

**SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN**

Leo Leljak

SIMULACIJA SUSTAVA NAJMA ELEKTRIČNIH VOZILA

PROJEKT

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ź D I N

Leo Leljak

Matični broj: 0016122912

Studij: Informacijsko i programsko inženjerstvo

SIMULACIJA SUSTAVA NAJMA ELEKTRIČNIH VOZILA

PROJEKT

Mentor/Mentorica:

Dr. sc. Bogdan Okreša Đurić

Varaždin, travanj 2022.

Leo Lejak

Izjava o izvornosti

Izjavljujem da je moj projekt izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

U ovome radu biti će implementiran sustav najma električnih vozila. Sustav će se sastojati od tri agenta. Prvi agent je centralni sustav koji upravlja prijavom neke stanice u sustav najma vozila, te se takav agent može više puta pokrenuti ukoliko nam je potrebno više različitih stanica. Drugi agent je sama stanica koja će se sastojati od dva različita ponašanja, te upravlja zahtjevima pojedinih klijenata koji žele iznajmiti vozilo. U radu ćemo proći detaljnije kroz samu strukturu pojedinih agenata kao i način njihove implementacije. Posljednji agent je krajnji korisnik koji unajmljuje vozila, te ih koristi. Agent krajnjeg korisnika biti će kreiran kao konačni automat te ćemo također kroz njegovu implementaciju proći detaljnije u radu.

U samom radu proći ćemo kroz arhitekturu rješenja kao i detalje implementacije pojedinog agenta i njegovih ponašanja. Rezultat je skalabilno rješenje koje se može lako proširiti dodatnim agentima pošto se stvara decentralizirani sustav najma vozila.

Ključne riječi: agent, simulacija, električna vozila, komunikacija, python

Sadržaj

1. Uvod	1
2. Metode i tehnike rada	2
3. Opis sustava i pregled stanja agenata	3
3.1. Višeagentni sustavi	3
3.2. Komunikacija između agenata	5
3.3. Sustav najma električnih vozila	6
3.4. Decentralizirani pristup	7
3.5. Rent Central Agent	9
3.6. Rent Station Agent	10
3.7. Rent User Agent	13
4. Implementacija rada	16
4.1. Rent Central Agent	16
4.2. Rent Station Agent	17
4.3. Rent User Agent	22
5. Prikaz rada aplikacije	25
6. Zaključak	29
7. Popis literature	30
Popis literature	30
Popis slika	31
1. Prilozi	32

1. Uvod

Simulacija sustava iznajmljivanja električnih vozila može biti izvedena na više različitih načina. U ovome radu koristimo pojednostavljeni pristup u kojemu se ne dotičemo procesa planiranja i zakazivanja pojedinih narudžba u određeno vrijeme pošto to nije u opsegu ovoga rada. Glavni je fokus na kreiranju različitih ponašanja koji su potrebni kod agenata koje implementiramo. Također koriste se različite vrste ponašanja - ciklično, periodično te konačni automat. U ovome radu prikazana je unakrsna komunikacija između klijenata kao i decentralizirani pristup kreiranja novih agenata u sustav iznajmljivanja električnih vozila. Na taj način sustav je skalabilniji te se može lakše nadodati novi agent, a da svi ostali agenti (pojedinačne postaje) znaju za novog agenta. Postoji jedan centralni agent koji zaprima nove stanice koje se žele uključiti u posao iznajmljivanja, no nakon toga svaka je stanica neovisna o centralnom sustavu što se tiče poslovanja i iznajmljivanja.

Električna vozila zbog svoje zelene tehnologije koju koriste sve su poželjnija vrsta prijevoznog sredstva kojeg krajnji korisnik želi unajmiti. Unutar rada koristi se 4 vrste električnih automobila i 3 vrste električnih bicikala, svaki sa svojim svojstvima kao što su maksimalni doomet, dnevna cijena najma, trenutna napunjenost baterije i ostalo. Kasnije će se detaljnije proći kroz sam opis svojstva i njihovu važnost unutar cjelokupnog sustava najma električnih vozila. Lista vozila proširiva je no za potrebu ovog rada smanjena je na 8 vozila ukupno kako bi fokus bio na arhitekturi rješenja kao i samoj komunikaciji između agenata.

2. Metode i tehnike rada

Agenti su implementirani u programskom jeziku Python korištenjem biblioteke "Smart Python Agent Development Environment" (SPADE) . SPADE je idealna biblioteka za implementiranje ovoga rješenja, a može se koristiti sa Pythonom 3.6, te biblioteka koristi asyncio i striktno slijedi principe PEP8 i Clean Code. [1]

Tijekom vježbi na kolegiju Višeagentni sustavi korištene su iste tehnologije, te slične tehnike rada. Na vježbama smo se mogli upoznati sa raznim ponašanjima koji se mogu koristiti i implementirati. Gotovo svako od tih ponašanja iskorišteno je u ovome radu pošto je simulacija iznajmljivanja električnih vozila idealan kandidat da bi se unutar strukture rješenja implementirale razne vrste ponašanja agenta. Tako je su u ovom radu korištena ponašanja kao što su ciklično, periodično i ponašanje konačnog automata.

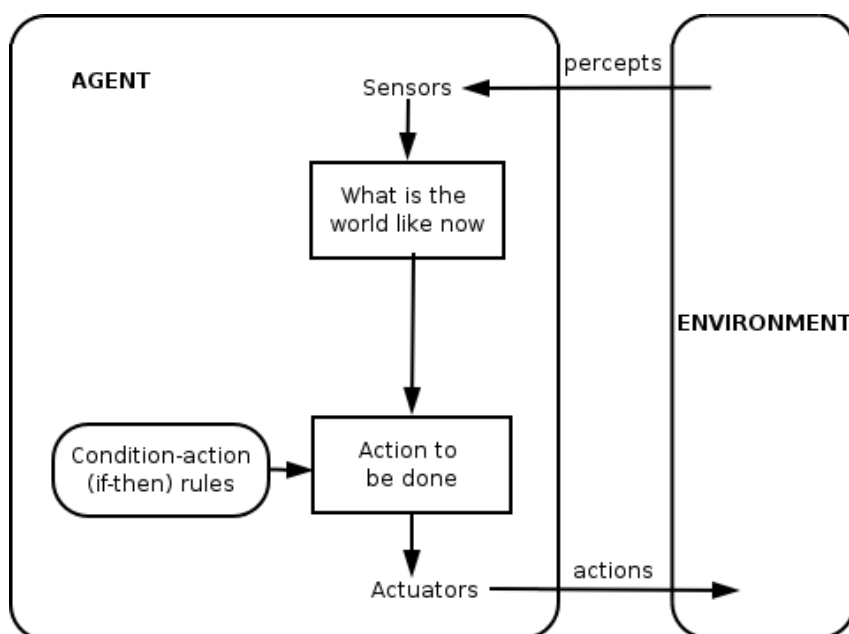
Korištene su paradigme objektnog programiranja. Kod je iskoristiv te za pokretanje više različitih klijenata koristi se parsiranje ulaznih argumenata kako bi se id i lozinka agenta mogla definirati prilikom pokretanja agenata. Na taj način moguće je pokrenuti više različitih agenata svaki sa svojim korisničkim podacima. Neki dijelovi koda su dokumentirani kako bi se čitaoc mogao lakše snalaziti prilikom pregleda izvornog koda sustava iznajmljivanja električnih vozila.

3. Opis sustava i pregled stanja agenata

U ovom dijelu opisan je način rada sustava za najam električnih vozila kao i interakcija između agenata. Proći će se kroz teorijsku podlogu višeagentnih sustava te će se nakon toga prijeći na shematski prikaz ponašanja koje agenti imaju. Sustav se sastoji od centralnog agenta, agenta stanica te pojedinačnih klijenata koji mogu izvršiti narudžbu. Klijent je napravljen preko ponašanja konačnog automata pa će i određene naglasak biti na izvedbi toga dijela.

3.1. Višeagentni sustavi

Prema [2] paradigma višegentnih agenata vezana je uz autonomne, reaktivne entitete koje mogu djelovati individualno i izvršavati specifične zadatke koje su mu dane. Agenti su odlični izvršioци zadataka za koje su namijenjeni te mogu samostalno određivati načine na koji dolaze do ciljeva. Ti načini mogu biti implementirani kao složen algoritam koji agent slijedi ili može postojati određena baza znanja na temelju koje agent odlučuje kako stići do cilja.

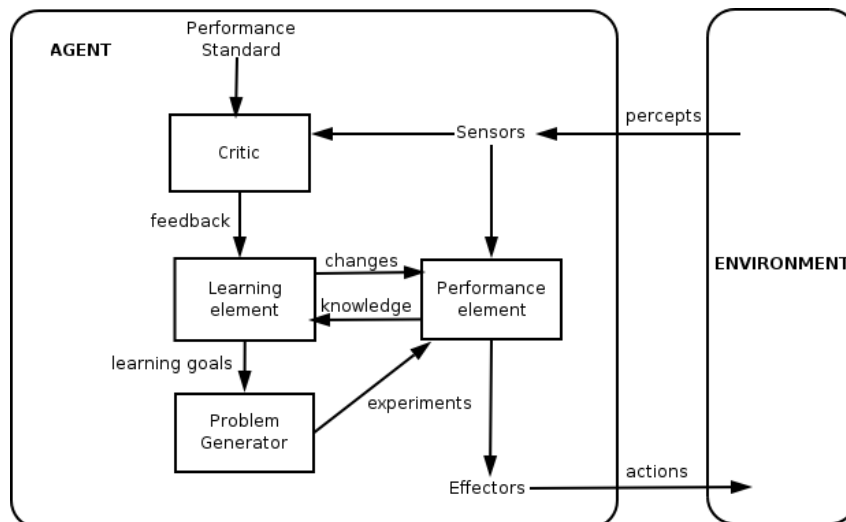


Slika 1: Dijagram jednostavnog reaktivnog agenta (Izvor: Wikipedia, 2022)

Na slici 1 može se vidjeti shematski primjer jednog agenta koji ima interakciju sa svojom okolinom. Agent reagira na promjene okoline preko senzora te postoji zadan set if-then pravila prema kojima on reagira i vrši određene zadatke. Ovdje se radi o primjeru jednostavnog reaktivnog agenta.

Na slici 2 vidimo shematski prikaz agenta učenja (eng. Learning agent). Radi se o agentu koji nije samo izvršioц zadataka prema nekom jasnom setu if-then pravila već ima određene ciljeve učenja te pri svakoj interakciji se stvara novi zapis u bazu znanja kako bi već pri sljedećoj interakciji mogao bolje odreagirati na promjene.

Agenti imaju nekoliko mogućih fleksibilnosti - reaktivni, proaktivni te društveni. Reak-



Slika 2: Dijagram jednostavnog agenta učenja (Izvor: Wikipedia, 2022)

tivni agenti su oni koji ostvaruju kontinuiranu interakciju sa svojom okolinom te pravovremeno reaguju na promjene koje se događaju unutar takve okoline. Proaktivni agenti su agenti koji su pogonjeni ciljevni, tj. Nisu pogonjeni isključivo događajima u okolini. Posljedna moguća fleksibilnost agenta je društvenost koja je sposobna interakcije agenta s drugim agentima putem nekog jezika za komunikaciju.

