Projektni zadatak iz predmeta

Računarstvo u oblaku

Školska 2024/2025. godina

Projekat predstavlja web aplikaciju za skladištenje, deljenje i slušanje muzičkog sadržaja. Aplikacija se razvija upotrebom **AWS cloud servisa** prateći **cloud native** arhitekturu.

Tipovi korisnika

- **Neautentifikovani korisnik** - Može da se registruje na sistem. Ako poseduje nalog može da se prijavi na sistem.
- **Administrator** - Autentifikovani korisnik u ulozi administratora. Ima mogućnost dodavanja i uređivanja muzičkog sadržaja i umetnika.
- **Redovan korisnik** - Redovan korisnik koji se uspešno prijavio na sistem. Ima mogućnost pretrage i pregledanja postojećeg sadržaja. Može da oceni sadržaj i kreira liste. Može da se pretplati na određeni sadržaj i rukovodi postojećim pretplatama. Dobija notifikacije za sadržaj na koji se pretplatio. Ima personalizovane preporuke na osnovu prethodnih interakcija sa sistemom. Može da lokalno preuzme okačeni sadržaj.

Delovi sistema

- **Klijentska aplikacija** - Pruža grafički interfejs preko kog korisnici pristupaju funkcionalnostima sistema.
- **Serverska aplikacija** - *Cloud-native* aplikacija koja sadrži poslovnu logiku sistema. Potrebno je osmisliti arhitekturu koja će ispuniti sve funkcionalne i nefunkcionalne zahteve sistema. Potrebno je iskoristiti **adekvatne** AWS servise.

1. Funkcionalni zahtevi

1.1. (7) Registracija korisnika (neautentifikovani korisnik)

Korisnik se registruje tako što unosi ime, prezime, datum rođenja, korisničko ime i email (moraju biti jedinstveni), i lozinku.

1.2. (7) Prijava na sistem (neautentifikovani korisnik)

Unosom korisničkog imena i lozinke korisnik može pristupiti svom nalogu i funksionalnostima sistema koje odgovaraju ulozi korisnika.

1.3. (6) Kreiranje umetnika (administrator)

Potrebno je omogućiti unos osnovnih informacija o umetniku, minimalno uključujući ime, biografiju i žanrove.

1.4. (6) Postavljanje muzičkog sadržaja (administrator)

Pored samog sadržaja, potrebno je skladištiti i prateće informacije o sadržaju. Minimalno je potrebno podržati: naziv datoteke, tip datoteke, veličinu datoteke, vreme nastanka i vreme poslednje izmene. Informacije o sadržaju je moguće dobiti iz metapodataka same datoteke. Pored navedenih, potrebno je podržati definisanje pratećih informacija od strane administratora, kao što su naziv, žanrovi, slika i slično. Sadržaj može biti vezan za jednog ili više umetnika. Sadržaj može i ne mora pripadati albumu (single). Administrator ima mogućnost da postavi i ceo album odjednom.

1.5. (6) Pregled sadržaja (administrator/redovan korisnik)

Korisnici mogu pregledati sadržaj koji je okačen. Pregled podrazumeva uvid u prateće informacije i mogućnost slušanja muzičkog sadržaja kroz klijentsku aplikaciju.

1.6. (8) Izmena sadržaja (administrator)

Administrator može menjati sadržaj ili dodatne informacije vezane za njega.

1.7. (8) Brisanje sadržaja (administrator)

Administrator može brisati bilo koji sadržaj (pesma, album, umetnik). Prilikom brisanja, brišu se i dodatne informacije vezane za sadržaj.

1.8. (7) Filtriranje sadržaja - discover stranica (redovan korisnik)

Redovan korisnik ima mogućnost filtriranja sadržaja putem discover stranice. Korisnik bira žanr, nakon čega mu se izlistavaju svi albumi/umetnici koji pripadaju tom žanru. Nakon toga, korisnik ima mogućnost odabira umetnika/albuma i izlistavanja svog sadržaja vezanih za njih.

