

# Dokumentacija za projekat "*P2-Replicator*"

## Sadržaj:

1. Osnovne informacije
2. Korišćene tehnologije
3. "*Component*" dijagram
4. Opšte karakteristike projekta (namena i upotreba)
5. Objašnjenje rada aplikacije (pokretanje)
6. Specifične klase i funkcije

## Osnovne informacije:

**Trajanje izrade projekta:** 28dana

**Mentor projekta:** Zorana Babić

**Učesnici u projektu:**

PR 7-2019 Ognjen Dačević

PR 14-2019 Nela Jović

PR 15-2019 Katarina Prodanović

PR 80-2019 Božidar Nešić

**Servisi za planiranje i distribuciju koda:** *"AzureDevOps"* za planiranje razvoja aplikacije, *"Github"* za distribuciju koda između članova tima

**Pristup razvoju aplikacije:** *"Scrum"*

## Korišćene tehnologije:

**Razvojno okruženje:** *"Visual studio code"*

**Programski jezik:** *"Python"*

**Alat za proveru kvaliteta koda:** *"SonarQube"*

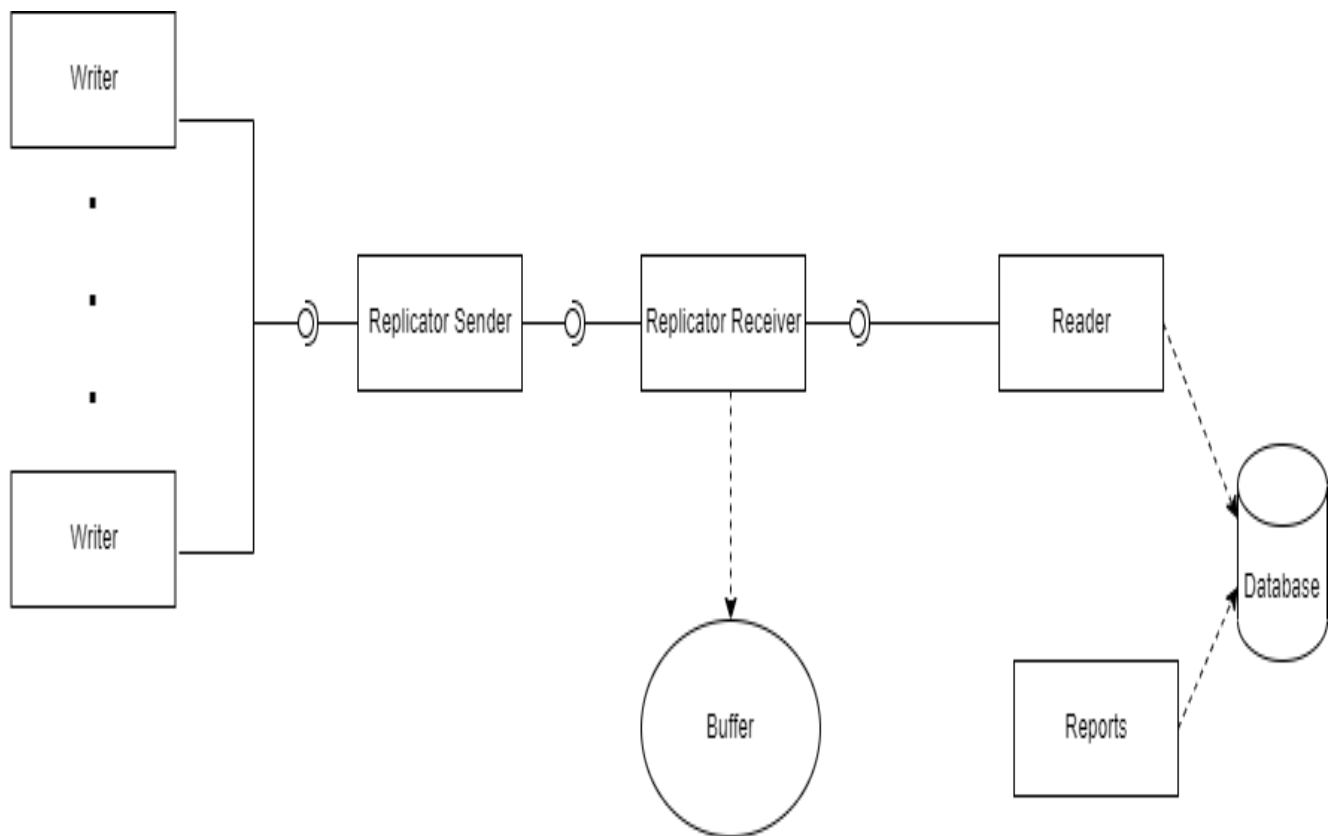
**Testiranje pokrivenosti koda:** *"python"* biblioteka *"coverage"*

