



UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA U
NOVOM SADU



Razvoj višeslojnih aplikacija u elektroenergetskim sistemima

Projektna Dokumentacija

Autor: Dejan Jovanić PR128/2016

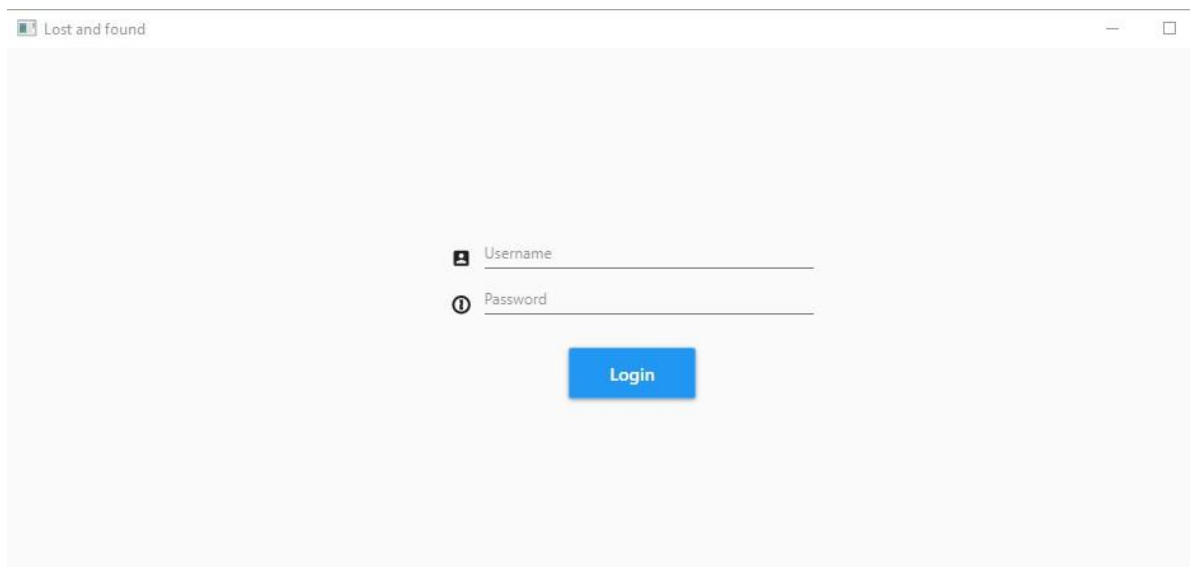
Asistent: Mirko Mikać

Profesor: Darko Čapko

Kratko uputstvo za upotrebu

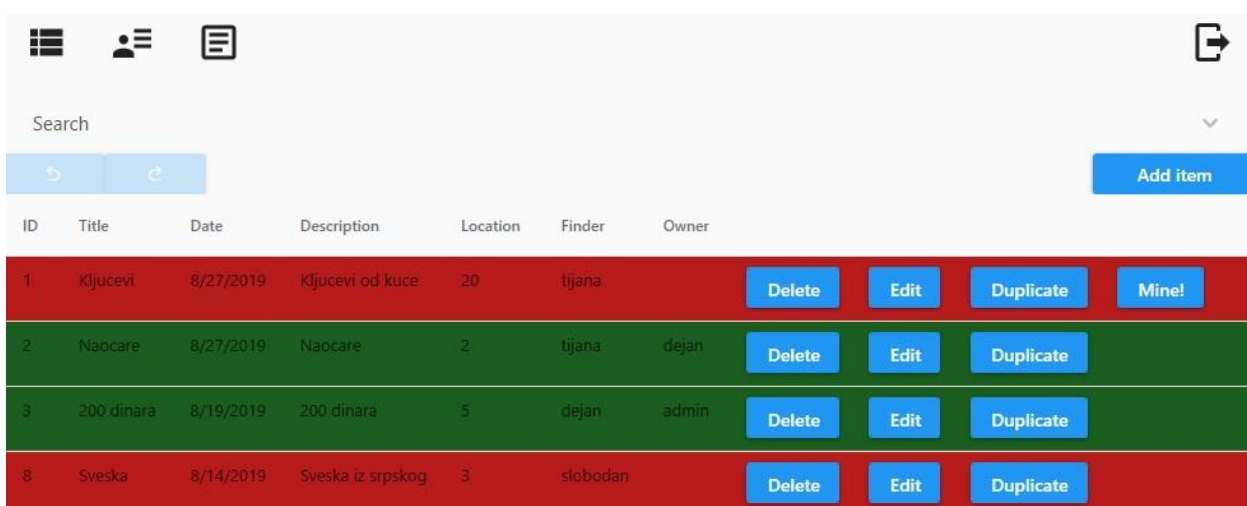
Pre samog uputstva za upotrebu, neophodno je napomenuti da je potrebno prvo pokrenuti serversku aplikaciju, a zatim proizvoljan broj instanci klijentske aplikacije.

