Agentski utemeljena paradigma u razvoju sustava u oblaku

Ziterbart, Marija

SVEUČILIŠTE U SPLITU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET

DIPLOMSKI RAD

AGENTSKI UTEMELJENA PARADIGMA U RAZVOJU SUSTAVA U OBLAKU

Marija Ziterbart

Split, svibanj 2017.

Sadržaj

Uvod

Inteligentni agenti i višeagentski sustavi sve više pronalaze uporabu u svim sferama razvoja programske podrške, tako i rješavanju problema u svakodnevnom životu. Računarstvo u oblaku i njegov najrasprostranjeniji sloj softver kao usluga idealna je podloga za kreiranje višeagentskog sustava.

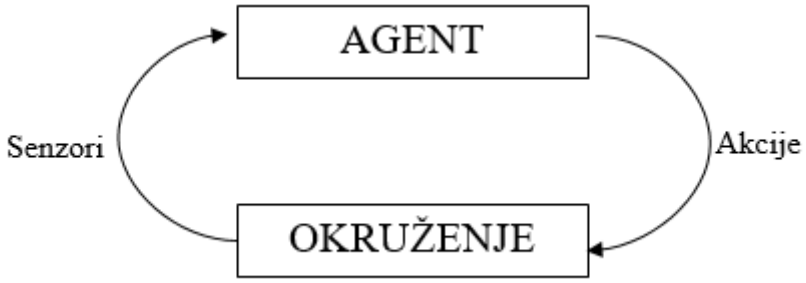
Prvo poglavlje donosi pregled inteligentnih agenata, njihovih okruženja i arhitektura. Slijedi poglavlje o višeagentskim sustavima gdje se bitno ističe pojam komunikacije agenata, kooperacije i pregovaranja. Pregled računarstva u oblaku, njegovih prednosti i slojeva donesen je u trećem poglavlju.

Cilj ovog rada je pokazati kako na izgled nepovezani pojmovi višeagentskog sustava i računarstva u oblaku čine skladnu zajednicu kad su u pitanju konkretna rješenja problema današnjice. Zato je u četvrtom poglavlju dan konkretan primjer implementacije sustava za upravljanje resursima koji donosi spoj ova dva svijeta, višeagentskih sustava i računarstva u oblaku. Sustav za upravljanje resursima koji je ovdje dan kao primjer je implementiran kao web aplikacija u oblaku.

S obzirom na konkretan primjer sustava za upravljanje resursima opći cilj ovoga rada je uključivanje što više malih tvrtki i malih poduzetnika u digitalizaciju za što uspješnije poslovanje.

1. Inteligentni agenti

Agentom se može nazvati sve što svoju okolinu opaža preko senzora i djeluje na tu okolinu. Tako, ljudski agent ima oči, uši i druge organe za senzore, a ruke, noge, usta i ostale dijelove tijela za djelovati na okolinu. Robot ima fotoaparat, tražitelje dometa infracrvenog zračenja za senzore i različite motore za djelovanje. Softverski agent ima tipke, sadržaj datoteke i mrežne pakete za senzorske ulaze, a djeluje na okoliš prikazivanjem na zaslonu, pisanjem datoteka i slanjem mrežnih paketa. [1]



Slika 1 Agent u njegovom okruženju

Još 1995. Wooldridge i Jennings definiraju agenta kao računalni sustav koji se nalazi u nekom okruženju, sposoban je za autonomno djelovanje u tom okruženju da bi zadovoljio vlastite ciljeve. (Slika 1) Predlažu i glavne značajke inteligentnog agenta: [2]

1. Reaktivnost: Inteligentni agent može uočiti svoje okruženje i pravovremeno odgovoriti na promjene koje se u njemu pojavljuju kako bi zadovoljio svoje ciljeve
2. Proaktivnost: Inteligentni agent mora pokazati ponašanje usmjereno prema cilju. Poduzimajući inicijativu kako bi zadovoljio svoje ciljeve.
3. Društvena sposobnost: Inteligentni agenti sposobni su za komuniciranje i djelovanje s drugim agentima kako bi zadovolji svoje ciljeve.

U literaturi, se uz pojam inteligentnog agenta često veže i pojam PACO (*eng. PAGE*). Gdje P predstavlja percepte (*eng. Percept*), A akcije (*eng. Action*), C cilj (*eng. Goal*) i O okolica (*eng. Environment*). Isto tako, uz gore navedene glavne značajke inteligentnog agenta često imamo puno više značajki inteligentnog agenta kao što su: autonomija, mobilnost, suradnja, komunikativnost, adaptivnost, itd.

1.1. Okruženje inteligentnog agenta

Agenti su jednostavno računalni sustavi koji su sposobni samostalno djelovati u nekom okruženju kako bi ispunili svoje ciljeve. Tipično, agent osjeti svoje okruženje (fizičkim ili softverskim senzorima) i ima na raspolaganju akcije koje će izvršiti i mijenjati svoje okruženje, i može se dogoditi nedeterministički odgovori na izvršenje tih akcija. Znači, okruženje se razlikuju po svojim svojstvima, 1995. Russell i Norvig predlažu sljedeću klasifikaciju okruženja inteligentnog agenta: [2]

- Pristupačna naprama nepristupačnoj. Pristupačno okruženje je ono u kojem agent može primati potpune, točne i ažurirane informacije o stanju okolice. Većina okruženja u stvarnom svijetu nisu pristupačna u ovakvom smislu.
- Deterministično naprama nedeterministično. Deterministično okruženje je ono u kojem svaka akcija ima jedan zajamčeni učinak. Nema neizvjesnosti o stanju koja će rezultirati iz izvođenja akcija.
- Statičko naprama dinamično. Statičko okruženje je ono za koje se pretpostavlja da ostaje nepromijenjeno, osim izvršavanjem radnji od strane agenata. Dinamičko okruženje, je ono na koje djeluju i drugi procesi i koje se mijenja i izvan kontrole agenata. Fizički svijet i Internet je jako dinamičko okruženje.
- Epizodno naprama ne-epizodno. Epizodno okruženje je ono okruženje u kojem postoji fiksni i konačni broj radnji i percepcija.

Najsloženija okruženja su ona koja su nepristupačna, nedeterministička, dinamička i epizodna. Okruženja koja imaju takva svojstva zovemo i otvorenim.

1.2. Arhitekture inteligentnih agenata

Svaki agentski program koji provodi funkciju agenta (mapiranje od percepcije do akcije) izvodi se na nekoj vrsti arhitekture računalnog uređaja s fizičkim senzorima i perceptima. Agenta predstavlja arhitektura i program.

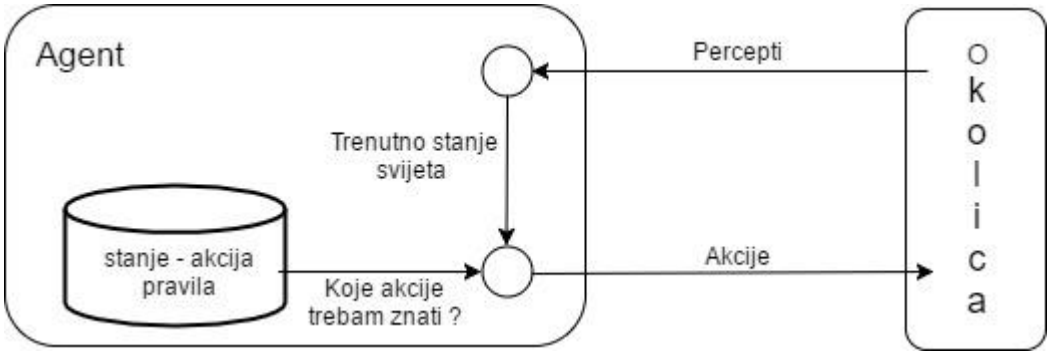
$$agent = arhitektura + program$$

1.2.1. Jednostavan reaktivni agent

Najjednostavnija vrsta agenta, svoje akcije temelji na trenutnom perceptu, ignorirajući ostatak povijesti percepata. Ponašanje jednostavnog reaktivnog agenta je temeljeno na pravilu uvjet -> akcija te se jednostavno opisuje funkcijom:

$$Ag: E \rightarrow Ac$$

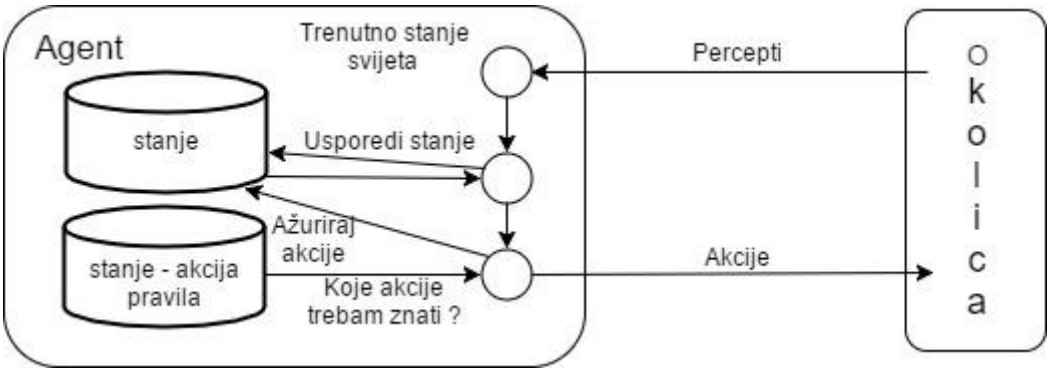
gdje E predstavlja okruženje (*eng. Environment*), a Ac predstavlja akciju (*eng. Action*). Na slici (Slika 2) je prikazana arhitektura jednostavnog reaktivnog agenta.



Slika 2 Arhitektura jednostavnog reaktivnog agenta

1.2.2. Agent sa stanjem

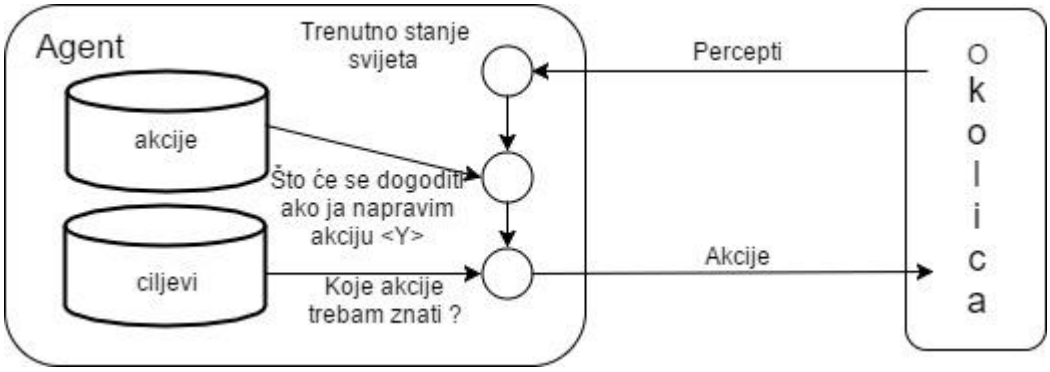
Agenti sa stanjem su agenti koji su sposobni zadržati stanje, opremljeni su nekom vrstom interne strukture podataka. Znaju kako se okolica mijenja i kako agent djeluje na okolicu. To znanje o okolici se prevodi jednostavnim Booleovim ciklusima te se često naziva modelom svijeta. Agent sa stanjem se tako često naziva i modelu orijentiran agent. Na slici (Slika 3) je prikazana arhitektura agenta sa stanjem.



Slika 3 Arhitektura agenta sa stanjem

1.2.3. Cilju orijentirani agent

Znati trenutno stanje okolice nije uvijek dovoljno za odlučiti što napraviti ako želimo doći do cilja. Ovaj agent je složeniji od prethodna dva. Osim trenutnog stanja, cilju orijentirani agent treba neku vrstu informacija o cilju do kojeg želi doći. Uključuje uvažavanje budućnosti i sposoban je rasuđivati zbog toga zahtjeva pretrage i planiranje. Vodi se osnovnim pitanjem: "*Što će se dogoditi ako ja napravim …*". Arhitekturu cilju orijentiranog agenta prikazuje Slika 4. Iako se cilju orijentirani agent, možda čini manje učinkovit dosta je fleksibilniji jer znanje koje podupire njegove odluke je reprezentirano eksplicitno i može se mijenjati.

