Fakultet tehničkih nauka

Novi Sad

Projektna dokumentacija za projekat “Bolnica”

Razvoj višeslojnih aplikacija u elektroenergetici

1. Uvod – opis projekta

Tema ovog projekta je “Bolnica” klijent – server aplikacija, koja predstavlja informacioni sistem za upravljanje bolnicama.

Omogućeno je kreiranje novih bolnica, dodavanje ljekara i pacijenata u bolnice. Vođenje evidencije o bolnicama, ljekarima i pacijentima se čuva u Entity Framework bazi podataka, a aplikacija se oslanja na MVVM(Model View View-Model) obrazac.

Obrazac je podijeljen u nekoliko dijelova, a jedan od tih je Model koji predstavlja apstrakciju stvarnih modela koji čine realan sistem bolnice i koji su bitni za izgradnju ovog informacionog sistema, a tu spadaju: Bolnica, Ljekar(osnovne informacije o ljekaru koje se potrebne za upis u bolnicu) i Pacijent (osnovne informacije o pacijentu).

View omogućuje izgled korisničke aplikacije. Pored View-a tu se nalazi i View Model – posrednik između View-a i Modela podataka, koji sadrži logiku zaduženu za prezentaciju podataka i za navigaciju kroz korisnički interfejs.

2. Komponente informacionog sistema

Gledajući sa najvišeg nivo, informacioni sistem se dijeli na server aplikaciju i više klijentskih aplikacija, odnosno uobičajena server-klijent arhitektura.

Naš informacioni sistem može da se podjeli na sledeće komponente:

- Common – sadrži interfejse za komunikaciju između klijenta i servera, takođe tu su sadržani i modeli podataka korišteni u razmjeni između servera i klijenta. U Common-u ne postoji nikakva konkretna implementacija.
- ServerBolnica – sadrži implementaciju interfejsa sadržanih u Common-u koji predstavljaju servise koje klijent može da koristi. Takođe zadužen je za pristup bazi, smještanje podataka u bazu kao i za autentifikaciju korisnika. Ovaj dio se moze podijeliti na sloj pristupa bazi podataka i sloj poslovne logike.
- KlijentBolnica– Klijentska aplikacija sa grafičkim korisničkim interfejsom. Zadužena da korisniku omogući interakciju sa serverom. U projektovanju ovog dijela je praćen MVVM šablon.

3. Tehnologije korištene u razvoju informacionog sistema

Pored osnovnih funkcionalnosti .NET framework-a korištene su još dodatne tehnologije koje omogućije brz i efikasan razvoj pojedinih slojeva aplikacije:

- Windows Communication Foundation (WCF) – tehnologija korištena za komunikaciju između klijenta i servera. Omogućuje lako i brzo pozivanje udaljenih procedura servera od strane klijenta i serijalizaciju podataka koji se koriste u komunikaciji. Tehnologija je korištena u klijentskoj i serverskoj aplikaciji, a u dijelu „Common“ se nalaze zajednički interfejsi i modeli podataka korištene u komunikaciji.
- Windows Presentation Foundation (WPF) – tehnologija korištena u klijentskoj aplikaciji za izradu korisničkog interfejsa, poštujući MVVM šablon pri projektovanju i razvoju. Omogućuje izradu interfejsa koji je prenosiv između različitih platformi, a MVVM šablon pomaže u jasnom razdvajanju „pogleda - View“ koji prikazuje podatke i interaktuje sa korisnikom i „pogled-modela – View Model“ koji čuva i obrađuje podatke
- Entity framework – tehnologija korištena u serverskoj aplikaciji koja omogućuje brz, i siguran pristup bazi podataka. Omogućuje upravljanje bazom kao sa uobičajenim kolekcijama.

4. Korišćeni obrasci

U dizajnu i razvoju aplikacije korišteni su sledeći obrasci:

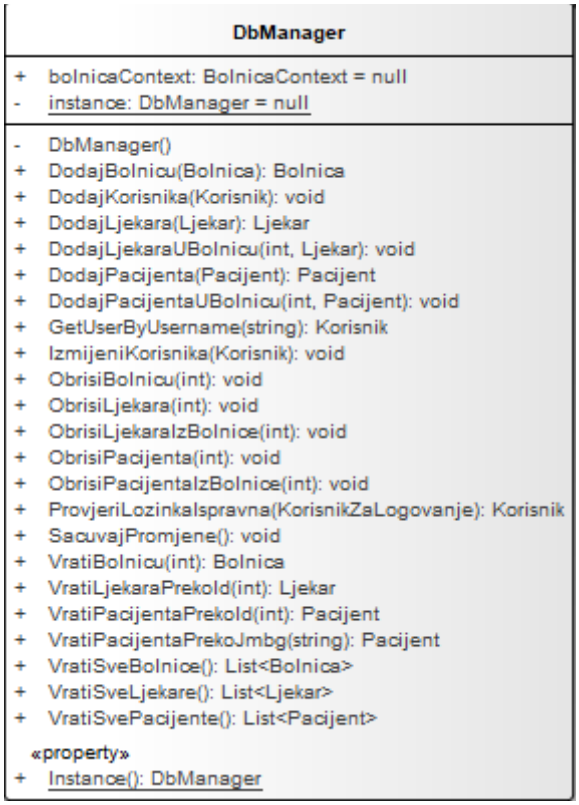
- Singleton
- Prototype
- Command
- Fascade
- Observer
- Proxy uz WCF

Singleton

Obezbeđuje da klasa ima samo jednu instancu i daje globalni pristup toj instanci. Odgovorna za kreiranje i rad sa svojom sopstvenom jedinstvenom instancom.