Na samom startu aplikacije, korisnika dočekuje "Login" prikaz (slika 1.) u kojem je potrebno uneti korisničko ime i lozinku.



Slika 1. Prikaz za "login"

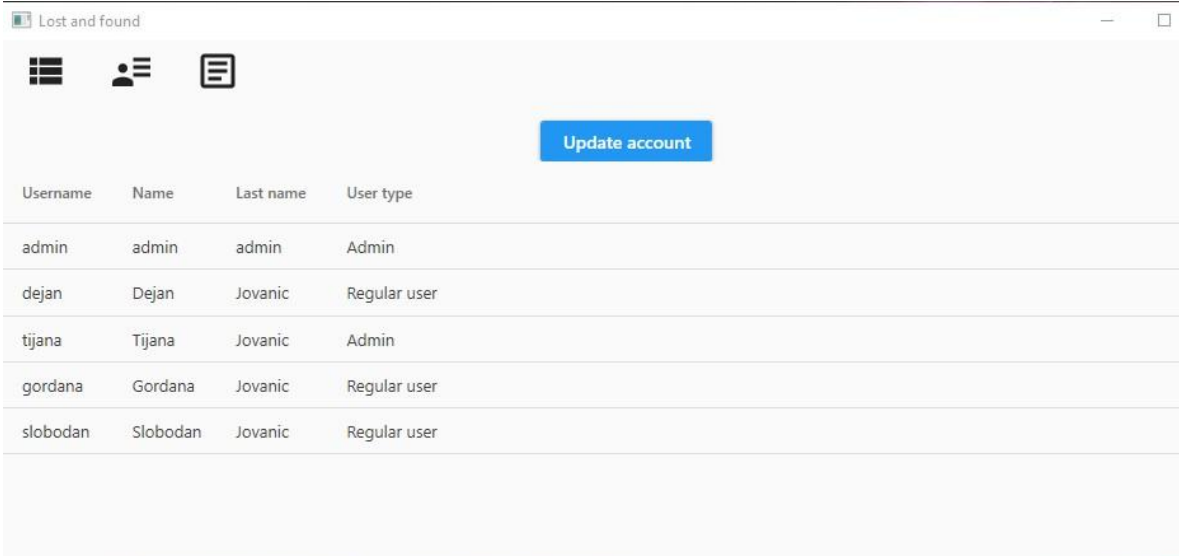
Nakon uspešnog prijavljivanja otvara se glavni prozor (Slika 2.) sa prikazom izgubljenih objekata:



ID	Title	Date	Description	Location	Finder	Owner				
1	Kljucevi	8/27/2019	Kljucevi od kuće	20	tijana		Delete	Edit	Duplicate	Mine!
2	Naocare	8/27/2019	Naocare	2	tijana	dejan	Delete	Edit	Duplicate	
3	200 dinara	8/19/2019	200 dinara	5	dejan	admin	Delete	Edit	Duplicate	
8	Sveska	8/14/2019	Sveska iz srpskog	3	slobodan		Delete	Edit	Duplicate	

Slika 2. Glavni prozor sa prikazanim predmetima

Ako korisnik iz menija odabere opciju za prikaz korisnika, prikazuje mu se sledeći prikaz(Slika 3.):



Lost and found

Update account

Username	Name	Last name	User type
admin	admin	admin	Admin
dejan	Dejan	Jovanic	Regular user
tijana	Tijana	Jovanic	Admin
gordana	Gordana	Jovanic	Regular user
slobodan	Slobodan	Jovanic	Regular user

Slika 3. Prikaz svih korisnika za “Regularnog” korisnika

U slučaju da je korisnik administrator, korisniku su prikazane dodatne mogućnosti obezbeđene njegovom privilegijom(Slika 4.):