**Tehnologije u kojima su pisani testovi:** *"python"* biblioteka *"unittest"* sa i bez *"mock"* objekata

**Komunikacija klijent-server:** *"socket"* biblioteka

**Konekcija na bazu podataka:** *"Oracle SQL developer"* alat

***“Component”*** diagram:



## Opšte karakteristike projekta (namena i upotreba):

*"P2-Replicator"* je aplikacija koja služi za komuniciranje klijenata sa serverom u cilju dobavljanja izveštaja o potrošnji vode na osnovu očitavanih podataka sa korisnikovog brojila.

Korisnika predstavlja komponenta *"Writer"* koja zahteva unos jedinstvenog obeležja korisnika (id korisnika), podatka o trenutnoj potrošnji vode, kao i unos podatka o mesecu u kom se potrošnja desila. Kako bi korisnik mogao da upisuje podatke, mora se prijaviti na sistem (podaci o korisnicima se izlistavaju u tekstualnoj datoteci i javno su dostupni zbog lakse prijave korisnika).

Podaci koje korisnik unese se dalje prosleđuju *"Replicator sender"* komponenti koja te podatke prepakuje i dalje šalje ka *"Replicator receiver"* komponenti.

*"Replicator receiver"* će te podatke slati ka *"Reader"* komponenti ali na tačno predefinisani interval. Drugim rečima, *"receiver"* će dobijene podatke skladištiti kod sebe lokalno u datoteku sve dok ne istekne neki period i tada će sve do tada skladištene podatke slati ka narednoj komponenti, a skladišteni podaci će biti obrisani i taj postupak se ponavlja u krug. Ako *"Reader"* komponenta nije pokrenuta, *"receiver"* će podatke čuvati kod sebe sve dok se *"Reader"* ne pokrene i tek tada će krenuti postupak sa intervalima.

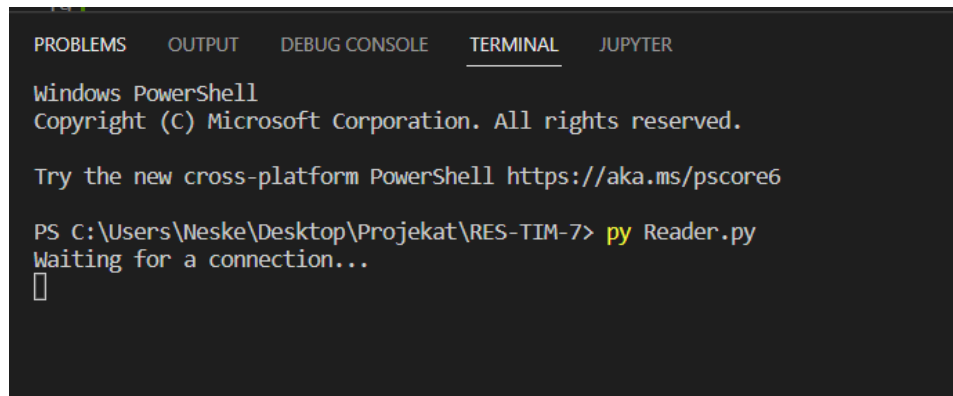
*"Reader"* komponenta dobijene podatke obrađuje i skladišti u relacionu bazu podataka i na taj način priprema podatke za izveštaj.

*"Reports"* komponenta je poseban deo sistema koji služi za iščitavanje podataka iz baze podataka preko *"SQL"* upita i generisanje izveštaja koji su korisniku od značaja.

## Obraštjenje rada aplikacije (pokretanje):

U “Visual studio code” je potrebno otvoriti minimalno 4 terminala kako bi aplikacija radila kako je zamišljeno.

