FP

Find Professor

Specifikacija zahteva

Verzija 1.0

Pregled izmena

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | **Verzija** | **Opis** | **Autor** |
| 20.04.2023. | 1.0 | Inicijalna verzija | Teodora, Nikola, Jana |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Sadržaj

1. Cilj dokumenta 5

2. Opseg dokumenta 5

3. Reference 5

4. Predstavljanje arhitekture 5

5. Ciljevi i ograničenja arhitekture 5

6. Pogled na slučajeve korišćenja 5

6.1 Dijagrami slučajeva korišćenja 6

6.2 Kratak opis slučajeva korišćenja 8

6.2.1 Izbor jezika 8

6.2.2 Pregled osnovnih podataka o laboratoriji 8

6.2.3 Pregled spiska članova 8

6.2.4 Pregled podataka o određenom članu laboratorije 8

6.2.5 Pregled publikacija po autoru 8

6.2.6 Pregled publikacija po tipu 8

6.2.7 Pregled publikacija po godini 8

6.2.8 Pregled spiska projekata 8

6.2.9 Pregled podataka o određenom projektu 8

6.2.10 Prijavljivanje 9

6.2.11 Ažuriranje podataka o članu 9

6.2.12 Dodavanje nove publikacije 9

6.2.13 Brisanje postojeće publikacije 9

6.2.14 Ažuriranje osnovnih podataka o laboratoriji 9

6.2.15 Kreiranje novog člana 9

6.2.16 Brisanje postojećeg člana 9

6.2.17 Arhiviranje postojećeg člana 9

6.2.18 Kreiranje projekta i postavljanje vođe 9

6.2.19 Ažuriranje podataka o projektu 9

7. Pogled na logičku arhitekturu sistema 9

7.1 Pregled arhitekture – organizacija paketa i podsistema u slojeve 10

7.1.1 Korisnički interfejs 10

7.1.2 Aplikaciona logika 10

7.1.3 Pristup podacima 10

7.1.4 HTML 10

7.1.5 PHP 11

7.1.6 MySQL 11

8. Pogled na procese 11

8.1 Procesi 11

8.1.1 Web čitač 11

8.1.2 Web server 11

8.1.3 PHP-CGI 11

8.1.4 MySQL Server 12

9. Pogled na raspoređivanje sistema 12

9.1 Klijent 12

9.2 Web server 12

9.3 DBMS server 12

10. Pogled na implementaciju sistema 12

10.1 Model domena 12

10.2 Šema baze podataka 13

10.3 Komponente sistema 14

10.3.1 Komponente korisničkog interfejsa 14

10.3.2 Komponente aplikacione logike 14

10.3.3 Komponente za pristup podacima 15

11. Performanse 16

12. Kvalitet 16

Arhitekturni projekat

# Cilj dokumenta

Cilj ovog dokumenta je detaljni opis arhitekture FindProfessor portala.

# Opseg dokumenta

Dokument se odnosi na FindProffesor portal koji ce biti razvijen od strane tima JTN. Namena sistema za pretrazivanje profesora u cilju zadovoljavanja potreba korisnika

# Reference

Spisak korišćene literature:

1. FindProfessor – Predlog projekta, JTN-FP-01, V1.0, 2023, JTN.
2. FindProfessor – Planirani raspored aktivnosti na projektu, V1.0, 2023, JTN.
3. FindProfessor – Plan realizacije projekta, V1.0, 2023, JTN.
4. FindProfessor – Vizija sistema, V1.0, 2023, JTN.
5. FindProfessor – Specifikacija zahteva, V1.0, 2023, JTN.

# Predstavljanje arhitekture

Arhitektura sistema u dokumentu je prikazana kao serija pogleda na sistem: pogled na slučajeve korišćenja, pogled na logičku arhitekturu sistema, pogled na procese, pogled na razmeštaj komponenti sistema i pogled na implementaciju. Ovi pogledi su predstavljeni odgovarajućim UML dijagramima.

# Ciljevi i ograničenja arhitekture

Ključni zahtevi i sistemska ograničenja koja imaju značajan uticaj na izbor arhitekture i projektovanje sistema su:

1. FindProfessor aplikacija će biti implementirana kao Web aplikacija zasnovana na React-u i .NET-u.
2. Klijentski deo FindProfessor aplikacije će biti optimizovan za sledeće Web čitače: Microsoft Edge, Opera, Firefox i Google Chrome.
3. Svi zahtevi u pogledu performansi dati u moraju biti uzeti u obzir pri izboru arhitekture i razvoju sistema.
4. Klasifikaciju publikacija treba standardizovati sa postojećom tipologijom propisanom od strane Ministarstva za nauku.

# Pogled na slučajeve korišćenja

U ovom odeljku je dat pogled na slučajeve korišćenja definisane u specifikaciji zahteva [5].

