

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Jerry John Antolos

AGENTI KAO KONAČNI AUTOMATI ZA PREGLED VIJESTI

PROJEKT
(*overleaf.com*)

VIŠEAGENTNI SUSTAVI

Varaždin, 2022.

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

Jerry John Antolos

JMBAG: 0016120073

Studij: Baze podataka i baze znanja

AGENTI KAO KONAČNI AUTOMATI ZA PREGLED VIJESTI

PROJEKT
(*overleaf.com*)

Mentor:

dr. sc. Bogdan Okreša Đurić

Varaždin, srpanj 2022.

Izjava o izvornosti

Izjavljujem da je moj projekt (overleaf.com) 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 potvrdio prihvaćanjem odredbi u sustavu Moodle

Sažetak

Cilj ovog projektnog zadatka bio je napraviti aplikaciju pomoću višeagentnog sustava koja će iz različitih javno dostupnih resursa dohvaćati vijesti i prikazivati ih korisniku. Razmišljajući o UX korisniku se omogućilo filtriranje vijesti prema ključnim riječima.

Ključne riječi: SPADE; konačni automat; višeagentni sustav; pretraživanje vijesti;

Sadržaj

1. Uvod	1
2. Teorijski uvod	2
2.1. SPADE	2
2.2. Agent kao konačni automat	2
3. Instalacija potrebnih paketa	4
3.1. Instalacija paketa <i>spade</i>	4
3.2. Instalacija paketa <i>requests</i>	5
3.3. Instalacija paketa <i>beautifulsoup4</i>	5
4. Implementacija	7
4.1. Implementacija agenta <i>GlavniAgent</i>	7
4.2. Implementacija agenata <i>IndexRSS</i>	9
4.3. Implementacija agenta <i>SearchAgent</i>	10
5. Zaključak	13
Popis literature	14
Popis slika	15
Popis isječaka koda	16

1. Uvod

2. Teorijski uvod

U sklopu predmeta Višeagentni sustavi na diplomskom studiju Baze podataka i baze znanja Fakulteta organizacije i informatike Sveučilišta u Zagrebu potrebno je bilo osmisлити, implementirati i dokumentirati zadatak na samostalno odabranu temu. Odabrao sam temu "Agenti kao konačni automati za pregled vijesti". Između ostalog zadatak je bio da se u sklopu teme implementiramo višeagentnog sustava koja će iz javno dostupnih resursa dohvaćati vijesti i prikazivati ih korisniku.

Na početku je svakako dobro postaviti pitanje ima li projekt smisla odnosno pitanje možemo malo proširiti pa postaviti ga na drugačiji način ima li smisla razvijati agente za internetske aplikacije i kakve su koristi od njih. Pa upravo na ovo pitanje daju odgovor autori Enembreck, Barthès i Ávila u svom radu [1] Oni kažu kako upravo korištenje višeagenatnih sustava može imati brojne prednosti kao što su skalabilnost, stabilnost ili balansiranje opterećenja.

Kada govorimo o skalabilnosti tada mislimo na to da sustav možemo ažurirati ili proširiti bez utjecaja na njegove ostale funkcije budući da su agenti neovisni sustavi. Pojam stabilnosti nam govori o tome da kad god se neki agent sruši uvijek drugi agent može koordinirati distribuciju usluga.

2.1. SPADE

SPADE je skraćenica koja dolazi iz engleskog govornog područja, a znači Smart Python multi-Agent Development Environment. Riječ je o platformi koja se temelji na XAMPP tehnologiji i pisana je u programskom jeziku Python. SPADE je platforma koja trenutno podržava isključivo Python programski jezik odnosno SPADE je Python modul za razvoj softverskih agenata. [2] Agenti mogu imati nekoliko ponašanja koji se mogu odvijati u isto vrijeme. Riječ je o sljedećim ponašanjima:

- Cyclic
- Periodic
- Time-Out
- One-shot
- Finite State Machine
- Event Behaviour

