

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

Tomislav Kolar

REAKTIVNI AGENTI ZA PRAĆENJE CIJENA PROIZVODA

SEMINARSKI RAD

Varaždin, 2022.

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

Tomislav Kolar

Matični broj: 35918/07–R

Studij: Baze podataka i baze znanja

REAKTIVNI AGENTI ZA PRAĆENJE CIJENA PROIZVODA

SEMINARSKI RAD

Mentor:

Dr. sc. Bogdan Okreša Đurić

Varaždin, veljača 2022.

Sadržaj

| | |
|--|-----------|
| 1. Uvod | 1 |
| 2. Raktivni agent i konačni automat | 2 |
| 3. Implementacija Agenata | 3 |
| 3.1. Prvi agent | 3 |
| 3.2. Drugi agent | 6 |
| 4. Prikaz rada aplikacije | 7 |
| 5. Zaključak | 10 |
| Popis literature | 11 |
| Popis slika | 12 |

1. Uvod

U ovom radu su obrađeni reaktivni agenti koji na temelju poruke izvršavaju svoju zadaću. Prvi agent će biti zadužen za prikupljanje podataka od korisnika i pokretanje pretraživanja na stranici Nabava.net. Nakon prtraživanja će agent objavijestiti drugog agenta s porukom da proizvod u rangui koji je on definirao postoji, te će drugi agent prikazati korisniku putem forme sve trgovine gdje se taj proizvod nalazi i po kojoj cijeni.

Motivacija za ovaj seminarski rad je taj da dodatno istražim kako izrađuje ovakva vrsta programa, a za realizaciju su korištene biblioteke i alati: Python, SPADE, Selenium i Tkiter.

Link za Overleaf: [**https://www.overleaf.com/read/jprhxxqyzxz**](https://www.overleaf.com/read/jprhxxqyzxz)

Link za GitHub: [**https://github.com/TKolar10/VAS-projekt**](https://github.com/TKolar10/VAS-projekt)

2. Raktivni agent i konačni automat

Agent se danas često koristi u raznim područjima koji se tiču tehnologija, a najviše se koristi u umjetnoj inteligenciji kako bi se umjetna inteligencija konstantno razvijala. Sustavi se mogu sastojati od jednog agenta koji samostalno radi u okruženju (reaktivni agent) , ali i u interakciji s korisnikom (proaktivni agent) najčešće se sastoji od više agenata i nazivaju se višeagenti sustavi koji se sastoje već od spomenutih agenata. [1]

Reaktivni agenti se nalaze u okolišu odnosno povezani su s okolišem kako bi mogli reagirati kada dogodi promjena u samom okolišu. Reaktivan ne mora komunicirati s korisnicima nego reagirati na promjene u okolinu što dovodi do toga da ga je najjednostavnije implementirati. [2]

Za realizaciju agenta koristiti SPADE platforma za višeagentne sustave, a implantacija istih je u programskom jeziku Python koja je bazirana na XMPP protokolu.

