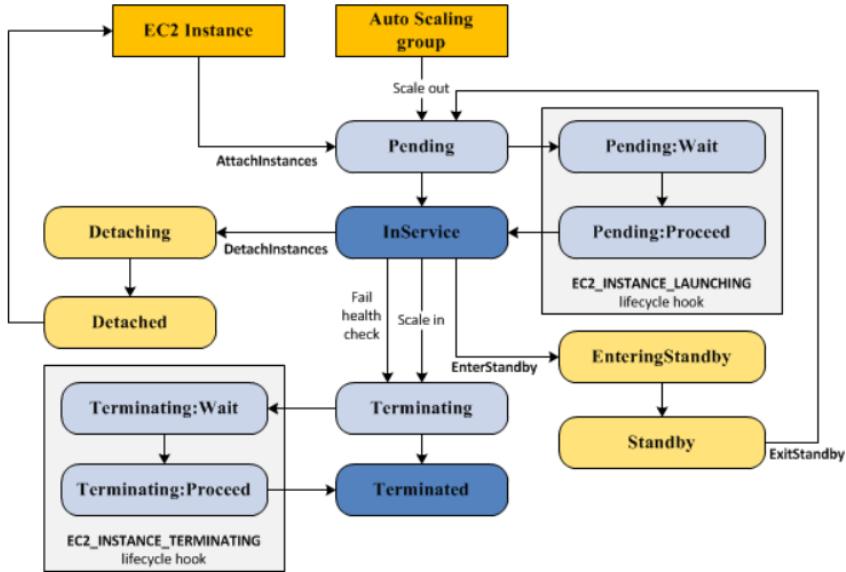
3.7. Auto Scaling, ELB

Amazon EC2 Auto Scaling [\[18\]](#) je servis koji služi za automatsko prilagođavanje kapaciteta na osnovu praćenja rada aplikacije da bi se održale dobre performanse uz najnižu moguću cijenu. Obezbjeduje da u svakom trenutku postoji onoliko EC2 instanci koliko je potrebno u zavisnosti od

opterećenja. Skup EC2 instanci aplikacije se naziva *Auto Scaling* grupa. Korisnik može da specificira minimalan, maksimalan i poželjan kapacitet grupe. Na osnovu toga EC2 *Auto Scaling* će osigurati da broj EC2 instanci u grupi nikad ne bude veći od maksimalnog ili manji od minimalnog, kao i da bude jednak broju poželjnih instanci ukoliko opterećenje ne zahtjeva povećanje ili smanjenje grupe.

Velika prednost ovakvog skaliranja je i mogućnost da se *Auto Scaling* grupa rasporedi na nekoliko zona dostupnosti, što znači da ako jedna zona postane nedostupna, pokrenuće se nove instance u nekoj drugoj zoni i aplikacija će nastaviti sa radom bez smetnji. Takođe, u slučaju da se desi otkaz u jednoj instanci, *Auto Scaling* može da detektuje otkaz, prekine instancu i pokrene novu instancu.

Prije pokretanja *Auto Scaling* grupe potrebno je podesiti konfiguraciju za pokretanje pojedinačne EC2 instance (engl. *launch configuration*). Ona podrazumijeva izbor AMI-a, tipa instance, *security* grupe, itd. Nova instanca se može pokrenuti ručnim dodavanjem instance, spajanjem već postojeće EC2 instance sa *Auto Scaling* grupom ili povećavanjem kapaciteta (engl. *scale-out*). Kada se instanca pokrene aktivira se mehanizam za provjeru ispravnosti instance (engl. *health check*), koji podrazumijeva provjeru rada odabranog *endpoint*-a. Ukoliko instanca prođe provjeru, nalazi se u stanju *InService*, a ukoliko ne prođe prelazi u stanje *OutOfService*. Instanca se terminira kada je u stanju *OutOfService*, kada se ručno prekine, kada se odvoji u posebnu EC2 instancu ili kada se smanjuje kapacitet grupe (engl. *scale-in*). Postoji i mogućnost da instanca pređe u stanje *standby*, što znači da i dalje pripada grupi, ali je trenutno neaktivna dok korisnik obavlja neke operacije nad njom. Navedena stanja su dio životnog ciklusa instance, koji je prikazan na slici 3.16. Pri promjeni stanja instance moguće je izvršiti akcije, npr. instaliranje softvera prilikom pokretanja ili kopiranje podataka prilikom terminiranja. To omogućuje *Lifecycle hooks* mehanizam, prevodeći instancu u stanje *Pending:Wait* ili *Terminating:Wait* sve dok se ne izvrši naznačena akcija.



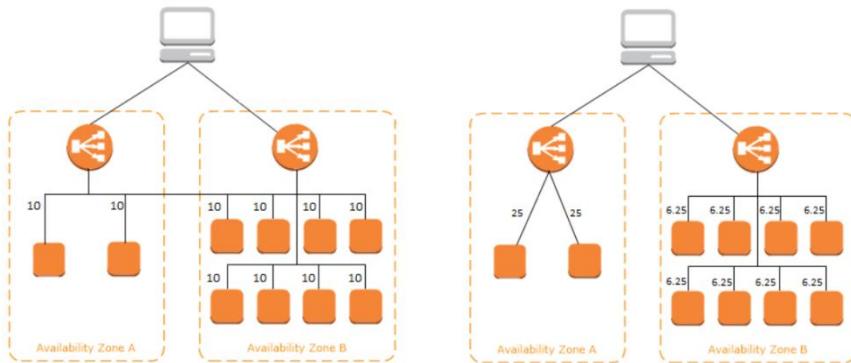
Slika 3.16. Životni ciklus EC2 instance u *Auto Scaling* grupi [18]

Zbog postojanja većeg broja instanci koje opslužuju zahtjeve upućene aplikaciji, javlja se potreba za mehanizmom koji će distribuirati pristigne zahtjeve. U tu svrhu se koristi *Elastic Load Balancer* (ELB) [23]. ELB automatski raspoređuje zahtjeve tako da se operećenje ravnomjerno podijeli na sve ispravne instance u jednoj ili više zona dostupnosti, kako nijedna ne bi bila preopterećena. Postoje četiri vrste ELB-a:

- *Classic Load Balancer* – rutiranje u transportnom (TCP/SSL) ili aplikacionom sloju (HTTP/HTTPS) OSI modela [29]; najčešće se koristi za stare EC2 instance koje su pokrenute izvan VPC-a;
- *Application Load Balancer* – rutiranje u aplikacijskom sloju; vezuje se sa *target* grupama EC2 instanci i dozvoljava definisanje logike za rutiranje, odnosno eksplicitno navođenje koja *target* grupa obrađuje koje zahtjeve;
- *Network Load Balancer* – rutiranje u transportnom sloju na osnovu podataka o adresi preuzetih iz zaglavlj zahtjeva;
- *Gateway Load Balancer* – rutiranje u mrežnom sloju.

Da bi ELB mogao da prima zahtjeve sa interneta, mora da se podesi opcija *internet-facing*, što znači da njegov DNS naziv može da se pretvoriti u javnu IP adresu.

U svakoj registrovanoj zoni dostupnosti ELB pravi čvor koji dalje raspoređuje zahtjeve na instance u toj zoni. Preporučuje se da se izabere opcija *Cross-zone load balancing*, jer u tom slučaju svaki čvor ravnomjerno raspoređuje saobraćaj na instance u svim zonama, dok u suprotnom čvor može da raspoređuje saobraćaj samo na instance svoje zone. Na slici 3.17 prikazan je primjer gdje svaki čvor raspoređuje po 50% saobraćaja. U prvom slučaju svaka instance će dobiti po 10% saobraćaja, dok će u drugom slučaju zahtjevi biti neravnomjerno raspoređeni.



Slika 3.17. ELB sa i bez *Cross-zone load balancing* [23]

Skaliranje (povećavanje ili smanjivanje) *Auto Scaling* grupe može da bude ručno, dinamičko, prediktivno i planirano. Ručno skaliranje se svodi na izmjenu minimalnog, maksimalnog i poželjnog kapaciteta u skladu sa tim koliko instanci korisnik procijeni da je potrebno. Nakon toga, *Auto Scaling* će sam da pokrene ili terminira instance u skladu sa novim vrijednostima. Prediktivno i planirano skaliranje se koristi kada se na osnovu iskustva može predvidjeti u kojim vremenskim periodima će opterećenje biti veće, te se u tim periodima povećava broj instanci. Na primjer, neke aplikacije su značajno opterećene u vrijeme radnih sati, dok u večernjim satima imaju dosta manji broj zahtjeva. Prednost ovakvog skaliranja je to što je unapred poznato kada je potrebno izvršiti skaliranje, pa se nove instance mogu pokrenuti prije nego što opterećenje krene da se povećava.

S druge strane, kod dinamičkog skaliranja nije unapred poznato kada će biti potrebno skaliranje. To se određuje dinamički na osnovu praćenja rada instanci pomoću *CloudWatch* servisa. Dakle, kada *CloudWatch* metrike pokažu da je opterećenje povećano, u tom trenutku se aktivira alarm i pokreće se nova instance. Isto važi i za smanjivanje broja instanci. Ovaj tip skaliranja

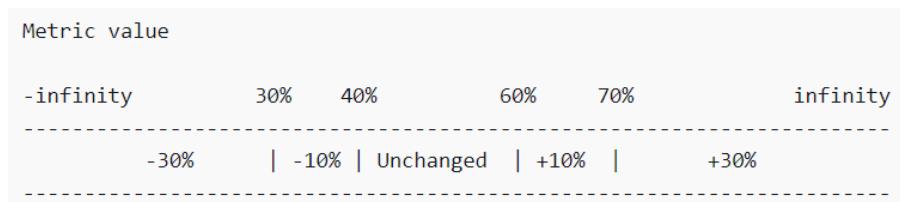
je koristan za aplikacije kod kojih se opterećenje mijenja, ali ne ciklično, tako da se ne može predvidjeti promjena. Postoje dvije vrste politika dinamičkog skaliranja: *Target Tracking Policies* i *Step and Simple Scaling Policies*. Kada se koristi *Target Tracking Policies*, prvo je potrebno izabrati metriku skaliranja i postaviti joj ciljnju vrijednost, najčešće izraženu u procentima. *Auto Scaling* će kreirati odgovarajuće alarne čijim aktiviranjem će se pokretati ili terminirati instance, tako da vrijednost izabrane metrike bude što bliže zadatoj vrijednosti. Moguće je definisati sopstvene metrike, a predefinisane metrike su:

- *ASGAverageCPUUtilization* – prosječna iskorišćenost procesora u *Auto Scaling* grupi;
- *ASGAverageNetworkIn* – prosječan broj bajtova primljen preko mreže;
- *ASGAverageNetworkOut* – prosječan broj bajtova poslat preko mreže;
- *ALBRequestCountPerTarget* – broj zahtjeva koje izvrši jedna instanca u *Application Load Balancer target* grupi.

Kod *Step and Simple Scaling policies*, korisnik bira metrike skaliranja i pragove čiji prelazak će aktivirati alarne, kao i koliko instanci da se pokrene ili terminira kada se aktivira alarm. Razlika između *simple* i *step* skaliranja je u tome što *step* skaliranje pruža posebna prilagođavanja (engl. *step adjustments*). Prilagođavanja omogućavaju da se izvrše različite akcije skaliranja u zavisnosti od toga koliko je prag prekoračen. Postoje tri načina da se specificira akcija skaliranja:

- *ChangeInCapacity* – navodi se tačan broj instanci koje se dodaju ili uklanjuju;
- *ExactCapacity* – navodi se tačan broj instanci koji treba da bude u grupi nakon skaliranja;
- *PercentChangeInCapacity* – grupa se povećava ili smanjuje za određeni procenat.

Za sva tri načina važi da ako je navedeni broj pozitivan, instance se dodaju, a ako je negativan instance se uklanjuju. Ako broj nije cijeli on se zaokružuje. Na slici 3.18 dat je primjer politike skaliranja grupe od 10 instanci. Prag metrike je postavljen na 50%, koristi se način prilagođavanja definisanjem procenta i definisana su sljedeća prilagođavanja: ako je vrijednost metrike za 10% veća ili manja od praga ništa se ne mijenja, ako je vrijednost metrike za 10% do 20% veća ili manja od praga prilagođavanje je 10% (odnosno -10%), i ako je vrijednost veća ili manja za 20% prilagođavanje je 30% (odnosno -30%). Primjenom ove politike, u slučaju kada je vrijednost metrike 60% aktiviraće se drugo prilagođavanje i grupi će se dodati jedna instance (10% od ukupno 10 instanci).



Slika 3.18. Primjer za *step scaling policy* [23]

3.8. Elastic Beanstalk

Elastic Beanstalk [17] je servis koji omogućava razvoj i upravljanje aplikacijama na *cloud*-u bez potrebe za učenjem o infrastrukturi na kojoj se one pokreću. Korisnik *upload*-uje zapakovan izvorni kod svoje aplikacije i daje opšte informacije o njoj, a *Elastic Beanstalk* pokreće okruženje i vodi računa o AWS resursima, kapacitetu, skaliranju, raspoređivanju opterećenja (engl. *load balancing*), nadgledanju (engl. *monitoring*) itd. Kada se aplikacija pokrene u okruženju, korisnik može da upravlja okruženjem i da razvija nove verzije svoje aplikacije. *Elastic Beanstalk* podržava aplikacije razvijene u programskim jezicima *Go*, *Java*, *.NET*, *Node.js*, *PHP*, *Python* i *Ruby*.

Moguće je kreirati novu aplikaciju na osnovu postojećeg primjera aplikacije. Podrazumijevano okruženje pri kreiranju nove aplikacije sadrži sljedeće resurse:

- EC2 instance – virtuelne maštine konfigurisane za pokretanje aplikacije na izabranoj platformi;
- *security grupa* za EC2 instance – konfigurisana tako da prima HTTP saobraćaj sa porta 80;
- S3 bucket – skladište za izvorni kod, log fajlove, i sl.
- *CloudWatch* alarmi – dva alarma za nadgledanje opterećenja EC2 instanci; kada se alarm okine *Auto Scaling* povećava ili smanjuje broj instanci;
- *CloudFormation* stek – kreiranje šablonu koji opisuju potrebne resurse, odnosno AWS servise, i pokretanje resursa definisanih šablonom;
- Internet domen – pridruženi domen koji završava sa „*elasticbeanstalk.com*“.

3.9. Spring

Spring [12,16] je radni okvir za razvoj aplikacija u programskom jeziku *Java*. Upravlja infrastrukturom, tako da korisnik može da se fokusira

na tehnologiju i realizaciju. Pomoću anotacija i konfiguracionih fajlova može se definisati na koji način *Spring* vodi računa o životnom ciklusu Java objekata (POJO), njihovom dobavljanju i povezivanju (*Dependency Injection*). U ovom radu korišćen je ***Spring Boot***, kao proširenje *Spring*-a koje pojednostavljuje konfigurisanje i razvoj aplikacije kroz skup gotovih rješenja (npr. jednostavnije pokretanje i upravljanje paketima, automatska konfiguracija, itd.).