2.2. Agent kao konačni automat

Kako bi uopće počeli implementaciju agenta kao konačnog automata moramo znati za pravo što je konačni automat kao takav. Prema definiciji konačnog automata koju možemo

pročitati na Wikipediji konačni automat je diskretni matematički model koji se sastoji od konačnog broja stanja, prijelaza između tih stanja, i akcija koje obavlja. [3]

Dok ako bi htjeli definirati agente kao konačne automate možemo se poslužiti i definicijom koju smo odradili na predavanjima. Ova definicija nam detaljizira agente kao konačne automate, a glasi:

Konačni automat je šestorka $(\Sigma, \Gamma, S, S_0, \delta, \omega)$ pri čemu je:

- Σ ulazna abeceda (konačni neprazni skup simbola)
- Γ izlazna abeceda (konačni neprazni skup simbola)
- S konačni neprazni skup stanja
- S_0 početno stanje ($S_0 \in S$)
- δ funkcija prijelaza $\delta : S \times \Sigma \rightarrow S$
- ω izlazna funkcija $\omega : S \times \Sigma \rightarrow \Gamma$

[4]

3. Instalacija potrebnih paketa

Na samom početku potrebno je postaviti razvojnu okolinu i instalirati potrebne pakete kako bi uspješno mogli razviti programski proizvod. U sljedećim poglavljima imati ćemo priliku vidjeti koji su nam sve paketi potrebni i kako ih instalirati na naš uređaj.

3.1. Instalacija paketa *spade*

Za početak krenuti ćemo s platformom SPADE. Kao i što smo naveli SPADE je platforma za razvoj višeagentnih sustava u programskom jeziku Python. Kako bi ju instalirali potrebno je pokrenuti naredbu:

```
.\pip install spade
```

Nakon što smo pokrenuli naredbu instalacija će se pokrenuti a po završetku instalacije ispis na ekranu konzole bi trebao izgledati kao na slici 1.

[illegible]

Slika 1: Instalacija paketa *spade* (Izvor: Antolos 2022)

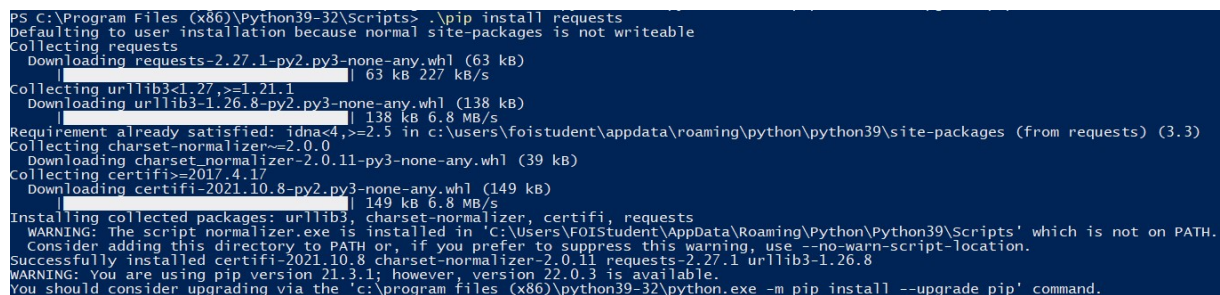
Sada je uspješno instaliran SPADE te možemo prijeći na sljedeći korak instalacije. Kao sljedeći korak, kako bi mogli raditi s HTTP zahtjevima, instalirati ćemo paket *requests*.

