



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У
НОВОМ САДУ



Бојан Власоњић

**ПРОЈЕКТОВАЊЕ СКАЛАБИЛНОГ
ВЕБ-БАЗИРАНОГ КОРИСНИЧКОГ
ИНТЕРФЕЈСА ЗА ПЛАТНЕ
СИСТЕМЕ СА ПОДРШКОМ ЗА
ПОДЕЉЕНО ПЛАЋАЊЕ**

ДИПЛОМСКИ РАД
- Основне академске студије -

Нови Сад, 2020.



УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА


Редни број, РБР:			
Идентификациони број, ИБР:			
Тип документације, ТД:		Монографска публикација	
Тип записа, ТЗ:		Текстуални штампани документ	
Врста рада, ВР:		Дипломски рад	
Аутор, АУ:		Бојан Власоњић	
Ментор, МН:		Др Дину Драган	
Наслов рада, НР:		ПРОЈЕКТОВАЊЕ СКАЛАБИЛНОГ ВЕБ-БАЗИРАНОГ КОРИСНИЧКОГ ИНТЕРФЕЈСА СА ПОДРШКОМ ЗА ПОДЕЉЕНО ПЛАЋАЊЕ	
Језик публикације, ЈП:		Српски/латиница	
Језик извода, ЈИ:		Српски	
Земља публиковања, ЗП:		Република Србија	
Уже географско подручје, УГП:		Војводина	
Година, ГО:		2020.	
Издавач, ИЗ:		Ауторски репринт	
Место и адреса, МА:		Нови Сад, трг Доситеја Обрадовића 6	
Физички опис рада, ФО: (поглавља/страна/ цитата/табела/слика/графика/прилога)		7/31/0/0/18/0/0	
Научна област, НО:		Интеракција човек рачунар	
Научна дисциплина, НД:		Интеракција човек рачунар	
Предметна одредница/Кључне речи, ПО:		Подела плаћања, платни систем, веб апликација, интерфејс, Ангулар, Спринг .	
УДК			
Чува се, ЧУ:		У библиотеци Факултета техничких наука, Нови Сад	
Важна напомена, ВН:			
Извод, ИЗ:		Платни системи са подршком за подељено плаћање су актуелна тема, која има потенцијал да унапреди тржиште . Успешност система диктирају корисничко искуство и употребљив, интуитиван дизајн. Такође, дизајн треба да се прилагоди различитим уређајима и димензијама екрана како би се повећала доступност кориснику.	
Датум прихватања теме, ДП:			
Датум одбране, ДО:			
Чланови комисије, КО:	Председник:	др Душан Гајић, доцент, ФТН Нови Сад	
	Члан:	др Вељко Петровић, доцент, ФТН Нови Сад	Потпис ментора
	Члан, ментор:	др Дину Драган, доцент, ФТН Нови Сад	



KEY WORDS DOCUMENTATION

Accession number, ANO :	
Identification number, INO :	
Document type, DT :	Monographic publication
Type of record, TR :	Textual printed material
Contents code, CC :	Bachelor thesis
Author, AU :	Bojan Vlasonjić
Mentor, MN :	dr Dinu Dragann
Title, TI :	Designing a scalable Web-based user interface with split payment support
Language of text, LT :	Serbian
Language of abstract, LA :	Serbian
Country of publication, CP :	Republic of Serbia
Locality of publication, LP :	Vojvodina
Publication year, PY :	2020.
Publisher, PB :	Author reprint
Publication place, PP :	Novi Sad, Dositeja Obradovića sq. 6
Physical description, PD : (chapters/pages/ref./tables/pictures/graphs/appendixes)	7/31/0/0/18/0/0
Scientific field, SF :	Human-computer interaction
Scientific discipline, SD :	Human-computer interaction
Subject/Key words, S/KW :	Split payment, payment system, web application, interface, Angular, Spring
UC	
Holding data, HD :	The library of the Faculty of Technical Sciences, Novi Sad, Serbia
Note, N :	
Abstract, AB :	Payment systems with split payment support are a current topic, which has the potential to improve the market. The success of the system depends on user experience and a usable, intuitive design. Also, the design should be adapted to different devices and screen dimensions in order to increase user accessibility.
Accepted by the Scientific Board on, ASB :	
Defended on, DE :	
Defended Board, DB :	President: dr Dušan Gajić, docent, FTN Novi Sad
	Member: dr Veljko Petrović, docent, FTN Novi Sad
	Member, Mentor: dr Dinu Dragan, docent, FTN Novi Sad

	Menthor's sign

	УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА 21000 НОВИ САД, Трг Доситеја Обрадовића 6	Датум:
	ЗАДАТАК ЗА ИЗРАДУ ДИПЛОМСКОГ (BACHELOR) РАДА	Лист/Листова:
		7/41

(Податке уноси предметни наставник - ментор)

Врста студија:	<input type="checkbox"/> Основне академске студије <input type="checkbox"/> Основне струковне студије
Студијски програм:	Софтверско инжењерство и информационе технологије
Руководилац студијског програма:	др Мирослав Зарић