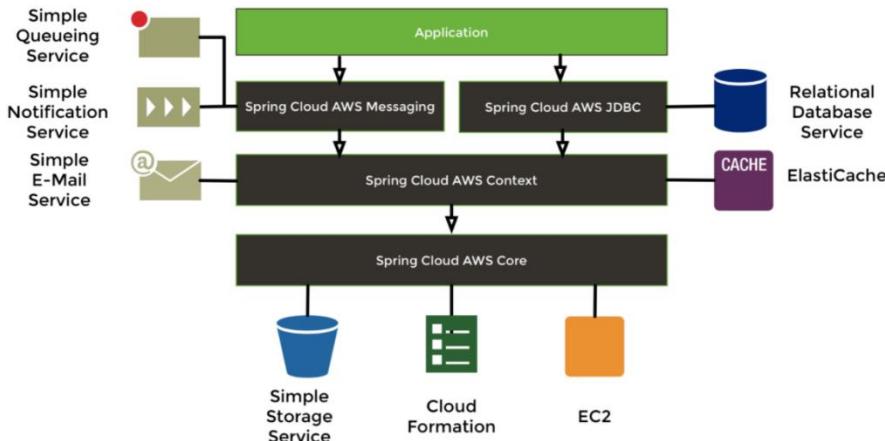
Spring obezbeđuje i podršku za rad sa veb aplikacijama pomoću ***web modula***, kao i podršku za REST (*REpresentational State Transfer*) [\[30\]](#) servise. REST je stil softverske arhitekture koji se zasniva na postojanju resursa kojima se upravlja putem skupa predefinisanih operacija (CRUD operacije - dodavanje, izmjena, brisanje, čitanje) preko dogovorenog protokola, najčešće HTTP. *Web* modul omogućava razvoj MVC arhitekture (*model-view-controller*), a kada je u pitanju REST koristi se samo kontroler koji radi sa *Java* objektima i bavi se formatom u kojem se resursi isporučuju klijentu, najčešće JSON (*JavaScript Object Notation*) format [\[31\]](#). Zahvaljujući REST tehnologiji, serverska strana i klijentska strana mogu da se razvijaju nezavisno jedna od druge i da komuniciraju slanjem resursa putem protokola. Prednost REST-a je i *stateless* komunikacija, odnosno serverska strana ne čuva podatke o sesiji, što omogućava da zahtjevi koji se šalju budu nezavisni jedan od drugog i olakšava skaliranje.

Za podršku radu sa bazama podataka zadužen je poseban *Spring* projekat nazvan ***Spring Data JPA***. On pojednostavljuje izvršavanje upita nad bazom podataka uvođenjem koncepta repozitorijuma. Repozitorijum predstavlja interfejs koji obuhvata osnovne operacije nad entitetom (CRUD), tako da korisnik ne mora sam da implementira te operacije, već samo navede entitet sa kojim radi. Pored toga, potrebna je i podrška za konekciju sa bazom podataka, odnosno JDBC *driver* (*Java Database Connectivity Driver*) [\[32\]](#), u ovom slučaju za *PostgreSQL*.

Za bezbjednost aplikacije koristi se ***Spring Security*** projekat. On presreće veb zahtjeve od strane klijenta i izvršava autentikaciju i autorizaciju. Uz pomoć JSON *web token*-a (JWT) [\[33\]](#) provjerava identitet klijenta i njegovo pravo pristupa određenim metodama. Takođe, ima ugrađenu podršku za skladištenje lozinki u šifrovanim obliku.

Da bi se *Spring Boot* aplikacija integrisala sa AWS koristi se ***Spring Cloud AWS***, koji predstavlja dio *Spring Cloud umbrella* projekta i pojednostavljuje korišćenje AWS servisa u *Spring* aplikaciji. Integriran je sa osnovnim servisima kao što su *CloudFormation*, EC2, S3, RDS, *ElastiCache*, SNS i dr. Da bi se koristio *Spring Cloud AWS* potrebno je konfigurisati odgovarajuće zavisnosti i podešiti kredencijale AWS

korisničkog naloga. Na slici 3.19 prikazani su *Spring Cloud AWS* moduli i servisi sa kojima su povezani [34].



Slika 3.19. *Spring Cloud AWS* moduli i servisi [34]

U ovom radu je korišćena integracija *Spring Boot* aplikacije sa RDS servisom, koja pruža dvije značajne prednosti. Ukoliko se koristi više zona dostupnosti i u jednoj od njih dođe do problema, konekcija se prekida i RDS-u je potrebno nekoliko sekundi da interna pređe na rad sa drugom zonom dostupnosti. U tom slučaju bio bi prijavljen izuzetak, ali *Spring Cloud* neće prijaviti izuzetak nego će pokušavati ponovo da uspostavi konekciju sve dok ne uspije. Druga prednost je mogućnost korišćenja *read* replika, odnosno označavanje *read only* transakcija koje će se proslijediti replikama i na taj način smanjiti opterećenje primarne baze.

3.10. Angular

Angular [13] je platforma za razvoj *single-page* veb aplikacija koja se oslanja na HTML (*Hypertext Markup Language*) [35], CSS (*Cascading Style Sheets*) [36] ili SCSS (*Syntactically Awesome Style Sheets*) [37], i *TypeScript* [38]. *Single-page* aplikacija učitava samo jednu veb stranicu i na osnovu interakcije sa korisnikom ažurira dijelove te stranice, što značajno ubrzava rad aplikacije. U ovom radu *Angular* je iskorišćen za razvoj klijentske strane aplikacije.

Arhitekturu *Angular* aplikacije čine komponente koje su organizovane u module. Aplikacija mora da sadrži korijenski modul, a mogu i da se kreiraju ili importuju novi moduli. Komponenta predstavlja osnovni gradivni blok

aplikacije i objedinjuje četiri fajla: HTML fajl koji definiše prikaz komponente, CSS ili SCSS fajl koji definiše stil HTML fajla, *TypeScript* fajl koji definiše logiku komponente i fajl koji služi za pisanje jediničnih testova komponente. Komponente su međusobno povezane i grade strukturu stabla. Pored njih, važnu ulogu imaju i servisi, kao klase koje sadrže logiku za komunikaciju sa udaljenim REST servisima. Oni su dostupni svim komponentama zahvaljujući *Dependency Injection* mehanizmu.

4. SPECIFIKACIJA SISTEMA

U radu je prikazan primjer jednostavne veb aplikacije koja zahtjeva elastičnost i visoku dostupnost. Aplikacija je namijenjena korisnicima koji žele da rezervišu karte za različite događaje (kulturne manifestacije, koncerte, sportske događaje) u okviru stadiona/arene. U trenutku kada se doda novi događaj u sistem, može se desiti da veliki broj korisnika odjednom krene da rezerviše kartu što će dovesti do velikog opterećenja, i u tom slučaju je potrebno obezbijediti više resursa i omogućiti visoku dostupnost. U nastavku je dat dijagram klase sistema, funkcionalnosti koje omogućava i prijedlog arhitekture koja može obezbijediti elastičnost i visoku dostupnost.

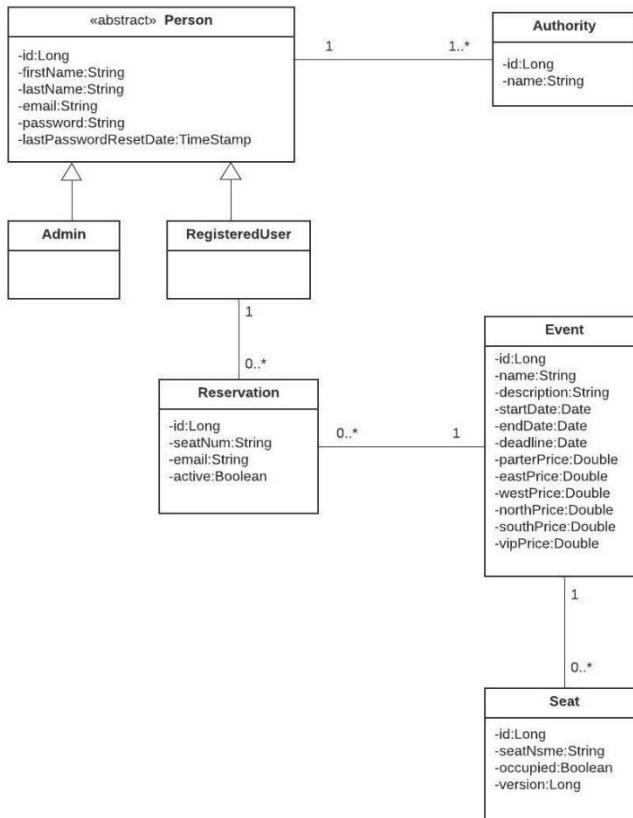
4.1. Dijagram klase

Model aplikacije „Arena“ sastoji se od sljedećih klasa (slika 4.1):

- *Person*
- *Admin*
- *RegisteredUser*
- *Authority*
- *Event*
- *Seat*
- *Reservation*

Klasa *Person* je apstraktna klasa koja sadrži osnovne podatke o korisniku sistema (ime i prezime) i podatke potrebne za prijavu na sistem (*email* i lozinka). Na osnovu šablona koji koristi *spring security*, svaki korisnik može biti vezan za jednu ili više uloga predstavljenih klasom *Authority*. U ovom sistemu postoje dvije uloge: administrator i obični korisnik. To je predstavljeno klasama *Admin* i *RegisteredUser* koje nasljeđuju klasu *Person*. Klasa *Event* predstavlja jedan događaj i sadrži sljedeće attribute: naziv događaja, opis događaja, datum i vrijeme početka i završetka održavanja, daum i vrijeme do kojeg je moguće rezervisati i otkazivati rezervacije i šest različitih cijena u zavisnosti od položaja sjedišta (parter, VIP loža, istočne, zapadne, sjeverne i južne tribine). Ukoliko vrijednost cijene nekog tipa sjedišta nije navedena, podrazumijeva se da se za konkretan događaj ne prodaju karte tog tipa sjedišta. Za svaki događaj vezana je lista sjedišta predstavljenih klasom *Seat*, sa podacima o nazivu sjedišta, da li je sjedište zauzeto ili nije, i verziji objekta u bazi podataka, što je značajno za održavanje konzistentnosti baze i o čemu će biti riječ u poglavljju o implementaciji. Obični korisnik ima listu svojih rezervacija predstavljenih klasom *Reservation*. Svaka rezervacija je vezana za jedan

događaj i sadrži naziv odabranog sjedišta, *email* korisnika koji je rezervisao sjedište i podatak da li je rezervacija trenutno aktivna, odnosno da li je otkazana ili nije.



Slika 4.1. Dijagram klasa

4.2. Dijagram slučajeva korišćenja

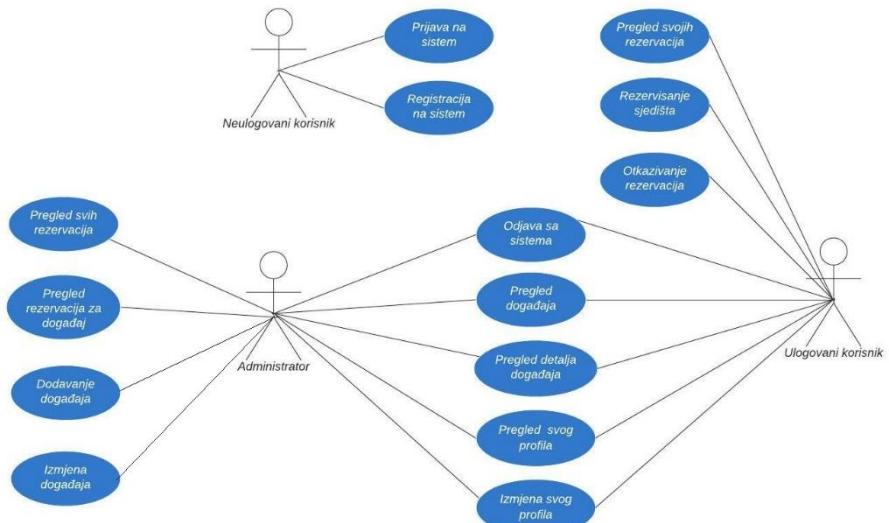
Aplikacija podržava tri tipa korisnika: neulogovani korisnik, obični korisnik i administrator. Na slici 4.2 prikazan je dijagram slučajeva korišćenja za svaki tip.

Neulogovani korisnik može da se registruje na sistem ili da se prijavi na sistem. Kada se prijavi dobija ulogu običnog korisnika ili administratora. Zajedničke funkcionalnosti ova dva tipa korisnika su: odjava sa sistema, pregled svog profila, izmjena svog profila (ime, prezime, lozinka), pregled svih događaja u sistemu sortiranih od najnovijeg ka najstarijem i detaljni

pregled svakog događaja. On uključuje vizuelni prikaz arene koji razlikuje zauzeta sjedišta, slobodna sjedišta i sjedišta zauzeta od strane ulogovanog korisnika.

Funkcionalnosti koje obavlja samo administrator su: pregled svih rezervacija u sistemu sortiranih od najnovije ka najstarijoj, pregled rezervacija za izabrani događaj, dodavanje novog događaja, izmjena postojećeg događaja. Pri dodavanju i izmjeni događaja administrator može da unese naziv, opis, cijene za različite tipove sjedišta, datum i vrijeme početka i završetka i datum i vrijeme do kojeg se mogu rezervisati sjedišta i otkazivati rezervacije. Sistem obaveštava administratora i ne dozvoljava izvršavanje operacije ako se uneseni datumi preklapaju sa datumima i vremenom nekog već postojećeg događaja ili ako nisu validni.

Obični korisnik ima mogućnost da pregleda svoje rezervacije, rezerviše sjedišta za neki događaj i otkazuje svoje rezervacije. Sistem vodi računa o tome da korisnik ne može da rezerviše sjedište koje je u međuvremenu neko drugi rezervisao, da ne može da rezerviše više od četiri sjedišta za jedan događaj i da ne može da rezerviše ili otkazuje rezervacije ukoliko je prošao datum do kojeg je to dozvoljeno. Pri uspešnoj rezervaciji ili otkazivanju korisniku se šalje *email* sa nazivom i vremenom događaja, nazivom sjedišta koje je rezervisao ili otkazao i njihovom cijenom.



Slika 4.2. Dijagram slučajeva korišćenja

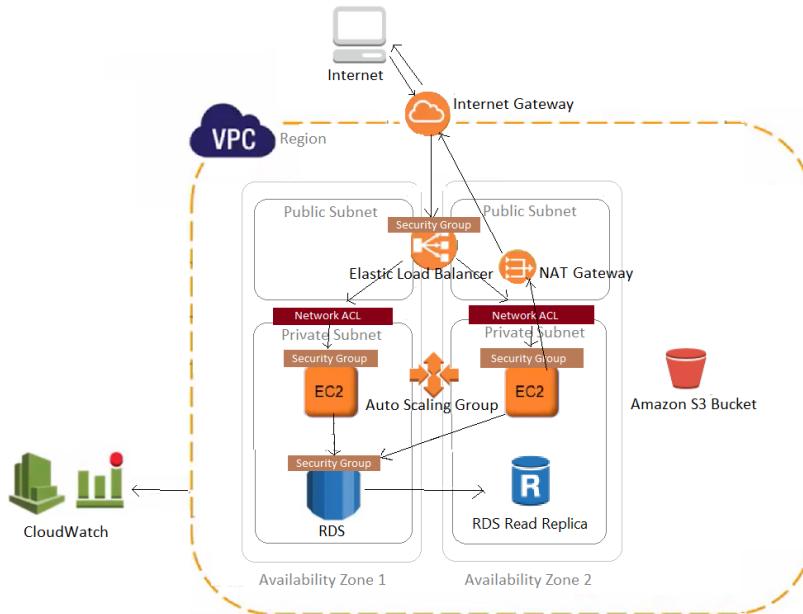
4.3. Arhitektura sistema

U ovom odjeljku opisana je predložena arhitektura sistema dizajnirana da omogući što veću elastičnost i dostupnost. Pošto neki od prethodno opisanih AWS servisa nisu besplatni, implementirana arhitektura nije optimalna, ali je dat teoretski opis naprednije arhitekture koja bi obezbijedila elastičnost i visoku dostupnost.