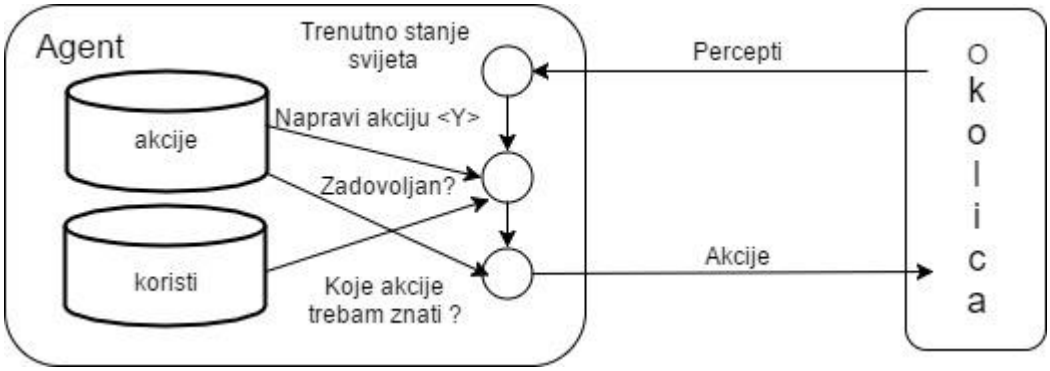


Slika 4 Arhitektura cilju orijentiranog agenta

1.2.4. Koristi orijentirani agent

Koristi orijentirani agent je agent koji ima mogućnost mjeriti vlastiti učinak. U većini okruženja ciljevi nisu dovoljni da bi se generiralo visokokvalitetno ponašanje. Ciljevi samo pružaju sirovo binarno razlikovanje, jesmo li „zadovoljni“ ili „nezadovoljni“ stanjem. Tako dolazimo do

pojma koristi. Učinak se odnosi na kvalitetu, ocjenu korisnosti. Koristi orijentirani agent mora modelirati i pratiti svoje okruženje, zadatke koje uključuju veliku količinu istraživanja o percepciji, obrazloženju, učenju. Istina je da je to jako inteligentan agent, ali sigurno ne i jednostavan. Slika 5 prikazuje arhitekturu koristi orijentiranog agenta.



Slika 5 Arhitektura koristi orijentiranog agenta

Postoji još arhitektura inteligentnih agenata kao što su slojeviti agent, agent sa deduktivnim rasuđivanjem, agent s praktičnim rasuđivanjem, ovdje su opisane samo najosnovnije arhitekture.

1.2.5. Konkretne arhitekture agenata

Ako se udaljimo od apstraktnog pogleda na agenta i gledamo arhitekture agenata prema načinu odabira akcije gdje akciju definiramo kao

$$akcija: P \rightarrow A$$

Gdje su P – percepti, A – akcije imamo slijedeću podjelu agentske arhitekture. [3]

Agenti temeljeni na logici

Najtradicionalniji način izgradnje agentskih sustava poznat kao i simbolička umjetna inteligencija. Sugerira se da se inteligentno ponašanje može stvoriti u sustavu dajući tom sustavu simbolički prikaz okoline i ponašanja. Odluke ovih agenata su teoremi u nekoj logici. Kao simbolička reprezentacija se koriste logičke formule, a simbolička manipulacija se izvodi dokazivanjem teorema.

Arhitekture reaktivnih agenata

Sredinom 1980-ih znanstvenici koji se bave agentskim sustavima sve više odbacuju simboličku umjetnu inteligenciju te pronalaze alternative simboličkoj paradigmi. Ideja je da se inteligentno, racionalno ponašanje gleda kao urođeno i povezano s agentovim okruženjem. Inteligentno ponašanje se smatra produktom interakcije koju agent ima sa svojom okolinom. Takvo inteligentno ponašanje proizlazi iz interakcije različitih jednostavnijih ponašanja.

Arhitektura tipa uvjerenje-želja-namjera

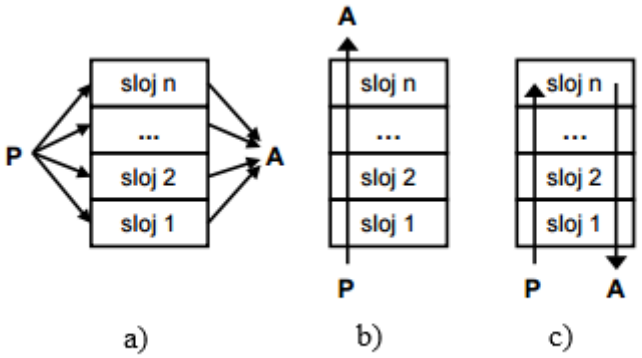
Arhitekture tipa uvjerenje-želja-namjera (*eng. BDI B-Belief, D-Desire, I-Intention*) imaju korijene filozofskoj tradiciji razumijevanja praktičnog zaključivanja. Osnove BDI modela je razvio filozof Michael Bratman [4]. U BDI modelu agent ima: uvjerenje – informacijsku komponentu u kojoj agent pohranjuje znanje o okolini u kojoj djeluje. To znanje stječe primanje informacija iz okoline ili predviđanjem ponašanja okoline. Želje – određeni skup operacija koje agent izvršava ovisno o svom stanju i stanju okoline. Namjere – želje agent može izvršiti, a namjere se odnose na operacije koje se trenutno izvršavaju. Znači, agent generira opcije želja na temelju onoga u što vjeruje, a u smjeru svojih namjera. Namjere stalno nadopunjuje ovisno o trenutnim namjerama, opcijama i vjerovanjima. Neke namjere će generirati nove opcije. Izvršivu namjeru preslikava u akciju. Uvjerenja modificira na temelju percepcija.

Slojevite arhitekture

S obzirom na uvjete da agent mora biti sposoban za reaktivno i proaktivno ponašanje, stvaraju se programski slojevi za različite razine apstrakcije problema. Ova ideja vodi prema klasi arhitekture u kojima su različiti podsustavi raspoređeni u hijerarhiju interaktivnih slojeva. Obično postoje najmanje dva sloja kako bi se lakše nosilo s reaktivnim i proaktivnim ponašanjem. Nema razloga da ne bude i mnogo više slojeva. Općenito govoreći, možemo identificirati dvije vrste protoka upravljanja unutar slojevite arhitekture.(Slika 6)

- U horizontalno slojevitim arhitekturama (Slika 6a), softverski slojevi su svaki izravno povezani s osjetilnim ulazom i akcijama na izlazu. Svaki sloj sam djeluje kao agent stvarajući prijedloge o tome koju će radnju izvršiti.
- U vertikalno slojevitim arhitekturama i osjetilni ulaz i akciju na izlazu obrađuje najviše jedan sloj. Kod vertikalno jednoprolazne slojevite arhitekture (Slika 6b) upravljački tok prolazi slijedno kroz sve slojeve, posljednji sloj generira akciju. Dok kod vertikalno

dvoprolazne slojevite arhitekture (Slika 6c) imamo uspon informacijskog toka do viših razina apstrakcije, pa silazak upravljačkog toka i konkretizacija do akcije.



Slika 6 a) Horizontalne b) Vertikalne jednoprolazne c) Vertikalne dvoprolazne

2. Višeagentski sustavi

Često u literaturi o višeagentskim sustavima naiđemo na slogan da ne postoji sustav samo sa jednim agentom. [2] Bit slogana je da su sustavi stalno u međusobnoj interakciji. Svi, osim trivijalnih običnih sustava koji uglavnom služe za razumijevanje i učenje, sadrže niz podsustava koji međusobno komuniciraju kako bi uspješno obavljali zadatke.

Višeagentski sustav (*eng. Multi-agent system, MAS*) možemo definirati relacijom

$$MAS = (E, O, A, R, Op, L)$$

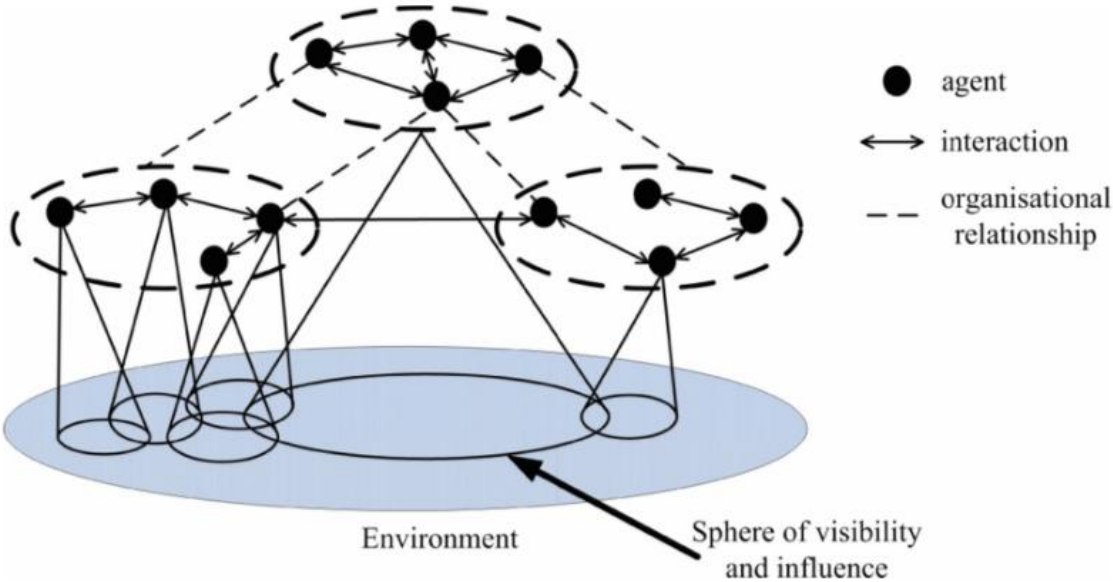
Gdje je E – okolina, O – objekti u okolini, A – agenti, također objekti, R – relacije između objekata, Op – operacije agenata nad objektima, L – zakoni svijeta.

Za sustav možemo reći da je višeagentski ako postoji paralelizam tj. paralelno djelovanje autonomnih agenata koji zajednički pokušavaju ostvariti cilj i ako postoji interakcija tj. komunikacijski protokoli, mehanizmi interakcije s okolinom.

Organizacijsku strukturu višeagentskog sustava čine skup agenata okarakteriziranih ulogama i relacije između agenata. Relacije između agenata mogu biti:

- Poznanstva koja su minimalne relacije između dva agenta
- Komunikacijske veze gdje agent ima mogućnost adresiranja drugog agenta
- Podređenost koja opet može biti ili dinamička ili statička
- Operativna veza gdje je uvjet obavljanja zadatka rezultat rada drugog agenta
- Informacijska veza gdje jedan agent ovisi o znanju drugog agenta
- Konfliktna veza gdje dolazi do konflikta nad resursom
- Natjecateljska veza gdje agenti imaju nekompatibilne ciljeve

Slika 7 [5] prikazuje tipičnu strukturu višeagentskog sustava. Sustav sadrži niz agenata koji komuniciraju jadni s drugima. Agenti su sposobni djelovati u vlastitom okruženju. Različiti agenti imaju različite „sfere utjecaja“, u smislu da imaju kontrolu nad različitim dijelovima okolice. U nekim slučajevima se te sfere podudaraju što znači da dovode do odnosa agenata jedne s drugima. Osim toga, agenti mogu biti povezani i drugim odnosima. Primjerice, odnosima „snage“ pa je jedan agent „šef“ drugome.



Slika 7 Tipična struktura višeagentskog sustava [5]

2.1. Komunikacija u višeagentskim sustavima

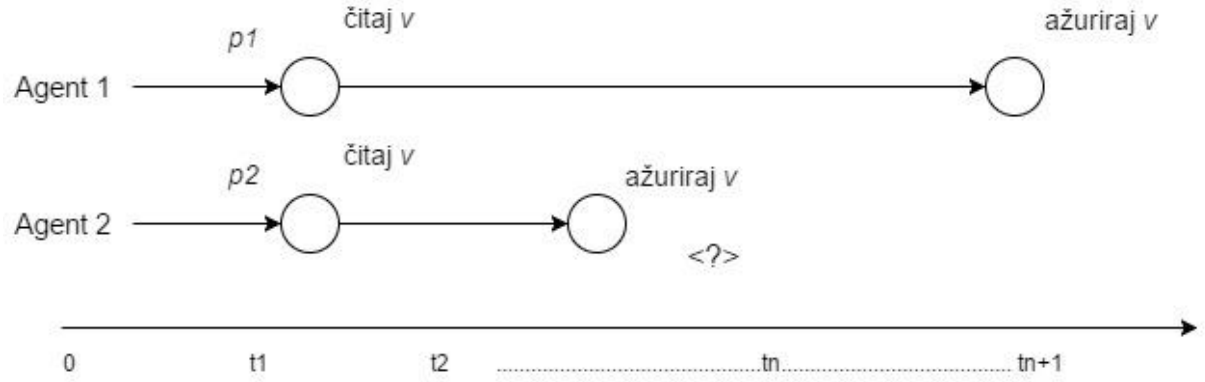
Jako je važan pojam komunikacije između agenta. Osnovni primitiv komunikacije je signal. Signal sam po sebi nema značaj, a propagira se prema nekim pravilima. Da bi se uz signal moglo vezati značenje mora se moći percipirati i interpretirati.

Do sada su razvijeni i mnogi komunikacijski protokoli kako bi se olakšala komunikacija u višeagentskim sustavima. Razlikujemo različite kategorije komunikacije:

- Veza pošiljatelj – primatelj – od točke do točke i broadcasting
- Priroda komunikacijskog medija – može biti kategorizirana na direktno usmjeravanje (direktno slanje poruke primatelju, sadržaj je skriven od neadresiranih agenata), usmjeravanje propagacijom signala (signal prema okolici i intenzitet padaju s povećanjem udaljenosti), usmjeravanje javnom porukom (oglasna ploča)
- Namjera za komunikacijom – agent izražava želju ili odluku za djelovanjem
- Incidentna komunikacija – poruka je odaslana nenamjerno, npr. tragovi, interpretacija je proizvoljna i ovisi o primatelju

Dva procesa trebaju biti sinkronizirana ukoliko postoji mogućnost da bi jedan mogao naštetiti drugome. Klasičan primjer takvog problema je scenarij „izgubljenog ažuriranja“, gdje dolazi do interferencije dva procesa i neispravnog ažuriranja vrijednosti neke varijable. Ako imamo agenta *a1* sa procesom *p1* i agenta *a2* sa procesom *p2*, oba agenta žele pristupiti dijeljenoj

varijabli *v*. Agent *a1* započinje proces mijenjanja vrijednosti varijable *v*, pročita trenutnu vrijednost, mijenja vrijednost varijable i na kraju novu vrijednost spremi u varijablu *v*. Između čitanja i spremanja nove vrijednosti koje izvršava agent *a1*, agent *a2* je promijenio vrijednost varijable *v* spremajući u nju neku vrijednost. Kada agent *a1* izvrši svoje spremanje izmjena koju je napravio agent *a2* je izgubljena. Na slici (Slika 8) je prikazana interferencija dva procesa.