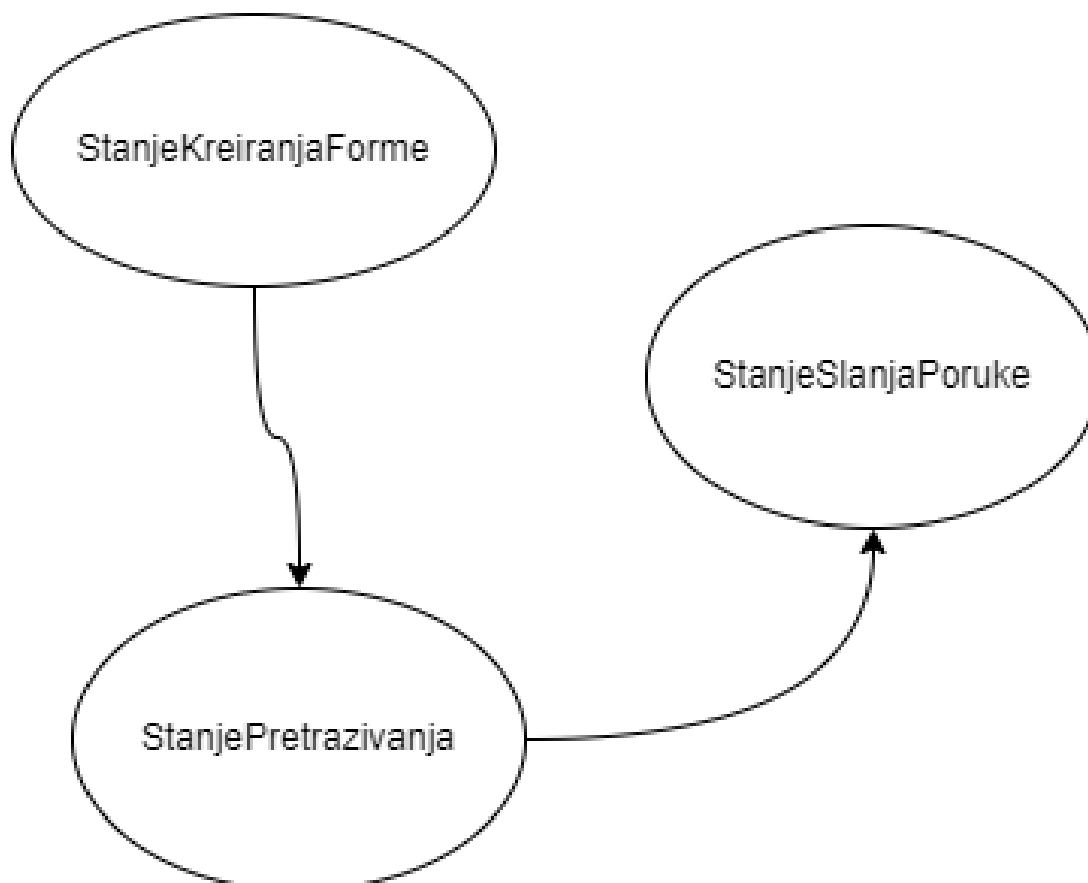
Konačni automat je zapravo sustav po matematičkom modelu s diskretnim ulazima i izlazima, te da se taj sustav može nalaziti u jednom od konačnog broja unutarnjih konfiguracija koje sa nazivaju "stanje". Sustav ima više stanja i prelazak i stanja u stanje se izvršava kad se zadovolji jedno stanje onda se može preći u drugo ili se može dogoditi da se stanje iz kojega se treba preći ponovo pokrene.[3]

Za realizaciju konačnog automata koristiti SPADE i implementirati će se SPADE.BEHAVIOUR kako bi moglo prelaziti iz stanja u stanje.

3. Implementacija Agenata

3.1. Prvi agent

Prvi agent se sastoji od 4 stanja, a to je StanjeKreiranjaForme u kojemu se kreira forma za unos željenog proizvoda i cijene od strane korisnika, StanjePretrazivanja u kojemu se pretražuje Nabava.net i StanjeSlanjaPoruke u kojemu se šalje poruka drugom agentu. Tranzicija je prikazana na Slika 1.



Slika 1: Stanje automata(Izvor: Osobna izrada)

StanjeKreiranjaForme kako mu i ime kaže je stanje u kojemu se kreira forma za upisivanje potrebnih podataka u nastavku je prikazan kod stanja

```
class StanjeKreiranjaForme(State):
    async def run(self):
        okvir = Tk()
        def ispis():
            a = proizvod.get()
            b = cijena.get()
            #c = vrijeme.get()
            if a != "" and b != "":
                okvir.destroy()
                self.agent.proizvod = a
                self.agent.cijena = b
```

```

        #self.agent.vrijeme=c
        self.set_next_state("StanjePretrazivanja")
    else:
        Label(okvir,text="Potrebno je unesti sva polja!",width=30,font=
            ("bold",15)).place(x=60,y=300)

    okvir.geometry("500x400")
    proizvod= StringVar()
    vrijeme = StringVar()
    cijena = StringVar()
    okvir.title("Forma unosa")
    labelNaslov = Label(okvir,text="Forma unosa",width=20,font=("bold",30)).
        place(x=0,y=60)

    labelProizvod = Label(okvir,text="Naziv proizvoda",width=20,font=("bold
        ",10)).place(x=80,y=150)
    unosProizvoda = Entry(okvir,textvariable=proizvod).place(x=250,y=150)

    labelCijena= Label(okvir,text="Cijena cijena",width=20,font=("bold",10)
        ).place(x=93,y=200)
    unosCijena = Entry(okvir,textvariable=cijena).place(x=250,y=200)

    gumb = Button(okvir, text='Submit',command = lambda:[ispis()], width=20,
        bg="black", fg="white").place(x=160,y=250)
    okvir.mainloop()

