TaF

The healthy Chef

Arhitekturni projekat

Verzija 1.1

Pregled izmena

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | **Verzija** | **Opis** | **Autor** |
| 23.04.2022 | 1.0 | Inicijalna verzija | Dušan |
| 12.05.2022 | 1.1 | Ažuriranje sadržaja. | Dušan |
|  |  |  |  |
|  |  |  |  |

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 Čitanje recepta 8

6.2.2 Čitanje blogova 8

6.2.3 Kreiranje naloga 8

6.2.4 Prijavljivanje (Log in) 8

6.2.5 Pregled preporučenih blogova za čitanje 8

6.2.6 Pregled preporučenih recepta za čitanje 8

6.2.7 Sortiranje recepata po datumu objavljivanja 8

6.2.8 Soriranje recepata po vremenu spremanja obroka 8

6.2.9 Sortiranje recepata po popularnosti 8

6.2.10 Sortiranje blogova po datumu objavljivanja 8

6.2.11 Sortiranje blogova po popularnosti 8

6.2.12 Dodela komentara receptu 8

6.2.13 Dodela komentara blogu 9

6.2.14 Dodela ocene receptu 9

6.2.15 Dodela ocene blogu 9

6.2.16 Ažuriranje podataka o sebi 9

6.2.17 Uvid profila nekog autora i njegovog sadržaja 9

6.2.18 Čekiranje izvršenih koraka nekog recepta 9

6.2.19 Čuvanje omiljenih recepata 9

6.2.20 Čuvanje omiljenih blogova 9

6.2.21 Pregled sačuvanih recepata 9

6.2.22 Pregled sačuvanih blogova 9

6.2.23 Kreiranje bloga 9

6.2.24 Kreiranje recepata 9

6.2.25 Izmena sadržaja recepata 9

6.2.26 Izmena sadržaja u blogu 10

6.2.27 Brisanje recepata 10

6.2.28 Brisanje bloga. 10

7. Pogled na logičku arhitekturu sistema 10

7.1 Pregled arhitekture – organizacija paketa i podsistema u slojeve 10

7.1.1 Korisnički interfejs 11

7.1.2 Aplikaciona logika 11

7.1.3 Pristup podacima 11

7.1.4 HTML 11

7.1.5 CSS 11

7.1.6 React 11

7.1.7 .NET 11

7.1.8 Neo4j 11

8. Pogled na procese 11

8.1 Procesi 11

8.1.1 Web čitač 12

8.1.2 Web server 12

8.1.3 .NET Framework 12

8.1.4 Neo4j Server 12

9. Pogled na raspoređivanje sistema 12

9.1 Klijent 13

9.2 Web server i Neo4j server 13

10. Pogled na implementaciju sistema 13

10.1 Model domena 13

10.2 Šema baze podataka 14

10.3 Komponente sistema 15

10.3.1 Komponente korisničkog interfejsa 15

10.3.2 Komponente aplikacione logike 16

10.3.3 Komponente za pristup podacima 16

11. Performanse 19

12. Kvalitet 19

Arhitekturni projekat

# Cilj dokumenta

Cilj ovog dokumenta je detaljni opis arhitekture TaF Web aplikacije.

# Opseg dokumenta

Dokument se odnosi na TaF aplikaciju koji će biti razvijen od strane Misthios-a. TaF predstavlja skraćenicu za The healthy Chef. . Namena aplikacije je pretraga recepata za spremanje zdrave hrane, čitanje blogova o istoj kao i mogućnost publikovanja samih recepta i blogova od strane ljudi koji poseduju adekvatna znanja o zdravoj ishrani.

# Reference

Spisak korišćene literature:

1. TaF – Predlog projekta, TaF-01, V1.0, 2022, Misthios.
2. TaF – Planirani raspored aktivnosti na projektu, V1.0, 2022, Misthios.
3. TaF – Plan realizacije projekta, V1.0, 2022, Misthios.
4. TaF – Vizija sistema, V1.0, 2022, Misthios.
5. TaF – Specifikacija zahteva, V1.1, 2022, Misthios.

# Predstavljanje arhitekture

Arhitektura aplikacije 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. TaF aplikacija će biti implementiran kao Web aplikacija zasnovana na React biblioteci i NoSQL(Neo4j) bazi podataka [4].
2. Klijentski deo TaF aplikacijaa će biti optimizovan za sledeće Web čitače: Internet Explorer 6.0 i noviji, Opera 8.0 i noviji, kao i Firefox (Mozilla) [4].
3. Svi zahtevi u pogledu performansi dati u [5] moraju biti uzeti u obzir pri izboru arhitekture i razvoju sistema.

# Pogled na slučajeve korišćenja

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

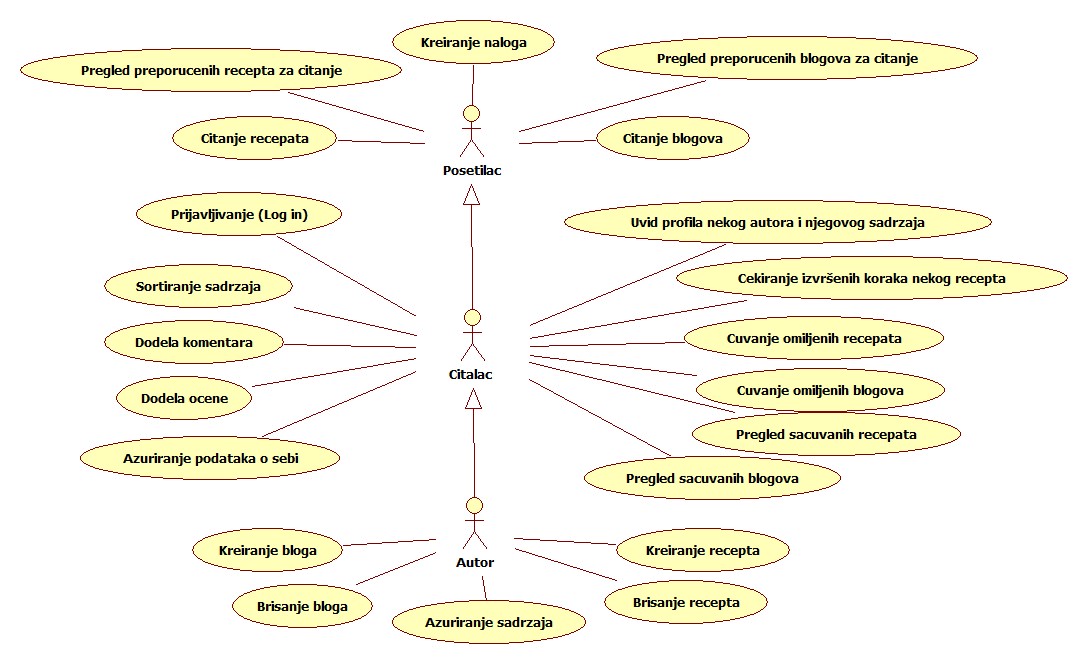
Slučajevi korišćenja TaF aplikacije su:

* Čitanje recepata
* Čitanje blogova
* Kreiranje naloga
* Prijavljivanje (Log in)
* Sortiranje sadržaja
  + - Soritanje recepata po datumu objavljivanja
    - Soriranje recepata po vremenu spremanja obroka
    - Sortiranje recepata po popularnosti
    - Soritranje blogova po datumu objavljivanja
    - Sortiranje blogova po popularnosti
* Dodela komentara
  + Dodela komentara receptu
  + Dodela komentara blogu
* Dodela ocene
  + Dodela ocene receptu
  + Dodela ocene blogu
* Ažuriranje podataka o sebi
* Uvid profila nekog autora i njegovog sadržaja
* Čekiranje izvršenih koraka nekog recepta
* Čuvanje omiljenih recepata
* Čuvanje omiljenih blogova
* Pregled sačuvanih recepata
* Pregled sačuvanih blogova
* Kreiranje recepata
* Kreiranje bloga
* Ažuriranje sadržaja
  + Izmena sadržaja recepata
  + Izmena sadržaja bloga
* Brisanje recepata
* Brisanje bloga

Ove slučajevi korišćenja mogu da iniciraju posetioci aplikacije, čitaoci i autori.

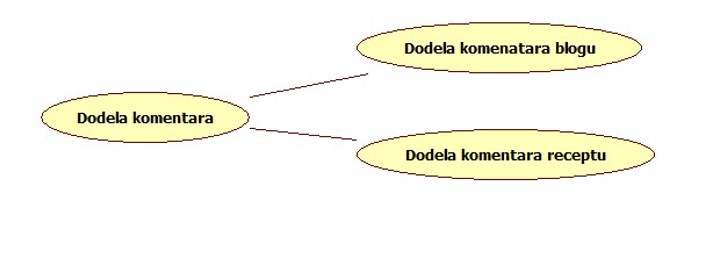
## Dijagrami slučajeva korišćenja

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

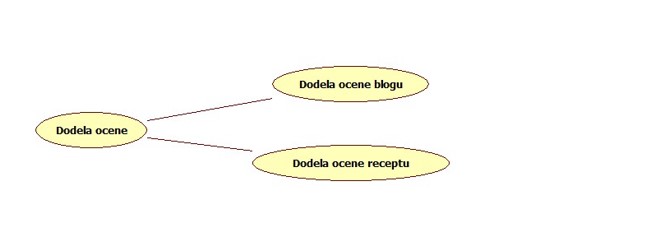


Slučajevi korišćenja *dodela komentara, dodela ocene, ažuriranje sadržaja, sortiranje sadržaja* obuhvataju složenije radnje koje se mogu razložiti dalje razložiti na pojedinačne slučajeve korišćenja.

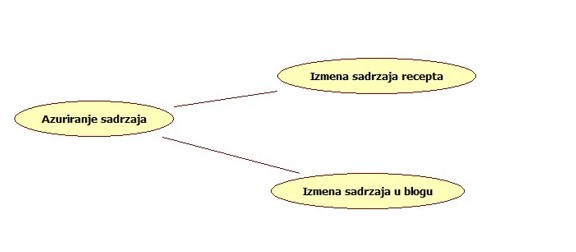
Detaljni UML dijagram za slučaj korišćenja *dodela komentara* je prikazan na sledećoj slici:



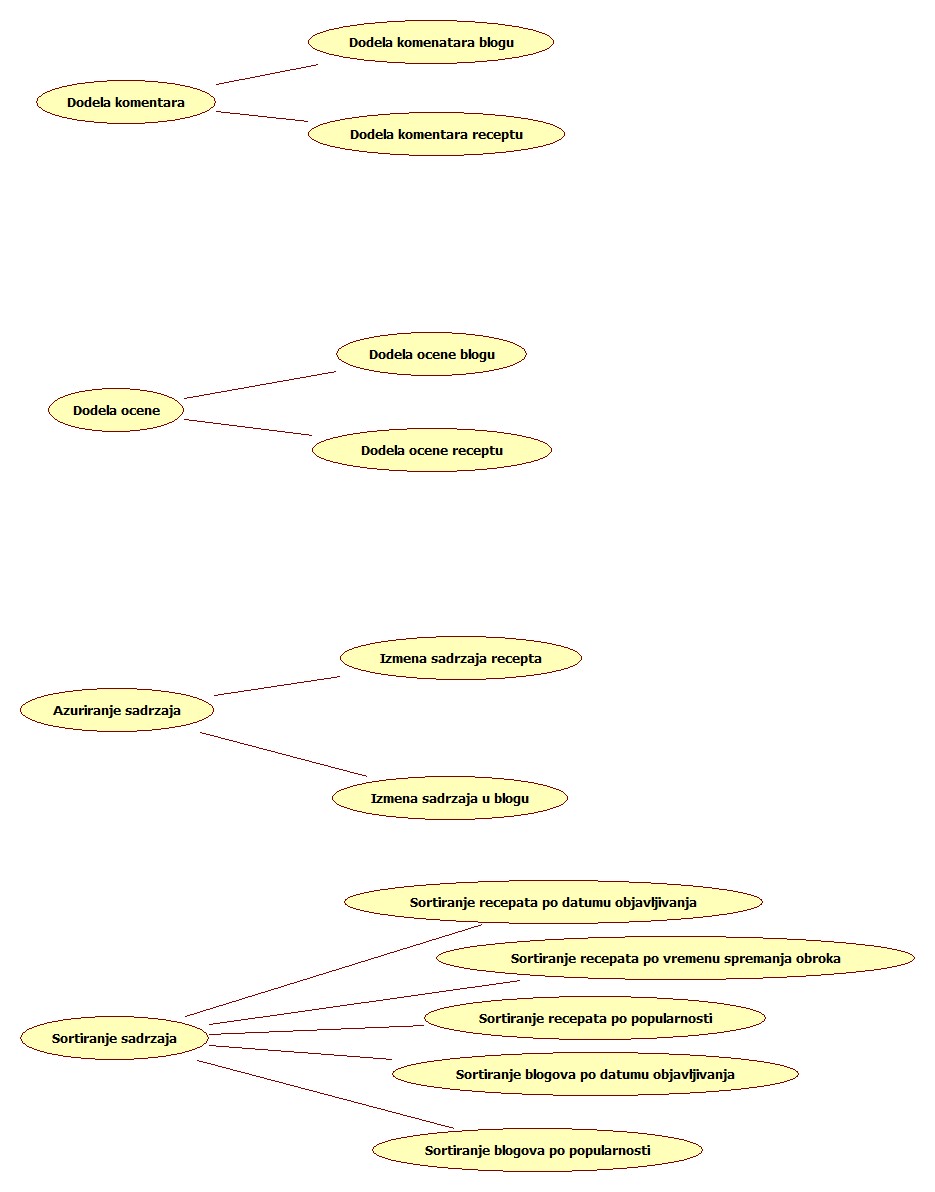
Detaljni UML dijagram za slučaj korišćenja *dodela ocene* je prikazan na sledećoj slici:



Detaljni UML dijagram za slučaj korišćenja *ažuriranje sadržaja* je prikazan na sledećoj slici:



Detaljni UML dijagram za slučaj korišćenja *sortiranje sadržaja* je prikazan na sledećoj slici:



## Kratak opis slučajeva korišćenja

### Čitanje recepta

Kratak opis: Prikaz recepta za spremanje zdrave hrane

Akteri koji iniciraju slučaj korišćenja: Posetilac aplikacije, čitaoci (studenti, srednjoškolci), autori.

### Čitanje blogova

Kratak opis: Prikaz bloga koji se tiče bitnosti zdrave hrane.

Akteri koji iniciraju slučaj korišćenja: Posetilac aplikacije, čitaoci (studenti, srednjoškolci), autori.

### Kreiranje naloga

Kratak opis: Prikaz stranice aplikacije sa formom za kreiranje naloga.

Akteri koji iniciraju slučaj korišćenja: Posetilac aplikacije.

### Prijavljivanje (Log in)

Kratak opis: Prikaz stranice sa formom za prijavljivanje na aplikaciju kako bi se pristupilo dodatnim fukcionalnostima aplikacije.

Akteri koji iniciraju slučaj korišćenja: Čitaoci (studenti, srednjoškolci) i autori.

### Pregled preporučenih blogova za čitanje

Kratak opis: Prikaz glavne stranice na kojoj se nalazi preporučeni blogovi za čitanje.

Akteri koji iniciraju slučaj korišćenja: Posetilac aplikacije, Čitaoci (studenti, srednjoškolci) i autori.

### Pregled preporučenih recepta za čitanje

Kratak opis: Prikaz glavne stranice na kojoj se nalazi preporučeni recepti za čitanje.

Akteri koji iniciraju slučaj korišćenja: Posetilac aplikacije, Čitaoci (studenti, srednjoškolci) i autori.

### Sortiranje recepata po datumu objavljivanja

Kratak opis: Stranica na kojoj se nalaze recepti se sortiraju po datumu objavljivanja, bira se opcija: najnoviji ili najstariji..

Akteri koji iniciraju slučaj korišćenja: Čitaoci (studenti, srednjoškolci) i autori.

### Soriranje recepata po vremenu spremanja obroka

Kratak opis: Stranica na kojoj se nalaze recepti se sortiraju po vremenu potrebnog da se spremi obrok, tako da se prvo prikazuju recepti za koje je potrebno najamanje vremena za spreamnje obroka.

Akteri koji iniciraju slučaj korišćenja: Čitaoci (studenti, srednjoškolci) i autori.

### Sortiranje recepata po popularnosti

Kratak opis: Stranica na kojoj se nalaze recepti se sortiraju popularnosti (prikazuju se prvo najpopularniji recepti, tj. recepti sa najviše pozitivnih ocena).

Akteri koji iniciraju slučaj korišćenja: Čitaoci (studenti, srednjoškolci) i autori.

### Sortiranje blogova po datumu objavljivanja

Kratak opis: Stranica na kojoj se nalaze blogovi se sortiraju po datumu objavljivanja, bira se opcija: najnoviji ili najstariji.

Akteri koji iniciraju slučaj korišćenja: Čitaoci (studenti, srednjoškolci) i autori.

### Sortiranje blogova po popularnosti

Kratak opis: Stranica na kojoj se nalaze blogovi se sortiraju popularnosti (prikazuju se prvo najpopularniji blogovi, tj. blogovi sa najviše pozitivnih ocena).

Akteri koji iniciraju slučaj korišćenja: Čitaoci (studenti, srednjoškolci) i autori.

### Dodela komentara receptu

Kratak opis: Korisnik dodeljuje komentar receptu.

Akteri koji iniciraju slučaj korišćenja: Čitaoci (studenti, srednjoškolci) i autori.

### Dodela komentara blogu

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

Akteri koji iniciraju slučaj korišćenja: Čitaoci (studenti, srednjoškolci) i autori.

### Dodela ocene receptu

Kratak opis: Korisnik dodeljuje ocenu receptu.

Akteri koji iniciraju slučaj korišćenja: Čitaoci (studenti, srednjoškolci) i autori.

### Dodela ocene blogu

Kratak opis: Korisnik dodeljuje ocenu blogu.

Akteri koji iniciraju slučaj korišćenja: Čitaoci (studenti, srednjoškolci) i autori.

### Ažuriranje podataka o sebi

Kratak opis: Korisnik ažurira svoje lične podatke u opisu profila.

Akteri koji iniciraju slučaj korišćenja: Čitaoci (studenti, srednjoškolci) i autori.

### Uvid profila nekog autora i njegovog sadržaja

Kratak opis: Pregled opštih informacija nekog autora i njegovih publikacija (recepata i blogova).

Akteri koji iniciraju slučaj korišćenja: Čitaoci (studenti, srednjoškolci) i autori.

### Čekiranje izvršenih koraka nekog recepta

Kratak opis: Praćenje izvršenih koraka prilikom spremanje hrane po nekom receptu.

Akteri koji iniciraju slučaj korišćenja: Čitaoci (studenti, srednjoškolci) i autori.

### Čuvanje omiljenih recepata

Kratak opis: Čuvanje omiljenih recepata koji su se svideli korisniku.

Akteri koji iniciraju slučaj korišćenja: Čitaoci (studenti, srednjoškolci) i autori.

### Čuvanje omiljenih blogova

Kratak opis: Čuvanje omiljenih blogova koji su se svideli korisniku.

Akteri koji iniciraju slučaj korišćenja: Čitaoci (studenti, srednjoškolci) i autori.

### Pregled sačuvanih recepata

Kratak opis: Korisnik vrši pregled omiljenih recepata koje je sačuvao i bira recept za čitanje.

Akteri koji iniciraju slučaj korišćenja: Čitaoci (studenti, srednjoškolci) i autori.

### Pregled sačuvanih blogova

Kratak opis: Korisnik vrši pregled omiljenih blogova koje je sačuvao i bira blog za ponovo čitanje.

Akteri koji iniciraju slučaj korišćenja: Čitaoci (studenti, srednjoškolci) i autori.

### Kreiranje bloga

Kratak opis: Pisanje i publikovanje bloga od strane autora.

Akteri koji iniciraju slučaj korišćenja: Autor

### Kreiranje recepata

Kratak opis: Pisanje i publikovanje bloga od strane autora.

Akteri koji iniciraju slučaj korišćenja: Autor.

### Izmena sadržaja recepata

Kratak opis: Autor ažurira sadžaj svog recepta

Akteri koji iniciraju slučaj korišćenja: Autor.

### Izmena sadržaja u blogu

Kratak opis: Autor ažurira sadžaj svog bloga.

Akteri koji iniciraju slučaj korišćenja: Autor.

### Brisanje recepata

Kratak opis: Autor briše svoj recept.

Akteri koji iniciraju slučaj korišćenja: Autor.

### Brisanje bloga.

Kratak opis: Autor briše svoj blog.

Akteri koji iniciraju slučaj korišćenja: Autor.

# 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.

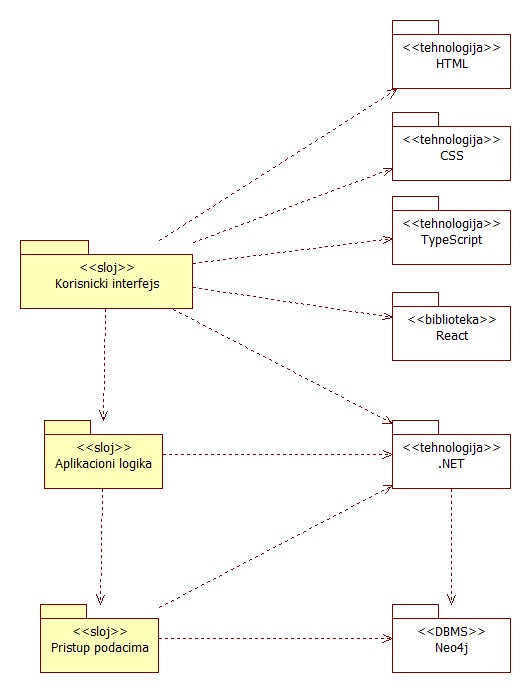
Logički pogled na TaF aplikaciju obuhvata 3 glavna paketa: Korisnički interfejs, Aplikaciona logika, Pristup podacima.

Paket *Korisnički interfejs* sadrži Web stranice, React komponente i multimedijalni sadržaj koji realizuju grafički dizajn.

Paket *Aplikaciona logika* predstavlja srednji sloj sistema koji sadrži .NET kod zadužen za realizaciju funkcionalnosti specifičnih za domen sistema koji se razvija.

Paket *Pristup podacima* sadrži .NET kod koji 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 aplikacije. U njemu su sadržane sve HTML, CSS, multimedijalni sadržaji i React komponente koje generišu HTML stranice preko kojih korisnici komuniciraju sa sistemom.

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

### Aplikaciona logika

sloj

Sloj aplikacione logike je srednji sloj u troslojnoj arhitekturi TaF aplikacije. Sadrži .NET kod koji realizuji funkcionalnost karakterističan za domen primene aplikacije i uspostavljaju vezu između korisničkog interfejsa i sloja za pristup podacima.

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

### Pristup podacima

sloj

Sloj za pristup podacima se nalazi na dnu troslojne arhitekture i sadrži .NET kod zadužen za pribavljanje, dodavanje i ažuriranje podataka koji se čuvaju u Neo4j bazi podataka.

Ovaj sloj ne zavisi od drugih slojeva, ali je zavisan od paketa .NET i Neo4j baze podataka.

### HTML

tehnologija

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

### CSS

tehnologija

Tehnologija koja je zadužena za estetski i vizuelni prikaz HTML stranice.

### React

biblioteka

Biblioteka React omogućava olakšano kreiranje stranica, kao i mogućnost višestrukog korišćenja komponti.

### .NET

tehnologija

.NET 5 se koristi za pristup i manipulaciju podataka iz baze podataka.

### Neo4j

DBMS

Neo4j je graf baza podataka zadužena za upravljanje i čuvanje podataka u bazi podataka.

# Pogled na procese

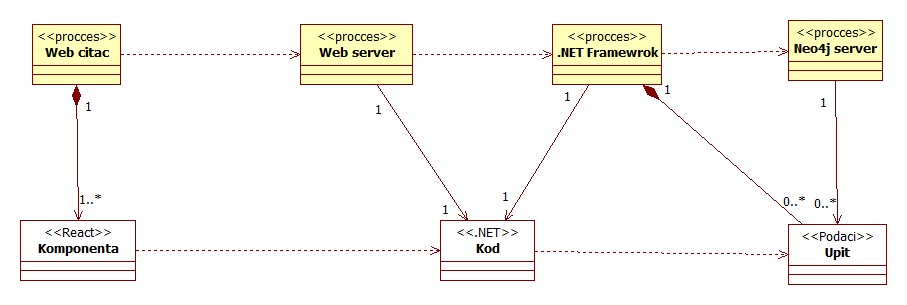
U ovom odeljku je sadržan pogled na procesnu arhitekturu sistema. Ovaj opis treba da sadrži specifikaciju različitih zadataka (procesa i niti) uključenih u rad sistema. Takođe je potrebno dati dijagrame koji pokazuju njihovu interakciju i konfiguraciju. Dodela objekata i klasa na određene zadatke takođe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

Web aplikacije zasnovane na PHP-u imaju relativno jednostavan procesni model koji je u potpunosti pod kontrolom Web servera. Sa stanovišta projektanta PHP 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 TaF aplikacijaa kao Web aplikacije.

## Procesi

Na sledećem UML dijagramu klasa prikazani su procesi koji učestvuju u izvršenju TaF aplikacije. Dijagram je opšteg tipa i može se primeniti na bilo koju Web aplikaciju zasnovanu na .NET-u, TypeScript-React i Neo4j 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 su zahtevani podaci iz baze podataka, Web server inicira izvršenje .NET koda koji obrađuje odgovarajući zahtev i generiše sadržaj koji se vraća čitaču.

### .NET Framework

.NET Framework proces obavlja posao obrade zadatog .NET koda i generiše odgovarajući sadržaj koji Web server šalje Web čitaču. Za izvršenje, ovaj proces može da zahteva usluge Neo4j servera. Komunikacija između .NET procesa i Neo4j servera se obavlja preko prosleđivanja upita i vraćanja rezultata.

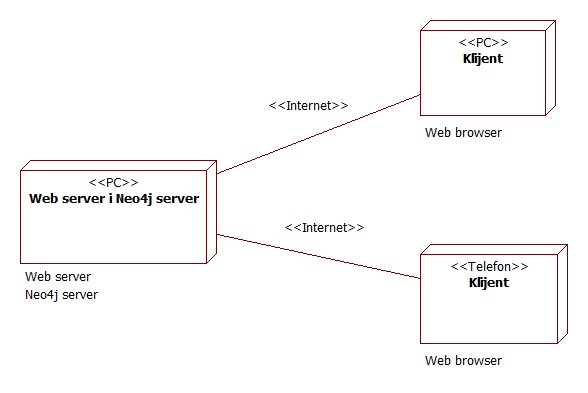
### Neo4j Server

Neo4j Server je proces koji izvršava funkcionalnost Neo4j sistema za upravljanje bazama podataka. Ovaj proces može konkurentno da prihvati određen broj upita, izvrši ih nad graf 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 TaF aplikaciju.



## Klijent

Pristup TaF aplikaciji se obavlja preko Web pretraživača sa PC ili telefonskih uređaja. Za povezivanje između klijenta i Web servera koristi se Internet infrastruktura tako da nema ograničenja u pogledu lokacije klijenta.

## Web server i Neo4j server

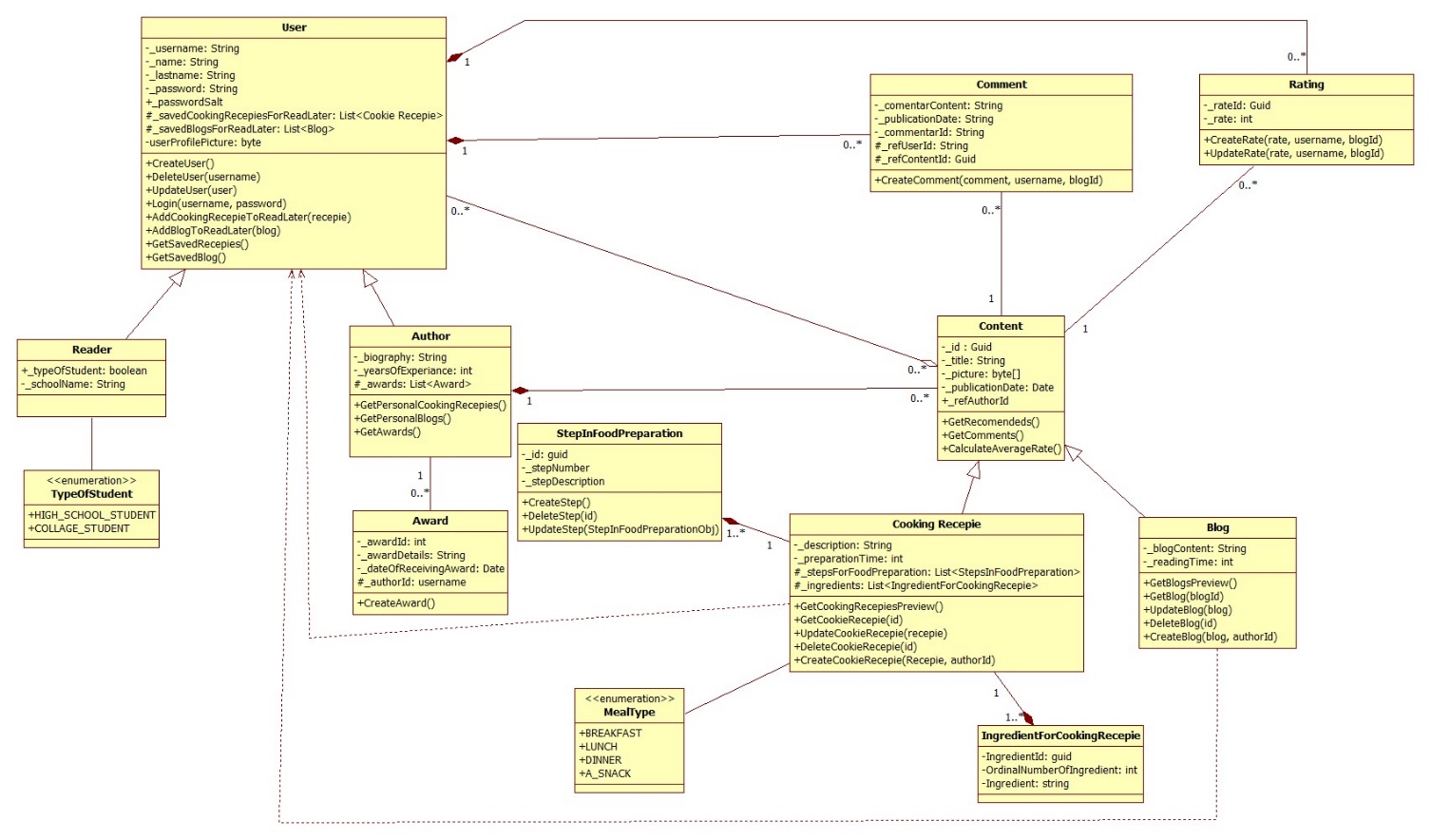
Računar na kome se izvršava Web server opslužuje više klijenata koji pristupaju preko Interneta, takođe na istom računaru se izvršava i Neo4j server zadužen za upravljanje bazom podataka.

# Pogled na implementaciju sistema

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

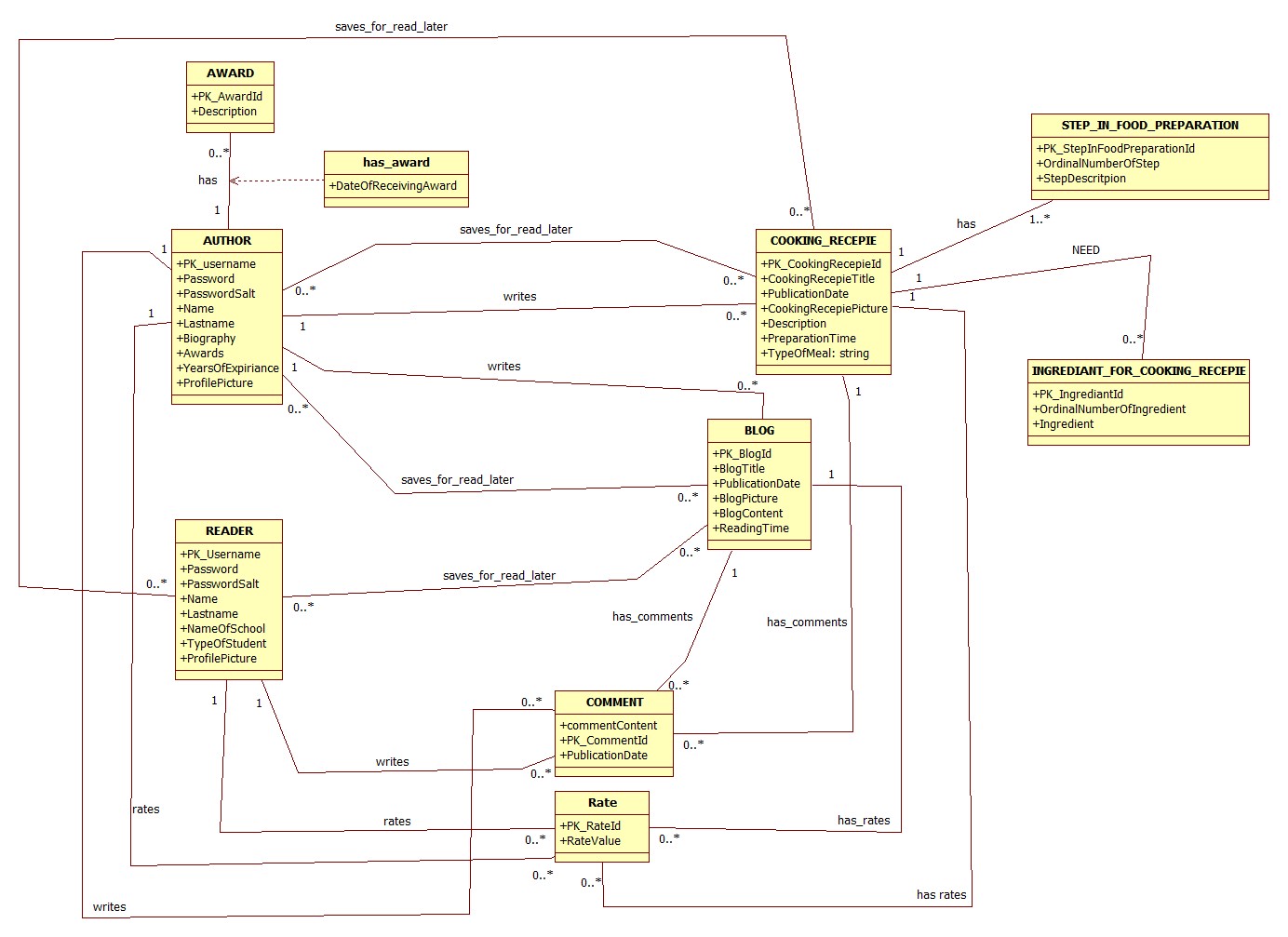
## Model domena

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



## Šema baze podataka

Detaljna šema baze podataka je prikazana na sledećem dijagramu. Baza podataka i dijagram su kreirani korišćenjem StarUML-a, dok je migracija obavljena pomoću code-first pristupa preko .NET-a.

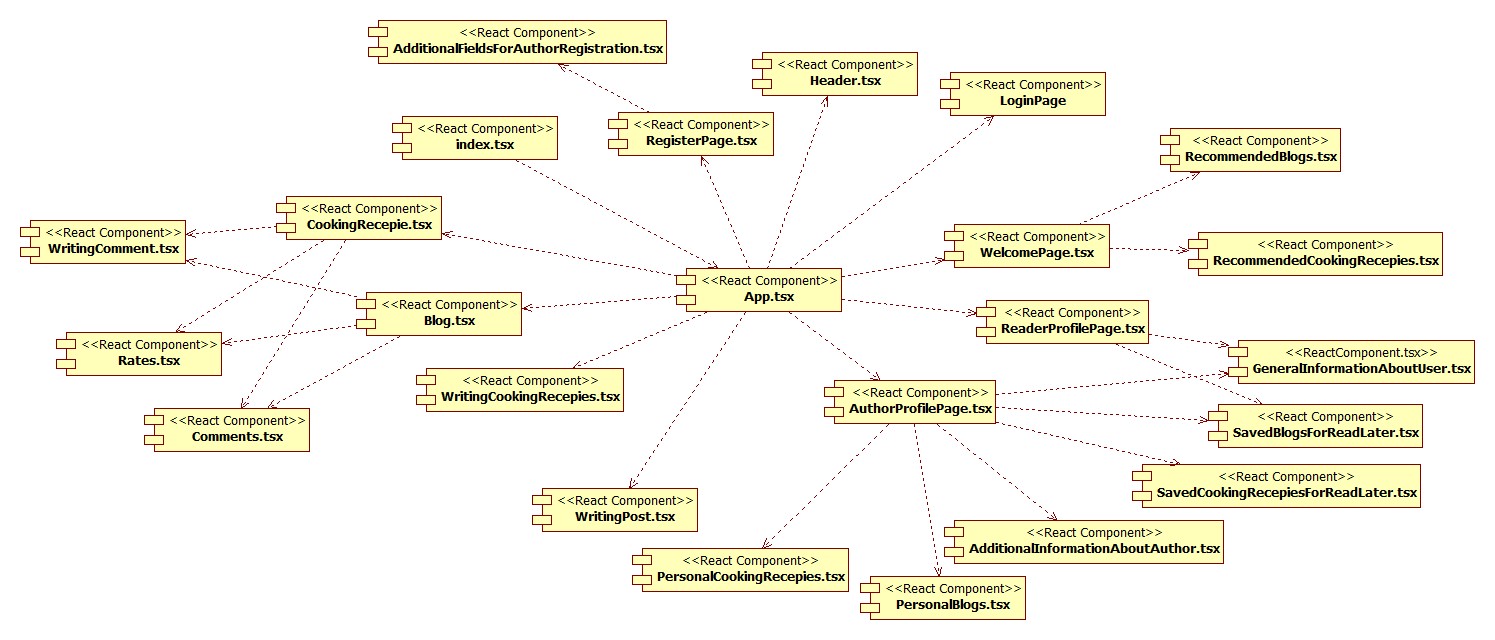


## Komponente sistema

Komponente sistema TaF aplikacije su React komponente i C# fajlovi čiji će pregled biti dat po arhitekturnim slojevima. Za ilustraciju će biti korišćeni UML dijagrami komponenti, ali i dijagrami klasa.

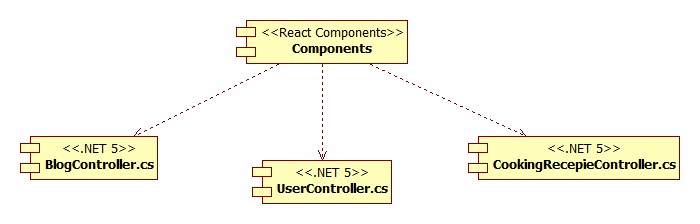
### Komponente korisničkog interfejsa

Dizajn korisničkog interfejsa biće implementiran preko TypeScript React komponeti:



### Komponente aplikacione logike

Komponente koje realizuju aplikacionu logiku su .NET i prateće klase. Komponete korisničkog interfejsa pristupaju kontrolerima (*controllers*) kako bi pozivali aplikacionu logiku. Na sledećem dijagramu su prikazane komponente ovog sloja i njihove međusobne zavisnosti:



**UserController.cs** – obezbeđuje kreiranje, prijavljivanje korisnika, ažuriranje podataka, pretragu korisnika, preuzimanje sačuvanih blogova, recepata i slično.

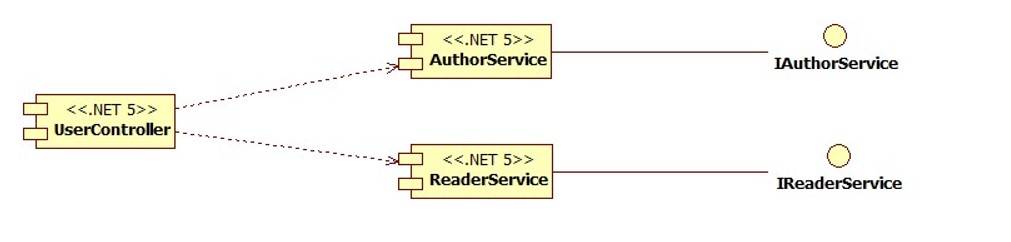
**BlogController.cs** – obezbeđuje kreiranje, ažuriranje bloga, pretragu konkretnog bloga, preuzimanje liste blogova kao i preuzimanje preporučenih blogova, dodela ocene, komentara blogu i slično.

**CookingRecepieController.cs** – obezbeđuje kreiranje, ažuriranje recepta, pretragu konkretnog recepta, preuzimanje liste recepata kao i preuzimanje preporučenih recepata, dodela ocene, komentara receptu i slično.

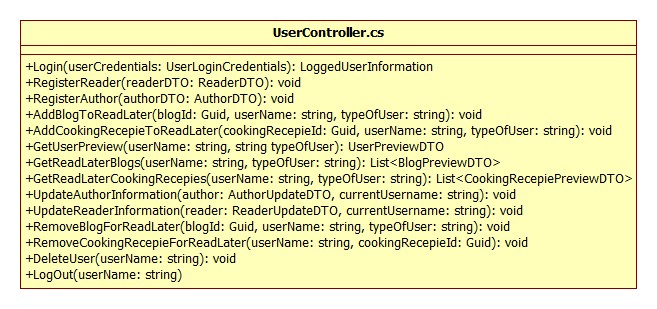
### Komponente za pristup podacima

Pristup bazi podataka je u potpunosti zatvoren u funkcije koje su definisane u okviru .NET kontrolera gde isti pristupaju .NET interfejsima gde se obavlja pristup bazi podataka. U nastavku se nalazi prikaz UML dijagrama, gde se ilustruje pristup kontrolera i odgovarajućih interfejsa i njihovih implementacija.

**UserController**

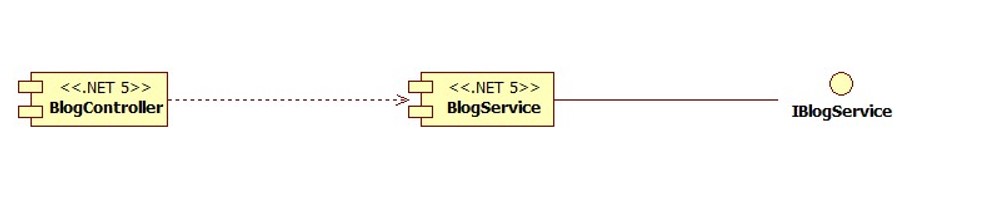


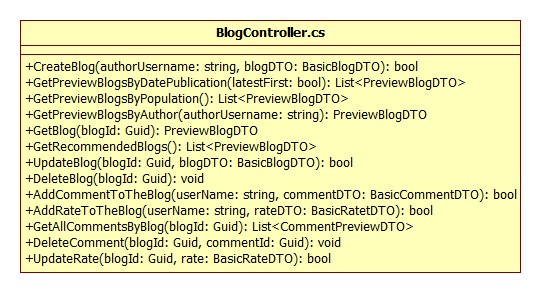
Na sledećem UML dijagramu klasa pobrojane su metode User kontrolera koji preko Author i Reader servisa pristupa bazi podataka.



Opis navedenih metoda:

* **Login –** prijavljivanje korisnika na aplikaciju, pribavljanje sesije.
* **RegisterReader –** kreiranje naloga studenta/srednjoškolca.
* **RegisteraAuthor -** kreiranje naloga autoru recepata i blogova.
* **AddBlogToReadLater –** korisnik dodaje blogove koje želi kasnije da pročita ili sačuva.
* **AddCookingRecepieToReadLater -** korisnik dodaje recepte koje želi kasnije da pročita ili sačuva.
* **GetUserPreview –** pribavljanje informacija o korisniku.
* **GetReadLaterBlogs –** pribavljanje blogova koje je korisnik sačuvao.
* **GetReadLaterCookingRecepies -** pribavljanje recepta koje je korisnik sačuvao.
* **UpdateAuthorInformation** i **UpdateReaderInformation –** ažuriranje podataka određenog korisnika.
* **RemoveBlogForReadLater –** izbacivanje bloga iz liste koji je korisnik sačuva.
* **RemoveCookingRecepieForReadLater -** izbacivanje recepta iz liste koji je korisnik sačuva.
* **DeleteUser –** brisanje naloga korisnika.
* **LogOut –** odjavljivanje korisnika sa aplikacije, brisanje sesije.

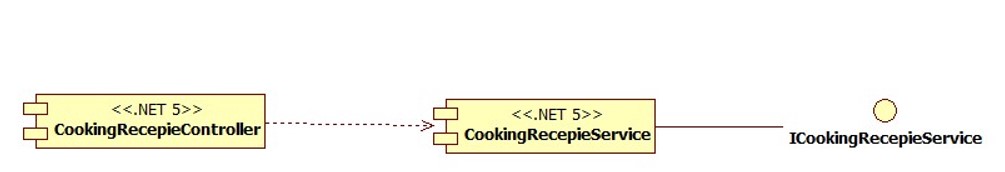
**BlogConBlogController**

****Na sledećem UML dijagramu klasa pobrojane su metoda Blog kontrolera koji preko Blog servisa pristupa bazi podataka.

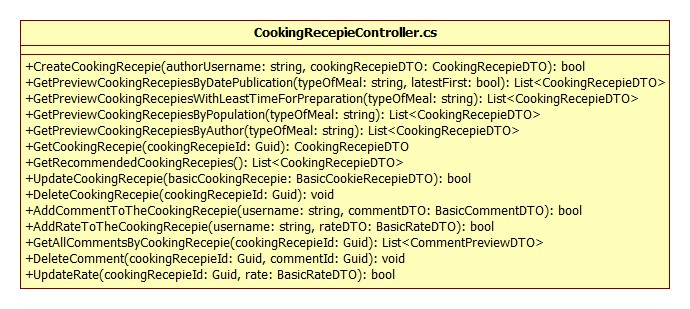
Opis navedenih metoda:

* **CreateBlog** – kreiranje bloga.
* **GetPreviewBlogsByDatePublication** – pribavljanje blogova. U zavisnosti od parametra *„latestFirst“* pribavljaju se najnoviji blogovi ili njastariji.
* **GetPreviewBlogsByPopulation**- pribavljanje blogova, ali tako da se na počektu nalaze najpopilarniji blogovi (sa njavisim ocenama).
* **GetPreviewBlogsByAuthor**- pribavljanje autorovih blogova.
* **GetRecommendedBlogs** – pribavljanje preporučenih blogova za čitanje.
* **GetBlog** – pribavljanje konkretnog bloga.
* **UpdateBlog** – ažuriranje sadržaja bloga.
* **DeleteBlog** – birsanje bloga.
* **AddCommentToTheBlog** – dodavanje komentara blogu.
* **AddRateToTheBlog** – dodela ocene blogu,
* **GetAllCommentsByBlog** – pribavljanje svih komentara koji pripadaju blogu,
* **DeleteComment** – brisanje komentara.
* **UpdateRate** – ažuriranje ocene bloga.

**CookingRecepieController**

****

Na sledećem UML dijagramu klasa pobrojane su metode CookingRecepie kontrolera koji preko CookingRecepie servisa pristupa bazi podataka.

****

Opis navedenih metoda:

* **CreateCookingRecepie** – kreiranje recepta.
* **GetPreviewCookingRecepiesByDatePublication** – pribavljanje recepta. U zavisnosti od parametra *„latestFirst“* pribavljaju se najnoviji recepti ili njastariji.
* **GetPreviewCookingRecepiesWithLeastTimeForPreparation-** pribavljanje recepta za koje je potrebno njamanje vremena za spremanja hrane.
* **GetPreviewCookingRecepiesByPopulation**- pribavljanje recepta, ali tako da se na počektu nalaze najpopilarniji recepti (sa njavisim ocenama).
* **GetPreviewCookingRecepiesByAuthor**- pribavljanje autorovih recepta.
* **GetCookingRecepie** – pribavljanje konkretnog recepta.
* **GetRecommendedBlogs** – pribavljanje preporučenih recepta za čitanje.
* **UpdateCookingRecepie** – ažuriranje sadržaja recepta.
* **DeleteCookingRecepie** – birsanje recepta.
* **AddCommentToTheCookingRecepie**– dodavanje komentara receptu.
* **AddRateToTheCookingRecepie** – dodela ocene receptu.
* **GetAllCommentsByCookingRecepie** – pribavljanje svih komentara koji pripadaju receptu.
* **DeleteComment** – brisanje komentara.
* **UpdateRate** – ažuriranje ocene recepta.

# Performanse

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

1. Sistem će da podrži do 1000 simultanih pristupa korisnika aplikacijau.
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. TaF aplikacija će biti dostupan 24 časa dnevno, 7 dana u nedelji. Vreme kada aplikacija nije dostupan ne sme da pređe 10%.
2. Srednje vreme između dva sukcesivna otkaza ne sme da padne ispod 120 sati.