Slika 8 Interferencija dva procesa

Sichman i suradnici 1994. godine iznose jedan način razumijevanja interakcije između agenata u višeagentskim sustavima. Osnovna ideja je da postoji odnos između dva agenta ako jedan od agenata zahtijeva naredbu od drugoga da bi postigao svoj cilj. Vodeći se tom mišlju postoji niz mogućih odnosa interakcije:

- Neovisna. Ne postoji ovisnost između dva agenta
- Jednostrana. Jedan agent ovisi o drugome, ali ne obrnuto
- Uzajamna. Oba agenta međusobno ovise jedan o drugome da bi postigli isti cilj
- Recipročna ovisnost. Prvi agent ovisi o drugome da bi postigao neki cilj, isto tako drugi ovisi o prvome da bi postigao neki, ne nužno isti cilj.

Agenti komuniciraju na višoj razini, jezici za komunikaciju su temeljeni na teoriji govornih činova. Na slici (Slika 9) je prikazan komunikacijski koncept u višeagentskim sustavima.

Slika 9 Komunikacijski koncept

2.1.1. Jezici za komunikaciju u višeagentskim sustavima

Prvo ćemo krenuti s jezicima, obzirom na važnost komunikacije u višeagentskim sustavima razvijaju se mnogi komunikacijski protokoli i jezici za komunikaciju da bi komunikacija među agentima u sustavima bila što bolja. Na razvijanje takvih jezika najviše su utjecale govor i formalni jezik i teorije vezane za njih (*eng. speech act theories*). Govorne teorije komunikaciju tretiraju kao akciju što temelje na pretpostavci da se akcije govora izvršavaju od strane agenata kao i bilo koje druge akcije koje dovode agenta do njegovog cilja.

U ranim 1990.-ima DARPA (*eng. Defence Advanced Research Projects Agency*) je osnovala KSE (*eng. Knowledge Sharing Effort*) s ciljem razvijanja protokola za razmjenu znanja između autonomnih sustava. [2] Od tada se razvilo mnogo jezika za komunikaciju, ovdje su samo predstavljeni neki od najpoznatijih.

2.1.1.1. KIF (eng. Knowledge Interchange Format)

KIF je razvijen s ciljem omogućavanja razmjene znanja među udaljenim računalnim sustavima. Bio je zamišljen kao jezik koji bi izrazio sadržaj poruke, a ne samu poruku. Baziran je na logici prvog reda. Koristeći KIF agent može izraziti: [2]

- Svojstva nekoga ili nečega. Na primjer, *Ivan je vegetarijanac. – Ivan ima svojstvo vegetarijanstva.*
- Odnose. Na primjer, *Ivan i Ana u braku. – Odnos braka postoji između Ivana i Ane.*

- Opća svojstva. Na primjer, *svi imaju majku.*

Relacija između dva objekta, napisana u jeziku KIF je prikazana u kodu (Kod 1). Označava cijena od m1 je 9 kuna. Definiramo novi koncept na način kako je prikazano u kodu. (Kod 2). Definiramo novi koncept *mlade* na način da je to *zena* koja nije *udana.* Relacije između dva svojstava definiramo na način prikazan u kodu (Kod 3). Osoba sa svojstvom da je *dijete* isto tako ima svojstvo i da je *covjek.*

```
(= (cijena m1) (scalar 9 kuna))
```

Kod 1 Relacija između 2 objekta

```
(defrelation mlada (?x) :=
      (and (zena ?x)
         (not (udana ?x))))
```

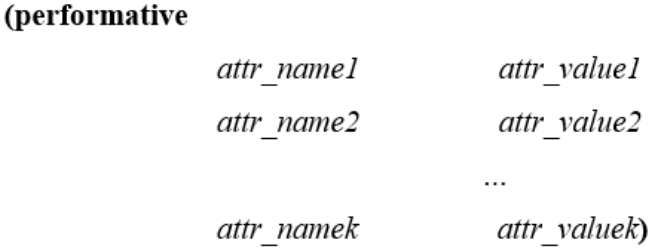
Kod 2 Definiranje novog koncepta

```
(defrelation (dijete ?x) :=> (covjek ?X))
```

Kod 3 Relacije između svojstava

2.1.1.2. KQML (eng. Knowledge Query Manipulation Manipulation Language)

KQML je porukama orijentiran (*eng. message-based*) jezik za komunikaciju između agenata. [2] KQML definira zajednički format poruka. Grubo gledajući, KQML poruka se može smatrati objektom (u smislu objektno orijentiranog programiranja). Svaka poruka (Slika 10) je instanca neke klase (*performative*) te poruka i ima određeni broj parametara (atributa). Tipovi performative i atributi su prikazani u tablicama (Tablica 1, Tablica 2)



Slika 10 Struktura KQML poruke

Kategorija performative	**Ime**
Obični upiti	`evaluate, ask-if, ask-one, ask-all, ask-about`
Višestruki odgovori (upiti)	`stream-about, stream-all, eos, subscribe`
Odgovori	`reply, sorry`
Generičke informacije	`tell, achieve, cancel, untell, unachieve`
Generiranja	`standby, ready, next, rest, discard, generator`
Definiranje kapaciteta	`advertise, recommend, recruit, broker monitor, import, export`
Umrežavanja	`register, unregister, forward, broadcast, route`

Tablica 1 Kategorije performativa

Parametar	**Značenje**
`:content`	Sadržaj poruke
`:in-reply-to`	Očekuje se labela u odgovoru
`:language`	Ime jezika (npr. KIF, Prolog)
`:ontology`	Ime ontologije
`:receiver`	Primatelj poruke
`:reply-with`	Očekuje li pošiljatelj odgovor, i ako očekuje, identifikator za odgovor
`:sender`	Pošiljatelj poruke

Tablica 2 Parametri poruke

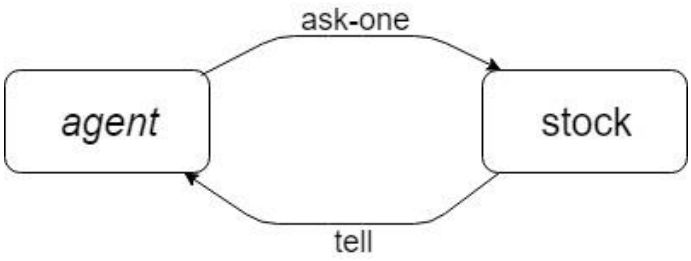
```
(ask-one
      :sender agent
      :content "price(ibm, Price)"
      :receiver stock
      :reply_with stock_query
      :language Prolog
      :ontology NYSE-TICKS
)
```

Kod 4 Primjer KQML poruke sa performativom ask-one

Na primjer, za jednostavnu komunikaciju između dva agenta, gdje prvi agent pita drugog neku informaciju, a drugi mu odgovara s traženom informacijom. Imamo kod (Kod 4), iz njega možemo lako pročitati što poruka radi. Performativa poruke je *ask*-one. Pošiljatelj *agent* pita primatelja *stock* za cijenu *ibm*, očekuje odgovor, sve u jeziku *Prolog* i ontologiji *NYSE-TICKS*. Odgovor na tu poruku također je prikazan u kodu (Kod 5). Performativa poruke je *tell*. Pošiljatelj je *stock,* primatelj *agent,* sadržaj je poruka *price(ibm, 123),* kao odgovor na *stock_query*, također u jeziku *Prolog* i ontologiji *NYSE-TICKS*. Način komunikacije između dva agenta je prikazan na slici (Slika 11)

```
(tell
      :sender stock
      :content "price(ibm, 123)"
      :receiver agent
      :in_reply_to stock_query
      :language Prolog
      :ontology NYSE-TICKS
)
```

Kod 5 Primjer KQML poruke s performativom tell



Slika 11 Komunikacija KQML porukama

KQML ima određene nedostatke, u početku nije imao formalnu semantiku, mehanizmi za prijenos poruka između agenata nikada nisu točno definirani. Fokus je bio na transferu znanja, ostvarivao je samo zahtjeve za nekom akcijom. KQML nije prikladan za distribuirano rješavanje problema. Svi ti nedostatci popravljaju se u FIPA jeziku.

2.1.1.3. FIPA ACL (eng. FIPA agent communication language)

FIPA (*eng. Foundations of Intelligent Physical Agents*) je agencija koja još 1995. počinje razvijati standarde za agentske sustave. FIPA komunikacijski jezik (Slika 13) sličan je KQML-u. Struktura poruka je ista, polja atributa jako slična. Najvažnija razlika između njih je u kolekciji performativa koje pružaju. Na slici (Slika 12) vidimo performative FIPA komunikacijskog jezika. S obzirom da je jedna od najčešćih i najozbiljnijih kritika KQML jezika bila nedostatak adekvatne semantike, FIPA-a je osjetila važnost pružanja sveobuhvatne formalne semantike u svom jeziku.

performative	passing info	requesting info	negotiation	performing actions	error handling
accept-proposal			x		
agree				x	
cancel		x		x	
cfp			x		
confirm	x				
disconfirm	x				
failure					x
inform	x				
inform-if	x				
inform-ref	x				
not-understood					x
propose			x		
query-if		x			
query-ref		x			
refuse				x	
reject-proposal			x		
request				x	
request-when				x	
request-whenever				x	
subscribe		x			

Slika 12 FIPA ACL performative

```
(inform
      :sender agent1
      :receiver agent2
      :content (price good 123)
      :language si
      :ontology auction
)
```

Slika 13 Primjer FIPA ACL poruke

2.1.2. Ontologija

Po slici (Slika 9) koja prikazuje komunikaciju višeagentskih sustava, jedina nepoznanica koja nam je još ostala je ontologija. Ako dva agenta žele komunicirati o nečemu trebaju imati zajedničku terminologiju koja opisuje njihov predmet komunikacije. Ako dva agenta komuniciraju o temperaturi, moraju oba znati što im ta temperatura predstavlja. Isto tako biti upoznati sa pojmovima Kelvin i Celzij. Ontologija je specijalizirana za taj set pravila. Znači ontologija je reprezentacija znanja kojom su opisane veze između činjenica i koncepata. Jezici

za definiranje ontologija zasnivaju se na nekom od logičkih formalizama. Na primjer logici prvog reda kao što smo spomenuli za jezik KIF. KIF jezik opisan u poglavlju 2.1.1.1. je primjer jezika za definiranje ontologija. KIF ontologija poznata i kao Ontolingua server je servis baziran na webu na kojem se mogu dijeliti ontologije razvijene od različitih autora, a isto tako i postići pregled svih ontologija.

2.2. Kooperacija i pregovaranje

Pojedine zadatke agenti mogu riješiti sami, dok za neke zadatke je potrebna interakcija s drugim agentima. Ciljevi agenata mogu biti kompatibilni ili nekompatibilni. Ciljevi su nekompatibilni ako postizanje jednog cilja onemogućava postizanje drugog. Ukoliko su ciljevi agenata kompatibilni dolazi do kooperacije. Ferber 1999. kaže da agenti surađuju ako učinkovitost raste tj. dodavanjem novog agenta povećava se grupna učinkovitost i ako se izbjegavaju konflikti i to ne samo da agenti izbjegavaju potencijalne konflikte već i rješavaju nastale.

Kooperaciju sačinjava raspodjela zadataka i resursa, koordinacija akcija i rješavanje konflikata (hijerarhija, pregovori). Povećanje sposobnosti preživljavanja, povećanje učinkovitosti i rješavanje konflikta su osnovne zadaće kooperacije.

Tehnike kooperacije su:

- Fizičko grupiranje agenata
- Komunikacija
- Specijalizacija – proces prilagodbe agenta njegovom zadatku
- Dijeljenje zadataka i resursa
- Koordinacija
- Rješavanje konflikta

Prednosti kooperativnog višeagentskog sustava su mnoge: obavljanje inače neizvedivog zadatka, povećanje produktivnosti svakog agenta, bolje korištenje resursa.

Ograničeni resursi dovode do sukoba i remećenja odnosa, najmoćniji mehanizam za upravljanje višeagentskim sustavima je pregovaranje. Pregovaranje je oblik interakcije u kojem grupa agenata s konfliktnim interesima i sa željom za kooperacijom pokušava postići uzajamno prihvatljiv sporazum o podjeli oskudnih resursa. [6] Svaki agent želi postići sporazum radije nego da pregovaranje završi bez sporazuma ipak svaki agent želi postići za sebe najpoželjniji sporazum.

