Fakultet tehničkih nauka

Novi Sad

# Projektna dokumentacija

Razvoj višeslojnih aplikacija u elektroenergetici

**Biljana Vukelić**

**PR20/2016**

# *SADRŽAJ*

# UVOD - OPIS PROJEKTA

Tema ovog projekta je “Parcela sa cvećem” klijent-server aplikacija, koja predstavlja informacioni sistem za upravljanjem parcelama cveća. Cilj projekta je napraviti aplikaciju koja omogućava kreiranje novih parcela, sa čuvanjem istih u Entity Framework bazi podataka, oslanjajući se na MVVM obrazac (Model-View-ViewModel). Korisnicima je omogućeno da kreiraju novu parcelu sa željenim izborom cveća i zemljišta, dodaju nove tipove cveća i zemljišta kao i da na jednostavan i brz način filtriranjem pronađu željene podatke, dupliraju iste, ažuriraju ili obrišu.

U Modelu su definisani podaci sa kojima se radi. View definiše izgled korisničkog interfejsa i omogućava prikaz traženih podataka. ViewModel predstavlja vezu između Modela i View-a. Reprezentuje podatake i navigaciju kroz korisnički interfejs.

Klijenti mogu imati dve uloge:

- Admin
- Korisnik

# KOMPONENTE SISTEMA

Sistem sadrži 3 komponente:

- Client
- Server
- Common

## *Klijentska aplikacija*

Klijentska strana je realizovana kroz WPF (*Windows Presentation Foundation*) tehnologiju koja omogućava korisnicima vrlo intuitivan rad sa aplikacijom. Zadužena je da korisniku omogući interakciju sa serverom, praćeno MVVM šablonom.

## *Serverska aplikacija*

Serverska strana je realizovana kroz WCF(*Windows Communication Foundation*) tehnologiju i komunicira sa bazom podataka u cilju usluživanja svih korisničkih zaheva. Sadrži implementaciju interfejsa koji se nalaze u Common-u, predstavljeni kao servisi koje klijent može da koristi.

### *Common*

Sadrži sve definicije interfejsa potrebnih za komunikaciju između klijenta i servera, kao i klase koje predstavljaju modele podataka koji su neophodni za razmenu između servera i klijenta.

## FUNKCIONALNOST

Pri pokretanju projekta, automatski se podiže i serverska i klijentska strana. Ukoliko ne postoje, kreiraju se baze na serverskoj strani-baza parcela, korisnika, cveća i zemljišta.

Na klijentskoj strani se otvara prozor za prijavu korisnika. Na sistemu trenutno postoje admin i "običan" korisnik sa kredencijalima *admin-admin* i *obican-obican*.

U slučaju neuspešne prijave, korisnik dobija poruku o pogrešnom unosu korisničkog imena/lozinke.


*Slika 1.* Forma za prijavu korisnika

Nakon uspešne prijave otvara se glavni prozor koji predstavlja radnu površinu.


*Slika 2.* Prikaz glavnog prozora

### *Admin*

Ukoliko je prijavljeni korisnik admin, na radnoj površini se pojavljuje dugme sa atributom „Visibility“ koje se prikazuje samo administratorima. Klikom na opisano dugme, otvara se prozor *(Slika 2)* pomoću kog admin može da rukuje korisnicima.

Samo admin može da vidi prikaz svih korisnika u sistemu i da doda novog korisnika, sa jedinstvenim korisničkim imenom. Osim toga, može da vrši izmene podataka nad datim korisnikom – lozinku, ime, prezime i ulogu. Korisničko ime ne može da se menja, ono ostaje jedinstveno. Pored dodvanja i izmene korisnika, admin može da izvrši brisanje korisnika. Da bi mogao da izvrši izmenu ili brisanje, neophodna je selekcija korisnika nad kojim želi da izvrši jednu od te dve akcije.

Kako bi se izvršilo dodavanje ili izmena korisnika, neophodno je popuniti sva polja koja postoje u prozoru koji mu se otvori. U  suprotnom neće imati mogućnost klika na dugme za potvrdu dodavanja/izmene.


