FAKULTET TEHNIČKIH NAUKA
Univerzitet u Novom Sadu

DOKUMENTACIJA PROJEKTA

Razvoj elektroenergetskog softvera

školska 2017./2018. Godina

Studenti
Irina Jovanović PR 11/2015, Vuk Bajić E3 96/2014

Funkcionalnost klasa

Projekat se sastoji iz dva dela – funkcionalni deo projekta i testovi. Funkcionalni deo projekta se sastoji iz sledećih klasa: Common,Client,Service i BazaPodataka.

Uloga klase Common je da svojim sadržajem obezbedjuje uspešnu komunikaciju izmedju Client-a I Servica kao I uspešno slanje podataka uz to sadrži I konstruktore nekih klasa koje kreiraju odgovarajuće objekte.

Uloga klase Client je dvojaka. Prvi deo klase šalje podatke Service-u na zahtev korisnika koji trebaju da se učitaju u bazu podataka,dok drugi deo šalje zahtev za filtriranje I prikaz podataka ka Service-u.

Uloga klase Service je da prima podatke od Clienta. U zavisnosti od zahteva Clienta, u prvom slučaju on priprema podatke za upis u bazu podataka I prosleđuje ih klasi BazaPodataka. Druga funkcija klase Service je primanje zahteva Clienta filtriranju podataka. On prima parametre za filtriranje I takođe šalje klasi BazaPodataka.

Uloga klase BazaPodataka je dvostruka. Prva je da primljene podatke od Servica upiše u bazu podataka. Druga uloga je primanje zahteva za filtriranje podataka koju Service klasa prosleđuje od klase Clienta. BazaPodataka filtrira podatke, formira ih u objekat DataStatic, potom ih smesti u Listu I prosledi klasi Service..

Drugi deo koda su testovi klasa. Testirane su sve metode i klase projekta(sem svih Program.cs i klase Client).

Svrha testova je da se proveri funkcionalnost i da se utvrdi ispravnost rada metoda i konstruktora klasa kao I da se uoče eventualni problemi.

Funkcionalnost metoda

Metode Client

1. `Public MainWindow() – Inicijalizacija komponenata.`
2. `private void Prikazi_Click(object sender, RoutedEventArgs e)-slanje parametara za filtriranje.`
3. `private void buttonNadji_Click(object sender, RoutedEventArgs e) – Otvaranje File Explorera za pronalazak xml dokumenta za ucitavanje.`
4. `private void buttonUvezi_Click(object sender, RoutedEventArgs e) – Nakon odabira xml-a za uvoz slanje direktorijuma xml dokumenta Service-u.`

Metode Service

1. `public byte[] returnStatistic(string ime , string zem, int od, int to)- Dvostruka`

namena. Prva jeste da šalje klasi BazaPodataka zahtev za filtriranim podacima, nakon primanja filtriranih podataka ova metoda šalje Client-u iste podatke.

2 `public bool upisuBazu(string xml)`- šalje xml dokument za upis u bazu podataka klasi BazaPodataka.

3 `public bool prongozirana(string s)`-proverava da li učitan xml dokument sadrži prognozirane ili izmerene podatke.

4. `public bool provera(string xml)`- metoda proverava da li je xml dokument ranije učitavan.

5. `public List<Stavka> ucitajXML(string xml)`- učitava xml dokument.

6. `public bool stavkeNisuKorektne(ListStavki listStavki)`- metoda proverava da li su učitani podaci.

Metode BazaPodataka

1. `public void UpisiPrognozirano(List<Stavka> stavke, string xml)` upisuje prognozirane podatke u bazu podataka.

2. `public void UpisiOstvareno(List<Stavka> stavke, string xml)` upisuje ostvarene podatke u bazu podataka.

3 `public void upisiuCSV(string xml)`-metoda koja upisuje nazive učitanih xml-ova u CSV dokument radi provere da se ne učitaju dva ista xml- dokumenta.

4. `private void Serializuj(ListStavki lista, String imeFajla)`- vrši serijalizaciju liste I upisivanje u bazu podataka.

5 `private void DeSerijalizuj(String imeFajla)` vrši deserijalizaciju liste I isčitavanje iz xml dokumenta.

6. `private Baza()` je konstruktor koji poziva metodu LoadData().

7. `private void LoadData()` je metoda koja se poziva pokretanjem aplikacije. Ona učitava u RAM memoriju imena već učitanih dokumenata radi bržeg I lakšeg pristupa istom.

8. `public List<DataStatistic> VratiFiltriranoPodatke(string ime, String oblast, int datumOd, int datumDo)`- vrši iščitavanje I filtriranje podataka paralelno iz dva xml dokumenta, učitava ih u dve privremene liste I prosleđuje je metodi `private List<DataStatistic> potrebniPodaci(List<Stavka> prog, List<Stavka>ostv)` koja vraca spojene dve liste u obliku `List<DataStatistic>` I prosleđuje je Service-u.

9. `private List<DataStatistic> potrebniPodaci(List<Stavka> prog, List<Stavka>ostv)` prima dve liste I spaja ih u objekte tipa DataStatistic I vraća listu tih objekata.

Modeli podataka

1. DataStatic

Polja: `string reg, int sat, int prog, int izm, double dev.`

2. ListDataStatic

Polja: `private List<DataStatistic> data;`

3. ListStavki

Polja `private List<Stavka> stavke;`

4. Stavka

Polja:
```
private string reg;
        private int sat;
        private int load;
        private string fajlUcitavanja;
        private string vremeUcitavanja;
```

Testovi

Pravimo testove za konstruktore: proveravamo da li su parametri ispravni prilikom kreiranja novog objekta..

Svaka metoda je posebno istestirana putem jednog ili više testova kako bi se pokrio što veći opseg pokrivenosti koda I grananja.

Rad aplikacije

Rad aplikacije se izvršava pomoću WCF arhitekture.

Pri pokretanju, pred korisnikom je interfejs jednostavnog izgleda. Korisnik ima dve opcije za rad. Prva opcija jeste uvoz xml dokumenata. Pri uvozu Client-ska strana pravi konekciju sa Service-om, prosleđuje mu putanju do xml dokumenta, Service proverava da li je taj xml dokument već učitan, ukoliko nije učitava taj xml dokument. Nakon toga proverava da li se xml dokument odnosi na prognozirane ili izmerene podatke. U zavisnosti od rezultata provere, Service šalje podatke klasi BazaPodataka koja ih upisuje u odgovarajuću bazu podataka.

Druga opcija je suprotna od prve. Korisnik može da izabere datum, region i vremenski raspon koji ga zanima radi računanja devijacije između prognozirane i izmerene potrošnje. Izabrani parametri se šalju Service-u koja ih prosleđuje BaziPodataka. BazaPodataka proverava da li su učitana oba xml dokumenta koji su potrebni za računanje devijacije, u slučaju da jeste BazaPodataka iščitava podatke iz baze, filtrira ih po zahtevu korisnika. Nakon filtriranja prosleđuje listu podataka Service-u koji ih prosleđuje Client-u na čijem se interfejsu ispisuju filtrirani podaci.