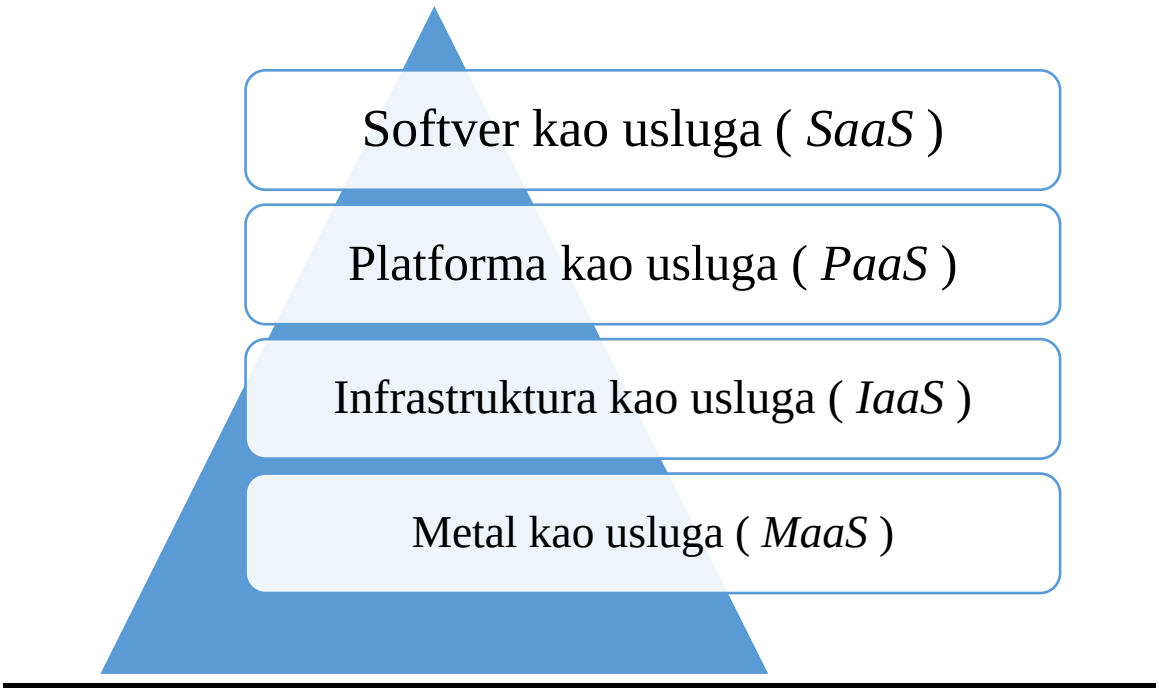
Razlikujemo različite vrste pregovaranja po broju agenata uključenih u pregovaranje pa tako jedan agent može pregovarati samo s jednim agentom, više agenata može pregovarati s jednim agentom i više agenata može istovremeno pregovarati s više agenta. Pregovarački agent mora biti u stanju predlagati svoje odgovore prihvatljive za postizanje sporazuma drugim agentima i odgovoriti na prijedloge drugih agenata prihvaćanjem ili odbijanjem njihovog prijedloga. Tako možemo zaključiti da pregovarački agent ima prijedloge, kritike i protuprijedloge i time može pokazati što želi. Ipak takvo pregovaranje između agenata bi moglo trajati jako dugo i biti neefikasno pa tako agenti nerijetko u komunikaciji s drugim agentima koriste i argumente. Argumentima agenti mogu opravdati svoje stajalište ili uvjeriti drugog agenta u svoje stajalište. Argumentima pokušavamo postići brži dogovor između pregovaračkih agenata.

3. Računarstvo u oblaku

Upotreba računala u današnje vrijeme je gotovo neizostavna. Svjedoci smo stalnog napretka tehnologije pa tako i samog korištenja računala. S razvojem tehnologije dolazi i do razvoja računarstva u oblaku (eng. *cloud computing*). Prema IBM-u, računarstvo u oblaku se jednostavno odnosi na „oblak“ (eng. *the cloud*), a označava dostavu računalnih resursa onda kada su potrebni korisniku, računalni resursi se korisniku dostavljaju putem interneta te mu se naplaćuju onoliko koliko ih koristi (eng. *pay-for-use*) [7]. Broj definicija i interpretacija računarstva u oblaku je iznimno velik, ali sve se svodi na ideju o iznajmljivanju računalnih resursa prema potrebi korisnika. Računalni resursi mogu bit određene komponente računala, različiti programi ili podaci. Nekoliko godina unazad, kada bismo koristili neku aplikaciju , ona se izvršavala na našem uređaju, bilo da je riječ o osobnom računalu, mobitelu ili tabletu. Razvojem računarstva u oblaku, to više nije čest slučaj, danas se aplikacije najčešće izvršavaju na jednom ili više servera s kojima smo povezani putem interneta, umjesto našeg lokalnog uređaja kako smo navikli. Kako bi to bilo moguće, prvo ti serveri moraju negdje zaista i postojati. Postoje tvrtke koje se bave gradnjom i održavanjem podatkovnih centara (eng. *data centers*). Njihov posao nije niti malo jednostavan jer je održavanje takvih centara poprilično zahtjevno, ali i skupo. Prije svega, trebalo bi pronaći prihvatljivu lokaciju za gradnju jednog takvog centra, područje ne bi trebalo bit podložno čestim potresima, poplavama ili nekim sličnim vremenskim nepogodama. Pronalaskom lokacije nastavlja se mukotrpan posao. Treba izgraditi prostorije, opremiti ih namještajem, strujom, jako dobrim hlađenjem i internetom od nekoliko desetaka gigabita. Kada se ti osnovni uvjeti ispune, potrebno je dobiti određene certifikate, od poslovnih pa sve do sigurnosnih, zatim je potrebno zaposliti ljude i bit spreman na promjenu servera i/ili diskova svakih tri do pet godina. Završetkom spomenutog procesa, tvrtka može ponuditi korisniku svoje usluge.

3.1. Slojevi računarstva u oblaku

Usluge se dijele na nekoliko razina, korisniku mogu biti ponuđena samo računala uključena u struju i internet, ali također mu se može ponudit i korisnički račun pomoću kojeg se može služit određenim aplikacijama. Tako razlikujemo nekoliko modela usluga u oblaku (Slika 14) [8].



Slika 14 Slojevi računarstva u oblaku

3.1.1. Metal kao usluga

Metal kao usluga (eng. *Metal as a Service*) je usluga u oblaku koja korisniku pruža računalo uključeno u struju i spojeno na internet. Korisnik ne mora brinuti gdje je računalo smješteno, ima li pristup internetu, hladi li se ispravno i hoće li nestati struje. S druge strane, korisnik je zadužen za cjelokupno podešavanje računala, počevši od BIOS-a i IP adrese pa nadalje. Točnije, zadužen je za konfiguraciju operacijskog sustava i svega iznad njega. Za ovakvu vrstu usluge obično se opredjeljuju napredni korisnici koji posjeduju određeno znanje i vještine za osposobljavanje računala kako bi obavljalo željene zadatke. Dobra strana metala kao usluge je iznimno kratko vrijeme potrebno za instalaciju operacijskog sustava te jednostavna instalacija operacijskog sustava na velikom broju računala istovremeno [7].

3.1.2. Infrastruktura kao usluga

Infrastruktura kao usluga (eng. *Infrastructure as a Service*) korisniku nudi računalo s instaliranim operacijskim sustavom, korisnik plaća korištenje procesorskih jezgri, RAM-a, tvrdog diska i ostalih komponenti računala, ali i uslugu njihove konfiguracije i to sve do samog operacijskog sustava. Dakle, više nije potrebno ručno konfigurirati IP adrese nego nam ih izabere i dodijeli sustav za upravljanje oblakom. Zadaća svakog korisnika je da samostalno održava operacijski sustav, instalira željene aplikacije i brine o njima.

3.1.3. Platforma kao usluga

Platforma kao usluga (eng. *Platform as a Service*) svojim korisnicima nudi skladištenje i održavanje računala te kompletno postavljanje i održavanje operacijskog sustava što korisniku dodatno smanjuje posao. Ovakva vrsta usluge uz operacijski sustav pruža i nekoliko osnovnih serverskih aplikacija kao što su mail server, baza podataka i/ili web server. Stoga, korisnikova zadaća je u ovom slučaju da se bavi instaliranjem i održavanjem aplikacija koje su mu konkretno potrebne za rad koji želi obavit. Preciznije, platforma kao usluga nudi okruženje koje ima sve potrebno za razvoj i isporuku web aplikacija [7].

3.1.4. Softver kao usluga

Sofver kao usluga (eng. Software as a Service) je najpopularniji model usluge u oblaku. Korisnik softvera kao usluge ima svoj korisnički račun pomoću kojega pristupa aplikacijama koje želi koristiti. Aplikacije koje se nalaze na korisničkom računu instalira i održava poslužitelj dok korisnik ne mora znati što se sve događa u pozadini. Ovakvu vrstu usluge najbolje implementiraju Google i Facebook. Danas je vjerojatno gotovo svaki korisnik računala ujedno i korisnik ove vrste usluge, dovoljno je da imate Gmail korisnički račun i već koristite Googleov softver kao uslugu [8].

3.2. Vrste oblaka prema pravima pristupa

Usluge u oblaku možemo razvrstati i u ovisnosti o tome tko im može pristupiti, tako razlikujemo tri vrste oblaka [9]:

1. Privatni oblak: U vlasništvu je neke organizacije, koja pruža hostingom usluge korisnicima unutar organizacije.
2. Javni oblak: Nije u vlasništvo organizacije, pruženim uslugama u tim oblacima može pristupiti bilo koja organizacija.
3. Hibridni oblak: U hibridnom oblaku, usluge se nude ograničenom i definiranom broju korisnika.

3.3. Prednosti računarstva u oblaku

Pet najvećih prednosti računarstva u oblaku su: [10]

1. Oporavak. Svaki mudar poduzetnik ima backup najvažnijih podataka. Implementacija backupa i oporavka temeljena na oblaku štedi puno vremena i živaca.
2. Ušteda. Zašto kupiti skupe hardvere kada se usluga iste kvalitete može lako postići po nižoj cijeni.
3. Brainstorming. Mnogo ljudi, mnoge ideje na jednom mjestu. Svaki član tima može pristupiti, dijeliti i uređivati dokumente u bilo koje vrijeme. Ovo rješenje omogućuje zajednički rad uz potpuni pregled te suradnje.
4. Sigurnost. Sa računarstvom u oblaku vaši dokumenti su dostupni s bilo kojeg uređaja koje ste izabrali. Velika je prednost i mogućnost da izbrišete svoje podatke na daljinu izravno s korisničkog računa, tako da nikako ne mogu doći u krive ruke.
5. Rad na udaljenost. Bilo da se radi o učenju ili o zaposleniku koji može raditi i biti efektivan jednako i u uredu i iz vlastite kuće. Možda, i o zaposlenicima koji se ne nalaze u istom gradu. Moderan model rada može se ostvariti upravo zahvaljujući računarstvu u oblaku. Vaši podaci su dostupni za vas u bilo koje vrijeme, bilo gdje, iz bilo kojeg odabranog uređaja.

4. Realizacija web aplikacije kao višeagentski sustav

4.1. Problem

U današnje vrijeme, doba digitalizacije, cijelo poslovanje se pokušava prebaciti s običnog papira i olovke na računalo. Ljudi počinju shvaćati prednosti digitalnih sustava i koliko im oni mogu pomoći u napretku poslovanja. Dolazi do mnogih promjena uvjetovanih razvojem tehnologija, te promjene zahvaćaju sve industrije poslovanja, tako da se tvrtke koje žele ostati uspješne moraju prilagoditi digitalnom vremenu.

Motivi zbog kojih tvrtke pristaju na digitalizaciju mogu biti bitno različiti, ali ono što je svima zajedničko je razmišljanje o poslovanju sutrašnjice. Sve više tvrtki susreće se s velikim količinama dokumenata u kojima se gube i vlasnici i pravnici. Digitalizacija u potpunosti mijenja i svijet medija. Promjene kojima svjedočimo nisu promjene digitalne strategije poslovanja, nego te promjene traže izgradnju poslovne strategije svake kompanije za digitalno doba.

Ako gledamo primjere oko nas obrti kao što su objekti brze hrane, prijevozi, trgovine već doživljavaju velike promjene u obliku različitih web aplikacija koje im pomažu u poslovanju. Potaknuti tim trendom web aplikacija koje pomažu ne samo velikim tvrtkama, nego i malim obrtima dolazimo do problema upravljanja resursima. Po definiciji koja kaže da upravljanje resursima (resursni menadžment) u organizacijskim znanostima označava učinkovitu i djelotvornu implementaciju organizacijskih resursa kada postoji potreba za njima. U te resurse mogu spadati financijski resursi, inventar, ljudske vještine, proizvodni resursi ili informacijska tehnologija (IT) [11] vidimo da je to jedan složeni pojam u kojem digitalizacija igra značajnu ulogu. Mnogi vlasnici ističu kako veliki problem u poslovanju predstavlja upravljanje ljudskim vještinama koje se često zbog financijskih resursa prebacuje u drugi plan. Zbog te potrebe radimo web aplikaciju u oblaku za upravljanje resursima. Sustav je prvenstveno prilagođen frizerskim i kozmetičkim salonima, sportskim objektima i dječjim igraonicama. Ipak uz minimalne promjene sustav se može prilagoditi i drugim profesijama. Sustav se razvijao u suradnji s jednim frizerskim salonom te je nazvan Planfection.