3.2. Instalacija paketa *requests*

Paket *requests* je biblioteka koja je razvijena s ciljem da ljudima koji ju koriste omogućiti jednostavniji rad s HTTP zahtjevima. Upravo ovaj razlog je i bio krucijalan da se odlučim za korištenje ovog paketa. Proces instalacije ovog paketa ne razlikuje se od instalacije bilo kojeg drugog paketa u Pythonu. Dakle, kako bi pokrenuli instalaciju ovog paketa dovoljno je pokrenuti sljedeću naredbu:

```
.\pip install requests
```

Po završetku instalacije u konzoli ispisati će se kako slijedi



```
PS C:\Program Files (x86)\Python39-32\Scripts> .\pip install requests
Defaulting to user installation because normal site-packages is not writeable
Collecting requests
  Downloading requests-2.27.1-py2.py3-none-any.whl (63 kB)
    | 63 kB 227 kB/s
Collecting urllib3<1.27,>=1.21.1
  Downloading urllib3-1.26.8-py2.py3-none-any.whl (138 kB)
    | 138 kB 6.8 MB/s
Requirement already satisfied: idna<4,>=2.5 in c:\users\foistudent\appdata\roaming\python\python39\site-packages (from requests) (3.3)
Collecting charset-normalizer~2.0.0
  Downloading charset-normalizer-2.0.11-py3-none-any.whl (39 kB)
Collecting certifi>=2017.4.17
  Downloading certifi-2021.10.8-py2.py3-none-any.whl (149 kB)
    | 149 kB 6.8 MB/s
Installing collected packages: urllib3, charset-normalizer, certifi, requests
WARNING: The script normalizer.exe is installed in 'C:\Users\foistudent\AppData\Roaming\Python\Python39\Scripts' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed certifi-2021.10.8 charset-normalizer-2.0.11 requests-2.27.1 urllib3-1.26.8
WARNING: You are using pip version 21.3.1; however, version 22.0.3 is available.
You should consider upgrading via the 'c:\program files (x86)\python39-32\python.exe -m pip install --upgrade pip' command.
```

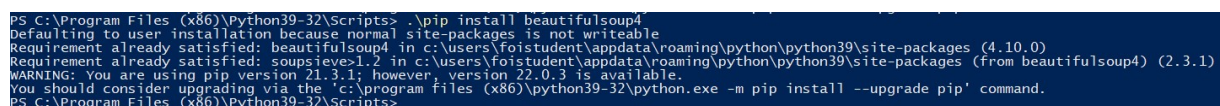
Slika 2: Instalacija paketa *requests* (Izvor: Antolos, 2022)

Nakon uspješne instalacije paketa *requests* skoro pa imamo sve potrebno da započemo rad na sustavu. Namjerno kažem skoro sve jer nedostaje glavni dio - paket koji će nam pomoći da izvučemo podatke iz HTML datoteke. Kako bi ovo učinili potreban nam je paket *beautifulsoup4*.

3.3. Instalacija paketa *beautifulsoup4*

beautifulsoup4 je paket koji služi za izvlačenje podataka zarobljenih između HTML oznaka u HTML dokumentu. Ovaj paket je važan kako bi mogli izvući novosti s dostupnih web izvora i prikazati ga korisnicima u sklopu našeg sustava. Kao i dosada pokrenuti ćemo instalaciju unosom sljedeće naredbe:

```
.\pip install beautifulsoup4
```

