Dokumentacija projektnog zadatka tima 17

Replikator

Članovi tima:

Milan Tomin PR 159/2017

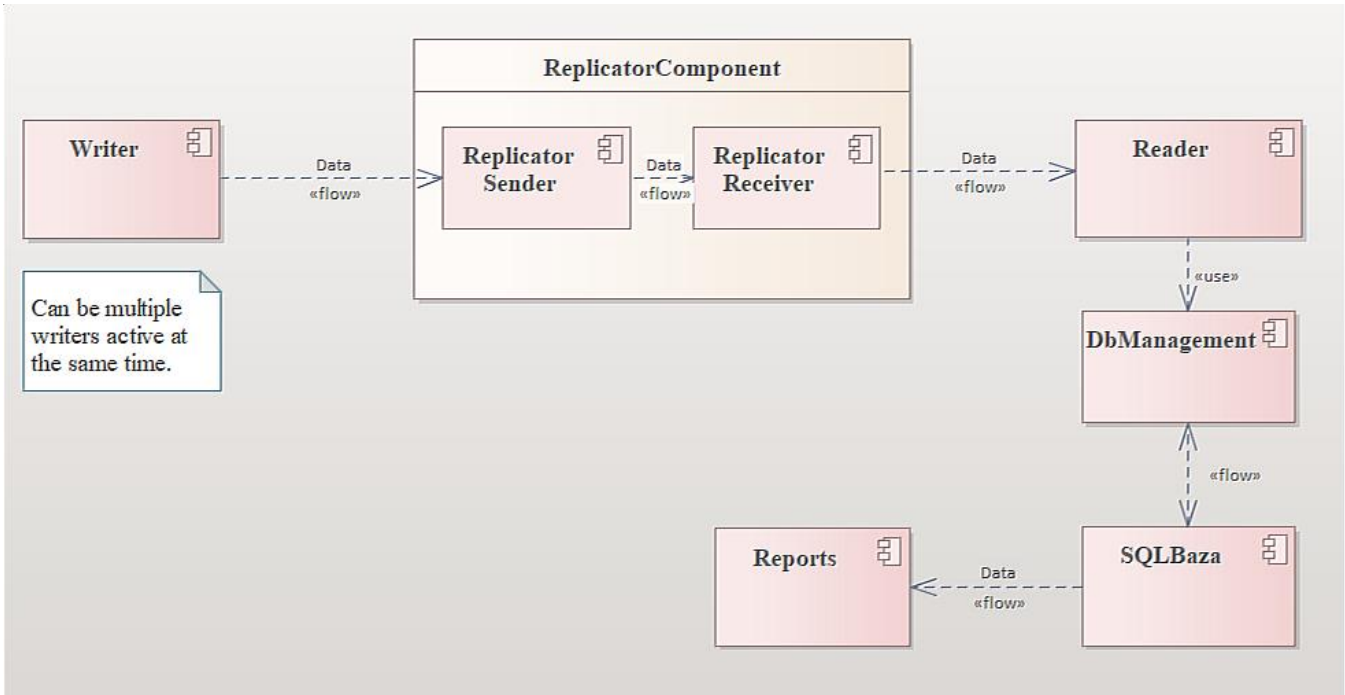
Marko Lazović PR 151/2017

Bojan Jakovljević PR 41/2016

Aleksandar Nestorović PR 13/2017

Sadržaj:

Projekat: Replikator



Slika 1: Component diagram Replicator aplikacije

Korištene tehnologije i biblioteke:

Implementacija: Projekat je implementiran u programskom jeziku visokog nivoa (high-level) Python. Biblioteke koje su korištene za implementaciju projekta su u paketu standardnog skupa biblioteka Python-a. Neke od važnijih biblioteka:

Socket: biblioteka socket podrazumeva mrežno programiranje na nižem nivou. Biblioteka komunicira i inicira sistemske pozive operativnog sistema. Ova biblioteka je python verzija socket biblioteke pisane za UNIX u programskom jeziku C. U projektu korištena je za komunikaciju između modula uz pomoć TCP protokola.

Sqlite3: je sistem za upravljanje bazom podataka koji je brz, visoko pouzdan sistem sa svim funkcionalnostima koje se očekuju od SUBP-a. SQLite format baze podataka je stabilan i kompatibilan na većini platformi. Za potrebe projekta korišten je kao baza podataka za skladištenje podataka o korisnicima i njihovoj potrošnji.

Testovi: za potrebe pisanja testova u projektu korištena je biblioteka **unittest**. Biblioteka unittest omogućava pisanje, pokretanje i agregaciju testova u kolekcije. Za potrebe proračuna pokrivenosti koda testovima korišćena je biblioteka **coverage**.py. Ova biblioteka vrši proračun pokrivenosti koda