1.9. (10) “Offline”* reprodukcija sadržaja (redovan korisnik)

Korisnik ima mogućnost odabira pesama koje će biti dostupne u “offline”* režimu. U tom slučaju, pesma se preuzima lokalno. Prilikom daljeg pregleda odabranog sadržaja, korisniku se reprodukuje preuzeta pesma.

Dodatno, korisnik ima mogućnost preuzimanja sadržaja bez dodatnih informacija (samo pesma) na lokalni fajl sistem (download).

*Ovim nije podržana potpuna offline reprodukcija (bez ikakve internet konekcije), već vid keširanja sadržaja. Ko želi, može istražiti i implementirati potpunu offline reprodukciju. **Nije neophodno podržati potpunu offline reprodukciju** - dodatak je samo za one koji žele dodatno da istraže.

1.10. (9) Ocenjivanje sadržaja (redovan korisnik)

Korisnik ima mogućnost ocenjivanja sadržaja. Potrebno je minimum podržati 3 ocene (na primer brojčano 1-3, love/like/dislike ili slično). Konkretan odabir prepušten je studentima (pojašnjenje zahteva).

1.11. (8) Pretplata na sadržaj (redovan korisnik)

Korisnik ima mogućnost pretplate na sadržaj koji ga interesuje. Pretplata može biti na osnovu umetnika, žanra i slično. Prilikom postavljanja novog sadržaja, svi korisnici koji su pretplaćeni na taj sadržaj automatski dobijaju notifikaciju.

1.12. (9) Rukovanje pretplatom na sadržaj (redovan korisnik)

Korisnik ima uvid u sve svoje pretplate i ima mogućnost brisanja svake pretplate.

1.13. (9) Personalizovan preporuke - “feed” (redovan korisnik)

Korisniku se nakon prijave na sistem prikazuje personalizovan “feed” na osnovu dotadašnje interakcije sa sistemom. Algoritam kreiranja personalizovanog “feed”-a treba da se sastoji iz više koraka i uzima u obzir različite faktore. Algoritam treba da uzme u obzir minimalno sledeće:

- ocene koje je korisnik ostavio na prethodno slušanom sadržaju,
- pretplate na sadržaj koje je korisnik kreirao,
- sadržaj koji je korisnik prethodno slušao i vreme kada ga je slušao.

Feed treba da se menja automatski usled dodavanja novog sadržaja od strane administratora ili kroz interakcije korisnika sa sistemom.

Primer 1: korisnik je pretplaćen na žanr pop i administrator je objavio novi pop album; korisniku će se u feed-u pojaviti novi album.

Primer 2: korisnik je ranije slušao lo-fi žanr uveče, ali u skorije vreme često u to vreme sluša i pozitivno ocenjuje rok pesme; korisniku će se u feed-u uveče pojaviti rok pesme umesto lo-fi pesama.

1.14. (10) Transkripcija teksta pesme (sistem)

Korisniku se uz pesmu prikazuje i njen tekst. Prilikom postavljanja sadržaja potrebno je implementirati automatsku obradu audio sadržaja i tekstualne transkripcije uz vođenje računa o greškama i ponavljanju obrade ukoliko do istih dođe. *Redovan korisnik može da sluša pesmu i pre nego što se završi transkripcija teksta pesme.*

2. Nefunkcionalni zahtevi

2.1. (6) Cloud-native arhitektura

Potrebno je modelovati sistem u skladu sa cloud-native arhitekturom. Neophodno je iskoristiti adekvatne AWS servise za podršku iste. Neophodno je kreirati dijagram arhitekture.

2.2. (6) Odvojeno skladištenje sadržaja i dodatnih informacija
Sadržaj i dodatne informacije je potrebno čuvati u odgovarajućim skladištima.

2.3. (7) Performanse sistema prilikom filtriranja
Prilikom filtriranja treba voditi računa o performansama sistema. Odabir i modelovanje podataka u okviru skladišta utiče na način i performanse dobavljanja podataka.