```

Kako bi izgled forme odnosno korisničko sučelje uopće moglo biti realizirano u Pythonu koristi se Tkinter paket. Forma se sastoji od okvira koji je 500x400, labela i polja za unos naziva proizvoda, cijene i gumba za prelazak u stanje StanjePretrazivanja

Drugo stanje StanjePretrazivanja koji pretražuje internetsku stranicu Nabava.net i dalje je prikazan kod stanja.

```

class StanjePretrazivanja(State):
    async def run(self):
        browser = webdriver.Firefox(executable_path="./drivers/geckodriver")
        wait = WebDriverWait(browser,5)
        browser.get('https://www.nabava.net')
        wait.until(EC.presence_of_element_located((By.XPATH,'/html/body/header/
            div/form/input[1]'))).send_keys(self.agent.proizvod)
        wait.until(EC.presence_of_element_located((By.XPATH,'/html/body/header/
            div/form/input[1]'))).send_keys(Keys.ENTER)
        print("Drtuga str")
        wait.until(EC.presence_of_element_located((By.XPATH,'//*[@id="kPostavke.
            pregledSortTrazilica"]'))).click()
        wait.until(EC.presence_of_element_located((By.XPATH,'//*[@id="kPostavke.
            pregledSortTrazilica"]/option[5]'))).click()

        time.sleep(1)

        skup = wait.until(EC.presence_of_element_located((By.XPATH,'/html/body/
            main/div[2]/div[1]/div[2]'))
        proizvodNaziv = skup.find_elements_by_class_name('offer__name')

```

```

cijena = skup.find_elements_by_class_name('offer__price')
nazivShopa = skup.find_elements_by_class_name('offer__store-logo')

brojac = len(cijena)
while brojac < 1:
    time.sleep(2)
    browser.refresh()
spisak = ispisTrgovineICijene(self, cijena, nazivShopa, brojac)
stringSpisak = ""
for x in spisak:
    stringSpisak += f"{x.trgovina}: {x.cijena}|"
while len(spisak) < 1:
    time.sleep(2)
    browser.refresh()

self.agent.stringSpisak = stringSpisak
browser.quit()
self.set_next_state("StanjeSlanjaPoruke")

```

Nakon što je korisnik ispunio formu i potvrdio je, podaci iz forme se proslijeđuju u sljedeće stanje odnosno StanjePretraživanja. Pretraživanje stranica iz Python skripte korišten je okvir Selenium sa internet preglednikom. Prvo je potrebno instalirati neki od preglednika u ovom radu korišten je Mozilla Firefox i Selenium driver koji je iste verzije kao i preglednik.

```

browser = webdriver.Firefox(executable_path="./drivers/geckodriver")
wait = WebDriverWait(browser, 5)
browser.get('https://www.nabava.net')

```

Prva linija u kojoj je *browser* nam služi kako bi se kreirao taj *driver* objekt, *wait* je također import *WebDriverWait* koji govori koliko će objekt *browser* pričekati do sljedeće operacije koja je *browser.get('https://www.nabava.net')* koja otvara stranicu. U nastavku koda su definirani elementi HTML-a kako bi se moglo pristupiti željenom proizvodu i njegovoj cijeni. Kada smo prikupili sve potrebne podatke kreira se lista objekata i u nju se uvrštava dobiveni podaci kako bi nam bilo lakše poslati drugom agentu.

Treće stanje je stanje StanjeSlanjaPoruke koji šalje objekt kroz poruku drugom agentu, implementacija je u nastavku.

```

class StanjeSlanjaPoruke(State):
    async def run(self):
        msg = spade.message.Message(
            to="primatelj@rec.foi.hr",
            body=f"{self.agent.stringSpisak}")
        await self.send(msg)
        print("Poruka je poslana")
        await self.agent.stop()

```

Slanje poruke odnosno obavješćavanje drugog agenta se izvodi tako da se u poruku navede kome se šalje i ono što se šalje. U našem slučaju predefiniran je primatelj, a to je "primatelj@rec.foi.hr", dok tijelo poruke je string koji će drugi agent dodatno obrađivati.