Студент:	Бојан Власоњић	Број индекса:	SW 76/2016
Област:	Интеракција човек-рачунар		
Ментор:	др Дину Драган		

НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА

ИЗДАЈЕ СЕ ЗАДАТАК ЗА ДИПЛОМСКИ (Bachelor) РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:

- проблем – тема рада;
- начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна;
- литература

НАСЛОВ ДИПЛОМСКОГ (BACHELOR) РАДА:

ПРОЈЕКТОВАЊЕ СКАЛАБИЛНОГ ВЕБ-БАЗИРАНОГ КОРИСНИЧКОГ ИНТЕРФЕЈСА ЗА ПЛАТНЕ СИСТЕМЕ СА ПОДРШКОМ ЗА ПОДЕЉЕНО ПЛАЋАЊЕ

ТЕКСТ ЗАДАТКА:

Проучити платне системе са посебним освртом на платне системе са подршком за подељено плаћање и скалабилну, Респонсив (Responsive), методологију пројектовања веб-базираног корисничког интерфејса. Одредити факторе који утичу на корисничко искуство у платним системима са подршком за подељено плаћање. Пројектовати и имплементирати прототип једног скалабилног корисничког интерфејса за платне системе са подршком за подељено плаћање. Прилагодити приказ конвенционалним десктоп рачунарима и паметним телефонима. Демонстрирати рад једног таквог интерфејса. Извући одговарајуће закључке.

Руководилац студијског програма:	Ментор рада:
др Зарић Мирослав	Др Дину Драган

Примерак за: ☐ - Студента; ☐ - Ментора

Sadržaj

1. UVOD	1
2. PODELJENO PLAĆANJE	3
2.1. <i>PayPal</i>	3
2.2. <i>Braintree</i>	4
3. SPECIFIKACIJA SISTEMA	5
3.1. Struktura sistema	5
3.2. Opis sistema	5
4. KORIŠĆENE TEHNOLOGIJE	7
4.1. <i>Spring boot</i>	7
4.2. <i>Angular</i>	8
4.3. <i>Biblioteke</i>	9
4.3.1. <i>Chart.js</i>	9
4.3.2. <i>Angular material</i>	10
4.3.3. <i>Bootstrap 4</i>	11
5. IMPLEMENTACIJA	13
5.1. <i>Backend</i>	13
5.1.1. Struktura projekta	13
5.1.2. Baza podataka	15
5.1.3. Dijagram klasa	16
5.2. <i>Frontend</i>	17
5.2.1. Struktura projekta	17
5.2.2. Način implementacije	19
5.2.3. Početna verzija	20
5.2.4. Konačna verzija	21
6. POKRETANJE APLIKACIJE	25
6.1. Pokretanje <i>spring boot</i> servera	25
6.2. Pokretanje <i>Angular</i> servera	26
7. ZAKLJUČAK	27
LITERATURA	29
BIOGRAFIJA	31

1. UVOD

Postoji više interpretacija i prevoda za izraz *split payment*. U načelu, izraz se prevodi kao „podela plaćanja“ ili „podeljeno plaćanje“. Međutim, ovaj pojam nije jednoznačno određen.

U kontekstu rada, podele plaćanja omogućuju korisniku da cenu određenog artikla podeli na više delova, gde se svaki od tih delova prenose na jedinstvene račune u banci. Na primer, za artikal po ceni od 100 dinara, korisnik određuje da 20 dinara prenosi za PDV, 2 dinara prenosi u humanitarne svrhe, 10 dinara ulaže u dažbine, 30 dinara je profit, itd. Prenos novca na račune bi se izvršavao automatski pri svakoj kupovini artikla. Ovakav sistem bi pomogao korisniku da, recimo, izmiri odgovarajuće dažbine krajem meseca i utvrdi količinu artikala koju treba da proda kako bi ostvario stabilne prihode.

Konstruisanje platnog sistema za podelu plaćanja je aktuelna tema. Javila se kao zahtev klijenta koji je želeo da napravi aplikaciju za online kupovinu, ali da se plaćanje ne vrši direktno, nego da bude raspoređeno na veći broj primalaca [1]. Za implementaciju korisničkog interfejsa u sklopu ovog sistema ne postoji jedinstveno rešenje, niti su rešenja široko rasprostranjena niti javno dostupna.

Cilj rada je da se napravi interfejs koji bi na najlakši i najintuitivniji način omogućio korisniku da izvrši ovakve podele plaćanja. Pri tome, rešenje treba da bude upotrebljivo na različitim uređajima i različitim dimenzijama ekrana kako bi bilo dostupno što većem broju korisnika. Prilikom implementacije celokupnog platnog sistema za podeljena plaćanja, upotrebljivost rešenja i korisničko iskustvo predstavljaju ključne faktore za uspešnost projekta.