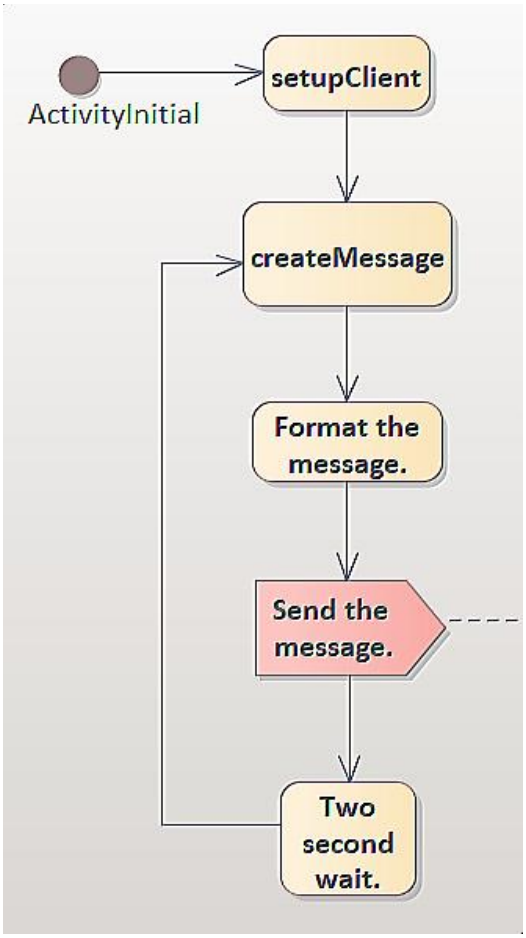
testovima, uključujući i individualne grane u aplikaciji. Takođe generiše izveštaje o pokrivenosti koda testovima u konzolnom tekst formatu, HTML, XML, JSON i LCOV-u.

Osnovni pregled funkcionalnosti:

Zadatak aplikacije je da simulira generisanje, obradu, čuvanje i ispis podataka o potrošnji vode, kao i generisanje i pregled podataka o potrošačima. *Writer* komponenta periodično generiše podatke o potrošaču i potrošnji vode, koje zatim prosleđuje *Replicator Sender* komponenti. Instanci *Writer* komponenti može biti više, tj. *Replicator Receiver* mora omogućiti server konekciju svim *Writer* instancama.

Replicator Sender prima Writer poruke o potrošnji u JSON formatu i prosleđuje *Replicator Receiver* komponenti. Ova komponenta privremeno skladišti podatke i periodično ih šalje *Reader* komponenti. *Reader* komponenta podatke skladišti i ažurira podatke u bazi podataka i čita iz iste posredstvom *DbManagement* komponente, statičke biblioteke funkcija koja služi za komunikaciju sa bazom podataka. Poslednja u nizu je *Reports* komponenta, koja služi za ispis izveštaja potrošnje u skladu sa izabranim kriterijumom. Da bi komponenta generisala ove izveštaje ona komunicira direktno sa bazom podataka.

Writer



Slika 2: Activity diagram Writer komponente

Writer komponenta služi za generisanje i periodično slanje podataka o potrošnji *ReplicatorSender* komponenti. Ona se sastoji iz četiri funkcije:

- *setupClient()* – Služi za uspostavljanje klijentske strane konekcije, kako bi se omogućilo slanje podataka. Povratna vrednost joj je objekat koji predstavlja klijentsku konekciju.
- *createMessage()* – Služi za generisanje vrednosti id(identifikacioni broj potrošača) i water consumption(količina potrošene vode). Povratne vrednosti su joj id i water consumption.
- *formatMessage(id, cnspn)* – Ulazne vrednosti ove funkcije su id i water consumption, dobijene iz prethodno opisane funkcije. Koriste se kako bi se oformio string (JSON format), koji se kasnije prosleđuje narednoj komponenti. Izlazna vrednost je formatirani string.
- *sendToSender(client, msg)* – Ulazne vrednosti su objekat koji predstavlja klijentsku konekciju i prethodno formatirani string sa id i water consumption vrednostima. Funkcija date vrednosti koristi kako bi poslala poruku *Replicator Sender-u*.

Funkcije se pozivaju u okviru main-a. Izuzev *setupClient()*, funkcije se pozivaju unutar beskonačne petlje. Nakon slanja poruke, ispisuje se poruka o uspešnom slanju i čeka se dve sekunde pre ponovnog ulaska u petlju.

test_Writer

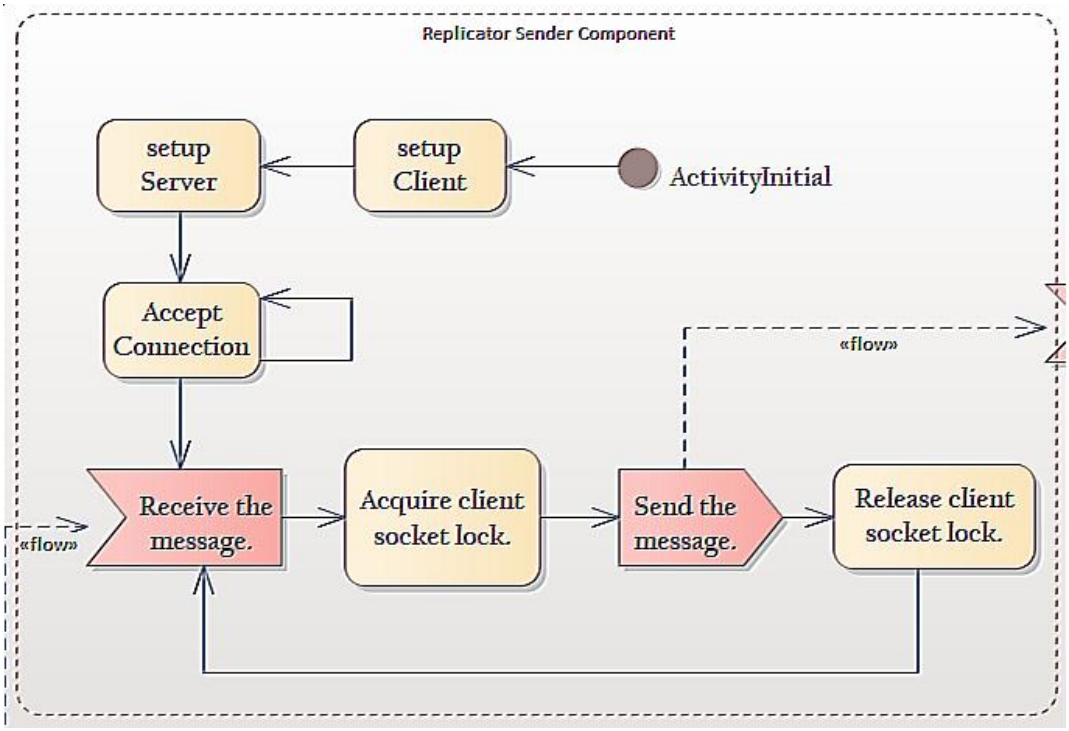
U sklopu testiranja *Writer* komponente, vrši se šest testova:

- *test_clientSetupIsNone()* – Test koji proverava da li je klijent uspešno kreiran.
- *test_address()* – Test koji vrši proveru da li je socket napravljen sa odgovarajućom IP adresom i portom.
- *test_messageFormat()* – Test koji proverava da li je poruka korektno formatirana direktnim poređenjem sa korektnim primerom stringa.
- *test_messageFormat1()* – Proverava da li je poruka korektno formatirana koristeći regularne izraze.
- *test_consumptionRange()* – Test koji proverava da li su podaci o potrošnji u odgovarajućem rasponu vrednosti.
- *test_sendMessage()* – Proverava da li je poruka poslata na odredište koristeći fake server pokrenut u niti koji simulira server kom se šalje poruka.

Replicator

Replikator je komponenta koja u svom okviru ima dve komponente *Replicator Sender* i *Replicator Receiver*. Kao komponenta replikator ima za cilj da poruke dobijene od Writer instanci prenese do komponente Reader, kako bi se podaci kasnije skladištili i ažurirali u bazi podataka. Možemo ovu komponentu smisleno zamisliti kao deo IT infrastrukture neophodnu kako bi potrošnje korisnika Writer komponenta stigle do mesta obrade i skladištenja (Reader komponente).

Replicator Sender komponenta:



Slika 3: Activity diagram Replicator Sender komponente.

Replicator Sender komponenta prilikom pokretanja otvara serversku konekciju koja prihvata klijentske konekcije Writer komponente. Takođe otvara se i klijentska konekcija Replicator Sender-a ka Replicator Receiver-u. Replicator Sender treba omogućiti prihvatanje više Writer klijenta. Za svaku prihvaćenu klijentsku konekciju pokreće se nit, koja uslužava Writer-a. U svakoj serverskoj niti pokrenutoj za zasebnu klijentku konekciju, vrši se prijem poruke i zatim prosleđivanje primljene poruke Replicator Receiver-u. Međutim, treba imati na umu, da je klijentska socket konekcija od Replicator Sender ka Replicator Receiver-a jedna, što znači da ako više serverskih niti pokušaju slati poruku neminovno je da će program imati grešku. Ovo proizilazi iz činjenice da send() i recv() metode socket biblioteke nisu

thread safe. Stoga sendToReceiver() funkcija u ovoj komponenti je kritična sekcija koju zaključavamo uz pomoć Acquire() metode Lock objekta. Svaka serverska nit izvršava handle_client funkciju kojoj je prosleđena referenca na Lock objekat, tj. sve niti imaju referencu na isti Lock objekat. Pozivom Acquite() metode, zaključavamo taj deo koda za ostale niti, koje čekaju da trenutna nit pozove metodu Release() kada naredna nit dobija pristup sendToReceiver() metodi, a ostale niti čekaju dok trenutna ne pozove Release() metodu.

Pregled funkcija:

- setupServer(conn) – funkcija koja otvara serversku konekciju. Povratna vrednost ove funkcije je otvorena serverska konekcija koja čeka accept() metodom pristigle klijentske konekcije Writer instanci.
- setupClient() – Replicator Sender ima uloga i klijenta i servera. Sender je klijent u odnosu na Replicator Receiver komponentu, a server u odnosu na Writer komponentu. Funkcija setupClient ima ulogu da otvoru klijentsku konekciju ka Replicator Receiver komponenti.
- receiveWriterMessage(conn) – funkcija uz pomoć koje se vrši prijem Writer poruka o potrošnji. Argument conn je prihvaćena klijentska konekcija sa kojom se vrši prijem Writer poruka.
- sendToReceiver(client, msg, lock) – Putem ove funkcije poruka primljena receiveWriterMessage(conn) funkcijom se prosleđuje Replicator Receiver komponenti. Kao što je gore pomenuto, klijentska konekcija ka Replicator Receiver komponenti je jedna, tako da više niti ne mogu istovremeno proslediti poruke preko nje, stoga je funkcija kritična sekcija koja na početku ima Acquire() metodu i Relase() metodu. Prosleđeni argumenti su klijentska konekcija ka Replicator Receiver komponenti, poruka za slanje i lock objekat.
- handle_client(conn) – Svaka pokrenuta nit izvršava handle_client() funkciju. Unutar nje je beskonačna petlja koja poziva funkcije receiveWriterMessage() i sendToReceiver(). Prosleđen argument je klijentska konekcija Writer-a koja je prihvaćena.

test_replicatorSender.py

Za svrhu testiranja Replicator Sender-a napisano je osam testova. Sledi kratak opis.

test_setupClient() – Postoje tri testa za testiranje klijentske konekcija ka Replicator Receiver komponenti. Prvi test pokreće simulirani server u niti, i testira da li se connect() metoda uspešno izvršila nad tim simuliranim serverom, povratna vrednost ne bi smela biti None u tom slučaju. Drugi test proverava da li je povratna vrednost nula, ako ne postoji pokrenut server koji bi prihvatio konekciju. Poslednji test za testiranje klijentske konekcije proverava da li je connect() metoda pozvana sa korektnom IP adresom i portom.