4.2. Planfection

Planfection je sustav u oblaku koji tvrtkama omogućava jednostavno kreiranje rasporeda s vlastitim zaposlenicima, uslugama, cijenama i klijentima. Pomoću baze klijenata i statističkih

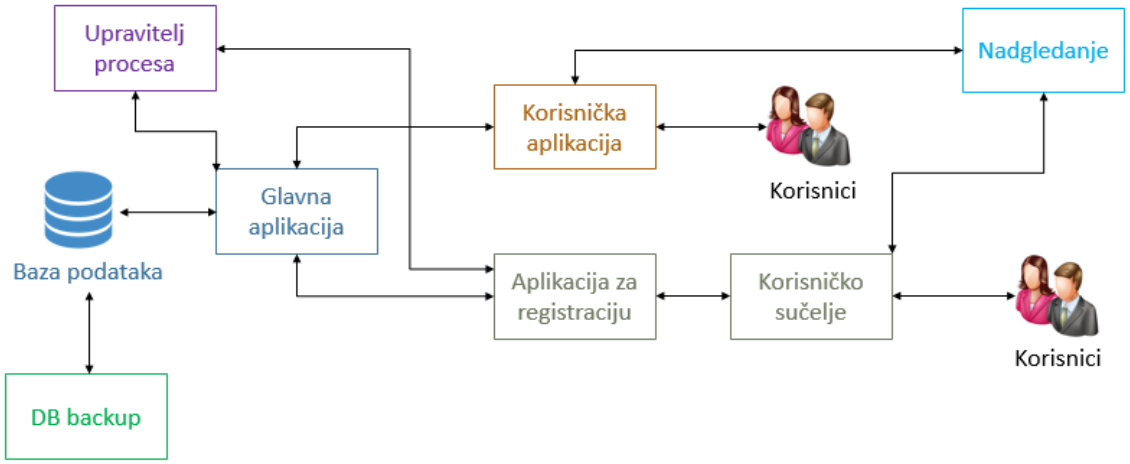
podataka, korisnicima je skraćeno vrijeme potrebno za upravljanje resursima uz povećanje kvalitete pružanja usluga. Prilagođen je svim uređajima od osobnih računala do tableta i mobitela. Sustav se sastoji od nekoliko aplikacija u oblaku koje međusobno komuniciraju. Trenutno broji oko 70 registriranih tvrtki, 80 korisnika (zaposlenici tvrtke), 3500 termina i 5000 pruženih usluga. Isto tako, nakon što je sustav razvijen da bi dobili i vanjsko vrednovanje sustava nevezano za njegove korisnike sustav je prijavljen na Poduzetnički kamp 2016. u Solinu gdje osvaja drugo mjesto.

Sustav Planfection implementiran je kao web aplikacija u oblaku i to softver kao usluga. Korisnik se samostalno registrira za korištenje aplikacije, na uneseni mail dobiva sve potrebne informacije o sustavu te može započeti s korištenjem.

Sustavu se pristupa inkrementalnom metodologijom s obzirom na suradnju s krajnjim korisnikom koji sudjeluje u svim fazama razvoja. Sustav se radi u modulima to jest inkrementima koji su jedan o drugome neovisni. Svaki modul je isproban i odobren od krajnjeg korisnika. Isto tako u sustav se uvijek mogu dodati i novi moduli.

4.2.1. Arhitektura programske podrške

Arhitektura programske podrške se može promatrati kao mikroservisna arhitektura. Mikroservisna arhitektura je princip razvoja aplikacija u obliku malih, izdvojenih servisa, pri čemu svaki servis ima svoj proces i ostvaruje komunikaciju putem jednostavnih mehanizama kao što je HTTP API. [12] Na slici (Slika 15) je prikazana arhitektura programske podrške sustava koja se sastoji od baze podataka, glavne aplikacije, aplikacije za registraciju, korisničke aplikacije, korisničkog sučelja, upravitelja procesa, nadgledanja te backupa baze podataka.



Slika 15 Arhitektura programske podrške

Sustav možemo promatrati kao višeagentski sustav u kojem autonomni agenti zajednički pokušavaju ostvariti kreiranje rasporeda i upravljanje resursima. Isto tako agenti su u interakciji tj. između njih postoje komunikacijski protokoli. Okolina sustava je server kao i web preglednici. Glavna aplikacija, aplikacija za registraciju, korisnička aplikacija, korisnička sučelja, upravitelj procesa, nadgledanje te backupa baze podataka predstavljaju agente. Agenti u ovom sustavu su reaktivni, uočavaju svoje okruženje i pravovremeno odgovaraju na promjene koje se u njemu pojavljuju. Agenti su društveno sposobni komuniciraju s drugim aplikacijama, procesima i bazom podataka. Okruženje agenata je determinističko, svaka akcija ima jedan zajamčeni učinak. Isto tako okruženje agenata je nepristupačno i dinamičko s obzirom da je okruženje agenta Internet koje je jako dinamičko okruženje.

Iznimno izražena je hijerarhijska arhitektura višeagentskog sustava, s obzirom da samo jedan agent komunicira s bazom podataka. Dok svi ostali agenti komuniciraju s njim. Isto tako često imamo recipročnu ovisnost agenata. Na primjer upis ili promjena podatka u bazi podataka je recipročna ovisnost agenta glavne aplikacije i agenta korisničke aplikacije. S obzirom da samo agent glavne aplikacije komunicira s bazom podataka, a agent korisničke aplikacije samo komunicira s korisnikom koji te promjene želi napraviti. S obzirom da je sustav sastavljen od više modula imamo i slojevitu arhitekturu sustava i to dvoprolazno vertikalnu jer jedan sloj (agent) primi podražaj dok drugi sloj (agent) odgovara na taj podražaj. Relacije među agentima su komunikacijske veze, agenti točno mogu adresirati drugog agenta kojemu šalju poruku.

Glavna aplikacija je reaktivni agent kojim komunicira s drugim aplikacijama, konkretno aplikacijom za registraciju i korisničkom aplikacijom, upraviteljem procesa te bazom podataka. Percepte glavne aplikacije predstavljaju rute koje služe za komunikaciju, akcije predstavljaju dodavanje, spremanje, brisanje itd. Na slici (Slika 16) je prikazan primjer komunikacije glavne aplikacije s korisničkom aplikacijom. Ruta */api/addClient* predstavlja percept glavne aplikacije na koji agent poziva akciju *AddClientManualy*.

```
app.post('/api/addClient',function(req,res){

  // allow cross-domain request
  CrossDomain.HandleCrossDomainAPI(req,res,CompanyApi.AddClientManualy);
});
```

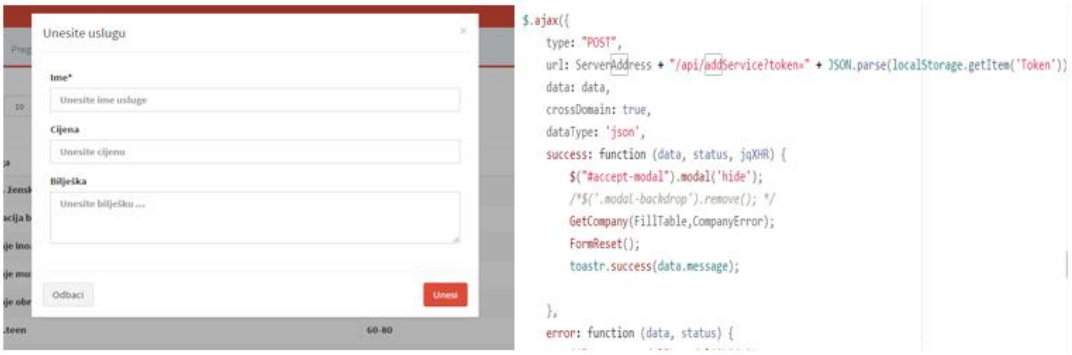
Slika 16 Komunikacija glavne aplikacije s korisničkom aplikacijom

Aplikacija za registraciju je reaktivan agent koji prima podražaje od korisnika putem korisničkog sučelja. Odgovara na te podražaje putem AJAX zahtjeva koji pozivaju API-je glavne aplikacije.

Glavna aplikacija i aplikacija za registraciju su implementirane pomoću Node.js [13]. Node.js je okvir za pisanje web aplikacija koji je napravljen na Google Chromeovom JavaScript Engineu (V8). Dolazi s okruženjem pomoću kojega se skripte napisane u programskom jeziku JavaScript mogu interpretirati i izvršavati, što omogućava izvršavanje JavaScript koda na bilo kojem uređaju izvan internet preglednika kako je to uobičajeno. Rezultat navedenih mogućnosti je pisanje serverskog koda web aplikacija koristeći se JavaScript programskim jezikom. Node.js dodatno omogućava korištenje širokog spektra biblioteka koje pojednostavljuju razvoj, testiranje i održavanje programske podrške. Svi API-ji Node.js biblioteka su asinkroni što znači da server nikada ne čeka da neki API vrati podatke, odnosno nema blokiranja servera. Svaki put kada Node.js pozove određeni API, odmah prijeđe na poziv slijedećeg API-ja, a mehanizam za manipulaciju događajima pomaže serveru da dobije odgovor od prethodnog poziva API-ja. Prednost Node.jsa je iznimno velika brzina izvođenja što je djelomično rezultat asinkronosti. Node.js se izvršava na jednoj niti te može odgovoriti na veliki broj zahtijeva u usporedbi sa serverima napisanim u drugim tehnologijama. Node.js je posebno dobro koristiti za razvoj aplikacija unutar kojih više ljudi obavlja neki zajednički rad, kao što je u ovom slučaju izrada rasporeda te upravljanje resursima tvrtke. Još jedna u nizu prednosti je što programski kod u cijeloj aplikaciji može bit pisan jednim programskim jezikom, tj. JavaScriptom. Kako bi bilo moguće u potpunosti se zadržati na JavaScriptu, potrebno je adekvatno odabrati bazu podataka koja omogućava pisanje koda koristeći se JavaScript programskim jezikom. Dobar izbor takve baze podataka je MongoDB [14]. Riječ je o bazi podataka koja se temelji na dokumentima. Dokumenti sadrže polja u obliku ključ-vrijednost. Ti dokumenti se pohranjuju u kolekcije. Kao što relacijske baze podataka obično koriste tablice, tako MongoDB koristi kolekcije. Prednost MongoDBa je fleksibilnost modela podataka, odnosno dinamička struktura dokumenata. Upiti su znatno brži u usporedbi s tradicionalnim bazama podataka te je izražena visoka skalabilnost koja može bit od velikog značaja u budućnosti aplikacije.

Korisnička aplikacija predstavlja reaktivnog agenta. Agent ima svoje percepte i akcije kojima odgovara na percepte. Korisnička aplikacija komunicira s glavnom aplikacijom, vanjskim sustavom nadgledanja i korisnicima. Korisnička aplikacija prima podražaje od korisnika putem formi i raznih akcija na korisničkom sučelju, a odgovara na te podražaje AJAX zahtjevima koji pozivaju API-je glavne aplikacije. Na slici (Slika 17) je prikazana komunikacija korisničke

aplikacije s korisničkim sučeljem gdje podražajem, klikom na botun *Unesi* dolazi do akcije pozivanja API-ja. Zatim, glavna aplikacija prima podražaj putem rute */api/addService* i izvršava svoje akcije.



Slika 17 Komunikacija korisničke aplikacije s korisničkim sučeljem

Upravitelj procesa upravlja stanjima aplikacije može je započeti, zaustaviti, ponovno pokrenuti ili brisati procese. Reaktivni je agent kojemu percepte predstavljaju protokoli za komunikaciju sa serverom pomoću kojih dobiva informacije o stanju procesa (aplikacije), dok na te percepte odgovara akcijom ponovnog pokretanja sustava u slučaju greške. Realiziran je pomoću PM2 [15]. PM2 je upravitelj procesa za Node.js. PM2 izravno komunicira s glavnom aplikacijom i aplikacijom za registraciju. Na slici (Slika 18) je prikazana komunikacija upravitelja procesa s glavnom aplikacijom. Upravitelj procesa, u ovom slučaju PM2 promatra datoteku server.js, u slučaju greške, promjene na toj datoteci ponovno pokreće sustav.

```
pm2 start server.js
```

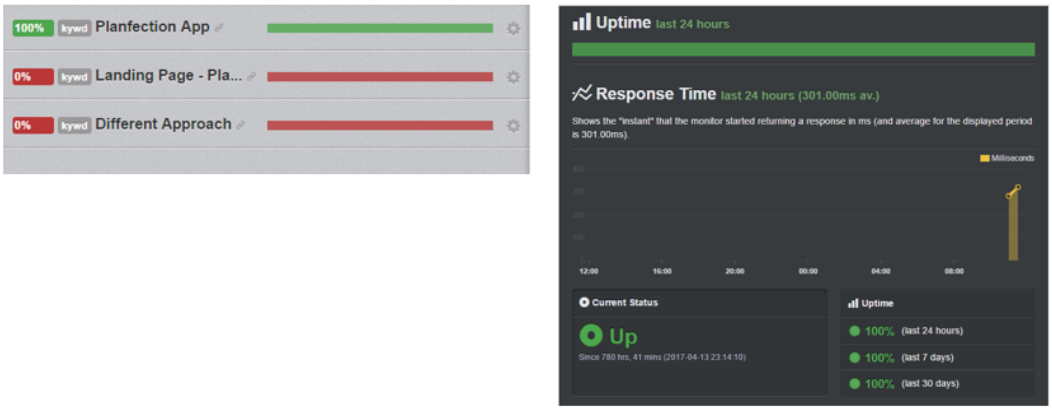
Slika 18 Komunikacija upravitelja procesa s glavnom aplikacijom

Na slici (Slika 19) su prikazani svi procesi odnosno aplikacije kojima upravlja upravitelj procesa PM2. Svi procesi su dva puta pokrenuti tako da u slučaju prekida rada jednog procesa, drugi može preuzeti izvršavanje aplikacije.