Često se stavljaju višeagentni sustavi u kontekst umjetne inteligencije te isto tako i ekspertnih sustava. Kod višeagentnih sustava u odnosu na ekspertne sustave razlika je u tome da ekspertni sustavi sadrže informacije, dok agenti nisu stvoreni za pohranjivanje informacija nego izvršavanje nekih konkretnih zadataka. Umjetna inteligencija najčešće ima intenciju oponašati razne situacije koje se nalaze u stvarnom svijetu, dok su agenti tu da izvršavaju neke jednostavnije operacije za koje su namijenjeni.

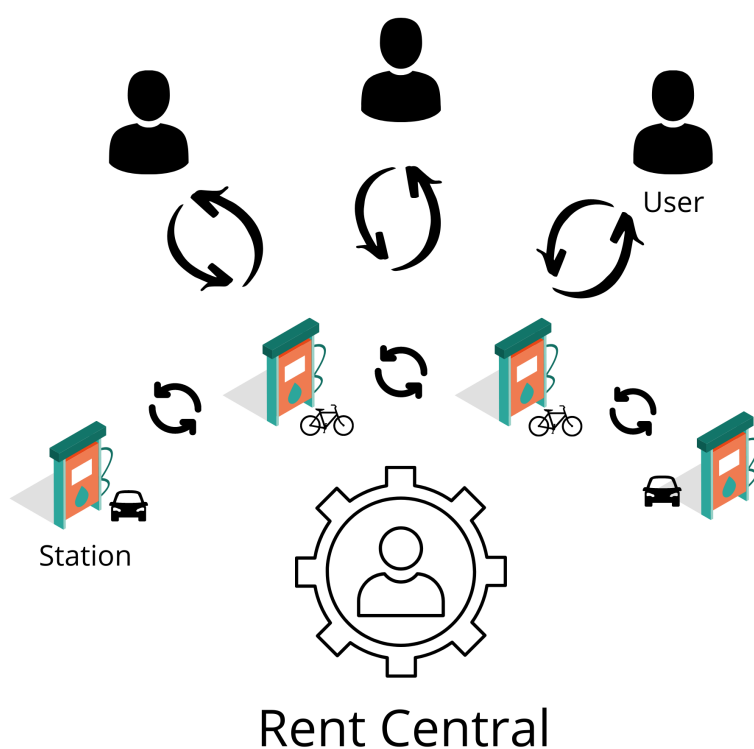
3.2. Komunikacija između agenata

Općenito je komunikacija između agenata ključna kod razvoja bilo kojeg višeagentnog sustava. Za ovaj projekt koristi se komunikacija prema FIPA ACL standardu koji se također koristio tijekom vježbi na kolegiju Višeagentni sustavi. Na taj način svi agenti međusobno mogu komunicirati preko jasnih poruka, te je komunikacija jednostavna. U ovom projektu svi agenti moći će zaprimiti poruku kao i poslati istu prema drugom agentu.

3.3. Sustav najma električnih vozila

U ovome poglavlju prikazan je predloženi sustav najma električnih vozila. Sustav je pojednostavljen kako bi u fokusu bila komunikacija između agenata, njihova interakcija te pojedina ponašanja agenata. Sustav se sastoji od tri glavna agenta:

- Rent Central
- Rent Station
- User



Slika 3: Dijagram sustava agenata (Izvor: U vlastitoj izvedbi, 2022)

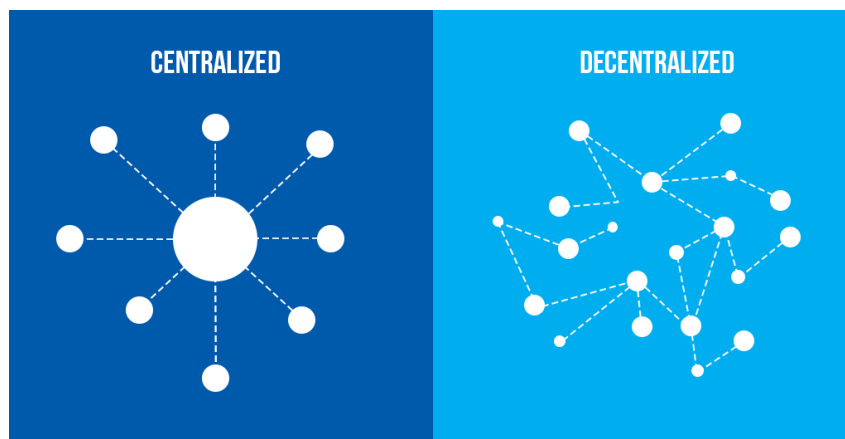
Unutar sustava postoje tri ključna agenta. Prvi agent Rent Central potrebno je pokrenuti samo jedamput i njegova je namjena da bude glavna tvrtka u koju se mogu učlaniti ostali Rent Stationi. Glavna centrala zadužena je za zaprimanje zahtjeva od Rent Stationa kod početne prijave samih stanica. Kada se prijavi nova stanica u sustav stanica svoj se stanici distribuira poruka o skupu svih trenutno prijavljenih i aktivnih stanica u sustav.

Na taj način se pokušava decentralizirati sustav. Alternativno tome pristupu bio bi potpuno centralizirani sustav a više o tome u sljedećem potpoglavlju. Krajnji korisnik kada želi unajmiti određeno električno vozilo (bicikl, automobil) može to učini direktno zahtjevom prema željenoj stanici. Ukoliko željena stanica nema vozilo koje korisnik želi unajmiti, ili je vozilo trenutno na punjenju tada stanica uzima neku drugu stanicu iz liste svih trenutno dostupnih stanica

i pruža tu informaciju korisniku. U tom slučaju korisnik je obavješten porukom na koju je stanicu upućen.

3.4. Decentralizirani pristup

Postoje više različitih pristupa izradi ovakvog sustava. U ovome radu korištena je varijanta decentraliziranog pristupa gdje svaka stanica zna za svaku stanicu i može međusobno komunicirati sa njom. Glavna centrala (Rent Central) zadužena je za mogućnost prijave posebnih stanica (Rent Station-a) u sustav najma električnih vozila. Taj dio je ostao centraliziran zbog razloga što mora postojati centralni sustav koji nadzire rad ostalih stanica. Također ukoliko bi to bio sustav koji radi na principu franšize, svaka stanica se mora pojedinačno prijaviti u takav sustav. Kada se pojedinaca stanica prijavi kod Rent Centrala, centralna stanica javlja svim ostalim stanicama koje su trenutno prijavljene trenutni skup svih prijavljenih stanica. Svaka stanica zaprimi trenutni skup svih prijavljenih stanica te ažurira svoje podatke o njima. Na taj način svaka stanica zna za sve stanice koje su trenutno prijavljene u njejoj okolini.



Slika 4: Dijagram sustava agenata (Izvor: aceinfoway.com/blog/centralized-vs-decentralized-apps, 2022)

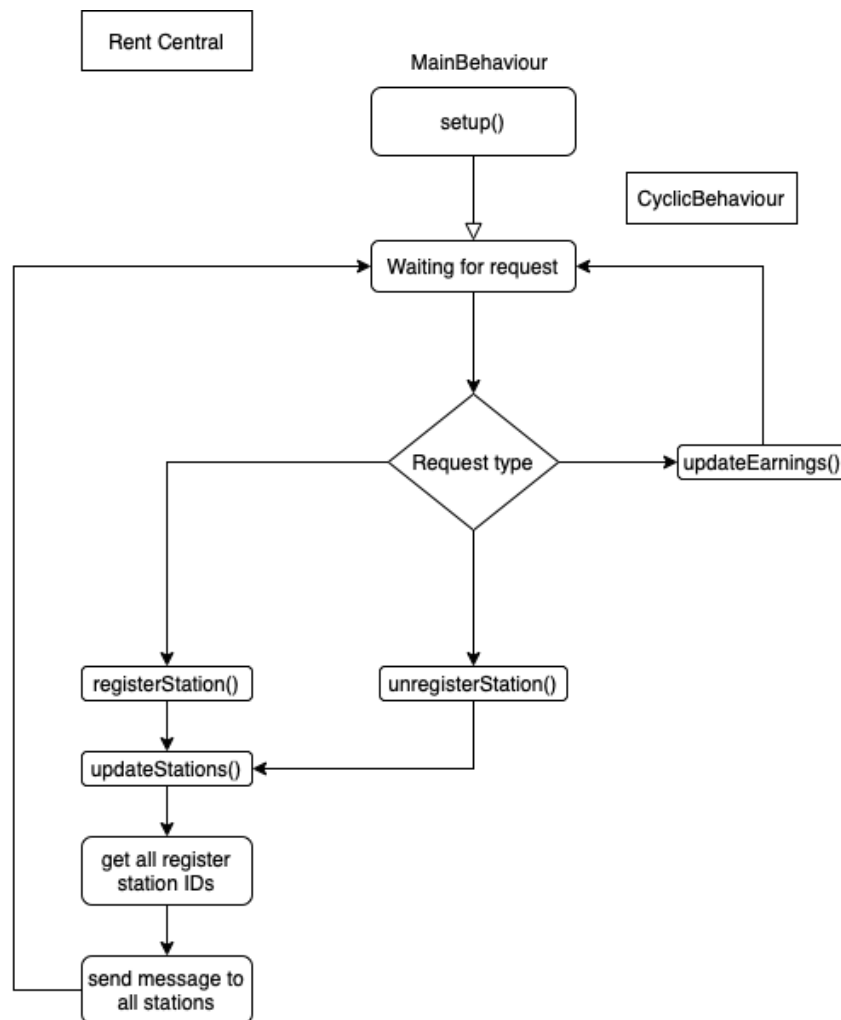
Alternativan pristup odabranome bio bi potpuno centralizirani sustav gdje bi Rent Central jedini znao za sve stanice koje su trenutno prijavljene u sustav. Na taj način bi svaki zahtjev nekoj drugoj stanici preko izvorne stanice morao ići preko centralnog sustava. Na taj način bi se stvorilo veće opterećenje na centralni sustav pošto bi svi zahtjevi išli preko njega. Decentraliziranim pristupom također eliminiramo jednu točku kvara (eng. Single point of failure). Stanice nisu ovisne o jednoj centralnoj stanici te u slučaju da dođe do kvara na centralnoj stanici, svaka stanica može nezavisno djelovati.

3.5. Rent Central Agent

Centralni agent jedini je dio koji je na neki način centraliziran zbog svoje zadaće koje izvršava. Na ovog agenta može se gledati kao na tvrtku koja prodaju svoja prava na rentanje električnih vozila drugim stanicama.