Sistem se sastoji od klijentske i serverske strane koje su postavljene na arhitekturu sastavljenu od AWS servisa. Klijentska strana je implementirana kao *Angular* aplikacija i postavljena kao statički veb sajt na S3 *bucket*. Serverska strana je implementirana kao *Spring Boot* veb aplikacija, i njen izvorni kod je takođe postavljen na S3 *bucket*, odakle ga preuzimaju i pokreću EC2 instance. Pristup izvornom kodu zaštićen je upotrebom IAM uloge, odnosno pristup je dozvoljen samo EC2 instancama kojima je dodijeljena odgovarajuća uloga.

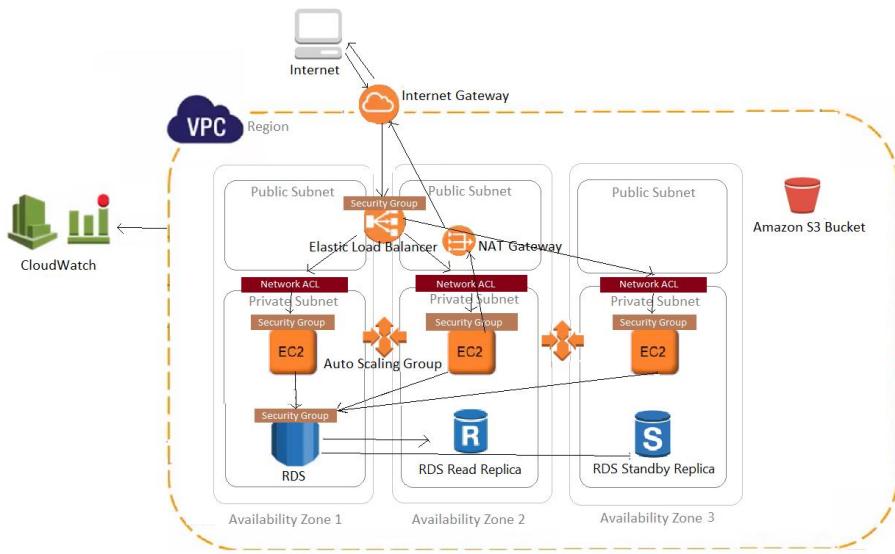
EC2 instance čine *Auto Scaling* grupu, čiji kapacitet se mijenja u zavisnosti od opterećenja, što obezbjeđuje elastičnost. Minimalan kapacitet je 1, a maksimalan 2, što je dovoljno za demonstraciju. Skaliranje se izvršava automatski, kao posljedica okidanja *CloudWatch* alarma koji prati opterećenje procesora *Auto Scaling* grupe. EC2 instance su raspoređene u dvije zone dostupnosti, što obezbjeđuje dostupnost jer će se u slučaju da dođe do otkaza u jednoj zoni pokrenuti nova instanca u drugoj zoni, a ako ona već postoji nastaviće da radi. Zahtjevi koji se šalju od klijenta ka serveru upućuju se *Application Load Balancer*-u (ALB), koji ih raspoređuje na EC2 instance *Auto Scaling* grupe. ALB je izabran iz razloga što radi sa HTTP zahtjevima i prilagođen je za veb aplikacije. Serveri šalju upite *PostgreSQL* bazi podataka pokrenutoj na RDS instanci. RDS instance ima *read* repliku kojoj se proslijeđuju svi upiti za čitanje i koja se nalazi u drugoj zoni dostupnosti. U slučaju otkaza prve zone, ona može ručno da se unapredi i da postane primarna baza.

Svi servisi su postavljeni u korisnički definisanom VPC-u u privatne ili javne podmreže. EC2 instance *Auto Scaling* grupe i RDS instance su postavljene u privatne podmreže, što znači da im se ne može pristupiti direktno sa interneta. Komunikacija u obrnutom smjeru je moguća, jer je u javnoj podmreži postavljen NAT *Gateway* koji obezbjeđujeinstancama pristup internetu i drugim servisima. Pristup im je neophodan zbog instalacije programskog jezika *Java* i preuzimanja izvornog koda pri pokretanju. Instance su zaštićene *security* grupama i ACL listom koje definišu da se bazi podataka može pristupiti samo preko EC2 instance, a EC2 instanci samo preko ALB-a. Dakle, ALB i NAT *Gateway* su jedini koji se nalaze u javnoj podmreži. Opisana arhitektura prikazana je na slici 4.3.



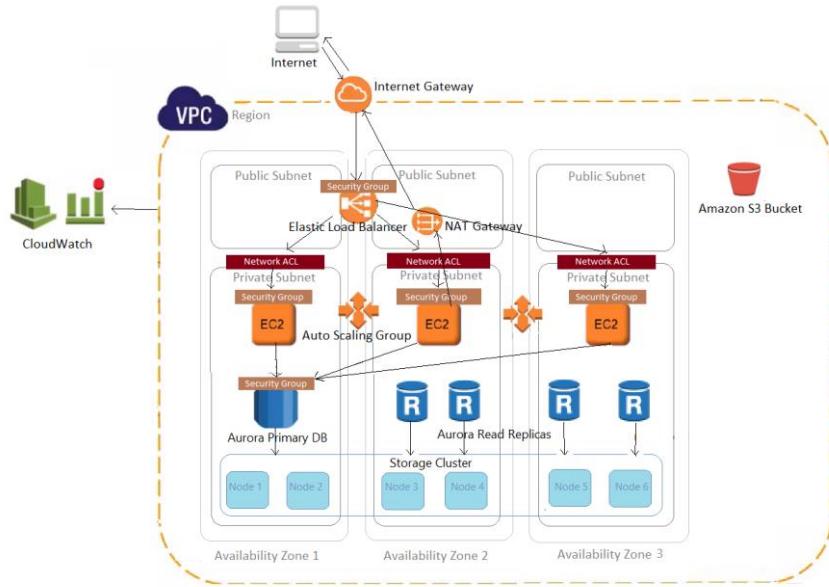
Slika 4.3. Implementirana arhitektura sistema

Mana ove arhitekture je to što nije obezbijeđena elastičnost i dostupnosti baze podataka. To bi se moglo omogućiti korišćenjem opcije *Multi-AZ Deployment* kod RDS instance. Ova opcija podrazumijeva automatsko kreiranje *standby* replike u nekoj drugoj zoni dostupnosti, koja bi služila kao rezervna kopija i u slučaju otkaza primarne baze preuzela bi njenu ulogu (slika 4.4).



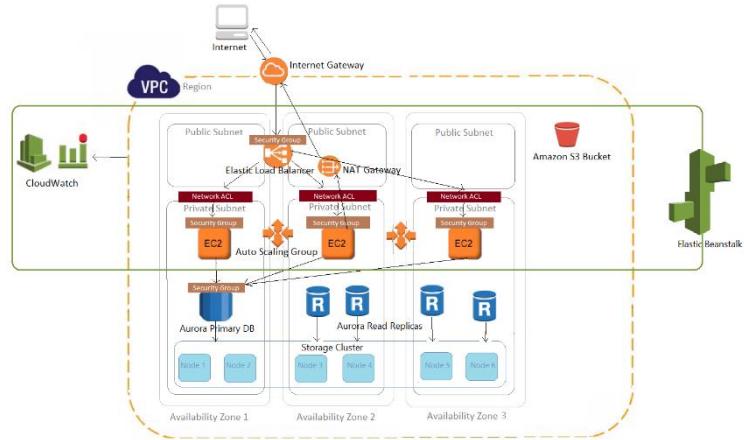
Slika 4.4. Arhitektura sistema sa RDS *multi-AZ*

Kako bi se obezbijedila i elastičnost može se koristiti *Amazon Aurora*, koja pruža mogućnost automatskog skaliranja *read* replika na osnovu praćenja opterećenja i definisane politike skaliranja (slika 4.5). *Amazon Aurora* nudi i prednosti vezane za performanse (v. odjeljak 3.5).



Slika 4.5. Arhitektura sistema sa *Amazon Aurora*

Navedena arhitektura može da se postigne i korišćenjem servisa *Elastic Beanstalk*, koji je namijenjen za postavljanje skalabilnih veb aplikacija i objedinjuje sve servise vezane za EC2 instance, rutiranje i skaliranje. Na slici 4.6 prikazano je koje servise može da zamjeni *Elastic Beanstalk*.



Slika 4.6. Arhitektura sistema sa *Elastic Beanstalk*

5. IMPLEMENTACIJA SISTEMA

U ovom poglavlju biće opisana implementacija aplikacije i koraci koji su izvršeni da bi se ona postavila na navedenu AWS arhitekturu. Aplikacija se sastoji od klijentske i serverske strane koje komuniciraju slanjem HTTP zahtjeva i odgovora. Kod aplikacije dostupan je na linku [\[39\]](#).

5.1. Klijentska strana

Za izradu klijentske strane korišćen je *Angular 11*. Funkcionalnosti i komponente su podijeljene na nekoliko modula koji su importovani u korijenski modul (*AppModule*):

- modul za rutiranje (*AppRoutingModule*);
- modul za navigaciju (*NavigationsModule*);
- modul za autentifikaciju (*AuthModule*);
- modul koji sadrži zajedničke komponente (*SharedModule*);
- modul za rad sa korisnicima (*UsersModule*);
- modul za rad sa događajima i rezervacijama (*EventsModule*).

Modul za rutiranje sadrži niz ruta, koje su definisane pomoću tri stavke: *path*, koja predstavlja URL (*Uniform Resource Locator*) prikaza stranice (engl. *view*), zatim *component*, koja predstavlja komponentu koja se učitava prilikom pristupa URL-u rute i *canActivate*, koja predstavlja *guard* rute. *Guard* je klasa koja implementira interfejs *CanActivate* sa istoimenom metodom i provjerava da li uloga korisnika ima pristup ruti. Ukoliko ima, vraća *true*, a ukoliko nema, preusmjerava korisnika na početnu stranicu i vraća *false*. Uloga korisnika je definisana kao enumeracija *UserRole* sa vrijednostima *UNAUTHORIZED*, *ROLE_ADMINISTRATOR* i *ROLE_USER*. Modul za navigaciju čini komponentu koja predstavlja meni aplikacije koji je vidljiv na svakoj stranici, odnosno *navigation bar*. U zavisnosti od uloge korisnika ona prikazuje različite opcije aktivirajući jednu od tri komponente: *NavigationAdminComponent*, *NavigationUserComponent*, *NavigationNotLoggedInComponent*.

Za prijavljivanje i registrovanje korisnika zadužen je modul za autentifikaciju, koji sadrži dvije komponente, *LogInComponent* i *SignUpComponent*. Prilikom prijavljivanja na sistem u *local storage*-u *browser*-a čuva se uloga korisnika, *email*, identifikator i *JWT (JSON web token)*. *JWT token* se zatim koristi pri slanju zahtjeva ka serveru, tako što klasa koja implementira interfejs *HttpInterceptor* presreće svaki zahtjev i ubacuje token u njegovo zaglavlje, kako bi se mogla izvršavati autentifikacija i autorizacija na serveru. Modul za rad sa korisnicima sadrži komponentu

ViewProfileComponent za prikaz i izmjenu osnovnih podataka ulogovanog korisnika, a modul za rad sa događajima sadrži četiri komponente: *CreateEventComponent* za dodavanje i izmjenu događaja, *EventComponent* za vizuelni prikaz arene i rezervisanje sjedišta za pojedinačni događaj, *ViewEventsComponent* za pregled događaja i *ViewReservationsComponent* za pregled rezervacija. Sve forme u aplikaciji implementirane su kao reaktivne forme koje su bazirane na modelu i kod kojih je većina logike implementirana u *TypeScript* fajlu, što daje mogućnost da se lakše implementira validacija svakog *FormControl*-a, odnosno polja u formi. Prilikom kreiranja forme pomoću *FormBuilder*-a svakom polju se proslijeđuju odgovarajući validatori. Validator predstavlja funkciju koja implementira interfejs *ValidatorFn* i vraća *null* ako korisnički unos ispunjava sve uslove, ili listu grešaka ako ih ne ispunjava. Pored osnovnog validatora koji zahtjeva da polje ne bude prazno, u formama koje traže unos lozinke koriste se i dva prilagođena validatora, koji zahtjevaju da lozinka ima bar 8 karaktera i da se lozinka i potvrda lozinke podudaraju. Navedene komponente se oslanjaju na tri zajedničke komponente: *TableComponent* za tabelarni prikaz podataka, *PaginationComponent* za paginaciju tabele i *YesNoPipe*, kao *pipe* funkciju koja transformiše prikaz *boolean* atributa.

Za slanje HTTP zahtjeva zadužene su servisne klase koje su injektovane u module pomoću *Dependency Injection* mehanizma, što znači da *Angular* okruženje vodi računa o njihovom životnom ciklusu i da su dostupni svim komponentama. Koriste servis *HttpClient*, koji šalje zahtjev serveru (*GET*, *POST*, *PUT* ili *DELETE*) i vraća *Observable* objekat. *Subscribe*-ovanjem na taj objekat dobija se odgovor od servera. API kojem su upućeni zahtjevi definisan je u fajlu *environments.prod.ts* kao globalna varijabla, i predstavlja *endpoint* definisan za *Load Balancer*, o čemu će biti riječi kasnije. Korišćenjem alata *Angular CLI* izvršava se instalacija svih potrebnih zavisnosti i modula komandom *npm install* i izgradnja aplikacije u produpcionom režimu komandom *ng build --prod*. Fajlovi koji se dobiju smještaju su u folder *dist*, a zatim se *upload*-uju na AWS [\[13\]](#).

5.2. Serverska strana

Serverska strana implementirana je pomoću *Spring Boot* tehnologije korišćenjem *Eclipse IDE* razvojnog okruženja sa *plugin*-om *Apache Maven* za izgradnju aplikacije [\[40\]](#). U fajlu *pom.xml* navedene su osnovne informacije o nazivu projekta, verziji izvornog koda, izlaznom formatu pri izgradnji aplikacije, bibliotekama i zavisnostima koji su potrebni za kompajliranje i izvršavanje. U fajlu *application.properties* navedena je konfiguracija aplikacije vezana za pristup bazi podataka i AWS nalogu i

slanje mejla. Aplikacija se pokreće na portu 8080. Ima troslojnu arhitekturu, sa prezentacionim slojem (*Presentation Layer*), slojem poslovne logike (*Business Logic Layer*) i slojem podataka (*Data Access Layer*).