Lost and found

Update account

Add person

Username	Name	Last name	User type	
admin	admin	admin	Admin	
dejan	Dejan	Jovanic	Regular user	Delete
tijana	Tijana	Jovanic	Admin	
gordana	Gordana	Jovanic	Regular user	Delete
slobodan	Slobodan	Jovanic	Regular user	Delete

Slika 4. Prikaz svih korisnika za Administratore

Za izradu same aplikacije, upotrebljene su sledeće tehnologije i obrasci:

- Entity Framework
- WCF Duplex
- Metadata Exchange (MEX)
- log4net biblioteka
- MaterialDesign
- WPF
- Data i Command Binding
- Obrasci:
 - MVVM
 - Command
 - Observer (prilagođen)
 - Bridge
 - Proxy (prilagođen)
 - Prototype
 - Singleton

Primenjeni obrasci

MVVM

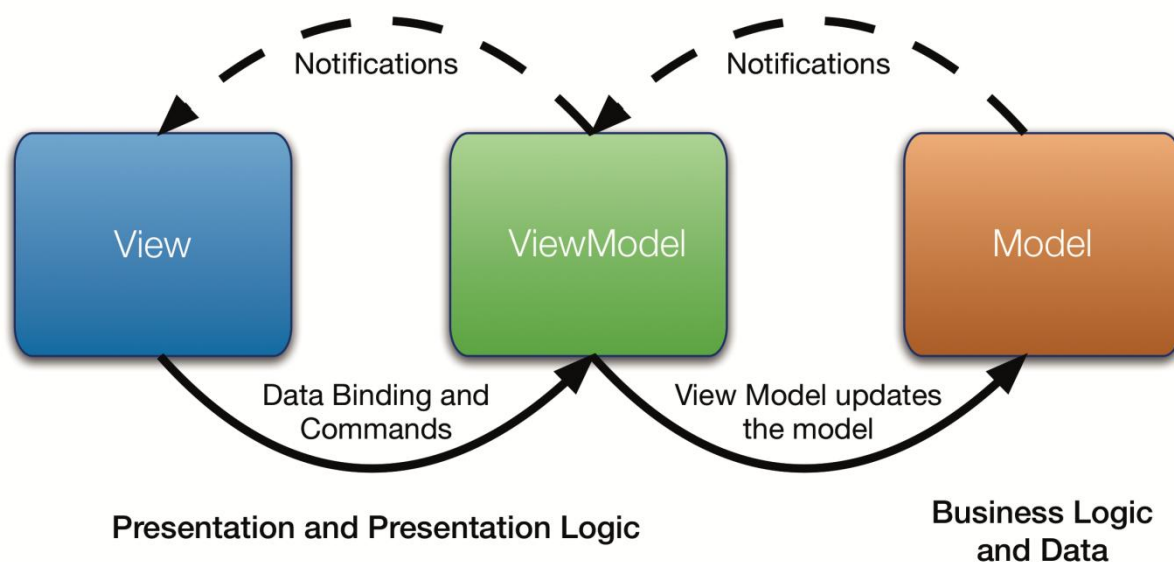
Model-View-ViewModel(MVVM) predstavlja obrazac čija je namena separacija zaduženja u klijentskoj aplikaciji (Slika 5.).

Sastoji se iz tri dela:

- Model - Model podataka, reprezentuje aplikativne podatke
- View - Prikaz podataka
- ViewModel - Interakcija korisnika sa podacima.

Sam tok komunikacije je sledeći:

- Korisniku se prikazuje određeni View
- Korisnikovu interakciju sa View-om obrađuje ViewModel
- ViewModel prosleđuje podatke Model-u na dalju obradu



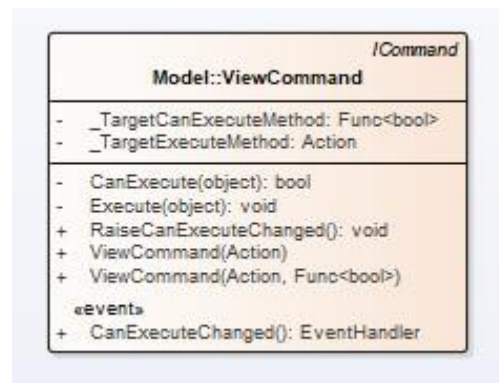
Slika 5. MVVM

Command

Komanda obrazac enkapsulira akcije u objekte. Često se koristi za:

- Parametrizaciju objekta operacijom koju treba da izvrši
- Za poništavanje akcija (Undo/Redo)
- Slabije sprezanje sistema