Rad se sastoji iz sedam poglavlja, gde prvo poglavlje predstavlja uvod u rad, a poslednje poglavlje zaključak. U sklopu drugog poglavlja je opisan način funkcionisanja podele plaćanja u sklopu poznatih platnih sistema. Treće poglavlje sadrži opis i specifikaciju aplikacije koja je implementirana za svrhe rada. U sklopu četvrtog poglavlja su opisane tehnologije u kojima je razvijena aplikacija, dok peto poglavlje sadrži detalje o implementaciji aplikacije. Šesto poglavlje specificira način upotrebe rešenja, tj. na koji način se aplikacija pokreće i koje je programe neophodno instalirati radi ostvarivanja uspešnog pokretanja.

2. PODELJENO PLAĆANJE

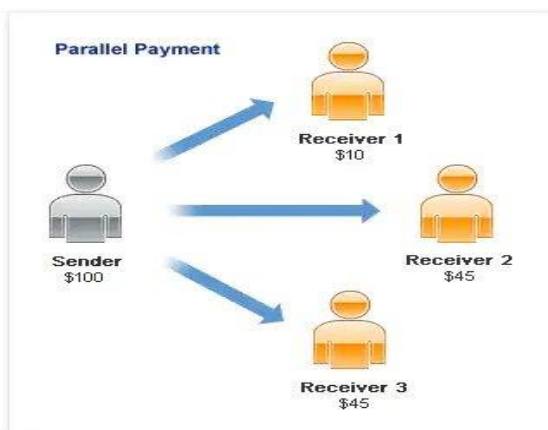
Podeljena plaćanja se mogu interpretirati na više različitih načina. U nastavku sledi objašnjenje o načinu funkcionisanja podeljenih plaćanja u poznatim platnim sistemima. Pri tome, podeljena plaćanja koje nude ti sistemi odgovaraju podeljenim plaćanjima u sklopu teme ovog rada.

2.1. PayPal

PayPal je internet orijentisana kompanija koja omogućava vršenje uplata i novčanih prenosa preko interneta. Omogućava korisnicima da otvore račun na svojoj platformi i da ga povežu sa svojom kreditnom karticom ili tekućim računom. Nakon potvrde identifikacije i validacije računa, korisnici mogu započeti slanje ili primanje uplata sa drugih *PayPal* računa [2].

PayPal nudi *API*¹ za “Adaptivna plaćanja” (*Adaptive payments API*), koja korisnicima omogućuju da naprave automatizovana plaćanja. U načelu, nudi se podrška za obradu plaćanja između pošiljaoca uplate i jednog ili više primalaca uplate [3].

Adaptivna plaćanja podržavaju nekoliko vrsta plaćanja, među kojima se vrsta plaćanja koja najviše priliči podeli plaćanja u kontekstu rada naziva paralelno plaćanje (*parallel payment*), čiji je primer prikazan na slici 2.1.



Slika 2.1. – primer paralelnog plaćanja u *PayPal*-u [3].

¹ *API* – softverski posrednik koji omogućuje komunikaciju između dve različite aplikacije [4].

Paralelna plaćanja omogućuju pošiljaocu da pošalje jednu uplatu na više primalaca. Tehnički, predstavlja skup višestrukih plaćanja izvršenih u jednom zahtevu za plaćanje. Broj primalaca može da varira između 2-6. Jedan od primera bi bilo jednokratno plaćanje za više proizvoda od različitih trgovaca. U tom slučaju, *PayPal* prilikom plaćanja oduzima novac sa računa pošiljaoca i šalje ga na račune trgovaca [3].

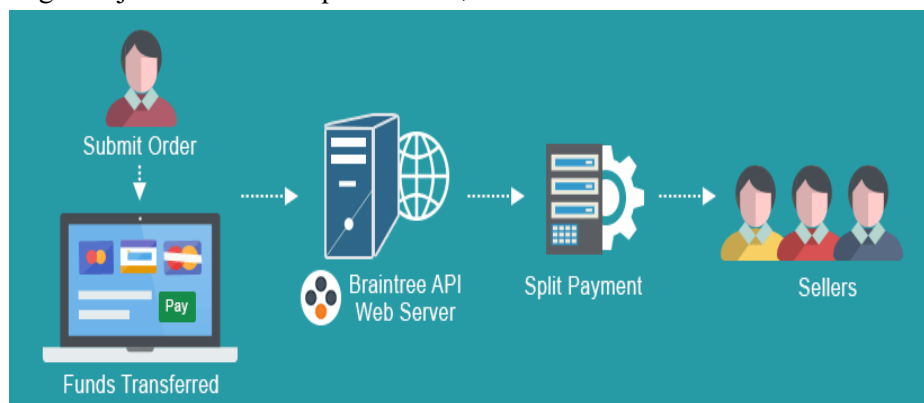
2.2. Braintree

Braintree kompanija se specijalizovala za sisteme plaćanja u sklopu kompanija koje podržavaju internet trgovinu (*e-commerce*). *Braintree* pripada *Paypal*-u i nudi slično rešenje poput adaptivnih plaćanja u *PayPal*-u uz fleksibilnost plaćanja putem različitih metoda poput kreditnih kartica, bankarskih prenosa i slično [1].