*Slika 3.* Prozor za upravljenje korisnicima

### *Običan korisnik*

Nakon uspešnog prijavljivanja, na radnoj površini se pojavljuje dugme sa atributom “Visibility” koje se prikazuje samo običnim korisnicima. Klikom na to dugme,otvara se novi prozor koji korisniku omogućava pregled svih njegovih informacija, kao i mogućnost izmene nad istim. Može da izmeni svoje ime, prezime i lozinku dok ostale podatke ne može da menja.

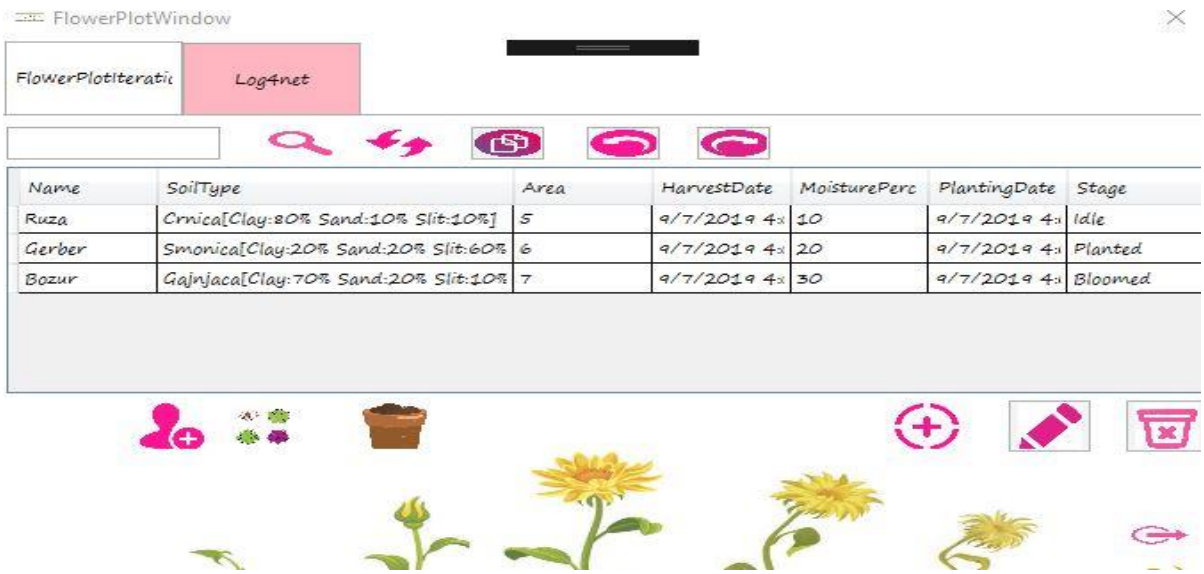
Korisnik ima mogućnost pretrage parcela, osvežavanja prikaza i dupliranja odabrane parcele. Najpre je neophodna selekcija željene parcele koju želi da duplira. Svaku promenu nad objektom korisnik može da vidi u tabeli,koja se nalazi na sredini prozora.

Korisnik može da doda novi tipa cveća koje želi da zasadi, klikom na dugme, otvara se novi prozor i kako bi uspešno dodao cvet, obavezan je unos naziva cveta koji mora biti jedinstven. U suprotnom, ako taj cvet već postoji u bazi, dodavanje će biti neuspešno.

Osim dodavanja novog tipa cveta, korisnik ima opciju da sam kreira tip zemljišta na kom želi da posadi cvet. Neophodno je uneti jedinstveni naziv zemljišta kao i da procenat gline,peska i mulja u sumi bude 100% , u suprotnom će dodavanje biti neuspešno.