Agent ima samo jedno ponašanje koje je ciklično. Ciklično ponašanje je poput petlje koja se ponavlja. Postoje tri glavne zadaće, odnosno zahtjeva koje ovo ponašanje može primiti:

- Register station
- Unregister station
- Update earnings



Slika 5: Dijagram agenta RentCentral (Izvor: U vlastitoj izvedbi, 2022)

Register station zahtjev šalju stanice kada se žele registrirati u sustav i dobiti informacije o svim drugim stanicama koje se trenutno nalaze u sustavu iznajmljivanja električnih vozila. Kada je takav zahtjev zaprimljen tada se u listu sprema id novoregistrirane stanice te se šalje poruka svim trenutno registriranim stanicama sa trenutnom listom registriranih stanica. Na taj

način svaka stanica je obavještena o trenutno registriranim stanicama te svaka stanica zna za svaku stanicu i može izravno komunicirati sa njom preko jedinstvenog id-a stanice.

Unregister station radi suprotnu stvar od sestrinske register station opcije. Naime sa ovim zahtjevom stanica šalje zahtjev za deregistracijom te se iz liste briše takva stanica i ponovno se šalje svim stanicama ažurirana lista svih aktivnih stanica.

Update earnings zahtjev šalju stanice u određenim intervalima. To je definirano ponašanje unutar agenta Rent Stationa kroz koje će se proći u detaljima vezanim uz strukturu agenta Rent Station.

3.6. Rent Station Agent

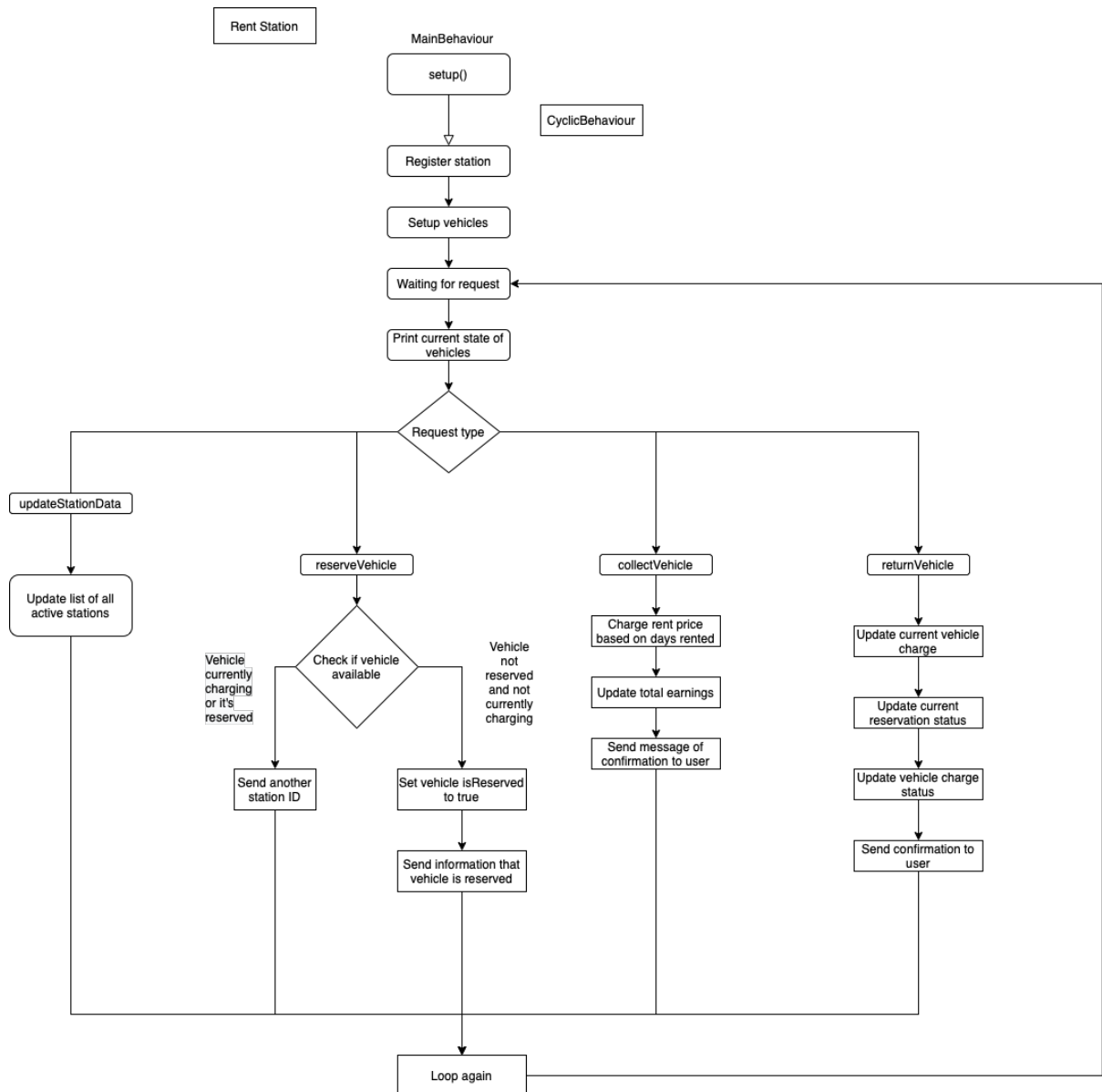
Rent Station agent sastoji se od dva ponašanja koja su različita po svojim svojstvima. Prvo ponašanje je ciklično ponašanje koje se ponavlja u petlju, te mu je glavna uloga čekanje zahtjeva i registriranje na iste. Drugo ponašanje koje ovaj agent ima je periodično ponašanje, a njegova ključna uloga je izvršavanje periodičnog punjenja baterija električnih vozila. U određenim intervalima takvo ponašanje provjerava trenutno stanje vraćenih vozila te za ona vozila kojima je potrebno punjenje takvo punjenje izvršava.

Glavno ciklično ponašanje sastoji se od ukupno 4 zahtjeva koje može obraditi:

- Update station data
- Reserve vehicle
- Collect vehicle
- Return vehicle

Na slici NAPIŠI BROJ SLIKE može se vidjeti dijagram prvog ponašanja koje ovaj agent ima. Takvo ponašanje je ciklično, a na samom početku vrše se određene pred radnje. Prijetimo se da se ovdje radi o stanici za najama električnih vozila koja se na samom početku nakon pokretanja mora registrirati u sustav najama električnih vozila. Ta registracija vrši se komunikacijom sa Central Rent agentom koji je opisan u prethodnom dijelu.

Pred radnje ovog agenta također uključuju, osim same registracije kod centralnog agenta, i inicijalno podešavanje dostupnih vozila. Dostupna vozila uključuju električne automobile i bicikle. Za potrebe ovog projekta takva lista je sužena na 3 električnih automobila, te 3 automobila. Također zbog pojednostavljena u sklopu ovog projekta svaka novo registrirana stanica ima isti set električnih automobila i električnih bicikala. Njih, kao i pripadne im karakteristike, može se pronaći u tablici 1 i tablici 2.



Slika 6: Dijagram agenta RentStation (Izvor: U vlastitoj izvedbi, 2022)

Za potrebe ove simulacije i pojednostavljanje računanja, za parametre u tablicama nisu definirane mjerne jedinice. Sam način računanja prikazan je u implementaciji koda.

Nakon što se obave podešavanja vozila prelazi se na čekanje zahtjeva. Posljedica zaprimanja novog zahtjeva ispis je trenutnog stanja svih vozila u voznom parku stanice.

Update station data zahtjev stiže od Rent Central agenta nakon što se nova stanica prijavi u sustav najma električnih vozila. Ovim zahtjevom obavještava se stanica sa ažuriranom listom svih aktivnih stanica u sustavu. Nakon zaprimljenog zahtjeva stanica ažurira svoju trenutnu listu.

Reserve vehicle zahtjev pristiže od Rent User agenta koji želi rezervirati određeno vozilo. Stanica provjerava da li je to vozilo dostupno. Dostupnost vozila ovisi o tome je li već prethodno rezervirano i je li trenutno na punjenju ili ne. Ukoliko vozilo nije na punjenju i nije trenutno rezervirano tada se takvo vozilo može unajmiti. U tom slučaju stanica šalje korisniku

informaciju o tome da je vozilo uspješno rezervirano. Ukoliko vozilo nije dostupno šalje se sljedeći ID stanice kako bi korisnik mogao poslati zahtjev prema sljedećoj stanici.

Collect vehicle zahtjev pristiže od Rent User agenta koji je došao pokupiti svoje rezervirano vozilo. U ovome trenutku unaprijed se naplaćuje iznos najma električnog vozila ovisno o broju dana koje je korisnik odabrao. Također se ažurira trenutni iznos ukupnih primanja za stanicu. Nakon što su obavljenе ove akcije šalje se potvrda korisniku da je proces gotov, odnosno da korisnik može početi koristiti vozilo.

Return vehicle zahtjev pristiže od Rent User agenta i odnosi se na vraćanje vozila nakon isteka najma. Stanica ažurira trenutnu informaciju vozila o stanju napunjenosti baterije. Nakon toga ažurira se trenutno stanje vozila po pitanju rezervacije i statusu punjenja. Na kraju se šalje potvrda korisniku.

Periodično ponašanje punjenja vozila ponavlja se u jednakim intervalima, te se odnosi na radnje vezane uz punjenje vozila koje su vraćena i koja trenutno nemaju potpuno napunjenu bateriju. Na početku svakog ciklusa ovog ponašanja ispisuje se trenutno stanje vozila i njegovu napunjenost baterijskog sustava. Nakon toga prolazi se kroz listu svih vozila i provjerava se da li je potrebno punjenje bilo kojeg vozila. Ukoliko je potrebno vrši se punjenje u nekom proizvoljnom vremenu. To vrijeme biti će definirano u samoj implementaciji agenata. Nakon što su sva vozila napunjena takvo će ponašanje biti izvršeno ponovno u zadanom intervalu.