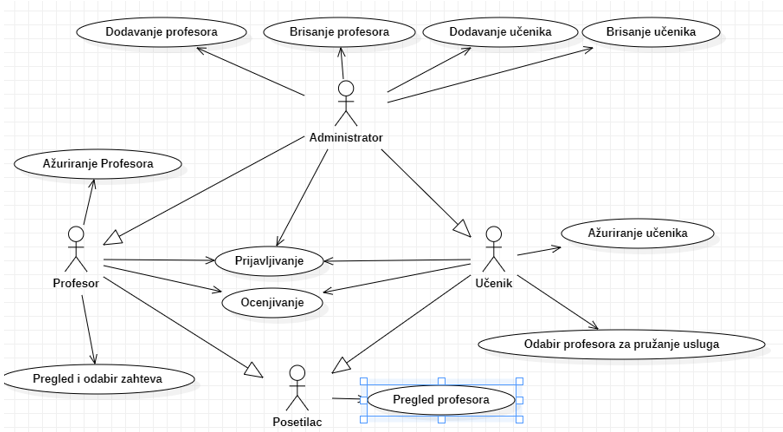
Slučajevi korišćenja FindProfessor aplikacije su:

* Prijavljivanje
* Odabir profesora
* Azuriranje ucenika
* Ocenjivanje profesora
* Dodavanje profesora
* Brisanje profesora
* Dodavanje ucenika
* Brisanje ucenika
* Azuriranje profesora
* Upravljanje registracijom
  + Odabir opcije za registrovanje izmedju ucenika i profesora
  + Unos email-a
  + Unos korisnicke lozinke
  + Potvrda korisnicke lozinke
* Upravljanje zahtevima
  + Odabir zahteva
  + Pregled zahteva
* Pregled profesora
  + Pregled licnih podataka
  + Pregled recenzija
  + Pregled ocena

Ove slučajevi korišćenja mogu da iniciraju posetilac portala, ucenik, profesor i administrator.

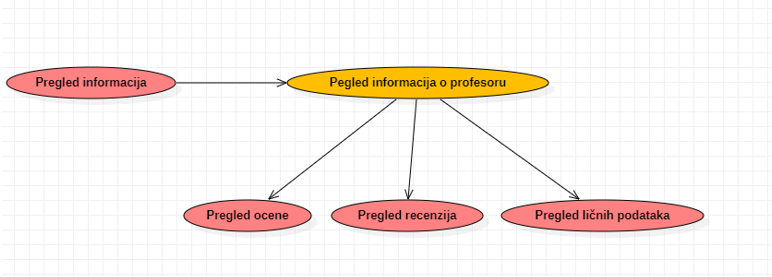
## Dijagrami slučajeva korišćenja

Osnovni UML dijagram koji prikazuje korisnike i slučajeve korišćenja FindProfessor aplikacije prikazan je na sledećoj slici:

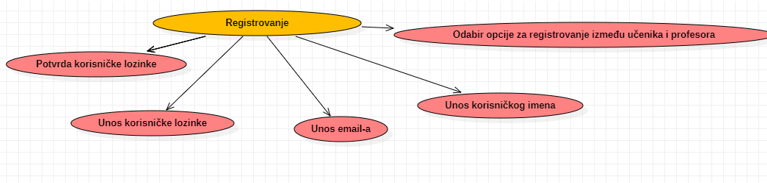


Slučajevi korišćenja *pregled informacija o profesorima, registrovanje i pregled i odabir zahteva* obuhvataju složenije radnje koje se mogu razložiti dalje razložiti na pojedinačne slučajeve korišćenja.

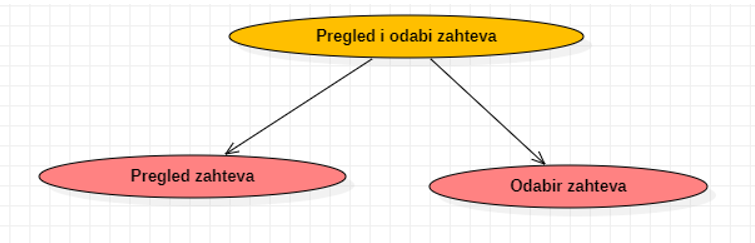
Detaljni UML dijagram za slučaj korišćenja *pregled informacija o profesorima* je prikazan na sledećoj slici:



Detaljni UML dijagram za slučaj korišćenja *registrovanje* je prikazan na sledecoj slici



Detaljni UML dijagram za slučaj korišćenja *pregled i odabir zahteva* je prikazan na sledećoj slici:



## Kratak opis slučajeva korišćenja

### Pregled profesora

Kratak opis: Prikaz stranice portala sa osnovnih infromacija o profesorima.

Akteri koji iniciraju slucaj koriscenja: Posetilac portala, Ucenici, Administrator

### Registrovanje

Prikaz stranice sa formom za prijavljivanje ili registrovanje i korisnik unosi odredjene informacije da bi se registrovao.

Akteri koji iniciraju slucaj koriscenja: Posetilac portala, Ucenik, Profesor

### Pregled zahteva

Kratak opis: Prikaz stranice portala sa svim pristiglim zahtevima od strane učenika.

Akteri koji iniciraju slucaj koriscenja: Profesor

### Odabir zahteva

Kratak opis: Prikaz stranice sa informacijama o svim pristiglim zahtevima i odabir odgovarajucih zahteva.

Akteri koji iniciraju slucaj koriscenja su: Profesor

### Odabir profesora

Kratak opis: Prikaz stranice sa svim registrovanim profesorima i odabir jednog od njih

Akteri koji iniciraju slucaj koriscenja: Ucenik

### Azuriranje ucenika

Kratak opis: Prikaz stranice sa opcijom za azuriranje podataka o uceniku, gde ucenik menja svoje podatke.

Akteri koji iniciraju slucaj koriscenja: Ucenik

### Prijavljivanje

Kratak opis: Kada profesor održi čas, učenik može da oceni i ostavi recenziju profesora.

Akteri koji inciraju slucaj koriscenja: Ucenik

### Ocenjivanje profesora

Kratak opis: Prijavljivanje korisnika na portal u cilju pristupa specifičnim funkcijama koje zahtevaju autorizaciju.

Akteri koji iniciraju slučaj korišćenja: Član laboratorije, Šef laboratorije, Vođa projekta, Administrator.