Dugme za dodavanje nove parcele, nudi mogućnost izbora samo onih zemljišta i cveća koja već postoje u bazi. Na jednoj parceli se gaji samo jedan tip cveća. Pri započinjanju iteracije, svaka parcela je u stanju - idle. Korisnik ima mogućnost izmene svake parcele, i tek nakon selektovanja željene parcele može da izvrši izmene nad njom. Samo u početnom stanju – idle, ima mogućnost promene tipa zemljišta i cveta. Iz stanja idle, može da ode u stanje posađen – planted iz kog prelazi u fazu cvetanja. Kada cveće procveta ono se seče i započinje se nova iteracija.

Kada završi sa svojim aktivnostima, korisnik može da se odjavi na dugme koje se nalazi u donjem desnom uglu.




*Slika 4.* Radna površina korisnika

## BELEŽENJE DOGAĐAJA

Za klijentsku i servesku aplikaciju, omogućeno je beleženje događaja koji su se odigrali. Za te potrebe, korišćena je biblioteka log4net. Svaki događaj koji se odigra, beleži se u datotekama “serverLog” na serverskoj strani i “clientLog” na klijentskoj strani.

Omogućen je grafički prikaz svakom klijentu o izvršenim događajima koja je inicirao. Ukoliko želi da obriše trenutni prikaz tabelarne komponente, korisnik to može učiniti klikom na dugme koje se nalazi ispod opisane komponente.*(Slika 5.)*


Slika 5. Prikaz tabelarne komponente sa dugmetom “clear”

## CLIENTCREDENTIALS

Korisnički kredencijali obezbeđuju da klijent bude siguran da je pristupio pravom serveru. Implementacija ovog mehanizma je izvršena i na klijentskoj i na serverskoj strani. *(Slika 6.)*

Pri uspostavi WCF komunikacije,server proverava kredencija koje je dobio od klijenta. Ukoliko server autentifikuje klijenta, odobrava mu pristup.

```
public static void Host()
{
    NetTcpBinding binding = new NetTcpBinding();
    binding.Security.Mode = SecurityMode.Transport;
    binding.Security.Transport.ClientCredentialType = TcpClientCredentialType.Windows;

    userServer = new ServiceHost(typeof(UserService));
    userServer.AddServiceEndpoint(typeof(IUserDatabase),
                                  binding,
                                  new Uri("net.tcp://localhost:4000/IUserDatabase"));
    userServer.Open();
```

Slika 6. Implementacija korisničkih kredencijala na serverskoj strani

# KORIŠĆENI OBRASCI

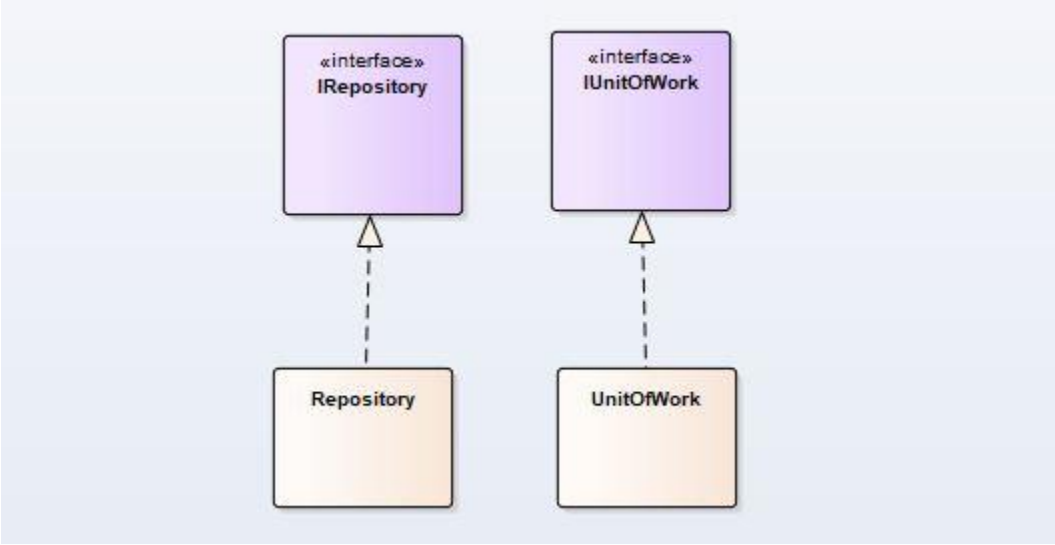
Projektni obrasci predstavljaju apstraktne primere rešenja na visokom nivou.

