I Faza projekta iz Arhitekture i projektovanja softvera

Arhitektura sistema

TaskIT

Marija Trajković, 18020

Elektronski fakultet, Niš

1. *Kontekst i cilj softverskog projekta*

TaskIT je višekorisnička web aplikacija, njena svrha je povezivanje korisnika koji mogu da postavljaju različite oglase za poslove koje žele da neko drugi obavi umesto njih i korisnike koji žele da te poslove obavljaju. Korisnik može da se prijavi na svoj nalog, da postavi oglas za posao koji želi da neko obavi za njega, da se i sam prijavljuje na oglase, da otkazuje poslove, da se pretplati na tipove poslova koje bi obavljao ili na korisnike za koje bi obavljao te poslove i dobija obaveštenja o izmenama oglasa. Dodatno, moguće je oceniti korisnika za urađen posao što bi pomoglo drugim poslodavcima da lakše izaberu ljude koji će obaviti posao za njih.

Cilj aplikacije je da oslobodi ljude neželjenih poslova za koje su spremni da angažuju nekoga kome bi ta izrada bila korisna, najčešće novčano.

2. *Arhitekturno specifični zahtevi, glavni funkcionalni zahtevi I ne-funkcionalni zahtevi (atributi kvaliteta), tehnička i poslovna ograničenja*
    a. Glavni funkcionalni zahtevi
        i. Kreiranje korisničkog naloga (Registracija)
        ii. Logovanje na nalog (Prijavljivanje)
        iii. Kreiranje oglasa za posao
        iv. Izmena oglasa za posao
        v. Brisanje oglasa za posao
        vi. Pregled dostupnih oglasa za posao
        vii. Praćenje poslova
        viii. Praćenje korisnika
        ix. Ocenjivanje korisnika za obavljeni posao
        x. Prijavljivanje za obavljanje posla
        xi. Prihvatanje prijave za posao od strane poslodavca
        xii. Otkazivanje prijave za neki posao od strane korisnika
        xiii. Obaveštavanje o novim oglasima za posao, o prijavi za obavljanje posla, o ponovo dostupnim poslovima i o izmenama u oglasima za posao
        xiv. Skladištenje podataka(o korisnicima, poslovima)

    b. Ne - funkcionalni zahtevi
        i. Pouzdanost i dostupnost (Reliability, Availability) – sistem treba da omogući perzistentnost akcija u konačnom vremenskom periodu, da bude dostupan korisnicima u svakom trenutku, kao i da bude otporan na otkaze i greške.
        ii. Modifikabilnost (Modifiability) – sistem treba organizovati tako da bude lako proširljiv bez potrebe za većim izmenama u sistemu u toku

njegovog životnog veka, a radi zadovoljenja novih (ne-) funkcionalnih zahteva.

iii. Skalabilnost (Scalability) – sistem je u stanju da podrži rast broja konkurentnih korisnika.
iv. Performanse (Performance) – sistem ima što manje vreme odziva kako bi se ispunio uslov real-time aplikacije.
v. Sigurnost (Security) – sistem je sposoban da garantuje bezbednost podataka korisnika od neovlašćenog pristupa i zloupotrebe podataka.
vi. Upotrebljivost (Usability)- sistem treba da bude jednostavan i intuitivan za širok spektar korisnika.
vii. Testabilnost (Testability) – sistem je jednostavnog dizajna i moguće ga je lako testirati.

c. Tehnička I poslovna ograničenja
   i. Kako je sistem web aplikacija, neophodno je korišćenje web tehnologija koje omogućuju pokretanje aplikacije u različitim web pretraživačima i jednostavnu i efikasnu komunikaciju korisnika i sistema.
   ii. Implementacija modela podataka i njihova reprezentacija u bazi podataka je sakrivena od korisnika aplikacije.
   iii. Komunikacija je podržana kroz sinhroni i asinhroni tip komunikacije. Sinhrona komunikacija se odvija između klijenta i servera, dok se asinhroni tip komunikacije odvija između samih klijenata prilikom postavljanja, brisanja i otkazivanja poslova i ocenjivanja korisnika.