Automobil	Domet	Vrijeme punjenja	Cijena (po danu)
Nissan	300	100	15
Mazda	250	50	12
Mercede	400	150	20

Slika 7: Tablica 1: Karakteristike električnih automobila (Izvor: U vlastitoj izvedbi, 2022)

Bicikl	Domet	Vrijeme punjenja	Cijena (po danu)
Trek	100	50	5
Greyp	150	30	7
KTM	200	20	10

Slika 8: Tablica 1: Karakteristike električnih bicikla (Izvor: U vlastitoj izvedbi, 2022)

3.7. Rent User Agent

Agent je definiran kao konačni automat stanja zbog svoje idealne predispozicije. Agent prolazi kroz niz stanja koji su definirani te na posljétku prekida svoj životni vijek. Definirana stanja agenta su:

- RESERVE_VEHICLE
- COLLECT_VEHICLE
- USE_VEHICLE
- CHARGE_VEHICLE
- RETURN_VEHICLE

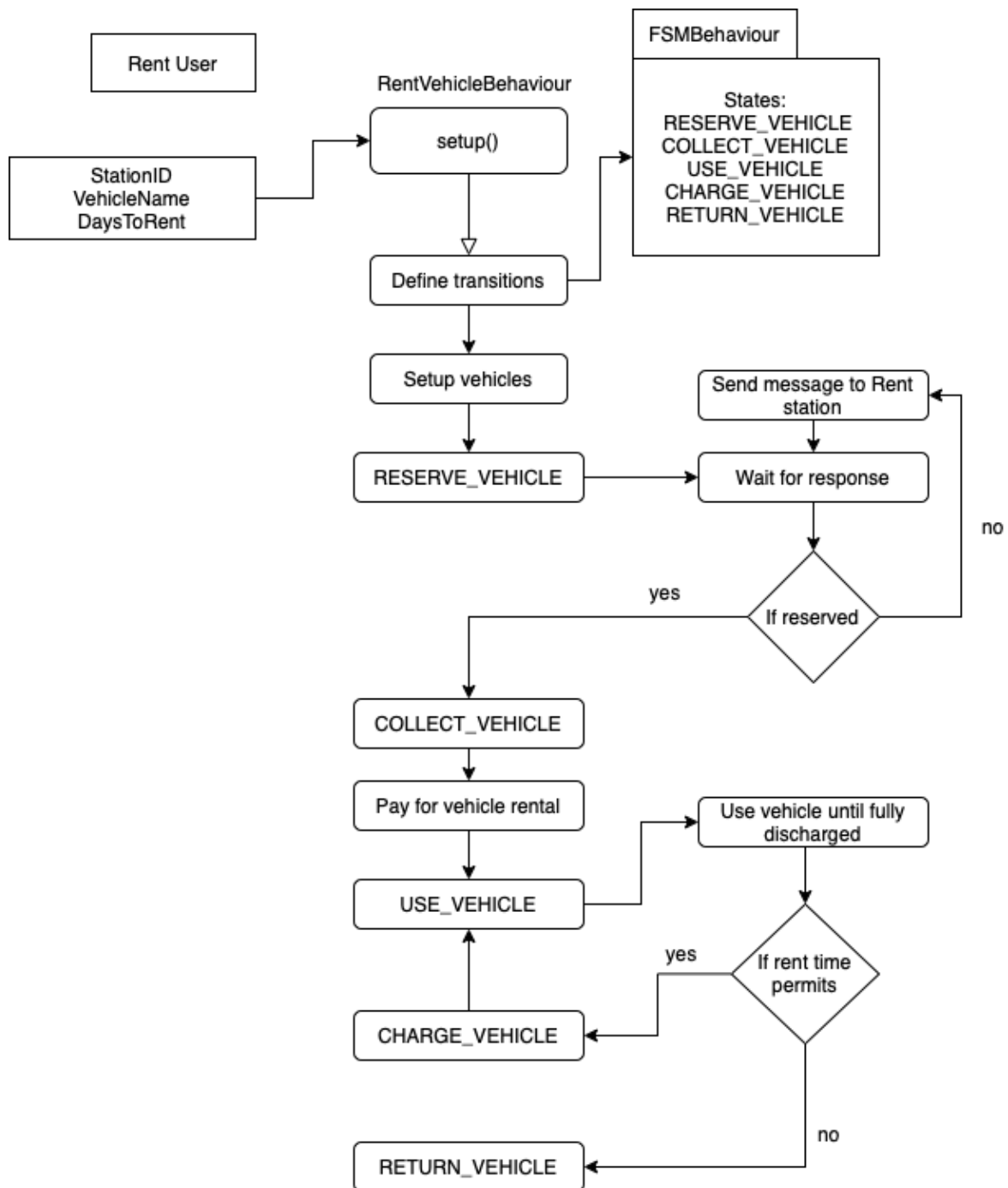
Nakon početne konfiguracije definiraju se tranzicije u kojima se automat može nalaziti. Te su tranzicije navedene u tablici 3.

Stanje	Moguće tranzicije
RESERVE_VEHICLE	COLLECT_VEHICLE
COLLECT_VEHICLE	USE_VEHICLE
USE_VEHICLE	CHARGE_VEHICLE, RETURN_VEHICLE
CHARGE_VEHICLE	USE_VEHICLE
RETURN_VEHICLE	

Slika 9: Tablica stanja i prijalaza RentUser agenta (Izvor: U vlastitoj izvedbi, 2022)

Sve počinje sa RESERVE_VEHICLE stanjem u kojem korisnik šalje poruku željenoj stanici da želi rezervirati određeno vozilo na određen broj dana. Ulazni parametri definirani su prilikom pokretanja samog agenta. Nakon toga čeka se na odgovor, a ako je odgovor afirmativan tada se prelazi u stanje COLLECT_VEHICLE. Ako odgovor nije afirmativan mijenja se željena stanica rezervacije i ponovno se prelazi u slanje poruke novoj stanici.

Nakon što se uspješno rezervira vozilo prelazi se u stanje COLLECT_VEHICLE gdje korisnik dolazi do određene stanice i preuzima vozilo. U tom trenutku korisnik plaća iznos najamnine vozila. Zatim se prelazi u stanje USE_VEHICLE gdje korisnik koristi svoje iznajmljeno vozilo sve dok ga ne isprazni. Kada ga isprazni provjerava se da li je moguće i dalje koristiti vozilo, odnosno da li vrijeme koje je ostalo za korištenje vozila je dovoljno da se vozilo napuni i isprazni do kraja. Ukoliko to vrijeme dozvoljava tada prelazimo u stanje CHARGE_VEHICLE. Ukoliko to vrijeme ne dozvoljava tada se vozilo mora vratiti u stanicu odnosno u stanje RETURN_VEHICLE. Kod CHARGE_VEHICLE stanja vozilo se puni do svoje maksimalne razine te se tada ponovno prelazi u stanje USE_VEHICLE. Kada se prijede u stanje RETURN_VEHICLE tada se javlja stanici da želimo vratiti vozilo, te kada se vozilo vrati završava životni ciklus ovog konačnog automata.



Slika 10: Tablica 1: Dijagram ponašanja RentUser agenta (Izvor: U vlastitoj izvedbi, 2022)

4. Implementacija rada

U ovom dijelu proći će se kroz implementaciju pojedinog agenta u samom kodu. Biti će izdvojeni ključni dijelovi, odnosno ključne zadaće koje izvršavaju pojedini agenti i njihova ponašanja. U prošlom poglavlju detaljno smo prošli kroz dijagrame i izvedbu agenata i njegovih ponašanja te je pojašnjena u detalje njihova uloga i međusobno komunikacija. Izdvojeni su dijelovi koda koja vrše ključne zadaće u agentima.

4.1. Rent Central Agent

Kroz dijagrame ponašanja ovog agenta prošli smo u prošlom poglavlju. Ovdje ćemo proći kroz implementaciju koda za najvažnije dijelove ovog agenta. Ovaj agent ima jedno glavno ponašanje koje je ciklično te služi zaprimanju i obradi zahtjeva od ostalih stanica.

MainBehaviour - implementacija čekanja zahtjeva

```
class MainBehaviour(spade.behaviour.CyclicBehaviour):
    async def on_start(self):
        print("I'm starting central behaviour")
        print("-----")

    async def run(self):
        print("Current stations:", self.agent.stations)
        print("-----")
        print("Waiting for requests by stations...")

        msg = await self.receive(timeout=100)
        if msg:
            if msg.body == 'registerStation':
                self.agent.stations.append(msg.sender)
                await self.updateStations()
            if msg.body == 'unregisterStation':
                await self.unregisterStation(msg.sender.localpart)
```

Ovdje je najveći naglasak na dva zahtjeva koje obrađuje ovaj agent, a to su register i unregisterStation. Nakon zaprimanja zahtjeva registerStation() tada se poziva funkcija updateStations().

updateStations()

```
async def updateStations(self):
    allStationIDs = []
    for station in self.agent.stations:
        allStationIDs.append(station.localpart + "@" + station.domain)

    for station in self.agent.stations:
        receiver = station.localpart + "@" + station.domain
```

```

msg = Message(to=receiver)
msg.set_metadata("performative", "inform")
msg.set_metadata("ontology", "updateStationData")
msg.body = ",".join(allStationIDs)
await self.send(msg)

```

U ovom asinkronoj funkciji nakon registracije novo prijavljenje stanice šaljemo informaciju o trenutno svim registriranim stanicama pojedinačno svakoj stanici koja je trenutno aktivna.

unregisterStation()

```

async def unregisterStation(self, stationID):
    for station in self.agent.stations:
        if station.localpart == stationID:
            self.agent.stations.remove(station)
    await self.updateStations()
    print("Unregistered_station")

```

Ova funkcija koristi se kada neka stanica želi izaći iz sustava trenutno aktivnih stanica te se odjavljuje, tj. deregistrira iz sustava. Također poziva se funkcija updateStations() kako bi se trenutni set aktivnih stanica ažurirao kod svake stanice posebno.

4.2. Rent Station Agent

Ovaj agent ima dva ponašanja kroz koja smo prošli u prošlom poglavlju. Glavno ponašanje koje je ciklično te služi zaprimanju i obradi zahtjeva od pojedinačnih klijenata, te također periodično ponašanje koje služi za periodično punjenje baterija električnih vozila.

MainBehaviour - on_start

```

print("I'm_starting_main_behaviour")
print("-----")
msg = Message(to="l1eljak@rec.foi.hr")
msg.set_metadata("performative", "inform")
msg.set_metadata("ontology", "myOntology")
msg.body = "registerStation"
await self.send(msg)

```

MainBehaviour - on_end

```

print("Ending_agent")
msg = Message(to="l1eljak@rec.foi.hr")
msg.set_metadata("performative", "inform")
msg.set_metadata("ontology", "myOntology")
msg.body = "unregisterStation"
await self.send(msg)

```

Na početku glavnog ponašanja radimo registraciju stanice u sustav svih aktivnih stanica. Ta poruka šalje se prema Rent Central agentu. Kod završetka ponašanja agenta šalje se unregisterStation poruka koja označava da se ta stanica želi odjaviti iz sustava najma električnih vozila. Takva se poruka također šalje centralnom agentu.

