REES Projektni zadatak – Back End Dokumentacija

Tema: DataCache (gr. 4.)

Asistent:

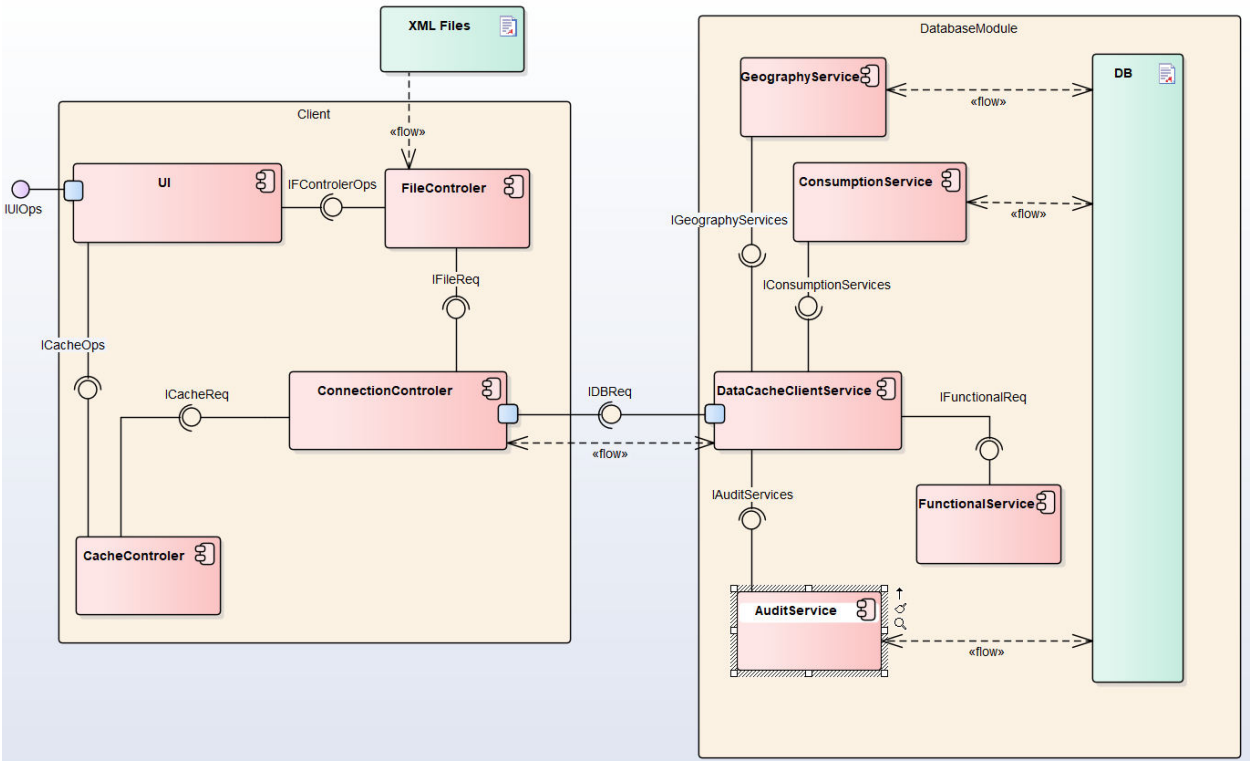
Student: **Asistent-Master**

Todorović Uglješa **Janković Zoran**

Novi Sad, juni 2021.

SADRŽAJ

1.Component Diagram



Sistem se sastoji od dvije cjeline:

- **Client** (klijentski agent za komunikaciju sa distribuiranom bazom podataka)
- **DatabaseModule** (serverska aplikacija za prihvat klijentskih zahjeva za komunikaciju sa bazom podataka).

1.1. Client

Klijentski modul se sastoji od 4 glavne logičke cjelnine :

- UI (komponenta sa kojom korisnik komunicira – prihvata korisničke zahtjeve posredstvom *IUIOps* interface – a i distribuira ih dublje u arhitekturu). Praktično realizovana kao DLL projekat. Na ovu komponentu zahtjevi stižu sa GUI_Integrator WPF aplikacije.

- FileControler (komponenta koja izvršava zahtjeve od strane UI komponente pristigle posredstvom *IFControlerOps* interface - a). Zadatak ove komponente jeste da iz file sistema učita ciljanu datoteku, isparsira je u predviđenu strukturu i proslijedi ConnectionControler komponenti. Praktično realizovan kao DLL projekat.

- ConnectionControler komponenta prihvata zahtjeve posredstvom jednog od sledećih interface-a:

- *ICacheReq* interface preko koga se prihvataju zahtjevi od strane CacheControler komponente. Ovaj interface je zapravo kompozicija (on nasljedjuje) interface za rukovanje Consumption, Audit i Geographic sadržajem kao i interface za funkcionalne pozive nad bazom podataka (pogledati klasni dijagram)

- *IFileReq* je interface preko koga ConnectionControler prihvata zahtjeve od strane FileControler komponente a u trenutnoj implementaciji je u pitanju jedino zahtjev za upisom sadržaja učitanog iz file sistema.

- CacheControler je komponenta koja svoje metode izlaže posredstvom *ICacheOps* interface-a kojeg koristi UI komponenta za rukovanje consumption, audit i geography sardžajem i preko nje ona posredno pristupa ConnectionControler-u i bazi podataka.

Praktično je realizovan kao WCF (client) projekat i predstavlja tačku izlaza sistema na mrežu.

1.2. DatabaseModule

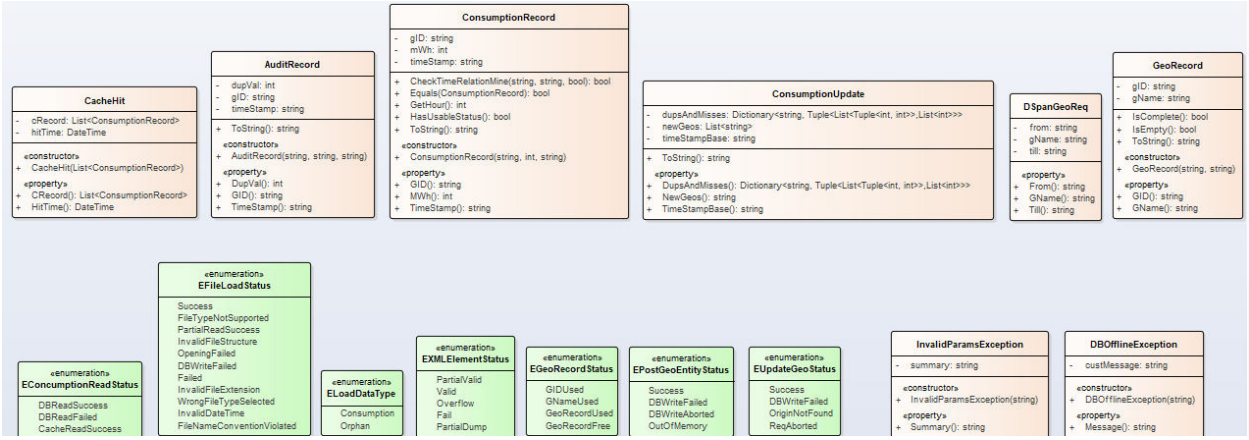
DatabaseModule se sastoji od:

- DataCacheClientService komponente koja implementira *IDBReq* interface i praktično je ralizovan kao WCF (server) projekat i predstavlja tačku ulaza u sistem za rukovanje bazom podataka (u širem smislu). U zavisnosti koja je metoda pozvana (kojoj grupi pripada: Audit, Geography, Consumption ili Functional) poziva se metoda adekvatnog servisa (servisni sloj baze podataka). Mozemo reći da se DataCacheClientService ponaša kao dispacher koji rasporedjuje zahtjev na njemu odgovarajući servisni objekat.

- GeographyService, AuditService, ConsumptionService su komponente koje zahtjeve prosljedjuju na na DAO sloj koji direktno komunicira sa bazom podataka. U trenutnoj implementaciji FunctionalService je izuzetak te on nema pristup bazi podataka nego služi za validaciju da li je baza podataka dostupna ili nije.

2. Class Diagram Client

2.1.1. Pomoćne klase



Sl. 2.1.1.1. Pomoćne klase, enumeracije i izuzeci

`CacheHit` je klasa koja predstavlja jedan pogodak pretrage koji se čuva u keširanoj strukturi. U sebi sadrži polja:

- `cRecord` tipa `List<ConsumptionRecord>` - Lista rezultata pogodjene pretrage
- `hitTime` tipa `DateTime` – Trenutak pogotka (koristi ga `CacheGarbageCollector`)

`AuditRecord` je kalsa koja opisuje jedan Audit zapis i u sebi sadrži polja:

- `dupVal` tipa `int` – vrijednost koja je registrovana kao duplikat u odnosu na originalni zapis. Ima vrijednost `-1` ako originalni zapis ne postoji a postoji zapis sa svim podacima sem vrijednosti potrošnje.
- `gID` tipa `string` – identifikator geografskog područja u kome se taj `AuditRecord` desio
- `timeStamp` tipa `string` – `yyyy - MM - dd - hh` format vrmenskog trenutka kada je taj `AuditRecod` zabilježen.

`ConsumptionRecord` je klasa koja opisuje jedan potpun zapis o potrošnji i sadrži polja:

- `gID` tipa `string` - identifikator geografskog područja u kome se taj ConsumptionRecod desio
- `mWh` tipa `int` – očitana vrijednost ostvarene potrišnje
- `timeStamp` tipa `string` - `yyyy - MM - dd - hh` format vrmenskog trenutka kada je taj `ConsumptionRecod` zabilježen

Pored property – ja ova klasa sadrži i metode:

- `CheckTimeRelationMine` koja za trenutni objekat provjerava da li je on ranije, identičan ili prije proslijeđenog vremenskog trenutka. Argumenti metode jesu: referentni datum (`string`), vremenska relacija (`string`: `>, <, >=, <=, =`) i indikator da li u provjeri treba ignorisati vrijednost sata (`bool`).
- `GetHour` metoda koja izdvaja sat iz `timeStamp -a`

- `HasUsableStatus` koji provjerava da li je konstruisani objekat valjano popunjen

`ConsumptionUpdate` klasa je klasa koja je nusprodukt upisa u bazu podataka i služi za dovođenje lokalne kopije u konzistentno stannje. U sebi sadrži polja:

- `dupsAndMisses` tipa `Dictionary<string, Tuple<List<Tuple<int, int>>,List<int>>>` i predstavlja riječnik koji za ključ ima geografsko područje a za vrijednost `Tuple` čiji je `item1` lista `Tuple` – ova koji imaju parove `sat` – `duplikatVrijednost` a `item2` lista časova čija potrošnja nije uspješno pročitana.
- `newGeos` tipa `List<string>` koja predstavlja listu geografskih identifikatora novih geografskih podrucja koja prije minulog zapisa nisu bila zabilježena. Treba napomenuti da su u ovom kontekstu identifiaktor i naziv identični stoga se prenosi samo jedan `string`.

`DspanGeoReq` je klasa koja enkapsulira sve podatke potrebne za pretragu i u sebi sadrži polja:

- `from` tipa `string` – najniži rubni datum
- `till` tipa `string` – najveći rubni datum
- `gName` tipa `string` – ciljano geografsko područje

`GeoRecord` je klasa koja enkapsulira geografski zapis i u sebi sadrži polja .

- `gID` tipa `string` – ID geografskog podrucja (poznat samo `CacheControler` –u ne i `UI`)
- `gName` tipa `string` – naziv geografskog područja (poznat `UI` i `CacheControler` - u).

pored property – ja ova klasa sadrži i metode `IsComplete` (provjerava da li je kompletan zapis) i `IsEmpty` (provjerava da li je zapis prazan).

2.1.2. Enumeracije

`EConsumptionReadStatus` enumeracija koja opisuje ishod zahtjeva za čitanjem potrošnje:

- `DBReadSuccess` – pročitan iz baze podataka
- `DBReadFailed` – nije pronadjen u lokalu a distrib. baza je nedosupna
- `CacheReadSucces` – pročitano iz `cache` – strukture kao interval ili podinterval

`EFileLoadStatus`:

- `Success` - uspješno pročitano i upisano u bazu podataka
- `FileTypeNotSupported` - tražen upis tipa falja koji nije podržan (trenutno je to sve što nije ostv)
- `PartialReadSuccess` – uspješno pročitano i zapisano ali sa prisutnim odbačenim zapisima (loše strukturirani ili neprihvatljivo nepotpuni)
- `InvalidFileStructure` - cio `XML` fajl je pogrešno strukturiran
- `OpeningFailed` - datoteka ili ne postoji ili je zauzeta od strane drugog procesa
- `DBWriteFailed` - baza podataka je nedostupna
- `Failed` - propalo iz nekog drugog razloga