U dizajnu i razvoju aplikacije primenjeni su sledeći obrasci:

- Repository
- Singleton
- Command
- Prototype
- Observer

## *Repository pattern*

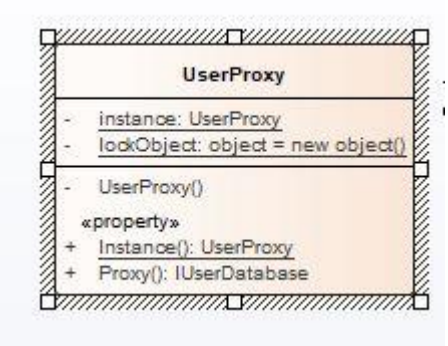
Repository obrazac je primenjen na serverskoj strani, gde se nalazi baza parcela, korisnika, cveća i zemljišta. Omogućava nezavisnost aplikacije od framework-a koji se koristi što dovodi do veće optimalnosti i preglednosti koda.




*Slika 7.* Repository pattern

### SINGLETON

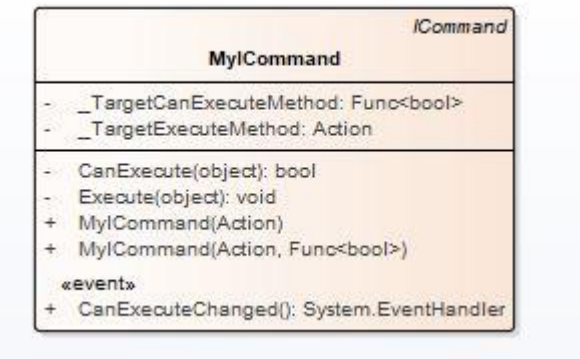
Obezbeđuje da klasa ima samo jednu instancu i daje globalni pristup toj instanci. Odgovorna je za kreiranje i rad sa svojom sopstvenom jedinstvenom instancom. Na klijentskoj strani postoje klase „UserProxy“, „FlowerPlotIterationProxy“, „FlowerProxy“ i „SoilProxy“ koje služe za kreiranje kanala između klijenta i servera u cilju potrebne komunikacije. Te klase su realizovana kao Singleton, kako bi se obezbedilo da one imaju samo jednu instancu kojoj je omogućen globalni pristup.




Slika 8. UserProxy

### COMMAND

Komanda(*Command*) obrazac enkapsulira zahtev za izvršenjem određene operacije u jedan objekat. Komanda se oslanja na ugrđeni C# interfejs *ICommand*. U ovom projektu namena komande je za upotrebu *Command Bindinga* za korisnički interfejs i sadrži metode *Execute* i *CanExecute*. Ukoliko je ispunjen određeni uslov za kliktanje na željeno dugme, automatski se izvršava metoda *CanExecute*, dok se *Execute* poziva pri samom kliktanju na dato dugme.