3. *Arhitekturni dizajn softverskog sistem*

- Arhitekturni obrasci koji će biti korišćeni
   1. Layered obrazac – sistem će biti realizovan u troslojnoj arhitekturi kako bi se omogućila modularnost sistema i nezavisnost u razvoju određenih delova sistema. Arhitektura sistema će se sastojati od sloja perzistencije , serverskog sloja i prezentacionog klijentskog sloja. Sloj perzistencije je u osnovi, komunicira sa serverskim slojem i omogućava skladištenje podataka tj. predstavlja samu bazu podataka. Prezentacioni sloj obezbeđuje interakciju korisnika sa sistemom preko korisničkog interfejsa. Povezan je sa serverskim slojem. Serverski sloj predstavlja vezu između sloja perzistencije i prezentacionog sloja. Izvršava se na serveru i implementira poslovnu logiku sistema, funkcije za persistenciju podataka i sinhronu i asinhronu komunikaciju sa klijentom.

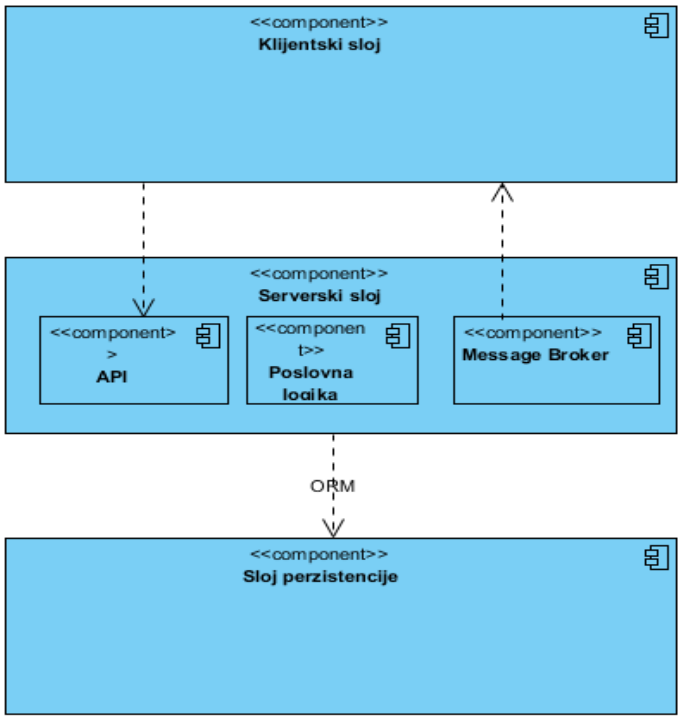
2. Publish - Subscribe obrazac – distribuirani sistem sam po sebi zahteva neki vid komunikacije. Za ostvarivanje asinhrone komunikacije ovaj obrazac iskorišćen je kroz SiganlR. Korisnici primaju obaveštenja o izmenama oglasa ali i o kreiranju novih oglasa preko tipova poslova koje prati ili poslodavaca. Takođe, poslodavac će biti obavešten o praćenju, o prijavi za posao i otkazivanju posla.

3. Model-View-Controller obrazac – ovaj obrazac biće implementiran u sistemu kroz svoja tri dela. Model predstavlja sloj domenskih klasa kojem će odgovarati organizacija podataka u bazi podataka. View deo će biti zadužen za izvršavanje aplikacije na klijentu tj. za sve ono što će klijent videti i omogućavaće klijentu da kroz logiku u Controller-u utiče na model podataka.

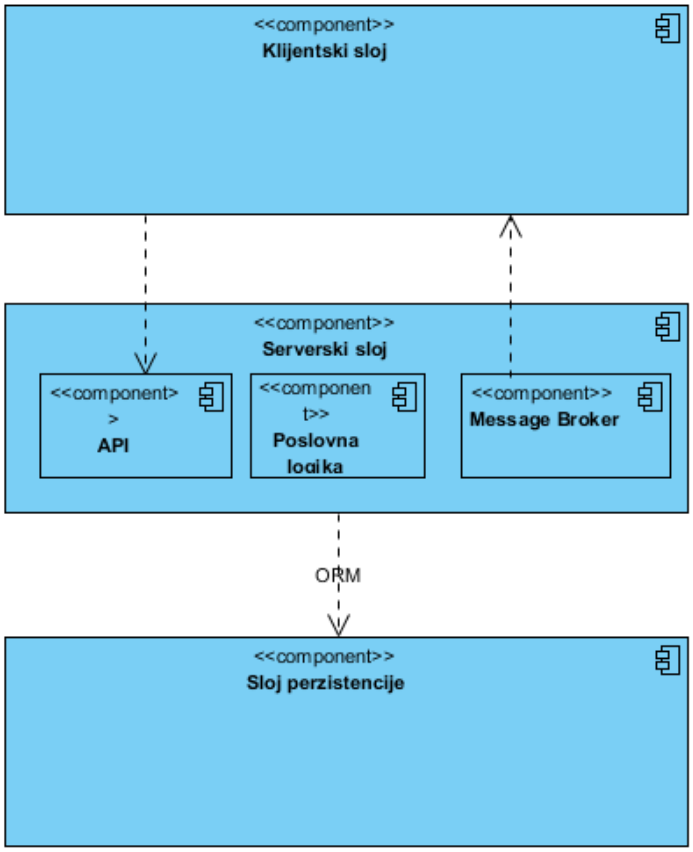
4. Repository obrazac – Perzistencija podatka se obavlja preko centralizovane baze podataka. Na sloju podataka koristi se Entity Framework koji sam po sebi već poseduje implementiran ovaj obrazac. Korišćenjem ovog obrasca biće postignuta jednostavnost implementacije logike za pristup bazi podataka što će olakšati izmenu i proširljivost sistema.