MainBehaviour - postavljanje vozila

Tijekom inicijalnog postavljanja agenta, također se postavljaju i vozila. Za potrebe ovog projekta korišten je suženi set vozila svaki sa svojim karakteristikama.

```
class Vehicle:
    def __init__(self, name, maxDistance, chargeTime, pricePerDay):
        self.name = name
        self.maxDistance = maxDistance
        self.charge = 100
        self.chargeTime = chargeTime
        self.pricePerDay = pricePerDay
        self.isReserved = False
        self.isCharging = False

    def printProperties(self):
        print(self.name, "-", self.charge, "-", self.isReserved)

    def printChargeProperties(self):
        print(self.name, "-", self.charge, "-", self.isCharging)

def setupCars(self):
    return [
        Vehicle("nissan", 300, 100, 15),
        Vehicle("mazda", 250, 50, 12),
        Vehicle("mercedes", 400, 150, 20)
    ]

def setupBikes(self):
    return [
        Vehicle("trek", 100, 50, 5),
        Vehicle("greyp", 150, 30, 7),
        Vehicle("ktm", 200, 20, 10)
    ]
```

MainBehaviour - run

Ključna uloga glavnog ponašanja definirano je u run funkciji. Ovo ponašanje zaduženo je za obradu zahtjeva koji pristižu prema RentStation agentu.

```
async def run(self):
    print("-----_RENTING_SYSTEM_BEHAVIOUR_-----")
    self.printCurrentState()
```

```

print("Waiting_for_requests_by_users_...")
print("")

### Handle requests
msg = await self.receive(timeout=100)
if msg and msg.metadata:
    if msg.metadata["ontology"] == 'updateStationData':
        self.agent.stations = msg.body.split(",")
    if msg.metadata["ontology"] == 'reserveVehicle':
        vehicleName = msg.body
        await self.reserveVehicle(vehicleName, msg.sender)
    if msg.metadata["ontology"] == 'collectVehicle':
        vehicleName = msg.body.split(",")[0]
        days = msg.body.split(",")[1]
        await self.collectVehicle(vehicleName, int(days), msg.sender
        )
    if msg.metadata["ontology"] == 'returnVehicle':
        vehicleName = msg.body.split(",")[0]
        chargeLeft = msg.body.split(",")[1]
        await self.returnVehicle(vehicleName, chargeLeft, msg.sender
        )

```

MainBehaviour - reserveVehicle

Funkcijom `reserveVehicle` obrađuje se zahtjev klijenta za rezerviranjem vozila. Pronalazi se vozilo koje korisnik želi rezervirati i ukoliko je rezervacija moguća (prema pravilima definiranim u prethodnom poglavlju) vrši se rezervacija vozila. Ukoliko se vozilo ne može rezervirati tada se šalje id neke druge stanice korisniku kako bi mogao poslati zahtjev na drugu adresu. Ukoliko se vozilo može rezervirati, šalje se potvrdna poruka o rezervaciji željenog vozila.

```

### Reserve vehicle for customer
###
async def reserveVehicle(self, vehicleName, sender):
    for vehicle in self.agent.vehicles:
        if vehicle.name == vehicleName:
            if not vehicle.isReserved and not vehicle.isCharging:
                vehicle.isReserved = True
                canReserveVehicle = True
            else:
                canReserveVehicle = False
            break
    reservationStation = str(self.agent.jid)

    if canReserveVehicle:
        msg = Message(to=sender.localpart + "@" + sender.domain)
        msg.set_metadata("performative", "inform")
        msg.set_metadata("ontology", "vehicleReserved")
        msg.body = reservationStation
        await self.send(msg)
    else:

```

```

numberOfStations = len(self.agent.stations) - 1
if numberOfStations == 0:
    msg = Message(to=sender.localpart + "@" + sender.domain)
    msg.set_metadata("performative", "inform")
    msg.set_metadata("ontology", "noOtherStations")
    msg.body = reservationStation
    await self.send(msg)
else:
    randomStation = random.randint(0, numberOfStations)
    reservationStation = self.agent.stations[randomStation].partition("@")
    [0]
    msg = Message(to=sender.localpart + "@" + sender.domain)
    msg.set_metadata("performative", "inform")
    msg.set_metadata("ontology", "otherStation")
    msg.body = reservationStation
    await self.send(msg)
    print("Vehicle_can't_be_reserved,_sending_another_station_details_to_customer...")

```

MainBehaviour - collectVehicle

Funkcijom collectVehicle korisnik javlja da je došao preuzeti vozilo. U tom trenutku također korisnik i plaća vozilo, te se ažurira ukupna zarada stanice. Također šalje se potvrdna poruka prema klijentu.

```

### Customer collects vehicle
###
async def collectVehicle(self, vehicleName, days, sender):
    for vehicle in self.agent.vehicles:
        if vehicle.name == vehicleName:
            self.agent.totalEarnings += days * vehicle.pricePerDay
            break

    msg = Message(to=sender.localpart + "@" + sender.domain)
    msg.set_metadata("performative", "inform")
    msg.set_metadata("ontology", "vehicleCollected")
    msg.body = str(self.agent.totalEarnings)
    await self.send(msg)

```

MainBehaviour - returnVehicle

Funkcijom returnVehicle korisnik javlja da je došao vratiti vozilo. Vozilo se stavlja u red na čekanje punjenja, te se njegova zastavica stanja rezervacije postavlja na false. Također se šalje potvrdna poruka prema klijentu.

```

### Customer returns vehicle
###
async def returnVehicle(self, vehicleName, chargeLeft, sender):

```

```

for vehicle in self.agent.vehicles:
    if vehicle.name == vehicleName:
        vehicle.charge = int(float(chargeLeft))
        vehicle.isCharging = True
        vehicle.isReserved = False
        break

msg = Message(to=sender.localpart + "@" + sender.domain)
msg.set_metadata("performative", "inform")
msg.set_metadata("ontology", "vehicleReturned")
msg.body = vehicleName
await self.send(msg)

```

ChargingBehaviour

Funkcijom `returnVehicle` korisnik javlja da je došao vratiti vozilo. Vozilo se stavlja u red na čekanje punjenja, te se njegova zastavica stanja rezervacije postavlja na `false`. Također se šalje potvrдна poruka prema klijentu.

```

async def on_start(self):
    print("I'm starting charging behaviour")

async def run(self):
    print("-----_CHARGING_BEHAVIOUR_-----")
    self.printCurrentState()

    needsCharging = False
    ### Charge vehicles if needed
    for vehicle in self.agent.vehicles:
        if vehicle.charge < 100:
            needsCharging = True
            break

    if needsCharging:
        print("Started charging vehicles")
        await asyncio.sleep(20)
        for vehicle in self.agent.vehicles:
            vehicle.charge = 100
            vehicle.isCharging = False
        print("All vehicles charged")
    else:
        print("No need to charge vehicles")

    print("")

```

Periodično ponašanje punjenja vozila ponavlja se u jednakim intervalima, te se odnosi na radnje vezane uz punjenje vozila koje su vraćena i koja trenutno nemaju potpuno napunjenu bateriju. Na početku svakog ciklusa ovog ponašanja ispisuje se trenutno stanje vozila i njegovu napunjenost baterijskog sustava. Nakon toga prolazi se kroz listu svih vozila i provjerava se da li je potrebno punjenje bilo kojeg vozila. Ukoliko je potrebno vrši se punjenje u nekom

proizvoljnom vremenu. Nakon što su sva vozila napunjenja takvo će ponašanje biti izvršeno ponovno u zadanom intervalu.

4.3. Rent User Agent

Ovaj agent kreiran je kao konačni automat stanja zbog svoje savršene predispozicije. Prijelaz iz stanja u stanje, te moguće tranzicije definirane su u prethodnom poglavlju gdje je detaljnije opisana arhitektura sustava. Ovdje ćemo proći kroz implementaciju konkretnih stanja u kodu.

RESERVE_VEHICLE stanje

Sa ovim stanjem počinje agent. Iz ovog stanje može ponovno prijeći u svoje stanje ili prelazi u stanje COLLECT_VEHICLE.

```
## Tell station that you want to reserve it - if it's unavailable station will give
    you
## other station from list.
##
print("-----_RESERVE_VEHICLE_-----")
print("Going_to_collect_vehicle_at_station_", self.agent.station, STATION_ADDRESS)
msg = Message(to=self.agent.station + STATION_ADDRESS)
msg.set_metadata("performative", "inform")
msg.set_metadata("ontology", "reserveVehicle")
msg.body = self.agent.vehicle
await self.send(msg)
print("Waiting_for_response...")
msgReceived = await self.receive(timeout=10000)
self.agent.station = msgReceived.body

if msgReceived and msgReceived.metadata:
    if msgReceived.metadata["ontology"] == 'vehicleReserved':
        print("Vehicle_", self.agent.vehicle, "_is_reserved_at_station_",
              msgReceived.body)
        print("Going_to_station_...")
        time.sleep(2)
        self.set_next_state(COLLECT_VEHICLE)
    elif msgReceived.metadata["ontology"] == 'noOtherStations':
        print("No_other_stations_available,_finishing_agent...")
        time.sleep(2)
        self.kill()
    elif msgReceived.metadata["ontology"] == 'otherStation':
        self.agent.station = msgReceived.body
        self.set_next_state(RESERVE_VEHICLE)
```

Prijelaz stanja ovisi o tome da li je korisnik uspješno rezervirao vozilo kod željene stanice. Ako je dobio povratnu informaciju sa novim id-em stanice tada ponovno prelazi u isto stanje i ponovno šalje zahtjev za rezervacijom na novu adresu stanice. Ukoliko dobiva povratnu informaciju da nema drugih stanica u blizini agent završava sa svojim radom.

COLLECT_VEHICLE stanje

Ukoliko je agent uspješno rezervirao vozilo, prelazi u stanje COLLECT_VEHICLE, te iz tog stanja prelazi u USE_VEHICLE stanje ukoliko je dobiven afirmativan odgovor.

```
## Tell station that you collected vehicle and pay at advance
##
print("\n-----_COLLECT_VEHICLE_-----")
print("Arrived_at_station")
msg = Message(to=self.agent.station)
msg.set_metadata("performative", "inform")
msg.set_metadata("ontology", "collectVehicle")
msg.body = ",".join([self.agent.vehicle, self.agent.days])
await self.send(msg)

print("Waiting_for_response...")
msgReceived = await self.receive(timeout=10000)
print("Vehicle_is_collected_and_rent_is_payed", msgReceived.body, "€")
```

USE_VEHICLE stanje

U ovom stanju agent koristi vozilo sve dok se njegova baterija ne isprazni. Korištena je skraćena logika vremena u sekunda podijeljena sa 10 tako da se ne treba dugo čekati kada se simulacija pokreće.

```
## Random usage, random charging after it's used and etc
## From this state to Charge or to ReturnVehicle
##
print("\n-----_USE_VEHICLE_-----")
print("I'm_using_vehicle")

for vehicleF in self.agent.vehicles:
    if vehicleF.name == self.agent.vehicle:
        vehicleChargeTime = vehicleF.chargeTime / 10
        vehicleRange = vehicleF.maxDistance
        break

fullRangeTime = vehicleRange/10
time.sleep(fullRangeTime)
print("Vehicle_fully_discharged")
self.agent.daysRemaining -= fullRangeTime
print("Rent_time_remaining:", self.agent.daysRemaining)

if self.agent.daysRemaining >= fullRangeTime + vehicleChargeTime:
    self.set_next_state(CHARGE_VEHICLE)
else:
    self.set_next_state(RETURN_VEHICLE)
```

Kada korisnik vozilo isprazni provjerava se da li je moguće i dalje koristiti vozilo, odnosno da li vrijeme koje je ostalo za korištenje vozila je dovoljno da se vozilo napuni i isprazni do kraja. Ukoliko to vrijeme dozvoljava tada prelazimo u stanje CHARGE_VEHICLE. Ukoliko to vrijeme ne dozvoljava tada se vozilo mora vratiti u stanicu odnosno u stanje RETURN_VEHICLE.