Rešenje koje *Braintree* nudi u sklopu problema podele plaćanja za određenu transakciju između više različitih primalaca je *Marketplace API*. U sklopu strukture ovog *API*-ja, se izdvajaju sledeći učesnici [1]:

- *Master merchant* (glavni trgovac) – predstavlja vlasnika tržišta.
- *Sub-merchants* (prodavci) – dodaje ih trgovac i odobrava im prodaju proizvoda na njegovom tržištu. Za uzvrat, prodavci u obavezi da mu isplate odgovarajuću naknadu.

Prilikom kupovine proizvoda, sprovodi se automatizovan prenos odgovarajuće svote novca prodavcima, demonstriran na slici 2.2.



Slika 2.2. – primer sprovedenih koraka i odgovarajućih učesnika prilikom kupovine proizvoda koristeći *Braintree Marketplace API* [6].

3. SPECIFIKACIJA SISTEMA

Definisanje potpuno funkcionalnog sistema za interfejs sa podrškom za podeljeno plaćanje, je veliki izazov. Definisanje bezbednosnih mehanizama i njihovo uvezivanje u transakcije kako bi se omogućio bezbedan prenos finansijskih sredstava za svaki od korisničkih proizvoda su prilično komplikovani zahtevi koje bi ovakav sistem trebao da podrži..

Međutim, u kontekstu rada, akcenat je na kreiranju jednostavnog i efikasnog interfejsa. Prema tome, sistem i entiteti sistema će biti pojednostavljen u skladu sa tim - ne postoje administratori sistema, trgovci se ne registruju već postoji jedan predefinisani trgovac, nisu pokriveni bezbednosni aspekti sistema, bankarski računi su izmišljeni i ne vrše se nikakve transakcije, podrazumevana valuta su dinari i slično.

3.1. Struktura sistema

Korisnici u sistemu:

- Trgovac.

Entiteti sa kojima korisnik ima dodira:

- Artikli (*Articles*).
- Bankarski računi (*Accounts*).
- Prenosi finansijskih sredstava na odgovarajuće račune u banci – podeljena plaćanja (*Payment splits*).

Funkcionalnosti:

- Kreiranje, pregled, modifikacija i brisanje artikala.
- Kreiranje, modifikacija, pretraga i odabir bankovnih računa prilikom konfigurisanja podela plaćanja.
- Kreiranje, pregled, modifikacija i brisanje podela plaćanja.

3.2. Opis sistema

Aplikacija je namenjena trgovcima, oni su glavni korisnici u sistemu. Svaki trgovac definiše svoje artikle unosom odgovarajućih podataka. Zarad jednostavnosti, podaci koji se pamte za svaki artikal su naziv i cena. Trgovcima treba omogućiti brz i efikasan pregled unesenih proizvoda, kao i mogućnost izmene podataka o proizvodu i brisanje konkretnog proizvoda. Radi dodatnog pojednostavljenja, nije moguće smanjiti cenu artikla ukoliko su kreirana podeljena plaćanja u sklopu njega, već je moguće samo uvećati cenu.

Za svaki artikal, trgovci definišu podele plaćanja. Ukoliko nijedan podela nije definisana, podrazumevano se puna cena artikla prenosi na račun prodavca. Kreiranjem podela plaćanja prodavac umanjuje svoju zaradu, ali izmiruje dažbine prenosom novca na odgovarajuće račune. Ostatak novca pri podeli sredstava predstavlja profit trgovca. Prilikom kreiranja podela plaćanja, prva stvar koju korisnik određuje je račun u banci na koji se novac prenosi.

Sistem bi trebao da ima memorisanu kolekciju o opšte poznatim i najčešće korišćenim bankarskim računima (npr. račun JKP Informatike). Za potrebe zadatka, ključni podaci o računu su broj računa, naziv primaoca i adresa primaoca. Pored predefinisanih računa, korisnik može i ručno da ukuca podatke odgovarajućeg računa, i sistem je u obavezi da pamti unesene račune i omogući korisniku efikasnu pretragu i odabir korišćenih računa.

Nakon što korisnik unese ili odabere odgovarajući račun, treba da odredi količinu finansijskih sredstava koju prenosi na račun. Količinu koju prenosi može definisati fiksno ili procentualno. Na taj način kreira delu plaćanja. Kreirane podele korisnik može modifikovati ili obrisati.

