Projektni zadatak iz predmeta

Servisno orijentisane arhitekture i NoSQL baze podataka

Školska 2023/2024. godina

Kroz predmetni projekat treba da implementirate platformu za ponudu i rezervisanje smeštaja.

Uloge u sistemu

- **Neautentifikovani korisnik (NK)** - Može da kreira novi host ili guest nalog ili se prijavi na postojeći. Pored toga, može da pretražuje smeštaj, ali ne može da ga rezerviše niti da postavlja nove ponude.
- **Host (H)** - Kreira novi smeštaj i upravlja njime. Za svaki smeštaj definiše opremljenost, periode dostupnosti i cenu. Host na svom nalogu može da vidi i pretražuje sve nekretnine, ali ne može da ih rezerviše.
- **Guest (G)** - Rezerviše smeštaj. Svaku rezervaciju može otkazati pre njenog početka. Dostupna mu je funkcionalnost ocenjivanja smeštaja i host naloga.

Komponente sistema

- **Klijentska aplikacija** - Pruža grafički interfejs preko kog korisnici pristupaju funkcionalnostima sistema.
- **Serverska aplikacija** - Mikroservisna aplikacija koja sadrži čitavu poslovnu logiku sistema. Sastoji se iz sledećih servisa:
  - **Auth** - Čuva kredencijale korisnika i njihove uloge u sistemu. Zadužen za proces registracije i prijave korisnika.
  - **Profile** - Sadrži informacije o osnovnim podacima korisnika kao što su ime, prezime, pol, starost, imejl itd.
  - **Accommodations** - Upravlja osnovnim informacijama o smeštaju (naziv, opis, slike itd)
  - **Reservations** - Kontroliše periode dostupnosti i cene smeštaja, kao i sve rezervacije
  - **Recommendations** - Podržava operacije preporuke smeštaja za goste
  - **Notifications -** Upravlja skladištenjem i slanjem notifikacija korisnicima

Funkcionalni zahtevi

1.1 Registracija naloga (uloge: NK)

Svaki nalog mora imati jedinstven username. Minimalan skup ličnih podataka koje korisnik mora uneti pri registraciji je: ime, prezime, email adresa i mesto stanovanja.

1.2 Prijava na sistem (uloge: NK)

1.3 Upravljanje nalogom (uloge: H, G)

Korisnik može menjati sve lične podatke koje je uneo prilikom registracije, kao i svoje kredencijale (username i lozinku).

1.4 Brisanje naloga (uloge: H, G)

Guest može obrisati nalog ako nema aktivnih rezervacija. Host može obrisati nalog samo ako nema aktivnih rezervacija u budućnosti ni za jedan smeštaj kojim upravlja. Kada host obriše nalog, uklanjaju se i svi smeštaji koje je kreirao.

1.5 Kreiranje smeštaja (uloge: H)

A. Za smeštaj je potrebno uneti minimalno naziv, lokaciju, pogodnosti (na primer wifi, kuhinja, klima, besplatan parking itd) i minimalan i maksimalan broj gostiju.
B. Proširiti proces kreiranja smeštaja tako da se za smeštaj dodaju i slike.
C. Proširiti funkcionalnost tako da postoji mogućnost definisanja dostupnosti i cene smeštaja prilikom kreiranja

1.6 Definisanje dostupnosti i cene smeštaja (uloge: H)

Potrebno je omogućiti definisanje i izmenu termina kada je neki smeštaj dostupan. On se može rezervisati samo za vremenski interval kada je dostupan i nije već zauzet. Pored toga, cena smeštaja može biti promenljiva. Na primer ona može biti veća u letnjim mesecima, vikendom ili za vreme praznika. Omogućiti host-u da definiše ovakve razlike u ceni. Cena može da se zada po gostu ili po celoj smeštajnoj jedinici, a host bira jednu od te dve mogućnosti na nivou nekretnine. Dostupnost smeštaja i cena mogu se menjati za neki interval samo ako u njemu nema rezervacija.