CHARGE_VEHICLE stanje

```
## Charging the vehicle
##
print("\n-----_CHARGE_VEHICLE_-----")
print("I'm_charging_vehicle")
for vehicle in self.agent.vehicles:
    if vehicle.name == self.agent.vehicle:
        vehicleChargeTime = vehicle.chargeTime / 10
time.sleep(vehicleChargeTime)
self.agent.daysRemaining -= vehicleChargeTime
print("Vehicle_charged_to_100%")
print("Rent_time_remaining:", self.agent.daysRemaining)
self.set_next_state(USE_VEHICLE)
```

U ovom stanju korisnik stavlja vozilo na punjenje. Nakon toga prolazi se u USE_VEHICLE stanje.

RETURN_VEHICLE stanje

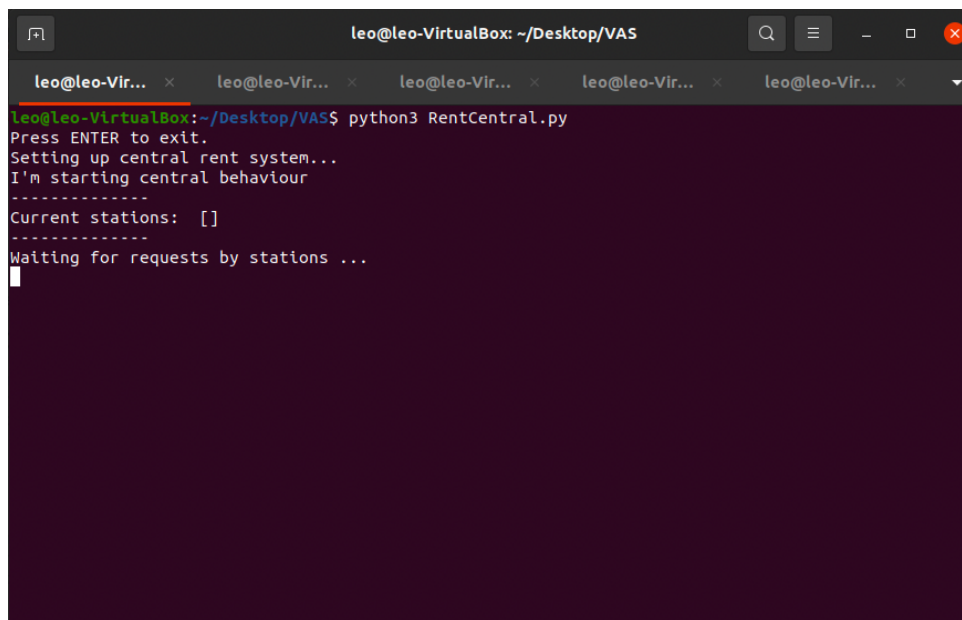
Kada je vrijeme najma vozila isteklo tada se prelazi u ovo stanje gdje se šalje zahtjev prema stanici gdje je vozilo iznajmljeno da se vozilo vraća. Također se u takvoj poruci šalje i informacija o tome koliko je trenutno baterije ostalo u vozilu kako bi stanica imala takav podatak i mogla ažurirati svoje stanje.

```
## Tell station that you are giving it back and how much charge is left
##
print("\n-----_RETURN_VEHICLE_-----")
print("I'm_returning_vehicle")
msg = Message(to=self.agent.station)
msg.set_metadata("performative", "inform")
msg.set_metadata("ontology", "returnVehicle")
msg.body = ",".join([self.agent.vehicle, str(self.agent.daysRemaining)])
print("Returning_vehicle", self.agent.vehicle, "with_charge_left:", self.agent.
      daysRemaining)
await self.send(msg)

print("Waiting_for_response...")
msgReceived = await self.receive(timeout=10000)
print("Vehicle_returned_succesfully,_heading_home...")
time.sleep(2)
```

5. Prikaz rada aplikacije

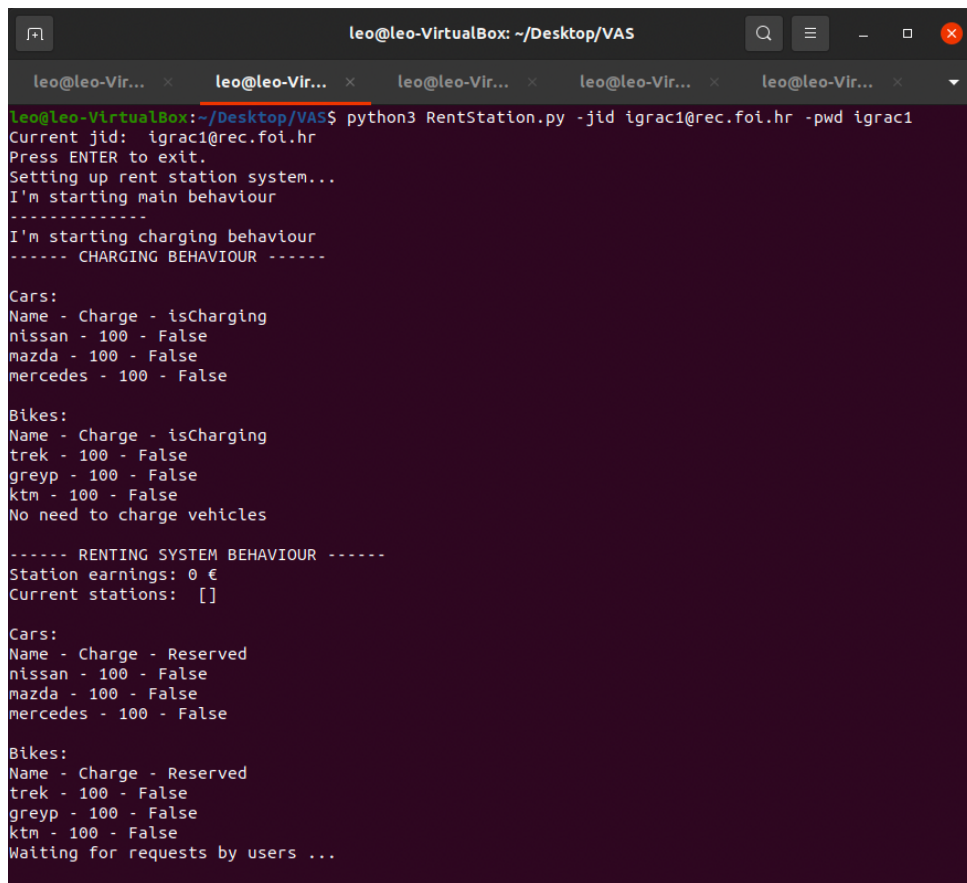
U ovom poglavlju proći ćemo kroz neke primjere korištenja aplikacije. Za početak ćemo pokrenuti glavni centralni agenta, a to je RentCentral.



```
leo@leo-VirtualBox: ~/Desktop/VAS
leo@leo-VirtualBox:~/Desktop/VAS$ python3 RentCentral.py
Press ENTER to exit.
Setting up central rent system...
I'm starting central behaviour
-----
Current stations: []
-----
Waiting for requests by stations ...
█
```

Slika 11: RentCentral pokretanje (Izvor: u vlastitoj izvedbi (snimka zaslona), 2022)

Nakon što se pokrene centralni program on ispisiuje trenutno prijavljene stanice koje su trenutno dostupne. Trenutno nema prijavljenih stanica pa je takva lista prazna. Potrebno je pokrenuti sljedećeg agenta, a to je RentStation. U ovom ćemo primjeru pokrenuti ukupno dvije različite stanice.



```
leo@leo-VirtualBox: ~/Desktop/VAS
leo@leo-VirtualBox:~/Desktop/VAS$ python3 RentStation.py -jid igrac1@rec.foi.hr -pwd igrac1
Current jid: igrac1@rec.foi.hr
Press ENTER to exit.
Setting up rent station system...
I'm starting main behaviour
-----
I'm starting charging behaviour
----- CHARGING BEHAVIOUR -----

Cars:
Name - Charge - isCharging
nissan - 100 - False
mazda - 100 - False
mercedes - 100 - False

Bikes:
Name - Charge - isCharging
trek - 100 - False
greyp - 100 - False
ktm - 100 - False
No need to charge vehicles

----- RENTING SYSTEM BEHAVIOUR -----
Station earnings: 0 €
Current stations: []

Cars:
Name - Charge - Reserved
nissan - 100 - False
mazda - 100 - False
mercedes - 100 - False

Bikes:
Name - Charge - Reserved
trek - 100 - False
greyp - 100 - False
ktm - 100 - False
Waiting for requests by users ...
```

Slika 12: Pokretanje stanice A (Izvor: u vlastitoj izvedbi (snimka zaslona), 2022)

```
leo@leo-VirtualBox: ~/Desktop/VAS
leo@leo-Vir... x leo@leo-Vir... x leo@leo-Vir... x leo@leo-Vir... x leo@leo-Vir... x
leo@leo-VirtualBox:~/Desktop/VAS$ python3 RentStation.py -jid igrac2@rec.foi.hr -pwd igrac2
Current jid: igrac2@rec.foi.hr
Press ENTER to exit.
Setting up rent station system...
I'm starting main behaviour
-----
I'm starting charging behaviour
----- CHARGING BEHAVIOUR -----

Cars:
Name - Charge - isCharging
nissan - 100 - False
mazda - 100 - False
mercedes - 100 - False

Bikes:
Name - Charge - isCharging
trek - 100 - False
greyp - 100 - False
ktm - 100 - False
No need to charge vehicles

----- RENTING SYSTEM BEHAVIOUR -----
Station earnings: 0 €
Current stations: []

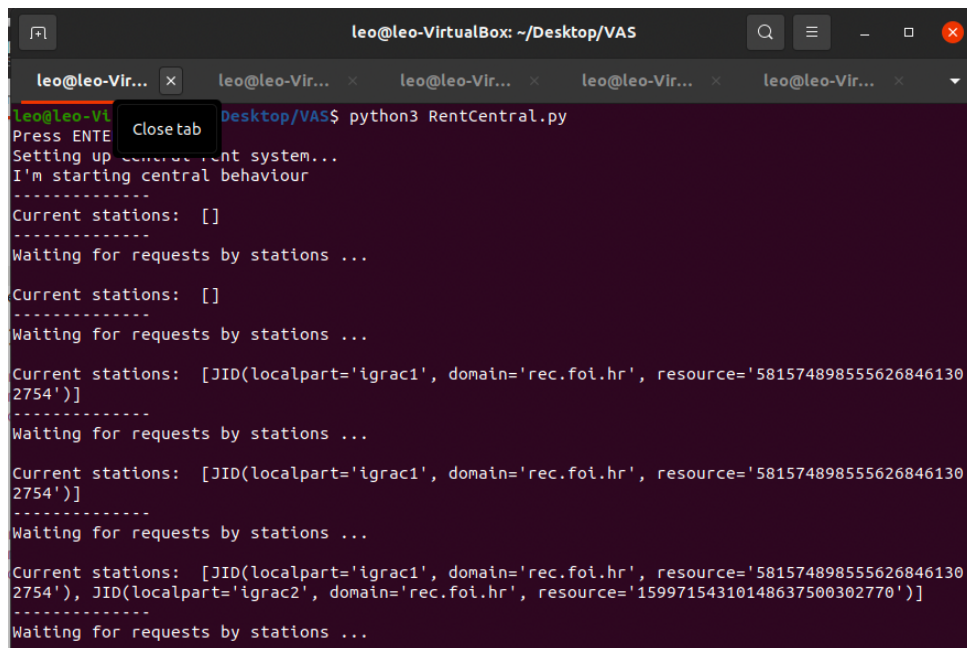
Cars:
Name - Charge - Reserved
nissan - 100 - False
mazda - 100 - False
mercedes - 100 - False

Bikes:
Name - Charge - Reserved
trek - 100 - False
greyp - 100 - False
ktm - 100 - False
Waiting for requests by users ...
```

Slika 13: Pokretanje stanice B (Izvor: u vlastitoj izvedbi (snimka zaslona), 2022)

Nakon što se takve stanice pokrenu pokreću se i njihova ponašanja, te se ispisuje trenutno stanje vozila kao i stanje na punjenju. U ovome trenutku imamo pokrenuto ukupno tri agenata od koji je jedan centralni, dok su dva stanice koje pružaju usluge.

