

DOKUMENTACIJA PROJEKTA

Evidencija prognozirane i ostvarene potrošnje električne energije

Predmet:

Virtuelizacija procesa

Autor:

Zlatko Čikić PR 1/2020

Sadržaj

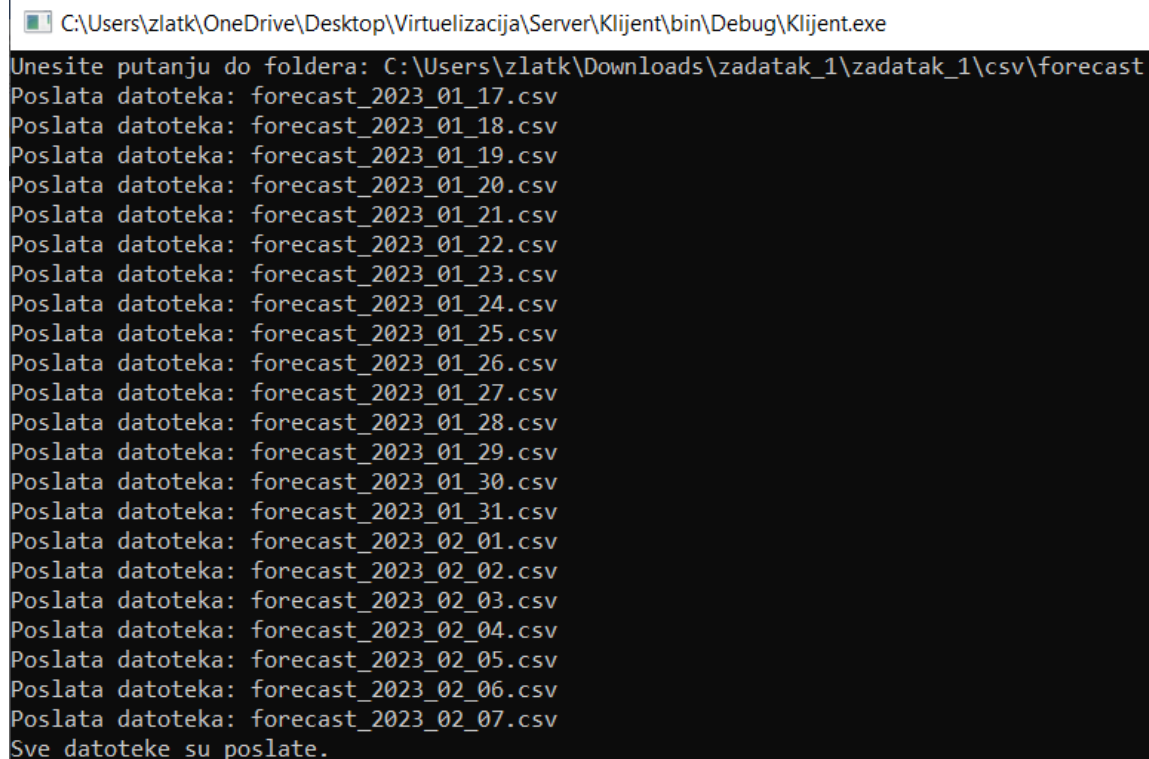
1. Opis projektnog zadatka
2. Arhitektura projekta
3. Opis interfejsa
4. Opis korišćenih tehnologija
5. Zaključak

Opis projektnog zadatka

Aplikacijom se uvoze podaci o planiranoj (prognoziranoj) i ostvarenoj potrošnji električne energije nakon čega se vrše odgovarajući proračuni. Podaci se uvoze iz CSV datoteka pri čemu naziv datoteke može biti na primer „forecast_2023_01_27.csv“ ili „measured_2023_01_27.csv“, forecast datoteka predstavlja podatke o prognoziranoj potrošnji, dok measured datoteka predstavlja podatke u ostvarenoj potrošnji.