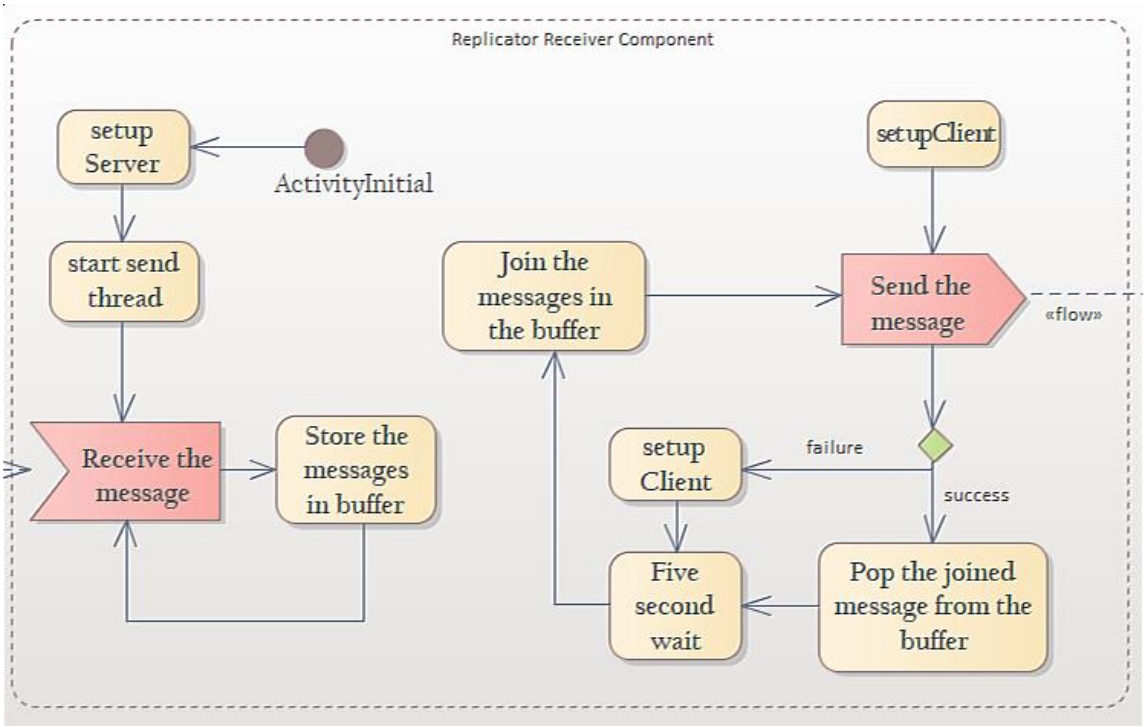
test_setupServer() – je jedna test funkcija koja proverava da li je bind() metoda unutar setupServer() funkcije pozvana sa koretnom IP adresom i portom.

test_sendMessage() – Postoje dva testa za testiranja poslate poruke. Prvi test proverava da li je poruka poslata funkcijom sendMessage() korektno primljena u niti koja izvršava simulirani server. Poslata poruka sendMessage() funkcijom i primljena poruka u niti trebale bi biti iste. Drugi test sendMessage() funkcije proverava da li je sendMessage thread safe. Tri klijentske niti pokušaće da izvrše jednovremeno funkciju sendMessage(). Očekivani ishod je da su poruke primljene zasebno od svake niti. Tako da ako svaka nit šalje poruku „Poruka“ očekivani prijem bi bio „PorukaPorukaPoruka“.

test_receiveWriterMessage() – Test koji proverava da li je poruka primljena uspešno uz pomoć receiveWriterMessage() funkcije. U zasebnoj niti se simulira klijentska konekcija koja šalje poruku. Očekivano ponašanje je da će poslata poruka simuliranim klijentim i primljena poruka receiveWriterMessage() funkcijom biti iste.

test_handle_client() – Test koji proverava da li prilikom iznenadnog gašenja Writer klijentske konekcije, se izlazi iz funkcije handle_client() sa povratnom vrednošću 0.

Replicator Receiver komponenta:



Slika 4: Activity dijagram Replicator Receiver komponente.

Replicator Receiver komponenta prilikom pokretanja otvara serversku konekciju koja prihvata klijentsku konekciju Replicator Sender-a. Nakon što je prihvaćena klijentska konekcija Replicator

Sender-a, pokreće se nit za periodično slanje podataka iz bufera ka Reader komponenti. S tim u vezi program ima dve niti, main (glavnu) nit i nit periodičnog slanja poruka Reader komponenti. Glavna nit prima poruke od Replicator Sender komponente i skladišti ih u bafer. Nit periodičnog slanja podataka, spaja poruke trenutno zateknute u baferu u string. Zasebne poruke u string odvojene su „;“ znakom. Nakon spajanja poruka u jedan string, pokušava se slanje poruke Reader komponenti. Ako je to slanje uspešno poruke koje su poslate unutar tog jednog stringa biće izopštene iz bafera. Međutim ako je slanje neuspešno zbog nedostupnosti Reader servisa, pokušava se rekonekcija ponovnim pozivom setupClient() funkcije. U slučaju neuspešnog slanja poruke se ne izopštavaju iz bafera kako bi bile poslate kada rekonekcija bude bila uspešna. Poruka se periodično šalje svakih 5 sekundi.

