Arhitektura i Projektovanje Softvera
FAZA1 - Arhitekturni Projekat

TheScientist

student: Miljana Petković, 17870

Sadržaj

Kontekst i cilj projekta

Projekat *TheScientist* je realtime web aplikacija čiji je cilj da korisnicima olakša kolaborativno kreiranje i editovanje naučnih radova. Aplikacija treba da funkcioniše tako što jedan korisnik (vođa/leader tima) kreira rad i njemu pridruži ostale korisnike koji će mu biti saradnici u izradi. Kreiranje projekta podrazumeva kreiranje prazne stranice u koji korisnici kasnije mogu da dodaju sekcije i podsekcije u zavisnosti od potreba određenog tipa rada.

Podrazumeva se da vođa tima ima sve privilegije po pitanju izradu kreiranog naučnog rada. Vođa tima kasnije ostalim korisnicima može da dodeli privilegije vezane za učešće na tom kontretnom naučnom radu (reviewer, editor, coleader). Korisnik sa “reviewer” privilegijom ima mogućnost pregleda i pisanja komentara u radu, korisnik sa “editor” privilegijom ima mogućnost da pored toga uredjuje rad ali promene koje izvrši moraju biti potvrđene od strane jednog od vođa tima.
Svi korisnici u opsegu jednog rada imaju mogućnost medjusobne komunikacije ostavljanjem komentara u radu, kreiranjem anketa vezanih za donošenje odluka o samom radu ili jednostavno putem grupnog chata otvorenog pri svakom kreiranju novog rada.

Ovakvom organizacijom korisnicima će se omogućiti jedinstven prikaz strukture celog rada, čime se olakšava pregled rada, organizacija obaveza i komunikacija korisnika.

Arhitekturni zahtevi

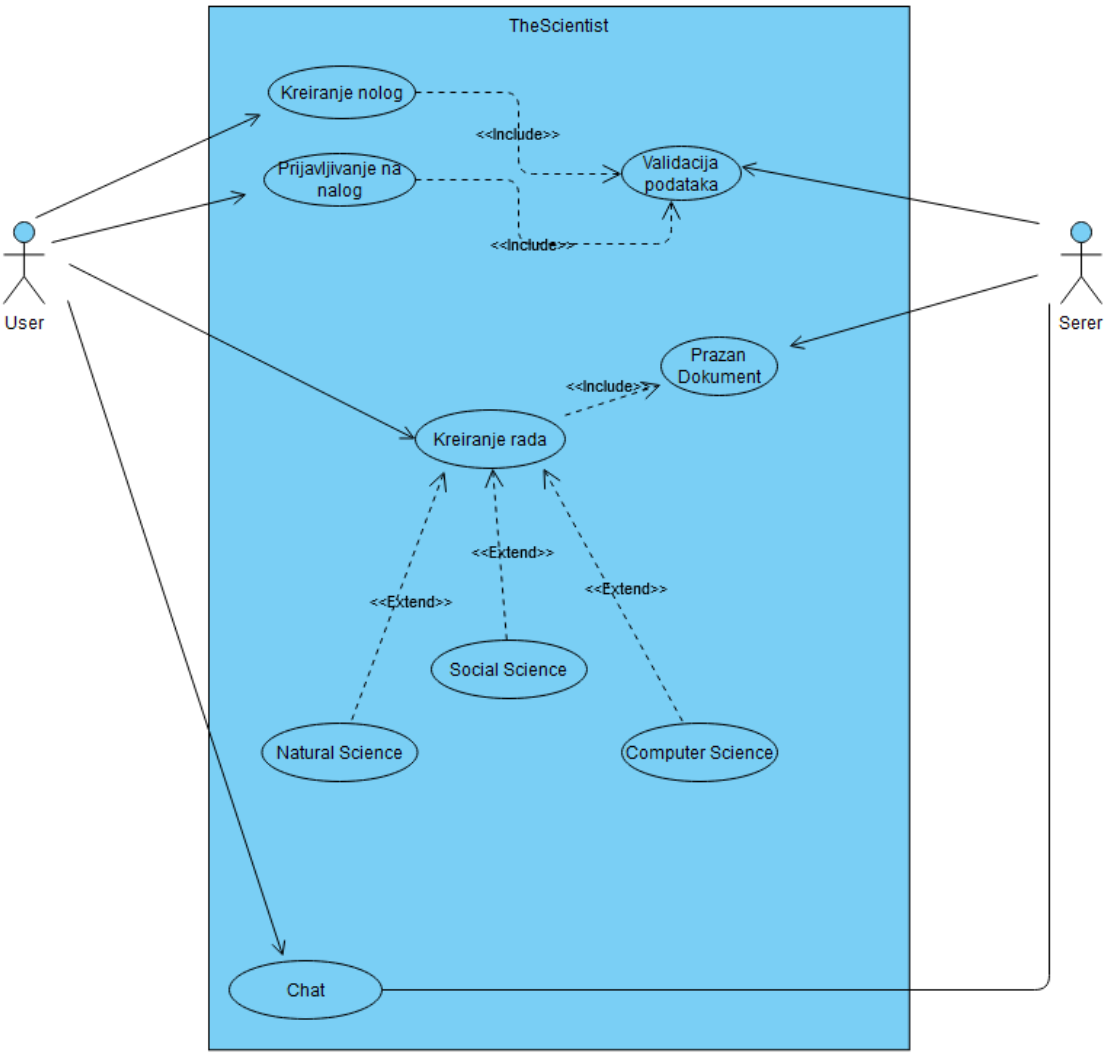
Ovde ćemo definisati arhitekturno značajne slučajeve korišćenja, glavne funkcionalne i ne-funkcionalne zahteve (atributi kvaliteta) i tehnička i poslovna ograničenja vezana za realizaciju projekta *TheScientist* .