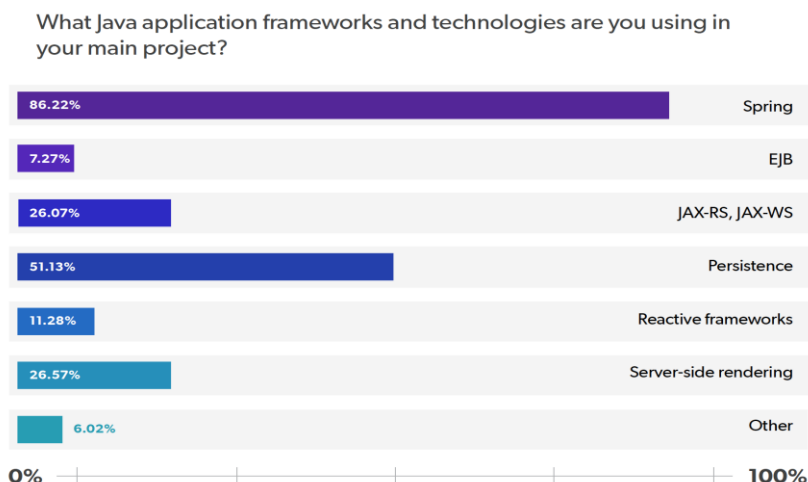
4. KORIŠĆENE TEHNOLOGIJE

Uzevši u obzir da bi aplikacija trebala biti široko rasprostranjena i dostupna trgovcima u svakom trenutku, razvijena je veb aplikacija. Veb aplikacija je aplikacija kojoj korisnik pristupa putem mreže (interneta), preko veb pretraživača (*web browser*) [7].

*Backend*² deo aplikacije je implementiran korišćenjem *spring boot* radnog okvira (*framework-a*³). *Frontend*⁴ deo aplikacije je implementiran korišćenjem *Angular framework-a*.

4.1. Spring boot

Spring je jedan od najčešće korišćenih *Java EE*⁵ *framework-a* za izgradnju aplikacija. Cilj mu je da pojednostavi razvoj veb aplikacija i pomogne programerima da budu produktivniji. Može se primeniti na bilo kojoj platformi [8]. Na slici 4.1 je prikazana popularnost java framework tehnologija u 2020. godini, i vidi se da je *spring* najpopularniji.



Slika 4.1 – najpopularnije java *framework* tehnologije za 2020. godinu [9].

² *Backend* - razvijanje logike i načina funkcionisanja veb aplikacije [10].

³ *Framework* – softver koji obezbeđuje osnovne funkcionalnosti na osnovu kojih programeri prave programe za određenu platform [11]. Platforma predstavlja operativni sistem [12].

⁴ *Frontend* – razvijanje izgleda veb aplikacije [10].

⁵ *Java EE* – java okruženje za razvoj i pokretanje mrežnih aplikacija [13].

U načelu, *spring boot* predstavlja ekstenziju (unapređenje) *Spring framework*-a time što je eliminisao gomilu koda koja je do tad bila neophodna za konfiguraciju i podešavanje *Spring* aplikacije [14].

Spring boot predstavlja *framework* otvorenog koda (*open-source* ⁶⁾ zasnovan na programskom jeziku Java koji se koristi za kreiranje mikro servisa⁷ [8]. Obezbeđuje programerima dobru platformu za razvijanje nezavisnih (*stand-alone application*) [15] veb aplikacija koje su namenjene za često korišćenje i široko upotrebu u sklopu intenzivnih okruženja (*production-grade application*) [16].

Neke od prednosti *spring boot*-a [8]:

- Olakšava razumevanje i razvoj *spring* aplikacija.
- Povećava produktivnost.
- Smanjuje vreme potrebno za razvoj aplikacije.

4.2. Angular

Angular je *framework* za razvoj klijentskih aplikacija na jednoj stranici (*single-page client applications* ⁸⁾ korišćenjem *HTML* i *Typescript* jezika [17]. *Angular* biblioteka nudi funkcije koje olakšavaju implementaciju složenih zahteva savremenih aplikacija. Pored toga, sadrži i detaljnu dokumentaciju i stilske smernice koje određuju kako aplikacija treba da se strukturiira i napravi korišćenjem *Angular*-a [18].

Neke od prednosti *Angular framework*-a su:

- Ponovna iskoristivost – aplikacija se gradi pomoću komponenti. Komponente se mogu koristiti u sklopu cele aplikacije i mogu sadržati jedna drugu.
- Olakšano testiranje – komponente su međusobno nezavisne tako da je olakšano testiranje pojedinačnih funkcionalnosti u sklopu komponente.
- Održivost – komponente su međusobno nezavisne tako da se mogu lako zameniti boljom implementacijom. Time je olakšano održavanje i ažuriranje koda.

⁶ *Open source* – softver sa javno dostupnim izvornim kodom, tako da svako može da ga pregleda, modifikuje ili unapredi [19].

⁷ Mikroservisi – tehnika razvoja softvera koja strukturiira aplikaciju kao kolekciju labavo povezanih servisa. Aplikacija se razgrađuje na posebne, manje servise [20].

⁸ *Single page application (SPA)* – aplikacija koja se izvršava u pretraživaču i koja ne zahteva ponovno učitavanje stranice za vreme interakcije korisnika sa aplikacijom, što značajno ubrzava aplikaciju i unapređuje korisničko iskustvo [21].