1.7 Pretraga smeštaja (uloge: NK, H, G)

Smeštaj se pretražuje prema lokaciji, broju gostiju i datumu početka i kraja putovanja. Rezultati treba da obuhvate sve dostupne smeštaje u zadatom intervalu koji se nalaze na traženoj lokaciji i mogu primiti zadati broj gostiju. Pored osnovnih informacija, za svaku nekretninu treba da se prikaže ukupna cena smeštaja za sva noćenja, kao i jedinična cena (po osobi po noćenju ili po smeštaju po noćenju).

1.8 Kreiranje rezervacije (uloge: G)

Kada se rezervacija kreira niko drugi ne može da rezerviše smeštaj za zauzete datume.

1.9 Otkazivanje rezervacije (uloge: G)

Guest može otkazati rezervaciju pre njenog početka. U tom slučaju smeštaj postaje ponovo slobodan za te datume.

1.10 Ocenjivanje host-ova (uloge: G)

Guest može oceniti host-a ocenom od 1 do 5 samo ako je u prošlosti imao bar jednu rezervaciju u nekom njegovom smeštaju koju nije otkazao. Ocena se može promeniti ili ukloniti. Moguće je videti svaku pojedinačnu ocenu (ko ju je dao i kada), kao i srednju ocenu.

1.11 Ocenjivanje smeštaja (uloge: G)

Guest može oceniti smeštaj ocenom od 1 do 5 samo ako je tu odseo barem jednom u prošlosti. Ocena se može promeniti ili ukloniti. Moguće je videti svaku pojedinačnu ocenu (ko ju je dao i kada), kao i srednju ocenu.

1.12 Istaknuti host (uloge: H)

Istaknuti host je status koji host može steći ako ispunjava sledeće uslove:

- Ima ocenu veću od 4.7
- Ima stopu otkazivanja manju od 5%
- Imao je barem 5 rezervacija za smeštaje u prošlosti
- Ukupno trajanje svih rezervacija je veće od 50 dana

Na profilu host-a se prikazuje ovaj status ukoliko ga ima. Status nije trajan i uklanja se čim neki od navedenih uslova više nije ispunjen.

1.13 Filtriranje smeštaja (uloge: NK, H, G)

Rezultati pretrage smeštaja treba da se filtriraju prema sledećim kriterijumima:

- Opseg cene
- Pogodnosti koje smeštaj mora da ima
- Smeštaj mora da pripada istaknutom host-u

1.14 Notifikacije (uloge: H)

Host treba da dobije notifikacije kada:

- Neko kreira rezervaciju
- Neko otkaže rezervaciju
- Neko ga oceni
- Neko oceni njegov smeštaj

Notifikacija treba da se prikaže na profilu korisnika, ali da mu se pored toga pošalje i mail poruka. Korisnik može da pregleda sve prethodne notifikacije na svom profilu.

1.15 Preporuka smeštaja (uloge: G)

Korisniku treba prikazati preporuku smeštaja na osnovu njegovih prethodnih aktivnosti. Algoritam za preporuku je sledeći:

- Pronađi slične korisnike. Sličan korisnik je neko ko je rezervisao isti smeštaj i davao slične ocene u prošlosti.
- Pronađi smeštaj koji su rezervisali slični korisnici i dali mu dobru ocenu
- Odbaci smeštaj koji ima više od 5 ocena manjih od 3 u prethodna tri meseca
- Rangiraj smeštaj prema ukupnoj oceni i preporuči prvih 10 rezultata

Opisani algoritam moguće je proširiti.

1.16 Statistike smeštaja (uloge: H)

Za svaki smeštaj treba prikazati dnevni i mesečni izveštaj koji sadrži broj poseta stranice smeštaja, prosečno vreme zadržavanja na stranici, broj ocena i broj rezervacija.

Nefunkcionalni zahtevi

2.1 Dizajn sistema

Za svaki servis treba navesti tip skladišta i model podataka. Na nivou sistema potrebno je definisati sve vidove komunikacije između svakog servisa (sinhrono, asinhrono), kao i funkcionalnost pri kojoj dolazi do te komunikacije. Potrebno je priložiti dijagram na kom je ova komunikacija naznačena.

2.2 API gateway

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

2.3 Kontejnerizacija

Sve mikroservise, API Gateway i baze podataka potrebno je pokrenuti kao Docker kontejnere i koristiti Docker Compose alat.

