Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/06BE46Miladinovic

GENERISANJE FORMI ZA POSLOVNE PROCESE KORIŠĆENJEM *MAGRITTE* RAZVOJNOG OKVIRA

FORMS GENERATION FOR BUSINESS PROCESSES USING *MAGRITTE* FRAMEWORK

Nina Miladinović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – SOFTVERSKO INŽENJERSTVO

Kratak sadržaj – *U radu su opisani poslovni procesi, karakteristike formi za unos podataka i programski jezik Smalltalk i njegov dijalekt Pharo. Opisani su razvojni okviri neophodni za implementaciju rešenja. Predstavljena su postojeća rešenja uz akcenat na delovima koji se tiču generisanja formi za unos podataka. Predstavljeno je implementirano rešenje, prikazane su klase koje su nosioci rada aplikacije i prikazano je dobijeno rešenje.*

Ključne reči: *Poslovni procesi, generisanje formi, Pharo programski jezik, Magritte razvojni okvir*

Abstract – *The thesis describes business processes, characteristics of data entry forms and the programming language Smalltalk and its dialect Pharo. Frameworks needed to be implemented into the solution are described. Existing solutions are presented, with an emphasis on parts concerning the generation of data entry forms. The implemented solution is presented, the classes that are the carriers of the application work are shown and screenshots of developed application are shown.*

Keywords: *process engine, generating forms, smalltalk – Pharo, Magritte framework*

1. UVOD

Poslovni proces predstavlja niz aktivnosti koje su potrebne da se izvrše kako bi se postigao jasno definisani cilj. Ove aktivnosti mogu biti izvršene na razne načine, od strane različitih učesnika. Ljudi kao učesnici i dalje su neophodni u izvršavanju određenih zadataka i to je deo koji još uvek ne može da bude automatizovan. Ovo dovodi do zahteva za korisničkim interfejsom koji će omogućiti korisnicima da izvršavaju potrebne zadatke. Zadatak ovog rada jeste projektovanje *web* aplikacije za upravljanje zadacima korisnika u poslovnim procesima. Ovo zahteva i logovanje korisnika, pregled i odabir zadataka i generisanje formi za popunjavanje podacima.

Rešenje je potrebno implementirati u *Pharo*[1] programskom jeziku uz upotrebu *Magritte*[2] razvojnog okvira. Ono treba da predstavi proširenje namenjeno *NewWave*[3] softveru, treba da predstavi proširenje namenjeno *NewWave* softveru, koji u *Pharo* programskom jeziku implementira funkcije potrebne za praćenje i upravljanje poslovnim procesima.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bila dr Gordana Milosavljević, vanr. prof.

[1] https://pharo.org/
[2] https://github.com/magritte-metamodel/magritte
[3] https://github.com/skaplar/NewWave

2. TEORIJSKE OSNOVE

U ovom poglavlju se nalazi opis teorijskih osnova na kojima je zasnovan ovaj rad.

2.1 Poslovni procesi

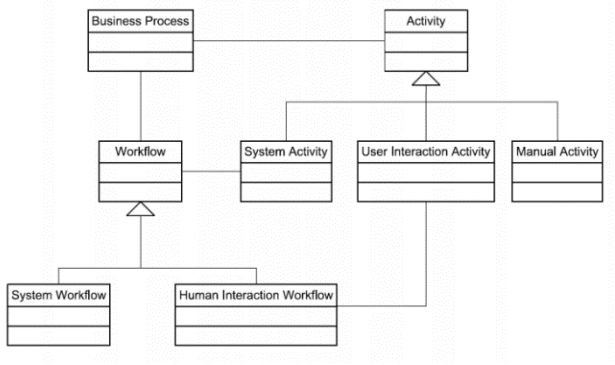
Svaki proizvod ili usluga koju kompanije nude zapravo je proizvod niza aktivnosti – poslovni proces. Kako se teži da se svaki problem prebaci u domen informatike, tako se i za poslovne procese našlo rešenje u oblasti pod nazivom *Business Process Management* (skraćeno BPM). Svaki proces predstavlja niz aktivnosti koje imaju tačno jedan ulaz, ali mogu imati više izlaza. Na izlazu se proizvodi nova vrednost za krajnjeg korisnika. Redosled kojim će se izvršavati aktivnosti unutar procesa jeste bitan.

