Zadatak 3

Čitanje i keširanje podataka o potrošnji

Svrha aplikacije je čitanje podataka o prognoziranoj i ostvarenoj potrošnji električne energije, uz uvažavanje zahteva performansi. Pošto je čitanje podataka iz eksternih skladišta (kao što su RDBMS ili File System) zahtevno sa stanovišta resura, ova aplikacija implementira istovremeno XML bazu podataka i In-Memory bazu podataka. Jednom pročitan podatak iz XML-a upisuje se u In-Memory bazu podataka, gde ostaje određeno vreme. Podaci se čitaju iz In-Memory baze podataka dok se nalaze u njoj, što poboljšava performanse čitanja.

1. Poslovna logika

Aplikacija se sastoji od servisa i od klijentske aplikacije. Servis i klijentska aplikacije komuniciraju putem WCF-a. Klijentska aplikacija je konzolna aplikacija koja ima opciju unosa datuma za koji se očekuju podaci.

Servis dobija od klijenta upit za čitanje podataka iz baze podataka, na osnovu prosleđenog datuma. Upit treba da vrati rezultat koji sadrži *Load* objekte za svaki sat za datum iz upita, ukoliko postoje u bazi podataka.

Servis pristupa bazi podataka koja se nalazi u dva oblika: XML datoteka i In-Memory baza podataka. Kada se servis pokrene, In-Memory baza podataka je inicijalno prazna.

Servis prvo pokušava čitanje iz In-Memory baze podataka. Ukoliko podaci postoje u In-Memory bazi podataka, ovi podaci se čitaju i prosleđuju se klijentskoj aplikaciji. Ukoliko podaci ne postoje u In-Memory bazi podataka, servis pokušava čitanje iz XML baze podataka. Ukoliko podaci ne postoje za prosleđeni datum ni u XML bazi podataka, kreira se novi Audit objekat sa odgovarajućom porukom, koji se upisuje u In-Memory bazu podataka i u XML bazu podataka. Ovaj Audit objekat se prosleđuje u vidu rezultata klijentskoj aplikaciji i klijentska aplikacija ispisuje tekst poruke na konzoli. Ako podaci postoje u XML bazi podataka, na osnovu tih podataka kreiraju se odgovarajući *Load* objekti. Jedan objekat klase *Load* predstavlja podatke o prognoziranoj i ostvarenoj potrošnji električne energije za jedan sat. Kreirani Load objekti upisuju se u In-Memory bazu podataka. Sledeći put kada se pojavi odgovarajući upit, ovi podaci će biti pročitani iz In-Memory baze poadataka.

Load objekti dobijeni na osnovu upita prosleđuju se kao rezultat klijentskoj aplikaciji.

Podaci se brišu iz In-Memory baze podataka kada prođe definisano *DataTimeout* vreme. Ovo vreme se definiše kao odgovarajući broj minuta u App.config datoteci servisne aplikacije. Podrazumevani broj *DataTimeout* minuta je 15.

Kada je klijentska aplikacija primila rezultate sa Load objektima, na osnovu njih kreiraju se CSV datoteke koje se upisuju na lokaciju definisanu u konfiguracionoj datoteci klijentske aplikacije App.config. Takođe, na konzoli klijentske aplikacije ispisuje se poruka o kreiranoj datoteci. Ova poruka sadrži i podatke o putanji i imenu datoteke.

2. Model podataka

Model podataka obuhvata sledeće klase:

- **Load** (polja: Id, Timestamp, ForecastValue, MeasuredValue)
- **Audit** (Id, Timestamp, MessageType, Message)

3. 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.

XML baza podataka već postoji i nalazi se u prilogu (TBL_LOAD.xml).

In-Memory baza podataka implementirana je kroz Dictionary ili ConcurrentDictionary strukture podataka. Svaka tabela je implementirana kroz jedan 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.

4. Tehnički i implementacioni zahtevi

1. 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

2. Rad sa datotekama treba da bude implementiran tako da se vodi računa o održavanju memorije, korišćenjem Dispose paterna
3. Aktiviranje događaja brisanja zastarelih podataka iz In-Memory baze podataka izvršava se korićenjem Event-a i Delegate-a. Delegat treba da pokazuje na odgovarajući metod brisanja podataka. Za ovu funkcionalnost **ne** koristiti ugrađene *Timer* objekte, već treba kreirati poseban thread koji će vršiti proveru da li postoje zastareli podaci i koji će nakon isteka *DataTimeout* vremena okidati odgovarajući Event.
4. Za aplikaciju treba da postoje sledeći dokumenti:
   - User manual
   - Dokument u kom je opisana arhitektura aplikacije