4.3. Biblioteke

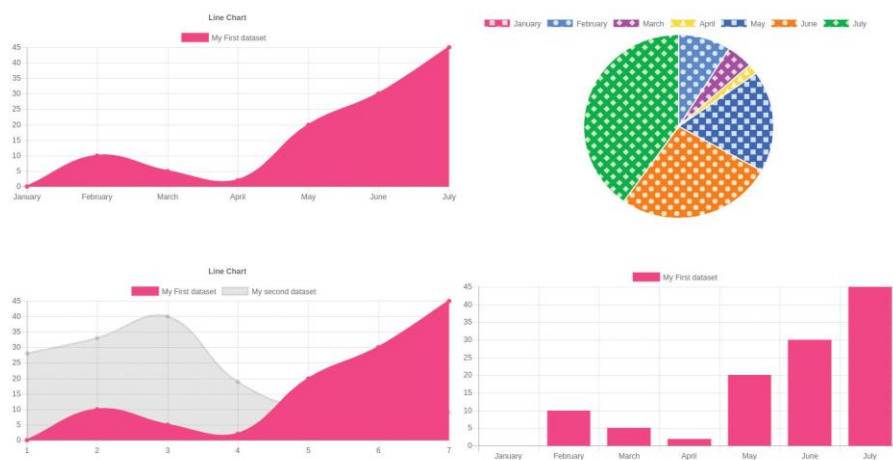
Pored standardnih biblioteka neophodnih za rad prethodno pomenutih tehnologija, dodatne biblioteke koje sam koristio za razvoj klijentskog dela aplikacije su:

- *Chart.js*
- *Angular material*
- *Bootstrap 4*

4.3.1. Chart.js

Chart.js je *open-source javascript* biblioteka sa *MIT*⁹ licencom. Odlikuju je dobre performance i prikaz u svim modernim pretraživačima. Sadrži detaljnu dokumentaciju i podršku za *Angular*, uz dodatnu upotrebu *ng2-charts* biblioteke. Omogućuje prikaz 8 različitih vrsta dijagrama (od kojih su pojedini prikazani na slici 4.2.), poput sledećih [22]:

- linijski dijagrami
- radarski dijagrami
- stubičasti dijagrami
- kružni i prstenasti dijagrami
- dijagrami sa mehurićima i sl.



Slika 4.2. – primeri interaktivnih dijagrama u *Chart.js* biblioteci [23].

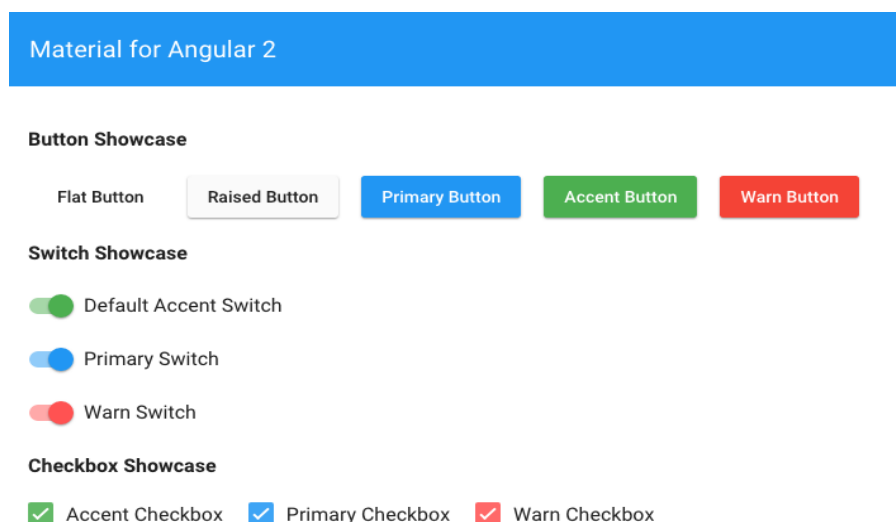
⁹ MIT licenca - licenca za slobodan softver, potiče od Masačusetskog tehnološkog instituta (MIT). Dozvoljava ponovnu upotrebu vlasničkog softvera [24].

4.3.2. Angular material

Angular material je biblioteka koja sadrži različite komponente korisničkog interfejsa namenjenih za programere koji koriste *Angular*. Ove komponente omogućavaju izradu atraktivnih, doslednih i funkcionalnih veb aplikacija, pridržavajući se savremenih principa veb dizajna poput kao što su nezavisnost uređaja i podrška za različite veb pretraživače [25].

Neke od odlika biblioteke su:

- Omogućuje dizajniranje upotrebom ugrađenih, odzivnih (*responsive*) komponenti.
- Sadrži nove verzije uobičajenih kontrola korisničkog interfejsa poput dugmića, tekstualnih polja i sl. Primeri komponenti su prikazani na slici 4.3.
- Uključuje poboljšane i specijalizovane funkcije poput kartica, trake za navigaciju (*navbar*), prevlačenja preko ekrana i sl.
- Sadrži tzv. prilagodljivo dizajniranje (*responsive design*), tako da će se veb stranica kreirana korišćenjem biblioteke adekvatno skalirati prema veličini uređaja (ekrana).
- Podržava senke i podebljane boje koje ostaju jednolike na različitim uređajima i platformama.
- Najvažnije, biblioteka je besplatna za upotrebu.