App name	id	mode	pid	status	restart	uptime	cpu	mem	watching
da_web_page	2	fork	10230	online	3866714	0s	33%	21.7 MB	disabled
da_web_page	6	fork	516	online	0	33D	0%	35.5 MB	disabled
jug	3	fork	10216	online	2072870	1s	66%	29.9 MB	disabled
jug	7	fork	14360	online	73295	32D	0%	39.2 MB	disabled
landing_page	1	fork	379	online	0	33D	0%	41.4 MB	disabled
landing_page	5	fork	10236	online	3895912	0s	33%	15.1 MB	disabled
server	0	fork	10202	online	2158042	1s	49%	27.6 MB	disabled
server	4	fork	504	online	0	33D	0%	46.6 MB	disabled

Slika 19 Stanje na serveru

Nadgledanje aplikacije predstavlja još jednog reaktivnog agenta. Ovdje su nadgledaju korisnička aplikacija i korisničko sučelje aplikacije za registraciju. Agente s kojima komunicira nadgledanje su agenti koji primaju podražaje izravno od korisnika. Akcije koje ovakvo nadgledanje može obavljati su upućivanje novih zahtjeva za daljnjim nadgledanjem i javljanje administratoru sustava da postoji greška u sustavu. Bitno je istaknuti kako se to odnosi na greške na korisničkom sučelju što je i razlog komunikacije nadgledanja samo s korisničkom aplikacijom i korisničkim sučeljem aplikacije za registraciju. Nadgledanje je implementirano sustavom UptimeRobot [16]. UptimeRobot svakih 5 minuta provjerava header koji dobiva kao odgovor na zahtjev za komunikaciju sa sustavom, npr. Ako je status kod 200-ok, čeka sljedećih 5 minuta i upućuje novi zahtjev. Ako je status kod 400+ i 500+ onda sustav ne radi, provjeri još par puta u 30 sekundi je li još uvijek isto stanje, ako se nije promijenilo javlja na određeni mail da postoji pogreška u sustavu. Slika 20 prikazuje nadgledanje sustava.



Slika 20 Nadgledanje sustava

Backup baze podataka još jedan reaktivni agent je realiziran obliku skripte koja se pomoću CentOSa automatski pokreće svaka 4 sata. Njegova jedina zadaća je da napravi kopiju podataka iz baze podataka tako da ukoliko dođe do pogreške i gubitka podataka postoji mogućnost njihovog povratka.

4.2.2. Komunikacija

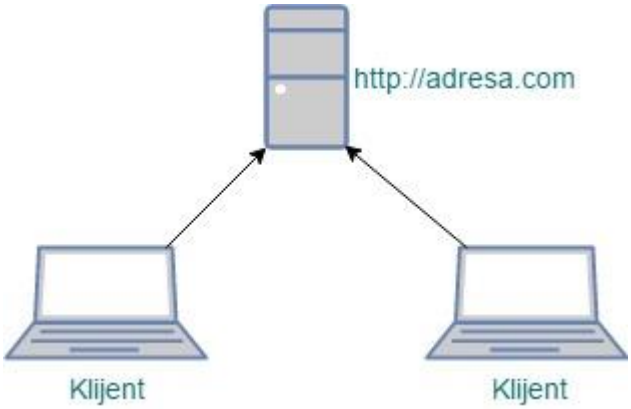
Krenimo od samih protokola. Protokoli su dogovorena pravila o komunikaciji. Taj skup protokola koji čine Internet objavljuje IETF (*Internet Engineering Task Force*). Misija IETF-a je poboljšanje rada Interneta proizvodnjom kvalitetnih, relevantnih tehničkih dokumenata koji utječu na način na koji ljudi oblikuju, koriste i upravljaju Internetom. [17] IETF objavljuje zahtjeve za protokolima nazvane RFC-ovi (*eng. Request for Comments*). Protokoli obuhvaćaju programske jezike, operacijske sustave i hardversku arhitekturu. Omogućuju komponentama napisanim od strane različitih ljudi i komponentama koje upravljaju različitim agentima da komuniciraju jedne s drugima preko cijelog svijeta. [18] Dva protokola prevladavaju kada govorimo o komunikaciji na Internetu. Internet protokol (IP) koji je odgovoran za jednostavno prenošenje paketa između dva domaćina na Internetu i protokol prijenosa (*eng. Transmission Control Protocol,* TCP) koji se može promatrati kao cijev koja se proteže preko cijelog Interneta i ima pouzdani tok bajtova u svakom smjeru između dvije krajnje točke. Zajedno TCP preko IP (TCP / IP) su jezgra transportnog sloja bezbroj mrežnih aplikacija.

Projekt World Wide Web je 1991. godine prvi put objavljen za javnost. Web je sustav hipertekstualni dokumenata pomoću URL-ova (*Universal Resource Locators*). Slovo U u imenu URL predstavlja univerzalno što ukazuje na tada revolucionarnu ideju da bi svi hipertekstualni dokumenti mogli biti međusobno povezani. Web veze na HTML dokumentima prema drugim dokumentima vežu se URL-ovima. Zato je smisleno da je web protokol prilagođen za takvo dohvaćanje resursa. HTTP (*Hypertext Transfer Protocol*) je aplikacijski protokol za distribuirane, suradničke i hipermedijske informacijske sustave. HTTP je temelj podatkovne komunikacije za World Wide Web.

Najranije web aplikacije koriste web obrasce i stalno ponovno učitavanje stranica. Svaki put kada korisnik želi poslati ispunjeni obrazac, preglednik sprema podatke i učitava novu stranicu. Svaki put kada su ažurirani podaci na stranici korisnik je mora osvježiti za dohvaćanje promijenjenog izvora pomoću HTTP-a.

Uz pomoć JavaScripta i XMLHttpRequest API razvijen je skup tehnika zvanih AJAX kako bi se omogućile bolje aplikacije bez naglih prijelaza tijekom svake interakcije. AJAX dopušta aplikacijama da dohvaćaju samo potrebne podatke i ažurira stranice bez ponovnog osvježavanja. S AJAX-om mrežni protokol je još uvijek HTTP.

HTTP radi dosta dobro za interakcije koje pokreću klijentske aplikacije, budući da klijent pokreće svaki HTTP zahtjev kako je prikazano na slici (Slika 21)



Slika 21 HTTP klijenti povezani na web server

S vremenom, HTTP je napravio ugrađenu podršku za tekst za podržavanje međusobno povezanih HTML stranica, URL-ove i HTTPS, zaštićeni HTTP.

Cijela komunikacija u sustavu je napravljena pomoću HTTP API-ja. Gledajući sa strane višeagentske komunikacije, ciljevi agenata su kompatibilni pa lako dolazi do kooperacije. S obzirom da samo agent glavne aplikacije ima pristup bazi podataka, isto tako agent korisničke aplikacije jedini prima podražaje od korisnika izbjegnuta je borba za resurse. Tehnikom kooperacije, komunikacijom napravljena je dobra raspodjela zadataka i resursa te koordinacija akcija.

Podaci unutar sustava su prezentirani JSON (*eng. JavaScript Object Notation*) formatom. Riječ je o standardnom formatu za prezentiranje podataka u web aplikacijama. Dodatna, olakotna okolnost je što je cijeli sustav pisan koristeći JavaScript programski jezik te je JSON format prirodno podržan. JSON format je izabran zbog tržišnih standarda te jednostavnog uklapanja u tehnologije u kojima je razvijen cijeli sustav. Da se koristi neki jezik za komunikaciju u višeagentskim sustavima, dodatno bi bilo potrebno razviti prevoditelj kako bi sve aplikacije mogle razumjeti poruke putem kojih komuniciraju. To je kompleksan posao koji oduzima dosta vremena i bespotrebno troši resurse. Za komunikaciju u sustavu se koristi HTTP protokol. Poruke unutar sustava su slične porukama koje se koriste u drugim jezicima za komunikaciju u višeagentskim sustavima. Slično kao kod KQML-a, svojstva poruke su u formatu ključ-vrijednost, samo što kod KQML-a ključevi su već prije određeni performativi. Navedeni su

primatelj i pošiljatelj, a informacije u poruci su objekt i to u JSON formatu. Na slici (Slika 22) je prikazana usporedba KQML-a i JSON-a.

```
(ask-one
      :sender agent
      :content "price(ibm, Price)"
      :receiver stock
      :reply_with stock_query
      :language Prolog
      :ontology NYSE-TICKS
)
```

```
{
      „type" : „ask-one",
      „sender" : „agent",
      „content" : „price(ibm, Price)",
      „reciver" : „stock",
      „reply_with" : „stock_query",
      „language" : „Prolog",
      „ontology": „NYSE-TICKS"
}
```

Slika 22 Usporedba KQML-a i JSON-a

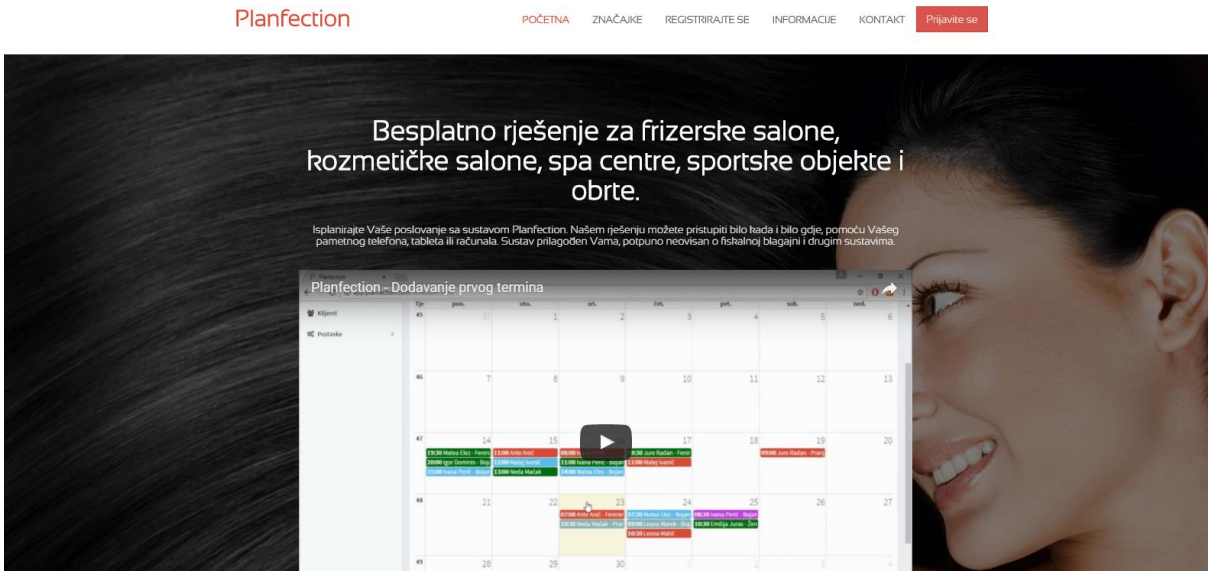
Kako ne bi bilo neželjenih poruka sustav koristi tokene za autentifikaciju. Za generiranje i provjeru tokena se koristi biblioteka JSON Web Tokens [19]. Prilikom svake prijave u sustav korisnik dobiva token putem kojega mu je omogućena komunikacija unutar sustava. U sustav je ugrađena provjera tako da obrađuje samo s dozvoljenih domena.

4.2.3. Moduli sustava

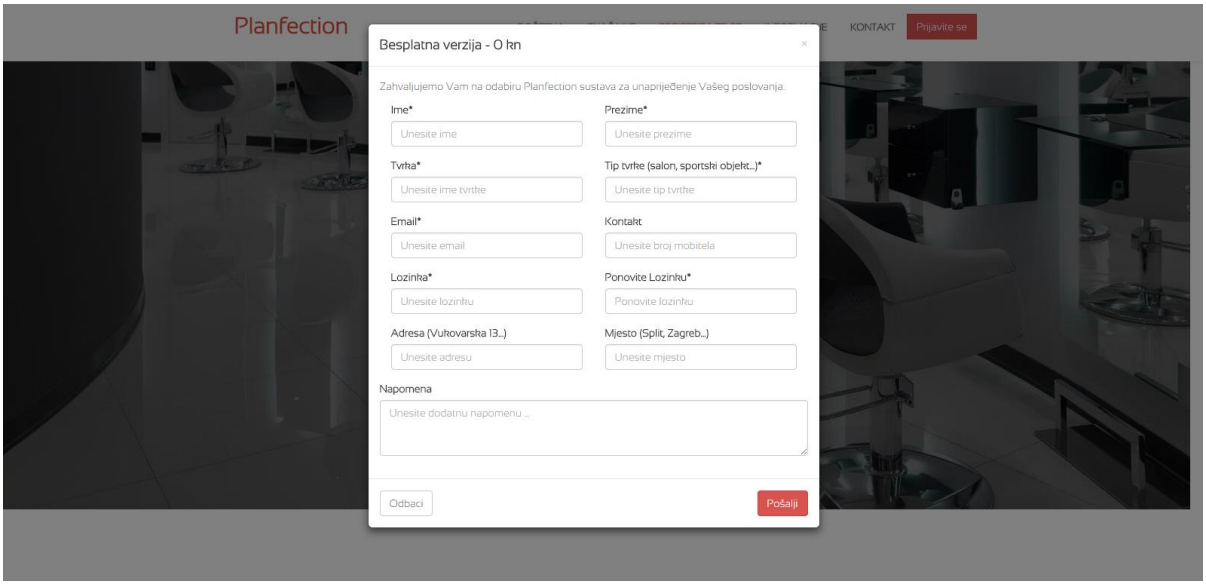
Kako je korištena inkrementalna metodologija za izradu sustava tako se sustav sastoji od više modula. Svaki modul je neovisan dio sustava i kao takav se može mijenjati i prilagođavati drugim zahtjevima korisnika. Svaki modul je izrađen u suradnji s strukom kako bi što više pokrio potrebe realnih korisnika. Validacija za svaki modul vrši i agent korisničkog sučelja i agent glavne aplikacije. U slučaju greške svaki modul generira detaljne povratne informacije. Sustav se sastoji od sljedećih modula.