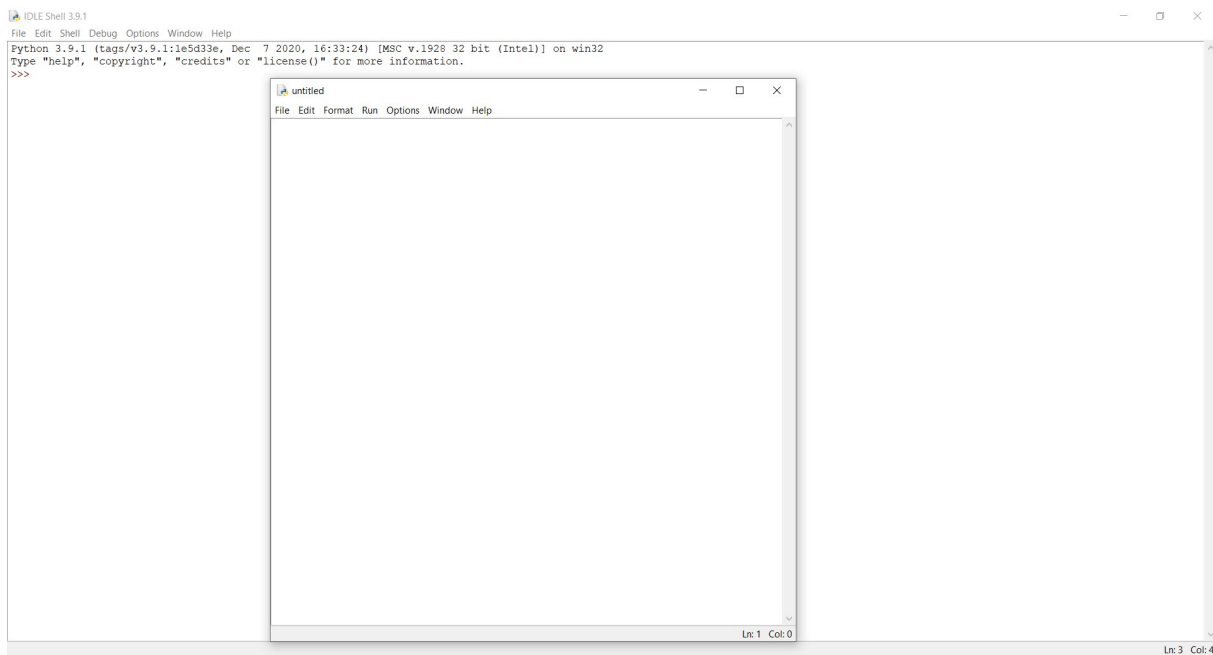


```
PS C:\Program Files (x86)\Python39-32\Scripts> .\pip install beautifulsoup4
Defaulting to user installation because normal site-packages is not writeable
Collecting beautifulsoup4
  Downloading beautifulsoup4-4.10.0-py3-none-any.whl (107 kB)
    | 107 kB 6.8 MB/s
Requirement already satisfied: soupsieve>1.2 in c:\users\foistudent\appdata\roaming\python\python39\site-packages (from beautifulsoup4) (2.3.1)
Requirement already satisfied: typing-extensions in c:\users\foistudent\appdata\roaming\python\python39\site-packages (from beautifulsoup4) (4.1.1)
Installing collected packages: beautifulsoup4
WARNING: You are using pip version 21.3.1; however, version 22.0.3 is available.
You should consider upgrading via the 'c:\program files (x86)\python39-32\python.exe -m pip install --upgrade pip' command.
PS C:\Program Files (x86)\Python39-32\Scripts>
```

Slika 3: Instalacija paketa *beautifulsoup4* (Izvor: Antolos, 2022)

Ako je paket već instaliran na računalu na kojem radimo onda će ispis u konzoli biti istovjetan ispisu prikazanom na slici 3.

Ovim posljednjim korakom instalacije paketa završili smo pripremu za početak razvoja višeagentnog sustava. Sada možemo pokrenuti Python IDLE razvojno okruženje i početi s razvojem. Nakon što pokrenemo razvojno okruženje i otvorimo novu datoteku za pisanje programskog koda naše razvojno okruženje izgleda kao na slici 4.



Slika 4: Python razvojno okruženje (Izvor: Antolos, 2022)

4. Implementacija

Kao što je i na početku ovog rada naznačeno tematika kojom ćemo se baviti je razvoj višeagentnog sustava koja će iz javno dostupnih resursa dohvaćati vijesti i prikazivati ih korisniku. Sustav se sastoji od jednog glavnog agenta, agenta za pretraživanje novosti na stranici Index.hr te agenta koji se bavi filtriranjem sadržaja.