### Dodavanje profesora

Kratak opis: Dodavanje korisnickog naloga profeosra.

Akteri koji iniciraju slucaj koriscenja: Profesor, Administrator.

### Brisanje profesora

Kratak opis: Administrator brise vec postojeceg profesora iz nekog razloga..

Akteri koji iniciraju slučaj korišćenja: Profesor, Administrator

### Dodavanje ucenika

Kratak opis: Dodavanje korisnickog naloga ucenika.

Akteri koji iniciraju slučaj korišćenja: Ucenik i Administrator.

### Brisanje ucenika

Kratak opis: Administrator brise vec postojeceg ucenika iz nekog razloga.

Akteri koji iniciraju slučaj korišćenja: Ucenik, Administrator.

### Azuriranje profesora

Kratak opis: Ukoliko profesor želi da izmeni neke podatke o sebi ima mogućnost ažuriranja svojih podataka.

Akteri koji iniciraju slučaj korišćenja: Profesor.

# Pogled na logičku arhitekturu sistema

U ovom odeljku je dat pregled logičke arhitekture sistema. Ovaj pogled sadrži opis najznačajnijih klasa, njihove organizacije u pakete i podsisteme, i organizacija podsistema u slojeve. U cilju opisivanja dinamičkih aspekata arhitekture, ovaj odeljak može da uključi opise realizacije najznačajnijih slučajeva korišćenja. Da bi se ilustrovala veza između arhitekturno značajnih klasa, podsistema, paketa ili slojeva moguće je uključiti i odgovarajuće dijagrame klasa.

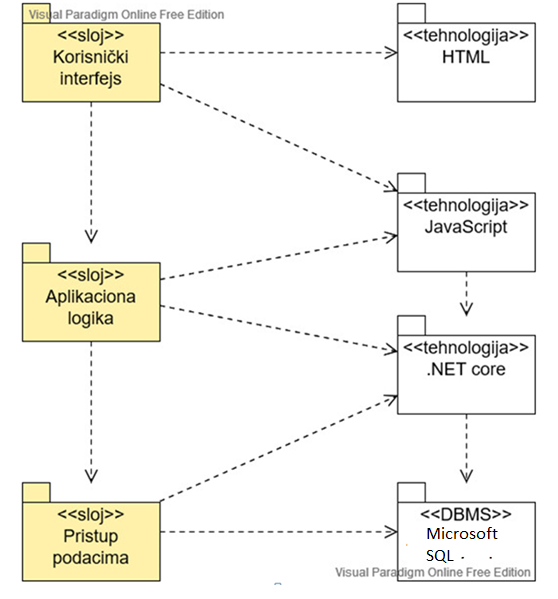
Logički pogled na FindProfessor aplikaciju obuhvata 3 glavna paketa: Korisnički interfejs, Aplikaciona logika, Pristup podacima.

Paket *Korisnički interfejs* sadrži Web stranice, JavaScript(ReactJS) i multimedijalni sadržaj koji realizuju grafički dizajn i forme preko kojih korisnici sistema komuniciraju sa sistemom.

Paket *Aplikaciona logika* predstavlja srednji sloj sistema koji sadrži JavaScript skripte zadužene za realizaciju funkcionalnosti specifičnih za domen sistema koji se razvija.

Paket *Pristup podacima* sadrži .NET CRUD operacije koje predstavljaja interfejs za pristup, dodavanje i ažuriranje podataka koji se čuvaju u bazi podataka.

## Pregled arhitekture – organizacija paketa i podsistema u slojeve



### Korisnički interfejs

sloj

Ovaj sloj realizuje korisnički interfejs portala. U njemu su sadržane sve HTML, multimedijalni sadržaji i ReactJS skripte koje generišu HTML stranice preko kojih korisnici komuniciraju sa sistemom.

Sloj korisničkog interfejsa zavisi od sloja aplikacione logike, kao i paketa HTML i ReactJS.

### Aplikaciona logika

sloj

Sloj aplikacione logike je srednji sloj u troslojnoj arhitekturi FindProfessor aplikacije. Sadrži JavaScript skripte koje realizuju funkcionalnost karakterističnu za domen primene portala i uspostavljaju vezu između korisničkog interfejsa i sloja za pristup podacima.

Ovaj sloj zavisi od sloja za pristup podacima i JavaScript paketa.

### Pristup podacima

sloj

Sloj za pristup podacima se nalazi na dnu troslojne arhitekture i sadrži .NET skripta zadužene za pribavljanje, dodavanje i ažuriranje podataka koji se čuvaju u Microsoft SQL localDb.

Ovaj sloj ne zavisi od drugih slojeva, ali je zavisan od JavaScript paketa i Microsoft SQL localDB.

### HTML

tehnologija

Tehnologija HTML-a definiše gradivne elemente stranica koje se prikazuju u Web čitaču i koje omogućavaju prikaz formatiranih informacija i realizaciju formi za unos i ažuriranje podataka.

### ReactJS

tehnologija

Tehnologija ReactJS-a obezbeđuje mehanizam za pisanje skripti koje će se izvršavati na klijentskoj strani, koje će dinamički generisati HTML elemente i povezati događaje za HTML elemente.

### ASP.NET

tehnologija

Tehnologija ASP.NET CORE obezbeđuje mehanizam za pisanje i izvršavanje koda na strani servera. Ovaj kod može generisati zahteve koji pristupaju bazi podataka u cilju pribavljanja, unosa i ažuriranja podataka.