Za bezbjednost aplikacije korišćen je *Spring Security*, koji vodi računa o autentikaciji korisnika i autorizaciji HTTP zahtjeva dobijenog od strane klijenta. U fajl *pom.xml* je uključena odgovarajuća zavisnost. Korisnik je predstavljen apstraktnom klasom *Person* i klasama *Admin* i *RegisteredUser* koje je nasljeđuju. Ove klase implementiraju interfejs *UserDetails*, koji specificira metodu za dobavljanje liste uloga korisnika. To znači da korisnik mora biti vezan za bar jednu ulogu predstavljenju klasom *Authority*. Takođe, korisnik ima *email* kao jedinstven identifikator i lozinku koja se hešira upotrebom klase *BCryptPasswordEncoder*. Ova klasa implementira interfejs *PasswordEncoder* sa metodama *encode* za šifrovanje i *match* za provjeru podudaranja lozinki. *Spring Security* koristi JWT da obezbijedi *stateless* komunikaciju. Prilikom prijave na sistem kreira se token sa informacijama o algoritmu, *email*-om, lozinkom, ulogom korisnika i potpisom. Prilikom obrade pristiglog zahtjeva, na osnovu *email*-a i lozinke izvršava se autentikacija korisnika, a na osnovu uloge izvršava se autorizacija, tj. provjera da li uloga ima pristup *endpoint*-u. Navedena podešavanja nalaze se u paketu *security* i u klasi *WebSecurityConfiguration* paketa *config*. U toj klasi su navedene i putanje kojima nije zaštićen pristup, a to su *endpoint*-i za prijavu i registraciju. U klasi *WebConfiguration* istog paketa obezbijedena je podrška za *Cross-Origin Resource Sharing* (CORS) za *endpoint* klijentske aplikacije, što znači da će *browser* dozvoliti pristup resursima sa drugog izvora, u ovom slučaju porta 8080.

Prezentacioni sloj implementiran je kao REST kontroler, što je označeno anotacijama *RestController* i *RequestMapping*, sa parametrima *value* koji predstavlja putanju i *produces* koji označava format resursa koji se šalje klijentu, u ovom slučaju JSON. Kontroleri sadrže odgovarajuće servisne klase kao atribute, injektovane pomoću anotacije *Autowired*, što znači da mogu da pozivaju njihove metode bez da vode računa o njihovom životnom ciklusu. Na listingu 5.1 prikazan je primjer metode kontrolera za rezervaciju koja je tipa *POST* i prima resurs u JSON format u okviru tijela zahtjeva. Resursi su objekati DTO klase (*Data Transfer Object*), koje predstavljaju posebne klase za slanje objekata, kako se ne bi slali perzistentni objekti modela. Anotacija *PreAuthorize* označava ulogu koja ima pristup datoj metodi. Kontroler vraća klijentu odgovor tipa *ResponseEntity* sa statusom *OK* ili *BadRequest*.

```

@PreAuthorize("hasRole('ROLE_USER')")
@RequestMapping(value = "/make-reservation", method =
RequestMethod.POST, consumes =
MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<Object> makeReservation(@RequestBody
ReservationDTO dto) {
    List<String> result =
reservationService.makeReservations(dto);
    if (result != null)
        return new ResponseEntity<>(result,
HttpStatus.OK);
    else
        return new ResponseEntity<>(
                "Reservation has failed. The
seats you marked have been occupied in the meantime.",
                HttpStatus.BAD_REQUEST);
}

```

Listing 5.1. Primjer metode kontrolera

Sloj poslovne logike čine servisne klase u kojima je implementirana logika sistema. Kao atribute sadrže injektovane druge servise i repozitorijumske interfejse. Metode servisa su označene anotacijom *Transactional* sa parametrom *readOnly* koji može da bude *true* ili *false*. Na listingu 5.2 prikazan je primjer logike provjere ispravnosti rezervacije, pri čemu se u slučaju nevalidnog datuma ili prevelikog broja rezervacija baca izuzetak *ReservationException*. Taj izuzetak će biti uhvaćen od strane globalnog *ExceptionHandler*-a koji će da presretne HTTP odgovor i podesi status *BadRequest* sa proslijedenom porukom.

```

if (e.getDeadline().before(new Date())) {
    throw new ReservationException("You cannot make a
reservation after " + sdf.format(e.getDeadline()));
}
if (reservationRepository.checkReserved(dto.getEmail(),
e.getId()) + dto.getSeatNums().size() > 4L) {
    throw new ReservationException("You cannot make more
than 4 reservations for one event.");
}

```

Listing 5.2. Primjer logike servisa

Poseban servis za slanje mejla korisniku pri uspješnoj rezervaciji ili otkazivanju koristi *JavaMailSender* interfejs i *smtp* protokol za slanje mejla. Metode ovog servisa označene su anotacijom *Async*, kako bi se mejlovi mogli slati asinhrono. U fajlu *application.properties* podešeni su parametri za *gmail*, što znači da parametar *host* ima vrijednost *smtp.gmail.com*, a parametar *port* vrijednost *587*.

Sloj za pristup bazi podataka implementiran je pomoću *Spring Data JPA*, koji omogućava rad sa relacionim bazama podataka, u ovom slučaju *PostgreSQL* bazom. JPA koristi objektno-relaciono mapiranje (ORM) između tabela baze podataka i entiteta modela aplikacije. Na listingu 5.3 prikazan je primjer entiteta *Seat* na osnovu kojeg se kreira tabela u bazi sa kolonama koje odgovaraju atributima. Kao identifikator se koristi kolona *id* čija vrijednost se određuje automatski pomoću *sequence* ili *identity* generatora. U ovoj klasi je takođe implementirano optimističko zaključavanje objekta [41] dodavanjem atributa *version*. Potrebno je voditi računa o verziji sjedišta jer se može desiti da jedan korisnik pročita podatke o trenutno slobodnom sjedištu iz baze, zatim neki drugi korisnik zauzme to sjedište, pa prvi korisnik pokuša takođe da ga zauzme. U tom slučaju trenutna verzija će se razlikovati od posljednje pročitane verzije i prvom korisniku će se prijaviti greška i onemogućiti akcija.

```

@Entity
@Table(name = "seats")
public class Seat implements Comparable<Seat> {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE,
generator = "row_generator")
    @SequenceGenerator(name = "row_generator", sequenceName
= "db_generator", allocationSize = 500)
    private Long id;

    @Column(name = "seat_name")
    private String seatName;

    @Column(name = "occupied")
    private Boolean occupied;

    @Version
    @Column(name = "version")
    private Long version;

```

Listing 5.3. Primjer klase modela

JPA koristi i *Entity Manager API* za slanje upita i transakcija bazi podataka, pomoću objektno orijentisanog upitnog jezika JPQL (*Java Persistent Query Language*). Primjer upita za traženje broja rezervacija jednog korisnika za jedan događaj dat je na listingu 5.4. Pisanje osnovnih upita je skraćeno upotrebon JPA repozitorijuma, pri čemu je dovoljno da se implementira repozitorijumski interfejs sa odgovarajućim entitetom da bi se koristile osnovne CRUD operacije. Da bi se aplikacija mogla koristiti, u skripti *import.sql* koja se izvršava pri pokretanju aplikacije definisani su po jedan predefinisani administrator i korisnik sa odgovarajućim ulogama.

```

@Query("SELECT COUNT(r.id) FROM Reservation r WHERE r.email =
?1 AND r.event.id = ?2 AND r.active = true")
Long checkReserved(String email, Long eventId);

```

Listing 5.4. Primjer JPQL upita

Pristup AWS servisima, odnosno RDS bazi podataka omogućen je upotrebom *Spring Cloud*-a. U fajlu *pom.xml* podešena je odgovarajuća zavisnost i verzija *Finchley.SR1*, a na listingu 5.5 data su podešavanja u fajlu *application.properties*. *AccessKey* i *secretKey* predstavljanju kredencijale za pristup AWS nalogu i oni mogu da se dobave preko AWS *Management Console*. Naveden je region koji se koristi, u ovom slučaju *us-east-2* ili *Ohio*. Parametar *stack.auto* je podešen na *false* jer se aplikacija ne pokreće preko servisa *CloudFormation*. Takođe, dodata su i dva podešavanja za logovanje koja govore sistemu da prikazuje samo logove tipa *error*. Druga grupa parametara namijenjena je konkretnoj instanci baze podataka, *database-arena*. Podešeni su podaci za pristup bazi (korisničko ime, lozinka i ime baze) sa vrijednostima koje su definisane preko AWS *Management Console*, atribut za podršku za rad sa *read* replikama koja omogućava da sve transakcije kod kojih je vrijednost parametra *readOnly* jednaka *true* budu preusmjerene na *read* repliku, i atribut za podršku za rad sa *batch* iskazima, odnosno mogućnost da se više upita (u ovom slučaju 500) nad bazom izvršava odjednom kako bi se poboljšale performanse.

```
cloud.aws.credentials.accessKey=AKIAUG5NI6YAQSA45
cloud.aws.credentials.secretKey=xV8WyGYP3inIBZd6u0R3wlMF4joD1Y
MnepcxG
cloud.aws.credentials.instanceProfile=true
cloud.aws.region.static=us-east-2
cloud.aws.stack.auto=false
logging.level.com.amazonaws.util.EC2MetadataUtils=error
logging.level.com.amazonaws.internal.InstanceMetadataServiceRe
sourceFetcher=error

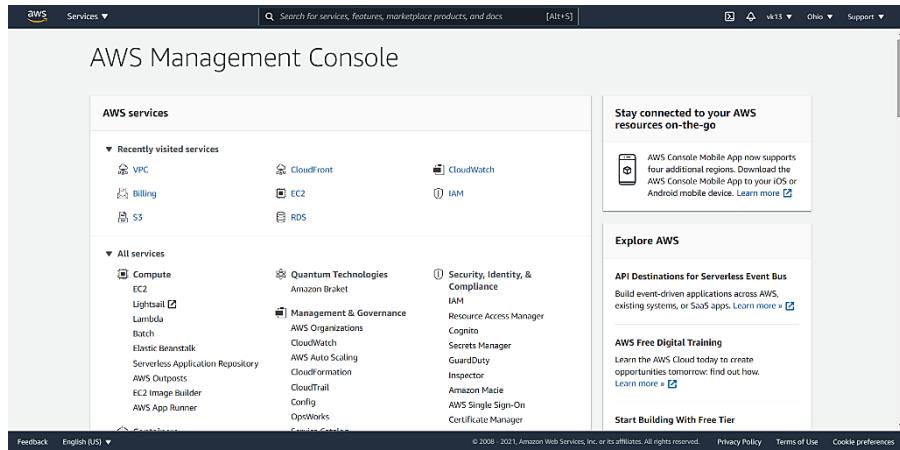
cloud.aws.rds.database-arena.username=postgres
cloud.aws.rds.database-arena.password=postgres
cloud.aws.rds.database-arena.databaseName=postgres
cloud.aws.rds.database-arena.readReplicaSupport=true
cloud.aws.rds.database-arena.rewriteBatchedStatements=true
```

Listing 5.5. Podešavanja za pristup AWS i RDS

Izvršavanjem maven komandi *clean compile install* u folderu *target* dobija se fajl *spring-boot-aws-arena.jar*, koji predstavlja zapakovan kod aplikacije i *upload-uje* se na AWS [\[12\]](#).

5.3. AWS

Za izgradnju arhitekture na *cloud-u* pomoću AWS servisa korišćen je interfejs *AWS Management Console*. Kreiran je nalog i odabran je region *Ohio (us-east-2)*, koji je i ponuđen kao podrazumijevani (slika 5.1). On nudi tri zone dostupnosti, *us-east-2a*, *us-east-2b* i *us-east-2c*. Preko *AWS Management Console* omogućeno je podešavanje i pokretanje svih servisa, pregled dokumentacije i pregled naplate korišćenih servisa.



Slika 5.1. AWS Management Console

Podešavanja vezana za mrežu odradena su preko VPC servisa. Kreiran je novi ARENA_VPC i dodijeljen mu je opseg IP adresa 10.0.0.0/16 iz IPv4 *pool-a*, što znači da može da koristi 65536 adresa. Pridružena mu je podrazumijevana tabela rutiranja i ACL. Atribut *Tenancy* ima vrijednost *default*, što znači da se koristi zajednički hardver, a ne poseban hardver za pojedinačni nalog. Parametri vezani za DNS *hostname*, odnosno jedinstveno ime koje odgovara jednoj IP adresi, podešeni su na podrazumijevane vrijednosti (slika 5.2). Kreiran je i *Internet Gateway* ARENA_IGW i pridružen ARENA_VPC-u.

vpc-0746842817b96852a / ARENA_VPC				
Details		Info		
VPC ID vpc-0746842817b96852a	State Available	DNS hostnames Enabled	DNS resolution Enabled	
Tenancy Default	DHCP options set dopt-e2305b89	Main route table rtb-0c9929395ee452614 / ARENA_MAIN_RTB	Main network ACL acl-0ef83c90d62eb53c0	
Default VPC No	IPv4 CIDR 10.0.0.0/16	IPv6 pool -	IPv6 CIDR -	
Route 53 Resolver DNS Firewall rule groups -	Owner ID 289736816129			

Slika 5.2. Podešavanja VPC-a

U okviru ARENA_VPC-a kreirane su četiri podmreže (slika 5.3), dvije javne i dvije privatne. Javne podmreže zauzimaju opsege IP adresa 10.0.1.0/24 i 10.0.2.0/24, a privatne 10.0.3.0/24 i 10.0.4.0/24. Maska vrijednosti 24 označava da podmreža ima 248 adresa na raspolaganju. Dvije podmreže od kojih je jedna javna i jedna privatna smještene su u zonu dostupnosti *us-east-2a*, a druge dvije u *us-east-2b*.

Subnets (4) Info						
<input type="button" value="C"/> Actions ▾						
Filter subnets <input type="text" value="VPC: vpc-0746842817b96852a"/> <input type="button" value="Clear filters"/>						
	Name	Subnet ID	State	VPC	IPv4 CIDR	
<input type="checkbox"/>	10.0.4.0_ARENA_PRIVATE2	subnet-0f755dc0bb0627445	Available	vpc-0746842817b96852a AR...	10.0.4.0/24	
<input type="checkbox"/>	10.0.1.0_ARENA_PUBLIC2	subnet-0ee8ac4e160b7f67f	Available	vpc-0746842817b96852a AR...	10.0.2.0/24	
<input type="checkbox"/>	10.0.1.0_ARENA_PUBLIC1	subnet-0423308e4a0c74bcd	Available	vpc-0746842817b96852a AR...	10.0.1.0/24	
<input type="checkbox"/>	10.0.3.0_ARENA_PRIVATE1	subnet-09e8df0bc4be31e28	Available	vpc-0746842817b96852a AR...	10.0.3.0/24	

Slika 5.3. Podmreže u ARENA_VPC-u

Kako bi se obezbijedio pristup internetu, alocirana je *Elastic IP* adresa 18.116.161.139, a zatim je kreiran NAT *Gateway* NGW_ARENA (slika 5.4) sa tom adresom. Smješten je u jednu od javnih podmreža na adresu 10.0.2.196 i podešen je kao *public*, što znači da ima pristup internetu, a ne samo drugim VPC-ovima.