U izradi projekta singleton obrazac se koristio u klasama DbManager i SesijaManager smještene u folderu “PristupBaziPodataka". Klasa SesijaManager je zadužen za praćenje i upravljanje sesijom ulogovanih korisnika.

DbManager je klasa koja je zadužena za pristup bazi i preko njenih metoda svi ostali servisi pristupaju podacima iz baze. Ovde je izabran singleton za izradu DbManager-a zato što olakšava implementaciju i pristupačan je svima preko jedne pristupne tačke.



Slika1. Klasa DbManager

Prototip

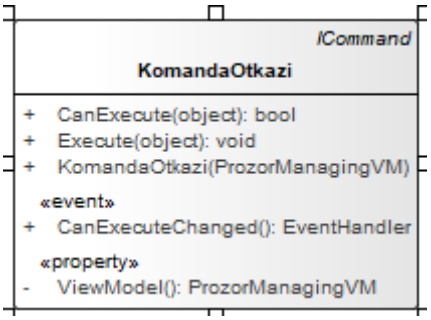
Definiše mehanizam kako da se pravljenje objekta-duplikata određene klase povjeri posebnom objektu date klase, koji predstavlja prototipski objekat te klase i koji se može klonirati. Ukratko, govori kako klonirati određenu instancu objekta.

U projektu prvenstveno je korišten zbog potrebne funkcionalnosti da se bolnica može klonirati u bazi. Implementiraju ga klase Bolnica, Ljekar i Pacijent.

Command

Enkapsulira zahtev za izvršenjem određene operacije u jedan objekat Umjesto da se direktno izvrši određena operacija, kreira se objekat-komanda, koji se potom prosljeđuje na izvršenje.

Ovaj obrazac kao prirodan i on je u tom vidu često korišten implementirajući komande korisničkog interfejsa kroz ICommand interfejs.



Slika3: Definicija KomandaOtkazi

Da bi se izbjeglo da svaka klasa implementira ICommand za svako dugme na korisničkom interfejsu, korišćen je RelayCommand koji imeplementira ICommand, a zahtjeve prosljedjuje kao delegat funkcijama koje je dobio u parametrima konstruktora.

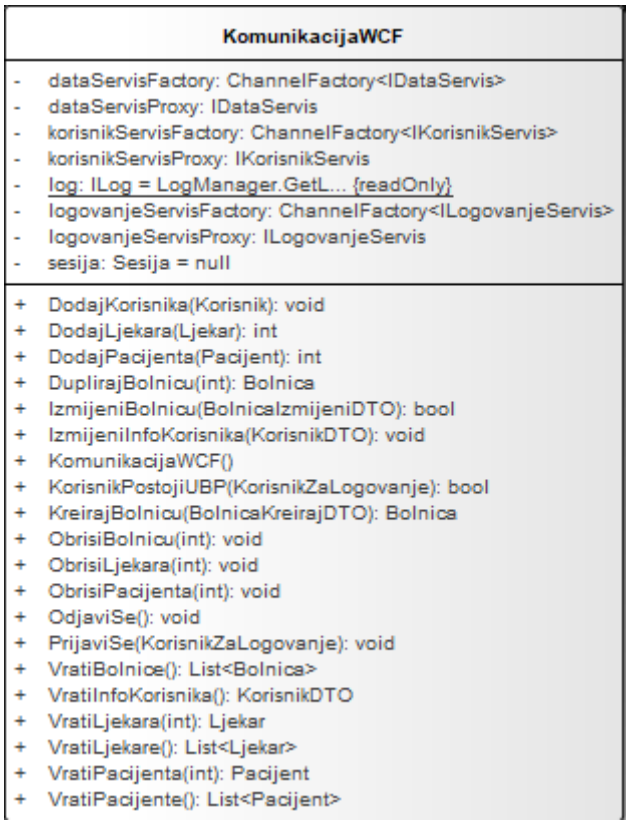
```
DodajLjekaraKomanda = new RelayCommand(DodajLjekaraUTabelu);
ObrisiLjekaraKomanda = new RelayCommand(ObrisiLjekara, SelektovanLjekar);
DodajPacijentaKomanda = new RelayCommand(DodajPacijenta);
ObrisiPacijentaKomanda = new RelayCommand(ObrisiPacijenta, SelektovanPacijent);
DodajBolnicuKomanda = new RelayCommand(SacuvajBolnicu, ValidacijaSacuvajBolnicu);

UndoKomanda = new RelayCommand(commandExecutor.Undo, commandExecutor.ValidacijaUndo);
RedoKomanda = new RelayCommand(commandExecutor.Redo, commandExecutor.ValidacijaRedo);

OtkaziKomanda = new RelayCommand(ZatvoriProzor);
```

Slika4: Primjer upotrebe RelayCommand-a u klasi DodajBolnicuVM.cs

Command šablon se koristio i pri implementaciji Undo/Redo funkcionalnosti nad listom Ljekara i listom pacijenata. Postoji intefejs IUndoKomanda koja definiše metode za realizaciju Undo/Redo komandi. Klasa CommandExecutor je zadužena za upravljanje nad komandama i omogućavanje Undo i Redo funkcionalnosti.

Fasada

Namjena ovog paterna je pružanje jedinstvenog interfejsa cijelog podsistema radi lakšeg korišćenja. Predstavlja interfejs na višem nivou sa ciljem minimizacije komunikacije i zavisnosti između klasa podsistema.



Slika 5: Upotreba fasade za komunikaciju sa servisima

Fasada je korišćena u cilju da sakrije od korisnika rukovanje sa 4 tabele koje su nalaze u bazi. To su: Bolnice, Ljekari, Pacijenti i Korisnici.