### Microsoft SQL

DBMS

Microsoft SQL predstavlja sistem za upravljanje bazama podataka koji će se koristiti za realizaciju FindProfessor aplikacije.

# Pogled na procese

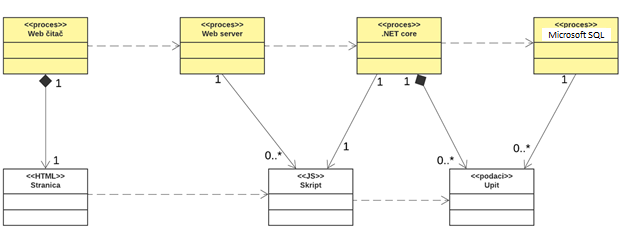
U ovom odeljku je sadržan pogled na procesnu arhitekturu sistema. Ovaj opis treba da sadrži specifikaciju različitih zadataka (procesa i niti) uključenih u rad sistema. Takođe je potrebno dati dijagrame koji pokazuju njihovu interakciju i konfiguraciju. Dodela objekata i klasa na određene zadatke takođe spada u opis procesne arhitekture.

Web aplikacije zasnovane na .NET-u imaju relativno jednostavan procesni model koji je u potpunosti pod kontrolom Web servera. Sa stanovišta projektanta .NET Web aplikacije nije potrebno voditi računa o načinu rada Web servera i načinu izvršavanja skripti.

Ilustracije radi u nastavku je dat opis procesa uključenih u izvršenje FindProfessor aplikacije kao Web aplikacije.

## Procesi

Na sledećem UML dijagramu klasa prikazani su procesi koji učestvuju u izvršenju FindProfessor aplikacije. Dijagram je opšteg tipa i može se primeniti na bilo koju Web aplikaciju zasnovanu na Microsoft SQL bazi podataka.



### Web čitač

Web čitač je proces koji izvršava funkcionalnost aplikacije za prikaz HTML stranica dobijenih od nekog Web servera. U najopštijem slučaju Web čitač u jednom trenutku može da prikazuje samo jednu HTML stranicu.

Web čitač zavisi od Web servera koji generiše i vraća odgovarajuću HTML stranicu na zahtev.

### Web server

Web server je proces koji izvršava funkcionalnost opsluživanja zahteva prispelih sa više Web čitača. Ukoliko je zahtevana stranica JavaScript skript, Web server inicira izvršenje ASP .NET CORE procesa koji obrađuje odgovarajući skript i generiše sadržaj koji se vraća čitaču. Web server može paralelno da inicira veći broj ASP .NET CORE procesa.

### ASP.NET

*ASP.NET*  proces obavlja posao obrade zadatog JavaScript skripta i generiše odgovarajući tekstualni sadržaj koji Web server šalje Web čitaču. Za izvršenje JavaScript skripta ovaj proces može da zahteva usluge SQL servera-a. Komunikacija između JavaScript i SQL servera se obavlja preko prosleđivanja upita i vraćanja rezultat.

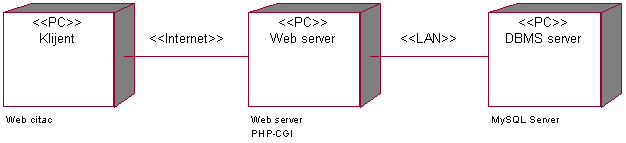
### Microsoft SQL Server

Microsoft SQL Server je proces koji izvršava funkcionalnost MySQL sistema za upravljanje bazama podataka. Ovaj proces može konkurentno da prihvati određen broj upita, izvrši ih nad bazom podataka i vrati rezultate procesu koji je upite postavio.

# Pogled na raspoređivanje sistema

Pogled na raspoređivanje sistema prikazuje različite fizičke čvorove za najopštiju konfiguraciju sistema. Fizičkim čvorovima koji predstavljaju procesore vrši se dodeljivanje identifikovanih procesa.

Na sledećoj slici dat je UML dijagram raspoređivanja FindProfessor aplikacije.



## Klijent

Pristup FindProfessor aplikaciji se obavlja preko klijentskih računara na kojima se izvršava Web čitač. Za povezivanje između klijenta i Web servera koristi se Internet infrastruktura tako da nema ograničenja u pogledu lokacije klijenta.

## Web server

Računar na kome se izvršava Web server opslužuje više klijenata koji pristupaju preko Interneta. Pored osnovnog procesa koji realizuje funkcionalnost Web servera, na ovom računaru mogu da se izvršavaju i procesi ASP .NET CORE-a koji vrše obradu zadatih JavaScript skripti. U najopštioj konfiguraciji DBMS se izvršava na cloud-u.

## DBMS server

DBMS server je računar na kome se izvršava AzureDataStudio proces koji realizuje funkcionalnost sistema za upravljanje bazama podataka. Zbog sigurnosti podataka koji se na ovom računaru čuvaju pristup bazi je ograničen samo na računare cija je ip adresa odobrena na AzureDataStudio serveru.