*Slika 9.* Definicija komande

## UNDO/REDO KOMANDA

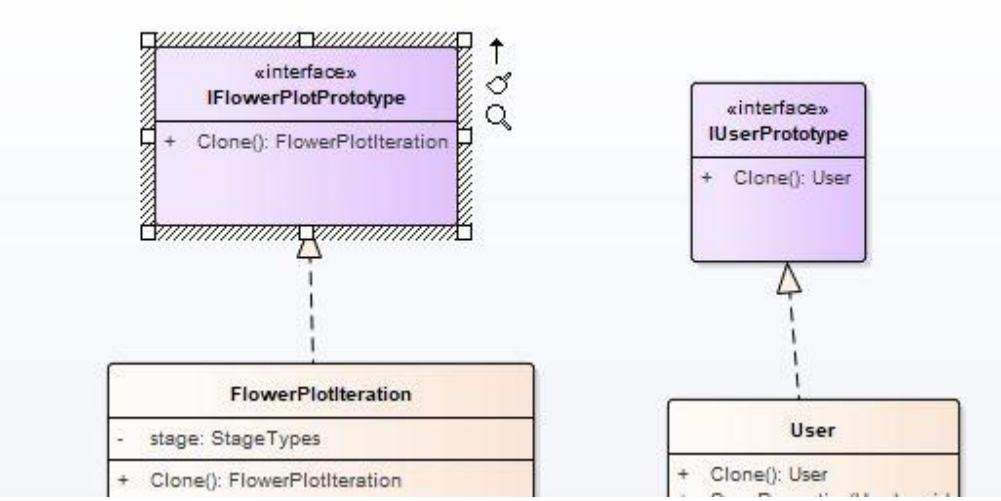
U ovom projektu postoji mogućnost undo/redo komande samo za brisanje željene parcele. Nakon selektovanja parcele sa cvećem i brisanja iste, komanda undo vraća dati objektat dok će komanda redo biti moguća samo nakon izvršavanja undo komande. Undo komanda se sastoji od dve metode: *CanUndo* i *ExecuteUndo*. *CanUndo* se automatski izvršava kada je ispunjen određen uslov za kliktanje na želejno dugme. *ExecuteUndo* se izvršava nakon što korisnik klikne na dugme Undo. Analogno važi za komandu *Redo*.

Ideja Undo/Redo komandi je bazirana na implementaciji interfejsa IUndoRedo koji će se sastojati od dve metode AddUndo() i AddRedo(). Formiraju se dva steka,jedan koji služi za undo komandu drugi za redo. Kada se izvrši bilo koja komanda nad željenim objektom, na Undo stek se dodaje izvršena komanda,kako bi korisnik imao mogućnost poništavanja. Nakon izvršene Undo komande, ona se uklanja sa undo steka i smešta na Redo stek.

Prethodno opisana ideja undo/redo komandi može biti unapređenje ovog projekta.

## Prototype

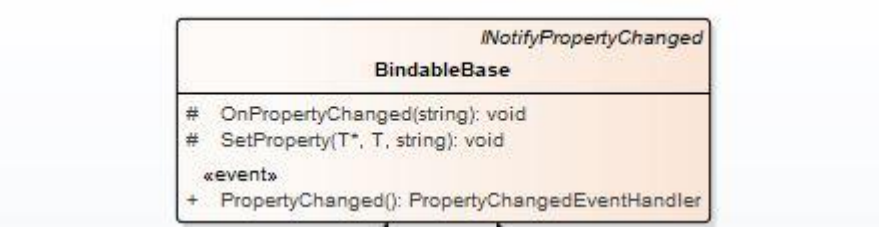
Definiše mehanizam kako da se pravljenje objekta-duplikata određene klase poveri posebnom objektu date klase, koji predstavlja prototipski objekat te klase i koji se može klonirati. Naime govori kako klonirati određenu instancu objekta. U projektu ga implementiraju klase FlowerPlotIteration i User, zbog 2 potrebne funkcionalnosti. Prva je da se dati objekat može klonirati u bazi, a druga služi za mehanizam za editovanje tj da u slučaju odustanka od bilo kakve vrste izmene korisniku vratimo stare podatke.




Slika 10. Upotreba prototype šablona

### *Observer*

Omogućava da se promena sadržaja u jednom elementu, odmah pojavi i u svim delovima programa koji dati element prikazuju u nekom obliku. Implementirana ga klasa "*BindableBase*" pomocu C# ugrađenog mehanizma event-a.("*INotifyPropertyChanged*"). U okviru mehanizma za subscribe, binding predstavlja pretplatu na taj event.




Slika 11.Definicija *BindableBase*