NAT gateway ID nat-0286dde81f7ba8143	Connectivity type Public	State Available	State message –
Elastic IP address 18.116.161.139	Private IP address 10.0.2.196	Network interface ID eni-0e27e5aecd2658cbe	VPC vpc-0746842817b96852a / ARENA_VPC
Subnet subnet-0ee8ac4e8160b7f67f / 10.0.1.0_ARENA_PUBLIC2	Created 2021/08/26 12:38 GMT+2	Deleted –	

Slika 5.4. Podešavanja NAT Gateway-a

Privatnim podmrežama pridružena je tabela rutiranja ARENA_MAIN_RTB koja sadrži dvije rute. Prva ruta je namijenjena lokalnom saobraćaju, pa je parametar *Destination* podešen kao vrijednost opsega IP adresa VPC-a, a parametar *Target* na vrijednost *local*, što znači da je omogućen saobraćaj namijenjen destinacijama unutar VPC-a. Druga ruta je namijenjena saobraćaju sa internetom, pa je parametar *Destination* podešen na 0.0.0.0/0, a vrijednost parametra *Target* je kreirani NAT Gateway (slika 5.5). Javnim podmrežama pridružena je druga tabela rutiranja, ARENA_RTB, koja upućuje saobraćaj namijenjen internetu na kreirani *Internet Gateway* (slika 5.6).

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	nat-0286dde81f7ba8143

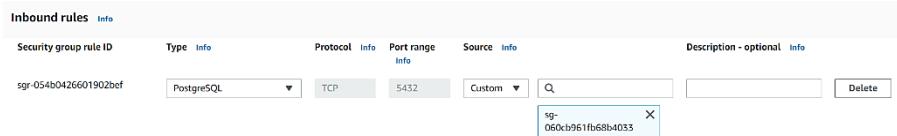
Slika 5.5. Tabela rutiranja za privatne podmreže

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	igw-062537ee6214736f7

Slika 5.6. Tabela rutiranja za javne podmreže

Posljednje podešavanje u okviru VPC servisa je kreiranje tri *security* grupe, ARENA_DB_SG koja se pridružuje bazi podataka, ARENA_SERVER_SG koja se pridružuje EC2 instancama i ARENA_ELB_SG koja se pridružuje *Elastic Load Balancer*-u. Za prvu grupu navedeno je pravilo koje dozvoljava da se prima isključivo *PostgreSQL* saobraćaj na portu 5432 koji dolazi iz druge *security* grupe, što znači da samo EC2 instance sa tom grupom mogu da pristupe bazi (slika 5.7). Slično, druga grupa je podešena tako da prima isključivo TCP saobraćaj na

portu 8080 koji dolazi iz treće grupe, odnosno od ELB-a (slika 5.8). Treća grupa je podešena da dozvoljava spoljašnji pristup sa interneta.

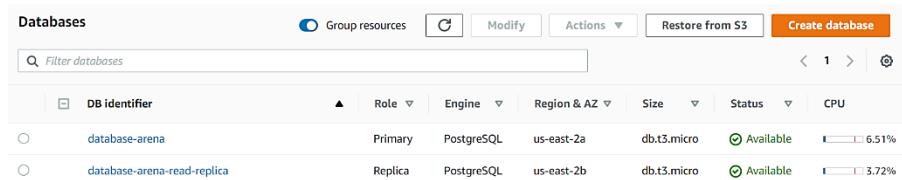


Slika 5.7. Pravilo *security* grupe za bazu podataka



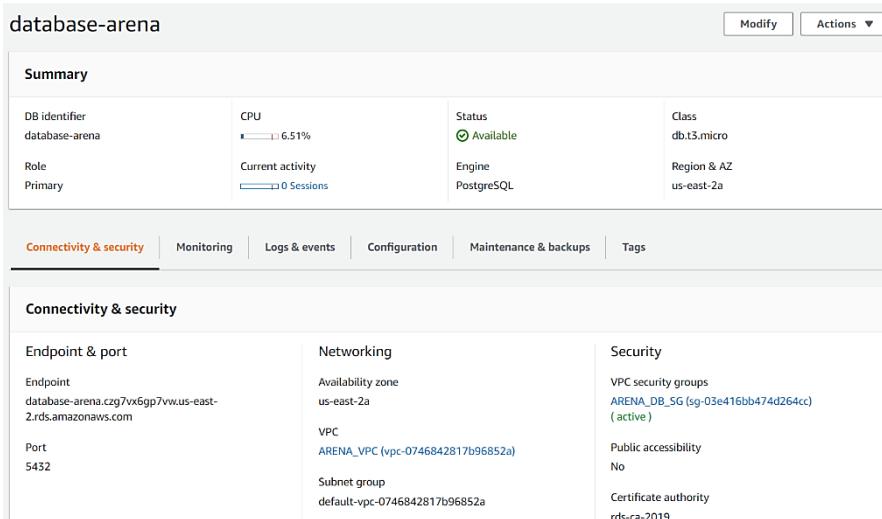
Slika 5.8. Pravilo *security* grupe za EC2 instance

Nakon VPC servisa, izvršena su podešavanja baze podataka pomoću RDS servisa. Pokrenuta je primarna baza podataka *database-arena* u zoni *us-east-2a* i njena *read* replika *database-arena-read-replica* u zoni *us-east-2b* (slika 5.9).



Slika 5.9. RDS instance

Izabrana je *PostgreSQL 13.3-R1* baza podataka i postavljeni su parametri za korisničko ime, lozinku i ime baze koji odgovaraju parametrima navedenim u *Spring* aplikaciji. Izabrana je klasa *db.t3.micro* iz grupe *Burstable classes*, sa dva procesora, 1GB RAM memorije i brzinom protoka 2,085Mbps. Alocirano je 20GB skladišta tipa *General Purpose SSD*, i dozvoljena je mogućnost automatskog skaliranja skladišta po potrebi. Instanci je dodijeljena ranije pomenuta *security* grupa ARENA_DB SG i izabrana je opcija da instanca ne bude javno dostupna (slika 5.10).



Slika 5.10. Podešavanja RDS instance

Pomoću EC2 servisa podešen je ELB i *Auto Scaling* grupa, a pomoću *CloudWatch* servisa podešeni su alarmi za datu grupu. Za ELB je izabran *Application Load Balancer* i dodijeljene su mu dvije zone dostupnosti, *us-east-2a* i *us-east-2b*, i *security* grupa ARENA_ELB_SG. Takođe, automatski mu je dodijeljen ARN (*Amazon Resource Identifier*) i DNS *name*, odnosno *endpoint* preko kojeg mu se pristupa (*arenaelb-1138954032.us-east-2.elb.amazonaws.com*) (slika 5.11).

Load balancer: ARENAELB

Description **Listeners** **Monitoring** **Integrated services** **Tags**

Basic Configuration

Name	ARENAELB
ARN	arn:aws:elasticloadbalancing:us-east-2:289736816129:loadbalancer/app/ARENAELB/f39e0246afccb948
DNS name	ARENAELB-1138954032.us-east-2.elb.amazonaws.com (A Record)
State	Active
Type	application
Scheme	internet-facing
IP address type	ipv4
Edit IP address type	
VPC	vpc-0746842817b96852a
Availability Zones	subnet-0423308e4a0c74bcd - us-east-2a IPv4 address: Assigned by AWS
	subnet-0ee8ac4e160b7f67f - us-east-2b IPv4 address: Assigned by AWS
Edit subnets	
Hosted zone	Z3AADJGX6KTTL2
Creation time	August 26, 2021 at 3:10:07 PM UTC+2

Security

Security groups	sg-07311fc099df0c8ec, ARENA_ELB_SG • elb security group
Edit security groups	

Slika 5.11. Podešavanja ELB-a

ELB raspoređuje saobraćaj na *target* grupe EC2 instanci, pa je potrebno podesiti *target* grupu koja će biti vezana sa *Auto Scaling* grupom. U ovom slučaju nema potrebe za kreiranjem više *target* grupe, jer se svi HTTP zahtjevi ravnomjerno raspoređuju na EC2 instance. Prilikom pokretanja instance u *target* grupi izvršava se provjera ispravnosti instance, odnosno *health check*. Ona podrazumijeva testiranje jednog izabranog *endpoint*-a aplikacije pokrenute na instanci. Ako je status odgovora 200, instance je ispravna. Na slici 5.12 prikazana je provjera definisana za *Arena* aplikaciju. Preko protokola HTTP na portu 8080 gađa se putanja /test koja poziva metodu posebnog kontrolera namijenjenog za testiranje i dostupnog neulogovanim korisnicima i vraća listu svih objekata klase *Event* u sistemu. Provjera se izvršava na svakih 11 sekundi (*Interval*). Ukoliko nakon 10 sekundi ne stigne odgovor (*Timeout*) ili se status odgovora razlikuje od 200 (*Success Codes*), instance ne prolazi provjeru. Kada instance uspješno prođe provjeru 2 puta smatra se ispravnom, a kada ne prođe provjeru 2 puta smatra

se neispravnom (*Healthy/Unhealthy threshold*). Na slici 5.13 prikazana je *target* grupa ARENAELB sa jednom ispravnom instancom.

Description	Targets	Health checks	Monitoring	Tags
		Protocol ⓘ HTTP		
		Path ⓘ /test		
		Port ⓘ 8080		
		Healthy threshold ⓘ 2		
		Unhealthy threshold ⓘ 2		
		Timeout ⓘ 10		
		Interval ⓘ 11		
		Success codes ⓘ 200		

Slika 5.12. *Health check instance target* grupe

Registered targets					
Instance ID	Name	Port	Availability Zone	Status	Description
i-037c330b71c81d993		8080	us-east-2b	healthy	This target is currently passing target group's health checks.
Availability Zones					
Availability Zone		Target count		Healthy?	
us-east-2b		1		Yes	

Slika 5.13. Primjer *target* grupe

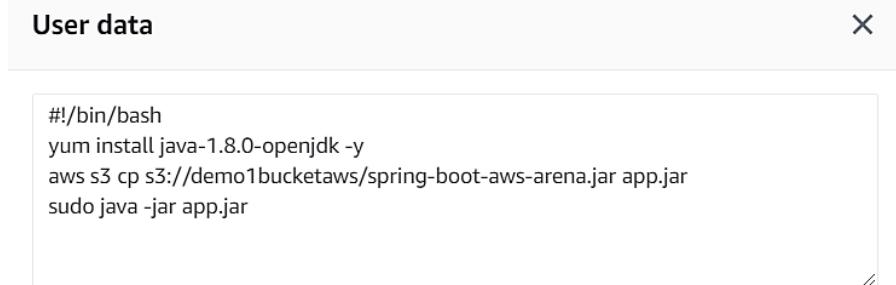
Sljedeći korak je kreiranje *Auto Scaling* grupe, a da bi se ona kreirala potrebno je definisati *launch configuration*, odnosno konfiguraciju za pokretanje pojedinačnih instanci. Definisana konfiguracija ARENA_LC specifičira *t2.micro* tip instance sa jednim procesorom, 1GB RAM memorije i EBS skladištem. AMI koji je odabran je *Amazon Linux 2 AMI 2.0.20210617.0 x86_64 HVM gp2*. Navedeni resursi su odbrani iz razloga što su pogodni za pokretanje veb aplikacije i što spadaju u *Free Tier*. Instanci je dodijeljen par ključeva (privatni i javni) koji omogućava konekciju preko SSH protokola, ali je pridruživanjem *security* grupe ARENA_SERVER_SG

zabranjen bilo kakav pristup instanci osim preko ELB-a pridruženog njenoj *target* grupi (slika 5.14).

AMI ID ami-0277b52859bac6f4b	Instance type t2.micro	IAM instance profile arn:aws:iam::289736816129:instance-profile/DownloadS3Role
Kernel ID -	Key name NOVI	Monitoring false
EBS optimized false	Security groups sg-060cb961fb68b4033	Spot price -
Create time Thu Aug 26 2021 15:13:20 GMT+0200 (Central European Summer Time)	RAM disk ID -	IP address type Default

Slika 5.14. Konfiguracija za pokretanje instance

Da bi se pokrenula aplikacija na instanci potrebno je da se preuzme kod aplikacije. To je odrđeno preko S3 i IAM servisa. Na S3 servisu kreiran je *bucket demo1bucketaws* na koji je *upload*-ovan *jar* fajl koji sadrži zapakovan kod *Spring* aplikacije. Pošto su resursi u S3 *bucket*-ima zaštićeni, da bi instanca mogla da pristupi *jar* fajlu mora da joj se dodijeli uloga sa odgovarajućim pravom pristupa. Pomoću IAM servisa kreirana je uloga *DownloadsS3Role* koja ima pravo da čita resurse S3 *bucket*-a, i pridružena je instanci, što se može vidjeti u konfiguraciji. Konfiguracija daje mogućnost definisanja skripte koja se izvršava pri pokretanju svake instance u polju *User data*. Pošto izabrani AMI nema instaliran *java* programski jezik, skripta prvo izvršava preuzimanje i instalaciju *java* jezika a zatim preuzimanje i pokretanje *Spring* aplikacije (slika 5.15).



```
#!/bin/bash
yum install java-1.8.0-openjdk -y
aws s3 cp s3://demo1bucketaws/spring-boot-aws-arena.jar app.jar
sudo java -jar app.jar
```

Slika 5.15. Kod koji se izvršava pri pokretanju EC2 instance

Nakon definisanja konfiguracije omogućeno je i definisanje *Auto Scaling* grupe. Kreiranoj grupi pridružena je konfiguracija ARENA_LC,

Load Balancer Target grupa ARENAELB, dvije zone dostupnosti, *us-east-2a* i *us-east-2b*, i dvije privatne podmreže, što znači da će se njene instance pokretati u navedenim zonama i podmrežama. Takođe, odabran je minimalan i poželjan kapacitet od 1 instance, i maksimalan kapacitet od 2 instance (slika 5.16).

Group details	
Desired capacity	Auto Scaling group name
1	DEMO1_ASG
Minimum capacity	Date created
1	Sat Jun 26 2021 13:33:50 GMT+0200 (Central European Summer Time)
Maximum capacity	Amazon Resource Name (ARN)
2	arn:aws:autoscaling:us-east-2:289736816129:autoScalingGroup:7b09881b-2b64-4dbb-8eb9-740647f60641:autoScalingGroupName/DEMO1_ASG

Slika 5.16.

Podešavanja *Auto Scaling* grupe

Kako bi se dinamički mijenjao kapacitet grupe, definisana je dinamička politika skaliranja tipa *Target Tracking Policy* koja koristi metriku *Average CPU utilization*, odnosno prosječno iskorišćenje procesora. Iskorišćenje procesora predstavlja procenat alociranih EC2 *compute unit*-a koje instanca trenutno koristi, a EC2 *compute unit* je standardna mjera za kapacitet procesora koja odgovara kapacitetu procesora *1.0-1.2 GHz 2007 Intel Xeon*. Cilj koji treba da se postigne je da vrijednost metrike kod svake instance bude što bliže 50% (slika 5.17).