2.4. (8-9) Infrastructure as Code
Instanciranje i konfiguraciju **svih** potrebnih servisa neophodno je odraditi upotrebom nekog od Infrastructure as Code alata. Moguće je koristiti deklarativan ili imperativan pristup pisanja IaC.

2.5. (9) Stil komunikacija između komponenti
Potrebno je voditi računa o stilu komunikacije između komponenti u okviru projekta. U zavisnosti od funkcionalnosti i toka zahteva, potrebno je koristiti sinhronu ili asinhronu komunikaciju gde za njih ima smisla, prateći principe *event-driven arhitekture*.

2.6. (7) API gateway
Predstavlja ulaznu tačku u sistem i sva komunikacija između klijentske i serverske aplikacije obavlja se putem nje. API gateway klijentima nudi REST API za komunikaciju.

2.7. (7) Deployment frontend aplikacije
Potrebno je odraditi deployment frontend aplikacije. Aplikacija treba da bude javno dostupna i da omogućava interakciju sa sistemom.

2.8. (8) Sistem notifikacija
Korisnik dobija notifikaciju kada se postavi novi sadržaj na koji je pretplaćen. Notifikacija treba da sadrži osnovne informacije o sadržaju.

Polaganje projekta

Konsultativna tačka pregledanja
Termin: 21.7-25.7.
Studenti imaju mogućnost da demonstriraju do tada implementirano i time dobiju povratnu informaciju od asistenta o dosadašnjem radu. Predlog je da pokušate da implementirate sve funkcionalnosti navedene u tabeli za ocenu 6. Uz to, zgodno je pokazati inicijalnu verziju (**dijagram**) potpune arhitekture rešenja prateći cloud-native arhitekturu i savladane AWS servise.

Finalna odbrana
Termin: druga nedelja septembra

Napomena: datumi nisu fiksni i uz dogovore može doći do izmene u toku semestra.

Osnovne informacije

- Projekat se implementira u **timovima od 3 člana**.
- Studenti imaju mogućnost odabira bilo kojih tehnologija i programskih jezika za implementaciju projekta. Jedino ograničenje je upotreba AWS servisa.
- Dodatni rokovi za finalnu odbranu projekta biće organizovani u zavisnosti od broja studenata i dogovora sa istima.
- Pitanja vezana za specifikaciju projekta i način polaganja predmeta možete postaviti predmetnim asistentima. Česta pitanja i odgovore će biti objavljena na dnu specifikacije. **Proveravajte spisak pitanja i odgovora.**

Ocenjivanje

1. Za sve ocene neophodno je kreirati **klijentsku aplikaciju**. Klijentska aplikacija mora biti frontend web aplikacija (zbog mogućnosti deployment-a), studentima se ostavlja slobodan izbor tehnologija. Klijentska aplikacija nije fokus predmeta i kvalitet i “lepota” (UX/UI) njene izrade neće se uzimati u obzir prilikom ocenjivanja.
2. Za svaku ocenu su navedeni zahtevi koje je potrebno implementirati. Ukoliko uradite sve zahteve za ocenu 6, ne znači da automatski dobijate ocenu 6. Potrebno je da sakupite dovoljno bodova (predispitne + ispitne obaveze) kako biste dobili određenu ocenu. Zahtevi su grupisani po težini i znanju koje se smatra odgovarajućim za određenu ocenu.
3. Zahtevi u tabeli koji su **označeni crvenom bojom su eliminacioni za tu ocenu**. Ukoliko tim ne implementira eliminacione zahteve, ne može dobiti tu ocenu i gubi mogućnost odbrane naredne ocene.

Bodovanje po zahtevima

Zahtevi - ocena 6	Broj bodova
1.3. Kreiranje umetnika	**2**
1.4. Postavljanje muzičkog sadržaja	**6**
1.5. Pregled sadržaja	**2**
2.1. Cloud-native arhitektura	**-**
2.2. Odvojeno skladište sadržaja i dodatnih informacija	**-**

Zahtevi - ocena 7	Broj bodova
1.1 Registracija	**2**

1.2. Prijava na sistem	**1**
1.8. Filtriranje sadržaja - Discover stranica	**5**
2.3. Performanse sistema prilikom filtriranja	**-**
2.6. API gateway	**-**
2.7. Deployment frontend aplikacije	**2**

Zahtevi - ocena 8	Broj bodova
1.6. Izmena sadržaja	**0.5**
1.7. Brisanje sadržaja	**1.5**
1.11. Pretplata na sadržaj	**5**
2.8. Sistem notifikacija	**-**
2.4. Infrastructure as Code - **deklarativno**	**3**

Zahtevi - ocena 9	Broj bodova
1.10. Ocenjivanje sadržaja	**1**
1.12. Rukovanje pretplatom na sadržaj	**1**
1.13. Personalizovan “feed”	**5**
2.5. Stil komunikacije	**-**
2.4. Infrastructure as Code - **imperativno**	**3**

Zahtevi - ocena 10	Broj bodova
1.9. “Offline” reprodukcija sadržaja	**3**
1.14. Transkripcija teksta pesme	**7**

FAQ

- **Da li je upotreba GitHub repozitorijuma neophodna?**
  - Da.
- **Šta treba da se nalazi na GitHub repozitorijumu?**
  - Svi source fajlovi Lambda funkcija, klijentske aplikacije i Infrastructure as Code konfiguracije, kao i sva ostala prateća dokumenta i dijagrami.

- **Šta podrazumeva izmena sadržaja (tačka 1.5)?**
  - Korisnik može da menja dodatne informacije vezane za sadržaj (opis, tagovi,...). Za izmenu samog sadržaja, dovoljno je implementirati upload novog dokumenta. Nema potrebe kreirati editor audio dokumenata u okviru klijentske aplikacije.

- **Da li je klijentska aplikacija neophodna za kontrolnu tačku?**
  - Nije.

- **Na koji način se mogu okačiti dodatne biblioteke uz Lambda kod?**
  - Upotrebom zip upload-a ili kreiranjem Lambda slojeva.
  Zip upload treba da sadrži sve zavisnosti koje su potrebne za izvršavanje.
  Više informacija o zip upload-u za Python: https://docs.aws.amazon.com/lambda/latest/dg/python-package.html
  Više informacija o zip upload-u za Node: https://docs.aws.amazon.com/lambda/latest/dg/nodejs-package.html
  Lambda sloj predstavlja dodatnu konfiguraciju virtuelne mašine na kojoj se Lambda funkcija pokreće. Isti sloj se može podesiti na više različitih Lambda funkcija što olakšava održavanje. Više informacija o kreiranju Lambda slojeva: https://docs.aws.amazon.com/lambda/latest/dg/invocation-layers.html

- **Na koji način jednostavnije mogu da rukujem bibliotekama u Python okruženju?**
  - Kroz serverless alat preporučen način za jednostavno rukovanje zavisnostima je upotreba plugin-a koji omogućava specifikaciju zavisnosti kroz *requirements.txt* fajl.
  Više na https://www.serverless.com/plugins/serverless-python-requirements

- **Da li je potrebno raditi validaciju podataka i sa serverske strane?**
  - Da.

- **Na koji način je potrebno implementirati proveru validnosti autentifikacionih tokena?**
  - Proveru validnosti tokena potrebno je uraditi u okviru API Gateway-a upotrebom Cognito servisa ili custom Lambda funkcije. Rešenje u kom se validnost tokena proverava u svakoj Lambda funkciji neće biti prihvaćeno.

- **Da li je neophodno na svakom endpoint-u podržati asinhronu komunikaciju?**
  - Ne, ne morate dodavati redove čekanja gde nisu neophodni. Ako imate negde direktnu komunikaciju dve ili više Lambda funkcija gde postoji čekanje na odgovor, to bi bilo dobro mesto sa asinhronu komunikaciju. Potrebno je pratiti najbolje prakse event-driven arhitekture koju AWS podržava. Ukoliko niste sigurni da li biste negde trebali da imati neki vid asinhrone komunikacije, konsultujte se sa asistentnom.

- **Da li je za sistem notifikacija dovoljno obavestiti korisnika na frontu na osnovu HTTP response statusa?**
  - Ne, potrebno je poslati email/SMS notifikaciju korisniku ili omogućiti dvosmernu komunikaciju upotrebom web-socketa.

- **Da li je upotreba AWS Composer i sličnih servisa dozvoljena za kreiranje infrastrukture (IaC)?**
  - Ne. Dozvoljena je upotreba AWS CloudFormation, AWS SAM, Serverless Framework-a i svih AWS CDK alata za kreiranje infrastrukture (IaC).

- **Da li je upotreba AWS Composer dozvoljena za kreiranje dijagrama arhitekture?**
  - Ne.

- **Da li je dozvoljeno kreiranje IaC dokumenata na osnovu resursa kreiranih ručno kroz AWS UI?**
  - Ne.

- **Šta sve treba da se kreira upotrebom IaC?**
  - Svi resursi koje vaša aplikacija koristi. To uključuje sve Lambda funkcije, API Gateway, resurse za skladište, permisije i role potrebne za komunikaciju između resursa,... AWS naloge koje koristite vi (IAM nalozi), kao razvojni tim, nije potrebno kreirati kroz IaC.

- **Da li je neophodno podržati upload velikih fajlova?**
  - API Gateway podržava zahteve čiji sadržaj može biti maksimalno 10 MB, što ograničava veličinu sadržaja koji može biti objavljen. Moguće je zaobići API Gateway prilikom objave filma upotrebom Presigned URL-ova. Nije neophodno podržati postavljanje velikog sadržaja, ali ukoliko želite možete to uraditi.

- **Da li je za ocenu 9 potrebno implementirati IaC i deklarativno i imperativno?**
  - Ne, dovoljno je implementirati IaC samo imperativno. Ukoliko želite ocenu 8, dovoljno je implementirati ga deklarativno. Ni u jednoj varijanti nije neophodno kreirati IaC na oba načina.

- **Kada feed treba da se ažurira?**
  - Potrebno je ažurirati feed prilikom svake promene koja može značajno uticati na njega, to jest koja se uzima u obzir u algoritmu za kreiranje feeda. Nije dovoljno kreirati/ažurirati feed kada se korisnik ode na početnu stranicu, prvenstveno zbog neefikasnosti.

- **Da li je dozvoljena upotreba AWS Amplify?**
  - Ne. Dozvoljeno je ogrničeno korišćenje samo za komunikaciju sa Cognito servisom iz frontend aplikacije, ali ni ovo nije neophodno (slobodno se možete

osloniti na starije biblioteke za ovaj slučaj)! Nije dozvoljeno korišćenje bilo kakvih gotovih komponenti i rešenja upotrebom AWS Amplify.

- **Kako rukovati pretplatom na umetnike ukoliko se desi da se isto zovu?**
  - Dovoljno je omogućiti korisniku da se pretplati na umetnika po nazivu/imenu i prezimenu. Na primer, ukoliko se pretplatimo na umetnika Peru Perića, i desi se da postoji više umetnika istog imena, sasvim je u redu obavestiti korisnika kada se pojavi pesma od strane bilo kog Pere Perića.

- **Da li treba omogućiti pretragu po parcijalnom unosu?**
  - Ne. Dovoljno je omogućiti pretragu kako je opisana u zahtevu.

- **Da li je potrebno iskoristiti SQS negde?**
  - **Da.** Na finalnoj odbrani je neophodno naglasiti gde je upotrebljen red čekanja i to smisleno objasniti. Razmislite o delovima sistema koje želite da odvojite, koji treba da budu nezavisni, a povezani na neki način, ovo su jedna od dobrih mesta gde biste iskoristili SQS.

- **Šta podrazumeva ocenjivanje (1.10)?**
  - Redovan korisnik može oceniti bilo koju pesmu jednom ocenom. Korisniku treba omogućiti samo **jedan sistem za ocenjivanje**. Studenti imaju slobodu da biraju da li će to biti zvezdice 1-5, ili će biti like/dislike/love ili bilo koja alternativa. Bitno je da korisnik ima mogućnost da ostavi ocenu sa barem tri nivoa - samo like pesme nije dovoljan.

- **Da li pesma/umetnik/album mogu imati više žanrova?**
  - Da.

- **Kako omogućiti peformantno filtriranje po žanru kada pesma/umetnik/album mogu imati više žanrova?**
  - Jedan od načina je kreiranje dodatne zasebne tabele koja će čuvati vezu sa žanrovima, gde biste dobavili sve za žanr (pored toga, postoji još pristupa). Svi pristupi će imati prednosti i mane. Ukoliko želite, možete odabrati i olakšani pristup gde je dovoljno izdvojiti primarni žanr (od svih žanrova koje sadržaj ima) po kome se radi performantno filtriranje.

- **Da li je moguće naknadno dodati pesmu u album?**
  - Nije neophodno, ali možete podržati. Ne bi trebalo da predstavlja veliku izmenu.

- **Šta opisuje album?**
  - Dovoljno je da album ima naziv.

- **Šta se dešava sa pesmama kada se obriše album?**
  - Brišu se i pesme koje mu pripadaju.

- **Šta se dešava sa pesmom koja ima više umetnika kada se obriše jedan od umetnika?**
  - Pesma se ne briše. Bitno je da umetnik bude obrisan. U prikazu pesme možete sadržati ime obrisanog umetnika ili staviti placeholder (unknown artist ili slično).

- **Napomena vezana za upotrebu AWS-a**
  - **Uzmite u obzir ograničenja Free Tier AWS naloga.**
  Obratite pažnju na ograničenja veličine dokumenata u S3 skladištu i u transportu u i iz skladišta (kao i ograničenja svih drugih servisa).

- **Problem sa naplatom: problem sa CloudWatch alarmima**
  - U slučaju ručnog kreiranja DynamoDB tabele ili ECS klastera, postoji mogućnost kreiranja servisa uz AutoScaling mehanizam koji može da izazove alarme. Obratite pažnju da li vam je ostala ručno kreirana DynamoDB tabela ili ECS klaster. U slučaju da jeste, **izmenite konfiguraciju tako da ne podržite automatsko skaliranje. AutoScaling mehanizam može dovesti do pojave CloudWatch alarma čiji broj je ograničen u okviru Free Tier-a.**

Dodatni materijali*

- **AWS Cognito** - User Pool VS Identity Pool
  User Pool se koristi za rukovanje korisnicima i njihovu autentifikaciju i autorizaciju u okviru aplikacije koja je hostovana na AWS platformi
  Identity Pool se koristi za dodelu AWS IAM uloga korisnicima kako bi direktno koristili AWS resurse.
  Detaljnije objašnjenje na:
  https://tutorialsdojo.com/amazon-cognito-user-pools-vs-identity-pools/
  Za potrebe projekta, sva komunikacija će ići preko API Gateway-a, te nije neophodno koristiti Identity Pool.
- **CI/CD Pipeline** - automatski deployment klasične web aplikacije na EC2 instance
  - Integracija sa GitHub-om: https://aws.amazon.com/blogs/devops/integrating-with-github-actions-ci-cd-pipeline-to-deploy-a-web-app-to-amazon-ec2/
  - Upotreba AWS Git repozitorijuma: https://aws.amazon.com/getting-started/hands-on/set-up-ci-cd-pipeline/
- **Frontend deployment**
  - Deployment Angular frontend aplikacije upotrebom **S3** bucket hosting-a statičkog sadržaja: https://linuxhint.com/deploy-angular-app-aws/
  - Deployment Angular frontend aplikacije upotrebom **CloudFront** AWS servisa:

https://jayanttripathy.com/how-to-host-angular-app-on-aws-s3-bucket-using-cloudfront/

- **AWS SAM pomoćni materijali**
  - Česti slučajevi konfiguracije upotrebom AWS SAM alata i dobre prakse: https://fusari-pool.medium.com/

***Dodatni materijali (osim frontend deployment-a) nisu obavezni za izradu projekta!** Ali su svakako zanimljivi. :)