Prije implementacije agenata potrebno je u programski kôd uključiti sve potrebne biblioteke za razvoj višeagentnog sustava na odabranu temu. Bibilioteke koje ćemo uključiti jesu *spade*, *beautifulsoup4*, *requests*. Uključivanje u programski kôd prikazano je u sljedećem isječku:

Isječak kôda 1: Potrebne biblioteke

```
1 import time
2
3 from spade.agent import Agent
4 from spade.behaviour import FSMBehaviour, State
5 from spade.message import Message
6
7 from bs4 import BeautifulSoup
8 import requests
```

4.1. Implementacija agenta *GlavniAgent*

GlavniAgent je glavni agent našeg višeagentnog sustava. Agent se sastoji od dva stanja: **PrvoStanje** i **DrugoStanje**. U stanju *PrvoStanje* agent šalje agentu **IndexRSS** poruku da krenu s radom. U stanju *DrugoStanje* agent čeka potvrdu da su novosti filtrirane te prikazuju novosti koje je korisnik zatražio. Agent po ispisivanju novosti završava s radom. Formalizirani prikaz ovog agenta u skladu s definicijom za agente kao konačne automate izgledao bi kako slijedi:

- $\Sigma = \{SaljiPoruku, CekaPoruku\}$
- $\Gamma = \{PorukaPoslana, PorukaPrimljena\}$
- $S = \{PrvoStanje, DrugoStanje\}$
- $S_0 = \{PrvoStanje\}$

Nakon formaliziranog prikaza glavnog agenta slijedi isječak programskog kôda.

Isječak kôda 2: Programski kôd za *GlavniAgent* agenta

```
1 STANJE_PRVO = "STANJE_PRVO"
2 STANJE_DRUGO = "STANJE_DRUGO"
3 STANJE_TRECE = "STANJE_TRECE"
4
```

```

5  class GlavniAgent (Agent):
6
7      listaFiltriranihNovostiIndexHr = []
8      class FSMPONASANJE(FSMBehaviour):
9          async def on_start(self):
10              print(f"{GlavniAgent.__name__}: Pokrećem se ... ({self.current_state})")
11
12          async def on_end(self):
13              print(f"{GlavniAgent.__name__}: Završavam s radom ... {self.
14                  current_state}")
15              await self.agent.stop()
16
17      class StanjePrvo(State):
18          async def run(self):
19              print(f"{GlavniAgent.__name__}: Šaljem poruku za početak prikupljanje
20                  novosti s portala ...")
21              msg = Message(to="vasprojekt2@jabber.eu.org")
22              msg.body = "Započni s prikupljanjem novosti s portala!"
23              await self.send(msg)
24              print(f"{GlavniAgent.__name__}: Poruka poslana!")
25              self.set_next_state(STANJE_DRUGO)
26
27      class StanjeDrugo(State):
28          async def run(self):
29              print(f"{GlavniAgent.__name__}: Nalazim se u završnom stanju. Čekam na
30                  primitak poruke ...")
31              msg = await self.receive(timeout=30)
32              print(f"{GlavniAgent.__name__}: Poruka zaprimljena!")
33
34              if msg:
35                  if msg.body != '':
36                      print(f"{GlavniAgent.__name__}: Zaprimio sam poruku sljedećeg
37                          sadržaja: \"{msg.body}\"")
38                      print(f"{GlavniAgent.__name__}: Novosti prikupljene i spremne za
39                          prikaz.")
40                      print("\n ----- \n")
41                      print("\n ----- N O V O S T I ----- \n")
42                      print("\n ----- \n")
43
44                      counter = 1
45                      for novost in GlavniAgent.listaFiltriranihNovostiIndexHr:
46                          print(f"{counter}. {novost}")
47                          counter += 1
48
49                  else:
50                      print("Nema odgovora.")
51
52          async def setup(self):
53              agent = self.FSMPONASANJE()
54              agent.add_state(name=STANJE_PRVO, state=self.StanjePrvo(), initial=True)
55              agent.add_state(name=STANJE_DRUGO, state=self.StanjeDrugo())
56              agent.add_transition(source=STANJE_PRVO, dest=STANJE_DRUGO)
57              self.add_behaviour(agent)

```

4.2. Implementacija agenata *IndexRSS*

Kako bi mogli prikazivati novosti korisnicima sustava te novosti potrebno je prvo preuzeti s novinskih portala na internetu. Za to preuzimanje zadužen je agent *IndexRSS*. Kako i sam naziv govori agent će preuzimati novosti s portala *Index.hr* koje su na stranici pohranjene u RSS obliku. Riječ je o implementaciji koja se izvodi *straightforward* a na identičan način možemo implementirati bilo kojeg agenta samo ćemo umjesto u atribut *addr* postaviti poveznicu na željeni RSS file.