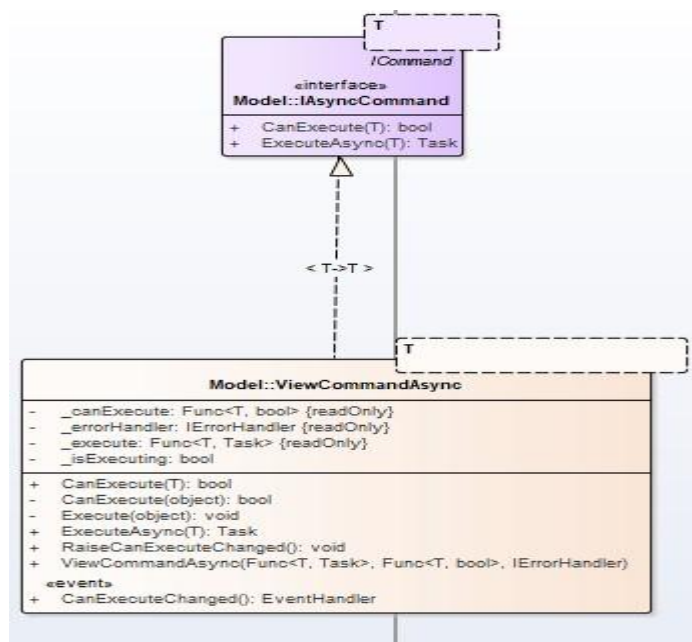
U projektu se koriste tri vrste komandi. Prva vrsta (Slika 6.) jeste sinhrona komanda, koja se oslanja na ugrađeni .Net interfejs *ICommand*. Namena ove vrste komandi je pri upotrebi **Command Bindinga** u komunikaciji sa korisničkim interfejsom.



Slika 6. Prva vrsta komandi

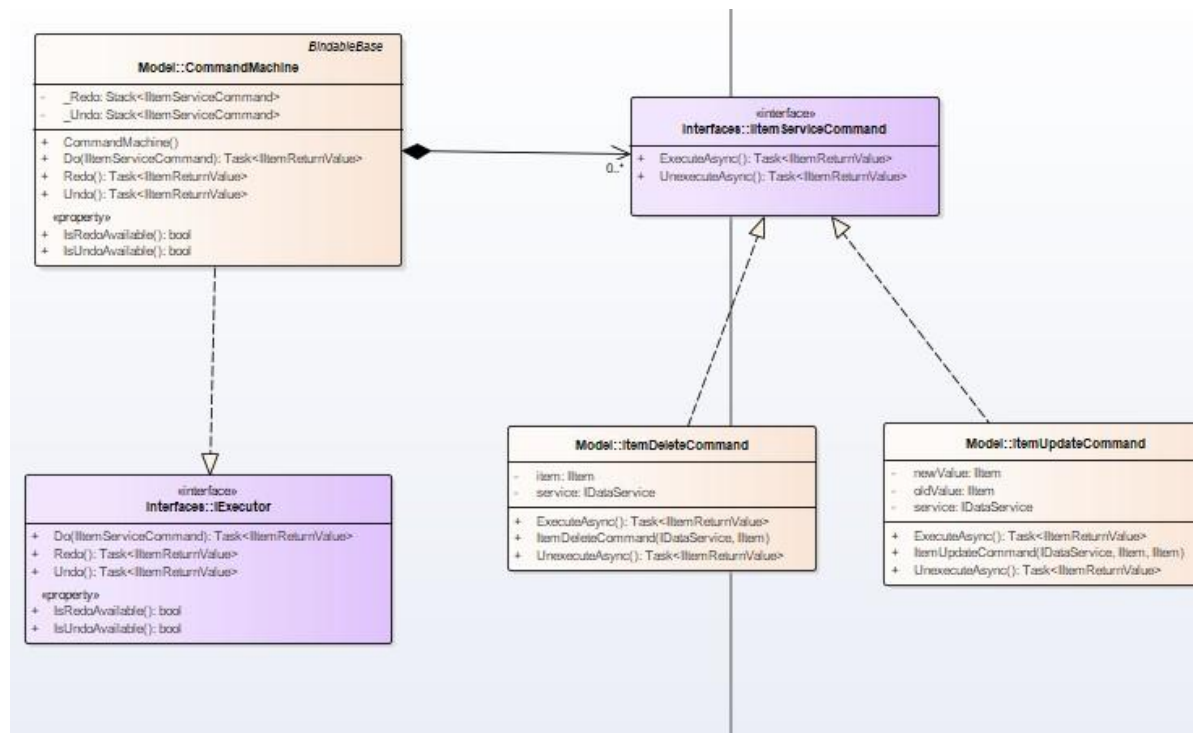
Druga vrsta (Slika 7.), komandi jeste asinhrona komanda koja se oslanja na izvedeni interfejs *IAsyncCommand*, koji nasleđuje .Net-ov *ICommand*. Interfejs i klasa su realizovani u parametrizovanoj i nepametrizovanoj varijanti. Ona za svaku akciju pokreće zasebnu nit – “Task”, koji se izvršava nezavisno od glavne niti.