Pregled funkcija:

setupServer() – Funkcija koja otvara servisnu konekcija i vraća takvu konekciju, na kojoj se očekuje klijentska konekcija Replicator Receiver-a.

setupClient() – Replicator Receiver je klijent u odnosu na Reader komponentu. Uz pomoć funkcije setupClient ostvaruje se klijentska konekcija ka Reader komponenti.

receiveSenderMessage(conn) – Funkcija koja prima poruke Replicator Sender komponente. Prosleđeni argument je klijentska serverska konekcija sa Replicator Sender komponentom sa kojom se vrši prijem.

makeDataString(list) – Funkcija od prosleđenog bafera primljenih poruka o potrošnji, pravi jedan string gde su te zasebne poruke odvojene „;“ karakterom. Ulazni argument funkcije je lista string poruka tj. bafer u kom su primljene poruke skladištene, dok je napravljen string i trenutni sadržaj bafera u trenutku spajanja stringova. Ovaj trenutni sadržaj bafera bitan je kao povratna vrednost kako bismo znali koje poruke da izopštimo iz bafera, nakon uspešnog slanja spojene poruke ka Reader komponenti.

deleteElements(lista1, lista2) – Funkcija koja uklanja elemente liste1 iz liste2.

sendToReader(cli, listEl) – Funkcija prvo vrši spajanja poruke primljenih u baferu uz pomoć makeDataString() funkcija, a zatim pokušava slanje te poruke. Nakon što je poruka poslata spojene poruke koje su poslate u okviru jednog stringa izopštavaju se iz bafera uz pomoć deleteElements() funkcije. Međutim, ako funkcije ne uspe da izvrši slanje zbog nedostupnosti Reader servisa, biće pokušana rekonekcije pozivom clientSetup() funkcije, i poruke neće biti izopštene iz bafera kako bi nakon rekonekcije sa Reader komponentom mogle biti Reader komponenti. Ulazni argumenti su klijentska konekcija ka Reader komponenti i bafer. Povratna vrednost je True ako je slanje bilo uspešno odnosno false ako je neuspešno.

periodicSend(listEl) – funkcija koja se vrši u zasebnoj niti, izvršava beskonačnu petlju pokušavajući izvršavanje sentToReder() funkcije periodično na svakih 5 sekundi.

test_ReplicatorReceiver.py

Za svrhe testiranja Replicator Receiver komponente napisano je osam testova. Sledi kratak opis:

test_setupClient() – Postoje tri testa za testiranje klijentske konekcija ka Replicator Receiver komponenti. Prvi test pokreće simulirani server u niti, i testira da li se connect() metoda uspešno izvršila nad tim simuliranim serverom, povratna vrednost ne bi smela biti None u tom slučaju. Drugi test proverava da li je povratna vrednost nula, ako ne postoji pokrenut server koji bi prihvatio konekciju. Poslednji test za testiranje klijentske konekcije proverava da li je connect() metoda pozvana sa korektnom IP adresom i portom.

test_setupServer() – je jedna test funkcija koja proverava da li je bind() metoda unutar setupServer() funkcije pozvana sa koretnom IP adresom i portom.

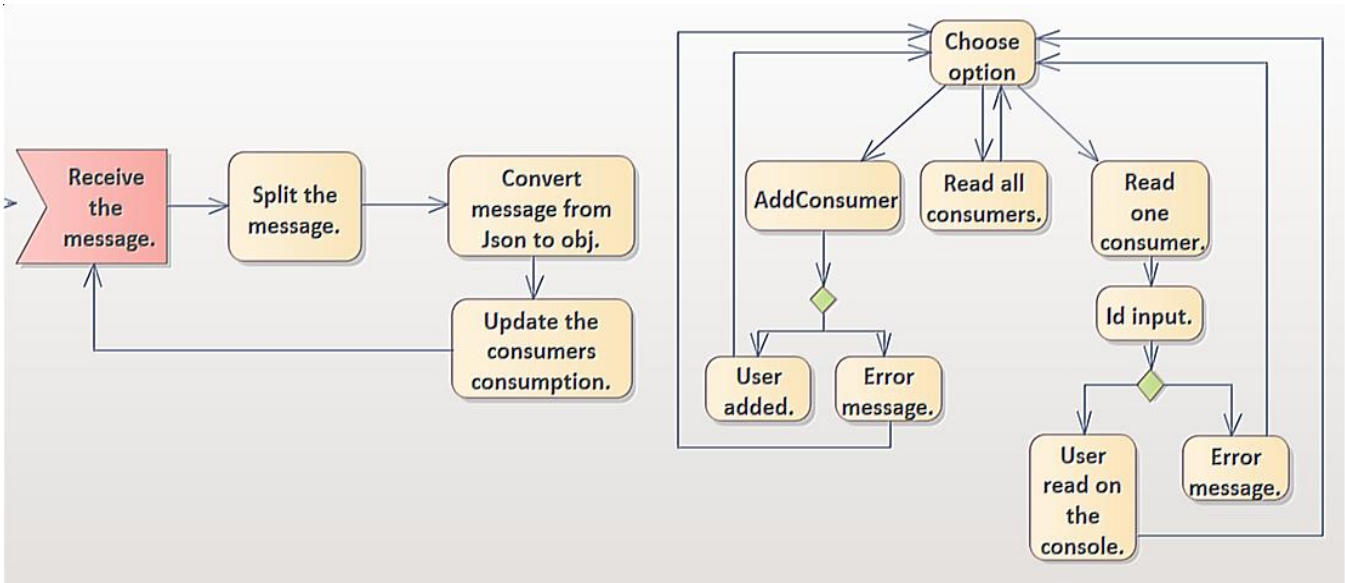
test_receiveWriterMessage() – Test koji proverava da li je poruka primljena uspešno uz pomoć receiveWriterMessage() funkcije. U zasebnoj niti se simulira klijentska konekcija koja šalje poruku. Očekivano ponašanje je da će poslata poruka simuliranim klijentim i primljena poruka receiveWriterMessage() funkcijom biti iste.

test_sendMessage() – Postoje dva testa za testiranje funkcije . Prvi test pokreće nit koja simulira Reader servis, klijent se konektuje na ovaj servis i šalje poruku sentToReader() funkcijom. Poslata poruka ovom funkcijom trebala bi biti ista kao i poruka primljena na simuliranom servisu.