3.2. Drugi agent

Drugi agent sa svojim radom započinje odmah kada i prvi, ali on osluškuje cijelo vrijeme kada će doći poruka kako bi mogao izvršiti svoj dio posla, implementacija je u nastavku.

```
class Agent(Agent):
    class Primi(CyclicBehaviour):
        async def run(self):
            msg = await self.receive(timeout=10)
            if msg:
                ispis = msg.body
                splitanje = ispis.split("|")
                okvir = Tk()
                br=1
                lbl = Listbox(okvir)
                for x in splitanje:
                    lbl.insert(br,x)
                    br+=1
                lbl.pack()
                okvir.mainloop()

            else:
                print("Nema poruke")
```

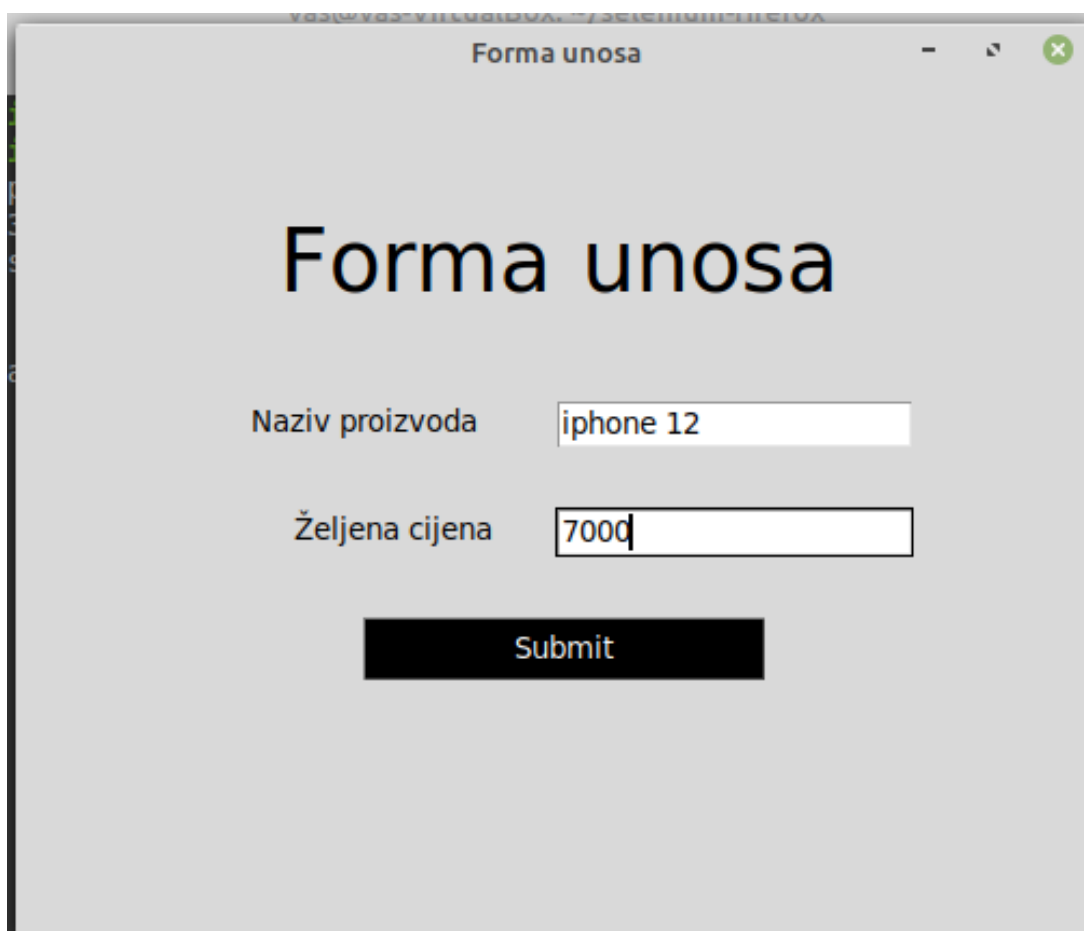
Drugi agent čeka sve dok ne dođe poruka i kada dođe poruka, ta poruka se formira odnosno razdvaja se dijelove pomoću znaka '|', te se zatim kreira Listbox u kojemu će biti ispisani sve trgovine i cijene tih proizvoda, opet se koristi Tkinter za kreiranje ispisa.