Namena ove vrste komandi je takođe za upotrebu **Command Bindinga** u komunikaciji sa korisničkim interfejsom.



Slika 7. Asihrona parametrizovana komanda

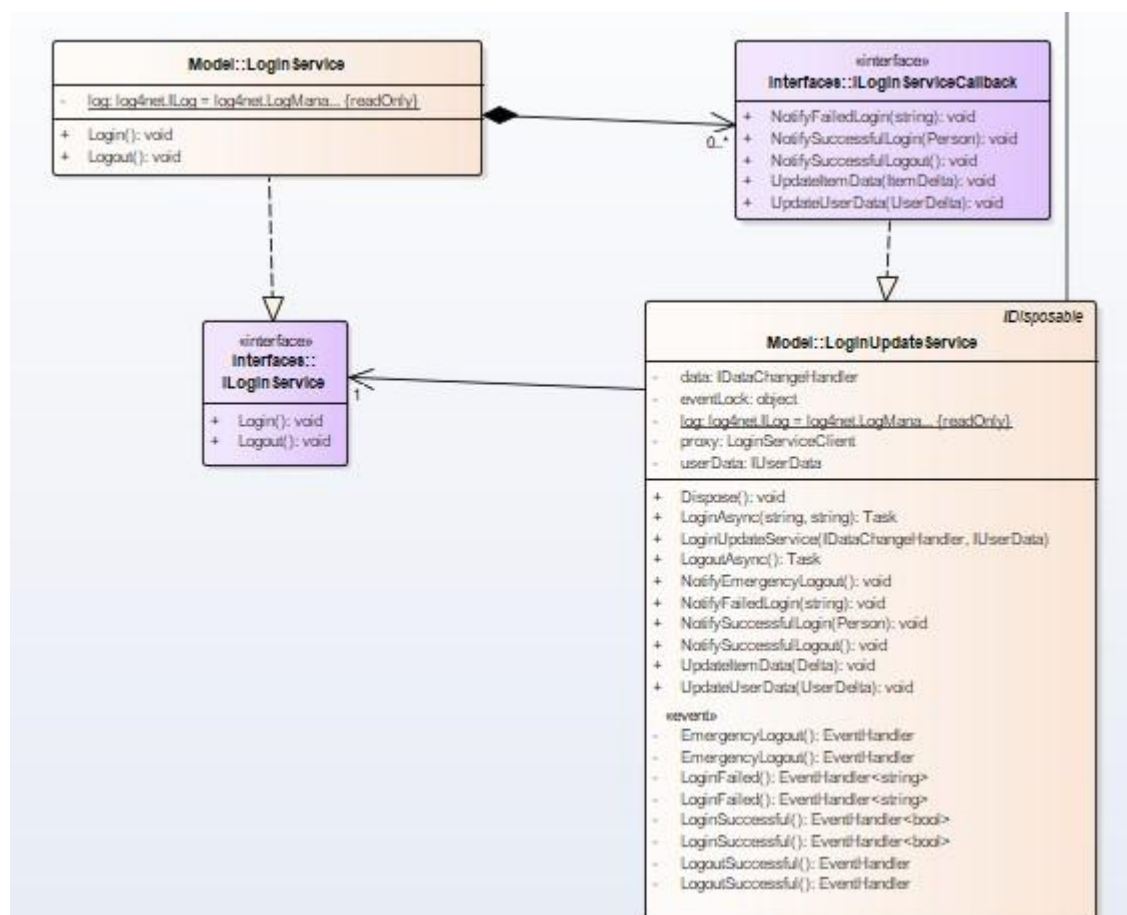
Treća vrsta komandi (Slika 8.) jeste asinhrona komanda, koja se oslanja na *ItemServiceCommand* interfejs. Ona omogućava enkapsulaciju zahteva ka servisu u komande, koje se posle mogu revertirati ili ponovo pozvati (Undo/Redo). Za izvršavanje operacija nad komandama, kao i njihovo skladištenje je zadužena *CommandMachine* klasa koja implementira *IExecutor* interfejs.