Kako se vrši uvoz:

Korisnik bira opciju za uvoz podataka iz CSV datoteka putem korisničkog interfejsa odnosno unosi putanju ka direktorijumu u kojem se nalaze CSV datoteke i sve datoteke se učitavaju jedna za drugom.



```
C:\Users\zlatk\OneDrive\Desktop\Virtualizacija\Server\Klijent\bin\Debug\Klijent.exe
Unesite putanju do foldera: C:\Users\zlatk\Downloads\zadatak_1\zadatak_1\csv\forecast
Poslata datoteka: forecast_2023_01_17.csv
Poslata datoteka: forecast_2023_01_18.csv
Poslata datoteka: forecast_2023_01_19.csv
Poslata datoteka: forecast_2023_01_20.csv
Poslata datoteka: forecast_2023_01_21.csv
Poslata datoteka: forecast_2023_01_22.csv
Poslata datoteka: forecast_2023_01_23.csv
Poslata datoteka: forecast_2023_01_24.csv
Poslata datoteka: forecast_2023_01_25.csv
Poslata datoteka: forecast_2023_01_26.csv
Poslata datoteka: forecast_2023_01_27.csv
Poslata datoteka: forecast_2023_01_28.csv
Poslata datoteka: forecast_2023_01_29.csv
Poslata datoteka: forecast_2023_01_30.csv
Poslata datoteka: forecast_2023_01_31.csv
Poslata datoteka: forecast_2023_02_01.csv
Poslata datoteka: forecast_2023_02_02.csv
Poslata datoteka: forecast_2023_02_03.csv
Poslata datoteka: forecast_2023_02_04.csv
Poslata datoteka: forecast_2023_02_05.csv
Poslata datoteka: forecast_2023_02_06.csv
Poslata datoteka: forecast_2023_02_07.csv
Sve datoteke su poslate.
```

Primer izgleda korisničkog interfejsa

Format datoteke:

Svaki red u CSV datoteci sadrži informacije o satu i potrošnji u mW/h. Broj redova u datoteci mora odgovarati broju sati u danu za koji se vrši unos. Nevalidne datoteke se odbacuju i pravi se audit objekat koji se zapisuje u bazu podataka sa informacijom o grešci. Takodje se za svaku učitanu datoteku kreira objekat klase ImportedFile koji se upisuje u bazu podataka.

Uspešno učitavanje datoteke:

Ukoliko je datoteka validna onda se za svaki sat iz datoteke kreira po jedan objekat klase Load i zapisuje se u bazu podataka, ako objekat sa tim datumom i satom već ne postoji. Ako objekat sa pristiglim datumom i satom postoji, ažuriraju se polja objekta na osnovu pristiglih podataka.

```
<row>
  <TIME_STAMP>2023-01-17T00:00:00</TIME_STAMP>
  <FORECAST_VALUE>6426.01794633789</FORECAST_VALUE>
  <MEASURED_VALUE>N/A</MEASURED_VALUE>
  <ABSOLUTE_PERCENTAGE_DEVIATION>N/A</ABSOLUTE_PERCENTAGE_DEVIATION>
  <SQUARED_DEVIATION>N/A</SQUARED_DEVIATION>
  <IMPORTED_FILE_ID>100</IMPORTED_FILE_ID>
</row>
```

Primer zapisa u XML bazi podataka

Proračun odstupanja:

Nakon što su učitani podaci upisani u bazu podataka, servis pristupa proračunu odstupanja između prognozirane i ostvarene potrošnje po satu.

Odstupanje može biti apsolutno procentualno odstupanje ili kvadratno odstupanje.

Korisnik bira koje odstupanje želi da bude proračunato promenom podešavanja u App.config datoteci servisnog dela aplikacije.

```
<!--izbor da li zelimo racunanje apsolutnog procentualnog odstupanja (abs)-->  
<!--ili kvadratnog odstupanja (sqr)-->  
<add key="tipRacunanja" value="sqr"/>
```

Apsolutno procentualno odstupanje izračunava se po formuli:

$$\frac{|ostvarena\ potrošnja - prognozirana\ potrošnja|}{ostvarena\ potrošnja} * 100$$

Kvadratno odstupanje izračunava se po formuli:

$$\left(\frac{ostvarena\ potrošnja - prognozirana\ potrošnja}{ostvarena\ potrošnja} \right)^2$$

Kada se završi proračun odstupanja, aktivira ažuriranja baze podataka proračunatim podacima korišćenjem event i delegate.