**Terminal 1:** Komunikacija je zamišljena da ide preko “socket” biblioteke. U ovom slučaju komuniciraju “Replicator receiver” i “Reader” komponente sa tim da “receiver” šalje podatke, to jest zahteva neku uslugu pa je on klijent, a “reader” pruža tu uslugu korišćenjem baze podataka pa je on server. Zato se prvo pokreće server kako bi klijent znao sa kim treba da uspostavlja komunikaciju a ne obrnuto.



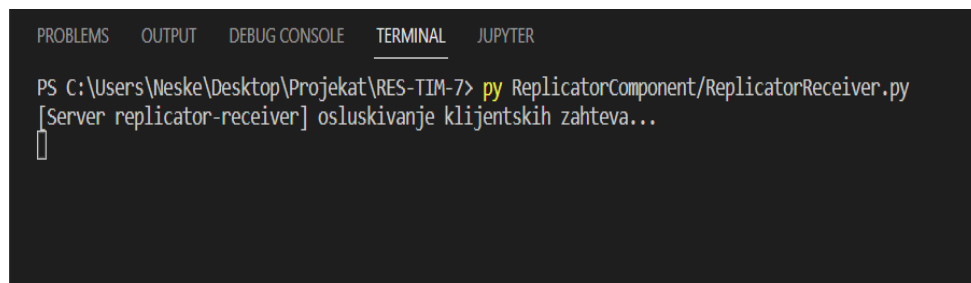
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Neske\Desktop\Projekat\RES-TIM-7> py Reader.py
Waiting for a connection...
█
```

Slika 1 – prikaz 1. terminala koji treba da se otvori

**Terminal 2:** Sada će se komunikacija izvoditi preko 3 komponente. “Replicator receiver” će primati zahteve od “Replicator sender” komponente u vidu poruka koje se prosleđuju, a “receiver” će slati zahteve ka “Reader” komponenti. Znači, “receiver” mora biti i klijent i server zavisno od funkcionalnosti koju pruža. Pokretanjem “receiver” komponente pokrenut je njegov server deo i čeka se neki klijentski zahtev da se poveže kako bi bio omogućen nastavak rada aplikacije.

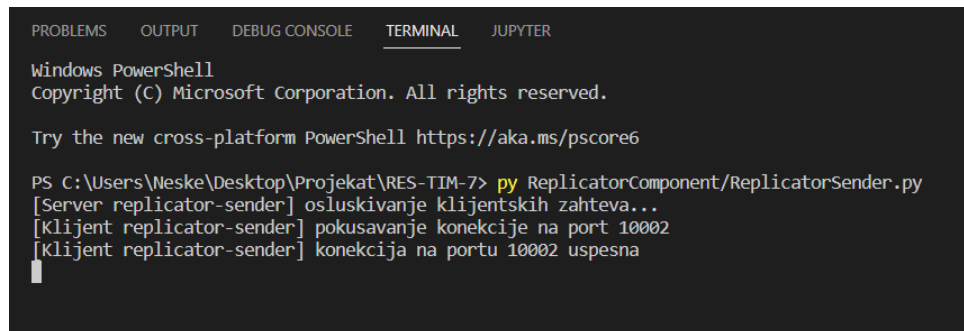


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

PS C:\Users\Neske\Desktop\Projekat\RES-TIM-7> py ReplicatorComponent/ReplicatorReceiver.py
[Server replicator-receiver] osluskivanje klijentskih zahteva...
█
```

Slika 2 – prikaz drugog. terminala

**Terminal 3:** Ovde se komunikacija takođe izvodi preko 3 komponente. “*Replicator sender*” komponenta će biti u ovom slučaju server “*Writer*” komponenti jer od nje prima podatke, a klijent “*Replicator receiver*” komponenti jer ka njoj šalje dobijene podatke.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