Agenata *IndexRSS* je agent koji se sastoji od tri stanja: **PrvoStanje**, **DrugoStanje** i **TreceStanje**. U stanju *PrvoStanje* agent čeka da primi poruku od primarnog agenta za početak rada. U stanju *DrugoStanje* agent preuzima novosti iz zadanog izvora i šalje ih agentu *SearchAgent*. U stanju *TreceStanje* agent šalje poruku da je gotov s preuzimanjem novosti agentu *SearchAgent*. Formalizirani prikaz ovog agenta u skladu s definicijom za agente kao konačne automate izgledao bi kako slijedi:

- $\Sigma = \{CekajPoruku, PreuzimanjeNovosti\}$
- $\Gamma = \{PorukaPrimljena, NovostiPreuzete\}$
- $S = \{PrvoStanje, DrugoStanje, TreceStanje\}$
- $S_0 = \{PrvoStanje\}$

Nakon formaliziranog prikaza agenta slijedi isječak programskog kôda.

Isječak kôda 3: Programski kôd za *IndexRSS* agenta

```
1 class IndexRSS (Agent):
2
3     class FSMPoslanje (FSMBehaviour):
4         async def on_start (self):
5             print (f"{IndexRSS.__name__}: Pokrećem se ... ({self.current_state})")
6
7         async def on_end (self):
8             print (f"{IndexRSS.__name__}: Završavam s radom ... {self.current_state}")
9             )
10            await self.agent.stop()
11
12     class StanjePrvo (State):
13         async def run (self):
14             msg = await self.receive (timeout=15)
15             print (f"{IndexRSS.__name__}: Zaprmio sam poruku sadržaja: \"{msg.body
16                 }\")
17             self.set_next_state (STANJE_DRUGO)
18
19     class StanjeDrugo (State):
20         async def run (self):
21
22             addr = requests.get ('https://www.index.hr/rss')
```

```

21
22         indexRSS = BeautifulSoup(addr.content, 'xml')
23         items = indexRSS.find_all('item')
24         for item in items:
25             SearchAgent.listaNovostiIndexHr.append(f"\n\nDatum: {item.pubDate.
                text}.\n\nNaslov: {item.title.text}\n\nSažetak: {item.description.
                text}\n\nPoveznica: {item.link.text}\n\n
                -----")
26         self.set_next_state(STANJE_TRECE)
27
28     class StanjeTrece(State):
29         async def run(self):
30             print(f"{IndexRSS.__name__}: Šaljem poruku da sam završio s
                prikupljanjem novosti ...")
31             msg = Message(to="vasprojekt3@jabber.eu.org")
32             msg.body = "Završio s prikupljanjem novosti s portala!"
33             await self.send(msg)
34             print(f"{IndexRSS.__name__}: Poruka poslana!")
35
36     async def setup(self):
37         agent = self.FSMPonasanje()
38         agent.add_state(name=STANJE_PRVO, state=self.StanjePrvo(), initial=True)
39         agent.add_state(name=STANJE_DRUGO, state=self.StanjeDrugo())
40         agent.add_state(name=STANJE_TRECE, state=self.StanjeTrece())
41         agent.add_transition(source=STANJE_PRVO, dest=STANJE_DRUGO)
42         agent.add_transition(source=STANJE_DRUGO, dest=STANJE_TRECE)
43         self.add_behaviour(agent)

