

Dokumentacija za projektni zadatak

Load Balancer

Sistem Load Balancer je kreiran tako da `system_handler` upravlja sa 4 glavne komponente:

1. **Writer**
2. **Worker**
3. **Load Balancer**
4. **Database (CRUD)**

`System_handler` ima 2 rečnika u kojima se nalaze do sada kreirani `Writer`-i i lista funkcija koje mogu da se odaberu u glavnom meniju.

Koristili smo biblioteku `PyInquirer2`. Ona predstavlja kolekciju interaktivnih komandi koje omogućavaju efikasniji user interface. Ove komande su realizovane u okviru terminala.

Kada se pokrene `system_handler` koristi metodu `UserPrompt` koja nudi osnovni meni sa listom opcija u terminal uz pomoć metode `prompt` iz `PyInquirer2` biblioteke. `Prompt` kreira UI kroz koji se možemo kretati strelicama i `enter` za potvrdu. `Prompt` nam kao povratnu vrednost šalje rečnik u kojem se nalazi izbor korisnika sa kojim nam rečnik `functions` poziva odgovarajuću metodu čiji se ključ poklapa sa izborom korisnika.

Writer

Metoda `ManageWriters` nudi novu listu sa sledećim opcijama "...", "Create writers" i "Destroy writers". Opcija "..." nas vraća na početni meni, "Create writers" poziva `writer_factory`. Tu kreiramo onoliko `Writer`-a koliko je korisnik zahtevao. Metodi je potrebno proslediti koliko `Writer`-a želimo, listu ključeva do sada kreiranih `Writer`-a koji su smešteni u rečnik i sam rečnik u koji ćemo smestiti nove `Writer`-e. Broj `Writer`-a koji unosemo se mora kasti u `int` jer nam `prompt` unetu vrednost uvek vraća kao rečnik sa tipom vrednosti `string`. Ključevi su njihov redni broj koji im fabrika dodeljuje, uvek kreće od najmanjeg

slobodnog prirodnog broja. Rečnik Writer-a sadrži sve kreirane Writer-e koji su u stanju da rukuju sa Worker-ima.

“Destroy writers” nam prikazuje postojeće Writer-e, sa checkbox poljima koje obeležavamo ako želimo da uništimo odgovarajući Writer. Koristimo ObjectParser da proverimo koji checkbox je označen i vratimo ključ njegovog Writer-a. Sa vraćenim ključevima brišemo Writer-e iz liste.

Na main listi imamo opciju “Write” koja poziva metodu sa istim imenom gde možemo birati koji Writer želimo da koristimo. Kada odaberemo želejnog Writer-a dobijemo novi meni za rukovanje sa Worker-om.

Prvo se poziva Execute metoda koja čisti prethodni prikaz uz pomoć ExecuteCleanMethod sa referencom na UserPrompt. On nam pravi meni sa novim izborom kako Writer može da upravlja sa Worker-ima.

Worker

ShowWorkerStatuses metoda pokreće novu paralelnu nit za metodu ViewWorkers koja ispiše i konstanto osvežava stanje Worker-a (10 puta u sekundi), ali se na ekranu ispis osveži samo kad dodje do neke izmene. Pokrece se nova nit koja izvršava metodu ViewWorkers da bi se omogućio real-time prikaz izmene stanja Worker-a. Nit ce biti zaustavljena kada korisnik pritisne Enter.

Metoda SendData je za slanje podatka putem Load Blancera, gde on uzima prvog slobodnog Worker-a, zatim se uzimaju novi podaci od korisnika uz pomoć metode InputDataEntry. Prvo se bira merač, zatim se unosi vrednost koja mora da se validira. Potrebno je unetu vrednost kastovati u int veći od nule. Poslednji podatak koji se unosi je mesec za koji je izmerena vrednost, njega korisnik bira iz unapred zadate liste stringova. Sa svim tim podacima pravimo novi DataPoint koji vraćamo da bi smo ga prosledili Load Balancer-u metodom RecieveData. Ako je Load Balancer odlučio da primi podatke sa RecieveData i buffer je napunjen, menja se status Worker-a i može se videti da on radi, ekran se dinamički osvežava. Dokle god se podaci smeštaju u buffer, da nije napunjen, samo javlja poruku da je podatak poslat.

Ručno slanje podataka(ManuallySendData) radi potpuno isto, osim što se korisniku dobavlja lista svih slobodnih Worker-a, pa on bira kojeg želi da uposli.

Novog Worker-a kreiramo uz pomoć WorkerFactory, potrebno je uneti koliko Worker-a želimo. Fabrika dodeljuje id za svakog novog Worker-a. Svaki kreirani Worker se unosi u rečnik Load Balancer-a, gde se nalaze svi radnici koji su spremni da obradjuju podatke.

Brisanje Worker-a je potpuno isto kao i brisanje Writer-a.

Izborom "Manage Workers" u meniju pozivamo metodu ChangeWorkersStates koja koristi checkbox dugmad da korisnik obeleži koje Worker-e želi da uključi ili isključi (metodama TurnOn i TurnOff iz worker.py). UpdateWorkerStates prolazi kroz listu označenih Workera i menja im stanje. Svi Worker-i su isključeni pri kreiranju.

