Zadatak 1

Evidencija prognozirane i ostvarene potrošnje električne energije

Klijent je kompanija za prenos električne energije. Aplikacije treba da se bavi evidencijom prognozirane i ostvarene potrošnje električne energije.

Slede korisnički zahtevi:

1. **Uvoz podataka o prognoziranoj i ostvarenoj potrošnji**

Aplikacijom se uvoze podaci o planiranoj i ostvarenoj potrošnji električne energije. Uvoz se vrši iz CSV datoteka. Datoteke se importuju izborom opcije korisničkog interfejsa – u konzolnu aplikaciju se unosi putanja sa koje se čitaju sve csv datoteke. Naziv datoteke se sastoji od tipa datoteke i datuma. Između ova dva podatka nalazi se donja crta. Tip datoteke može biti „prog“ za prognoziranu potrošnju i „ostv“ za ostvarenu potrošnju. Datum je u obliku yyyy_mm_dd_hh . Primeri naziva Datoteka su forecast_2023_01_27.csv i measured_2023_01_27.csv. Prva datoteka sadrži podatke o prognoziranoj potrošnji, a druga datoteka o ostvarenoj potrošnji, za dan 27. januar 2023. godine.

Podaci u svakom redu su sat na koji se potrošnja odnosi i iznos potrošnje u mW/h. U svakoj datoteci može biti onoliko redova koliko ima sati u danu (24, 23 ili 25). Ako u nekoj datoteci broj sati ne odgovara broju sati u danu za koji se vrši unos, cela datoteka se odbacuje kao nevalidna. Ako se datoteka odbacuje kao nevalidna, kreira se novi *Audit* objekat koji se upisuje u bazu podataka. Podatak koji je potrebno evidentirati u tekstu poruke u audit tabeli za nevalidnu datoteku je tekst greške.

Za svaki sat iz CSV datoteka kreira se po jedan objekat klase *Load*, 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. Jedan objekat klase *Load* predstavlja podatke o prognoziranoj i ostvarenoj potrošnji električne energije za jedan sat. Kreirani objekti se upisuju u bazu podataka. Posebno se prate podaci o prognoziranoj i ostvarenoj potrošnji.

Za svaku obrađenu CSV datoteku kreira se objekat *ImportedFile* i upisuje se u bazu podataka.

2. **Proračun odstupanja između prognozirane i ostvarene potrošnje**

Nakon što su učitani podaci upisani u bazu podataka, servis pristupa proračunu odstupanja između prognozirane i ostvarene potrošnje po satu. Izračunavanje se vrši samo za one objekte za koje su pristigli podaci i o prognoziranoj i ostvarenoj potrošnji. Odstupanje može da se izračunava kao apsolutno procentualno odstupanje ili kvadratno odstupanje.

Apsolutno procentualno odstupanje se izračunava po formuli:

$$\frac{|ostvarena\ potrošnja - prognozirana\ potrošnja|}{ostvarena\ potrošnja} x100$$

Kvadratno odstupanje se izračunava po formuli:

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

Odluka o tome da li će se koristiti apsolutno procentualno odstupanje ili kvadratno odstupanje donosi se na osnovu podešavanja u App.config datoteci servisnog dela aplikacije.

Odstupanje po satu upisuje se u bazu podataka, za svaki red pojedinačno, u posebnom polju.

Kada se završi proračun odstupanja i za posledji red, aktivira se događaj ažuriranja baze podataka proračunatim podacima.

3. Model podataka

Model podataka obuhvata sledeće klase:

- **Load** (polja: Id, Timestamp, ForecastValue, MeasuredValue, AbsolutePercentageDeviation, SquaredDeviation, ForecastFileId, MeasuredFileId)
- **ImportedFile** (polja: Id, FileName)
- **Audit** (Id, Timestamp, MessageType, Message)
  MessageType može da ima vrednosti Info, Warning i Error

4. Implementacija baze podataka

Baza podataka treba da bude implementirana kao XML baza podataka i kao In-Memory baza podataka.
XML baza podataka sadrži XML datoteke u koje se upisuju podaci. Svaka tabela je implementirana kroz jednu XML datoteku. Ukoliko XML datoteka ne postoji, potrebno je da bude kreirana automatski. Primeri XML tabela nalaze se u prilogu.

In-Memory baza podataka implementirana je kroz Dictionary ili ConcurrentDictionary strukture podataka. Svaka tabela je implementirana kroz jedan (Concurrent)Dictionary, pri čemu je Key ID reda u tabeli, a Value je objekat odgovarajuće klase (*Load*, *ImportedFile* i *Audit*). Podaci u In-Memory bazi podataka postoje samo dok je servis pokrenut.

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

5. Tehnički i implementacioni zahtevi

- Aplikacija treba da bude u višeslojnoj arhitekturi. Aplikacija treba da sadrži najmanje sledeće komponente:
  - baza podataka (XML baza podataka i In-Memory baza podataka)
  - servisni sloj
  - korisnički interfejs – konzolna aplikacija
  - Common – projekat koji je zajednički za sve slojeve

Komunikacija između klijentske aplikacije i servisa obavlja se putem WCF-a

- Rad sa datotekama treba da bude implementiran tako da se vodi računa o održavanju memorije, korišćenjem Dispose paterna
- Celokupna poslovna logika obavlja se na servisnoj strani aplikacije. Parsiranje datoteka se takođe obavlja na servisnoj strani. Slanje datoteka sa klijentske ka serverskoj aplikaciji obavlja se korišćenjem MemoryStream-a.
- Aktiviranje događaja ažuriranja baze podataka proračunatim podacima izvršava se korićenjem Event-a i Delegate-a. Delegat treba da pokazuje na odgovarajući metod – upis u XML bazu podataka ili u In-Memory bazu podataka
- Za aplikaciju treba da postoje sledeći dokumenti:
  - User manual
  - Dokument u kom je opisana arhitektura aplikacije