2.4 Otpornost na parcijalne otkaze sistema

U slučaju da neki servis trenutno nije dostupan, svi ostali servisi treba da nastave da rade normalno i podrže funkcionalnosti koje nisu zavisne od navedenog servisa.

2.5 Tracing

Pomoću Jeager alata implementirati tracing u celoj mikroservisnoj aplikaciji.

2.6 Keširanje

Slike smeštaja treba keširati u Redis-u.

2.7 Saga

Funkcionalnost kreiranja novog smeštaja treba da bude implementirana preko saga šablona. Na odbrani je potrebno demonstrirati uspešan tok, ali i sve verzije neuspešnih tokova.

2.8 Event sourcing i CQRS

Prikupljanje i prikaz statistika o smeštaju treba implementirati upotrebom event sourcing i CQRS šablona. Svaka relevantna akcija se čuva kao događaj u event store-u po izboru. Pogledi u vidu dnevnih i mesečnih izveštaja mogu se čuvati u istoj ili u drugačijoj bazi.

2.9 Kubernetes

Sve komponente sistema treba pokrenuti unutar Kubernetes klastera.

Ocenjivanje

- Za ocenu **6** treba implementirati zahteve 1.1, 1.2, 1.5A, 1.6, 1.8, 2.1, 2.2, 2.3, 2.4
- Za ocenu **7** treba implementirati sve navedeno za ocenu 6 i zahteve 1.3, 1.4, 1.7, 1.9
- Za ocenu **8** treba implementirati sve navedeno za ocenu 7 i zahteve 1.5B, 1.10, 1.11, 1.14, 2.6
- Za ocenu **9** treba implementirati sve navedeno za ocenu 8 i zahteve 1.13, 1.15, 2.5
- Za ocenu **10** treba implementirati sve navedeno za ocenu 9 i zahteve 1.5C, 1.12, 2.7
- Za ocenu **10+** treba implementirati sve navedeno za ocenu 10 i zahteve 1.16, 2.8 i 2.9

  Napomena: Kvalitet implementacije se ocenjuje i svaku odluku koju ste doneli po pitanju arhitekture, komunikacije i modela podataka morate obrazložiti

  Napomena: Za zahtev 2.4 možete naći detaljno objašnjenje šta se od vas traži za koju ocenu u materijalima za vežbu 8 (readme.txt fajl)

Pravila polaganja

- Projekat se radi u timovima od po 4 člana. Članovi tima ne moraju slušati vežbe u istom terminu, sve dok mogu da se organizuju da na vežbe iz MRS predmeta dolaze u istom terminu.
- Za implementaciju serverske i klijentske aplikacije možete koristiti programske jezike i radne okvire po želji. Ukoliko odaberete tehnologije koje se razlikuju od onih koje su pokrivene na vežbama, pomoć koju asistenti mogu pružiti pri rešavanju problema je ograničena.
- Klijentska aplikacija služi da demonstrirate rad sistema i ne ocenjuje se.

- Ako implementirate zahteve navedene za ocenu 10+ oslobođeni ste polaganja teorijskog dela ispita na SOA i NoSQL predmetima.
- Sredinom decembra održaće se kontrolna tačka za koju je neophodno implementirati funkcionalnosti navedene za ocenu 6. Odrađene funkcionalnosti neophodno je demonstrirati kroz klijentsku aplikaciju (nije dovoljna upotreba alata poput Postman-a ili cURL-a).
- Ko izađe na kontrolnu tačku i bude zadovoljan ocenom, ne mora da dolazi na finalnu odbranu, koja će se održati početkom februara.
- Ako ne izađete na kontrolnu tačku, **ocena vam se smanjuje za jednu na finalnoj odbrani** (na primer morate odraditi funkcionalnosti za ocenu 10 kako biste dobili ocenu 9).
- U septembru će biti održan još jedan termin finalne odbrane, na kom vam se **ocena smanjuje za jednu**.