Slika 8. Komande za Undo/Redo operacije

Observer

Posmatrač obrazac omogućuje notifikaciju zavisnih klasa o novonastalim promenama u primarnoj klasi.

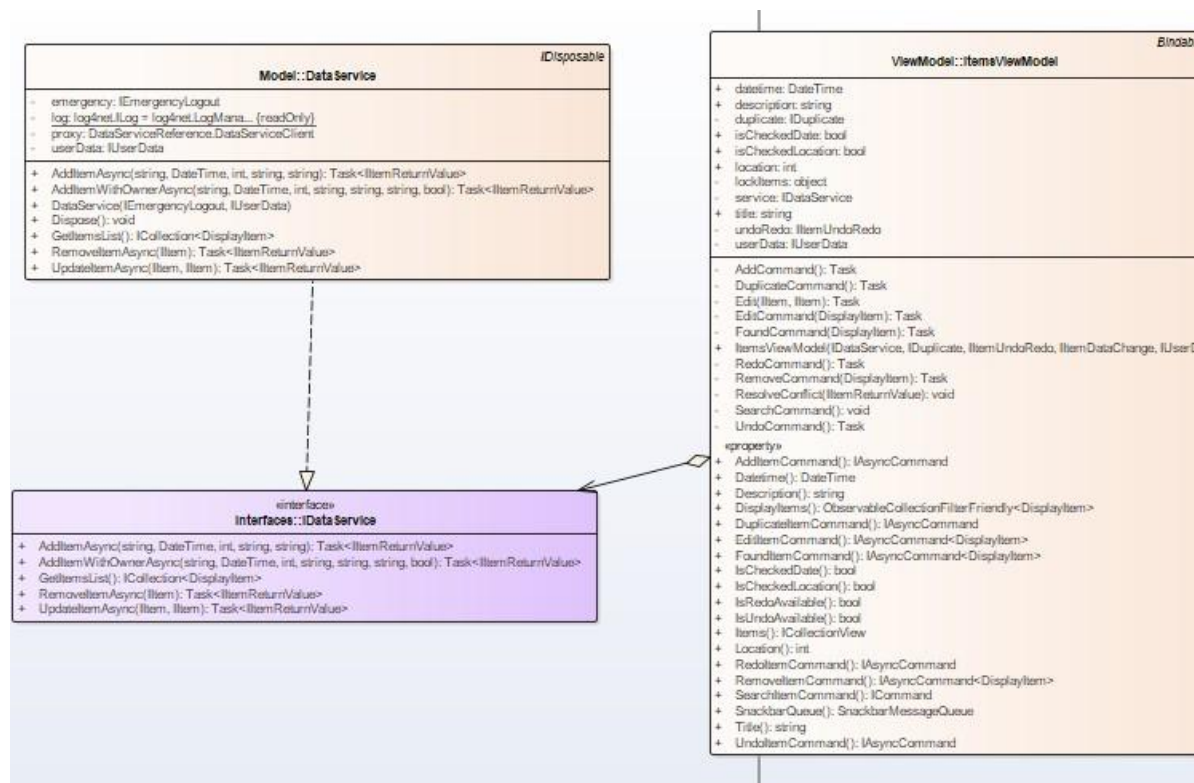


Slika 9. WCF Duplex posmatrač

U projektu je upotrebljen je *WCF Duplex* (Slika 9.), mehanizam uz pomoć kojeg i klijent i server komuniciraju putem jednosmernih kanala. Klijenti kontaktiraju servis koji implementira interfejs *ILoginService* i na njemu pozivaju metodu **Login**. Na poziv ove metode, server skladišti povratni kanal sa potpisom **ILoginCallback** interfejsa, koji implementira klijent. Pri svakoj nastaloj promeni, server obaveštava sve svoje trenutno pretplaćene klijente pozivajući metode **UpdateItemData**, odnosno **UpdateUserData**, pri čijem pozivu klijenti reaguju na promenu podataka. U svakom momentu, klijent može pozvati metodu **Logout**, pri čemu server poziva metodu **NotifySuccessfulLogout ILoginCallback** interfejsa i brise ga iz liste trenutno pretplaćenih (“ulogovanih”) korisnika.