Arhitektura projekta

Aplikacija je troslojna. Sadrži:

1. Korisnički interfejs
2. Servisni sloj
3. Baza podataka

Projekat obuhvata sledeće klase:

1. Load (polja): Id, Timestamp, ForecastValue, MeasuredValue, AbsolutePercentageDeviation, SquaredDeviation, ImportedFileId
2. ImportedFile (polja): Id, FileName
3. Audit (polja): Id, Timestamp, MessageType, Message

Korisnički interfejs:

U korisničkom interfejsu je potrebno da korisnik unese komandu, odnosno putanju do foldera sa CSV datotekama nakon čega se datoteke šalju servisnom sloju gde se vrši dalja obrada.

Servisni sloj:

Prihvata datoteke poslate od strane korisničkog interfejsa, vrši parsiranje datoteka i kao što je ranije napomenuto ukoliko datoteka nije validna kreira se objekat klase Audit koji se upisuje u bazu podataka, ukoliko je validna kreiraju se objekti klase Load koji se upisuju u bazu. Takođe se za svaku datoteku kreira objekat klase ImportedFile koji se takođe upisuje u bazu podataka.

Nakon što su poslati podaci i za prognoziranu i za ostvarenu potrošnju vrše se proračuni odstupanja. Proračuni se vrše samo za

objekte za koje su pristigli podaci i o prognoziranoj i o proračunatoj potrošnji. Koji će se proračun izvršiti zavisi od podešavanja u App.config servisnog dela aplikacije. Zatim se, nakon uspešnog proračuna, baza podataka ažurira proračunatim vrednostima.

Baza podataka:

Baza podataka je implementirana kao XML ili In-Memory baza što zavisi od odluke korisnika.

XML baza podataka:

Svaka tabela u bazi je predstavljena odgovarajućom XML datotekom. Ako datoteka ne postoji, automatski se kreira. U suprotnom se datoteka samo ažurira podacima. Postoje 3 tabele, za Load objekte, za Audit objekte i za ImportedFile objekte. Primeri zapisa sve 3 tabele se nalaze na slikama ispod:

```
<row>
  <TIME_STAMP>2023-01-17T00:00:00</TIME_STAMP>
  <FORECAST_VALUE>6426.01794633789</FORECAST_VALUE>
  <MEASURED_VALUE>6234</MEASURED_VALUE>
  <ABSOLUTE_PERCENTAGE_DEVIATION>N/A</ABSOLUTE_PERCENTAGE_DEVIATION>
  <SQUARED_DEVIATION>0.000948746190718455</SQUARED_DEVIATION>
  <IMPORTED_FILE_ID>122</IMPORTED_FILE_ID>
</row>
```

Primer zapisa u datoteci za Load objekte

```
<row>
  <ID>100</ID>
  <TIME_STAMP>2023-09-01T14:16:09.9028407+02:00</TIME_STAMP>
  <MESSAGE_TYPE>Info</MESSAGE_TYPE>
  <MESSAGE>Datoteka forecast_2023_01_17.csv je uspesno procitana</MESSAGE>
</row>
```

Primer zapisa u datoteci za Audit objekte

```
<row>
  <ID>100</ID>
  <FILE_NAME>forecast_2023_01_17.csv</FILE_NAME>
</row>
```

Primer zapisa u datoteci za ImportedFile objekte

In-Memory baza podataka:

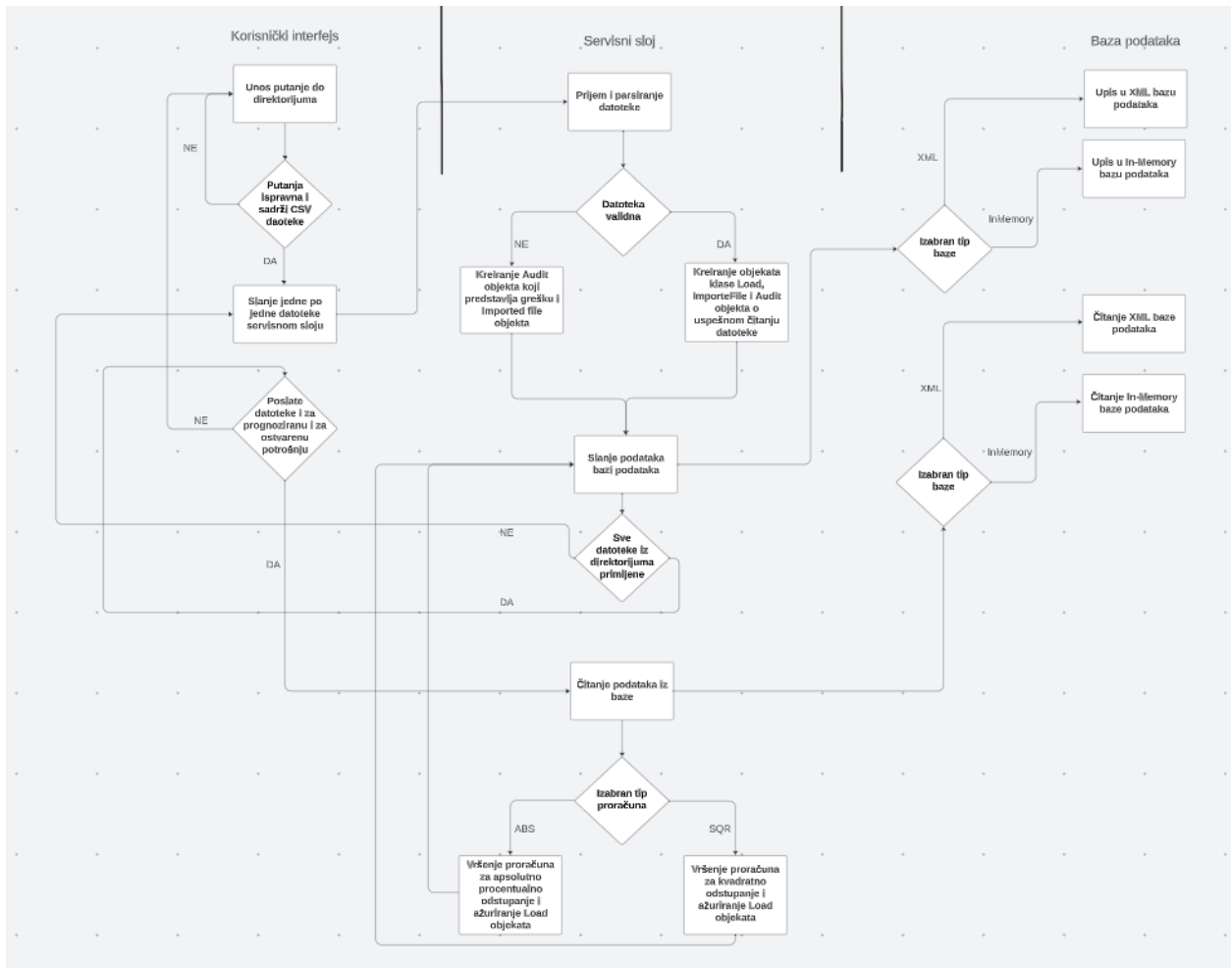
Implementirana je pomoću Dictionary strukture podataka. Svaka tabela je predstavljena odgovarajućim Dictionary-em, gde je ključ ID reda u tabeli, a vrednost je objekat odgovarajuće klase (Load, ImportedFile i Audit). Postoje tri Dictionary-ja, po jedan za svaku klasu objekata. Podaci u In-Memory bazi postoje samo dok je servis pokrenut.

