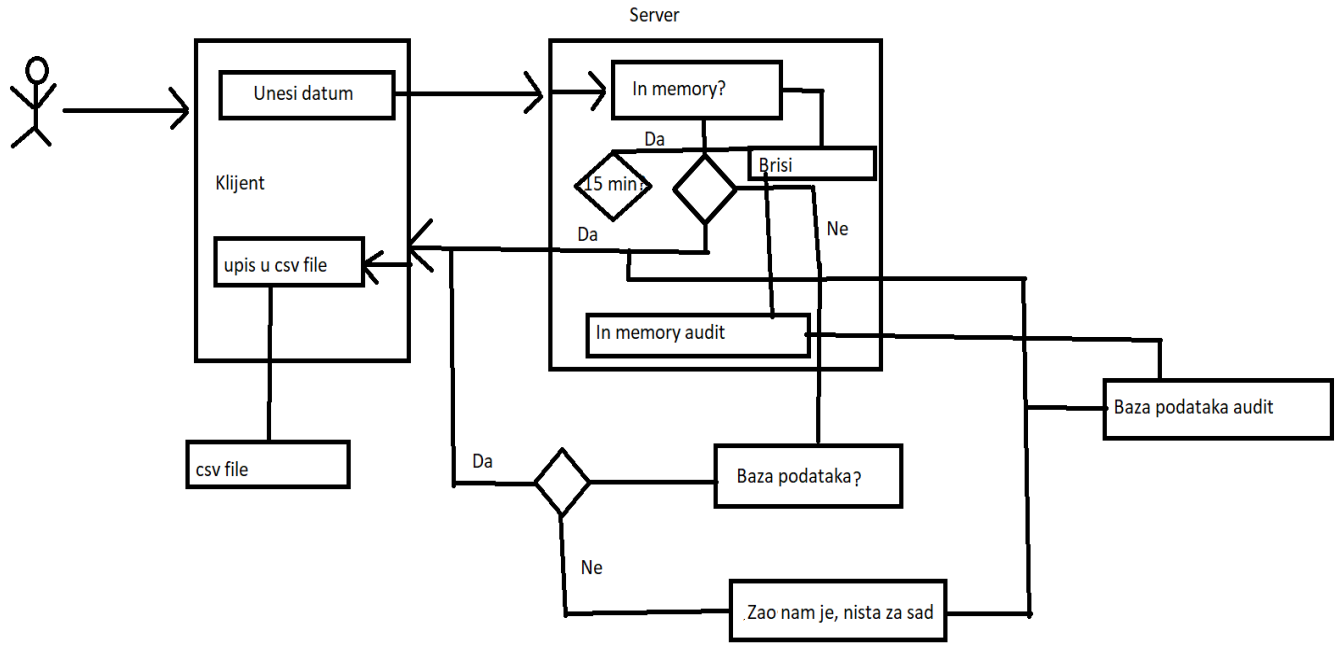
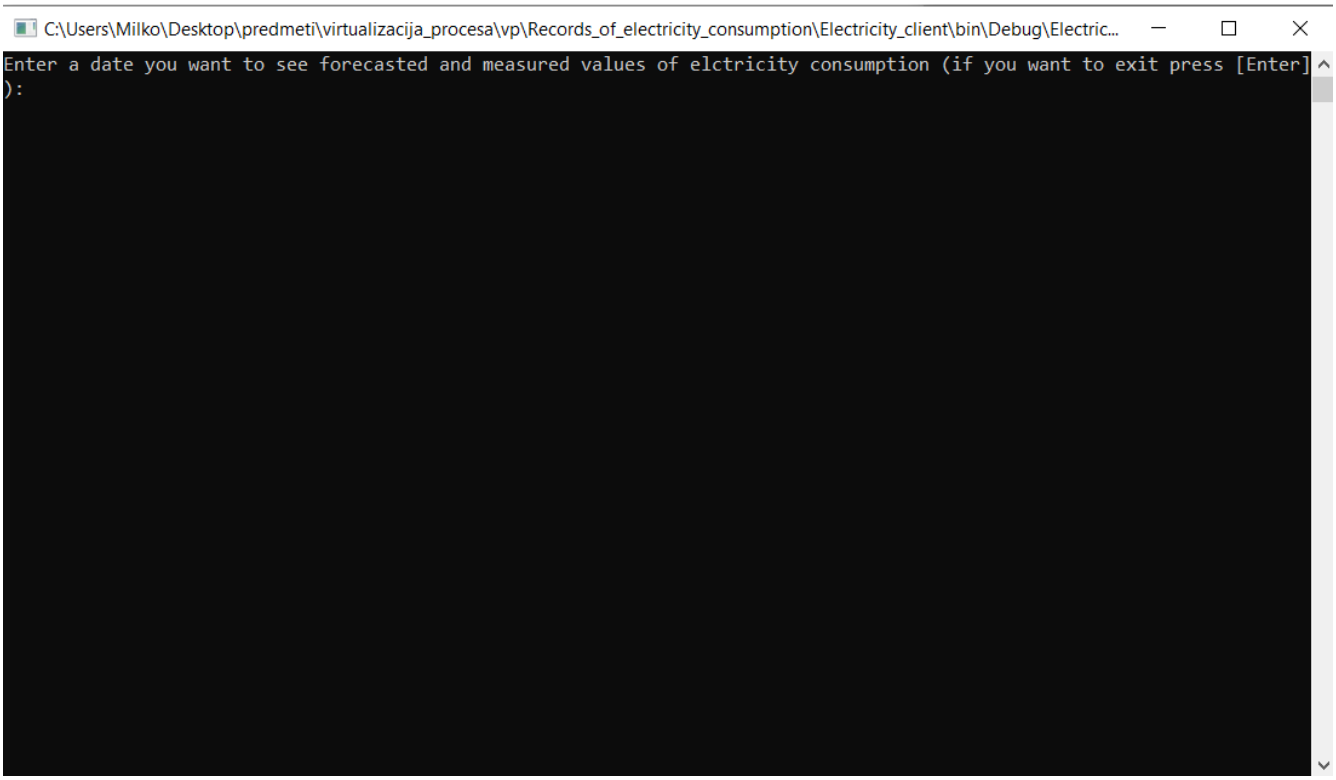
Records of electricity consumption