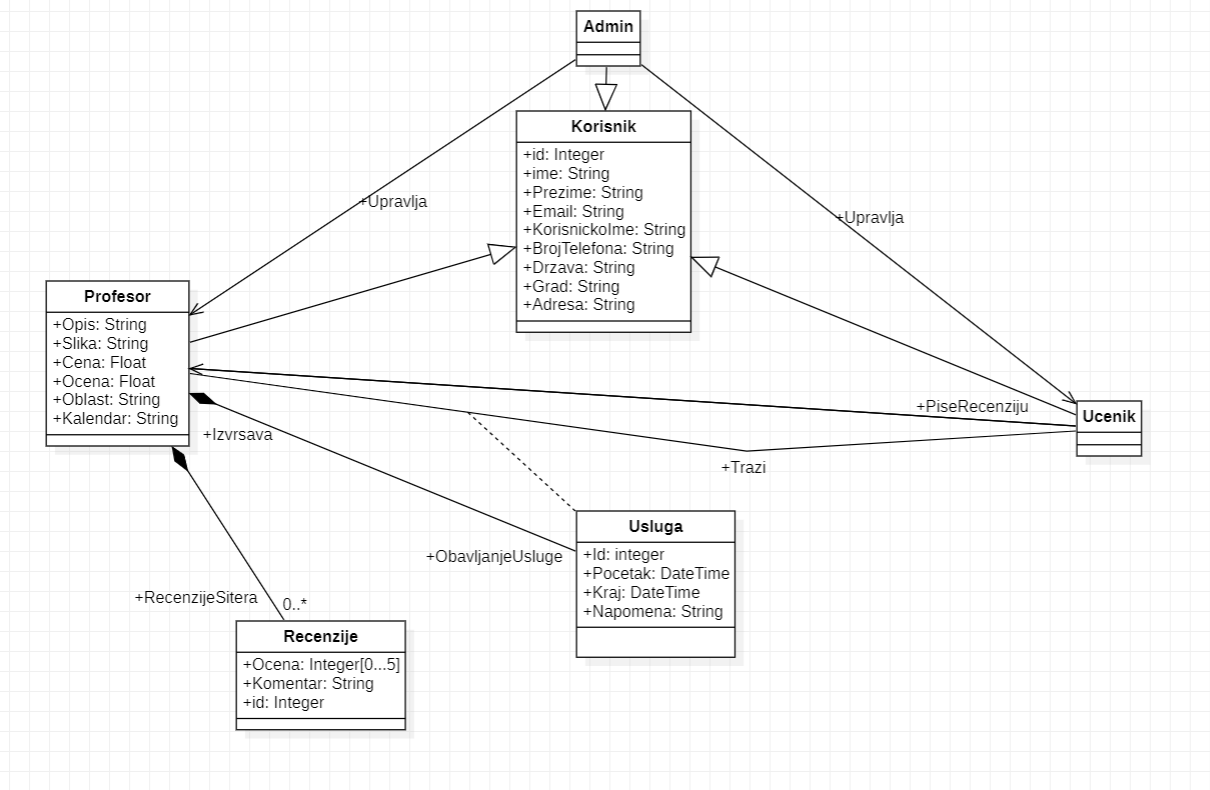
# Pogled na implementaciju sistema

Pogled na implementaciju prikazuje različite aspekte bitne za implementaciju sistema. U slučaju FindProfessor aplikacije ovaj odeljak sadrži model domena, šemu baze podataka i prikaz komponenti sistema razvrstanih u ranije identifikovane pakete.

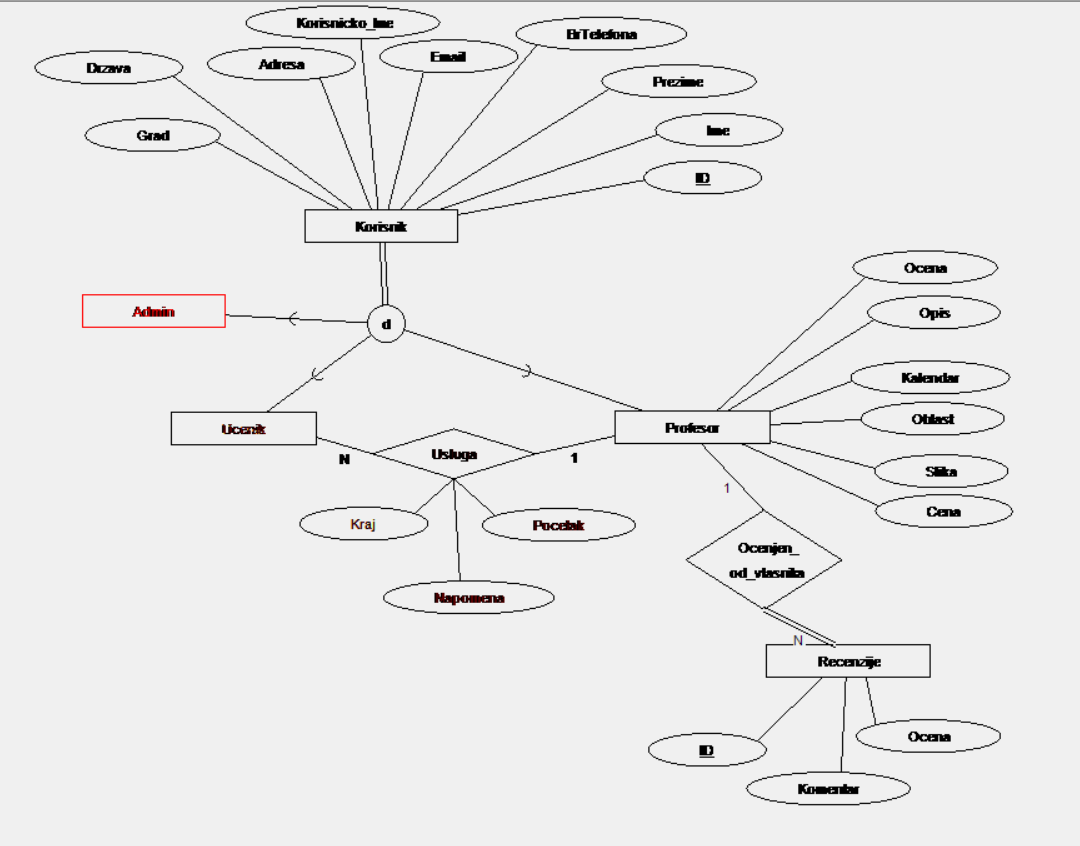
## Model domena

Model domena za koji se FindProfessor aplikacije projektuje je ilustrovan UML dijagramom klasa. U njemu su prikazane domenske klase, neki od njihovih atributa, kao i veze koje se mogu identifikovati između njih.