Životni ciklus poslovnog procesa predstavlja sve faze kroz koje on prolazi od prvog trenutka pa dalje kroz razvoj i primenu. Prva faza u životnom ciklusu jesu dizajn i analiza poslovnog procesa. Druga faza je konfigurisanje i ona se dešava samo u slučaju kada implementacija rešenja zahteva korišćenje softverskih sistema za upravljanje poslovnim procesima. Izvršavanje je treća faza životnog ciklusa koja predstavlja realno izvršavanje poslovnog procesa pokretanjem instanci, u svrhu ostvarivanja poslovnih ciljeva kompanije. Četvrta faza u životnom ciklusu poslovnog procesa je evaluacija, odnosno ocena funkcionalnosti poslovnog procesa.

Iz ugla IT sektora, sistem za upravljanje poslovnim procesima je softverski sistem koji definiše, kreira i upravlja izvršavanjem radnih procesa (*workflow*) korišćenjem odgovarajućeg softvera, koji je u sposoban da interpretira definiciju procesa, obezbedi interakciju sa učesnicima radnog procesa i obezbedi (gde je to potrebno) korišćenje IT alata i aplikacija [1]. U okruženju i procesima u kome su promene česte, fiksan redosled izvršavanja operacija kod aplikacija je neželjena karakteristika. Zato upotreba sistema za upravljanje poslovnim procesima omogućava laku izmenu sistema u skladu sa novim zahtevima i neophodnim promenama. Problem koji ovakvi sistemi donose jeste pitanje raspodele zadataka, jer se kod korisnika stvara osećaj da ga softversko odlučivanje preopterećuje. Takođe, ekspertskim korisnicima se mora pružiti fleksibilnost da se ne bi osećali sputano, ali im se ipak ne sme ostaviti prostor za narušavanje opšteg toka procesa.

2.1.3 Meta-model poslovnih procesa

Meta-model poslovnih procesa je dat na slici 1. Da bi se postigao željeni cilj, u poslovnim procesima se moraju koordinisano izvršavati aktivnosti koje mogu biti sistemske, interaktivne (korisničke) i manuelne.



Slika 1. *Meta-model poslovnog procesa* [1]

Meta-model procesa definiše osnovne elemente modela (elementi modela su instance elemenata meta-modela). Njega čine model procesa, čvorovi i grane.

2.2 Forme za unos podataka

Prilikom definisanja formi za unos podataka, potrebno je voditi računa o pronalaženju minimalnog skupa funkcija potrebnih za izvršenje zadatka. Ovakav način definisanja formi ne zahteva pamćenje veće količine informacija i ubrzava proces učenja.. Komponentu koja će se upotrebiti za unos podataka za određeno polje je potrebno pažljivo odabrati.

Na primer, u slučaju potrebe za unosom većeg teksta treba odabrati drugačiji element u odnosu na potrebu unosa samo nečijeg imena. Elementi koji se u praksi koriste su sledeći: *textField*, *passwordField*, *textArea*, *checkBox*, *radioButton*, *selectionList*, *comboBox* i *label*.

3. *SMALLTALK* – PROGRAMSKI JEZIK

Smalltalk je objektno orijentisani programski jezik pomoću kog se može napisati bilo koja desktop ili *web* aplikacija. On je osmišljen ranih sedamdesetih godina u *Xerox*-ovom centru za istraživanje *Palo Alto Research Center* (PARC).

1980. godine *je Smalltalk*-80 verzije 1 postala prva verzija ovog jezika koja je data u javnu upotrebu, ali samo određenom broju kompanija poput *Apple*, *Hewlett-Packard* i *UC Berkley*.

Tri godine kasnije, *Smalltalk*-80 verzije 2 je dat na korišćenje široj programerskoj javnosti i ovo je verzija na koju se generalno misli pri pomenu *Smalltalk* jezika. *American National Standards Institute* je 1998. godine ratifikovao zvaničnu verziju Smalltalk-a na kojoj se zasnivaju savremene implementacije. Dok u drugoj polovini devedesetih godina kompanije IBM i *ObjectShare* radile na komercijalnom razvoju *Smalltalk*-a, nekoliko *open-source* dijalekata *Smalltalk*-a (između ostalih i *Pharo)*, objavljeni su i dobili su značajan udeo na tržištu. Ipak, tokom 2000-ih, rast *Smalltalk*-a je zaustavljen, ali danas uživa u preporodu zahvaljujući uspehu razvojnih okvira poput *Seaside* i *AIDA/web*-a [2].

3.1 *Pharo* programski jezik

Pharo je objektno orijentisani programski jezik i moćno okruženje koje je fokus stavilo na jednostavnost i neposredni odgovor na svaku liniju koda. On podržava programiranje „uživo“: neposrednu manipulaciju objektima i programskim kodom u okviru pokrenutog programa. *Pharo* ima 2 karakteristična principa: „sve je objekat, pa čak i klasa i poruka“ i „sve se dešava slanjem poruka objektu“ [3]. Poruka predstavlja nameru, ono što treba da se desi. Metode implementiraju ono kako treba da se desi. Novi objekat može biti kreiran slanjem poruke jednog objekta ka drugom.

3.2 Seaside razvojni okvir

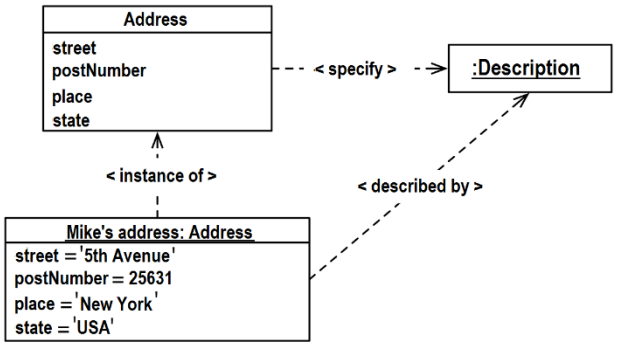
Seaside je razvojni okvir koji omogućava lako razvijanje naprednih i dinamičnih web aplikacija uz korišćenje *Smalltalk*-a. Klase unutar *Seaside*-a se mogu nasleđivati i tako prilagođavati individualnim potrebama. *Seaside* se zasniva na mogućnost stvaranja nezavisnih komponenti koje se iznova mogu koristiti i uklapati za prikaz u *web* pregledaču. Svaka komponenta je odgovorna za svoj prikaz, stanje i tokove kontrole. *Seaside* nudi i mehanzam povratnog poziva (*callback*) koji klikom na odgovarajući deo komponente poziva izvršavanje određenog dela koda. Velika prednost *Seaside*-a je debagovanje aplikacije uz pomoć veoma moćnog dinamičkog alata, kao i mogućnost izmene koda dok je server još aktivan – takozvano „*on the fly“* [4]. Komponenta u *Seaside*-u može da ima interakciju sa drugim komponentama na dva načina. Prvi je da može da ugradi sadržaj i funkcionalnost drugih komponenta u sopstvenu *web* stranicu. Drugi način interakcije je da može da pozove druge komponente i time im omogući da preuzmu *web* stranicu dok ne vrate rezultat glavnoj komponenti.

3.3 *Magritte* razvojni okvir

Magritte je razvojni okvir za opis meta-podataka. Pomoću *Magritte*-a se objektima dodeljuju opisi, a mogu se dobiti *Seaside* komponente koje imaju uključenu validaciju. Štaviše, *Magritte* opisuje sam sebe, omogućavajući automatsko generisanje *meta-editora* koji se može prilagoditi i iskoristiti za izgradnju *web* aplikacije. Ključna ideja koja stoji iza *Magritte*-a je sledeća: potrebno je da postoji jedan objekat sa skupom atributa i opisom tih atributa, na osnovu kojih će se automatski kreirati alati koji obrađuju opise i eventualno kreiraju i *Seaside* komponente. Program će interpretirati opise date u klasi i to na jedan od više načina: kao upit za bazu podataka, kao generički interfejs ili kao *Seaside* komponentu [4]. Primer klase koja predstavlja model i koja sadrži potreban *Magritte* opis dat je na slici 2.

Unutar klase, dodaju se metode čiji nazivi po konvenciji počinju sa „*description*“. Ove metode se koriste za definisanje opisa različitih entiteta ili atributa. U listingu 1 metoda klase *Address*-a kreira objekat *Magritte* opisa koji ima labelu „*Street*“, prioritet i dva pristupa atributu *Street*.

Opisi su centar *Magritte*-a i sadrže mehanizam za pristup vrednostima (*accessor*s) i ograničenja vrednosti (*conditions*). Jednom kada je objekt opisan, *Seaside* komponenta se može dobiti tako što se objektu pošalje poruka „*Object >> asComponent*“.



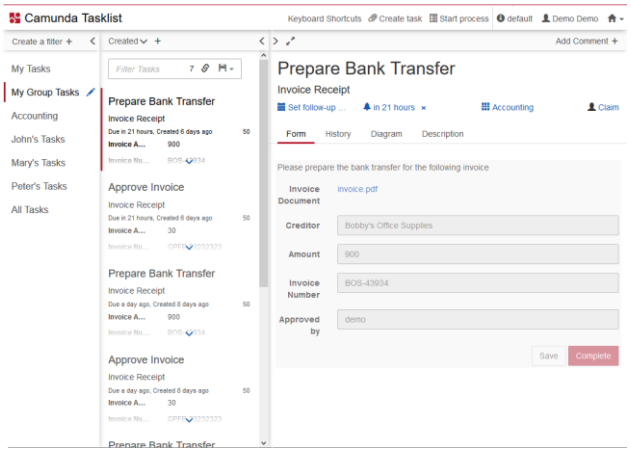
Slika 2. *Objekat koji sadrži opis nad klasom* [4]

```
Address >> descriptionStreet
        ^ MAStringDescription new
          accessor: #street;
          label: 'Street';
          priority: 100;
          yourself
```

Listing 1. Metoda *descriptionStreet* klase *Address*-a

4. POSTOJEĆA REŠENJA

Kako bi se bolje razumeli zahtevi postavljeni zadatkom generisanja formi za poslovne procese, poželjno je imati uvid u neka već postojeća rešenja.



Slika 3. *Camunda Tasklist A* aplikacija

4.1 *Camunda*

Camunda[4] je razvojni okvir zasnovan na *Java*-i koji podržava BPMN notaciju za poslovne procese i automatizaciju procesa. On se koristi kroz *Java* biblioteku koja je odgovorna za izvršavanje BPMN 2.0 procesa. Ima nezahtevno POJO (*Plain Old Java Object*) jezgro, a koristi se i relacionom bazom podataka. ORM (*Object-relational mapping*) je podržan uz pomoć *MyBatis* razvojnog okvira [7]. *Camunda* se sastoji od nekoliko delova za upravljanje poslovnim procesima, među kojima je *Camunda Tasklist A* (*web* aplikacija za upravljanje korisničkim zadacima) najrelevantnija za ovaj rad. Postoje dve različite vrste formi koje krajnji korisnici mogu primeniti za rešavanje korisničkog zadatka ukoliko se forma generiše od strane *Camunda web* aplikacije: generisane forme zadataka i ugrađene forme zadataka (HTML fajl/stranica ugrađena u projekat) [5]. Na slici 3 je prikazana jedna izgenerisana forma u okviru *Camunda Tasklist A* aplikacije.

4.2 *Activiti*

Activiti je razvojni okvir za upravljanje poslovnim procesima pisan u *Java* programskom jeziku i može izvoditi poslovne procese opisane pomoću BPMN 2.0 notacije. Tehnički zahtevi koje ima *Activiti* su *Java runtime* i *Apache Tomcat*[5], iako se zapravo može koristiti bilo koji drugi *web container* jer je oslonac na servletima. *Activiti* pruža pogodan i fleksibilan način za dodavanje formi za unos podataka u poslovnim procesima. Podržava dve strategije za rad sa formama: prikaz ugrađene forme i prikaz eksterne forme.

4.3 *jBPM*

jBPM je alat za izgradnju poslovnih aplikacija koje pomažu u automatizaciji poslovnih procesa i odluka. Poslovna aplikacija može se definisati kao automatizovano rešenje, izgrađeno u odabranim okvirima i sa mogućnostima koji implementiraju poslovne funkcije ili poslovne probleme. Napisan je potpuno u *Java* sistemu i radi na bilo kojoj JVM-i (*Java Virtual Machine*) i dostupan je i u *Maven Central*[6] repozitorijumu.