```

4.3. Implementacija agenta *SearchAgent*

Agent *SearchAgent* kao što mu i samo ime kaže služi za pretraživanje odnosno filtraciju novosti koje su preuzete s agentom *IndexRSS*. Svrha ovog agenta očituje se u funkciji filtriranja novosti relevantnih traženom pojmu. Kao i ostali agenti u ovom sustavu i ovaj se na početku inicijalizira u sustavu i postavlja u stanje čekanja. Sastoji se od tri stanja: **PrvoStanje**, **DrugoStanje** i **TreceStanje**. Stanje čekanja je stanje **PrvoStanje**. Nakon što agent završi s preuzimanjem novosti prelazi u stanje **TreceStanje** gdje se šalje zahtjev agentu *SearchAgent* za pretraživanje novosti po unesenom pojmu. Nakon što je stanje **PrvoStanje** primilo poruku da je agent *IndexRSS* završio s prikupljanjem novosti agent *SearchAgent* prelazi u stanje **DrugoStanje** gdje prima korisnički unos za pretraživanje po određenom pojmu. Kada pretraživanje završi, a vijesti s tim pojmom budu prikupljene agent *SearchAgent* prelazi u stanje **TreceStanje** i šalje poruku agentu *GlavniAgent* da je gotov. Sada agent *GlavniAgent* ispisiuje pretražene novosti. Nakon ispisa višeagentni sustav završava s radom.

Foramlizirani prikaz ovog agenta u skladu s definicijom za agente kao konačne automate izgledao bi kako slijedi:

- $\Sigma = \{CekajPoruku, FiltriranjeNovosti, SaljiPoruku\}$
- $\Gamma = \{PorukaPrimljena, NovostiFiltrirane\}$

- $S = \{PrvoStanje, DrugoStanje, TreceStanje\}$
- $S_0 = \{PrvoStanje\}$

Nakon formaliziranog prikaza agenta slijedi isječak programskog kôda filtracije.

Isječak kôda 4: Programski kôd za SearchAgent agenta

```

1  class SearchAgent (Agent) :
2
3
4      listaNovostiIndexHr = []
5      class FSMPosnasanje (FSMBehaviour) :
6          async def on_start (self) :
7              print (f"{SearchAgent.__name__}: Pokrećem se ... ({self.current_state})")
8
9          async def on_end (self) :
10             print (f"{SearchAgent.__name__}: Završavam s radom ... {self.
11                 current_state}")
12             await self.agent.stop()
13
14     class StanjePrvo (State) :
15         async def run (self) :
16             msg = await self.receive (timeout=15)
17             print (f"{SearchAgent.__name__}: Zaprímio sam poruku sadržaja: \"{msg.
18                 body}\"")
19             self.set_next_state (STANJE_DRUGO)
20
21     class StanjeDrugo (State) :
22         async def run (self) :
23
24             unos = input ("Unesi pojam za pretraživanje (\"all\" - prikaz svih
25                 vijesti): ")
26
27             if (unos != "all") :
28                 for novost in SearchAgent.listaNovostiIndexHr :
29                     if (unos in novost) :
30                         GlavniAgent.listaFiltriranihNovostiIndexHr.append (novost)
31                         print ("Vijesti su filtrirane!")
32                         self.set_next_state (STANJE_TRECE)
33                     else :
34                         for novost in SearchAgent.listaNovostiIndexHr :
35                             GlavniAgent.listaFiltriranihNovostiIndexHr.append (f"{novost}")
36                             self.set_next_state (STANJE_TRECE)
37
38     class StanjeTrece (State) :
39         async def run (self) :
40             print (f"{SearchAgent.__name__}: Šaljem poruku da sam završio s
41                 filtriranjem novosti ...")
42             msg = Message (to="vasprojekt1@jabber.eu.org")
43             msg.body = "Završio s filtracijom novosti s portala!"
44             await self.send (msg)
45             print (f"{SearchAgent.__name__}: Poruka poslana!")