Drugi test odnosi se na to da li je sendToReader() funkcija vratila False u slučaju da se pokuša slanje poruke kada simulirani Reader servis nije pokrenut. Tada sendToReader() funkcija neće uspeti poslati poruku i trebala bi vratiti False iz razlog što Reader servis nije dostupan.

test_makeDataString() – Test koji proverava da li su poruke korektno spojene u jedan string putem funkcije makeDataString(). Prosleđivanjem lista string primera porede se povratna vrednost makeDataString() funkcije i očekivan test primer string poruke.

Reader



*Slika 5:*Activity diagram Reader komponente

Reader komponenta služi za izvršavanje operacija(CRUD operacije)za dodavanje korisnika, pregled njihovih informacija i ažuriranje potrošnje koje pristižu od *Replicator Receiver* komponente. Podaci se ažuriraju u bazi podataka posredstvom *DbManagement* komponente koja komunicira sa SQL bazom podataka. Reader komponenta ima ulogu pružanja administratoru sistema dodavanje korisnika i iščitavanja korisnika, za koje potrebe je napravljen meni u konzoli. Sastoji se iz osam funkcija i jedne klase. Program se izvršava iz dve niti. Glavna(main) nit ima zadatak da vrši ažuriranje potrošnje na osnovu pristiglih poruka Replicator Receiver-a, dok se u drugoj niti izvršava meni.

Funkcije koje se tiču update niti *Reader* komponente:

- *setupServer()* – funkcija otvara serversku konekciju na zadatoj adresi i portu i vraća je kao povratnu vrednost. Kasnije se sa tom otvorenom konekcijom unutar main-a, očekuje klijentska konekcija *Replicator Receiver* komponente putem accept() naredbe.
- *reciveReciverMessage()* – Ulazni argument funkcije je socket na kojem je prihvaćena klijentska konekcija. Funkcija služi za prijem poruke od *Replicator Receiver* komponente. Povratna vrednost funkcije je poruka.
- *jsonToObj(message)* – Ulazna vrednost ova funkcije je string koji sadrži više JSON poruka o potrošnji korisnika koje su odvojeni tačka zarezom. Ovaj string predstavlja skup poruka dobijenih od Replicator Receiver-a nakupljenih tokom nekog perioda. Funkcija vrši splitovanje dobijenog stringa kako bi se dobile individualne JSON poruke o potrošnji. Zatim te poruke se iz

JSON stringa konvertuju u objekte consumerConsumption. Povratna vrednost funcije je lista takvih objekata.

U glavnoj niti se za svaki od konvertovanih objekata vrši ažuriranje potrošnje pozivanjem updateConsumer *DbManagement* komponente koja ažurira potrošnju u bazi pod.

Funkcije koje se tiču meni niti Reader komponente:

- *menuListing()* – Funkcija služi za izlistavanje menija. Unutar beskonačne petlje se poziva funkcija *choose()* za izlistavanje menija.
- *choose()* – Funkcija služi za izlistavanje menija, koji korisniku omogućava dodavanje novog potrošača, izlistavanje svih potrošača i izlistavanje potrošača sa zadatim ID-em. Povratna vrednost funkcije je jedan ukoliko je unos nije broj, ili 0 ukoliko je korisnik uneo nepostojecu opciju.
- *readAllCons()* – funkcija ispisuje sve potrošače unutar baze podataka. Funkcija poziva *readAllConsumers()* komponent DbManagement, koja vraća sve potrošače iz baze podataka.
- *readOneConsumer()* – funkcija ispisuje korisnika sa ID-em koji je korisnik uneo. Povratna vrednost je 0 ako korisnik nije uneo broj.
- *addConsumerTroughtConsole()* – funkcija omogućava korisniku unos podataka preko konzole. Korisnik unosi podatke o potrošaču koji se prosleđuju *DbManagement* komponenti koja vrši upis u bazu podataka. Povratna vrednost funkcije je 0 ukoliko potrošač već postoji ili 1 ukoliko je neispravan unos podataka od strane korisnika.

Unutar klase consumerConsumption se nalaze polja ID i cnspn koje predstavlja potrošnju.

Funkcije se pozivaju u okviru main-a. Funkcija za izlistavanje menija se pokrece u zasebnoj niti. Funkcije *reciveReciverMessage()* i *jsonToObj()* se pozivaju unutar beskonačne petlje. Funkcija jsonToObj() primljenu poruku pretvara u niz objekata consumerConsumption. Za svaki od tih objekata vrši se ažuriranje potrošnje pozivom updateConsumer() DbManagement komponente koja ažurira potrošnju potrošača unutar baze podataka

test_Reader

U okviru testiranja Reader komponente, vrši se sedam testova:

- *test_setupServer()* – Test koji proverava da li je funkcija bind serverskog socket-a pozvana sa potrebnom adresom i portom.
- *test_reciveReciverMessage()* – Test koji proverava da li je poruka primljena ispravno. Provera se vrši aktiviranjem niti koja simulira klijentsku konekciju i slanje podataka. Primeljena poruka mora biti jednaka poslatoj, da bi test bio uspešan.