5. IMPLEMENTACIJA REŠENJA

NewWaveFieldUserTask je proširenje (*plug-in*) za *NewWave Process Engine*[7]. *NewWave* se razvija na Katedri za informatiku Fakulteta tehničkih nauka u Novom Sadu kao softversko rešenje otvorenog koda. Poput *Camunda Tasklist A* ili *Activiti Explorer*-a, ima namenu da omogući krajnjim korisnicima pristup odabiru i rešavanju zadataka, a rešavanje podrazumeva prikaz forme za unos podataka.

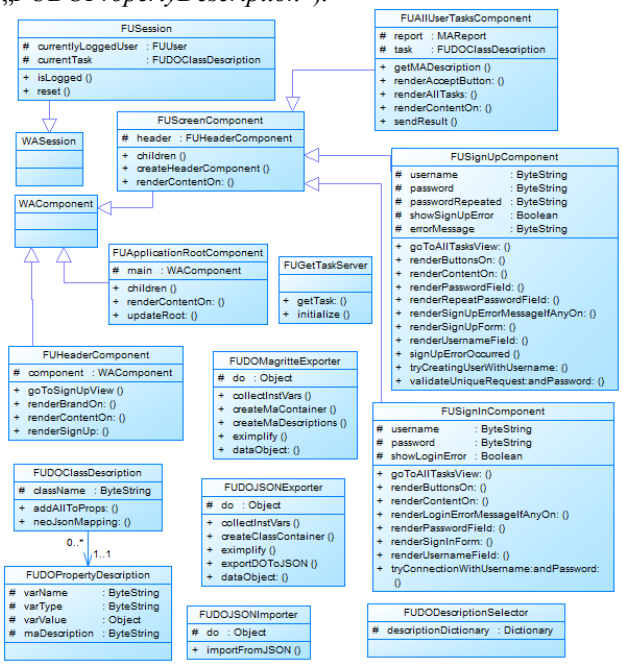
Programski jezik korišćen za implementaciju aplikacije je *Pharo*, a razvojno okruženje *Pharo Virtual Machine*. Rešenje je implementirano korišćenjem *Magritte* i *Seaside* razvojnih okvira. Za stilizovanje izgleda u web prezenteru, korišćen je *Bootstrap wrapper* za *Seaside*. Komunikacija sa *NewWave* okruženjem se odigrava preko REST servisa, a uz pomoć razvojnog okvira *Zinc HTTP Components*[8]. *NeoJSON*[9] razvojni okvir se koristi za mapiranje JSON opisa na odgovarajuće klase.

Na slici 4 prikazan je dijagram klasa. U okviru projekta korišćene su *Seaside* klase „*WASession*“ i „*WAComponent*“. „*WASession*“ nasleđena je od strane „*FUSession*“ klase i pomoću nje se prati sesija. Druga klasa „*WAComponent*“ predstavlja *Seaside* komponentu koja služi za prikaz u *web* pregledaču. Nju nasleđuju klase „*FUApplicationRootComponent*“, „*FUHeader*“ i „*FUScreenComponent*“. „*FUDOClassDescription*“ predstavlja klasu za koju je potrebno izgenerisati formu i ona je zapravo zadatak koji je prosleđen korisniku. Sadrži

[4] https://camunda.com/

[5] http://tomcat.apache.org/
[6] https://search.maven.org/
[7] https://github.com/skaplar/NewWave
[8] https://ci.inria.fr/pharo-contribution/job/EnterprisePharoBook/lastSuccessfulBuild/artifact/book-result/Zinc-HTTP-Client/Zinc-HTTP-Client.html
[9] https://ci.inria.fr/pharo-contribution/job/EnterprisePharoBook/lastSuccessfulBuild/artifact/book-result/NeoJSON/NeoJSON.html

atribute „*className*“ (naziv klase) i „*properties*“ (niz polja odnosno atributa klase koji su tipa „*FUDOPropertyDescription*“).