```



```
43
44     async def setup(self):
45         agent = self.FSMPonasanje()
46         agent.add_state(name=STANJE_PRVO, state=self.StanjePrvo(), initial=True)
47         agent.add_state(name=STANJE_DRUGO, state=self.StanjeDrugo())
48         agent.add_state(name=STANJE_TRECE, state=self.StanjeTrece())
49         agent.add_transition(source=STANJE_PRVO, dest=STANJE_DRUGO)
50         agent.add_transition(source=STANJE_DRUGO, dest=STANJE_TRECE)
51         agent.add_transition(source=STANJE_DRUGO, dest=STANJE_TRECE)
52         self.add_behaviour(agent)
```

5. Zaključak

U ovom projektu za zadatak je bilo implementirati višeagentni sustav za preuzimanje, filtriranje i prikazivanje novosti koristeći SPADE i Python. Tema je bila osobni izbor svakog studenta dakle proizvoljna ali u skladu s akademskim okruženjem. Očekivanja koja sam imao od teme na početku izrade bila jako niska jer sama svijest da pristupam radu s Python programskim jezikom mi je bila zastrašujuća. Kako je projekt išao k svome vrhuncu nadao sam se da ću negdje u dubini svoje duše zavoliti rad u Pythonu ali to se nije dogodilo. Uz pisanje dokumentacije u \LaTeX -u sintaksa programskog jezika Python i biblioteke koje prvi put koristim su mi bili najveći izazovi ovog projekta.

Osim na kolegiju Višeagentni sustavi do sada nisam imao prilike se susresti ovako opsežno s Python programskim jezikom tako da sam sada imao mnoge prepreke koje sam morao savladati kako bi uspješno realizirao projekt.

U samoj konačnici realizirao sam mali višeagentni sustav za prikupljanje novosti. Smatram ovaj sustav dobrim početkom. Na sustavu bi se moglo još poraditi u smislu da se nadopuni još dodatnim agentima na određene stranice. Filtriranje informacija bi moglo biti malo bolje realizirano. Sustav bi mogao nadopuniti da preuzima cijele vijesti ali to bi zahtjevalo malo veću doradu. Sve u svemu zadovoljan sam rezultatom iako se moglo više i bolje.

Popis literature

- [1] F. Enembreck, J.-P. Barthès i B. C. Ávila, „Personalizing Information Retrieval with Multi-agent Systems,” *Cooperative Information Agents VIII*, M. Klusch, S. Ossowski, V. Kashyap i R. Unland, ur., Berlin, Heidelberg: Springer Berlin Heidelberg, 2004., str. 77–91, ISBN: 978-3-540-30104-2.
- [2] Wikipedia. „Spade.” (12. 2021.), adresa: <https://hr.wikipedia.org/wiki/Spade>.
- [3] —, „Konačni automat.” (1. 2022.), adresa: https://hr.wikipedia.org/w/index.php?title=Kona%C4%8Dni_automat.
- [4] M. Schatten. „Agenti sa sposobnošću deduktivnog rezoniranja.” (9. 2021.), adresa: <https://elf.foi.hr/mod/resource/view.php?id=9050>.
- [5] J. J. Antolos. „Instalacija paketa SPADE.” (2. 2022.).
- [6] —, „Instalacija paketa Requests.” (2. 2022.).
- [7] —, „Instalacija paketa BeautifulSoup4.” (2. 2022.).
- [8] —, „Python razvojno okruženje.” (2. 2022.).

Popis slika

1.	Instalacija paketa <i>spade</i> (Izvor: Antolos 2022)	4
2.	Instalacija paketa <i>requests</i> (Izvor: Antolos, 2022)	5
3.	Instalacija paketa <i>beautifulsoup4</i> (Izvor: Antolos, 2022)	5
4.	Python razvojno okruženje (Izvor: Antolos, 2022)	6

Popis isječka koda

1.	Potrebne biblioteke	7
2.	Programski kôd za GlavniAgent agenta	7
3.	Programski kôd za IndexRSS agenta	9
4.	Programski kôd za SearchAgent agenta	11