- Generalna arhitektura (box-line model)

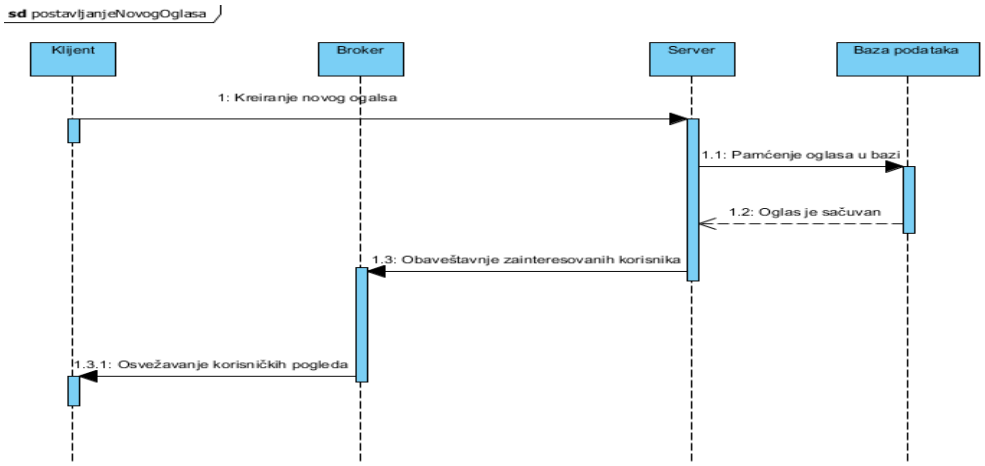
Arhitektura Sistema se sastoji od tri osnovna dela klijentskog sloja, serverskog sloja i sloja baze podataka:



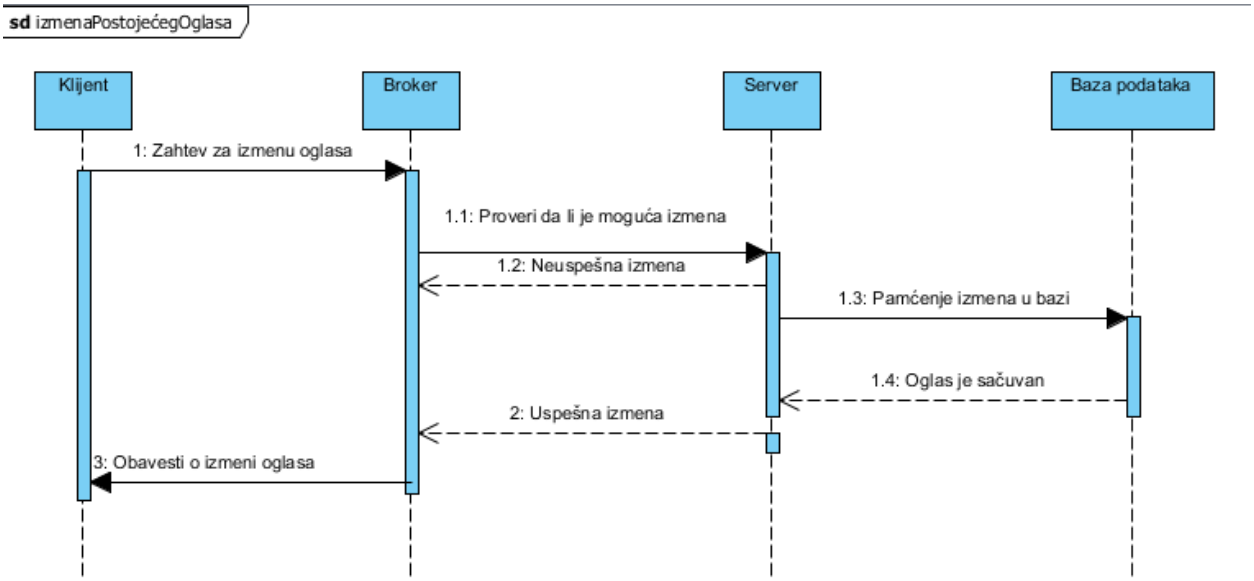
- Dijagrami osnovnih modula (komponenti) i njihovih veza (konektora)
  - Strukturni pogled



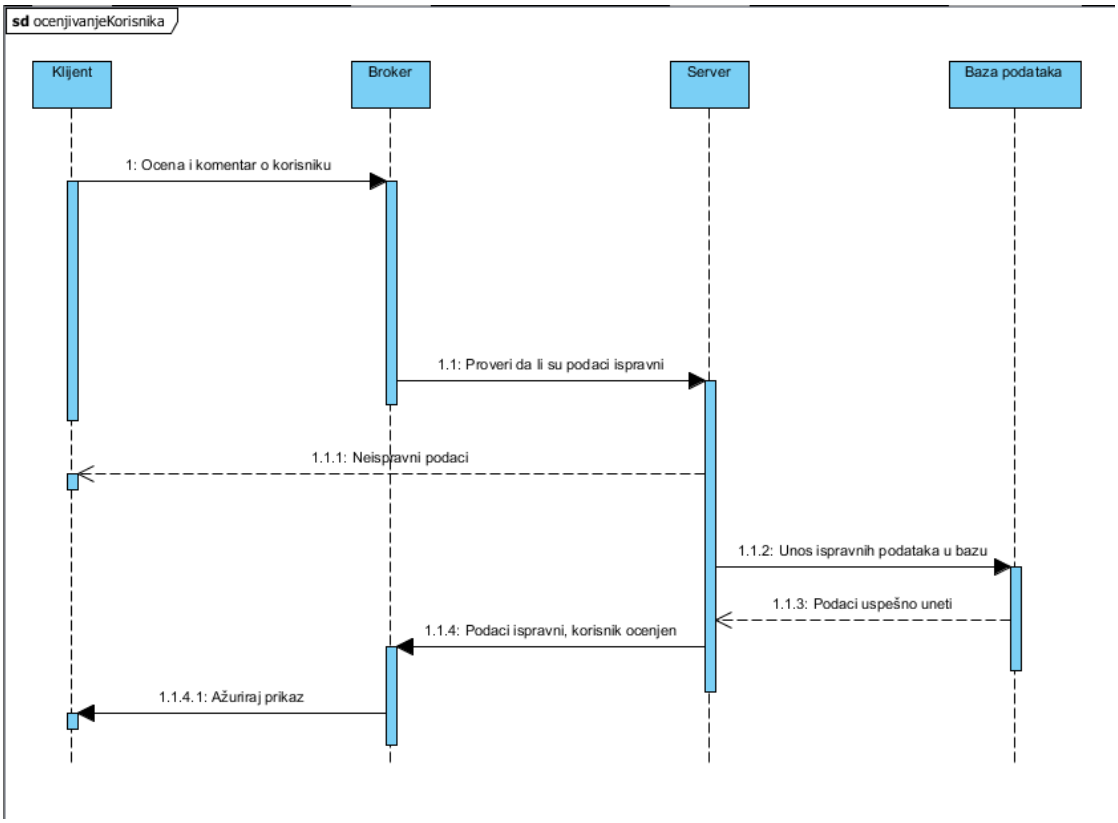
  - Bihevioralni pogled
    - Slučaj postavljanja oglasa i obaveštavanje korisnika



- Slučaj izmene oglasa



- Slučaj ocenjivanja korisnika



4. *Aplikacioni okvir(i)*

   Od tehnologija, koristiću React na frontendu, ASP.NET Web API za serverski deo aplikacije, SQL Server za rad sa bazom podataka, Entity Framework kao ORM alat, a za ostvarivanje real-time komunikacije SignalR.

5. *Analiza arhitekture*

   Potencijalni rizik u razvoju aplikacije TaskIT jeste povećanje broja konkurentnih korisnika i preopterećenje server zahtevima. Jedno od rešenja bi bilo projektovanje sistema tako da podela skladišta u budućnosti bude lako izvodljiva što moze da se postigne težnjom na što većoj nezavisnosti sloja perzistencije i viših slojeva aplikacije.