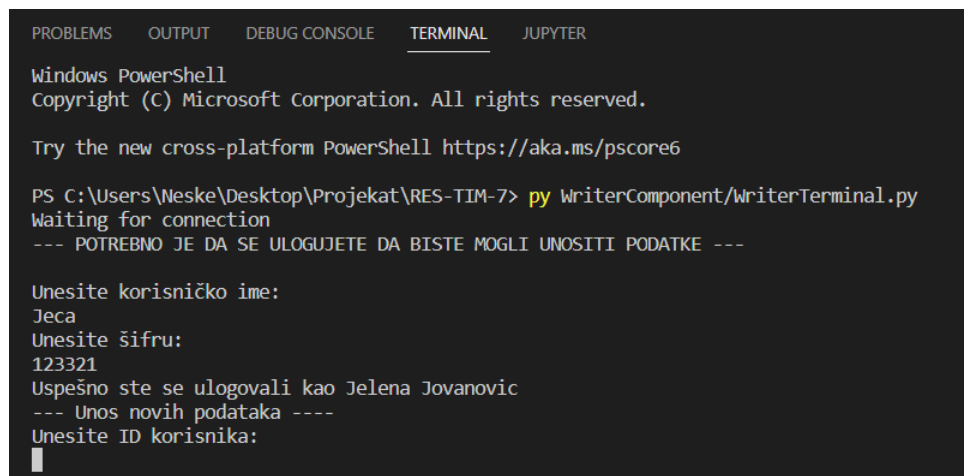
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Weske\Desktop\Projekat\RES-TIM-7> py ReplicatorComponent/ReplicatorSender.py
[Server replicator-sender] osluškivanje klijentskih zahteva...
[Klijent replicator-sender] pokušavanje konekcije na port 10002
[Klijent replicator-sender] konekcija na portu 10002 uspesna
```

Slika 3- prikaz trećeg pokrenutog terminala

**Terminal 4:** Za kraj se pokreću “*Writer*” komponente i njih može biti beskonačno mnogo. Da bi se pokrenuo novi “*Writer*”, to jest korisnik, dovoljno je pokrenuti dodatan terminal istog izgleda kao na Slici 4. “*Writer*” komponenta će biti samo klijent.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Neske\Desktop\Projekat\RES-TIM-7> py WriterComponent/WriterTerminal.py
Waiting for connection
--- POTREBNO JE DA SE ULOGUJETE DA BISTE MOGLI UNOSITI PODATKE ---

Unesite korisničko ime:
Jeca
Unesite šifru:
123321
Uspešno ste se ulogovali kao Jelena Jovanovic
--- Unos novih podataka ----
Unesite ID korisnika:
```

Slika 4 – prikaz poslednje komponente interakcije

**Dodatno:** Moguće je pokretanje dodatnog terminala radi korišćenja “*Reports*” komponente. Ona nema veze sa ostatkom sistema, već zavisi samo od informacija koje se nalaze u bazi podataka. Zbog toga, ukoliko se ova komponenta ne pokrene, to neće narušiti funkcionalnost ostatka sistema.

## Specifične klase i funkcije:

U ovom delu će biti opisana svaka funkcionalnost u samom projektu kako bi čitaoc dokumentacije mogao da razume implementaciju aplikacije.

### **“Writer” komponenta:**

- **logovanje()** - ova funkcija se nalazi u fajlu „*WriterTerminal.py*“ .Ona omogućava korisnicima da se uloguju pre početka unosa podataka. Dok god korisnik ne unese ispravno korisničko ime i šifru biće onemogućen nastavak rada sa aplikacijom. U njoj pozivamo i druge funkcije ove komponente kao što su:

- **otvori\_fajl(ime\_fajla)** - ova funkcija je napravljena kako bi pomogla priverifikaciji korisnika. Povratna vrednost je pokazivač na fajl.
- **splitovanje(red\_u\_fajlu)** - priprema parametara za proveru postojanja korisnika. Povratne vrednosti su ime, prezime, korisničko ime i šifra.

Povratna vrednost ove funkcije je ime i prezime ulogovanog korisnika.

- **unos\_podataka(ime\_korisnika,prezime\_korisnika)** - omogućava ulogovanom korisniku da unosi jedinstveno obeležje korisnika (id korisnika), podatka o trenutnoj potrošnji vode, kao i unos podatka o mesecu u kom se potrošnja desila. Povratna vrednost je objekat klase „*Message*“ . U okviru ove funkcije poziva se i funkcija *provera\_ispravnosti\_podataka(id\_korisnika,potrošnja,mesec)* koja proverava ispravnost podataka.

- **konekcija()** - ova funkcija služi kako bi omogućila komunikaciju i slanje podataka ka „*ReplictorSender*“ komponenti.