Uvodna stranica

Uvodna stranica sustava osim što daje sve potrebne podatke o sustavu i predstavlja sami sustav nudi mogućnost registriranja novih korisnika, kao i vezu za prijavu postojećih korisnika. Autentifikacija se vrši putem maila i lozinke. Novi korisnici koji se registriraju dobiju mail s potrebnim informacijama i uputama. Uvodna stranica sadrži i kontakt formu za sve upite o sustavu. Slika 23 i Slika 24 prikazuju uvodnu stranicu.



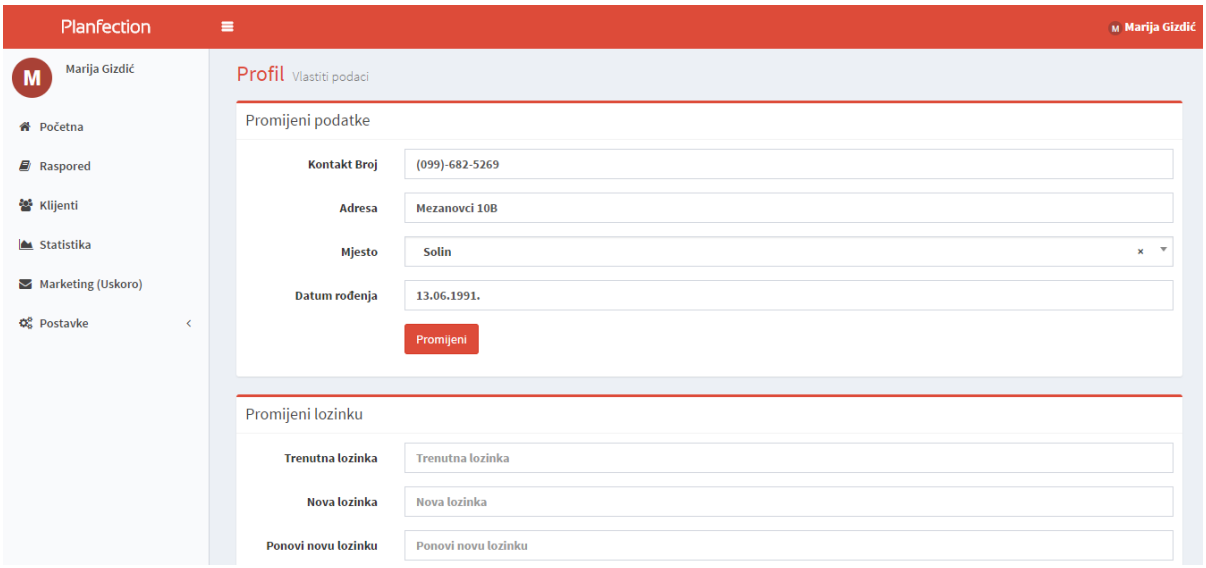
Slika 23 Uvodna stranica



Slika 24 Registracija na uvodnoj stranici

Profil korisnika

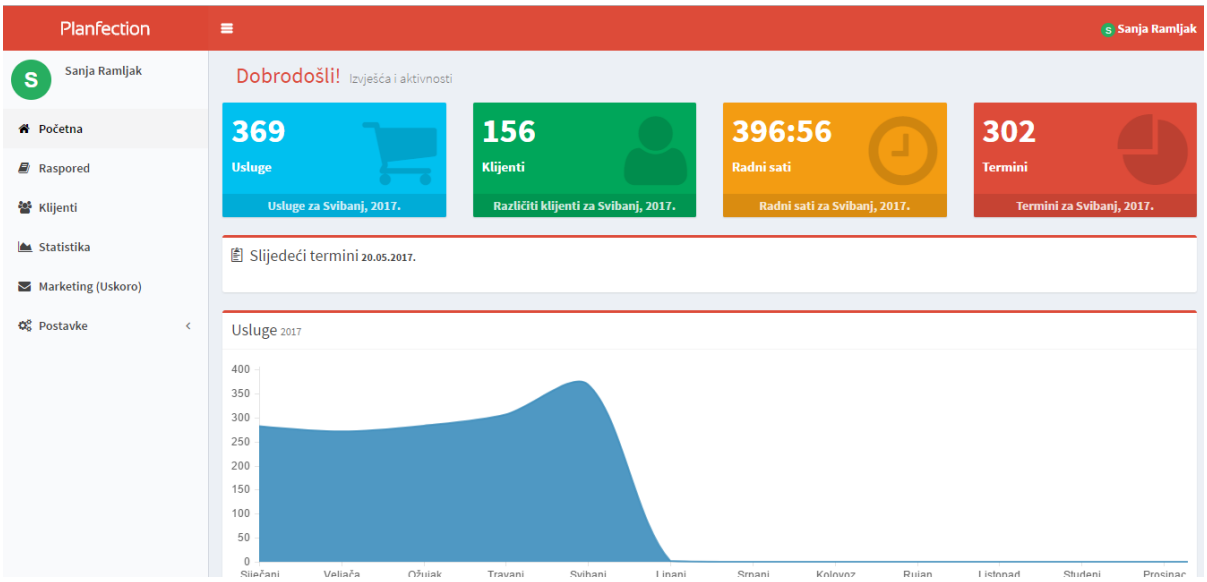
Korisnik nakon prijave može mijenjati svoje osobne podatke, isto tako može mijenjati lozinku koju je unio prilikom registracije. Te funkcije su obuhvaćene u modulu profil korisnika koje je prikazan na slici (Slika 25).



Slika 25 Profil korisnika

Početna stranica

Korisnik na početnoj stranici može vidjeti statističke podatke za tekući mjesec (broj usluga, klijenata, radnih sati i termina). Isto tako ima uvid u nadolazeće termine i klijente s odbrojavanjem u realnom vremenu. Grafički prikaz statističkih podataka za tekuću godinu je isto tako dostupan na početnoj stranici koja je prikazana na slici (Slika 26)

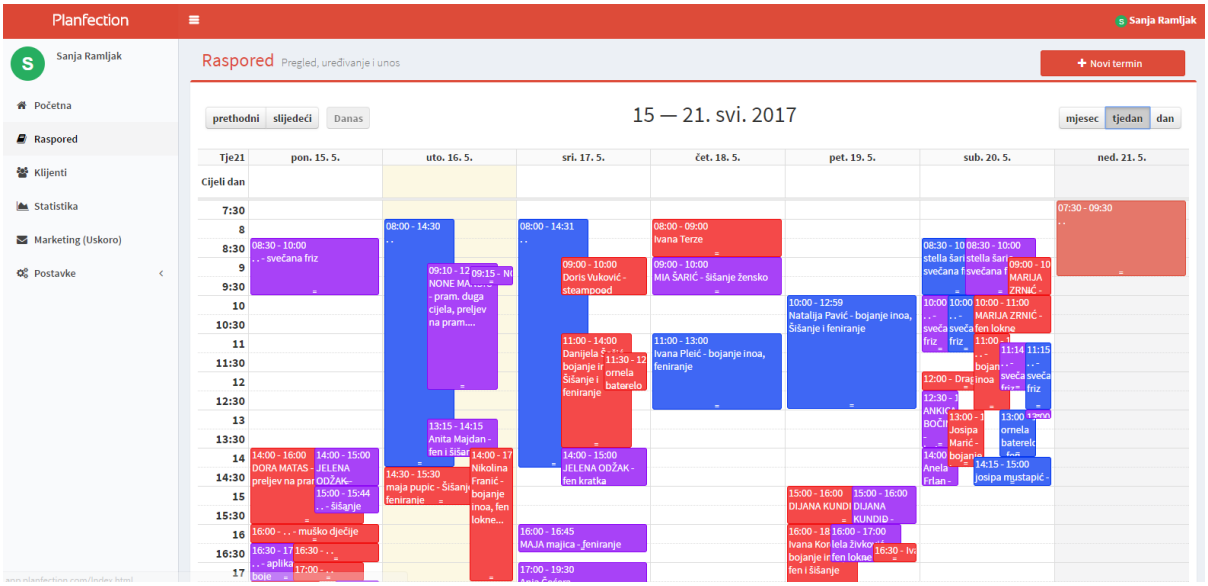


Slika 26 Početna stranica

Raspored

U modulu raspored korisnik ima mogućnost pregleda rasporeda po mjesecu, tjednu, danu i satu, pregled rasporeda po zaposleniku (za dan), unošenje novih termina, uređivanje postojećih

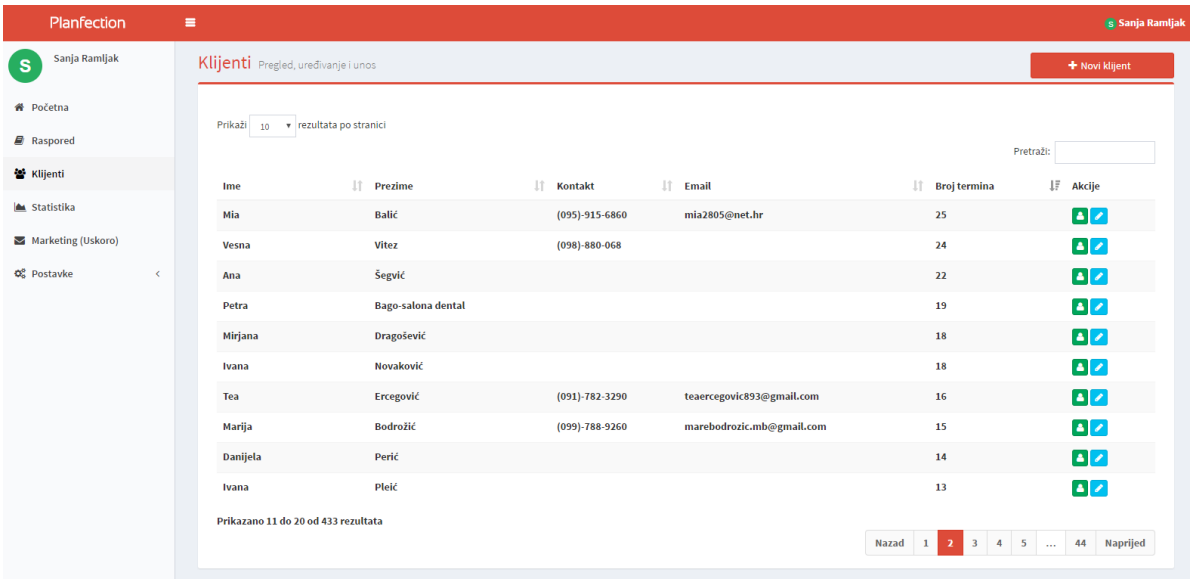
termina, dodavanje novog klijenta izravno iz forme za unos termina, grafički prikaz rasporeda, kalendara i sata prilikom unosa/uređivanja termina. Slika 27 prikazuje grafički prikaz rasporeda po tjednu, satu i zaposleniku. Zaposlenike označavaju boje.



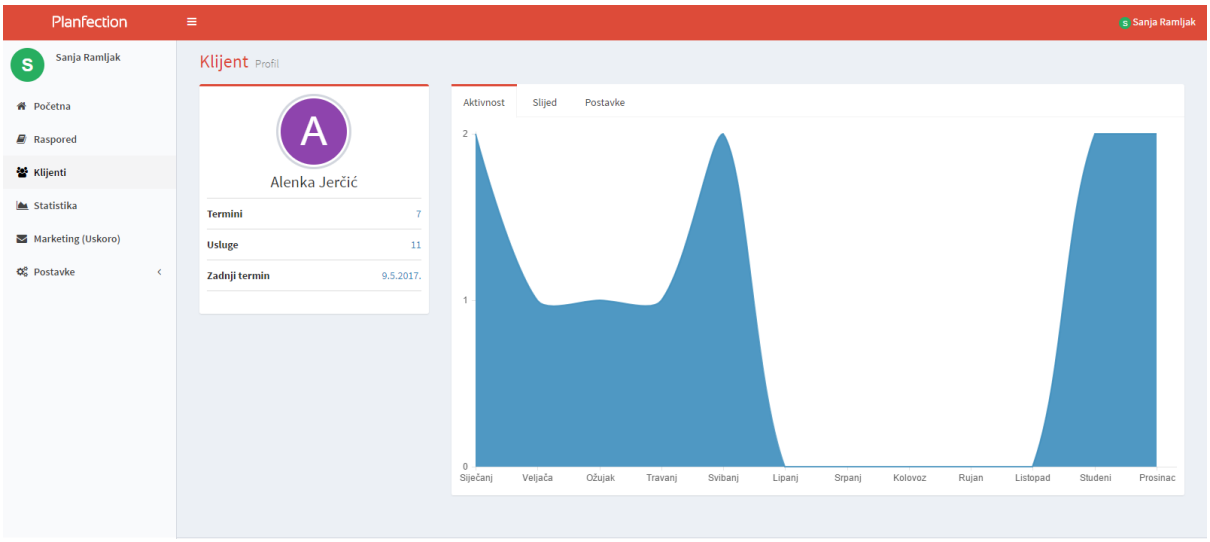
Slika 27 Raspored

Klijenti

U modulu klijenti je korisniku omogućen tablični prikaz klijenata, sortiranja klijenata prema bilo kojem unesenom podatku, pretraga klijenata prema bilo kojem podatku, unošenje i uređivanje klijenata. Svaki klijent ima svoj profil prikazan na slici (Slika 29) gdje su prikazani statistički podaci o klijentu kao što su broj termina, usluga i prikaz zadnjeg termina, grafički prikaz aktivnosti klijenta te povijest svih termina odabranog klijenta s datumom, satom, uslugom i zaposlenikom koji je napravio tu uslugu. Slika 28 prikazuje tablični prikaz klijenata.



