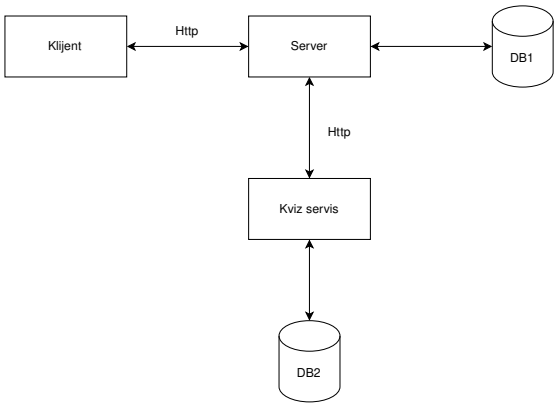
Distribuirani računarski sistemi
2025/2026

Kviz platforma

Potrebno je implementirati platformu koja simulira osnovne funkcionalnosti platforme za kreiranje i pokretanje kviza. Sistem se sastoji od pet komponenata:

1. **Korisnički interfejs**
2. **Servis za obradu zahteva i podataka (Server)**
3. **Baza podataka 1**
4. **Servis za rad sa kvizom**
5. **Baza podataka 2**



1. Klijent

Korisnički interfejs je React ili Angular web aplikacija koja komunicira sa **Serverom** putem **REST API** poziva i **WebSocket-a** za real-time događaje.

2. Server

Python Flask aplikacija koja opslužuje klijenta i komunicira direktno sa bazom podataka. Zadatak je da obradi sve REST API zahteve koje dobije od klijenta, kao i emitovanje real-time događaja.

3. Baza podataka 1

Baza podataka predstavlja mesto za čuvanje podataka u sistemu. Podaci koje baza čuva su podaci o registrovanim korisnicima I drugi podaci po potrebi. U okviru projekta potrebno je doneti odluku o tipu baze podataka — SQL ili NoSQL, u zavisnosti od prirode i strukture podataka.

4. Kviz servis

Python Flask aplikacija dobija novokreirane kvizove i čuva ih. Vrši njihove izmjene, brisanja i obrade.

5. Baza podataka 2

Ova baza podataka čuva isključivo podatke o kvizu. U okviru projekta potrebno je doneti odluku o tipu baze podataka — SQL ili NoSQL, u zavisnosti od prirode i strukture podataka.

Specifikacija zadatka

Autentifikacija

Korisnik se na platformu prijavljuje unošenjem kredencijala:
- Email
- Lozinka

Neophodno je implementirati autentifikaciju zasnovanu na JWT (JSON Web Token). Na serverskoj strani pratiti neuspešne pokušaje prijave (pogrešan imejl ili lozinka) na sistem i ukoliko korisnik ima tri neuspešna pokušaja privremeno mu blokirati pristup npr. na 15 minuta (za testiranje na npr. 1 minut).

Omogućiti korisniku odjavu sa sistema.

Upravljanje korisnicima

Novi korisnički nalog se kreira prilikom registracije korisnika na platformu, pri čemu se unose:
- Ime
- Prezime
- Email
- Datum rođenja
- Pol
- Država
- Ulica
- Broj

Novom korisniku nakon otvaranja korisničkog naloga se dodeljuje uloga **IGRAČ**. Korisnik koji želi da postavlja nove kvizove na platformu mora da bude verifikovan kao **MODERATOR**, odnosno **ADMINISTRATOR** treba da mu postavi ulogu moderatora. Nakon dodjele moderatora igraču, korisnik dobija mail da mu je promijenjena uloga.

Administrator može da izlista sve korisnike platforme, briše korisničke naloge i mijenja uloge korisnicima. Korisnik, bilo da je IGRAČ ili MODERATOR, može da izmijeni sve podatke o svom korisničkom profilu, kao i da doda sliku za svoj korisnički profil.
Podatke o registrovanim korisnicima obradjivati na Serveru i čuvati u DB 1.

Rad sa kvizovima

Novi kviz na platformu može da postavi moderator, pri čemu unosi sledeće podatke.
- Naziva kviza
- Pitanje (od 1 do n)
- Za svako pitanje postaviti ponuđene odgovore (od 2 do m)
- Za svako pitanje odrediti koji broj bodova nosi to pitanje
- Odrediti tačne odgovore (može ih biti više)
- Vrijeme trajanja kviza (za testiranje postavljati u sekundama)
- Podatak ko je autor kviza

Kada moderator kreira kviz takav kviz se u realnom vremenu prikazuje administratoru (koristiti **Web Socket**) koji treba da prihvati ili odbije kviz. Ukoliko se prihvati takav kviz on postaje vidljiv igračima. Ukoliko administrator želi da odbije kviz mora da ostavi razlog u vidu teksta i tada se vraća moderatoru na izmjenu.

Kviz može da briše moderator ili administrator dok igrači mogu samo da igraju.

Kada igrač pokrene kviz potrebno je pokrenuti tajmer za koje vrijeme je potrebno da se kviz izvrši. Kada igrač završi kviz potrebno je poslati na obradu odgovore. Obrada se izvršava asinhrono što igraču omogućava da nastavi rad na platformi. (dodati neki sleep kako bi simulirali duze trajanje obrade kviza)

Kada rezultat kviza bude obrađen igraču se šalje mail sa rezultatima i ti rezultati se čuvaju u bazu. Kreiranje, izmjene, brisanje i obradu kviza izvršava Kviz servis dok se podaci o kvizu čuvaju u DB2. Za svaki kviz je moguće vidjeti rang listu gdje je dostupan podatak ko je uradio kviz, utrošeno vrijeme i broj osvojenih bodova.

Generisanje izvještaja

Administrator ima mogućnost da kreira PDF izvještaj o rezultatima svakog kviza. Kreirani izvještaj mu stiže na mail.

Napomene

- Neophodno je imati validaciju kako na klijentskoj strani, tako i na strani servera.
- Lozinke se moraju čuvati hešovane, nikako u čitljivom formatu.
- Prilikom odabira tipa baze podataka prvo istražite prednosti i mane SQL i NoSQL (nikako ne koristiti SQLite).
- Istražiti načine organizacije projekta kako u React/Angular, tako i u Flask-u. (struktura foldere)
- Ukoliko se odlučite za React, koristiti Vite.js alat.
- Koristiti ORM za komunikaciju sa bazom podataka (npr. SQLAlchemy za SQL baze).
- Koristiti ODM za komunikacuju sa bazom podataka (npr. PyMongo za MongoDB).
- Koristiti DTO klase za komunikaciju izmedju frontend-a i backenda ili izmedju dva servisa.
- U okviru projekta neophodno je na smislenom mestu upotrebiti procese.
- Koristiti virtuelno okruženje (venv, pipenv…).

Na odbranu projekta doći sa pripremljenim podacima za testiranje.

Način ocenjivanja

1. Aplikacija je funkcionalna i postoji Flask aplikacija – **51 poen**
    a. Flask aplikacija + Baza podataka
2. Zaseban UI projekat (React, Angular…) koji komunicira sa Engine-om – **10 poena**
3. Implementacija keš baze podataka – **14 poena**
4. Koriscenje procesa prilikom implementacije – **10 poena**
5. Dokerizacija aplikacije (moraju sve komponente) - **10 poena**
6. Pokretanje na više računara – **5 poena**

BROJ OSVOJENIH POENA SE MNOŽI SA 0.9