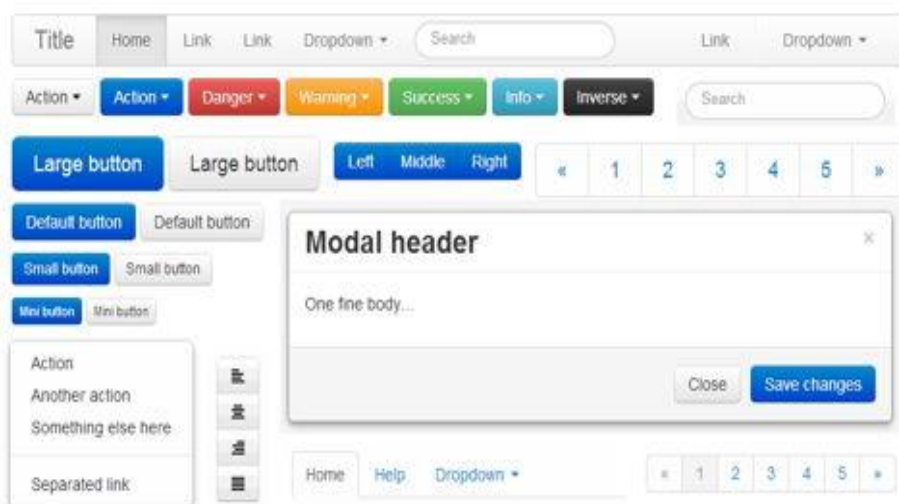
Slika 4.3. – primeri komponenti u sklopu *Angular material*-a [26]

4.3.3. Bootstrap 4

Bootstrap je ogromna kolekcija praktičnih, ponovno iskoristivih delova koda napisanih u jezicima *HTML*, *CSS* i *Javascript*. Takođe je i *frontend framework* koji omogućava programerima i dizajnerima da brzo naprave potpuno odzivne (*responsive*) veb aplikacije [27]. Otvorenog je koda i besplatan je za korišćenje.

Neke od karakteristika i prednosti su:

- *Responsive grid* (odzivna mreža) – ugrađen mrežni sistem od redova i kolona. Omogućuje korisniku da pravi kontejnere ispunjene sadržajem veb stranice. Za redove i kolone je moguće definisati položaj, veličinu, procep, centriranost i omogućuje skaliranje sadržaja na različite dimenzije ekrana. Primer mreže je prikazan na slici 4.5.
- *Responsive images* (odzivne slike) – automatsko menjanje veličine slika na osnovu dimenzija ekrana.
- *Bootstrap components* – različite komponente korisničkog interfejsa poput navigacionih traka, padajućih menija, sličica i ikonica, dugmića i traka za napredak (*progress bar*). Primeri komponenti su prikazani na slici 4.4.
- Dokumentacija – sadrži listu komponenti, uz objašnjenje, uputstvo i primere za korišćenje.
- MIT licenca.



Slika 4.4. – primeri komponenti u sklopu *Bootstrap*-a [28].

EXAMPLE

.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1
.col-md-4				.col-md-4				.col-md-4			
.col-md-4				.col-md-8							
.col-md-6						.col-md-6					
.col-md-12											

Slika 4.5. – primer *bootstrap*-ovog mrežnog sistema [29].

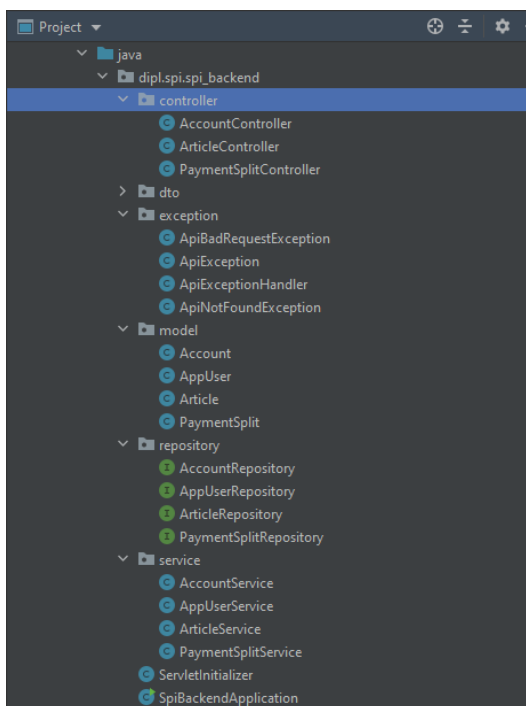
5. IMPLEMENTACIJA

Jedan od glavnih izazova prilikom implementacije je bila funkcionalnost za kreiranje podela plaćanja. Korisnik bi trebao da bacanjem pogleda na interfejs jednostavno zaključi koliko podeljenih plaćanja je kreirao, koja je svota novca u sklopu svake podele, kao i koliki je profit korisnika. U nastavku će biti opisana struktura veb aplikacije i detalji o implementaciji.