U projektnom zadatku se trazilo napravit aplikaciju sa klijent server arhitekturom, u kom klijent I server komuniciraju preko WPF-a. Klijent salje poruku serveru o datumu za koji zeli vidit izmjerene i prognozirane vrijednosti o potrosnji elektricne energije, a server mu salje te vrijednosti za svaki sat u tom datumu. Klijent to onda upisuje u csv datoteku definisanu u App.config fajlu. Server sve trazene podatke upisuje u In memory bazu podataka, ako oni postoje u xml bazi podataka. Ako ne, kreira se audit objekat sa porukom o neuspjelom citanju iz baze podataka I salje se klijentu. Takodje se kreira I xml fajl ako ne postoji za taj audit objekat. Svi objekti u in memory bazi podataka se brisu nakon 15 minuta od njihovog unosa u In memory bazu podataka, uz pomoc event-a I delegate-a koristeci thread klasu I njene metode. Kada load objekat koji sadrzi podatak za izmjerene I prognozirane vrijednosti elektricne energije ne postoji nigdje u bazi podataka, onda se kreira nova tabela u toj bazi I u njoj server ispisuje izmjerene I prognozirane vrijednosti za svaki sat za taj datum I te podatke istovremeno upisuje u In memory bazu podataka.



Arhitektura I dijagram toka podataka u istoj slici



Interfejs klijenta, u kome klijent unosi datum za koji je zainteresovan. Unosi ga u formaatu yyyy-MM-dd, I ako ga ne unese pravilno ili unese nepostojeci datum, prilika za unos datuma mu se ponovo pruza dok unos datuma ne uradi na pravilan nacin. Kada unese pravilno datum, server mu salje informaciju o tom datumu, I klijent ponovo moze unijet datum ako hoce, a ako nece moze pritisnut Enter dugme I izac iz aplikacije. To moze uradit I u samom pocetku ako pozeli.

Wcf je koriscen za komunikaciju izmedju kiljenta I servera, sto se tice tehnologije u kojoj je to napravljeno. Sve ostale funkcionalnosti su napravljene uz pomoc ugradjenih klasa u C#. Dakle brisanje iz in memory baze podataka je napravljeno uz pomoc event-a I delegate-a I Thread-a. Pravljenje xml fajlova je realizovano uz pomoc XmlWriter klase, a koristen je I XmlDocument kad nesto treba ucitat iz xml fajla. Klasa Directory je koriscena za pretragu u direktorijumu. CsvWriter je koriscen za upis podataka u csv file, zajedno sa StreamWriter klasom. StreamReader je koriscen za ucitavanje podataka iz csv datoteke cisto radi toga da se odredjeni podaci ne bi upisivali vise puta ako vec postoje u csv fajlu. In memory baza podataka je realizovana uz pomoc Dictionary<DateTime,Audit> za audit objekte, I Dictionary<DateTime,List<Load>> za load objekte. List<Load> zato se drukcije nije moglo, Load objekte nije bilo moguce upisat u Dictionary<DateTime,Load> zbog nacina na koji se oni upisuju: kljuc je DateTime.Now.AddMinutes(-30) I uzastopno dodavanje Load objekata je uvijek rezultovalo u izuzetku.

Zadatak je uradjen tako da je moguce prosirit projekat uz dodavanje Replicator komponente, tako da sve ono sto se upise u In Memory bazu podataka u jednom serveru se replicira I na drugi, a moguce je napravit autentifikaciju I autorizaciju tako da samo odredjeni korisnici mogu koristit ovaj app, a ti korisnici dalje mogu imat odredjena prava citanja I upisa podataka, s tim da upis podataka u ovom projektu nema mnogo smisla ali bi se korisniku mogla pruzit mogucnost da sam upisuje mjerenja I prognoze potrosnje elektricne energije ako zeli. Mogla bi se aplikacija izmjenit tako da korisnik umjesto konzolne alikacije ima neki graficki interfejs kroz koji moze komunicirat sa serverom, napravljen uz pomoc WPF-a ili Windows Forms-a.