Model domena predstavlja osnovu za projektovanje baze podataka, ali i identifikaciju nekih od komponenti (ReactJS skripti) koje će biti implementirane.



## Šema baze podataka

Detaljna šema baze podataka je prikazana na sledećem dijagramu. Baza podataka i dijagram su kreirani korišćenjem *SharpEER programa*. 

## Komponente sistema

Komponente sistema FindProfessor aplikacije su ReactJS skripte čiji će pregled biti dat po arhitekturnim slojevima. Za ilustraciju će biti korišćeni UML dijagrami komponenti, ali i dijagrami klasa. U slučajevima gde je ReactJS skripta prikazana kao klasa atributa koji predstavljaju ulazne podatke koji se uzimaju iz GET ili POST dela HTTP poruke, dok metodi predstavljaju funkcije definisane u okviru skripta.

### Komponente korisničkog interfejsa

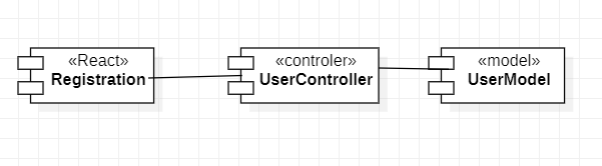
Dizajn korisničkog interfejsa je obuhvaćen dvema komponentama:



Komponenta **index.js** je implementira stranicu portala čiji sadržaj može da varira od parametra koji joj se proslede u zahtevu.

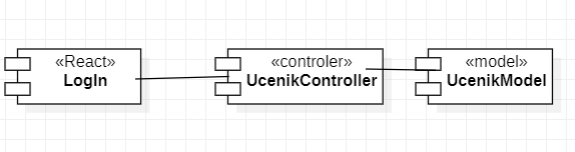
Komponenta **style.css** predstavlja opis stilova za pojedine HTML elemente koji se javljaju na različitim stranicama.

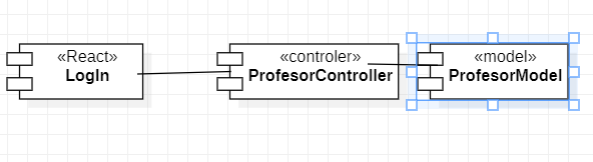
**Registracija:**



Potencijalni korisnik portala na stranici za kreiranje naloga unosi svoje podatke. ReactJS skripta prosleđuje te podatke User Controlleru za registraciju koji ih prosleđuje User Modelu koji vrši njihovu proveru i vraća korisnika na stranicu za prijavljivanje ili ga obaveštava o nevalidnosti određenih podataka. Te podatke pokuplja ReactJS i prosleđuje ih korisniku. Ukoliko je registracija uspešna korisnik se prosleđuje na portal kao ulogovani korisnik.

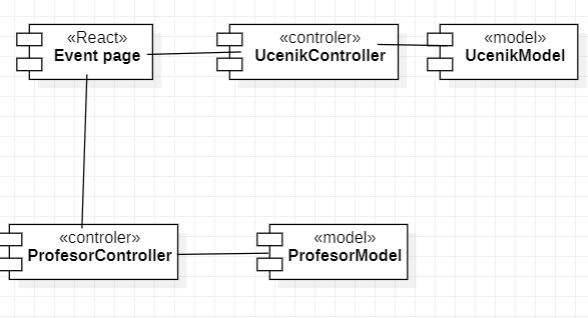
**Prijavljivanje:**





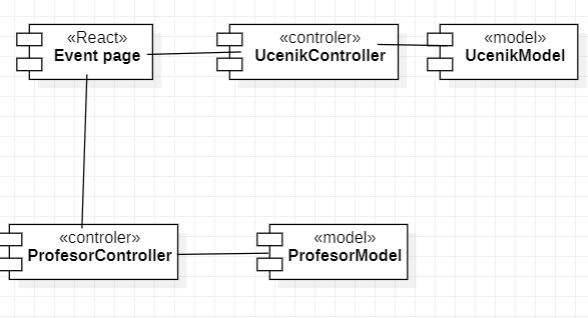
Korisnik na Login stranici popunjava formu sa svojim podacima za prijavu. ReactJS te podatke prihvata i prosleđuje Ucenik/Profesor Controlleru koji ih dalje prosleđuje Ucenik/Profesor Modelu koji vrši proveru heširanih podataka kao što su korisnicko ime i password i ukoliko je prijava uspešna, tu informaciju pokuplja ReactJS koji prosleđuje korisnika na portal kao ulogovanog korisnika. Ukoliko je prijava neuspešna, ReactJS skripta prikazuje tu informaciju korisniku.