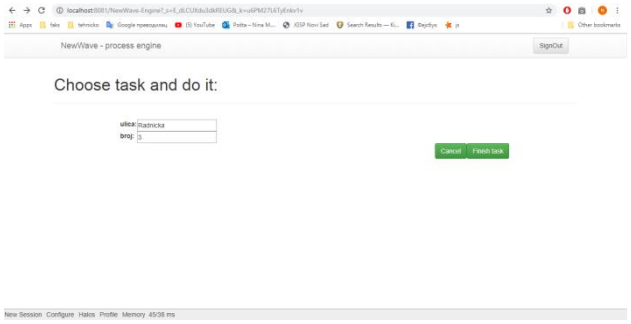
Slika 4. *Dijagram klasa NewWaveFieldUserTask proširenja*

Sadržaj klase „*FUDOClassDescription*“ se dostavlja u vidu JSON opisa kao odgovor na REST zahtev upućen serveru u okviru *NewWave*-a (prikazan na listingu 26). Ovaj opis se prosleđuje klasi „*FUDOJSONImporter*“ slanjem JSON opisa kako parametra u poruci „*importFromJSON*“. Povratna vrednost ove metode je „*FUDOClassDescription*“.

Klasa „*FUDOJSONExporter*“ definiše metode koje se koriste za konstruisanje JSON opisa. Klasa navedene u dijagramu kao „*FUDOMagritteExporter*“ sadrži metode za generisanje *Magritte* opisa („*createMaDescription*“) i kontejnera („*createMaContainer*“). Metoda „*createMa-Container*“ kao povratnu vrednost ima objekat (prikazan na slici 16) koji nasleđuje *Seaside* klasu „*WAComponent*“. Klasa „*FUGetTaskServer*“ sadrži metodu čijim se pozivanjem formira i upućuje REST zahtev ka *NewWave* serveru. Prikaz izgenerisane forme za prikupljanje podataka dat je na slici 5.

6. ZAKLJUČAK

U radu su opisani poslovni procesi, date su smernice i podele koje se tiču formi za unos podataka, opisan je programski jezik *Smalltalk* i njegovi najznačajniji razvojni okviri. Predstavljena su postojeća rešenja, kao i njihov pristup problemu generisanja formi. Prezentovano je rešenje, objašnjene najznačajnije klase i metode i dat je primer korišćenja aplikacije. Kreirano rešenja može biti korisno za NewWave *Process Engine* napisan u *Pharo* programskom jeziku, ali da bi ovo rešenje našlo primenu u praksi, ono mora biti značajno prošireno. Takođe, korišćenje *Magritte* razvojnog okvira, donelo je velike prednosti i olakšanja u generisanju polja u formi.



Slika 5. *Prikaz forme za popunjavanje*

Rad u ovoj oblasti ukazao je na veliki broj mesta i prilika za proširenje. Generisanje formi može biti odrađeno na više načina, ne samo upotrebom *Magritte* opisa. Ipak, *Magritte* opisi pružaju veliku pomoć pri generisanju upita za bazu podataka. Takođe, aplikacija za generisanje formi je napravljena isključivo za korisnike, ali se može proširiti tako da se njom mogu koristiti i operatori procesa ili administratori.

7. LITERATURA

[1] dr Miroslav Zarić, Slajdovi sa predavanja iz predmeta „Upravljanje poslovnim procesima“, Katedra za informatiku, FTN, Novi Sad, 2019. godine

[2] Jon Penland, *Smalltalk: The Original Object-Oriented Programming Language?*, https://www.whoishostingthis.com/resources/smalltalk/, 2019. godine

[3] Pharo MOOC, http://mooc.pharo.org/#week1, 2019. godine

[4] Stéphane Ducasse, Lukas Renggli, C. David Shaffer, Rick Zaccone with Michael Davies, *Dynamic Web Development with Seaside*, 2019. godine

[5] Felix Müller, Mike Winters, *Introducing Embedded Forms Generator: A Camunda Modeler Plugin for Easier Embedded Form Building*, https://blog.camunda.com/post/2018/08/modeler-plugin-embedded-form-generator/, 2018. godine

Kratka biografija:

Nina Miladinović rođena je 1995. godine u Novom Sadu. Završila je Osnovnu školu „Đorđe Natošević“ i gimnaziju „Jovan Jovanović Zmaj“ u Novom Sadu kao nosilac diploma „Vuk Karadžić. 2014. godine, upisuje se na studije na Fakultetu tehničkih nauka, odsek Računarstvo i automatika. Za dalje usmerenje u okviru pomenutog odseka, odlučuje se za smer Primenjene računarske nauke i informatika. Diplomirala je 4. septembra 2018. godine i zatim se upisuje na master studije na istom odseku i smeru, a za podsmer je odabrala Elektronsko poslovanje.