- `InvalidFileExtension` - nije podržana ekstenzija (trenutno sve izuzev `.xml`)
- `WrongFileTypeSelected` – u `GUI` – ju je pokrenut program za obradu tipa podatka koji se ne slaže sa odabranim tipom podatka
- `InvalidDateTime` - timestamp nije valjan u kontekstu odabranog tipa podatka ili je uopšteno neprihvaljiv
- `FileNameConventionViolated` - konvencija zapisa naziva fajla nije valjana

`ELoadDataType` enumeracija predstavlja tipove fajlova koji se koriste:

- `Consumption` – potrošnja
- `Orphan` - apstrakcija nepodržanih tipova (korišteno u `Unit` testovima)

`EXMLElementStatus` je enumeracija koja opisuje stanje jednog `XML` čvora koji predstavlja (tenutno) potošnju:

- `PartialValid` - nepotpun ali dovoljno za audit
- `Valid` - potpun
- `Overflow` - ima viška podčvorova ali je uzeto šta je u trenutnoj implementaciji potrebno
- `PartialDump` - nepotpun ali nedovoljno da se iskoristi
- `Fail` - totalno pogrešna struktura elementa (npr. više od 1 polja za potrošnju i sl.)

`EGeoRecordStatus` je enumeracija koja opisuje stanje kandidata za zapis novog geografskog područja:

- `GIDUser` - ključ je već u upotrebi
- `GNameUsed` - ime već u upotrebi
- `GeoRecodUsed` - identičan zapis već postoji
- `GeoRecordFree` - nije u upotrebi , može se slobodno upisati

`EPostGeoEntityStatus` je enumeracija koja opisuje ishod operacije za upis novog geografskog područja:

- `Success` - uspješno
- `DBWriteFailed` - distrib. baza podataka je nedostupna
- `DBWriteAborted` - parametri poslati na upis nisu valjani ili su nepotpuni
- `OutOfMemory` - predviđena struktura za keširanje geografskih područja je prepunjena

`EUpdateGeoStatus` je enumeracija koja opisuje ishod operacije za ažuriranje postojećeg geografskog područja:

- `Success` - uspješno
- `DBWriteFailed` - distrib. baza podataka je nedostupna
- `OriginNotFound` - izvorni zapis ne postoji
- `ReqAborted` - parametri zahtjeva su nepotpuni ili nisu valjani

2.1.3. Izuzeci

`InvalidParamsException` je izuzetak koji se baca kada se metodi proslijedi `null`, prazne, nepotpune ili loše vrijednosti. Sadrži opis greške u polju `Summary`.

`DBOfflineException` je izuzetak koji se baca kada se na izlasku na mrežu desi izuzetak na kanalu (`EndpointNotFoundException` ili `CommunicationObjectFaultedException`) te je detaljnije opisan poljem `Message`.

Grupa izuzetaka koja se nalazi u okviru `DistributedDB` nije korištena na klijentskoj strani i namjenjeni su samom administratoru i developeru tokom razvoja:

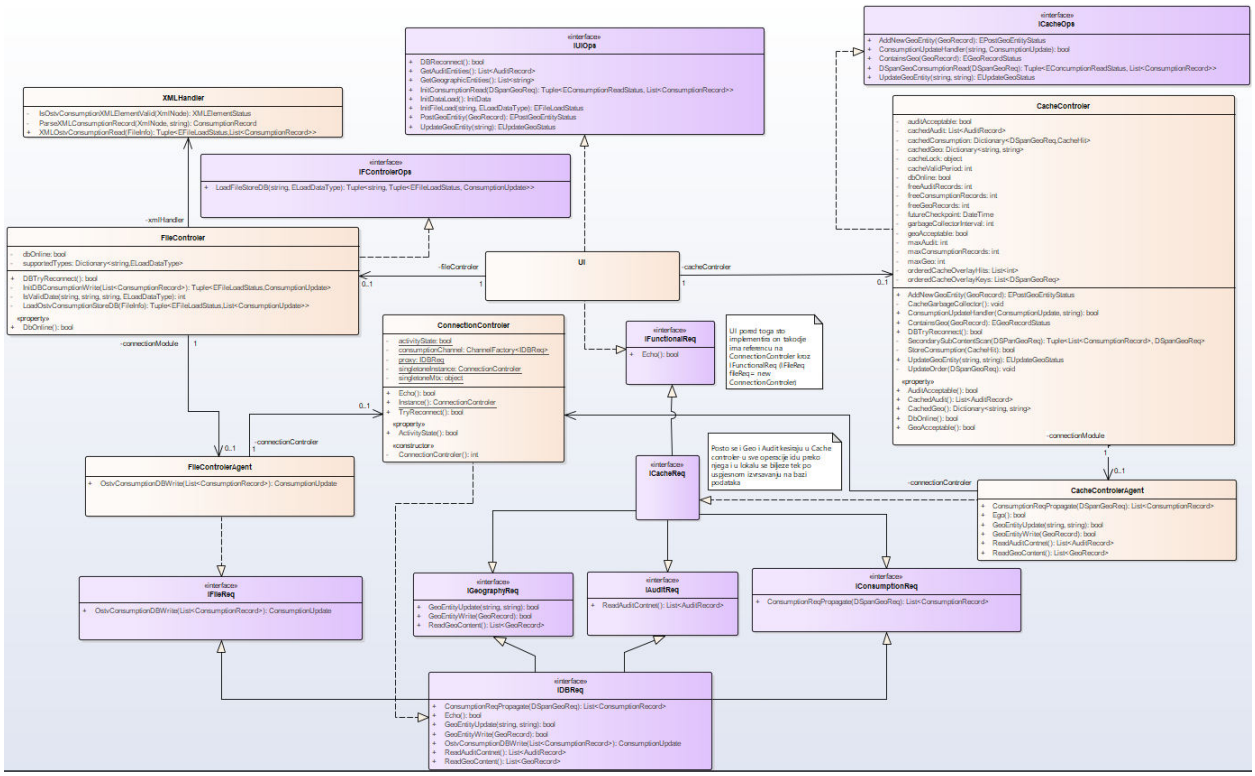
`DBLoginFailed` – izuzetak kada parametri za konekciju na bazu podataka nisu valjani (`login_params.txt`)

`ForbiddenOrderException` – sličan `StillAttachedException` – u ali se odnosi na relaciju nadklasa podklasa izmedju `EES Record` –a i `Consumption/Audit Record` –a . (pogledati poglavlje o bazi podataka).

`PrimaryKeyConstraintViolation` – izuzetak kada administrator ili neki drugi client agent pokuša ručno unošenje zapisa sa primarnim ključem koji je već u upotrebi.

`StillAttachedException` - izuzetak kada se pokuša uklanjanje zapisa u bazi podataka iako su oni i dalje referencirani od strane drugih objekata (npr. brisanje geografskog područja ako i dalje postoje zapisi koji su za njega vezani).

2.2 UI klasa



Sl. 2.2.1. Klasni dijagram klijentskog dijela aplikacije

`UI` kalsa predstavlja apstrakciju `UI`-a i instancira se pri podizanju aplikacije. Ova klasa predstavlja jedinu ulaznu tačku u sistem te ona skriva sve detalje implementacije i korisniku pruža poziv samo onih metoda koje ona izlaže kroz *IUIOps*.

IUIOps interface sadrži sljedeće metode:

- `bool DBReconnect()` – Ponovno konektovanje na distribuiranu bazu podataka

- `EUpdateGeoStatus UpdateGeoEntity(string oldName, string newName)` – Ažuriranje
- imena postojećeg geografskog zapisa (izmjena u lokalu i u DB)

- `EPostGeoEntityStatus PostGeoEntitiy(GeoRecord gEntity)` – Zadavanje novog geografskog područja (izmjena u lokalu i u DB)

- `EFileLoadStatus InitFileLoad(string filePath, ELoadDataType dataType)` – Učitavanje konkretnog tipa podatka (ostvarena postrošnja u našem slučaju) sa proslijeđene putanje

- `Tuple<EConcumptionReadStatus, List<ConsumptionRecord>> InitConsumptionRead(DSpanGeoReq dSpanGeoReq)` – Čitanje podataka o potrošnji za zadato geografsko područje i zadati datumski opseg.

- `List<string> GetGeographicEntities()` – Dobavljanje keširanih geografskih pojmova

- `List<AuditRecord> GetAuditEntities()` – Dobavljanje keširanih Audit zapisa

`UI` klasa ima reference na `FileControler` (preko koga inicira čitanje fajlova i automatski upis u bazu podataka) i `CacheControler` (preko koga dobavlja Audit, Geography i Consumption sadržaj te vrši rekonekciju na bazu podataka ako je ona bila nedostupna.). `UI` kasa takođe nakon uspješnog upisa u bazu podataka posredstvom metode `LoadFileStoreDB` koristi povratnu vrijednost ove metode kao atribut pozivajući metodu `CacheControler` – a `ConsumptionUpdateHandler` koja prosljeđenu joj vrijednost koristi za dovođenje spostvene lokalne kopije Gaography i Audit zapisa u konzistentno stanje.

2.3. FileControler & XMLHandler klasa

`UI` komponenta učitavanje datoteke inicira kroz metodu `LoadFileStoreDB` proslijeđujući joj identične parametre proslijeđene funkciji `InitFileLoad` (koja je njen pozivalac). U ovoj metodi `FileControler` –a se provjerava da li je podržan prosljedjeni tip datoteke (u trenutnoj implementaciji samo ostv.), da li je datum (izvučen iz naziva datoteke) validan za proslijedjeni tip (da li je uopste validan + da li je prošlo vijeme jer je u pitanju ostvarena potrosnja).

Validnost datuma provjerava privatna metoda `IsValidDate`. Nakon prolaska svih provjera za odgovarajući tip datoteke (ostv) se poziva adekvatna privatna metoda (u trenutnoj implementaciji je to privatna metoda `LoadOstvConsumptionStoreDB`). Unutar `LoadOstvConsumptionStoreDB` metode se provjerava koja je ekstenzija fajla (u trenutnoj implementaciji je to samo `.xml`).

`XMLHandler` posredstvom metode otvara (pokušava da otvori) `XML` fajl, provjerava konvenciju njegovog imena i redom parsira sve one čvorove koji su valjani. Metodom `IsOstvConsumptionXMLEmentValid` on provjerava valjanost čvora (ranije spominjana stanja čvora u sekciji enumeracija) te u svim onim zadovoljavajućim stanjima poziva `ParseXMLConsumptionRecord` metodu koja parsira čvor i vraća ga enkapsuliranog u `ConsumptionRecord` objektu.

Nakon uspješnog parsiranja `XML` fajla u listu zapisa poziva se metoda `ConnectionAgent` – a `InitDBConsumptionWrite` u kojoj se učitani zapisi proslijeđuju na upis. Povratna vrijednost ove metode (`ConsumptionUpdate`) se dalje vraća kroz lanac pozivanja sve do `UI` (sa tim da se u `LoadFileStoreDB` povratna vrijednost proširuje sa `timeStampBase`). `TimeStampBase` nam je potreban sa obzirom na to da `ConsumptionUpdate` nosi informacije o duplikatima i promašajima po satima pa tim satima treba dodati kontekst datuma koji je zajednički na nivou učitane datoteke.

Napomena: `FileControler` i `UI` imaju istoimene klase sa sufiksom `Agent` koje su njihovi proxy – ji i skrivaju ostale metode `ConnectionControler` – a koje oni nemaju pravo da „vide“. Alternativa ovome bi bilo dodjeljivanje reference na konekcioni modul promjenjivoj tipa interface.

2.4. ConnectionControler klasa

`ConnectionControler` klasa je klasa koja implementira *IDBReq* interface koji objedinjuje *IFileReq* interface (koji izlaže šta FileControler može da zahtjeva od `ConnectionControler` –a), *IGeographyReq* (svi zahtjevi geografske prirode), *IAuditReq* (svi zahtjevi audit prirode), *IConsumptionReq* (svi zahtjevi vezani za porošnju), *IFileReq* (svi funkcionalni zahtjevi). Ova komponenta predstavlja izlaz na mrežu i realizuje WCF klijenta. Podaci o serveru koga gadja su smješteni u `App.Config` fajlu `GUI_Integrator` projekta. Ova klasa je realizovana prema singletone design pattern – u da bi se osigurala jedinstvena tačka izlaza na mrežu.

2.5. CacheControler kalsa

Klasa koja realizuje `cache` mehanizam i služi kao kapija za izlaz `UI` komponente na mrežu (u kontekstu čitanja potrošnje). Metode koje `UI` može da poziva klasa izlaže kroz *ICacheOps* interface. Prilikom instanciranja u konstruktoru se podešavaju određeni parametri i inicijalizuju strukture koje klasa koristi za keširanje. Inicijalizuje se i `CacheControlerAgent` koji ima istu funkciju kao i `FileControlerAgent` i unutar njega se dobavlja singletone instaca `Connection controler` – a. Nakon toga se osluškuje baza i provjerava da li je dostupna, ako jeste onda se dobavljaju inicijalni Audit i Geo zapisi i smiještaju u strukture njima namjenjene za čuvanje. Smještanje se obavlja ako je pročitano u okvirima maksimalne predviđene veličine dok se u suprotnom to čitanje odbacuje i diže se flag koji korisnika informiše o tome. Takodje stanje dostupnosti baze je ispraćeno adekvatnim flag – om. Na kraju konstruktora pokreće se detach-ovan `CacheGarbageCollector` posredstvom task – a.

Ova nit se budi svaki minut i provjerava da li su neki od `cache` zapisa prestarili, ako jesu on ih briše. Period važenja je 3 časa dok je rezolucija brisanja (buđenja niti koja briše) jednaka 1 minut. Proces provjere i oslobađanja se obavlja pod mutual – exclusive pristupu jer sve metode koje rukuju sa podacima koji se keširaju moraju zauzeti mutex da ne bi došlo do štetnog preplitanja.

Strategija oslobadjanja memorije za novu pretragu ako nema dovoljno memorije (trenutne nisu prestarile) je ta da se zbog brzine čitanja koristi `Dictionary` sa `CacheHit` – ovima (`CacheHit` sadrži rezultat pretrage i vrijeme kada je prvi put pogođena) koji se pretražuje po parametrima pretrage (ključ je `DspanGeoReq`). Pošto je `Dictionary` neuredjena struktura nad njim su realizovane dvije liste koje su po indeksima uparene: `orderedCacheOverlayKeys` koja sadrži listu ključeva sortiranih po opadajućem redu prema broju pogodaka za njima odgovarajući hit. Druga lista jeste `orderedCacheOverlayHits` koja sadrži broj pogodaka za upareni ključ (`orderedCacheOverlayKeys[0]` ključ ima `orderedCacheOverlayHits[0]` pogodaka i analogno).

Na ovaj način se lako eliminiše onoliko najstarijih (zadnjih) zapisa kada je potrebno osloboditi memoriju.

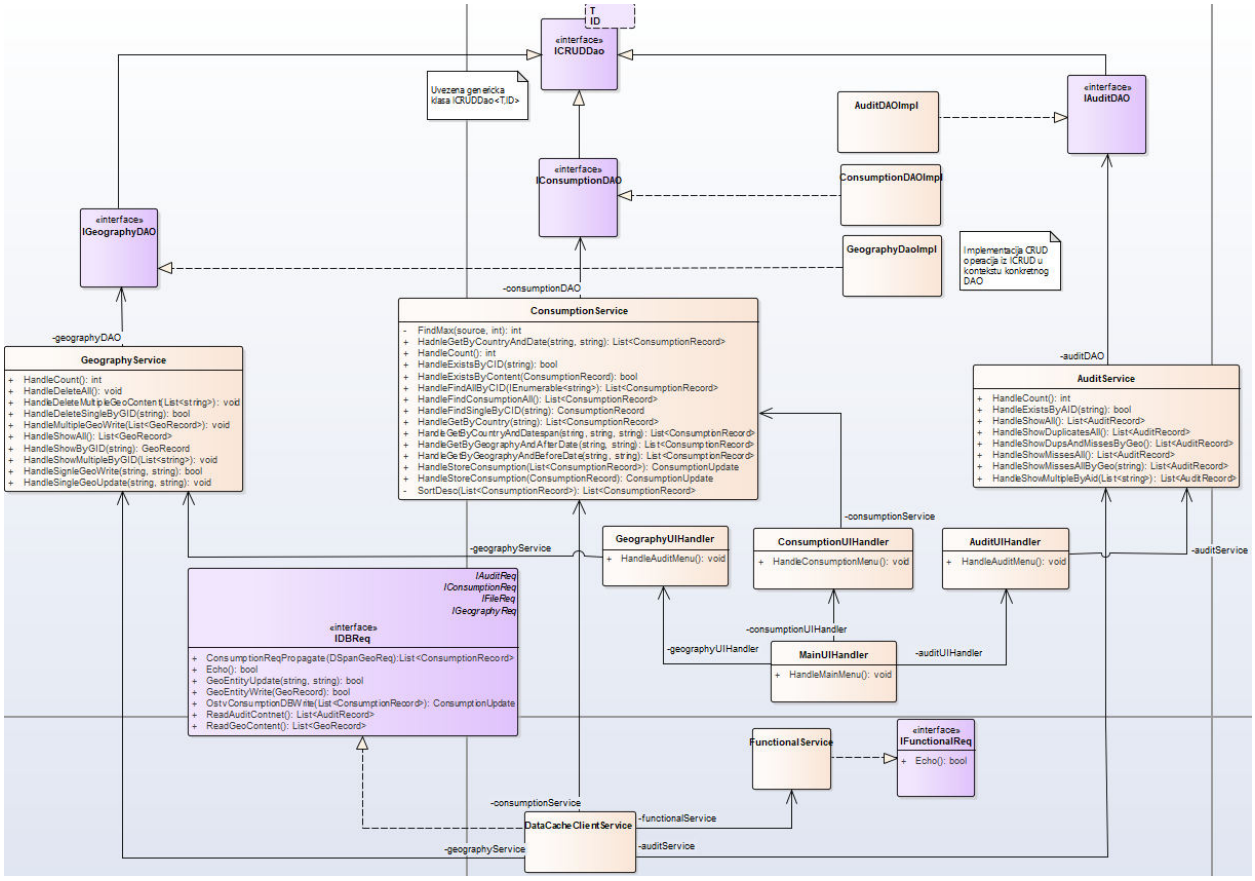
Pored trivijalnih metoda posebno treba pomenuti `DspanGeoConsumptionRead` metodu koja dobavlja tražene zapise. Ako je pretraga zabilježena ona se odma isporučuje iz `cache` – a i njen `HitRate` se povećava te se njena pozicija u sortiranim nizovima ažurira (metoda `UpdateOrder`). Ako je tražena pretraga podinterval neke druge pretrage (metoda `SecondarySubcontentScan`)

tada se taj podinterval izvlači i dostavlja korisniku a `HitRate` se tretira na isti način kao u prethodnom slučaju. Ako mapiranje nije poznato onda se ono dobavlja iz baze podataka kešira, u lokalu i onda dostavlja korisniku te se podaci o tom mapiranju smještaju na kraj sortiranih listi (sigurno ima najmanje hit-ova).

Sve ostale trivijalne metode koje izlaze na mrežu prvo izvrše promjene distribuirano pa se tek onda ažurira lokalna kopija.

3. Class Diagram Distributed Database

Distribuirana baza podataka je relizovana kao WCF projekat koji u svojoj `Main` metodi podiže `ServiceHost` koji host - uje servis sa metodama *IDBReq* kojeg implementira klasa `DataCacheClientService`. Ova klasa ima reference na sva 4 servisa (Geography, Audit, Consumption i Functional) te u zavisnosti koja je metoda (distribuirano) pozvana ona je propagira na odgovarajuću metodu servisnog sloja. Servisni sloj zahtjev propagira dalje na `DAO` sloj gdje se izvršavaju upiti nad bazom podataka. Konekcioni parametri baze podataka se dobavljaju preko `DBConnectionParams` klase pomocu regularnih izraza te se sistem može lako prevezati na drugu bazu podataka promjenom konekcionih parametara unutar `login_params.txt` datoteke.



Sl. 3.1. Klasni dijagram distribuirane baze podataka

Unutar `DistributedDB` projekta je realizovana i klasa `MainUIHandler` koja se instancira nakon podizanja servisa. Pošto je `DistributedDB` projekat realizovan kao konzolna WCF aplikacija te je iskorištena klasa `MainUIHandler` realizovan je "Administrator Agent" nad bazom podataka koji je korišten pri testiranju i ima širi spektar naredbi koje su implementirane nad bazom podataka.

Ovaj dokument se zbog sažetosti neće osvrtati na implementaciju samog `DAO` sloja tj. upita te je detaljnije objašnjenje istih ostavljeno u vidu komentara unutar samog projekta.

4. Struktura baze podataka

DataCache

Model baze podataka u kojoj se skladiste podaci o potrosnji, geografskim podrucjima i Audit log-ovi

RECID - ID upisa (moze i kao kompozitni kljuc TimeStamp+GID ali se ovako radi da bi se ID zavisnost rasteretila

TimeStamp - Vrijeme zapisa (Datum + Sat)

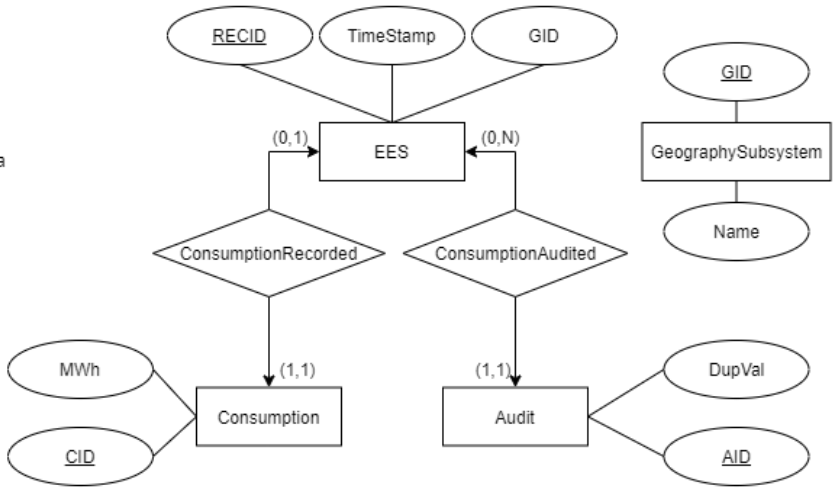
GID - ID geografskog podrucja

CID - ID zabiljezene potrosnje

AID -ID auditovan-og zapisa

MWh - Potrosnja geografskog prodrucja u zabiljezenom trenutku

DupVal - Vrijednost zabiljezenog duplikata



EES je apstrakcija koja opisuje trenutak koji se razmatra, na taj trenutak se pomocu poveznika ConsumptionRecorded "kaci" originalna potrosnja biljezena u tabeli Consumption kao i dodatni auditovani duplikati smjesteni u tabeli Audit posredstvom poceznika Audited. Zarad ustede u memoriji biljezenje zapisa kojima nedostaje podatak o potrosnji cemo mock-ovati postojanjem EES zapisa bez i jednog Consumption i Audit zapisa.

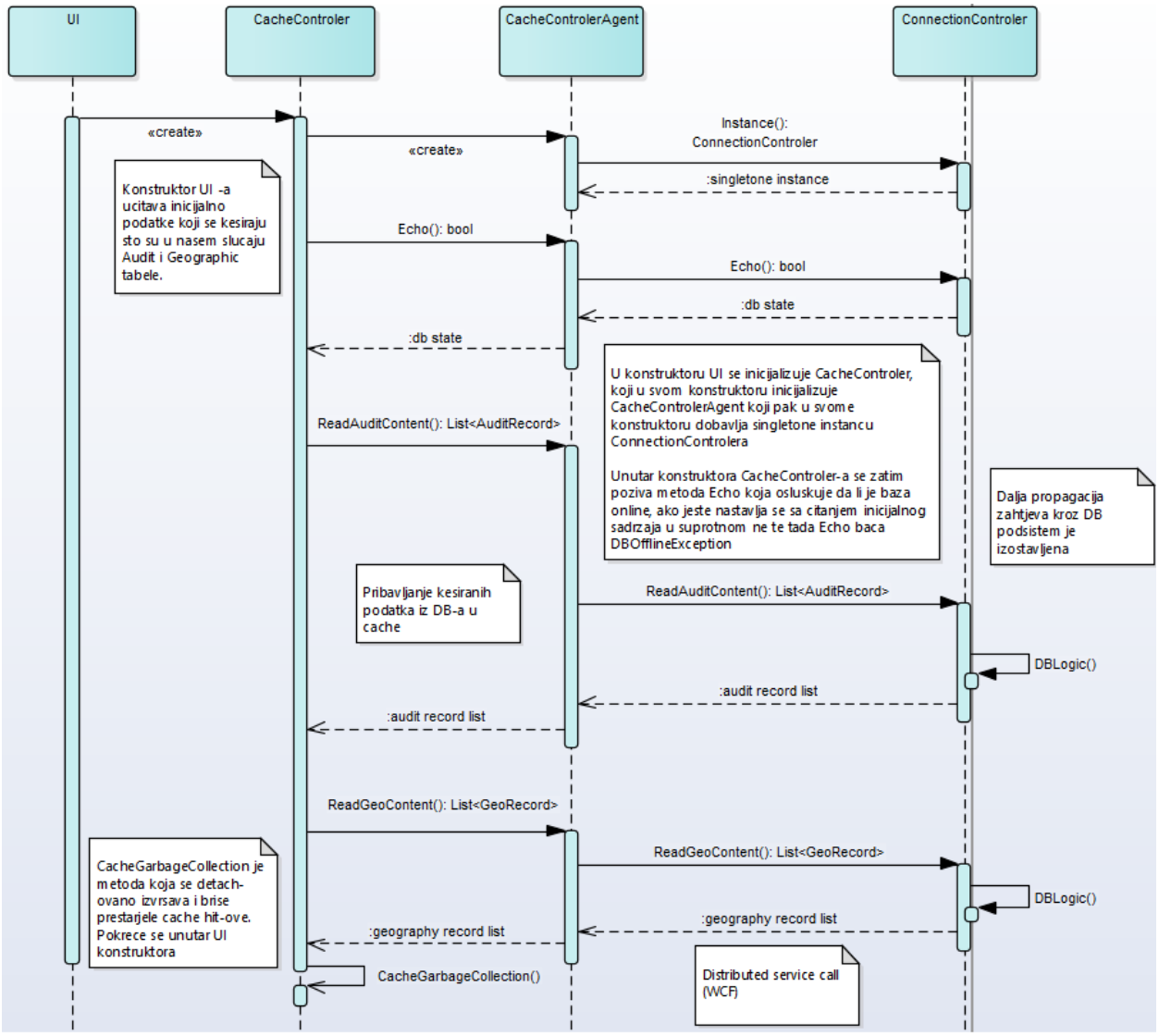
Brisanjem EES zapisa brisu se i Consumption i Audit zapisi. Brisanjem Consumption zapisa brisu se i njegovi Audit-i koji postaju besmisleni jer ni jedan vise ne mozemo proglasiti originalom. U samom softveru ce biti regulisana nemogucnost postojanja Audit zapisa ako ne postoji njima odgovarajuci Consumption.

Potrebno je naglasiti i to da je zbog konzistentnosti baze prisutno pitanje postojanja EES zapisa ciji GID ne odgovara niti jednom zapisu iz GeographySystem tabele tj. drugacijie formulisano: Moguce je obrisati zapis unutar GeographySubsytem -a iako je ona logicki vezan za njemu odgovarajuce zapise iz EES tabele. Zbog specificnosti implementacije da bi se izbjeglo dupliranje podataka (poveznik izmedju EES i GeographySybsystem) pomenuta problematika ce biti rjesena unutar relizacije metoda brisanja i dodavanja. Iz ovoga proizlazi to da konzistencija baze zagarantovana samom semom baze podataka nego slojem koji komunicira sa ORACLE SUBP-om.

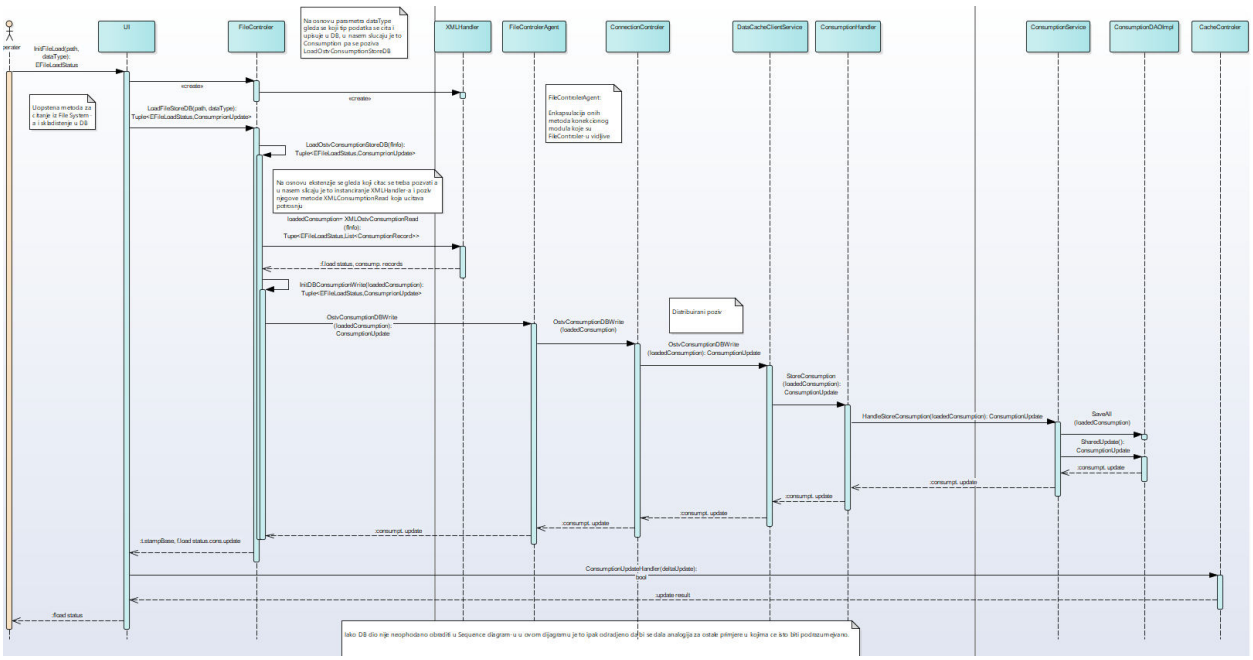
Ovako realizovana baza podataka omogućava da za jedan vremenski trenutak u jednom geografskom području (jedan `EES` record) možemo vezivati jedan `ConsumptionRecord` i sa njime vezanih više `AuditRecord` – a (dupliakta) bez nepotrebnog dupliranja podataka apstrahovanih unutar `EES` zapisa.

5. Sequence Diagrams

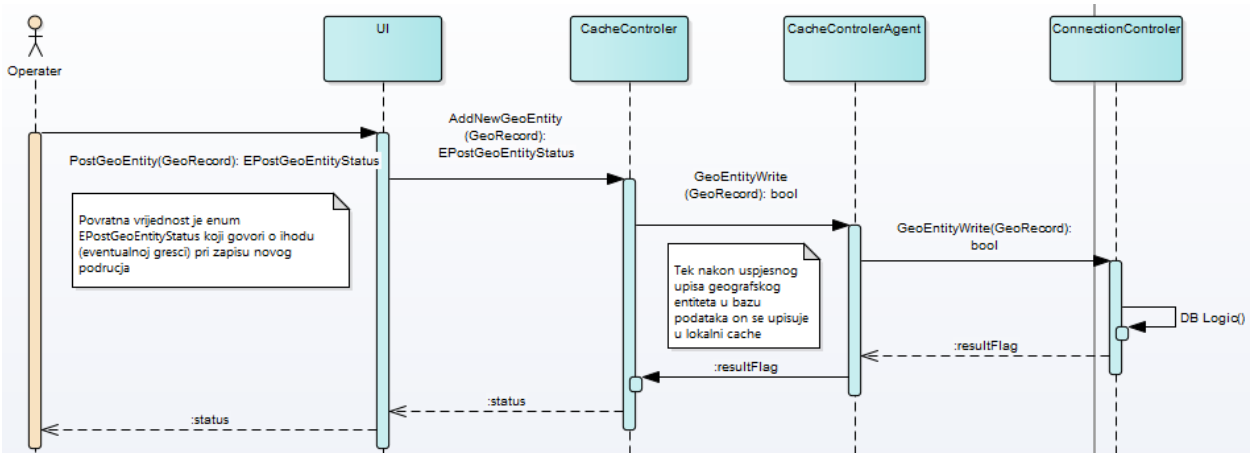
U ovom poglavlju su istaknuti dijagrami sekvence ključnih operacija koje su već detaljno rastumačene u ranijem dijelu dokumenta.



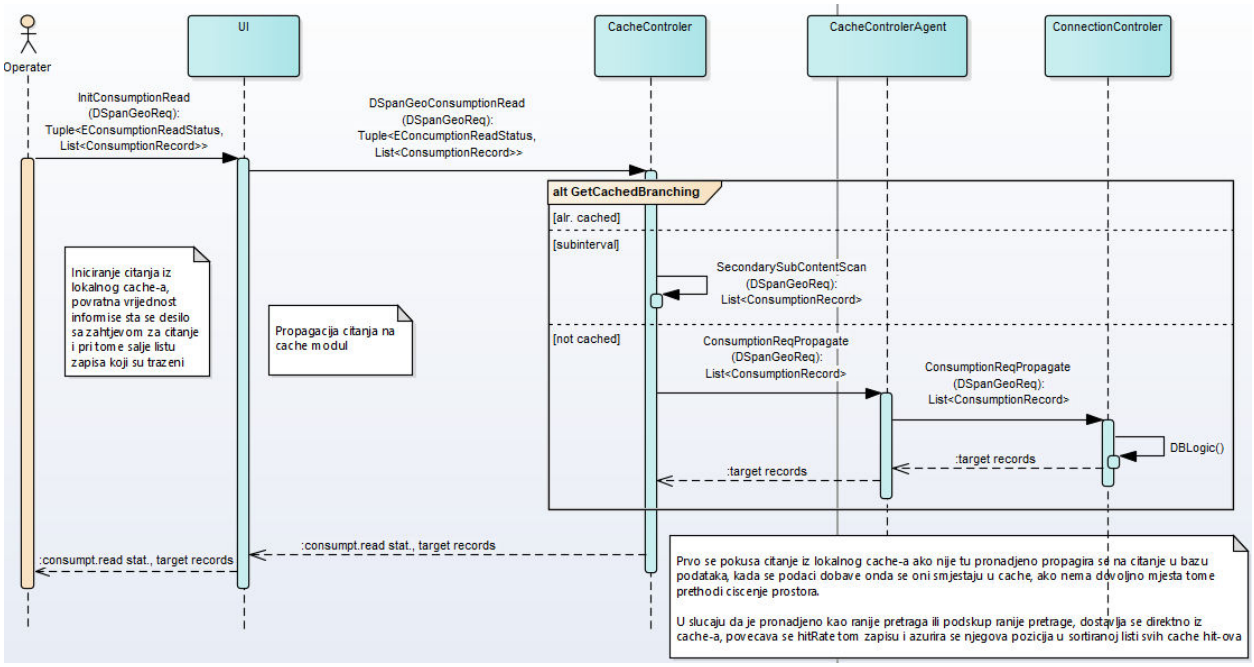
Sl. 5.1. Caching Initial Data



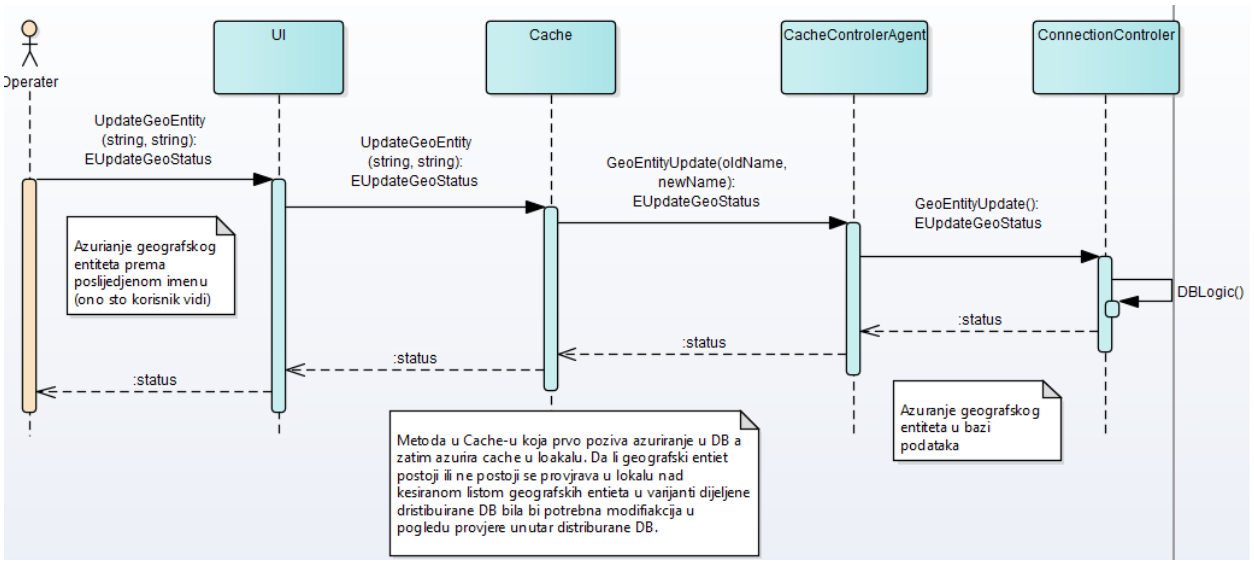
Sl. 5.2. Importing Data From XML File



Sl. 5.3. Posting Geographic Entity



Sl. 5.4. Reading Consumption



Sl. 5.5. Updating Geographic Entity

NAPOMENA: Unutar projekta u vidu komentara su ostavljena pojašnjenja pojedinih metoda kao i dalje smijernice za proširenje i poboljšanje projekta.