Arhitekturno značajni slučajevi korišćenja

Funkcionalni zahtevi sistema web aplikacije *TheScientist*:

- Kreiranje korisničkog naloga, logovanje i editovanje
- Kreiranje/Editovanje naučnog rada
- Dodavanje/Editovanje korisnika kao kolaboratora u radu
- Dodavanje/Editovanje segmenta u rad (svaki segment može da ima podsegmente)
- Čuvanje i pregled obaveza za svaki rad u kom učestvuje korisnik

- Mogućnost dobijanja notifikacija o promenama na projektima
- Mogućnost realtime praćenja promena izvršenih nad dokumentom
- Mogućnost slanja poruka u grupni chat otvoren pri svakom kreiranju novog rada (i dodavanje korisnika u chat sa svakim dodavanjem novog korisnika u rad)
- Mogućnost pisanja komentara u radu i dobijanje notifikacija o komentarima
- Skladištenje podataka



Use Case dijagram - osnovni funkcionalni zahtevi

Ne-funkcionalni zahtevi

Tokom projektovanja sistema cilj je ostvarati sledeće ne-funkcionalne zahteve (atribute kvaliteta):

- Pouzdanost i dostupnost: Potrebno je omogućiti da aplikacija bude dovoljno pouzdana kako nebi došlo do gubitka podataka u slučaju gubitka konekcije sa serverom. Treba omogućiti da aplikacija bude dostupna korisnicima 24h dnevno
- Skalabilnost: Potrebno je obezbediti i adekvatan i efikasan rad permanentnog skladišta podataka sa povećanjem količine podataka koju skladišti, sa povećanjem broja korisnika
- Lakoća korišćenja: Korisnički interfejs treba da bude dovoljno intuitivan za korisnike sistema
- Lakoća testiranja: Potrebno je da sistem izbegava provereno lošu praksu u dizajniranju sistema, kako bi se izbeglo teško održavanje i proširenje sistema
- Sigurnost: Sistem treba poštovati bazične principe kontrole (autorizacije i autentifikacije) naloga igrača, što se odnosi na lične podatke korisnika kojima može pristupiti samo vlasnik naloga i podataka

Arhitekturni dizajn

Ovde ćemo opisati arhitekturne obrasce korišćene u projektovanju arhitekture aplikacije TheScientist, generalna arhitektura i konačno, strukturni i bihevioralni pogled na aplikaciju.

Arhitekturni obrasci

Za projetovanje arhitekture TheScientist aplikacije biće korišeni arhitekturni obrasci: Layered, MVC, Repository i Publish/Subscribe.