Bridge

Most obrazac nalaže razdvajanje klasa na dve komponente, komponentu koja izvršava logiku i komponentu koja prezentuje podatke. Te dve komponente treba razdvojiti u dve zasebne klase, i one treba da budu u kompozicionoj vezi, i da pritom razgovaraju uz pomoć apstrakcije.

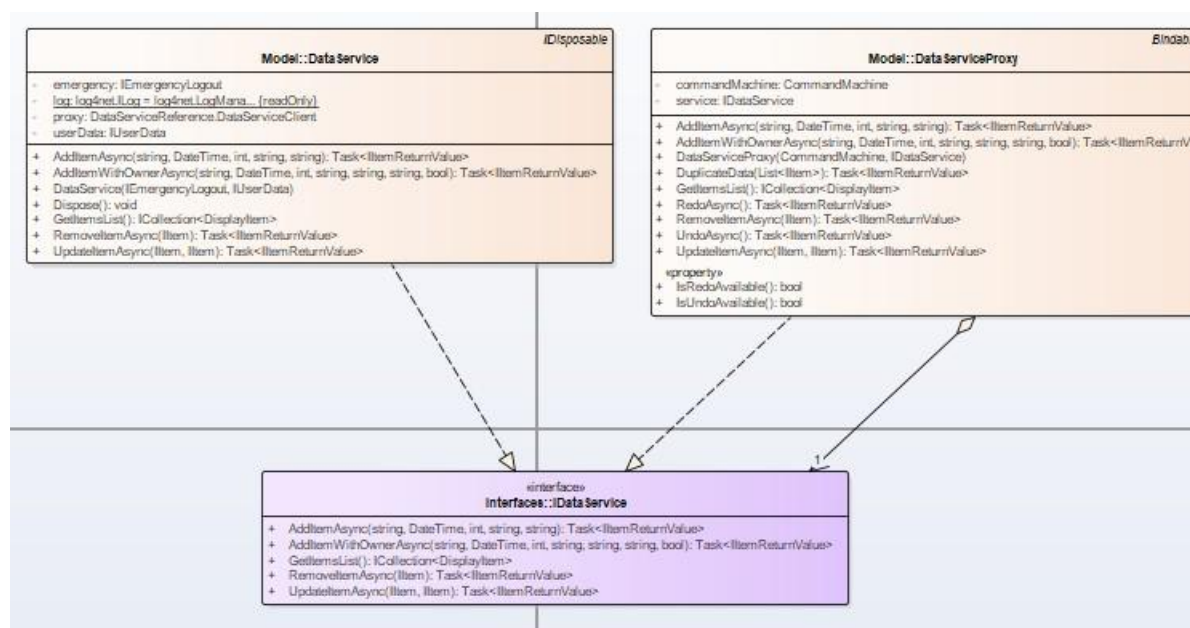


Slika 10. Bridge

U projektu(Slika 10.) se **Bridge** obrazac koristi u komunikaciji ViewModel-a sa Model-om, odnosno *WCF Servisom*. ViewModel u svom konstruktoru prima referencu na interfejs koji mu je potreban za komunikaciju sa korespondentnim servisom. Na ovaj način Model može biti promenjen, bez potrebe za menjanjem ViewModela, ali takođe pospešuje fundamentalni princip MVVM obrazca, koji nalaže da ViewModel treba da služi samo kao sprega između View-a i Model-a, bez potrebe za poznavanjem detalja ijedne strane.

Proxy

Proksi obrazac obezbeđuje pozivanje klase posredno preko druge, proksi, klase koja može izvršiti kako kontrolu pristupa klasi, tako i neke druge operacije neposredno pre ili posle izvršenja operacije primarne klase. Važna napomena je da proksi klasa ne menja ponašanje primarne klase, odnosno rezultat operacije nad proksijem ce biti isti kao i rezultat poziva operacije nad primarnom klasom. Proksi implementira iste interfejsse kao i primarna klasa.