Ukoliko sada pogledamo centralnog agenta možemo vidjeti da je on registrirao obje stanice te da ima podatke o njima.



```
leo@leo-VirtualBox: ~/Desktop/VAS
leo@leo-VirtualBox:~/Desktop/VAS$ python3 RentCentral.py
Press ENTER to start
Setting up central system...
I'm starting central behaviour
-----
Current stations: []
-----
Waiting for requests by stations ...

Current stations: []
-----
Waiting for requests by stations ...

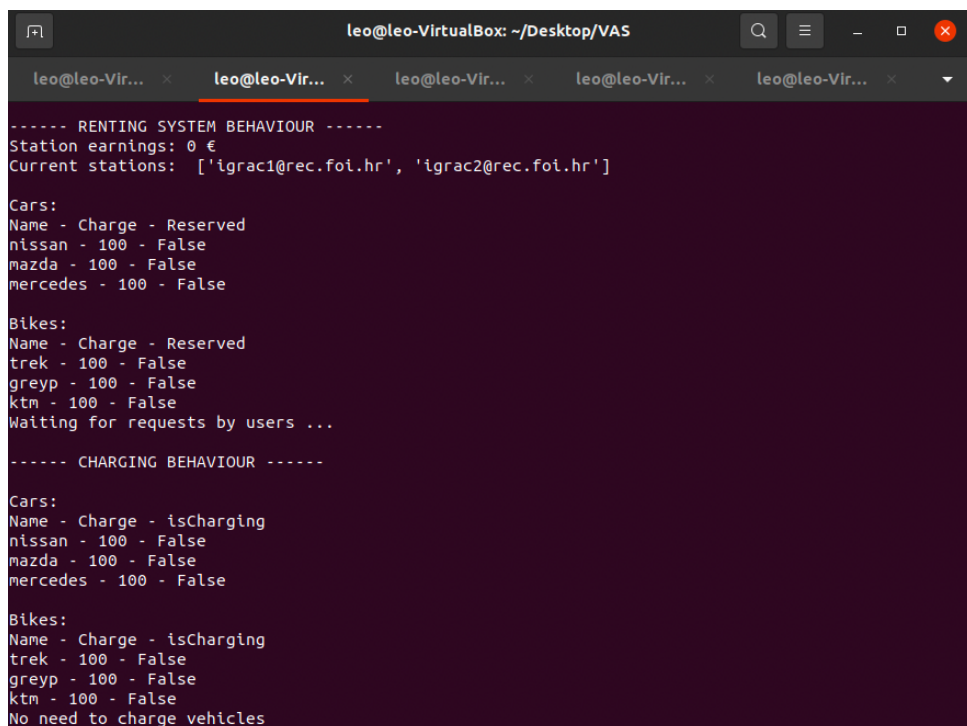
Current stations: [JID(localpart='igrac1', domain='rec.foi.hr', resource='5815748985556268461302754')]
-----
Waiting for requests by stations ...

Current stations: [JID(localpart='igrac1', domain='rec.foi.hr', resource='5815748985556268461302754')]
-----
Waiting for requests by stations ...

Current stations: [JID(localpart='igrac1', domain='rec.foi.hr', resource='5815748985556268461302754'), JID(localpart='igrac2', domain='rec.foi.hr', resource='15997154310148637500302770')]
-----
Waiting for requests by stations ...
```

Slika 14: RentCentral pregled trenutnih stanica (Izvor: u vlastitoj izvedbi (snimka zaslona), 2022)

Također kod prve stanice, zovimo je **stanica A** možemo vidjeti da ona ima također znanja o svim trenutno registriranim stanicama.



```
leo@leo-VirtualBox: ~/Desktop/VAS
leo@leo-VirtualBox:~/Desktop/VAS$ python3 RentCentral.py
Press ENTER to start
Setting up central system...
I'm starting central behaviour
-----
Current stations: []
-----
Waiting for requests by stations ...

Current stations: []
-----
Waiting for requests by stations ...

Current stations: [JID(localpart='igrac1', domain='rec.foi.hr', resource='5815748985556268461302754')]
-----
Waiting for requests by stations ...

Current stations: [JID(localpart='igrac1', domain='rec.foi.hr', resource='5815748985556268461302754')]
-----
Waiting for requests by stations ...

Current stations: [JID(localpart='igrac1', domain='rec.foi.hr', resource='5815748985556268461302754'), JID(localpart='igrac2', domain='rec.foi.hr', resource='15997154310148637500302770')]
-----
Waiting for requests by stations ...

----- RENTING SYSTEM BEHAVIOUR -----
Station earnings: 0 €
Current stations: ['igrac1@rec.foi.hr', 'igrac2@rec.foi.hr']

Cars:
Name - Charge - Reserved
nissan - 100 - False
mazda - 100 - False
mercedes - 100 - False

Bikes:
Name - Charge - Reserved
trek - 100 - False
greyp - 100 - False
ktm - 100 - False
Waiting for requests by users ...

----- CHARGING BEHAVIOUR -----

Cars:
Name - Charge - isCharging
nissan - 100 - False
mazda - 100 - False
mercedes - 100 - False

Bikes:
Name - Charge - isCharging
trek - 100 - False
greyp - 100 - False
ktm - 100 - False
No need to charge vehicles
```

Slika 15: Stanica A - prikaz svih stanica (Izvor: u vlastitoj izvedbi (snimka zaslona), 2022)

Kod **stanice B** imamo isti slučaj, ona također je zaprimila poruku od glavne stanice te ima podatke o trenutno svim prijavljenim stanicama u sustav.

```
leo@leo-VirtualBox: ~/Desktop/VAS
leo@leo-Vir... x leo@leo-Vir... x leo@leo-Vir... x leo@leo-Vir... x leo@leo-Vir... x
----- RENTING SYSTEM BEHAVIOUR -----
Station earnings: 0 €
Current stations: ['igrac1@rec.foi.hr', 'igrac2@rec.foi.hr']

Cars:
Name - Charge - Reserved
nissan - 100 - False
mazda - 100 - False
mercedes - 100 - False

Bikes:
Name - Charge - Reserved
trek - 100 - False
greyp - 100 - False
ktm - 100 - False
Waiting for requests by users ...

----- CHARGING BEHAVIOUR -----

Cars:
Name - Charge - isCharging
nissan - 100 - False
mazda - 100 - False
mercedes - 100 - False

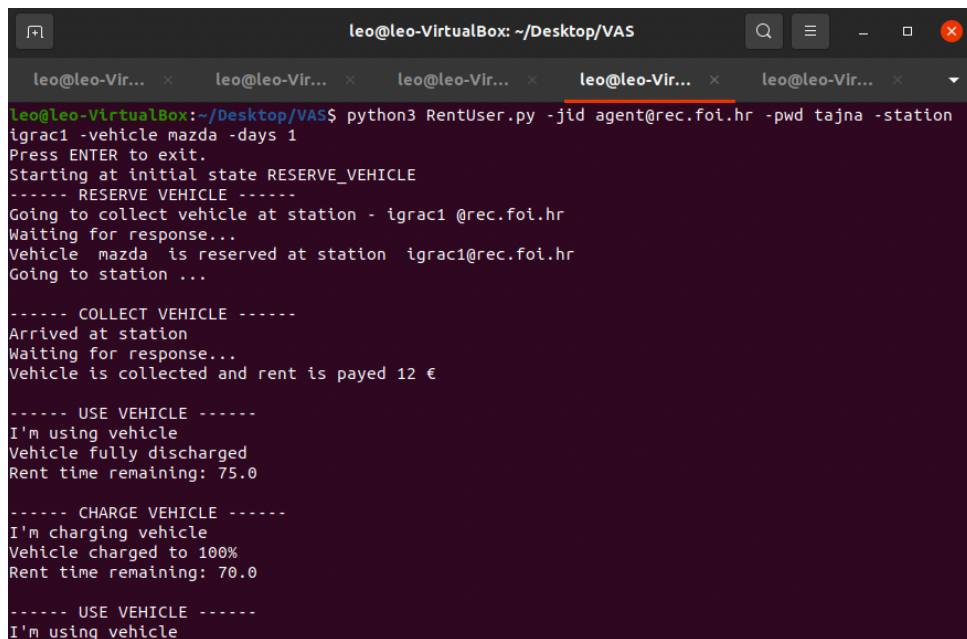
Bikes:
Name - Charge - isCharging
trek - 100 - False
greyp - 100 - False
ktm - 100 - False
No need to charge vehicles
```

Slika 16: Stanica B - prikaz svih stanica(Izvor: u vlastitoj izvedbi (snimka zaslona), 2022)

Sad kad imamo spremno sve kako bismo mogli iznajmiti električno vozilo, pokrenimo zahtjev za najmom električnog vozila preko agenta RentUser. Pokrenuti ćemo zahtjev za vozilom mazda i želimo ga rezervirati na 1 dan.

```
python3 RentUser.py -jid agent@rec.foi.hr -pwd tajna -station igrac1 -vehicle mazda -days 1
```

Zahtjev se šalje stanici A.



```
leo@leo-VirtualBox: ~/Desktop/VAS
leo@leo-VirtualBox:~/Desktop/VAS$ python3 RentUser.py -jid agent@rec.foi.hr -pwd tajna -station
igrac1 -vehicle mazda -days 1
Press ENTER to exit.
Starting at initial state RESERVE_VEHICLE
----- RESERVE VEHICLE -----
Going to collect vehicle at station - igrac1@rec.foi.hr
Waiting for response...
Vehicle mazda is reserved at station igrac1@rec.foi.hr
Going to station ...

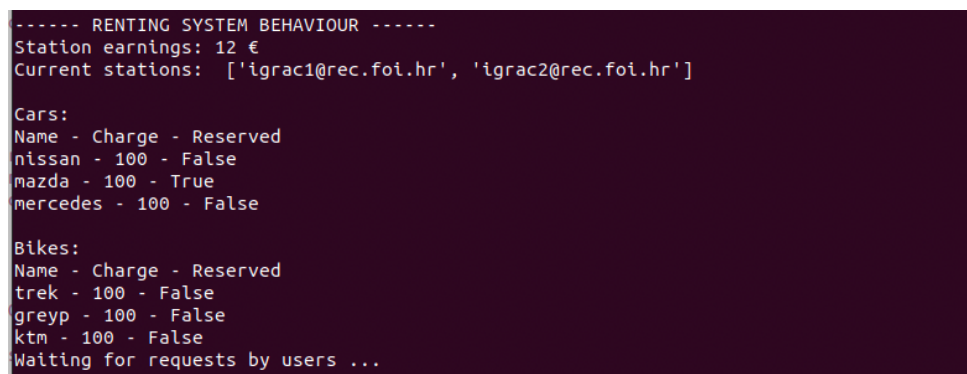
----- COLLECT VEHICLE -----
Arrived at station
Waiting for response...
Vehicle is collected and rent is payed 12 €

----- USE VEHICLE -----
I'm using vehicle
Vehicle fully discharged
Rent time remaining: 75.0

----- CHARGE VEHICLE -----
I'm charging vehicle
Vehicle charged to 100%
Rent time remaining: 70.0

----- USE VEHICLE -----
I'm using vehicle
```

Slika 17: UserRent zahtjev za najmom vozila(Izvor: u vlastitoj izvedbi (snimka zaslona), 2022)



```
----- RENTING SYSTEM BEHAVIOUR -----
Station earnings: 12 €
Current stations: ['igrac1@rec.foi.hr', 'igrac2@rec.foi.hr']

Cars:
Name - Charge - Reserved
nissan - 100 - False
mazda - 100 - True
mercedes - 100 - False

Bikes:
Name - Charge - Reserved
trek - 100 - False
greyp - 100 - False
ktm - 100 - False
Waiting for requests by users ...
```

Slika 18: Stanica A pruža najam vozila korisniku (Izvor: u vlastitoj izvedbi (snimka zaslona), 2022)

Na gornje dvije slike može se vidjeti kako je zahtjev obrađen te je korisnik uspješno rezervirao vozilo kod stanice A. Stanica A promijenila je stanje vozila u rezervirano, te je nakon toga korisnik došao pokupiti svoje vozilo. Nakon toga stanica je uspješno naplatila najam vozila te se je trenutno stanje ukupne zarade promijenilo sukladno troškovima. Korisnik je počeo koristiti vozilo te se na slici vidi jedna iteracija korištenja gdje je korisnik ispraznio vozilo do kraja, napunio ga i ponovno počeo koristiti vozilo. Sada ćemo pogledati kako izgleda kraj korištenja kada više korisnik nema vremena za korištenje vozila, odnosno vrijeme najma je isteklo.


```
leo@leo-VirtualBox: ~/Desktop/VAS
----- CHARGE VEHICLE -----
I'm charging vehicle
Vehicle charged to 100%
Rent time remaining: 70.0

----- USE VEHICLE -----
I'm using vehicle
Vehicle fully discharged
Rent time remaining: 45.0

----- CHARGE VEHICLE -----
I'm charging vehicle
Vehicle charged to 100%
Rent time remaining: 40.0

----- USE VEHICLE -----
I'm using vehicle
Vehicle fully discharged
Rent time remaining: 15.0

----- RETURN VEHICLE -----
I'm returning vehicle
Returning vehicle mazda with charge left: 15.0
```

Slika 19: Vraćanje vozila (Izvor: u vlastitoj izvedbi (snimka zaslona), 2022)

Kada je vrijeme isteklo korisnik je vratio vozilo u stanicu.

```
leo@leo-VirtualBox: ~/Desktop/VAS
leo@leo-Vir... x leo@leo-Vir... x leo@leo-Vir... x leo@leo-Vir... x leo@leo-Vir... x
----- CHARGING BEHAVIOUR -----
Cars:
Name - Charge - isCharging
nissan - 100 - False
mazda - 15 - True
mercedes - 100 - False
Bikes:
Name - Charge - isCharging
trek - 100 - False
greyp - 100 - False
ktm - 100 - False
Started charging vehicles
All vehicles charged
----- CHARGING BEHAVIOUR -----
Cars:
Name - Charge - isCharging
nissan - 100 - False
mazda - 100 - False
mercedes - 100 - False
Bikes:
Name - Charge - isCharging
trek - 100 - False
greyp - 100 - False
ktm - 100 - False
No need to charge vehicles
```

Slika 20: Ponašanje punjenja kod Stanice A (Izvor: u vlastitoj izvedbi (snimka zaslona), 2022)

Stanica je obradila zahtjev te se vozilo stavilo na punjenje. Na gornjoj slici možemo vidjeti da je pokrenuto ponašanje punjenja, te su sada sva vozila kojima je bilo potrebno punjenje napunjena.

Također proći ćemo kroz još jedan primjer u radu aplikacije. Naime ukoliko korisnik iznajmi vozilo, te drugi korisnik želi iznajmiti isto to vozilo na istoj stanici takvo vozilo nije dostupno. U tom će ga slučaju stanica A preusmjeriti u stanicu B te će korisnik uspješno rezervirati vozilo u drugoj stanici što možemo vidjeti na potonjoj slici.

```
leo@leo-VirtualBox: ~/Desktop/VAS
leo@leo-Vir... x leo@leo-Vir... x leo@leo-Vir... x leo@leo-Vir... x leo@leo-Vir... x
leo@leo-VirtualBox:~/Desktop/VAS$ python3 RentUser.py -jid vasovac@rec.foi.hr -pwd vasvas -stati
on igrac1 -vehicle mazda -days 4
Press ENTER to exit.
Starting at initial state RESERVE_VEHICLE
----- RESERVE VEHICLE -----
Going to collect vehicle at station - igrac1 @rec.foi.hr
Waiting for response...
----- RESERVE VEHICLE -----
Going to collect vehicle at station - igrac1 @rec.foi.hr
Waiting for response...
----- RESERVE VEHICLE -----
Going to collect vehicle at station - igrac1 @rec.foi.hr
Waiting for response...
----- RESERVE VEHICLE -----
Going to collect vehicle at station - igrac2 @rec.foi.hr
Waiting for response...
Vehicle mazda is reserved at station igrac2@rec.foi.hr
Going to station ...
----- COLLECT VEHICLE -----
Arrived at station
Waiting for response...
Vehicle is collected and rent is paid 48 €
```

Slika 21: Prikaz rezervacije prema usmjerenoj stanici B (Izvor: u vlastitoj izvedbi (snimka zaslona), 2022)

6. Zaključak

Prikazanim sustavom najma električnih vozila ostvaren je cilj projektnog zadatka gdje je glavni fokus na međusobnu komunikaciju agenata i njihovu suradnju. Gotovo svaki agent imao je mogućnost zaprimanja zahtjeva, ali je i slao zahtjeve prema drugim agentima. Pokušaj decentralizacije sustava je uspio na način da svaka stanica je neovisna, te može raditi i ukoliko glavni centar prestane sa svojim radom. Sustav je pojednostavljen kako bi se veća pozornost svela na međusobnu komunikaciju agenata i na implementaciju raznih ponašanja kod istih. U ovome radu korištene su različite vrste ponašanja poput cikličnog, periodičnog, te konačni automat stanja. Konačni automat stanja bio mi je najzanimljiviji od njih zbog svoje specifičnosti izvedbe gdje su stanja unaprijed jasno definirana i moguće je lako prenijeti zamišljeno u kod. Ciklično ponašanje uglavnom se koristilo kod zaprimanja i obrade zahtjeva zbog potrebe da se takva radnja vrti u krug i da agenti mogu ponovno zaprimiti novi zahtjev.

7. Popis literature

- [1] SPADE documentation. Preuzeto sa <https://spade-mas.readthedocs.io/en/latest/readme.html>.
- [2] M. Beer, M. Inverno, M. Luck, N. Jennings, C. Preist i M. Schroeder, Negotiation in Multi- Agent Systems, 1999.
- [3] Multi-Agent Vehicle Share System, Roman i ostali, 2019, Preuzeto sa https://www.researchgate.net/Agent_Vehicle_Share_System

Popis slika

1.	Dijagram jednostavnog reaktivnog agenta (Izvor: Wikipedia, 2022)	3
2.	Dijagram jednostavnog agenta učenja (Izvor: Wikipedia, 2022)	4
3.	Dijagram sustava agenata (Izvor: U vlastitoj izvedbi, 2022)	6
4.	Dijagram sustava agenata (Izvor: aceinfoway.com/blog/centralized-vs-decentralized-apps , 2022)	7
5.	Dijagram agenta RentCentral (Izvor: U vlastitoj izvedbi, 2022)	9
6.	Dijagram agenta RentStation (Izvor: U vlastitoj izvedbi, 2022)	11
7.	Tablica 1: Karakteristike električnih automobila (Izvor: U vlastitoj izvedbi, 2022) .	12
8.	Tablica 1: Karakteristike električnih bicikla (Izvor: U vlastitoj izvedbi, 2022)	13
9.	Tablica stanja i prijalaza RentUser agenta (Izvor: U vlastitoj izvedbi, 2022)	14
10.	Tablica 1: Dijagram ponašanja RentUser agenta (Izvor: U vlastitoj izvedbi, 2022)	15
11.	RentCentral pokretanje (Izvor: u vlastitoj izvedbi (snimka zaslona), 2022)	25
12.	Pokretanje stanice A (Izvor: u vlastitoj izvedbi (snimka zaslona), 2022)	25
13.	Pokretanje stanice B (Izvor: u vlastitoj izvedbi (snimka zaslona), 2022)	25
14.	RentCentral pregled trenutnih stanica (Izvor: u vlastitoj izvedbi (snimka zaslona), 2022)	25
15.	Stanica A - prikaz svih stanica (Izvor: u vlastitoj izvedbi (snimka zaslona), 2022)	25
16.	Stanica B - prikaz svih stanica(Izvor: u vlastitoj izvedbi (snimka zaslona), 2022) .	25
17.	UserRent zahtjev za najmom vozila(Izvor: u vlastitoj izvedbi (snimka zaslona), 2022)	27
18.	Stanica A pruža najam vozila korisniku (Izvor: u vlastitoj izvedbi (snimka zaslona), 2022)	27
19.	Vraćanje vozila (Izvor: u vlastitoj izvedbi (snimka zaslona), 2022)	27
20.	Ponašanje punjenja kod Stanice A (Izvor: u vlastitoj izvedbi (snimka zaslona), 2022)	28

21. Prikaz rezervacije prema usmjereojoj stanici B (Izvor: u vlastitoj izvedbi (snimka zaslona), 2022)	28
--	----

1. Prilozi

- RentCentral.py
- RentStation.py
- RentUser.py