**Ocenjivanje profesora:**

****

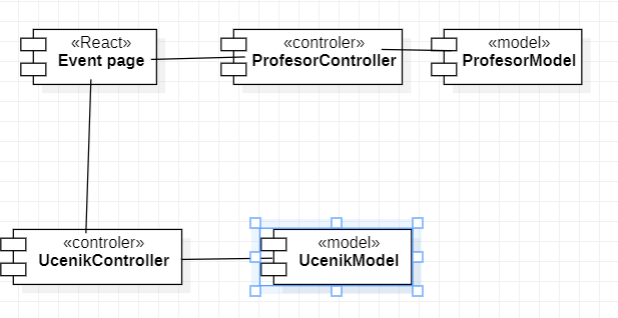
Nakon što se svaka uspesno pruzena usluga obavi, biće omogućeno da ucenik oceni profesora. Cilj ocenjivanja je da se postigne određeni nivo poverenja o korisniku i da se poboljša korisničko iskustvo. Ocene korisnika biće dostupne i javne na stranici korisničkog profila, da bi ih pored profesora, videli i naivni korisnici, tj. posetioci portala.

**Odabir profesora:**



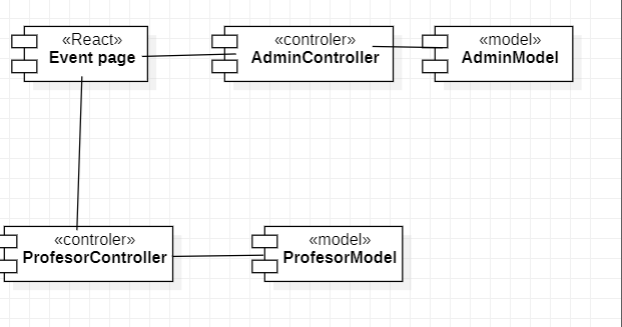
Ucenik ima mogucnost odabira profesora na stranici web portala. Ne postoji neki, unapred definisani sablon prilikom odabira profesora, vec ce se odabir profesora vrsiti na osnovu potreba ucenika. U pomoc ce pristici razlicit broj informacija o profesoru (ocene profesora,komentari o profesoru, prethodno iskustvo...).

**Odabir zahteva:**



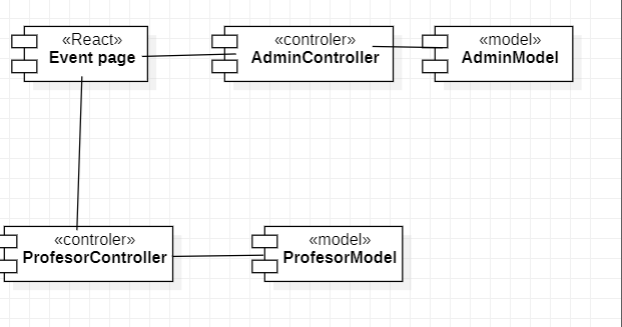
Kada profesor dobije zahtev, ukoliko njemu odgovara dati termin on ce moci da ga prihvati.

**Brisanje korisnika:**



Na adminovoj stranici se nalazi kontrolna tabla sa svim prijavljenim profesorima, koji su ranije bili dodati od strane administratora. Ukoliko profesor bude neaktivan na duzi vremenski period, posao administratora jeste da obrise tog profesora. Ova akcija se prosledjuje dalje Profesor Controller-u.