Informaciona bezbednost 2023/2024 zahtevi

1 BEZBEDNOST SISTEMA I ZAŠTITA PODATAKA

Razvoj bezbednog softvera podrazumeva ugrađivanje bezbednosnih kontrola u softver tokom njegovog razvoja, prateći koncept poznat kao *built-in security*. Trošak ugrađivanja bezbednosti na ovaj način je najmanji i, po pravilu, bezbednost je implementirana najkvalitetnije, jer se koncipira bezbedan dizajn koji će kod poštovati, umesto da se bezbednost *ad hoc* prilagođava već napisanom kodu.

Potrebno je implementirati sledeće bezbednosne mehanizme:

1.1 Validaciju podataka

- Sprečiti relevantne *Injection* napade;
- Sprečiti XSS napade;
- Izvršiti validaciju podatka, koristeći kriterijume validacije definisane po najboljim praksama za pisanje bezbednog koda.

1.2 HTTPS komunikaciju

- Potrebno je demonstrirati bezbednu komunikaciju između API gateway-a i klijentske aplikacije

1.2a Demonstracija bezbedne komunikacije između servisa

1.3 Autentifikaciju i kontrolu pristupa

- Omogućiti mehanizme za potvrdu naloga, oporavak lozinke i promenu lozinke;
- Kontrolisanje pristupa *endpoint*-ima po RBAC modelu;
- Kontrola pristupa frontend dela (detalji implementacije prepušteni studentima);
- Testirati i demonstrirati da sve kontrole pristupa rade (pozitivan i negativan ishod);

1.4 Zaštita podataka

Osetljivi podaci sa kojim aplikacija radi treba da budu obezbeđeni u skladištu, u transportu i tokom upotrebe. Identifikovati osetljive podatke, definisati i implementirati prikladne bezbednosne kontrole. Podaci čije skladištenje se ne može izbeći treba da budu šifrovani ili heširani ukoliko je to prikladno. Poruke u internoj komunikaciji treba da imaju očuvanu poverljivost, integritet i neporecivost, kao i da budu zaštićene od *replay* napada.

2 LOGGING I RANJIVE KOMPONENTE

Log zapisi koje generišu aplikacije i operativni sistemi nekog postrojenja su veoma korisni, kako sa aspekta debagovanja problema, tako i za potrebe bezbednosti. Log zapisi predstavljaju osnovni mehanizam za postizanje neporecivosti. Dodatno, kolekcije log zapisa se mogu slati alatima za *monitoring*, poput SIEM alata, čiji zadatak je da prati događaje u sistemu i da okine alarm svaki put kada se sumnjivo ponašanje detektuje. U sklopu odvojene priče, današnji softverski sistemi značajno zavise od komponenti koje nisu dizajnirali i programirali inženjeri originalnog sistema. Od infrastrukture (operativni sistem, baza podataka) do alata (radni okvir, biblioteke), značajan deo koda nije pod našom kontrolom. Međutim, to ne smanjuje našu odgovornost kada nam softver bude eksploatisan zbog ranjivosti u nekoj *third-party* komponenti jer iako nismo pravili tu komponentu, svesno smo je integrisali u naše rešenje.

Potrebno je implementirati logging mehanizam koji ispunjava sledeće zahteve:

1. **Kompletnost** – log zapis mora da sadrži dovoljno informacija da dokaže neporečivost i svaki događaj za koji je neporečivost potrebna treba da bude zabeležen. Dodatno, svaki *security-related* događaj, interesantan za potrebe *monitoring*-a, treba da bude zabeležen.

2. **Pouzdanost** – logging mehanizam treba da bude pouzdan, što podrazumeva dostupnost samog mehanizma (gde je neophodno voditi računa o memorijskom zauzeću log datoteka i napraviti mehanizam za rotaciju logova), kao i integritet log datoteka. Dodatno, dizajnirati kod tako da aplikacija nastavi sa radom u slučaju da logging mehanizam otkaže.

3. **Konciznost** – logging mehanizam treba da proizvodi najmanju količinu zapisa koji su potrebni da ispuni svoju svrhu. Dodatno, optimizovati svaki zapis da sadrži sve informacije, a zauzima najmanju količinu memorije.

2.1 Ranjivosti

Neophodno je:

- Definisati alate koji će se koristiti za proveru različitih skupova komponenti.
- Isproveravati svaku od komponenti i sakupiti listu ranjivosti.
- Analizirati ozbiljnost ranjivosti i mogućnost eksploatacije.
- Definisati i izvršiti strategiju za razrešenje mogućih rizika.
- Kreirati izveštaj neke vrste, koji će istaći temeljnost analize i krajnje rezultate. Format i tačan sadržaj je proizvoljan.

Zahteve navedene iznad je moguće formalno ispuniti bez istraživanja i mnogo truda, no to rešenje neće biti kvalitetno. Kvalitet je upravo ono što se ocenjuje, i da bi se date stavke ispunile neophodno je razmotriti savete i najbolje prakse koje možete pronaći online, poput onih navedenih u OWASP ASVS standardu.

Napomena:

- Obratiti pažnju na *best practice* konfiguraciju bezbednosnih funkcija koje se koristite.
- Potrebno je implementirati funkcionalnosti tek toliko da se podrži smislena demonstracija bezbednosnih kontrola.
- Pojedine tačke je moguće rešiti uz pomoć tehnologije za implementaciju softvera (jezika, radnog okvira) ili alata – ovo je dozvoljeno, no neophodno je razumeti kako tehnologija rešava problem i o čemu treba voditi računa da se pružena bezbednosna kontrola „ne pokvari“.
- Prilikom istraživanja i implementacije kontrola, neophodno je voditi računa o bezbednoj konfiguraciji kontrole – skup parametara koje kontrola ima i njihova *best practice* vrednost.

Ocenjivanje

Za ocenu 6 neophodno je implementirati sledeće zahteve:

- **1.1**
- **1.3**
- **1.4**

Za ocenu 7 neophodno je implementirati sve za ocenu 6 i sledeće zahteve:

- 1.2
- integrisati postojeći sistem sa nekim od alata za statičku analizu koda (SonarCloud, SonarQube, …). Analizirati i rešiti ranjivosti u skladu sa preporukama alata.

Za ocenu 8 neophodno je implementirati sve za ocenu 7 i sledeće zahteve:

- **2**

Za ocenu 9 neophodno je implementirati sve za ocenu 8 i sledeće zahteve:

- 2.1

Za ocenu 10 neophodno je implementirati sve za ocenu 9 i sledeće zahteve:

- 1.2a

Kontrolne tačke

U toku semestra rad na projektu biće proveravan 2 puta. Prva kontrolna tačka će se održati sredinom semestra (okvirno kraj novembra), a druga kontrolna tačka ujedno će predstavljati i finalnu odbranu projekta (okvirno januar). Od ukupno 35 bodova koliko nosi KT1, na finalnoj odbrani-KT2 može se nadoknaditi maksimalno 10 bodova.

KT1 (35 bodova)

- 1.1 za implementirani deo projekta na drugim predmetima
- 1.4 za implementirani deo projekta na drugim predmetima
- 1.3
- 1.2

KT2 (36 bodova)

- 1.1 za ceo projekat
- 1.4 za ceo projekat
- 1.2a
- 2
- 2.1

Usmeni ispit

- Usmeni ispit **je obavezan za ocene 9 i 10**.
- Usmeni ispit **nije obavezan za ocene 6,7,8**.
- Ko na predispitim obavezama ispuni zahteve za ocene 6 i 7 može izaći na usmeni maksimalno za ocenu 8.

Dodatni rokovi

Pored odbrane na KT2 u januaru, organizovaće se još dva termina odbrane predispitnih obaveza, u junu i septembru. Odbrane će biti održane maksimalno za ocenu 6, bez mogućnosti izlaska na usmeni ispit.

Projekat za studente koji polažu samo informacionu bezbednost

Potrebno je implementirati sve funkcionalne zahteve iz specifikacije SOA-NoSQL osim stavki 1.12 1.14, 1.15 i 1.16. Aplikaciju po izboru implementirati kao monolitnu ili mikroservisnu. Kao bazu podataka koristiti baze po izboru. Za ocenu 10 aplikacija mora poštovati mikroservisnu arhitekturu. Ovu verziju projekta izrađuju jednočlani timovi.