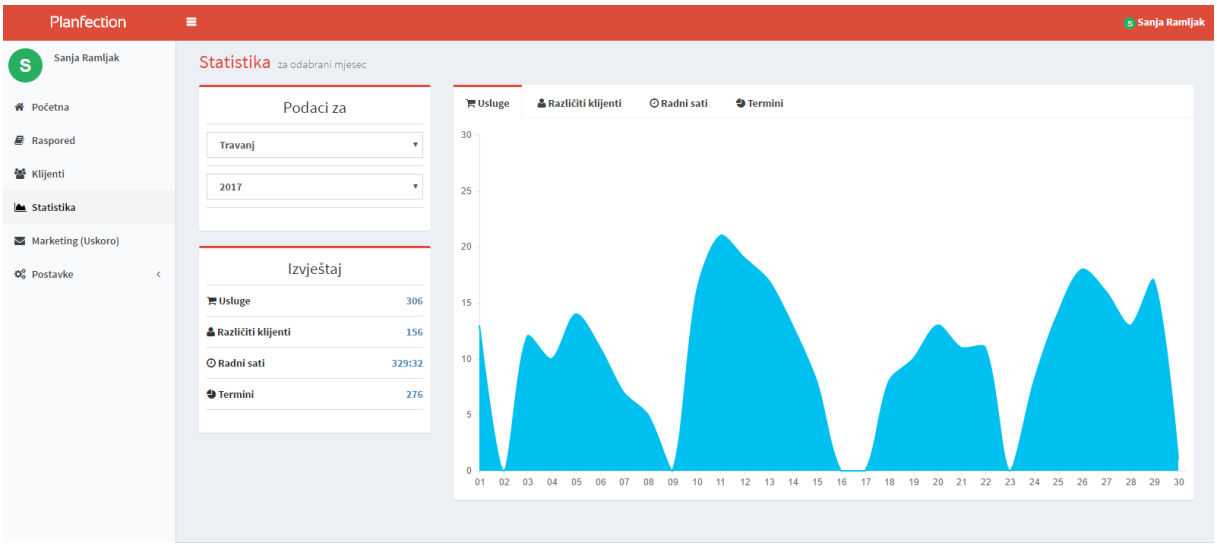
Slika 28 Klijenti



Slika 29 Profil klijenta

Statistika

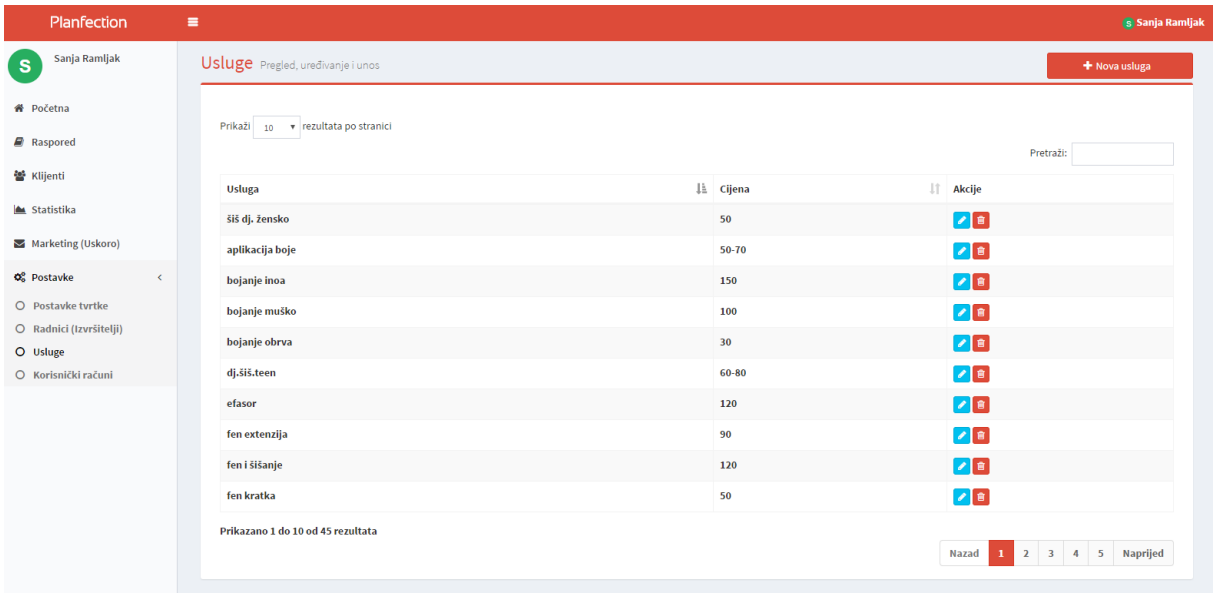
Korisnici sustava mogu pratiti i statistiku svog poslovanja pomoću modula statistika. Ovaj modul pruža detaljan pregled usluga, klijenata, radnih sati i termina prema odabranom mjesecu i godini. Isto tako korisnici imaju izvješće za svaki mjesec. Slika 30 prikazuje statistiku za mjesec travanj 2017. godine.



Slika 30 Statistika

Postavke tvrtke/aplikacije

Korisnik prilikom registracije unosi samo osnovne podatke o sebi i svojoj tvrtki. U modulu postavke tvrtke postavlja aplikaciju za ispravno korištenje. Tako tu uređuje podatke o tvrtki (email, kontakt broj, web stranica i radno vrijeme), unosi, uređuje, pregledava i briše zaposlenike, usluge i korisničke račune. Slika 31 prikazuje tablični prikaz usluga.



Slika 31 Postavke usluga

4.2.4. Mogućnosti nadogradnje sustava

S obzirom na modularnost sustava, sustav je jednostavno nadograditi sa mnogim opcijama koje se krajnjem korisniku učine korisne za njegovo poslovanje. Može se nadograditi sustavom za naplatu, u tom slučaju korisniku za poslovanje ne bi bio potreban nijedan drugi sustav osim ovoga. Sustav se može nadograditi i detaljnim praćenjem skladišta, tada bi za svaku uslugu korisnik trebao unijeti materijal i sustav bi mogao pratiti trenutno stanje skladišta. Korisnik bi u tom slučaju unosio i inventure i prijam i odbitak robe.

Jedna zanimljiva nadogradnja ovakvog sustava bila bi u vidu inteligentnog, koristi orijentiranog agenta koji bi predlagao kreiranje rasporeda. Takav agent bi modelirao i pratio svoje okruženje, zadatke koji uključuju veliku količinu istraživanja o percepciji, obrazloženju i učenju odnosno o trenutnom stanju rasporeda. Ako još tog agenta zamislimo i u BDI arhitekturi, agent bi mogao pametno predlagati najbolje vrijeme za pojedine usluge. Agent bi imao uvjerenje tj. informacije o okolini, o samom rasporedu. Želje akcije koje agent planira izvršiti i namjere akcije koje agent trenutno izvršava. S takvim agentom, bi korisnici uštedjeli na vremenu koje potroše kreiranjem rasporeda gdje moraju dobro rasporediti zaposlenike, usluge i želje klijenata.

Klijentska aplikacija je još jedan način nadogradnje ovog sustava. Klijentska aplikacija kao neovisna aplikacija bi služila za klijente tvrtki koje koriste sustav Planfection. Klijenti bi tako imali uvid u tvrtke koje pružaju tražene usluge, mogli bi se naručiti na odabranu uslugu. Isto tako klijenti bi mogli pratiti kada su gdje bili i na kojoj usluzi, mogli bi zabilježiti koliko su bili zadovoljni dobivenom uslugom. Dok bi se tvrtke kroz takvu vrstu aplikacije mogle i reklamirati klijentima. Na taj način, takvom klijentskom aplikacijom, ne bi bili zadovoljni samo klijenti već i trenutni korisnici sustava Planfection.

Zaključak

Upravljanje resursima u poslovanju česta je tema mnogih poduzetničkih skupova. U današnje vrijeme, doba digitalizacije svaki se poduzetnik koji misli na sutrašnjicu svoje tvrtke želi uključiti u proces digitalizacije. Ovakav sustav za upravljanje resursima donosi upravo uključivanje malih tvrtki i malih poduzetnika u proces digitalizacije. Ne samo to, ovakav sustav donosi izvrstan primjer sustava koji može poslužiti kao osnova za nadogradnju složenijeg sustava za vođenje cijelog poslovanja.

Konkretan primjer implementiranog sustava donosi spoj višeagentskih sustava i računarstva u oblaku stoga može poslužiti i kao primjer kod podučavanja i jednog i drugog koncepta. S jedne strane agentski utemeljena paradigma sa svim svojim pojmovima i temeljima za nadogradnju. S druge strane, web aplikacija u oblaku s točno specificiranim arhitekturama i programskom podrškom.

Budući radovi u ovom području mogu biti usmjereni prema poboljšanju svakog agenta i uključivanje inteligentnih agenata sve više u svijet web aplikacija.

Slike

Tablice

Kod

Literatura

[1] Stuart J. Russell and Peter Norvig, Artificial Intelligence A Modern Approach Third Edition, 2010.

[2] M. WOOLDRIDGE, An Introduction to MultiAgent Systems, 2009.

[3] M. Wooldridge, »Intelligent Agents: The Key Concepts,« 2002.

[4] M. Bratman, Intention, Plans, and Practical Reason, 1987.

[5] »ResearchGate,« [Mrežno]. Available: https://www.researchgate.net/figure/260623380_fig21_Typical-structure-of-a-multiagent-system.

[6] D. Walton, E. Krabbe, Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning, 1995.

[7] IBM, »IBM Cloud,« 11 12 2016. [Mrežno]. Available: https://www.ibm.com/cloud-computing/learn-more/what-is-cloud-computing.

[8] »Računarstvo u oblaku,« *Mreža,* 2013.

[9] C. JOHN PAUL Dr. R. SANTHI, »A Study of E-Learning in Cloud Computing,« *International Journal of Advanced Research in Computer Science and Software Engineering,* 2014.

[10] T. Wiśniewski, »elearningindustry.com,« 2016. [Mrežno].

[11] »https://hr.wikipedia.org/wiki/Upravljanje_resursima,« [Mrežno].

[12] M. Fowler, »https://www.thoughtworks.com/,« [Mrežno].

[13] [Mrežno]. Available: https://nodejs.org/en/.

[14] [Mrežno]. Available: https://www.mongodb.com/.

[15] [Mrežno]. Available: http://pm2.keymetrics.io/.

[16] [Mrežno]. Available: https://uptimerobot.com/.

[17] »https://www.ietf.org/,« [Mrežno].

[18] Frank Salim, Peter Moskovits, Paul S. Wang, The Definitive Guide to HTML5 WebSocket, 2013.

[19] [Mrežno]. Available: https://jwt.io/.

[20] R. Rajagopal, Introduction to Microsoft Windows NT Cluster Server: Programming and Administration, 1999.

[21] »http://www.yourdictionary.com/multiprocessor,« [Mrežno].

[22] S. G. Shiva, Computer Organization, Design, and Architecture, 2007.

Sažetak

Agent je sve što može promatrat svoju okolinu pomoću senzora i djelovati na tu okolinu. Višeagentski sustav je sustav sastavljen od barem dva agenta koja međusobno komuniciraju kako bi izvršili složene zadatke. Računarstvo u oblaku označava dostavu računalnih resursa onda kada su potrebni korisniku, računalni resursi se korisniku dostavljaju putem interneta te mu se naplaćuju onoliko koliko ih koristi. Cilj je uzimajući u obzir temeljne pojmove agenata i višeagentskih sustava razmotriti konkretnu web aplikaciju u oblaku.

Ključne riječi: agent, višeagentski sustav, računarstvo u oblaku

Summary

Agent can be anything that can observe environment by using sensors and acting on that environment. The multi-agent system is a system composed of at least two agents which communicate with each other in order to perform complex tasks. Cloud computing is a term that describes delivering computer resources to the user, computer resources are delivered to the user via the internet and charged as much as they are used. The goal is to take into account the basic concepts of agents and multi-agent systems to consider a specific web application in the cloud.

Keywords: Agent, Multi-agent system, cloud computing