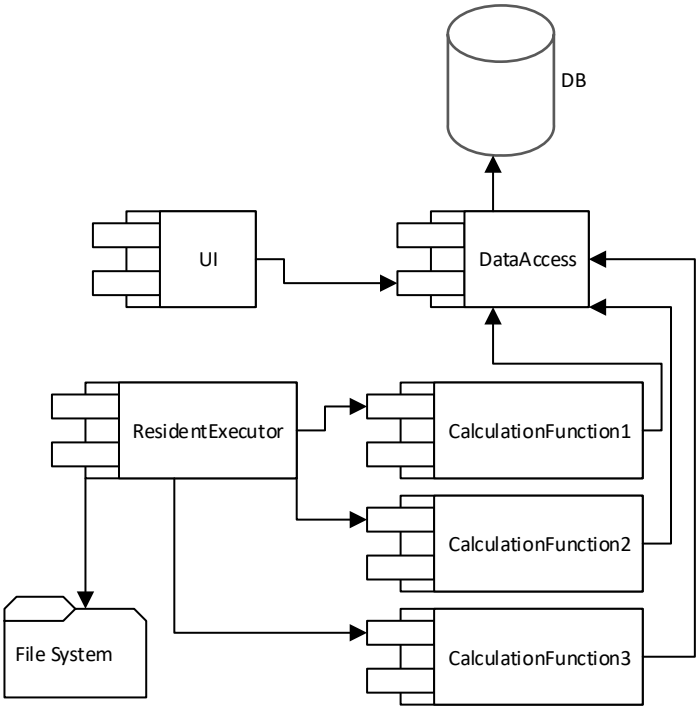
Projekat rezidentnog izvršavanja funkcija proračuna

U aplikacijama koje se bave matematičkim proračunima često je potrebno da se proračuni vrše automatski, u određenim vremenskim intervalima. U ovom zadatku potrebno je kreirati aplikativni sistem koji će omogućavati kako ručno, tako i automatsko / rezidentno izvršavanje funkcije proračuna. Rešenje treba da uključi komponentu *Resident Executor*, koja služi za rezidentno izvršavanje funkcija nezavisno od implementacije. Takođe je potrebno kreirati i tri konkretne implementacije rezidentne funkcije koje će biti pozivane, a odnosiće se na proračune u vezi sa potrošnjom električne energije za određeno geografsko područje.

Implementaciono rešenje treba da sadrži module kao što sledi:



DB – baza podataka
DataAccess – modul pristupa bazi podataka. Služi za perzistenciju podataka.
ResidentExecutor – modul automatskog/rezidentnog izvršavanja funkcija
CalculationFunction1, CalculationFunction2, CalculationFunction3 – konkretne funkcije proračuna koje se pozivaju kroz ResidentExecutor
UI – Korisnički interfejs

Scenario rada aplikacije

Kroz UI upisuju se satni podaci o potrošnji električne energije za jedno geografsko područje. Upisuju se sledeći podaci:

- Timestamp (datum plus vreme)
- Potrošnja u mW/h

Komponenta CalculationFunction1 izračunava prosečnu potrošnju tekućeg dana i upisuje je u bazu podataka, sa podacima o vremenu proračuna, poslednjem vremenu merenja i vrednošću prosečne potrošnje. Ako je proračun sa poslednjim vremenom merenja već urađen, proračun se ne izvršava.

Komponenta CalculationFunction2 izračunava minimalnu potrošnju tekućeg dana i upisuje je u bazu podataka, sa podacima o vremenu proračuna, poslednjem vremenu merenja i vrednošću minimalne potrošnje. Ako je proračun sa poslednjim vremenom merenja već urađen, proračun se ne izvršava.

Komponenta CalculationFunction3 izračunava maksimalnu potrošnju tekućeg dana i upisuje je u bazu podataka, sa podacima o vremenu proračuna, poslednjem vremenu merenja i vrednošću maksimalne potrošnje. Ako je proračun sa poslednjim vremenom merenja već urađen, proračun se ne izvršava.

Komponente CalculationFunction1, CalculationFunction2 i CalculationFunction3 imaju svoja jedinstvena obeležja.

Komponenta ResidentExecutor čita iz XML datoteke identifikaciona obeležja funkcija koje treba da se izvršavaju. Funkcije se smeštaju u red za izvršavanje i vrši se njihovo izvršavanje na svakih 10 sekundi. ResidentExecutor ne treba da bude svestan logike proračuna ni za jednu od funkcija.

U UI komponenti vrši se čitanje rezultata izvršenih funkcija.

Evidencija geografskih područja

Geografsko područje sadrži ime geografskog područja i šifru geografskog područja (skraćeno ime). Šifra geografskog područja se nalazi u CSV fajlovima.

Kroz korisnički interfejs nije potrebno da se vrši evidentiranje geografskih područja. Geografska područja treba da postoje u bazi podataka.

Tehnički i implementacioni zahtevi

1. U dizajnu i arhitekturi aplikacije potrebno je definisati moguće use case-ove, klase, aktivnosti objekata klasa, interakciju između objekata klasa i sofverske komponente aplikacije.
2. Aplikacija treba da bude u multy-component arhitekturi. Aplikacija treba da sadrži najmanje sledeće komponente:
   - baza podataka
   - servisni sloj (opciono može da bude razdvojen na sloj pristupa bazi podataka i sloj poslovne logike).
   - korisnički interfejs (konzolna, Web ili desktop aplikacija)

   Slojevi mogu da komuniciraju direktno, odnosno servisni sloj ne mora da egzistira na aplikativnom serveru.

   Baza podataka može da bude implementirana kroz neki od SUBP (MS SQL Server, Oracle), kroz neki od embeded sistema za baze podataka (SQLite, MS Access) ili kroz XML.

3. Servisni sloj treba da bude pokriven unit testovima. Pokrivenost unit testova treba da bude najmanje 60%
4. Aplikacija treba da bude razvijana poštujući Agile/Scrum metodologiju razvoja, korišćenjem TFS-a

Kriterijum ocenjivanja

1. Dizajn I arhitektura rešenja
2. Korišćenje Scrum metodologije razvoja – definisanje User Story-a i taskova, planiranje i estimacija
3. Implementacija rešenja
4. CI ciklus
   a. Build
   b. UnitTestovi
   c. Pokrivenost koda testovima