- *test_Menu_1* – Test koji proverava da li funkcija *choose()* na osnovu validnog unosa vrši poziv funkcija iz menija na osnovu unosa.
- *test_addConsumerTroughtConsole()* – Test koji proverava da li je vraćen kod greške 0, ako potrošač sa unetim identifikacionim brojem već postoji unutar baze podataka.
- *test_addConsumerTroughtConsole_1()* – Test koji proverava da li je vraćen kod greške 1, ako su unete neispravne vrednosti prilikom dodavanja.
- *test_readAllCons()* – Test koji proverava da li je ispis na konzoli o potrošačima ispravan.
- *test_readOneConsumer()* – Test koji proverava da li je ispis na konzoli jednog potrošača ispravan.
- *test_readOneConsumer_1()* – Test koji proverava da li je povratni kod greške nula, prilikom unosa identifikacionog broja koji nije brojna vrednost.
- *test_jsonToObj()* – Test koji proverava da li je ispravno pretvoren string ispravno splitovan i zatim lista splitovanih JSON poruka pravilno konvertovana u objekte.

DbManagement (Database.py)

DbManagement služi za skladištenje i obradu podataka određenih potrošača. Baza podataka sadrži dve tabele – consumers_info(u kojoj se čuvaju podaci o potrošačima) i consumption_info(u kojoj se čuvaju podaci o potrošnji). Sastoji se iz sledećih funkcija:

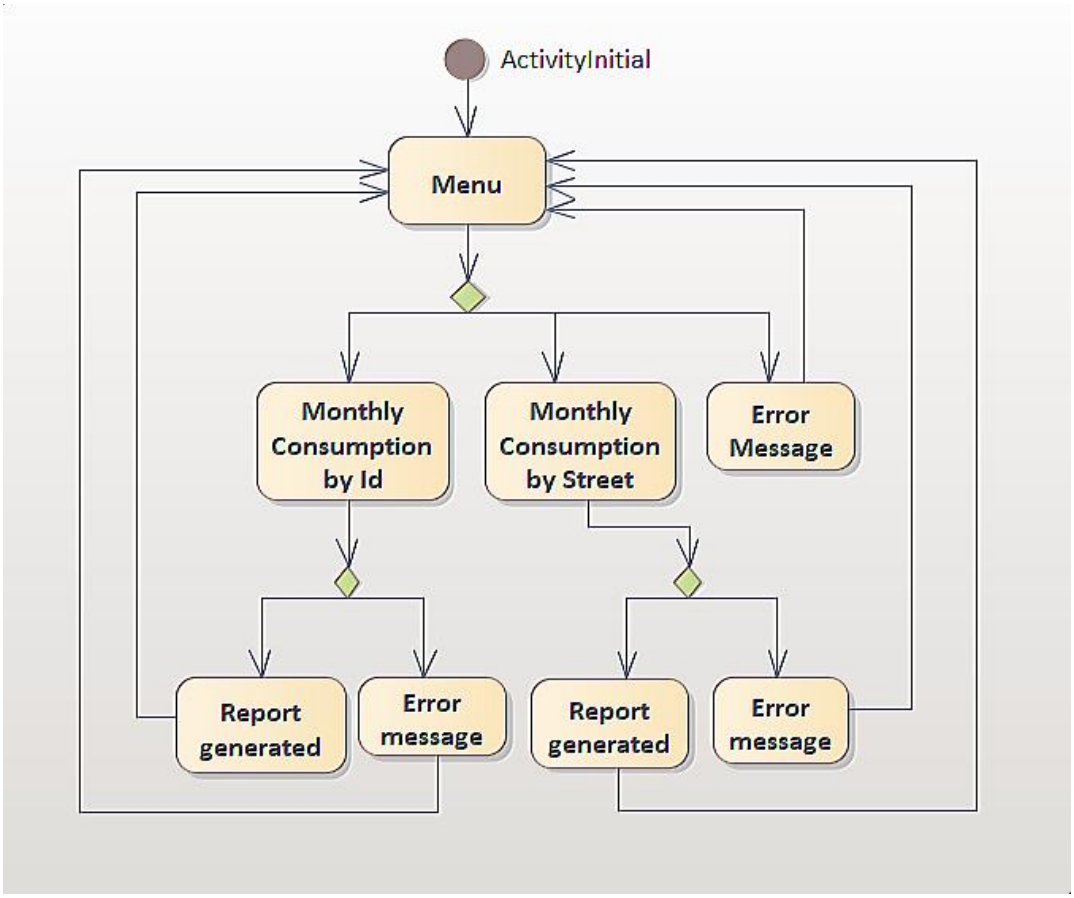
- *createTable()* – funkcija za kreiranje baze podataka i tabela u bazi podataka.
- *addConsumer()* – funkcija za dodavanje potrošača u tabelu consumers_info. Ulazni parametri ove funkcije su sve kolone koje se nalaze u tabeli(ID brojila, ime i prezime potrošača, ulica, broj, poštanski broj i grad). U slučaju da se pokuša dodavanje potrošača sa postojećim id-em, baca se IntegrityError.
- *readConsumersInfo(id)* – funkcija koja služi za čitanje podataka za potrošača sa zadatim ID-em koji se prosleđuje kao ulazni parametar. Povratna vrednost funkcije je string koji sadrži vrednosti svih polja iz tabele za određenog potrošača. U slučaju da potrošač sa datim identifikacionim brojem nije pronadjen u bazi podataka povratna vrednost će biti None.
- *readAllConsumers()* – funkcija koja pronalazi i izlistava informacije za sve potrošače koji se nalaze u bazi podataka. Povratna vrednost funkcije je lista svih potrošača. U slučaju da potrošač sa datim identifikacionim brojem nije pronadjen u bazi podataka povratna vrednost će biti None.
- *updateConsumer(id, cnspn)* – funkcija kao ulazne parametre prima ID brojila i potrošnju tog brojila. Ukoliko u tabeli postoji potrošač sa prosleđenim ID- em funkcija će sabrati prosleđenu potrošnju sa potrošnjom koja je već postojala, a ukoliko ne postoji korisnik sa zadatim ID-em u tabeli za potrošnju, funkcija će izvršiti dodavanje potrošača u tabelu sa potrošnjom za određeni mesec.