Create dynamic scaling policy

Policy type

Target tracking scaling

Scaling policy name

Target Tracking Policy

Metric type

Average CPU utilization

Target value

50

Instances need

300

 seconds warm up before including in metric

Disable scale in to create only a scale-out policy

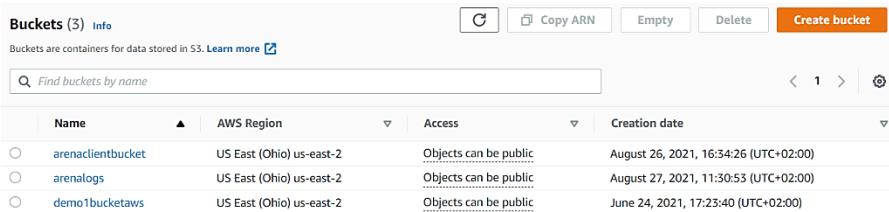
Slika 5.17. Politika skaliranja Auto Scaling grupe

Na osnovu definisane politike skaliranja automatski su kreirana dva alarma. Prvi alarm izvršava akciju *scale out*, odnosno dodaje instancu, kada vrijednost iskorišćenja procesora prelazi 50 za 3 uzastopna *datapoint-a* u 3 minute. To znači da se validacija alarma izvršava u periodu od 1 minute. Drugi alarm izvršava akciju *scale down*, odnosno terminira instancu, kada je vrijednost metrike ispod 35 za 15 *datapoint-a* u 15 minuta (slika 5.18).

Alarms (2)						<input type="checkbox"/> Hide Auto Scaling alarms	<input type="checkbox"/> Clear selection	<input type="checkbox"/>	Create composite alarm	Actions	<input type="button" value="Create alarm"/>
	Name	State	Last state update	Conditions	Actions						
<input type="checkbox"/>	TargetTracking-DEMO1_ASG-AlarmHigh-64d2b294-f127-4da8-8fc7-2f36cafd81b	OK	2021-08-26 23:57:02	CPUUtilization > 50 for 3 datapoints within 3 minutes	Actions enabled						
<input type="checkbox"/>	TargetTracking-DEMO1_ASG-AlarmLow-cc023728-71bb-49e6-a8d7-c8a2957e8bac	OK	2021-08-26 23:56:53	CPUUtilization < 35 for 15 datapoints within 15 minutes	Actions enabled						

Slika 5.18. Alarmi definisani na osnovu politike skaliranja

Klijentska strana aplikacije pokrenuta je pomoću S3 servisa. Pored *bucket-a* u kojem se čuva *Spring* kod i *bucket-a* u kojem se čuvaju log fajlovi sa podacima o saobraćaju u VPC-u, kreiran je *bucket arenaclientbucket* u koji se smješta *Angular* kod, osnosno fajlovi iz foldera *dist* nakon pokretanja komande *ng build -prod* (slika 5.19). Da bi ovi resursi bili dostupni u *browser-u* potrebno je izabrati opciju *make public* za sve fajlove u *bucket-u*. Takođe, potrebno je izabrati opciju za omogućavanje hostovanja statičke veb aplikacije u podešavanjima *bucket-a*. Nakon toga aplikaciji će biti dodijeljen *endpoint* na osnovu kojeg joj se može pristupiti u *browser-u*.

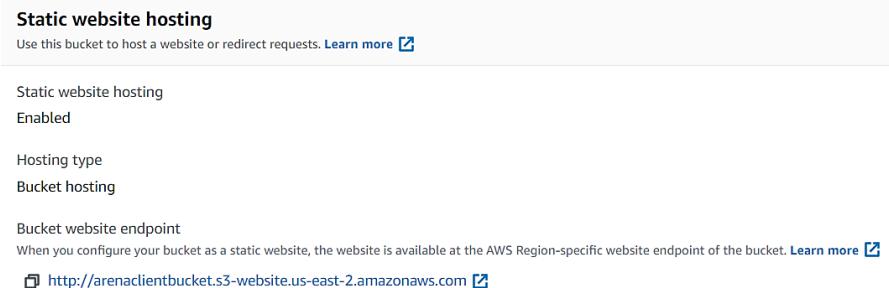


The screenshot shows the AWS S3 Buckets list interface. At the top, there are buttons for 'Create bucket' (orange), 'Empty', and 'Delete'. Below that is a search bar with placeholder 'Find buckets by name'. A table lists three buckets:

Name	AWS Region	Access	Creation date
arenaclientbucket	US East (Ohio) us-east-2	Objects can be public	August 26, 2021, 16:34:26 (UTC+02:00)
arenalog	US East (Ohio) us-east-2	Objects can be public	August 27, 2021, 11:30:53 (UTC+02:00)
demo1bucketaws	US East (Ohio) us-east-2	Objects can be public	June 24, 2021, 17:23:40 (UTC+02:00)

Slika 5.19.

S3 *bucket-i* sa *Spring* i *Angular* kodovima i log fajlovima



The screenshot shows the 'Static website hosting' configuration for the 'arenaclientbucket'. It includes sections for 'Static website hosting' (Enabled), 'Hosting type' (Bucket hosting), and 'Bucket website endpoint' (http://arenaclientbucket.s3-website.us-east-2.amazonaws.com). A note at the bottom says: 'When you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket.'

Slika 5.20.

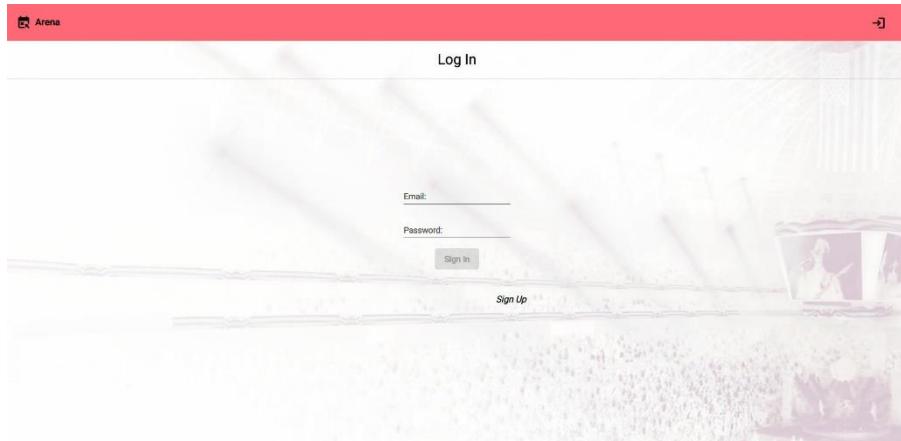
Hostovanje statičkog veb sajta

6. PRIKAZ IMPLEMENTIRANOG SISTEMA

U ovom poglavlju biće opisane i prikazane funkcionalnosti sistema iz ugla administratora i običnog korisnika. Zatim će biti demonstrirano kako sistem postiže elastičnost i dostupnost.

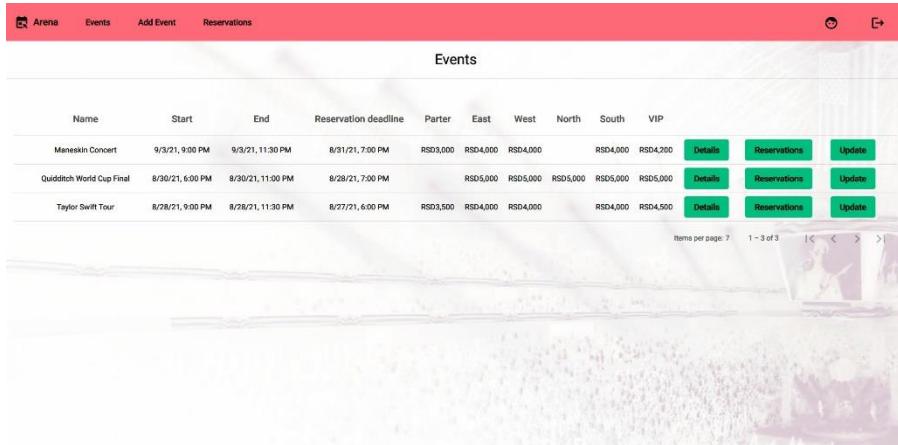
6.1. Prikaz funkcionalnosti sistema

Prilikom pristupa aplikaciji svaki korisnik ima ulogu neprijavljenog korisnika. Dostupna mu je stranica za prijavu (slika 6.1) i stranica za registraciju klikom na dugme *Sign Up*. Da bi se prijavio potrebno je da je prethodno registrovan i da unese svoj *email* i lozinku sa kojima se registrovao. Da bi se registrovao potrebno je da unese *email* koji mora biti jedinstven u sistemu (korisnik ne može da pravi više naloga sa istim *email*-om), ime, prezime, lozinku sa najmanje 8 karaktera i potvrdu lozinke. U sistemu postoji jedan predefinisani administrator, a svaki novi korisnik koji se registruje dobija ulogu običnog korisnika.



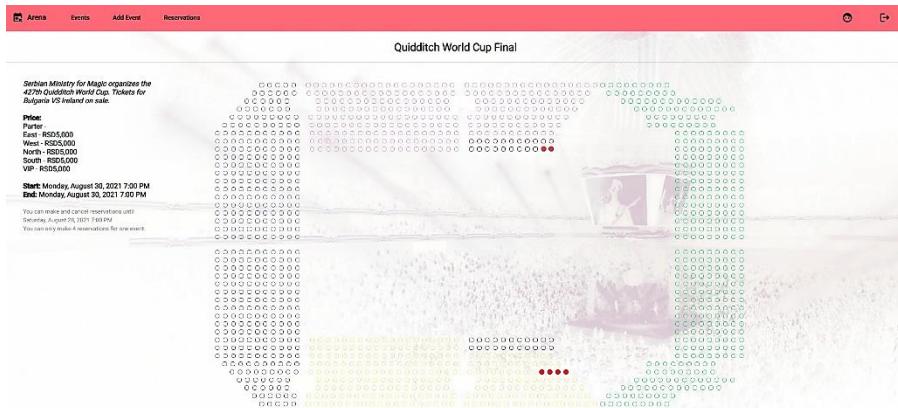
Slika 6.1. Prikaz stranice za prijavu

Kada se korisnik prijavi kao administrator, može da vidi početnu stranicu sa tabelom u kojoj su prikazani svi događaji u sistemu (slika 6.2). Za svaki događaj prikazan je naziv, datum i vrijeme početka, datum i vrijeme završetka, datum i vrijeme do kojeg korisnici mogu rezervisati i otkazivati sjedišta i cijene za različite vrste sjedišta (*parter, east, west, north, south, VIP*). Administrator ima opcije da pogleda detaljan prikaz događaja, sve rezervacije za taj događaj i formu za izmjenu podataka o događaju.



Slika 6.2. Prikaz početne stranice „Events“ za administratora

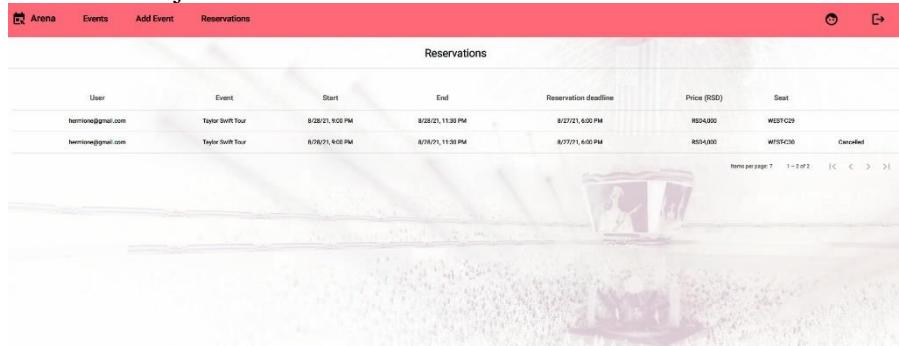
Izborom opcije za detaljan prikaz događaja administrator pristupa stranici koja pored osnovnih podataka o događaju sadrži opis događaja i vizuelni prikaz rasporeda sjedišta arene. Sjedišta različite vrste, odnosno različite cijene, prikazana su različitom bojom. Sjedišta koja su rezervisana označena su crvenom bojom (slika 6.3).



Slika 6.3. Detaljan prikaz događaja za administratora

Pored opcija vezanih za događaj, administrator može da dodaje novi događaj i pregleda sve rezervacije u sistemu, a te opcije se nalaze u *navigation bar*-u. Stranica za pregledanje svih rezervacija i rezervacija za događaj prikazana je na slici 6.4. Za svaku rezervaciju može da se vidi

korisnik koji je rezervisao, naziv događaja, datumi početka, završetka i roka, cijena, broj sjedišta i u slučaju da je rezervacija otkazana, dodatna kolona sa tom informacijom.



User	Event	Start	End	Reservation deadline	Price (RSD)	Seat
herman@gmail.com	Taylor Swift Tour	8/28/21, 9:00 PM	8/28/21, 11:30 PM	8/27/21, 6:00 PM	RSD4000	WESTC29
herman@gmail.com	Taylor Swift Tour	8/28/21, 9:00 PM	8/28/21, 11:30 PM	8/27/21, 6:00 PM	RSD4000	WESTC26

Slika 6.4. Prikaz stranice „Reservations“ za administratora

Za dodavanje novog i izmjenu postojećeg događaja koristi se ista forma (slika 6.5). Administrator mora da popuni sva polja osim onih koja označavaju cijenu. Ukoliko ona ostanu nepotpunjena podrazumijeva se da se ne izdaju rezervacije za tu vrstu sjedišta. Takođe, administrator mora da vodi računa da datumi i vremena budu uređeni tako da imaju smisla i da se događaji vremenski ne poklapaju. To se provjerava na serverskoj strani, i ukoliko uslov nije ispunjen administratoru se prikazuje odgovarajuće obavještenje.



Name:
Maneskin Concert

Description:
Eurovision winners for the first time in Serbia!

Partner price(RSD): 3000	West price(RSD): 4000	East price(RSD): 4000	South price(RSD): 4000	VIP price(RSD): 4200
------------------------------------	---------------------------------	---------------------------------	----------------------------------	--------------------------------

Choose start time:
8/28/2021, 9:00:00 PM

Choose end time:
8/28/2021, 11:00:00 PM

Choose deadline for reservation:
8/27/2021, 7:04:00 AM

Finish

Slika 6.5. Prikaz forme za dodavanje i izmjenu događaja

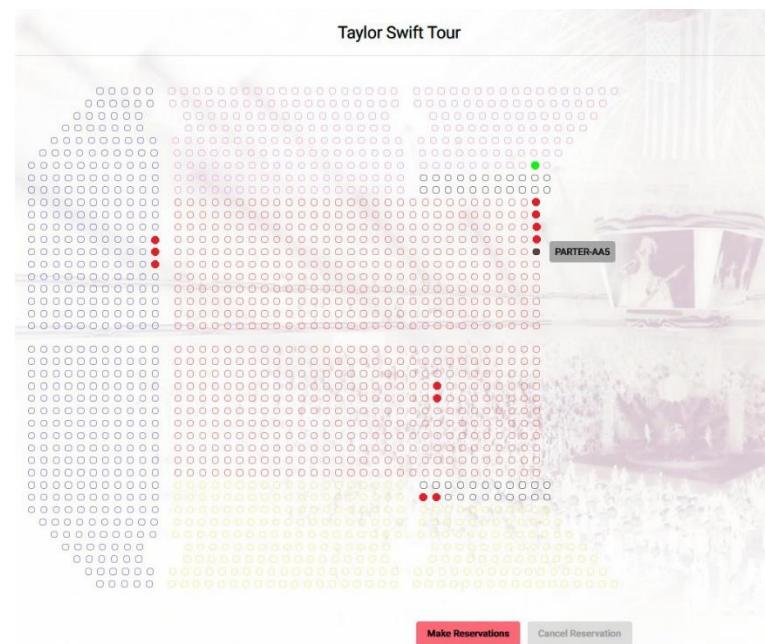
Kada se prijavi na sistem, obični korisnik takođe može da vidi stranicu sa događajima, ali njegova jedina opcija je pregled detaljnog prikaza događaja (slika 6.6). Pored toga, može i da pregleda svoje rezervacije, a ta opcija se nalazi u *navigation bar-u*.

Name	Start	End	Reservation deadline	Parter	East	West	North	South	VIP
Maneskin Concert	9/3/21, 9:00 PM	9/3/21, 11:30 PM	8/31/21, 7:00 PM	RSD3,000	RSD4,000	RSD4,000	RSD4,000	RSD4,200	<button>Details</button>
Quidditch World Cup Final	8/30/21, 6:00 PM	8/30/21, 11:00 PM	8/28/21, 7:00 PM	RSD3,000	RSD5,000	RSD5,000	RSD5,000	RSD5,000	<button>Details</button>
Taylor Swift Tour	8/28/21, 9:00 PM	8/28/21, 11:30 PM	8/27/21, 6:00 PM	RSD3,500	RSD4,000	RSD4,000	RSD4,000	RSD4,500	<button>Details</button>

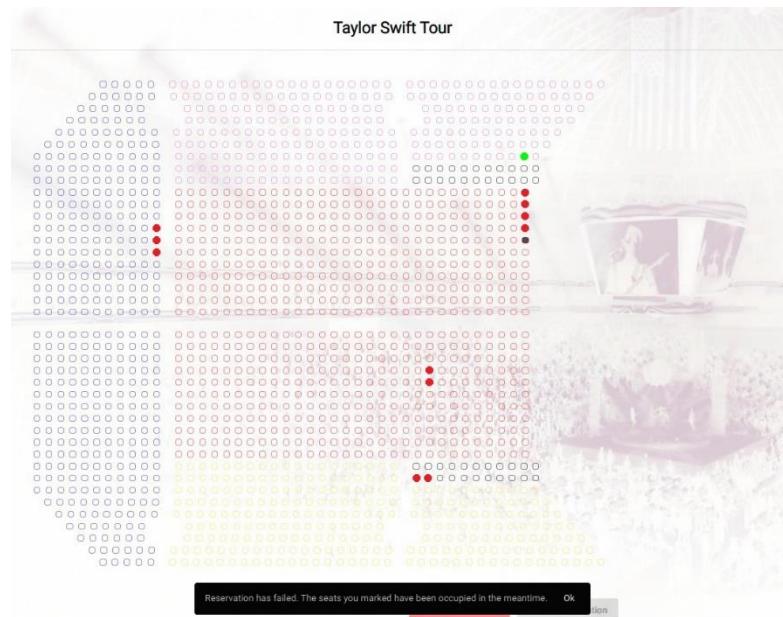
Slika 6.6. Prikaz početne stranice „Events“ za korisnika

Kod detaljnog prikaza događaja korisnik može da vidi sjedišta rezervisana od strane drugih korisnika označena crvenom bojom, svoja rezervisana sjedišta označena zelenom bojom koja može da otkaže i slobodna sjedišta koja može da rezerviše. Prelaskom miša preko sjedišta korisniku se prikazuje broj tog sjedišta. Rezervacija ili otkazivanje se izvršava tako što korisnik selektuje izabrana sjedišta i klikne na odgovarajuće dugme ispod. Nemoguće je odjednom rezervisati i otkazivati, odnosno ako korisnik selektuje slobodno sjedište s namjerom da ga rezerviše, dugme za otkazivanje se onemogućava, i obrnuto (slika 6.7). Nakon uspješne akcije korisnik dobija mejl sa informacijama o rezervaciji.

Ukoliko korisnik pokuša da rezerviše sjedišta tako da ukupno ima više od 4 rezervacije za jedan događaj, ili ukoliko pokuša da rezerviše sjedišta za događaj kojem je prošao rok za rezervisanje, prikazaće mu se poruka o neuspješno obavljenoj akciji. Isto će se desiti i ako neko drugi u međuvremenu rezerviše selektovano sjedište (slika 6.8).

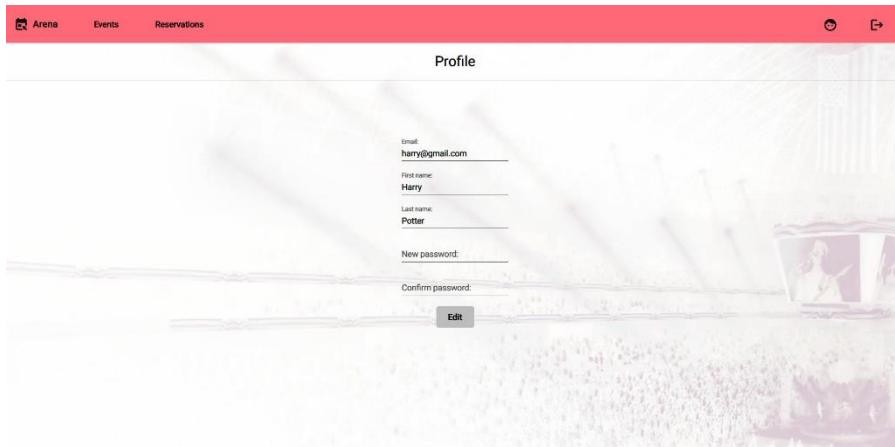


Slika 6.7. Prikaz rezervisanja sjedišta



Slika 6.8. Prikaz neuspješnog rezervisanja sjedišta

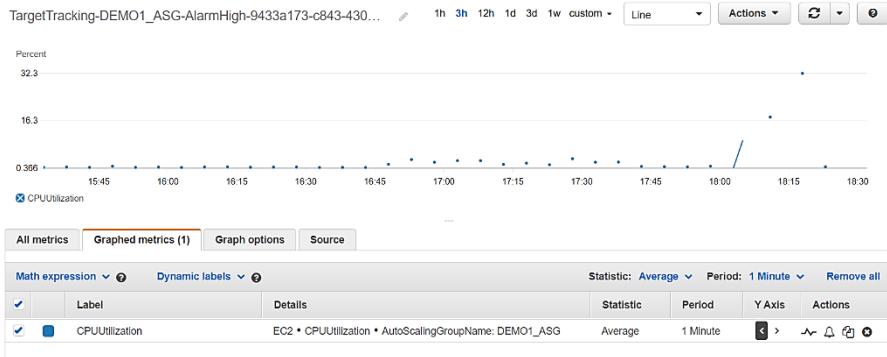
Oba tipa korisnika na desnoj strani *navigation bar-a* imaju opciju za pregled i izmjenu svojih podataka i odjavu sa sistema. Na slici 6.9 prikazana je forma za izmjenu koja dozvoljava izmjenu imena, prezimena i lozinke.



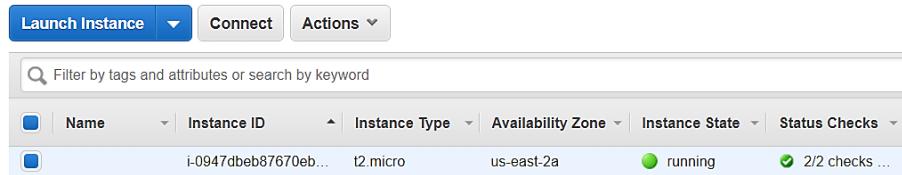
Slika 6.9. Prikaz starnice za izmjenu svog profila

6.2. Prikaz elastičnosti i dostupnosti sistema

Kao što je rečeno u poglavlju o implementaciji, definisana su dva alarma koja prate iskorišćenje procesora EC2 instanci. Jedan alarm se aktivira kada je izmjereno prosječno iskorišćenje veće od 50% tri puta zaredom (*AlarmHigh*), a drugi kada je iskorišćenje manje od 35% 15 puta zaredom (*AlarmLow*). Na slici 6.10 prikazan je grafik na kojem se vidi stanje alarma *AlarmHigh* kada vrijednost metrike ne prelazi 50 i alarm nije aktiviran. *Auto Scaling* grupa na koju se odnosi alarm u tom trenutku sadrži jednu ispravnu EC2 instancu (slika 6.11).

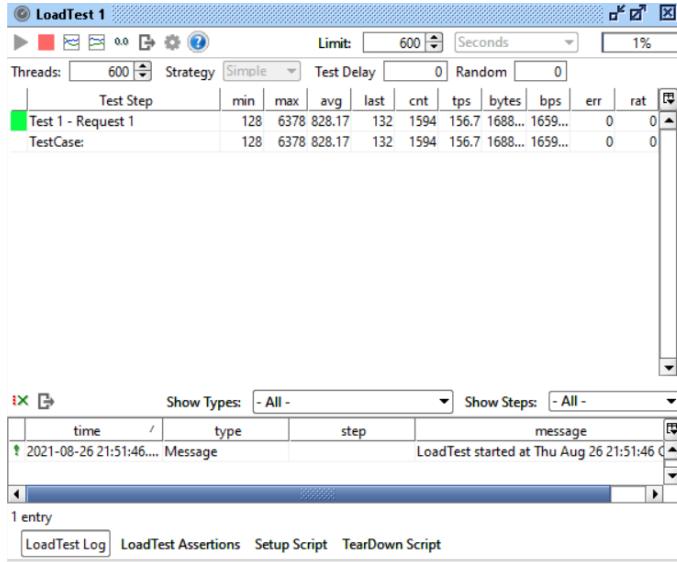


Slika 6.10. Prikaz *AlarmHigh* alarma u stanju OK



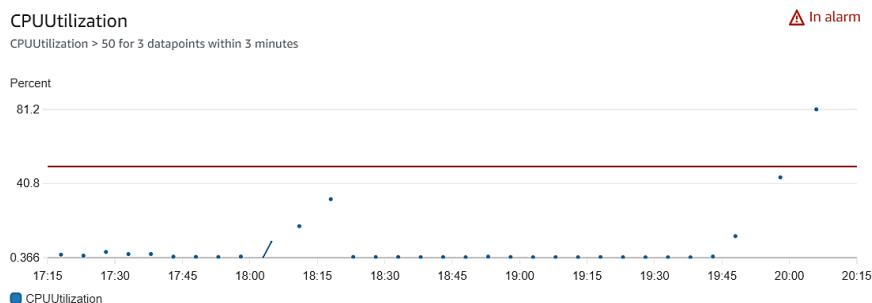
Slika 6.11. Prikaz EC2 instance

Kako bi se pokazala mogućnost sistema da alocira i pušta resurse po potrebi, odnosno povećava i smanjuje broj EC2 instanci, izvršen je test opterećenja (engl. *load test*) nad aplikacijom. Test je održan pomoću alata *SoapUI* [42] koji omogućava slanje velikog broja zahtjeva aplikaciji odjednom, u ovom slučaju 600 (slika 6.12). To je potrebno da bi se izazvalo povećanje iskorišćenosti procesora.



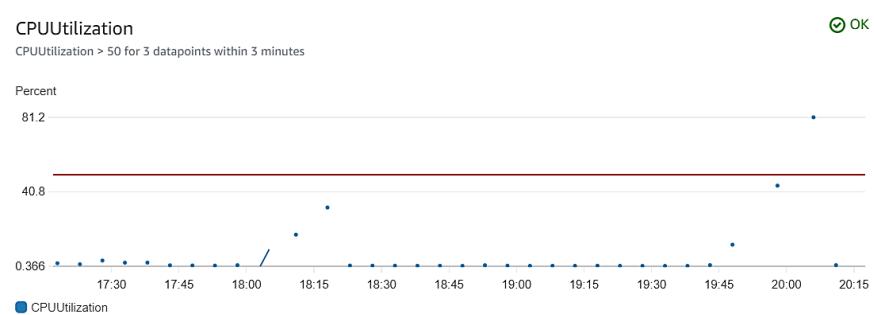
Slika 6.12. Prikaz load testa

Na slici 6.13 prikazan je *AlarmHigh* nekoliko minuta nakon pokretanja testa. Na x osi je predstavljeno vrijeme, a na y osi procenat iskorišćenja procesora. Alarm je aktiviran jer je u posljednje tri minute tri puta izmjereno iskorišćenje procesora koje prelazi prag. Odgovor sistema na aktiviranje ovog alarma je pokretanje još jedne EC2 instance u sklopu *Auto Scaling* grupe (slika 6.14). Ona će omogućiti da se opterećenje rasporedi i da se smanji iskorišćenje procesora pojedinačne instance. Kada se opterećenje smanji i mjerena više ne pokazuju da metrika prelazi prag, alarm se vraća u stanje OK (slika 6.15).

Slika 6.13. Prikaz grafika aktiviranog alarma *AlarmHigh*

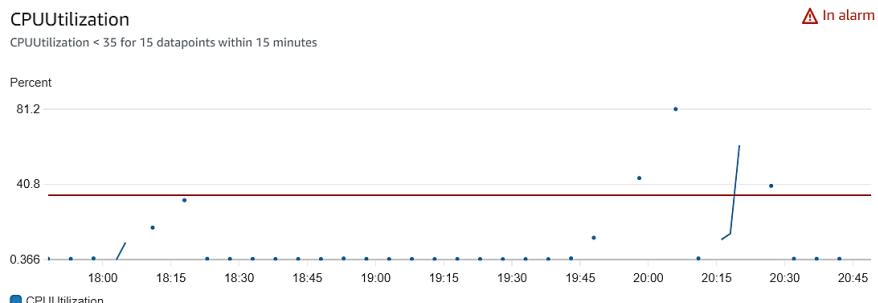
Instance Type	Availability Zone	Instance State	Status Checks
t2.micro	us-east-2b	● running	hourglass Initializing
t2.micro	us-east-2a	● running	checkmark 2/2 checks ...

Slika 6.14. Prikaz odgovora sistema na aktiviranje alarma *AlarmHigh*



Slika 6.15. Prikaz grafika alarma *AlarmHigh* ponovo u stanju OK

Drugi alarm, *AlarmLow*, služi da bi se kapacitet grupe vratio na 1 kada više nije potrebno da postoje dvije instance. Na slici 6.16 prikazan je aktivirani *AlarmLow*, nakon što je izvršio 15 mjerena pri kojima je vrijednost metrike bila ispod praga.



Slika 6.16. Prikaz grafika aktiviranog alarma *AlarmLow*

Na slici 6.17 prikazana je aktivnost alarma *AlarmLow*, gdje se može vidjeti da je nakon prelaska alarma u stanje ALARM izvršena akcija

uklanjanja instance iz *Auto Scaling* grupe. Uklonjena instanca više ne prolazi provjeru ispravnosti, pa se uklanja iz *target* grupe (slika 6.18).