4. Prikaz rada aplikacije

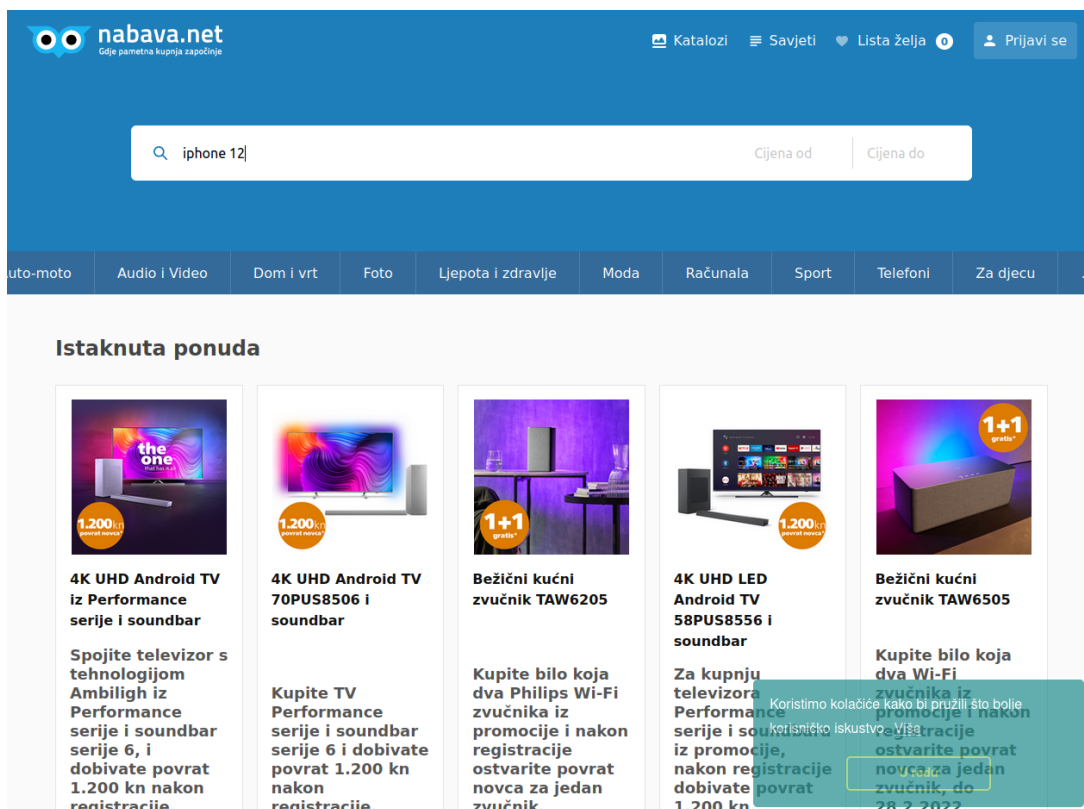
Kako bi agenti počeli s radom potrebno je pokrenuti obje skripte s naredbama

```
python3 prvi.py  
python3 drugi.py
```

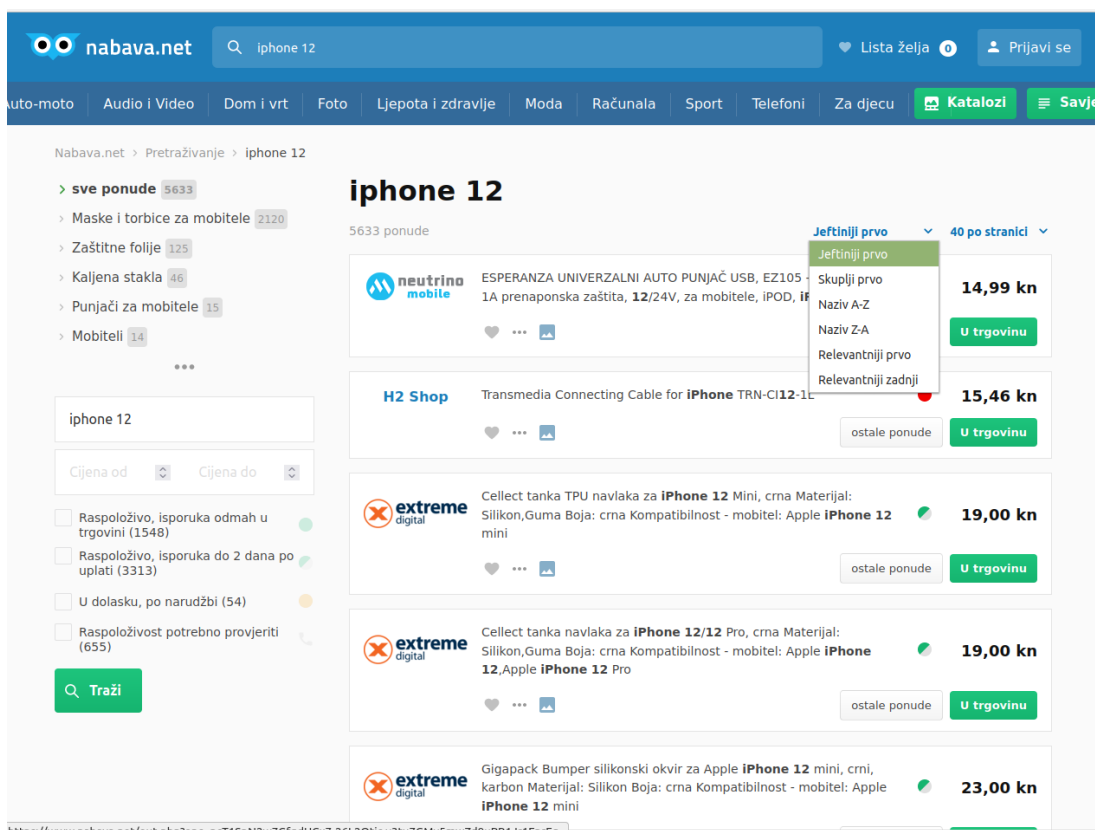
Prva skripta kreće s izvođenjem i stvara se korisničko sučelje za unos, korisnik je u polje Naziv proizvoda unio Iphone 12 što znači da će StanjePretraživanja dobiti uputu da na Nabava.net pretražuje Iphone 12 i cijena proizvoda mora biti manja od 7000 što je definirano u polju Željena cijena Slika 1. Klikom na gumb Submit na Formi unosa pokreće se StanjePretraživanja i korištenjem Seleniuma u tražilicu se unosi Iphone 12, a unosi se tako da je prethodno definirana putanja tog elementa, te nakon unosa se zadaje agentu da pritisne ENTER kako bi se krenulo se pretraživanjem Slika 2. Otvara se stranica s Iphone 12, ali je posložena tako da su proizvodi sortirani po Jeftinije prvo, što zapravo ne želimo, nego opet preko elemenata u HTML-u pronalazimo taj određeni element i mijenjamo u Relevantnije prvo Slika 3. Sad kad je generirano prema Relevantnije prvo u pozadini se prikupljaju proizvodi koji zadovoljavaju korisnikove zahtjeve i šalje se poruka drugom agentu Slika 4. Drugi agent pomoću Tkintera kreira listbox koji ispisuje u kojoj trgovini je Iphone 12 i po kojoj cijeni Slika 5.



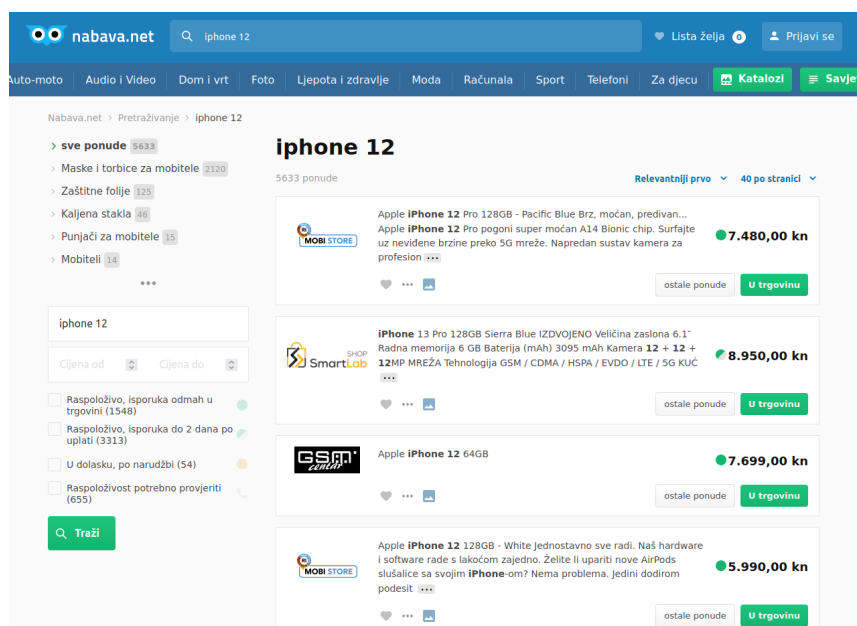
Slika 2: Forma unosa(Izvor: Osobna izrada)