- Layered Arhitektura: Arhitekturni obrazac koji forsira podelu strukture aplikacije u slojeve. Svaki sloj u arhitekturi formira apstrakciju oko posla koji treba da se uradi da bi se zadovoljio određeni poslovni zahtev.[1]
  Kod projektovanja aplikacije *TheScientist* biće korišćena troslojna arhitktura, standard za projektovanje web aplikacija. Tri sloja su : klijentski, serverski i sloj podataka(sloj baze podataka).
  - Klijentski sloj: lokalni interfejs koji se koristi za kreiranje, modeliranje, analizu, predstavljanje, izveštavanje i distribuciju različitog sadržaja. Ovaj sloj predstavljaće sloj prikazan korisnicima u Web browseru preko kog oni mogu da interaguju sa ostalim slojevima aplikacije.
  - Serverski sloj: Zahtevi primljeni sa klijentskog sloja se prihvataju i šalju dalje odgovarajućem agentu. Ovaj sloj će klijentu omogućiti sinhronu komunikaciju preko RESTful API poziva i asinhronu komunikaciju pomoću message brokera.
  - Sloj baze podataka: Tokom rada ovako projektovane aplikacije (komunikacija klijenta i srvera) nastaće određeni podaci koje će biti

neophodno očuvati radi omogućavanja normalne funkcije programa. Ovaj sloj će omogućiti trajnu perzistenciju ovako nastalih podataka.

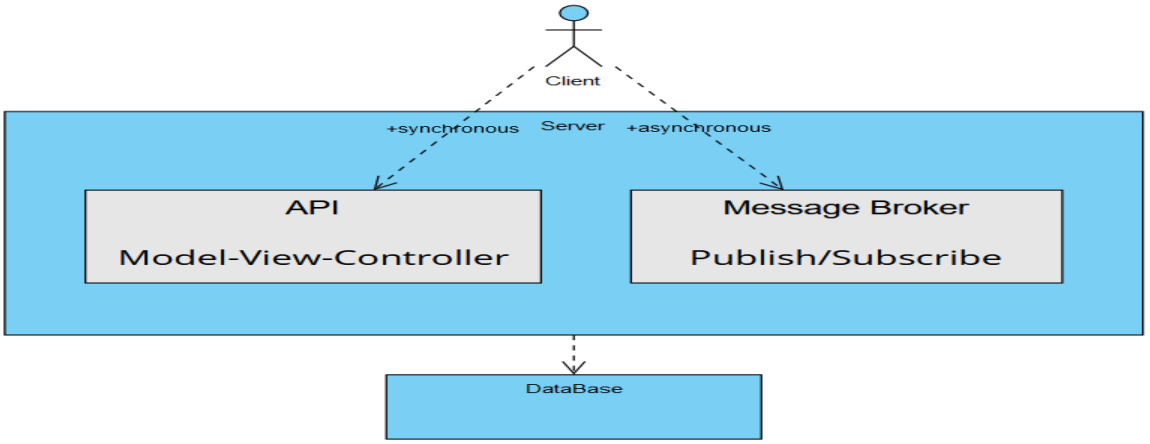
- MVC (Model-View-Controller): Arhitekturni obrazac koji odvaja aplikaciju na tri glavne logičke komponente: model, pogled i kontroler. Svaka od ovih komponenti rukuje specifičnim razvojnim aspektima aplikacije.[2]
  Ovaj obrazac u aplikaciji *TheScientist* će biti upotrebljen za projektovanje RESTful API-ja (sinhrona komunikacija sa klijentom).
  -Model: Centralna komponenta obrasca, dinamička struktura podataka aplikacije, nezavisna od korisničkog interfejsa. Ova komponenta se bavi perzistencijom domenskih klasa.
  -View: Bilo koj grafički prikaz podataka koji su projektovani u modelu. Ova komponenta u aplikaciji *TheScientist* neće biti implementirana tako da serverska strana kreira i vraća delove interfejsa, već će se interfejs implementirati korišćenjem JavaScript frameworka Angular, koj će od servera dobijati podatke u JSON formatu.
  -Contreoller: Dobijeni input pretvara u komande koje se koriste za model ili view. Ova komponenta obezbeđuje korišćenje usluga samog RESTful APIja, čije funkcije poziva klijent, funkcija zatim vrši obradu ulaznih podataka i podataka iz modela i tako kreira izlaz u vidu JSON objekata koje klijent lako može da parsira i koristi za svoje potrebe.