History (4)			
Date	Type	Description	
2021-08-26 20:48:53	Action	Successfully executed action arn:aws:autoscaling:us-east-2:289736816129:scalingPolicy:bc5a0576-b803-4456-869f-1d173afa3933:autoScalingGroupName/DEMO1_ASG:policyName/Target Tracking Policy	
2021-08-26 20:48:53	State update	Alarm updated from OK to In alarm	
2021-08-26 20:12:53	State update	Alarm updated from Insufficient data to OK	
2021-08-26 20:12:02	Configuration update	Alarm "TargetTracking-DEMO1_ASG-AlarmLow-cc023728-71bb-49e6-a8d7-c8a2957e8bac" created	

Slika 6.17. Prikaz aktivnosti alarma *AlarmLow*

Port	Availability Zone	Status	Description
8080	us-east-2a	healthy	This target is currently passing target group's health checks.
8080	us-east-2b	draining	Target deregistration is in progress

Slika 6.18. Prikaz instanci *target* grupe nakon aktiviranja alarma *AlarmLow*

Da bi se pokazalo kako sistem radi u slučaju kada jedna od zona dostupnosti koje koristi postane nedostupna, npr. zbog vremenskih nepogoda, potrebno je simulirati otkaz zone dostupnosti. To je odrđeno kreiranjem nove ACL liste TEST_ACL koja zabranjuje sav saobraćaj (slika 6.19). Ta lista se zatim pridružuje privatnoj podmreži u kojoj se pokreću instance, a koja se nalazi u zoni dostupnosti *us-east-2a*.

Network ACLs (1/4) <small>Info</small>					
<input type="button" value="Create network ACL"/> Actions <small>▼</small>					
<input type="button" value="Filter network ACLs"/>					
Name	Network ACL ID	Associated with	Default	VPC ID	
<input checked="" type="checkbox"/> TEST_ACL	acl-0846dd425203448e	subnet-09e8df0bc4be31e28 / 10.0.3.0_AREN...	No	vpc-0746842817b96852a / ARENA_VPC	<input type="button" value="Edit"/> <input type="button" value="Actions"/>
<input type="checkbox"/> -	acl-f9e09792	3 Subnets	Yes	vpc-9b40d6f0	
<input type="checkbox"/> ARENA_ACL	acl-003b57d49dad9d2...	subnet-0f755dc0bb0627445 / 10.0.4.0_AREN...	No	vpc-0746842817b96852a / ARENA_VPC	
<input type="checkbox"/> -	acl-0ef83c90d62eb53c0	3 Subnets	Yes	vpc-0746842817b96852a / ARENA_VPC	

acl-0846dd425203448e / TEST_ACL					
Details	Inbound rules	Outbound rules	Subnet associations	Tags	

Inbound rules (1)						
<input type="button" value="Edit inbound rules"/> Actions <small>▼</small>						
<input type="button" value="Filter inbound rules"/>						
Rule number	Type	Protocol	Port range	Source	Allow/Deny	
*	All traffic	All	All	0.0.0.0/0	<input checked="" type="radio"/> Deny	

Slika 6.19. Prikaz ACL liste za simulaciju otkaza zone dostupnosti

Nakon dodavanja nove ACL liste, instanca pokrenuta u zoni *us-east-2a* više ne prolazi provjeru ispravnosti (slika 6.20). Kada se to desi, sistem će pokrenuti novu instancu kako bi kapacitet ostao isti, u ovom slučaju 1 (slika 6.21), i to u drugoj zoni dostupnosti, *us-east-2b*. Na slici 6.22 prikazana je *target* grupa nakon pokretanja nove instance, koja uklanja neispravnu instancu i nastavlja da koristi ispravnu.

Registered targets					
Instance ID	Name	Port	Availability Zone	Status	Description
i-0332ca6afeadc92fa		8080	us-east-2a	unhealthy	Request timed out
Availability Zones					
Availability Zone	Target count	Healthy?			
us-east-2a	1	Healthy?	No (Availability Zone contains no healthy targets)		

Slika 6.20. Prikaz *target* grupe u slučaju otkaza zone dostupnosti

	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks
		i-00df94a8fee400fcf	t2.micro	us-east-2b	pending	Initializing
		i-0332ca6afeadc92fa	t2.micro	us-east-2a	running	2/2 checks passed

Slika 6.21. Prikaz pokretanja nove instance u slučaju otkaza zone dostupnosti

Registered targets					
Instance ID	Name	Port	Availability Zone	Status	Description
i-0332ca6afeadc92fa		8080	us-east-2a	draining	Target deregistration is in progress
i-00df94a8fee400fcf		8080	us-east-2b	healthy	This target is currently passing target group's health checks.

Slika 6.22. Prikaz *target* grupe nakon pokretanja nove instance

U ovakovom slučaju kada sistem koristi dvije zone dostupnosti, on će biti nedostupan u kratkom periodu kada pokreće novu instancu. Zbog toga je poželjno koristiti tri zone dostupnosti, jer će u tom slučaju postojati još jedna ispravna instanca koja radi cijelo vrijeme.

7. ZAKLJUČAK

U ovom radu opisana je izgradnja skalabilne, elastične i visoko dostupne softverske arhitekture na *cloud*-u pomoću AWS servisa. Dat je kratak opis drugih značajnih dobavljača servisa kao što su *Microsoft Azure* i *Google Cloud Platform*, a zatim i teoretski opis AWS servisa koji mogu da se koriste za izgradnju takve arhitekture. Objasnjen je rad sa mrežom na *cloud*-u pomoću VPC servisa, rad sa bazama podataka pomoću RDS servisa, skladištenje pomoću S3 servisa i rad sa virtuelnim serverima, njihovo nadgledanje i skaliranje pomoću servisa EC2, *Auto Scaling* i *CloudWatch*.

U drugom dijelu rada prikazan je model predložene arhitekture i data je specifikacija jednostavne veb aplikacije koja je postavljena na navedenu arhitekturu. Takođe, opisana je implementacija aplikacije pomoću tehnologija *Spring* i *Angular* i koraci i podešavanja koja su potrebna za postavljanje aplikacije na AWS. Na kraju su prikazane funkcionalnosti sistema i testirana je elastičnost i dostupnosti.

Arhitektura sistema bi mogla da se unapredi upotrebom servisa koji nisu besplatni, što je opisano u okviru poglavљa o modelu predložene arhitekture. Pošto korišćeni servisi ne obezbjeđuju visoku dostupnost baze podataka, mogu se koristiti RDS *Multi-AZ Deployment* ili *Amazon Aurora* koji to omogućavaju. Takođe, aplikacija može da se postavi na skalabilnu arhitekturu automatski korišćenjem servisa *Elastic Beanstalk*.

LITERATURA

- [1] Michael Wittig, Andreas Wittig (2016). Amazon Web Services in Action
- [2] HTTP dokumentacija, <https://httpwg.org/specs/>
- [3] AWS dokumentacija, <https://docs.aws.amazon.com/>
- [4] AWS dokumentacija - *Well-Architected Framework*,
<https://aws.amazon.com/architecture/well-architected>
- [5] AWS dokumentacija - koncepti *Well-Architected Framework-a*,
<https://wa.aws.amazon.com/wellarchitected/2020-07-02T19-33-23/wat.concepts.wa-concepts.en.html>
- [6] AWS dokumentacija - *Free Tier*, <https://aws.amazon.com/free>
- [7] AWS dokumentacija - globalna infrastruktura,
<https://aws.amazon.com/about-aws/global-infrastructure>
- [8] Microsoft Azure dokumentacija, <https://docs.microsoft.com/en-us/azure>
- [9] Google Cloud Platform dokumentacija, <https://cloud.google.com/docs>
- [10] Gartner (2021). *Magic Quadrant for Cloud Infrastructure and Platform Services*, <https://www.gartner.com/doc/reprints?id=1-271OE4VR&ct=210802&st=sb>
- [11] AWS dokumentacija – servisi,
<https://www.amazonaws.cn/en/products/>
- [12] Spring dokumentacija, <https://docs.spring.io/spring-framework/docs/current/reference/html/>
- [13] Angular dokumentacija, <https://angular.io/docs>
- [14] AWS dokumentacija – *Management Console*,
<https://docs.aws.amazon.com/awsconsolehelpdocs/index.html>
- [15] AWS dokumentacija - EC2,
<https://docs.aws.amazon.com/ec2/index.html>
- [16] AWS dokumentacija – VPC,
<https://docs.aws.amazon.com/vpc/index.html>
- [17] AWS dokumentacija – *Elastic Beanstalk*,
<https://docs.aws.amazon.com/elastic-beanstalk/index.html>
- [18] AWS dokumentacija – *Auto Scaling*,
<https://docs.aws.amazon.com/autoscaling/index.html>
- [19] AWS dokumentacija - S3,
<https://docs.aws.amazon.com/s3/index.html>
- [20] AWS dokumentacija – *Glacier*,
<https://docs.aws.amazon.com/glacier/index.html>

- [21] AWS dokumentacija – RDS,
<https://docs.aws.amazon.com/rds/index.html>
- [22] AWS dokumentacija – DNS,
<https://aws.amazon.com/route53/what-is-dns/>
- [23] AWS dokumentacija – *Elastic Load Balancer*,
<https://docs.aws.amazon.com/elasticloadbalancing/index.html>
- [24] AWS dokumentacija – *CloudWatch*,
<https://docs.aws.amazon.com/cloudwatch/index.html>
- [25] AWS dokumentacija – IAM, <https://docs.aws.amazon.com/iam/>
- [26] AWS dokumentacija – SNS, <https://docs.aws.amazon.com/sns/>
- [27] Simplilearn (2016), Amazon VPC Tutorial,
<https://www.youtube.com/watch?v=fpxDGu2KdkA>
- [28] RDP dokumentacija, <https://docs.microsoft.com/en-us/troubleshoot/windows-server/remote/understanding-remote-desktop-protocol>
- [29] OSI model dokumentacija,
<https://docs.snowme34.com/en/latest/reference/network/osi-model.html>
- [30] REST dokumentacija,
<https://www.mulesoft.com/resources/api/what-is-rest-api-design>
- [31] JSON dokumentacija, <https://www.json.org/json-en.html>
- [32] Java dokumentacija – JDBC API,
<https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/>
- [33] JWT dokumentacija, <https://jwt.io/introduction>
- [34] Spring dokumentacija – *Spring Cloud*,
<https://cloud.spring.io/spring-cloud-aws/spring-cloud-aws.html>
- [35] HTML dokumentacija, <https://devdocs.io/html/>
- [36] CSS dokumentacija, <https://developer.mozilla.org/en-US/docs/Web/CSS>
- [37] SCSS dokumentacija, <https://sass-lang.com/documentation>
- [38] TypeScript dokumentacija, <https://www.typescriptlang.org/docs/>
- [39] Kod aplikacije, <https://github.com/verak13/Arena>
- [40] Apache Maven dokumentacija, <https://maven.apache.org/guides/>
- [41] PostgreSQL dokumentacija – kontrola konkurentnog pristupa,
<https://www.postgresql.org/docs/9.1/mvcc.html>
- [42] SoapUI dokumentacija, <https://www.soapui.org/docs/>

BIOGRAFIJA

Vera Kovačević je rođena 13.12.1998. godine u Kozarskoj Dubici. Osnovnu školu „Vuk Stefanović Karadžić“ završila je 2013. godine kada upisuje gimnaziju u MSŠ „Nikola Tesla“ u Kozarskoj Dubici, a završila je 2017. godine. Iste godine upisala se na Fakultet tehničkih nauka, smjer Softversko inženjerstvo i informacione tehnologije. Položila je sve ispite predviđene planom i programom.

KLJUČNA DOKUMENTACIJSKA INFORMACIJA

Redni broj, RBR:	
Identifikacioni broj, IBR:	
Tip dokumentacije, TD:	Monografska publikacija
Tip zapisa, TZ:	Tekstualni štampani dokument
Vrsta rada, VR:	Diplomski-master rad
Autor, AU:	Vera Kovačević
Mentor, MN:	dr Siniša Nikolić, docent, FTN Novi Sad
Naslov rada, NR:	Izgradnja skalabilne aplikacije u oblaku korišćenjem AWS servisa
Jezik publikacije, JP:	Srpski
Jezik izvoda, JI:	Spiski / engleski
Zemlja publikovanja, ZP:	Srbija
Uže geografsko područje, UGP:	Vojvodina
Godina, GO:	2021
Izdavač, IZ:	Autorski reprint
Mesto i adresa, MA:	Novi Sad, Fakultet tehničkih nauka, Trg Dositeja Obradovića 6
Fizički opis rada, FO:	Br. poglavljia 7/ stranica 79/ citata 42/ tabela 0/ slika 69/ grafikona 0/ priloga 0
Naučna oblast, NO:	Softversko inženjerstvo i informacione tehnologije
Naučna disciplina, ND:	Web programiranje
Predmetna odrednica / ključne reči, PO:	AWS platforma, softverska arhitektura, elastičnost, visoka dostupnost
UDK	
Čuva se, ČU:	Biblioteka Fakulteta tehničkih nauka, Trg Dositeja Obradovića 6, Novi Sad
Važna napomena, VN:	
Izvod, IZ:	U radu su opisani osnovni AWS servisi i definisana je AWS arhitektura koja omogućava elastičnost i visoku dostupnost. Prikazana je specifikacija i implementacija demo aplikacije i način na koji je ona postavljena na predloženu arhitekturu. Takođe, prikazan je i način korišćenja aplikacije i demonstrirana je elastičnost i visoka dostupnost sistema.
Datum prihvatanja teme, DP:	
Datum odbrane, DO:	
Članovi komisije, KO:	
predsednik	dr Goran Sladić, vanr. prof., FTN Novi Sad
član	dr Željko Vuković, docent, FTN Novi Sad
mentor	dr Siniša Nikolić, docent, FTN Novi Sad
	Potpis mentora

KEY WORDS DOCUMENTATION

Accession number, ANO:	
Identification number, INO:	
Document type, DT:	Monographic publication
Type of record, TR:	Textual material
Contents code, CC:	BSc thesis
Author, AU:	Vera Kovačević
Mentor, MN:	Siniša Nikolić, PhD
Title, TI:	Development of a scalable application using AWS services
Language of text, LT:	Serbian
Language of abstract, LA:	Serbian / English
Country of publication, CP:	Serbia
Locality of publication, LP:	Vojvodina
Publication year, PY:	2021
Publisher, PB:	Author's reprint
Publication place, PP:	Novi Sad, Faculty of Technical Sciences, Trg Dositeja Obradovića 6
Physical description, PD:	no. of chapters 7/ pages 79/ quotes 42/ tables 0/ pictures 69/ graphs 0/ appendix 0
Scientific field, SF:	Software engineering and information technologies
Scientific discipline, ND:	Web programming
Subject / Keywords, S/KW:	AWS platform, software architecture, elasticity, high availability
UDC	
Holding data, HD:	Library of the Faculty of Technical Sciences, Trg Dositeja Obradovića 6, Novi Sad
Note, N:	
Abstract, AB:	This paper describes basic AWS services and defines production environment using various AWS services which enables elasticity and high availability. It contains specification and implementation of a demo applicaton which is deployed on presented architecture. In the end, the use of application is shown and elasticity and availability is tested.
Accepted by sci. board on, ASB:	
Defended on, DE:	
Defense board, DB:	
president	Goran Sladić, PhD, assoc. prof, FTN Novi Sad
member	Željko Vuković, PhD, assist. prof, FTN Novi Sad
mentor	Siniša Nikolić, PhD, assist. prof., FTN Novi Sad
	Mentor's signature