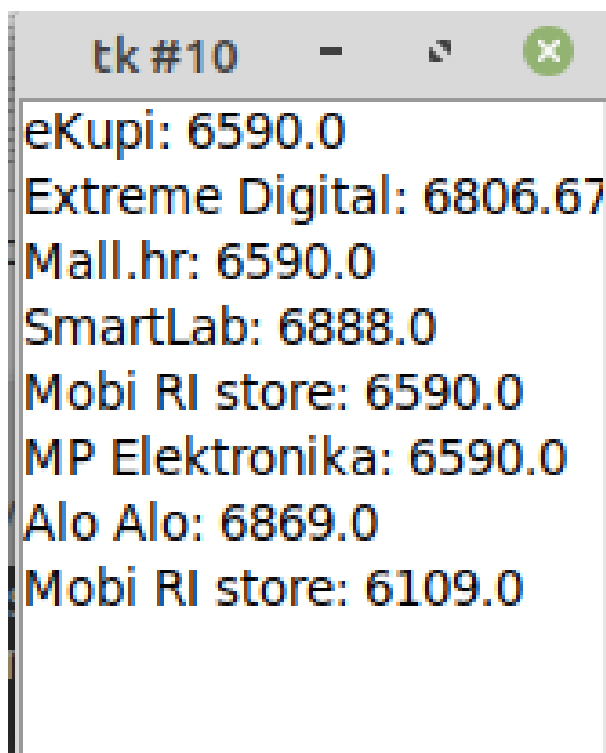
Slika 3: Pretraživanje stranice(Izvor: Osobna izrada)



Slika 4: Pretraživanje stranice(Izvor: Osobna izrada)



Slika 5: Pretraživanje stranice(Izvor: Osobna izrada)



Slika 6: Izlazna forma(Izvor: Osobna izrada)

5. Zaključak

Kroz ovaj rad se moglo vidjeti kako se kreiraju agenti, kako vrše komunikaciju, ali kako mogu i razne zadatke svladati kao što je pretraživanje stranica i odabir sortiranja. Nakon pokretanja više puta i u različita vremena agenti su vraćali najpovoljnije proizvode. Naravno, agenti nisu definirani u potpunosti i postoji prostor za napredak što znači da bi u budućnosti s novim nadogradnjama mogli dati preciznije odgovore, a brže.

Popis literature

- [1] Bellifemine, „Developing Multi-Agent Systems with JADE,” 2007. adresa: <https://www.wiley.com/en-us/Developing+Multi+Agent+Systems+with+JADE-p-97804700584049>.
- [2] A. Moreno, „Agent properties,” 2010. adresa: <https://www.slideshare.net/ToniMorenoURV/agent-properties>.
- [3] Pukeng, „An intelligent agent of finite state machine in educational game,” 2019. adresa: <https://iopscience.iop.org/article/10.1088/1742-6596/1341/4/042006/pdf>.

Popis slika

| | | |
|----|--|---|
| 1. | Stanje automata(Izvor: Osobna izrada) | 3 |
| 2. | Forma unosa(Izvor: Osobna izrada) | 7 |
| 3. | Pretraživanje stranice(Izvor: Osobna izrada) | 8 |
| 4. | Pretraživanje stranice(Izvor: Osobna izrada) | 8 |
| 5. | Pretraživanje stranice(Izvor: Osobna izrada) | 9 |
| 6. | Izlazna forma(Izvor: Osobna izrada) | 9 |