- Repository: Arhitekturni obrazac koji aplikaciji omogućava podelu na repozitorijume, klase ili komponente koji obuhvataju logiku potrebnu za pristup izvorima podataka.Oni centralizuju zajedničku funkcionalnost pristupa podacima, obezbeđujući bolju mogućnost održavanja i razdvajajući infrastrukturu ili tehnologiju koja se koristi za pristup bazama podataka od sloja modela domena.[3]
  Ovaj obrazac biće korišćen radi postizanja postojanja interfejsa između centralizovanog skladišta (baza podataka) i mapiranja tih podataka(za koji će biti korišćen Entity Framework ORM). Klijent će graditi upite koje šalje u skladi[te za odgovore. Repozitorijumi će enkaplsulirati skupove objekata iz baze podataka i operacije koje se mogu izvršiti nad njima, obezbeđujući put koji je bliži sloju perzistencije.

- Publish/Subscribe: Arhitekturni obrazac koji obezbedjujej okvir za razmenu poruka izmedju onoga ko ih proizvodi (publisher) i pretplatnika (subscriber). Publisher i Subscriber se oslanjaju na message brokera, koji poruke prenosi od proizvođača do pretplatnika.[4]
  Ovaj obrazac biće korišćen radi predavanja asinhronih poruka klijentu od strane servera, putem message broker komponente.

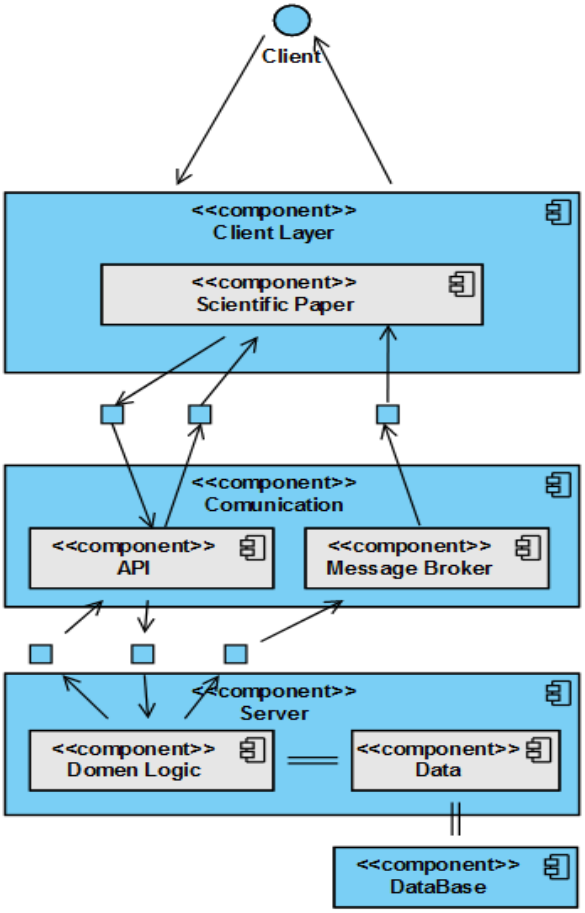
Generalna Arhitektura

Na slici ispod je prikazana generalna arhitektura *TheScientist* aplikacije:



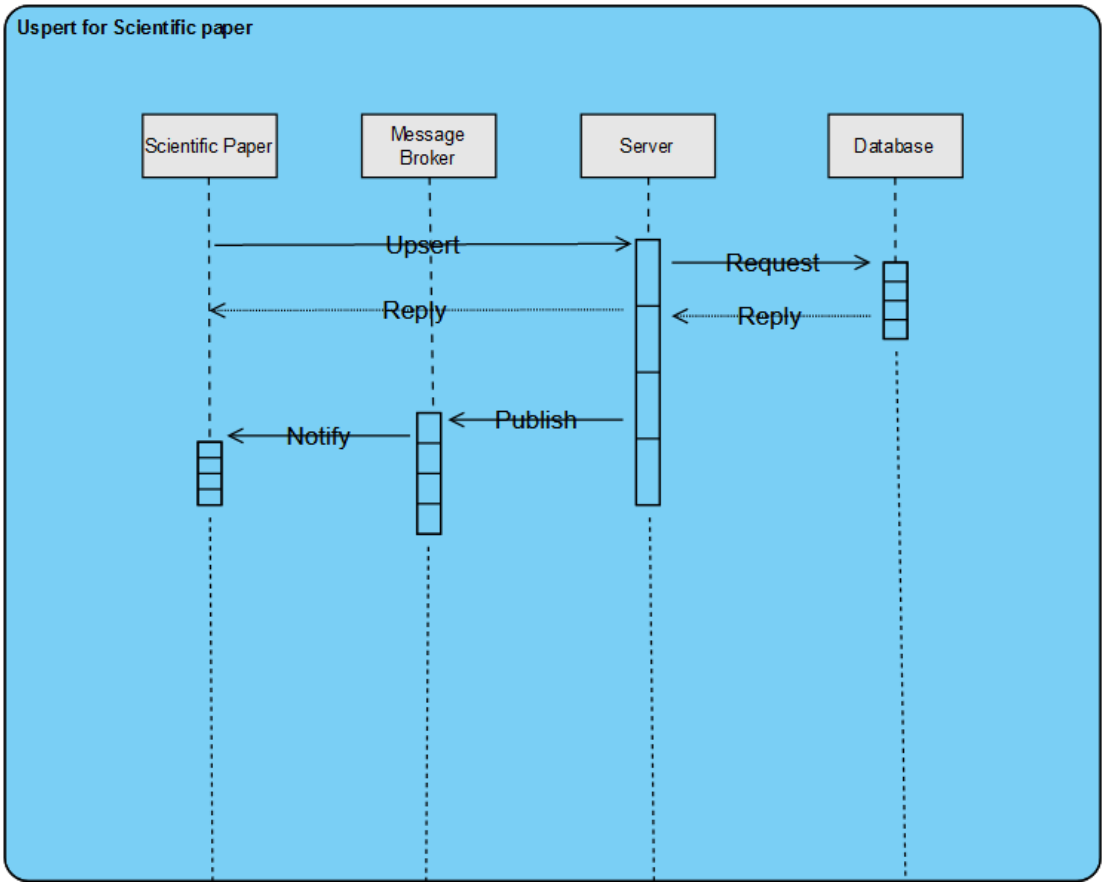
Strukturni pogled

Na slici ispod prikazana je strukturu komponenti *TheScientist* aplikacije, njihova povezanost i medjusobne komunikacije.



Bihevioralni pogled

Na slici ispod prikazan je primer bihevioralnog pogleda *TheScientist*, sa specifičan primer najvažniju akcije Insert/Update naučnog rada. Ukoliko korisnik promeni nešto na naučnom radu potrebno je da se ta promena sačuva u bazi podataka (Data base layer) i da se obavesti message broker koji će tu promenu prikazati svim korisnicima koji posmetraju taj rad.



Implementacija

Ovde ćemo definisati biblioteke, komponente i okvire koji će biti korišćeni u realizaciji projekta *TheScientist*.

- **Angular** - klijentska strana aplikacije
  Razvojna platforma, izgrađena na Typescript-u, uključuje kolekciju dobro integrisanih biblioteka koje pokrivaju širok spektar funkcija, uključujući rutiranje, upravljanje obrascima, komunikaciju klijent-server itd. [5]
- **ASP.NET Core** - serverska strana aplikacije
  Open-source okvir za kreiranje modernih aplikacija. Dizajniran je tako da obezbedi optimizovani razvojni okvir za aplikacije koje se postavljaju u oblak ili se pokreću lokalno. [6]
- **MS SQL Server** - relaciona baza podataka
- **SIgnal R** - biblioteka za uspostavljanje real-time komunikacije
- **Entity Framework** - okvir za objektno-relaciono mapiranje

Refecence

[1]https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html

[2]https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet

[3]https://learn.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design

[4]https://ably.com/topic/pub-sub

[5]https://angular.io/guide/what-is-angular

[6]https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet