ReMIND

Vaš lični asistent.

Arhitekturni projekat

Verzija 1.0

Pregled izmena

| **Datum** | **Verzija** | **Opis** | **Autor** |
| --- | --- | --- | --- |
| 06.05.2022. | 1.0 | Inicijalna verzija | Nikola |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Sadržaj

[**Cilj dokumenta**](#_heading=h.gjdgxs) **5**

[**Opseg dokumenta**](#_heading=h.30j0zll) **5**

[**Reference**](#_heading=h.1fob9te) **5**

[**Predstavljanje arhitekture**](#_heading=h.2et92p0) **5**

[**Ciljevi i ograničenja arhitekture**](#_heading=h.tyjcwt) **5**

[**Pogled na slučajeve korišćenja**](#_heading=h.3dy6vkm) **5**

[Dijagrami slučajeva korišćenja](#_heading=h.2bn6wsx) 6

[Kratak opis slučajeva korišćenja](#_heading=h.qsh70q) 8

[Prijavljivanje na sistem](#_heading=h.3as4poj) 8

[Kreiranje obaveza](#_heading=h.1hmsyys) 9

[Kreiranje naloga](#_heading=h.41mghml) 9

[Kreiranje timova](#_heading=h.2grqrue) 9

[**Pogled na logičku arhitekturu sistema**](#_heading=h.28h4qwu) **9**

[Client](#_heading=h.37m2jsg) 10

[Server](#_heading=h.1mrcu09) 10

[C#](#_heading=h.2lwamvv) 10

[.NET WPF](#_heading=h.111kx3o) 10

[SQLite](#_heading=h.3l18frh) 10

[**Pogled na procese**](#_heading=h.206ipza) **11**

[Procesi](#_heading=h.4k668n3) 11

[Desktop aplikacija](#_heading=h.2zbgiuw) 11

[Web server](#_heading=h.1egqt2p) 11

[API server](#_heading=h.3ygebqi) 11

[SQLite Server](#_heading=h.2dlolyb) 11

[**Pogled na raspoređivanje sistema**](#_heading=h.sqyw64) **11**

[Client](#_heading=h.3cqmetx) 12

[Web server](#_heading=h.1rvwp1q) 12

[SQLite server](#_heading=h.4bvk7pj) 12

[**Pogled na implementaciju sistema**](#_heading=h.2r0uhxc) **12**

[Model domena](#_heading=h.1664s55) 12

[Šema baze podataka](#_heading=h.3q5sasy) 13

[Komponente sistema](#_heading=h.25b2l0r) 13

[Komponente korisničkog interfejsa](#_heading=h.kgcv8k) 13

[Komponente aplikacione logike](#_heading=h.34g0dwd) 14

[Komponente za pristup podacima](#_heading=h.1jlao46) 14

[**Performanse**](#_heading=h.43ky6rz) **15**

[**Kvalitet**](#_heading=h.2iq8gzs) **15**

Arhitekturni projekat

# Cilj dokumenta

Cilj ovog dokumenta je detaljni opis arhitekture ReMIND sistema..

# Opseg dokumenta

Dokument se odnosi na ReMIND sistem koji će biti razvijen od strane tima “Samo Ventilatori”. ReMIND predstavlja englesku reč sa značenjem “podsetiti”. Ovo aludira na osnovnu namenu i funkciju istog, što je efikasna organizacija i podsetnik tekućih obaveza pojedinca, grupe ili firme.

# Reference

Spisak korišćene literature:

1. D01\_Predlog\_Projekta.docx
2. D02\_Vizija\_Sistema.docx
3. D03\_Raspored\_Aktivnosti.pod
4. D03\_Plan\_Realizacije.docx
5. D04\_Spec\_Zahteva.docx

# 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. ReMIND sistem će biti implementiran kao Desktop aplikacija zasnovana na C# jeziku i koristiće SQLite bazu podataka na serverskom delu, sa .NET 5 Web API interfejsom [4].
2. Klijentski deo ReMIND sistema će biti optimizovan za Windows 7 operativni sistem i novije. Preporučen operativni sistem je Windows 10 ili noviji. [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].

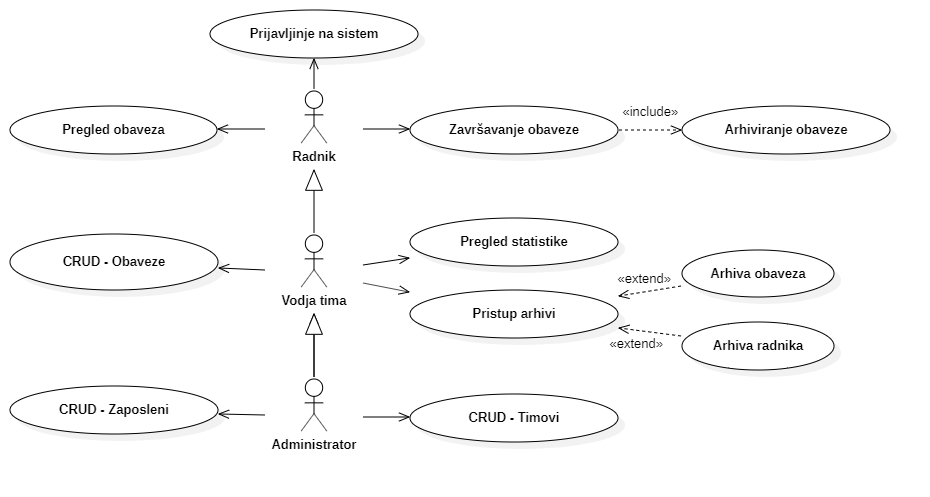
Slučajevi korišćenja ReMIND sistema su:

* *Prijavljivanje na sistem*
* *Pregled obaveza*
  + *Pregled samo svojih obaveza*
  + *Pregled obaveza jednog tima*
  + *Pregled obaveza svih timova*
  + *Vizuelni prikaz*
    - *Kalendar*
    - *Lista*
* *Završavanje obaveza*
  + *Arhiviranje obaveza*
* *CRUD - Obaveze*
  + *Kreiranje obaveza*
    - *Povezivanje obaveza sa osobom*
    - *Povezivanje obaveza sa timom*
  + *Brisanje obaveza*
  + *Azuriranje obaveza*
* *Pregled statistike*
* *Pristup Arhivi*
  + *Arhiva obaveza*
  + *Arhiva radnika*
* *CRUD - Zaposleni*
  + *Kreiranje naloga*
  + *Brisanje naloga* 
    - *Arhiviranje naloga*
  + *Azuriranje Naloga*
* *CRUD - Timovi*
  + *Kreiranje Tima*
    - *Dodavanje vodje tima*
    - *Dodavanje radnika*
  + *Brisanje Tima*
  + *Azuriranje Tima*

Ove slučajevi korišćenja mogu da iniciraju radnik firme, vođa tima i administrator sistema.

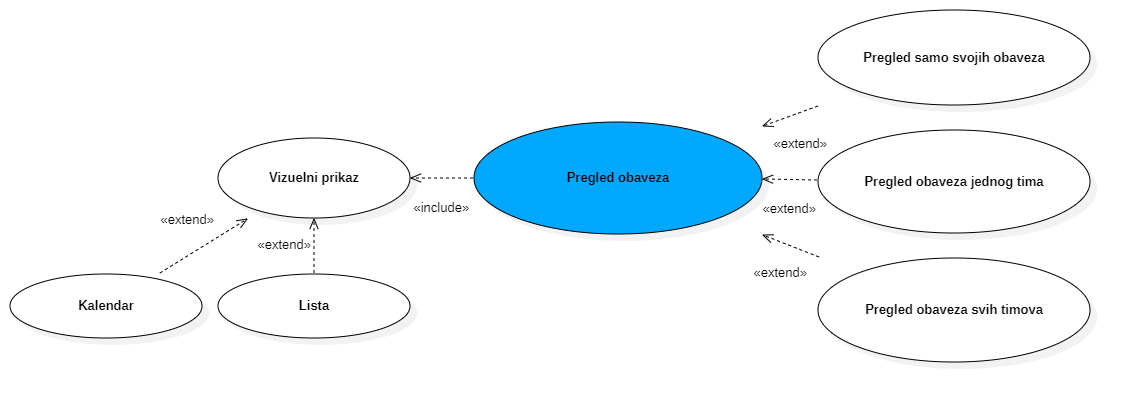
## Dijagrami slučajeva korišćenja

Osnovni UML dijagram koji prikazuje korisnike i slučajeve korišćenja ReMIND sistema prikazan je na sledećoj slici:

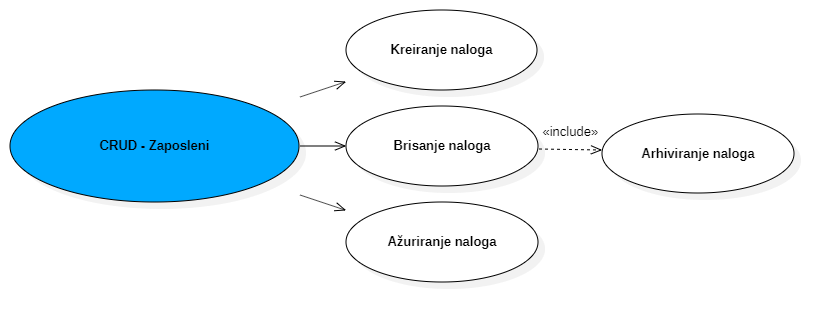


Slučajevi korišćenja *CRUD - Zaposleni, Obaveze, Timovi* i *Pregled obaveza* obuhvataju složenije radnje koje se mogu razložiti dalje na pojedinačne slučajeve korišćenja.

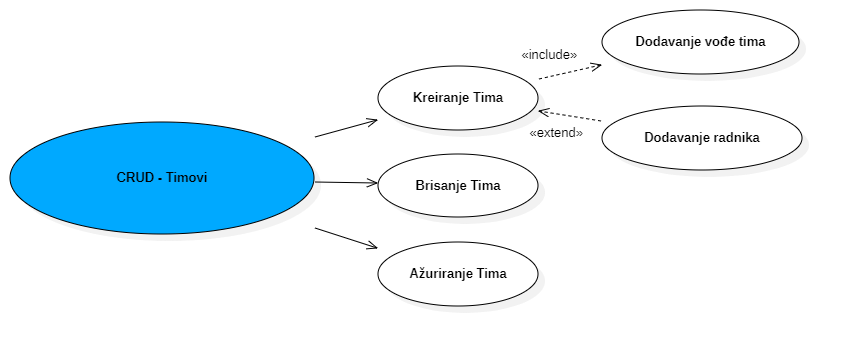
Detaljni UML dijagram za slučaj korišćenja *Pregled obaveza* je prikazan na sledećoj slici:



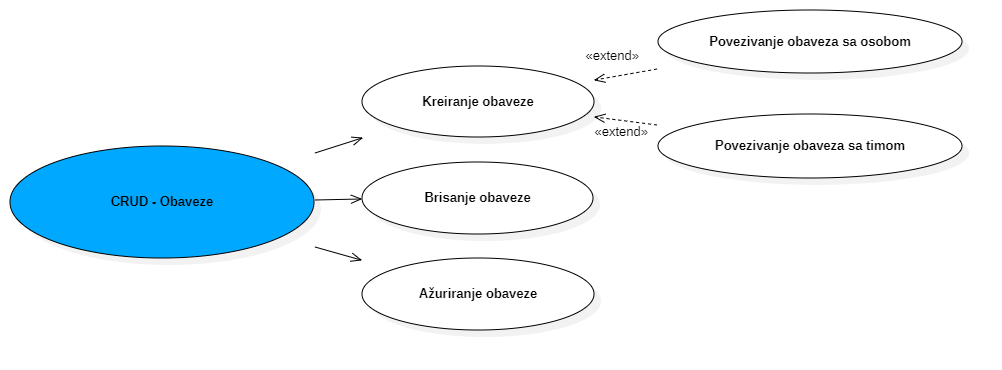
Detaljni UML dijagram za slučaj korišćenja *CRUD - Zaposleni* je prikazan na sledećoj slici:



Detaljni UML dijagram za slučaj korišćenja *CRUD - Timovi* je prikazan na sledećoj slici:



Detaljni UML dijagram za slučaj korišćenja *CRUD - Obaveze* je prikazan na sledećoj slici:



## Kratak opis slučajeva korišćenja

### Prijavljivanje na sistem

Kratak opis: Popunjavanjem forme sa individualnim podacima za pristup sistemu.

Akteri koji iniciraju slučaj korišćenja: Radnik, Vođa tima, Administrator.

* + 1. *Pregled obaveza*

Kratak opis: Prikaz svih obaveza individualne osobe ili celog tima u zavisnosti od prava pristupa.

Akteri koji iniciraju slučaj korišćenja: Radnik, Vođa tima, Administrator.

* + 1. *Završavanje obaveza*

Kratak opis: Označavanje neke obaveze završenom kako bi odgvorni za nju mogli da provere da li je obaveza završena i spremna za arhiviranje.

Akteri koji iniciraju slučaj korišćenja: Radnik[opcionalno], Vođa tima.

* + 1. *CRUD - Obaveze*

Kratak opis: Sve bitne opcije za rad sa obavezama.

Akteri koji iniciraju slučaj korišćenja: Vođa tima, Administrator.

*Pristup Arhivi*

Kratak opis: Pogled na sve uradjene poslove sa njihovim informacijama kada su urađeni i od koga.

Akteri koji iniciraju slučaj korišćenja: Vođa tima, Administrator.

* + 1. *Pregled statistike*

Kratak opis: Prikaz statistike odrađenih poslova u izabranom formatu nad grupom radnika ili tima u određenom vremenskom periodu.

Akteri koji iniciraju slučaj korišćenja: Vođa tima, Administrator.

* + 1. *CRUD - Zaposleni*

Kratak opis: Sve bitne opcije za upravljanje sa zaposlenima.

Akteri koji iniciraju slučaj korišćenja: Administrator.

* + 1. *CRUD - Timovi*

Kratak opis: Sve bitne opcije za upravljanje sa timovima.

Akteri koji iniciraju slučaj korišćenja: Administrator.

* + 1. *Arhiviranje obaveza*

Kratak opis: Prikaz stranice sa informacijama o određenom projektu.

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

### Kreiranje obaveza

Kratak opis: Kreiranje nove obaveze i njeno povezivanje sa osobom ili timom koji istu treba završiti.

Akteri koji iniciraju slučaj korišćenja: Vođa tima, Administrator

### Kreiranje naloga

Kratak opis: Kreiranje novog naloga za novo zaposleno lice kako bi imali pristup sistemu.

Akteri koji iniciraju slučaj korišćenja: Administrator.

### Kreiranje timova

Kratak opis: Kreiranje tima, dodavanje svih bitnih informacija, dodavanje radnika i vođe tima.

Akteri koji iniciraju slučaj korišćenja: Administrator.

# Pogled na logičku arhitekturu sistema

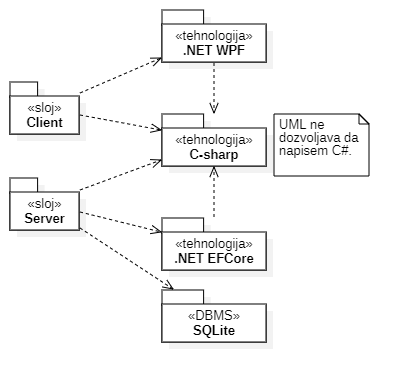
U ovom odeljku je dat pregled logičke arhitekture sistema. Ovaj pogled sadrži opis najznačajnijih klasa, njihove organizacije u pakete i podsisteme i organizacija podsistema u slojeve. U cilju opisivanja dinamičkih aspekata arhitekture, ovaj odeljak može da uključi opise realizacije najznačajnijih slučajeva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

Logički pogled na ReMIND sistem obuhvata 3 glavna paketa: Korisnički interfejs, Aplikaciona logika, Serverska logika, Pristup podacima.

Sloj *Client* sadrži Windows prozore i multimedijalni sadržaj koji realizuju grafički dizajn i forme preko kojih korisnici sistema interaguju sa sistemom. Unutar njega će biti implementirana logika koja sadrži C# kod zadužen za realizaciju funkcionalnosti specifičnih za domen sloja.

Sloj *Server* predstavlja logički sloj serverske strane sistema i pristup podacima unutar istog. Sastoji se iz C# koda zaduženog za realizaciju funkcionalnosti specifičnih za domen sloja i LINQ-SQL funkcija koje predstavljaju interfejs za pristup, dodavanje i ažuriranje podataka koji se čuvaju u bazi podataka.

Svaki sloj će biti adekvatno organizovan specijalizovanim klasama raspoređenim u folderima.

Pregled arhitekture – organizacija paketa i podsistema u slojeve

### Client

sloj

Ovaj sloj realizuje korisnički interfejs sistema. U njemu su sadržani svi Windows prozori, multimedijalni sadržaji i C# kod koji služi generisanju ovih prozora, njihovoj interakciji i komunikaciji sa serverskim delom sistema.

Sloj *Client* zavisi paketa C# i .NET WPF.

### Server

sloj

Sloj serverskog dela sistema je serverski deo ReMIND sistema. Sadrži bazu podataka, CRUD operacije prema istoj, logiku sa specifičnim funkcionalnostima potrebnim za svoj domen i WebAPI interfejs za komunikaciju sa klijentskim delovima sistema.

Ovaj sloj zavisi od sledećih paketa: C#, .NET EFCore(Entity Framework Core) i SQLite.

### C#

tehnologija

Tehnologija C#-a definiše programski jezik kreiran od strane kompanije Microsoft. Podržava objektno orijentisano programiranje na različitim operativnim sistemima. Logički delovi sistema će biti realizovani u C#-u. Ovo je osnova za sve naprednije tehnologije korišćene u projektu.

### .NET WPF

tehnologija

.NET WPF, akronim za *Windows Platform Foundation*,obezbeđuje mehanizme za kreiranje prozora, njihovih animacija i upravljanje njihovim okidačima. Ovime se realizuje korisnički interfejs klijentskog dela sistema.

### SQLite

DBMS

SQLite predstavlja sistem za upravljanje bazama podataka koji će se koristiti za realizaciju ReMIND sistema.

# Pogled na procese

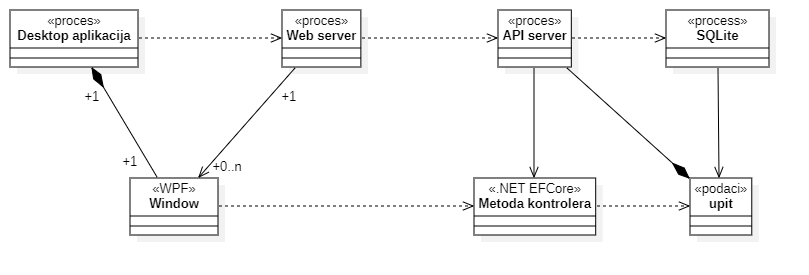
U ovom odeljku je sadržan pogled na procesnu arhitekturu sistema. Ovaj opis treba da sadrži specifikaciju različitih zadataka (procesa i niti) uključenih u rad sistema. Takođe je potrebno dati dijagrame koji pokazuju njihovu interakciju i konfiguraciju. Dodela objekata i klasa na određene zadatke takođe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

Desktop aplikacije zasnovane na .NET WPF imaju relativno jednostavan procesni model koji je podržan od strane Web servera.

Ilustracije radi u nastavku je dat opis procesa uključenih u izvršenje ReMIND sistema kao desktop aplikacija.

## Procesi

Na sledećem UML dijagramu klasa prikazani su procesi koji učestvuju u izvršenju ReMIND sistema.



### Desktop aplikacija

Desktop aplikacija je proces koji izvršava funkcionalnost aplikazije za prikaz WPF prozora popunjenim sa informacijama dobijenih od strane Web servera. Desktop aplikacija moze da prikazuje samo jedan prozor u jednom trenutku.

Podaci desktop aplikacije zavise od Web servera koji generise i vraca odgovarajuce JSON stringove na zahtev.

### Web server

Web server je proces koji izvršava funkcionalnost opsluživanja zahteva prispelih sa više Desktop aplikacija. Ukoliko je zahtevana API funkcija, Web server inicira prosleđuje zahtev API serveru koji obrađuje odgovarajću metedu kontrolera i generiše sadržaj koji se vraća aplikaciji.

### API server

API server obrađuje zahtev od strane Web servera i generiše odgovarajući tekstualni sadržaj koji potom šalje Desktop aplikaciji. On za određene zahteve može da zahteva usluge SQLite server-a.

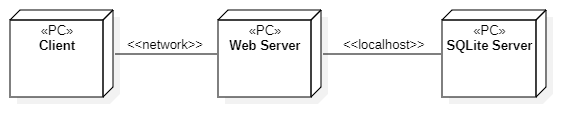
### SQLite Server

*SQLite* Server je proces koji izvršava funkcionalnost *SQLite* 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 ReMIND sistema.



## Client

Pristup ReMIND sistemu se obavlja preko klijentskih računara na kojima se izvršava Dekstop aplikacija. Za povezivanje između klijenta i Web servera koristi se infrastruktura mreže.

## Web server

Računar na kome se izvršava Web server opslužuje više klijenata. Pored osnovnog procesa koji realizuje funkcionalnost Web servera, na ovom računaru mogu da se izvršavaju i automatizovani poslovi i skripte. U najopštioj konfiguraciji DBMS se izvršava na istoj mašini kao i Web Server.

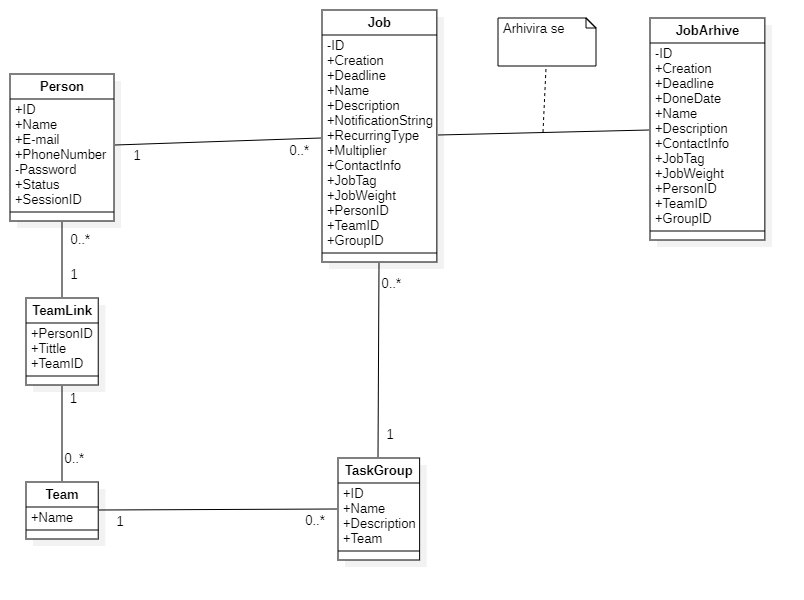
## SQLite server

SQLite server je računar na kome se izvršava SQLite proces koji realizuje funkcionalnost sistema za upravljanje bazama podataka.

# Pogled na implementaciju sistema

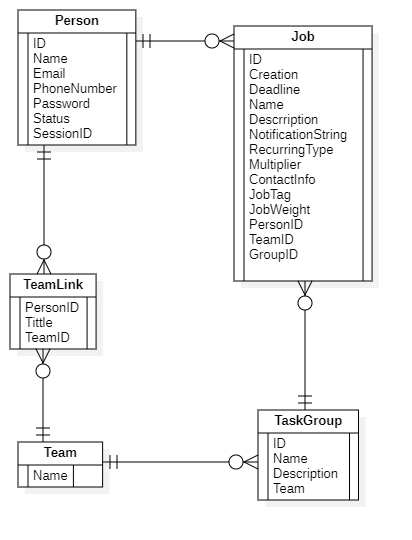
Pogled na implementaciju prikazuje različite aspekte bitne za implementaciju sistema. U slučaju ReMIND sistema ovaj odeljak sadrži model domena, šemu baze podataka i prikaz komponenti sistema razvrstanih u ranije identifikovane pakete.

## Model domena

Model domena za koji se ReMIND sistem 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 koje će biti implementirane.