### **“Replicator” komponenta:**

- **obrada(poruka)** - nalazi se u datoteci “*ReplicatorComponent/ObradaPoruke.py*”. Kao parametar prima poruku koja je stigla sa “Writer” komponente na “Replicator sender” i prosleđena ka “Replicator receiver”. Iz poruke se uzimaju samo podaci od interesa i to id korisnika, potrošnja vode i mesec potrošnje. Kao povratna vrednost se vraća nova poruka koja se dalje prosleđuje.

- **konekcija\_klijent(broj\_porta, tip\_klijenta)** - nalazi se u datoteci “*ReplicatorComponent/ReplicatorKonekcija.py*”. tip\_klijenta je tekstualni parametar koji samo služi za ispis na terminal, a broj\_porta je port na kojem će klijent tražiti usluge servera. Ova funkcija pokušava da uspostavi konekciju klijenta sa serverom, i u slučaju uspešne konekcije vraća deskriptor klijentske utičnice i indikator uspešnosti (indikator = 0 ako je sve u redu, indikator = 1 u slučaju greške).

- **konekcija\_server(broj\_porta, tip\_servera)** - nalazi se u datoteci “*ReplicatorComponent/ReplicatorKonekcija.py*”. tip\_servera je tekstualni parametar koji samo služi za ispis na terminal, a broj\_porta je port na koji će se server pozicionirati (eng “*bind*”) i na kojem će slušati (eng “*listen*”). U slučaju uspešnog izvršavanja vraća se deskriptor serverske utičnice.

- **vise\_klijenata(connection, clientConnection, tipServera)** - nalazi se u datoteci "ReplicatorComponent/ReplicatorKonekcija.py". tip\_servera je tekstualni parametar koji samo služi za ispis na terminal, connection (serverska utičnica) i clientConnection (klijentska utičnica) predstavljaju deskriptore utičnica preko kojih se odvija komunikacija na "Replicator sender" komponenti. U slučaju uspešnog izvršavanja u beskonačnoj petlji se primaju i šalju podaci između komponenti. U slučaju nasilnog gašenja "Writer" komponente, javlja se "ConnectionResetError" i tada se izlazi iz petlje.

- **sender()** - nalazi se u datoteci "ReplicatorComponent/ReplicatorSender.py". "Replicator sender" je server koji sluša na definisanom portu i u beskonačnoj petlji čeka da mu se neki klijent obrati. Kada dođe do komunikacije, u novoj beskonačnoj petlji se šalju podaci sve dok se klijent, u ovom slučaju "Writer" komponenta, nasilno ne ugasi (gašenjem terminala).

#### "ReplicatorComponent/ReplicatorReceiver.py" funkcije:

- **upis\_u\_fajl(obrađena\_poruka, putanja)** - **obrađena\_poruka** je tekst koji će se upisivati na kraj tabele jer je tabela u ovoj funkciji otvorena u režimu "append", putanja je putanja do datoteke koja se otvara radi upisa u nju. Povratna vrednost ove funkcije će biti deskriptor datoteke koja se otvorila.

- **citanje\_iz\_fajla(putanja)** - **putanja** je putanja do datoteke koja se otvara u "read" režimu, što znači da se iz nje može samo čitati i ništa drugo (nema izmena). Povratna vrednost funkcije će biti lista iščitanih redova iz datoteke, dužina te liste (ovaj parametar se vraća zbog kasnijeg testiranja) kao i deskriptor otvorene datoteke.

- **obrada\_primljene\_datoteke(replikator\_poruka)** - **replikator\_poruka** parametar je tekstualna poruka koja je došla kao posledica korisničkog zahteva i ta poruka se transformiše radi izvlačenja korisnih podataka. U ovoj funkciji se samo poziva već opisana funkcija **obrada(poruka)**. Povratna vrednost će i ovde biti transformisana poruka.

- **prijem\_preko\_mreze(soket\_deskriptor)** - **soket\_deskriptor** predstavlja utičnicu koja će da prima podatke sa mreže pomoću funkcije **recv()**. Povratna vrednost će da bude primljena poruka koja je dekodirana pomoću funkcije **decode()**.

- **prijem\_upis\_fajl(inputs, putanja)** - **inputs** predstavlja niz utičnica na kojima će se primati poruke. Ovo se radi jer je utičnica koja ima operaciju čitanja sa mreže stavljena u neblokirajući režim kako bi se poruke slale ka "Reader" komponenti bez čekanja da klijent pošalje nešto što može i da se ne desi već se poruke šalju na predefinisani period vremena. Kako koja poruka stigne tako se upisuje u bafer (tekstualni fajl). Povratna vrednost funkcije je poruka koja je primljena sa klijenta.