test_database.py

Napomena: Za potrebe testiranja DbManagement komponente, koristi se testna baza podataka. Zbog ovoga svaka od funkcija prethodno izloženih ima podrazumevani parametar db_name koji predstavlja bazu nad kojom će se funkcionalnost izvršiti. Podrazumevana vrednost parametra db_name je consumers.db koja je produkciona baza. Međutim, u test pozivima funkcija prosleđuje se testDB.db.

- *test_AddConsumer()* – Test koji proverava da li je potrošač uspešno dodat u bazu podataka.
- *test_ReadConsumersInfo()* – Test koji proverava da li je ispis potrošača u ispravnom obliku, i da li su očekivani potrošač zaista u bazi podataka.
- *test_ReadAllConsumers()* – Test koji proverava da li je ispis svih potrošača iz baze podataka u ispravnom obliku.
- *test_UpdateConsumer_1()*– Test koji proverava da li je update za određenog potrošača odrađen ispravno, u slučaju kada u tabeli potrošnje postoji podataka o potrošnji sa zadatim id-em. Ako u tabeli potrošnje postoji podatak o potrošnji sa zadatim id-em tada se na tu postojeću vrednost potrošnje dodaje tekuća.
- *test_UpdateConsumer_2()* – Test koji proverava da li je funkcija ispravno odradila ažuriranje tabele o potrošnji u slučaju kada ne postoji podatak o potrošnji sa prosleđenim id-em. Tada se u tabelu potrošnje dodaje podatak sa prosleđenim id-em i potrošnjom.
- *test_UpdateConsumer_3()* – Test koji proverava da li je kod greške nula, kada se pokuša dodati potrošnja sa id-em potrošača koji ne postoji u tabeli potrošača.
- *test_CreateTable()* – Proverava da li su tabele consumers_info i consumption_info uspešno kreirane i da li su kreirani pod ispravnim imenima tabela.

Reports



Slika 6: Activity diagram Reports komponente.

Reports komponenta služi za ispis izveštaja o potrošnji na osnovu podataka iz baze. Sastoji se iz tri funkcije:

- *monthlyStreetConsumption(street)* – Ulazni parametri su naziv ulice i default parametar sa imenom baze podataka koju funkcija koristi. Na osnovu naziva ulice, funkcija šalje upit u bazu, kako bi dobila podatke o potrošnji u svim mesecima za datu ulicu. Nakon dobavljanja podataka, funkcija ih sakuplja u zajedničku kolekciju(Dictionary), koju vraća kao povratnu vrednost, zajedno sa nazivom ulice.
- *monthlyConsumerConsumption(id)* – Ulazni parametri su id potrošača i default parametar sa imenom baze podataka koju funkcija koristi. Na osnovu id-a, funkcija šalje upit u bazu, kako bi dobila podatke o potrošnji u svim mesecima za datog potrošača. Nakon dobavljanja podataka, funkcija ih sakuplja u zajedničku kolekciju(Dictionary), koju vraća kao povratnu vrednost, zajedno sa id-om potrošača.

- *menu()* – Koristi se za formiranje načina interakcije sa korisnikom. Na početku ispisuje meni koji nudi moguće vrste izveštaja. Kada korisnik izabere odgovarajući meni, ispisuje mu se poruka o zahtevu unosa parametra pretrage(id potrošača ili naziv ulice, u skladu sa izabranim izveštajem). Nakon unošenja parametra, funkcija ispisuje izveštaj sa podacima o potrošnji vode podeljene po mesecima u godini. U slučaju nepravilnog unosa, ispisuje se poruka o grešci i korisnik se vraća na početni meni.

test_Reports

U sklopu testiranja *Reports* komponente, vrši se pet testova:

- *test_mothlyStreetConsumption1()* – Test koji proverava da li funkcija vraća kolekciju kao povratnu vrednost i da li vraćena kolekcija sadrži iste podatke o potrošnji zadate ulice kao i baza podataka.
- *test_mothlyStreetConsumption2()* – Test koji proverava da li funkcija vraća vrednosti (0, 0) ukoliko se prosledi nepostojeći naziv ulice.
- *test_monthlyConsumerConsumption1()* – Test koji proverava da li funkcija vraća kolekciju kao povratnu vrednost i da li vraćena kolekcija sadrži iste podatke o potrošnji zadatog potrošača kao i baza podataka.
- *test_monthlyConsumerConsumption2()* – Test koji proverava da li funkcija vraća vrednosti (0, 0) ukoliko se prosledi nepostojeći id potrošača.
- *test_menu()* – Test koji proverava da li funkcija *menu()* pravilno poziva druge funkcije i da li pravilno ulazi u sva uslovna grananja.