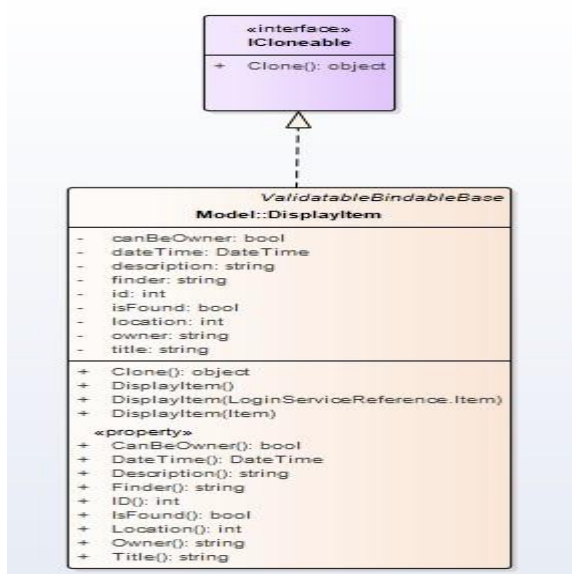


Slika 11. Proksi

U projektu (Slika 11.), proksi se koristi za “Wrap”-ovanje klase klijenta koja poziva servis za upravljanje “Item”-ima. Obe klase implementiraju *IDataService* interfejs. Proksi klasa je u kompoziciji sa *CommandMashine* klasom, koja izvršava pozive i pruža usluge Undo/Redo operacija. Proksi sve klijentske zahteve obavlja komandama, koje prosleđuje *CommandMashine* klasi dalje na izvršavanje. Ovakva struktura dovodi do toga da se servis može pozivati i bez komandi i undo/redo funkcije, ali i do toga da ViewModel nije svestan detalja funkcionisanja navedenih funkcija.

Prototype

Prototip obrazac je obrazac koji nadležnost kreiranja objekta klase ostavlja samom objektu, i tako otklanja potrebu za poznavanje kreacionih detalja objekta.

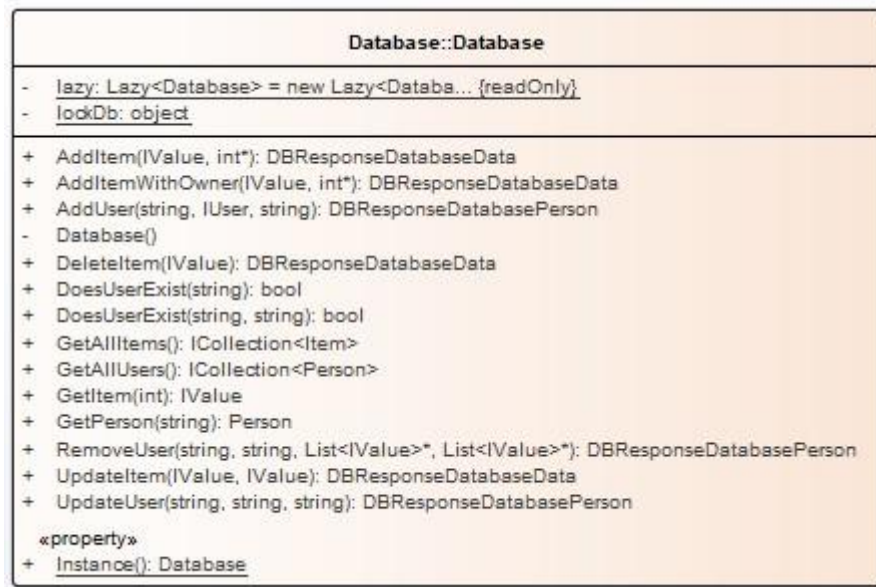


Slika 12. Prototip

U projektu (Slika 12.), sve verzije "Item" i "Person" klasa koje nisu vezane za bazu podataka, implementiraju .NET-ov ugrađeni interfejs *ICloneable*. Ovaj interfejs u sebi sadrži jedinu metodu, *Clone*, čiji je zadatak da vrati referencu na novonastali objekat. To dovodi do pojednostavljenja čestog kopiranja objekata.

Singleton

Unikat obrazac nam osigurava postojanje samo jedne instance klase, kao i njen globalni pristup.



Slika 13. Unikat

U projektu (Slika 13.) se unikati koriste kod čuvanja referenci ka klijentima i kod pristupa bazi podataka na serveru, kao i kod čuvanja lozinke i korisničkog imena trenutnog korisnika na klijentu. Ovaj pristup je odabran zbog potrebe za globalnim pristupom tim podacima. Takođe, na serveru se tada olakšava korišćenje serverskih objekata kraćeg veka, ali kao manu ima veći nivo sprezanja klasa.