Glavna metoda u fajlu worker.py je ProcessWorkerAction. Ona pokrece svaku metodu iz rečnika actions u novoj niti da bi se proces izvršavao u pozadini. Za kreiranje nove niti je potrebno proslediti referencu na metodu i argumente koji će trebati pri pozivu metode.

AddMeterConsumptions dobija listu potrošnji i za svaku potrošnju poziva ProcessMeterConsumption. Tu se pokreće StateChange, a on menja stanje Worker-a da je zauzet i osvežava stanje zauzetosti Worker-a. Zatim uspavamo nit na 2 do 5 sekundi, da bi se simuliralo da se obavlja neki posao, onda se bazi prosledjuje potrošnja i na kraju ponovo menjamo stanje Worker-a, sada više nije zauzet.

Proces iz ProcessMeterConsumption se ponavlja u metodama ProcessMeterAdd, ProcessMeterUpdate i ProcessMeterDelete, osim što se poziva druga metoda za bazu.

Od metoda imamo za preuzimanje podataka iz baze za:

- Ključeve merača u rečniku (GetMetersKeys metoda)
- Sve merače (GetAllMeters)

I prikaz svih vrednosti uz pomoć PyInquirer2 biblioteke za:

- Gradove (SelectCity)
- Merače (SelectMeter)

Load Balancer

Load Balancer je komponenta koja služi za ravnomerno raspoređivanje posla. Posao raspoređuje Workerima. Load Balancer prima podatke od Writer komponente (RecieveData) i prosleđuje ih nekom od slobodnih Workera na obradu (GetWorker). Load Balancer privremeno smešta podatke kod sebe (u buffer) i nakon prikupljenih 10 vrednosti te podatke prosledjuje prvom slobodnom worker (ProcessData).

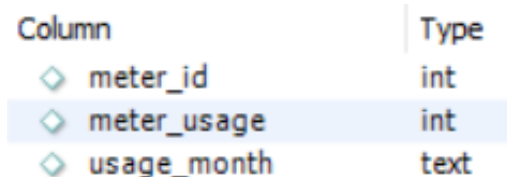
GetAllAvailableWorkers dobavlja sve Workere koji su aktivni i nisu zauzeti.

GetAvailableWorker dobavlja samo jednog Worker-a da obradi podatke.

Database

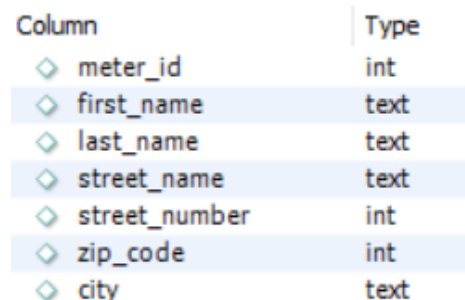
Za rad sa bazom podataka koristili smo tehnologije: MySQL, MySQL Server, MySQL Workbench, konektor za python, mysql biblioteku.

Baza podataka sadrži 2 tabele, potrošnju na meračima(slika 1) i sam merač sa svim njegovim podacima(slika 2).



Column	Type
meter_id	int
meter_usage	int
usage_month	text

Slika 1



Column	Type
meter_id	int
first_name	text
last_name	text
street_name	text
street_number	int
zip_code	int
city	text

Slika 2

Konekcija ka bazi se vrši preko statičke metode GetConnection koja se nalazi u connection.py.

Svi upiti ka bazi se nalaze u sql.py u constants folderu.

Upravljanje bazom možemo sa više mesta, preko Writer-a da uposlimo Worker-a za unos potrošnje na meraču i preko system_handler-a, možemo izvršavati CRUD funkcionalnost (ManageDatabase) i da dobavimo analitiku baze podataka(ReportsPrompt).

ManageDatabase nam nudi meni sa opcijama koje pozivaju metode AddMeter, ModifyMeter i DeleteMeters.

AddMeter koristi prompt za svaku vrednost koju je potrebno uneti za kreiranje novog merača. Podaci se vraćaju u rečniku, a svaki input se preko ključa dobavlja. Prvi slobodni Worker se angaržuje da upiše podatke u bazu.

ModifyMeter izvršava Update funkcionalnost. Zapošljava jednog Worker-a da izlista sve merače gde korisnik bira kojeg želi da modifikuje (SelectMeters metoda). Zatim korisnik mora da popuni sva polja za taj merač, kao kada kreira novog, osim id polja po kojem se identifikuje taj merač. ParseUpdateMeter menja polja od prosledjenog merača.

DeleteMeters upošljava Worker-a za ispis svih merača, gde preko checkbox-a korisnik bira koje merače želi da obriše. Kad se iz prompt-a vrate podaci uposleni Worker poziva svoju Delete metodu, koja ide do baze i briše merače.

ReportsPrompt nam nudi 2 opcije za ispis podataka:

1. City report
2. Meter report

City report poziva ProvideCityPerMonthReport metodu koja zapošljava Worker-a da dovuče iz baze potrošnju na meračima po mesecima za određeni grad koristeći sql upite.

Mater report poziva ProvideMeterPerMonthReport koja izvlači iz baze potrošnju po mesecima na određenom meraču koristeći sql upite.