Odluka o izboru baze podataka:

Odluka da li će podaci biti upisani u XML bazu podataka ili In-Memory bazu podataka donosi se na osnovu podešavanja korisnika u App.config datoteci servisnog dela aplikacije.

```
<appSettings>
  <!--izbor da li zelimo da upisemo u inMemory ili xml bazu podataka-->
  <add key="tipBaze" value="xml"/>
</appSettings>
```


Dijagram arhitekture:



Opis interfejsa

Interfejs koji se koristi za komunikaciju između klijentske i serverske strane je IServis interfejs koji sadrži sledeće metode:

- prijemDatoteke – metoda koja prihvata CSV datoteku od klijenta čita podatke iz nje i zapisuje potrebne informacije u bazu podataka kao što su Load, Audit, ImportedFile objekti
- sviPodaciUcitani – metoda koja se poziva nakon što su prosleđeni i podaci o prognoziranoj i o ostvarenoj potrošnji i vrši odgovarajuće proračune odstupanja

```
namespace Server
{
    [ServiceContract]
    public interface IServis
    {
        [OperationContract]
        void prijemDatoteke(MemoryStream datoteka, string nazivDatoteke);

        [OperationContract]
        void sviPodaciUcitani();
    }
}
```

Interfejs koji se koristi za komunikaciju između serverske strane i baze podataka je IBazaPodataka interfejs koji sadrži sledeće metode:

- UpisUXmlBazu - metoda koja se poziva nakon što je parsiranje datoteke završeno i nakon što su kreirani svi potrebni objekti za upis u bazu podataka i ako je u App.config podešavanjima

servisnog dela aplikacije izabrano korišćenje XML baze od strane korisnika

- UpisUInMemorzBazu - metoda koja se poziva nakon što je parsiranje datoteke završeno i nakon što su kreirani svi potrebni objekti za upis u bazu podataka i ako je u App.config podešavanjima servisnog dela aplikacije izabrano korišćenje In-Memory baze od strane korisnika
- CitanjeXmlBaze – metoda koja se poziva da bi se pročitali svi potrebni podaci o ostvarenoj i prognoziranoj potrošnji radi vršenja proračuna odstupanja i ako je u App.config podešavanjima servisnog dela aplikacije izabrano korišćenje XML baze od strane korisnika
- CitanjeInMemoryBaze – metoda koja se poziva da bi se pročitali svi potrebni podaci o ostvarenoj i prognoziranoj potrošnji radi vršenja proračuna odstupanja i ako je u App.config podešavanjima servisnog dela aplikacije izabrano korišćenje In-Memory baze od strane korisnika

```
public interface IBazaPodataka
{
    [OperationContract]
    3 references
    void UpisUXmlBazu(List<Load> loadList, Audit audit, string nazivDatoteke);

    [OperationContract]
    3 references
    void UpisUInMemoryBazu(List<Load> loadList, Audit audit, string nazivDatoteke);

    [OperationContract]
    2 references
    void CitanjeXmlBaze(out List<Load> loadList);

    [OperationContract]
    2 references
    void CitanjeInMemoryBaze(out List<Load> loadList);
}
```

Opis korišćenih tehnologija

Projekat je razvijen koristeći sledeće tehnologije:

- Programski jezik: C#
- Razvojno okruženje: .NET Framework
- Komunikacija između klijentske, serverske aplikacije i baze podataka: Windows Communication Foundation (WCF) protokol
- Rad sa datotekama: Implementiran je tako da se efikasno upravlja memorijom, uključujući korišćenje Dispose paterna.

Zaključak

Aplikacija omogućava precizno praćenje potrošnje električne energije i proračun odstupanja između prognozirane i ostvarene potrošnje kao i upis tih vrednosti u bazu podataka radi mogućnosti kasnijeg pregledanja podataka. Ova funkcionalnost će značajno olakšati rad, na primer, kompanije za prenos električne energije i unaprediti efikasnost u upravljanju potrošnjom električne energije.

Budući pravci istraživanja:

- Dalje istraživanje može se usmeriti na optimizaciju performansi aplikacije kako bi se smanjilo vreme obrade i proračuna odstupanja.
- Ispitivanje mogućnosti integracije sa drugim sistemima za dalje poboljšanje razmene podataka.
- Razvijanje grafičkog korisničkog interfejsa sa grafikonima i drugim načinima vizualizacije za lakše praćenje potrošnje.
- Automatizacija procesa uvoza podataka kako bi se smanjila potrebna ljudska interakcija.
- Ispitivanje primene složenijih algoritama za proračun odstupanja i njihovih prednosti u odnosu na trenutne metode.
- Razmatranje mogućnosti za uvoz i obradu različitih formata podataka osim CSV datoteka.