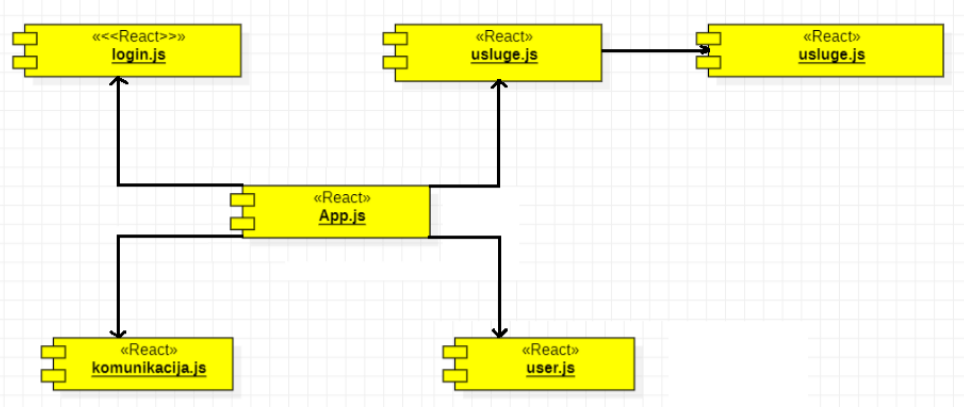
**Dodavanje profesora:**

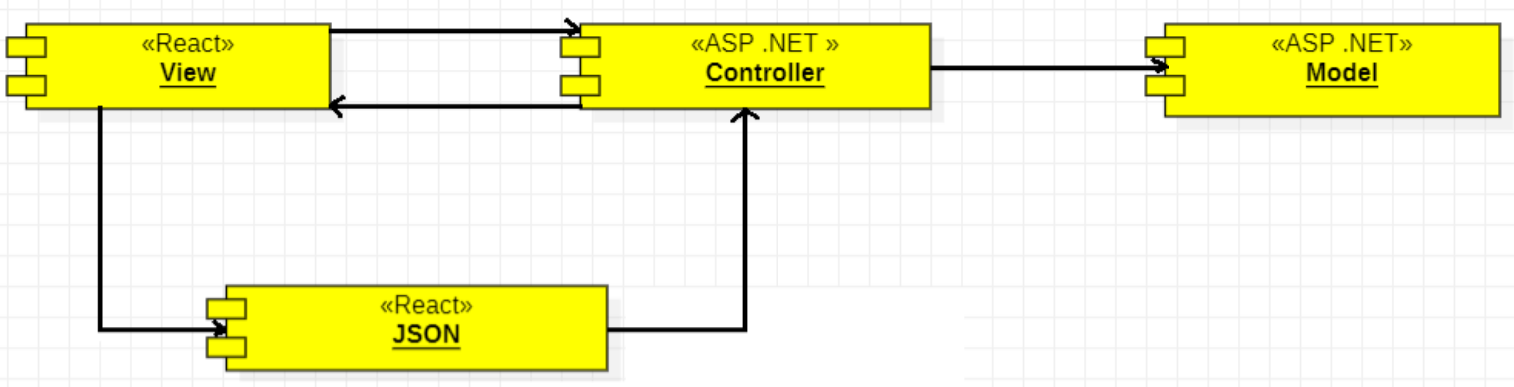


Prilikom registracije ukoliko naivan korisnik(posetilac sajta) izabere opciju za registrovanja za profesora, zahtev se salje adminu da se taj zahtev prihvati.Kada admin prihvati zahtev, dalje se ta obavljena akcija prosledjuje Profesor Controlleru, i sada korisnik koji se registrovao postaje profesor.

### Komponente aplikacione logike

Komponente koje realizuju domen problema se uključuju isključivo preko **App.js** komponente korisničkog interfejsa. Na taj način zadržavaju sva podešavanja stila definisana u ovom skriptu. Na sledećem dijagramu su prikazane komponente ovog sloja i njihove međusobne zavisnosti:



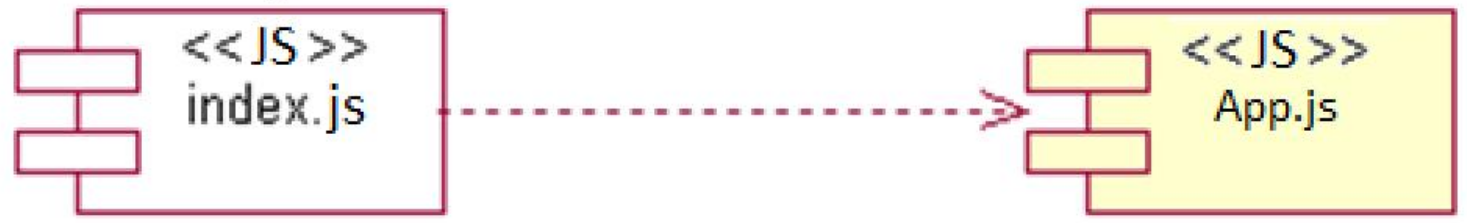


FindProfessor aplikacija je bazirana na Model-View-Controller (MVC) projektnom obrascu. MVC je softverski pristup koji razdvaja aplikacionu logiku od prezentacione.

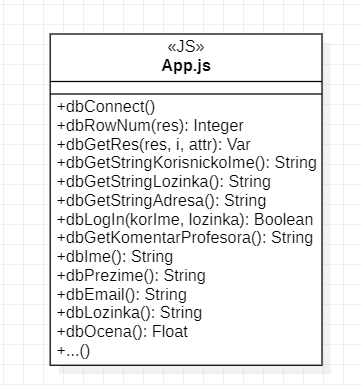
* **Model** predstavlja strukturu podataka. Mode klase sadrži funkcije pomoću kojih preuzimamo, dodajemo i ažuriramo podatke u bazi podataka.
* **View** je skup informacija koje se predstavljaju korisniku. View može biti stranica ali isto tako može biti deo kao što je header ili footer.
* **Controller** je međusloj između modela, view-a i bilo kog drugog resursa potrebnog za obradu HTTP zahteva i generisanja web stranice.

### Komponente za pristup podacima

Komponente koje realizuju domen problema se uključuju isključivo preko **App.js** komponente korisničkog interfejsa. Na taj način zadržavaju sva podešavanja stila definisana u ovom skriptu. Na sledećem dijagramu su prikazane komponente ovog sloja i njihove međusobne zavisnosti:



Na sledećem UML dijagramu klasa pobrojane su funkcije za pristup podacima iz baze:



Navedene funkcije obavljaju sledeće zadatke:

* dbConnect – povezivanje na bazu podataka, poziva se u samom skriptu
* dbRowNum – vraća broj slogova sadržanih u rezultatu upita
* dbGet – vraća vrednost polja iz rezultata upita
* dbLogin – loguje korisnika i vraća true ili false
* dbGetStringKorisnickoIme– vraća korisnicko ime korisnika
* dbPrezime – vraća prezime korisnika
* dbOcena– izracunava i vraca prosecnu ocenu
* ... – ostale funkcije

# Performanse

Izabrana arhitektura softvera podržava zahteve u pogledu broja korisnika koji mogu simultano pristupati sistemu i vremena odziva za pristup bazi podataka specificirane u zahtevima u pogledu performansi [5]:

1. Sistem će da podrži do 1000 simultanih pristupa korisnika portalu.
2. Vreme potrebno za pristupanje bazi podataka u cilju izvršenje nekog upita ne sme da bude veće od 5 sekundi.

Zahtevane performanse su zadovoljene izborom tehnologija na kojima će sistem biti razvijen i definisane hardverske platforme [5].

# Kvalitet

Izabrana arhitektura softvera podržava zahteve u pogledu dostupnosti i srednjeg vremena između otkaza specificirane u zahtevima u pogledu pouzdanosti [5]:

1. FindProfessor aplikacija će biti dostupna 24 časa dnevno, 7 dana u nedelji. Vreme kada aplikacija nije dostupna ne sme da pređe 10%.
2. Srednje vreme između dva sukcesivna otkaza ne sme da padne ispod 120 sati.