|  |  |
| --- | --- |
| **Naziv projekta**: Projekat 3. Razvoj elektroenergetskog softvera | |
| **Datum početka**: 13/05/2019 | **Datum završetka**: 04/06/2019 |
| **Fakultet**: Fakultet tehničkih nauka Novi Sad | |
| **Mentor**: Stefan Dejanović | |
| **Članovi tima**: Nikola Drčelić i Marko Tatić | |
| **Postavka zadatka:**  Worker    Reader  Worker    Load Balancer  Writer  Worker  Worker  Projektovati C# aplikaciju koja se sastoji od četri komponente [ Writer, Reader, Load Balancer, Worker] povezanu sa SQL bazom podataka .  Za arhitekturu projekta izabran je Client-Server model, gde    imamo dve vrste nezavisnih klijenata (Writer i Reader), a  serversku stranu čine LoadBalancer, Workeri i DataBase gde  je za medjusobnu vezu korišćen WCF.  Klijentske strane su implementirane kao konzolne aplikacije,  kao i serverska strana radi lakšeg praćenja toka simulacije.  Pre same implementacije rešenja, nacrtani su sledeći dijagrami: Class dijagram, Activity dijagram, Sequence dijagram i Commponent dijagram. Na osnovu postavke zadatka ali i test scenaria formirane su osnovne komponente kao i njihove medjusobne veze.  **Component dijagram:**    **Sequence dijagram:**    Writer komponenta nakon pokretanja i na zahtev klijenta otvara konekciju sa LoadBalancer komponentom. Nakon otvaranja konekcije, u zasebnom Tasku, se odvija slanje Random podataka Code i Value sa Writer strane na LoadBalancer. Nakon otvranja konekcije na Writer strani, pojavljuje se novi meni koji omogućava klijentu da pokrene nove Workere pritiskom na taster (+), da ugasi postojeće Workere pritiskom na taster (-), dobije pregled trenutno aktivnih workera (.) kao i da napusti meni i završi izvršavanje aplikacije (x).  Ukoliko korisnik(Writer) pokrene novog Workera, obaveštava se komponenta LoadBalancer koja dodaje novog Workera u listu aktivnih Workera. LoadBalancer od ovog trenutka provera da li je lista aktivnih workera prazna, u tom slučaju ona nastavlja da skladišti lokalno podatke koji pristižu svake dve sekunde, ako u listi Workera ima bar jedan aktivan, LoadBalancer mu prosledjuje podatke na obradu. U slučaju da ima više od jednog aktivnog workera, LoadBalancer komponenta rasporedjuje posao po Round Robin principu na sve Workere.  Komponenta Worker nakon dodeljivanja zadatka od strane LoadBalancera, prepakuje primljene podatke u lokalnu strukturu, a zatim vrši obradu podatka u odnosu na DataSet. Nakon provere vrednosti podatka, odnosno provere DeadBanda(manji od 2%) vrši se upis vrednosti u bazu podataka. Nakon upisa u bazu, na serverskoj strani se vrši ispis o uspešnosti ove aktivnosti.  Reader komponenta omogućava klijentu pregled vrednosti koje su do sada upisane u Bazu podataka. Klijentu je na zahtev neophodno isporučiti poslednje vrednosti upisane u bazu podataka za zadati CODE. Takodje klijent ima mogućnost da mu se isporuče i sve vrednosti pristigle i sačuvane u bazi podataka za odredjeni CODE u odredjenom vremenskom intervalu.  Kada je reč o bazi podataka, u implementaciji rešenja je korišćena SQLite baza. Ono što karakteriše ovu bazu je pre svega mogućnost kreiranja .dll file kao i .db file u okviru samog projekta, te povezati ConnectionString u okviru AppConfiga tako da nije potrebno menjati adresu ukoliko se promeni fizička mašina, odnosno navodi se relativna putanja u okviru projekta.  **SQLite baza podataka sa 4 tabele**: | |