Dokumentacija projekta
"Čitanje i keširanje podataka o potrošnji"

Predmet: Virtuelizacija procesa

Profesor: Bojan Jelačić

Naziv zadatka: Čitanje i keširanje podataka o potrošnji

Autor projektnog zadatka: Milica Klincov PR70/2020

Sadržaj

1. Opis projektnog zadatka

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

Server treba da vrati rezultat koji sadrži Load objekte za svaki sat za prosljeđen datum iz upita.

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.

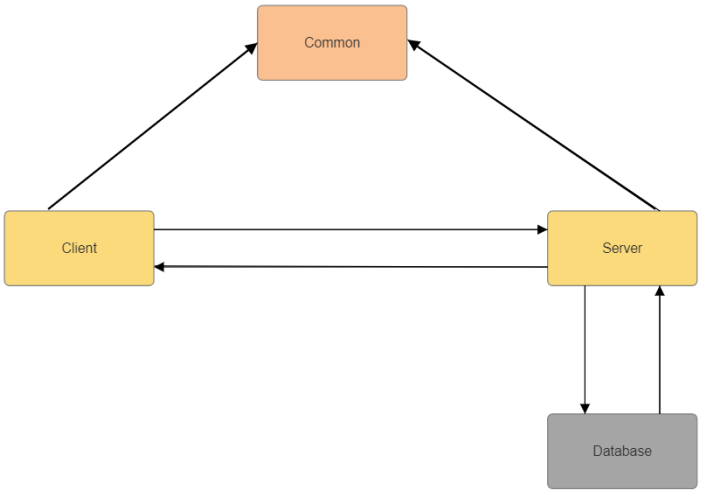
Servis prvo pokušava čitanje iz In-Memory baze podataka. Ukoliko podaci postoje u In-Memory bazi podataka, ovi podaci se čitaju i prosleđuju se klijentskoj aplikaciji. Ukoliko podaci ne postoje u In-Memory bazi podataka, servis pokušava čitanje iz XML baze

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 prosljeđuje 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 vrijeme. Ovo vrijeme se definiše kao odgovarajući broj minuta u App.config datoteci servisne aplikacije. Podrazumijevani 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. Arhitektura projekta



Projekat se temelji na višeslojnoj arhitekturi koja obuhvata:

Service (konzolna aplikacija) – aplikacija koja sadrži poslovnu logiku projekta. U Servisu se obrađuje zahtjev klijenta, komunikacija sa bazom i slanje odgovora klijentu

Client (konzolna aplikacija) – korisnički interfejs implementiran kao konzolna aplikacija koja omogućava korisniku komunikaciju sa serverom

Common (Class library) – zajednički sloj koji sadrži kod koji je djeljen između slojeva aplikacije. U njemu su smještene zajedničke klase i resursi koji se koriste

Baza podataka (XML i In-Memory) - XML baza podataka koristi se za čuvanje podataka u obliku XML datoteka. Svaka tabela podataka implementirana je kroz odvojenu XML datoteku.

In-Memory baza podataka implementirana je kroz strukturu podataka Dictionary. Svaka tabela podataka predstavljena je kao Dictionary, gde je ključ reda u tabeli, a vrijednost je objekat odgovarajuće klase (Load, Audit).

3. Opis interfejsa

ILoadService – ovaj interfejs definiše kontrakt za servis za dobavljanje podataka. Sadrži metodu GetLoads(DateTime dateTime) koja prima datum i vrijeme kao argument i vraća rezultat koji sadrži informacije za dati datum i vrijeme.

ICheckFile – ovaj interfejs služi za provjeru da li je određen fajl u upotrebi . Sadrži metodu IsFileInUse(string filePath) koja prima putanju do fajla kao argument i vraca true ako je fajl trenutno u upotrebi.

4. Korišćene tehnologije

Projekat je razvijen koristeći sledeće tehnologije:

C# programski jezik za razvoj aplikacije.

Windows Communication Foundation (WCF) za komunikaciju između klijenta i servisa. Windows Communication Foundation (WCF) je tehnologija koja omogućava razvoj i implementaciju distribuiranih aplikacija koje komuniciraju preko mreže.

5. Zaključak

Projekat "Čitanje i keširanje podataka o potrošnji" je uspješno razvijen i omogućava efikasno čitanje i keširanje podataka o potrošnji električne energije. Mogući pravci budućeg istraživanja i proširenja zadatka uključuju dodatnu optimizaciju performansi, podršku za različite izvore podataka i proširenje korisničkog interfejsa za veću funkcionalnost. Proširenje zadatka bi moglo da se ogleda u tome da

korisnik ima mogućnost unosa nekog drugog kriterijuma osim datuma.