- **slanje\_ka\_readeru(tekst\_fajl, reader\_klijent)** - **tekst\_fajl** predstavlja listu u kojoj je svaki element jedan red iz datoteke, a **reader\_klijent** je deskriptor utičnice koja šalje podatke ka "Reader" komponenti. Unutar funkcije se čita red po red i ako je server upaljen podaci se šalju ka njemu, u suprotnom se javlja greška i ispis "Reader server se ugasio". Povratna vrednost funkcije je promenljiva koja prikazuje trenutno stanje servera i može biti True ili False.



- **klijentska\_utičnica(replikator\_server, naziv\_komponente)** - **replikator\_server** predstavlja deskriptor serverske utičnice koja će prihvatati klijentske zahteve (u ovom slučaju *“Replicator receiver”* prihvata *“Replicator sender”* zahteve), a **naziv\_komponente** je tekstualni parametar koji samo služi za ispis na terminal. Funkcija ima namenu da kreirane utičnice usled prihvatanja klijentskih zahteva prevede u neblokirajući režim. Kao povratna vrednost se vraća niz utičnica koje su prebačene u neblokirajući režim.

- **receiver()** - nalazi se u datoteci *“ReplicatorComponent/ReplicatorReceiver.py”*. Funkcija se kreće u beskonačnoj petlji i prima podatke za upis u bafer (tekstualnu datoteku). Takođe računa se vreme koje je proteklo od poslednjeg prosleđivanja podataka iz bafera na *“Reader”* komponentu i kada to vreme istekne, sve što se nalazi u baferu se briše iz istog i prosleđuje ka *“Reader”* serveru. Ako server nije pokrenut, primeljeni podaci sa *“Replicator sender”* se čuvaju u baferu sve dok se server ne pokrene i tada kreće merenje vremena nakon kog se podaci šalju ka serveru.

#### ***“Reader”* komponenta:**




- **konekcija()** - nalazi se u *„Reader.py“* datoteci. Služi za uspostavljanje konekcije sa *„Replicator Receiver“* komponentom kako bi mogla da preuzima poruke koje joj ta komponenta šalje. Uokviru nje pozivamo i sledeće funkcije:

- **splitovanje\_parametara\_za\_bazu(poruka)** - priprema dobijene podatke zaunos u bazu.
- **upis\_u\_bazu(id,potrosnja,mesec,connection\_string)** - ova funkcija je namenjena za upis informacija u bazu. Pre pisanja SQL upita proveravamo ispravnost i postojanost podataka preko funkcije *provera\_podataka(id,mesec)* kako ne bi došlo do narušavanja konzistentnosti.




#### ***“Reports”* komponenta:**

- **izvestaj\_ulica(ulica)** - ova funkcija se nalazi u datoteci *„Izvestaji.py“* i služi da bi se dobio izveštaj o potrošnji za određenu ulicu. Kao parametar dobija naziv ulice za koju je potrebno napraviti izveštaj. Funkcioniše tako što se pravi *“SQL”* upit gde se pristupa tabelama i izvlače se informacije o ukupnoj količini potrošnje za prosledjenu ulicu za svaki mesec. Nakon

toga se pravi nova tekstualna datoteka sa jedinstvenim nazivom u koji se upisuju te informacije.

 IzvestajUlica1.txt IzvestajUlica2.txt IzvestajUlica3.txt

- **izvestaj\_brojilo(brojilo)** - ova funkcija se nalazi u datoteci „*Izvestaji.py*“ i služi da bi se dobio izveštaj o potrošnji za konkretno brojilo. Kao parametar dobija id brojila za koji je potrebno napraviti izveštaj. Funkcioniše tako što se pravi “*SQL*” upit gde se pristupa tabelama iizvlače se informacije o ukupnoj količini potrošnje za prosledjeni id brojila za svaki mesec. Nakon toga se pravi nova tekstualna datoteka sa jedinstvenim nazivom u koji se upisuju te informacije.

 IzvestajBrojilo1.txt IzvestajBrojilo2.txt IzvestajBrojilo3.txt