## Šema baze podataka

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

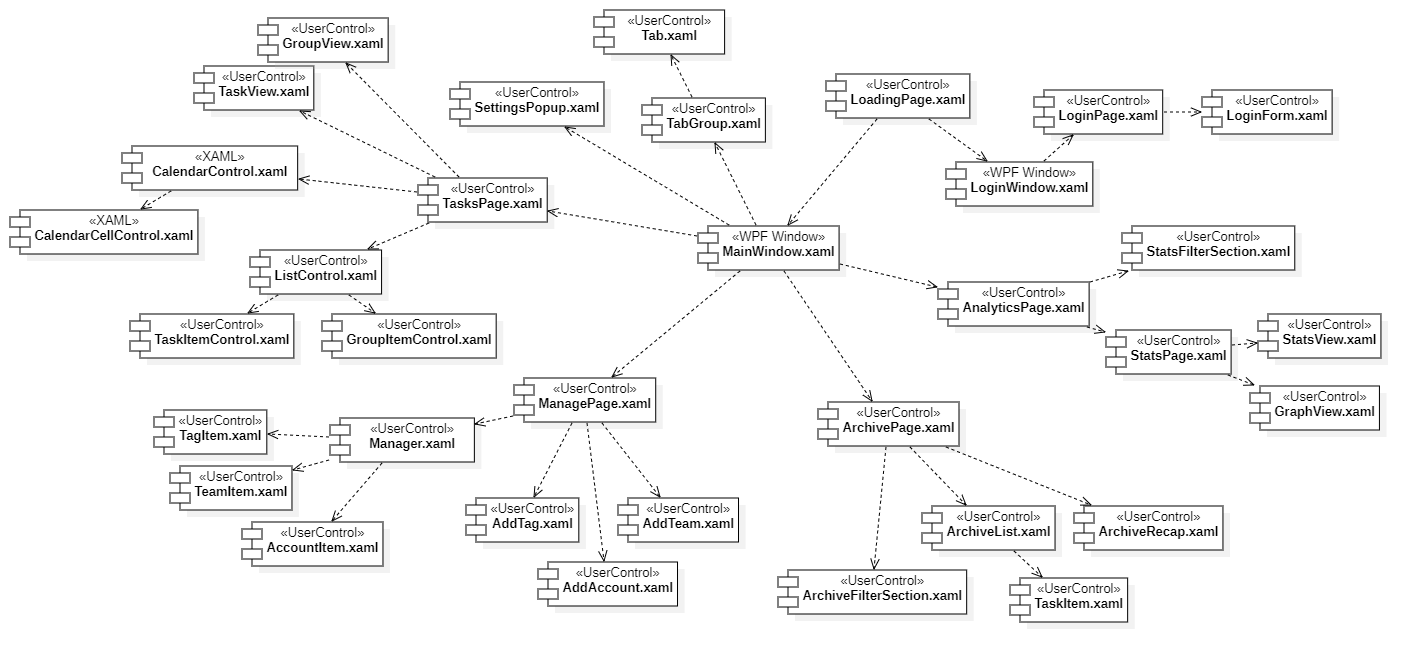


## Komponente sistema

Komponente ReMIND sistema su XAML(Extensible Application Markup Language) fajlovi, svaki od kojih je podržan svojim istoimenim pratećim .cs(C#) klasama. Za ilustraciju će biti korišćeni UML dijagrami komponenti.

### Komponente korisničkog interfejsa

Dizajn korisničkog interfejsa je obuhvaćen grupom komponenti, predstavljene sledećim dijagramom:



* Komponenta **LoginWindow.xaml** predstavlja početni prozor koji dozvoljava korisniku da se prijavi na sistem. Sadrži se iz komponenti **LoginPage.xaml** i **LoginForm.xaml** koje predstavljaju korisnički interfejs i aplikacionu logiku za domen problema.
* Komponenta **MainWindow.xaml** predstavlja prozor koji sadrži sve ostale komponente sistema potrebne za rad sa istim nakon prijavljivanja.
* Komponenta **LoadingPage.xaml** predstavlja prozor učitavanja. Biće prikazan u prikladnim trenucima, kako bi korisnik znao da treba sačekati par trenutaka.
* Komponenta **SettingsPopup.xaml** predstavlja mali prozor namenjen pregledu informacija naloga podešavanjima vezanih za isti.
* Komponenta **TabGroup.xaml** zajedno sa **Tab.xaml** predstavlja grupu kartica na vrhu aplikacije koje pružaju korisniku mogućnost da menja trenutnu stranicu aplikacije dok daje do znanja korisniku na kojoj se stranici trenutno nalazi.
* Komponenta **TasksPage.xaml** predstavlja glavnu i prvu stranicu aplikacije. Njen korisnički interfejs i aplikaciona logika će biti podeljena u dva dela: **CalendarControl.xaml** zadužen za kalendarski prikaz obaveza i **ListControl.xaml** zadužen za prikaz obaveza pomoću liste.
* Komponenta **AnalyticsPage.xaml** predstavlja pregledan prikaz istorije obaveza izabranih timova/radnika. Sastoji se iz **StatsFilterSection.xaml** koji služi za biranje podataka za prikaz, **StatsPage.xaml** koji predstavlja grafički prikaz analitike uz njegova podešavanja, realizovan pomoću **GraphView.xaml** i **StatsView.xaml**.
* Komponenta **ArchivePage.xaml** predstavlja arhivu obaveza. Realizovana je pomoću selekcije podataka za prikaz **ArchiveFilterSection.xaml**, liste **ArchiveList.xaml** prikazanih poslova **TaskItem.xaml** i kratkog rezimea prikazanih podataka **ArchiveRecap.xaml**.
* Komponenta **ManagePage.xaml** predstavlja pristup nalozima i timovima firme.

### Komponente aplikacione logike

Komponente koje realizuju domen problema se uključuju u svaku komponentu korisničkog interfejsa u vidu C# klase. Na ovaj način smo aplikaciju podelili u manje delove koji su idealni za grupno razvijanje sistema i prikladni za dalje proširivanje istog.

### Komponente za pristup podacima

Pristup bazi podataka je u potpunosti zatvoren u funkcije koje su definisane u okviru .NET Core kontrolera za svaki od entiteta. Pomenuti kontroleri izvršavaju metode koje vraćaju JSON podatke kao odgovor na HTTP zahteve.

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

| **Person**  **Controller** | **Job**  **Controller** | **JobArchive**  **Controller** | **JobTag**  **Controller** |
| --- | --- | --- | --- |
|  |  |  |  |
| +create()  +update()  +delete()  +get()  +getAll()  +getByTeam()  +changePass()  +login()  +logout()  +checkLastActive() | +create()  +update()  +delete()  +getByID()  +getByDatePerson()  +getByDateTeam()  +getByDateGroup()  +archive() | +update()  +delete()  +getByID()  +getByPerson()  +getByTeam()  +getPage() | +create()  +update()  +delete()  +getAll() |

| **TaskGroup**  **Controller** | **Team**  **Controller** | **TeamLink**  **Controller** |
| --- | --- | --- |
|  |  |  |
| +create()  +update()  +delete()  +getByID()  +getByTeam()  +getActive()  +getFinished() | +create()  +update()  +delete()  +getByID()  +getByPerson()  +getByGroup()  +getAll() | +create()  +update()  +delete() |

Navedene funkcije obavljaju sledeće zadatke:

Person Controller:

* create() - kreira novog radnika u bazi podataka
* update() - menja podatke za postojeceg radnika
* delete() - obrise radnika iza baze podataka
* get() - informacija o radniku
* getAll() - vraca listu osoba
* getByTeam() - vraca listu osoba jednog tima
* changePass() - menjanje sifre.
* login() - prijavljivanje korisnika i vracanje sessionID
* logout() - odjavljivanje korisnika i obrisati sessionID
* checkLastActive() - izlogovati korisnika ako nije bio aktivan posle izvesnog vremena.

Job Controller:

* create() - kreira novi posao u bazi podataka
* update() - menja postojece podatke o jednom poslu
* delete() - obrise posao iz baze podataka
* getByID() - vraca poslove po ID-u
* getByDatePerson() - vraca poslove vezane za jednu osobu u odredjenom intervalu
* getByDateTeam() - vraca poslove vezane za jednu grupu u odredjenom intervalu
* getByDateGroup() - vraca poslove vezano za jednu grupu poslova u odredjenom intervalu
* archive() - prebacuje posao u tabelu arhiviranih

JobArhive Controller:

* update()
* delete() - obrise arhivirani posao iz baze podataka
* getByID() - vraca arhiviran posao po ID-u
* getByPerson() - vraca arhivirane poslove za jednu osobu
* getByTeam() - vraca arhivirane poslove za jedan tim
* getPage() - vraca jednu stranu arhiviranih poslova kako ne bi vratili celu bazu podataka

Job Tag Controller:

* create() - kreira novi tag za poslove
* update() - menja postojeci tag za poslove
* delete() - obrise tag za poslove
* getAll() - vraca listu svih tagova

TaskGroup Controller:

* create() - kreira novu grupu poslova
* update() - menja postojecu grupu poslova
* delete() - obrise grupu poslova
* getByID() - vraca grupu poslova po ID-u
* getByTeam() - vraca grupe poslova vezano za jedan team
* getActive() - vraca ne zavrsene poslove iz grupe poslova
* getFinished() - vraca zavrsene poslove iz grupe poslova

Team Controler:

* create() - kreira novi tim u bazi podataka
* update() - menja postojeci tim u bazi podataka
* delete() - obrise tim iz baze podataka
* getByID() - vraca tip po ID-u
* getByPerson() - vraca tim vezan za osobu
* getByGroup() - vraca tim vezan za grupu poslova
* getAll() - vraca listu timova

TeamLink Controler:

* create() - kreira spoj izmedju osobe i tima
* update() - azurira spoj izmedju osobe i tima
* delete() - obrise spoj izmedju osobe i tima

# 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. ReMIND sistem će biti dostupan 24 časa dnevno, 7 dana u nedelji. Vreme kada sistem nije dostupan ne sme da pređe 10%.
2. Srednje vreme između dva sukcesivna otkaza ne sme da padne ispod 120 sati.