5.1. Backend

Backend deo veb aplikacije je implementiran korišćenjem *spring boot framework*-a i *MySQL* baze podataka. Implementacija je krajnje pojednostavljena, ne pokriva bezbednosne aspekte za autentifikaciju i rukuje minimalnim skupom entiteta neophodnim da korisnik isproba najosnovnije funkcionalnosti na klijentskom delu aplikacije.

5.1.1. Struktura projekta



Slika 5.1.1. – struktura *spring boot* projekta.

U sklopu *spring boot* projekta se mogu izdvojiti sledeće celine:

- *Controller* (kontroleri) – implementirani pomoću *RESTful API* ¹⁰. Sadrže odgovarajuće putanje i metode za dobavljanje, kreiranje, modifikaciju i brisanje entiteta u sistemu. Svaki entitet se posmatra kao resurs, i svaka od metoda omogućava klijentu manipulaciju nad tim resursom.
- *Dto (transfer objekti)* - klijent ne dobavlja entitete koji se memorišu, već se ti entitetu upakuju u tzv. transfer objekte i njima klijent rukuje.
- *Exception* (izuzeci) – neminovno je da će dolaziti do grešaka prilikom korišćenja aplikacije. Predviđene greške se obrađuju unutar servisa i klijentu se vraća odgovarajuća poruka.
- *Model* (model) – sadrži osnovne entitete koji se memorišu u bazi podataka. Klijent preko transfer objekata, putem kontrolera i servisa sprovodi odgovarajuće operacije nad entitetima.
- *Repository* (repozitorijum) – zadužen za komunikaciju sa bazom podataka i memorisanje entiteta. Kreiran pomoću *spring-ove JPA* ekstenzije čiji cilj je da smanji količinu koda potrebnog za komunikaciju sa bazom podataka [30].
- *Service* (servisi) – nakon što korisnik pozove odgovarajuću metodu nad resursom u sklopu kontrolera, logika se izvršava unutar servisa. Izvršavaju se odgovarajuće provere, u skladu sa tim se ili baca izuzetak ili se entitet obrađuje i memoriše i vraća se odgovor klijentu.

¹⁰ *RESTful API* - mrežni servis u sklopu kog resursi imaju sopstveni URL (putanju) koja ih identifikuje. Pristup do resursa je definisan *HTTP* protokolom (protokol za prenos informacija na vebu [31]). Po pravilu, isti URL se koristi za sve informacije, ali se menja *HTTP* metoda koja definiše vrstu operacije – dobavljanje (*get*), kreiranje (*post*), modifikacija (*put*) i brisanje (*delete*) [32].

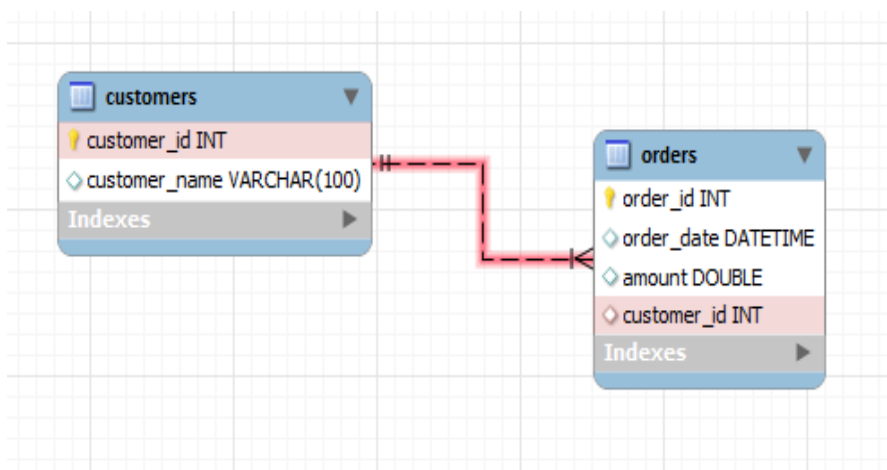
5.1.2. Baza podataka

Entiteti aplikacije se memorišu pomoću *MySQL* baze podataka. Baza podataka je posebna kolekcija koja čuva zbirku podataka. Svaka baza podataka ima jedan ili više različitih *API*-ja za kreiranje, pristup, upravljanje, pretragu i replikaciju podataka unutar nje [33].

MySQL je relacionala baza podataka. U prevodu, svi podaci se čuvaju u tabelama i odnosi između podataka se uspostavljaju pomoću njihovih ključeva. Ključevi predstavljaju jedinstven identifikator za svaku instancu entiteta u bazi podataka. Primer relacije između dve tabele je prikazan na slici 5.1.2.

Neki od faktora koji utiču na popularnost *MySQL*-a su [33]:

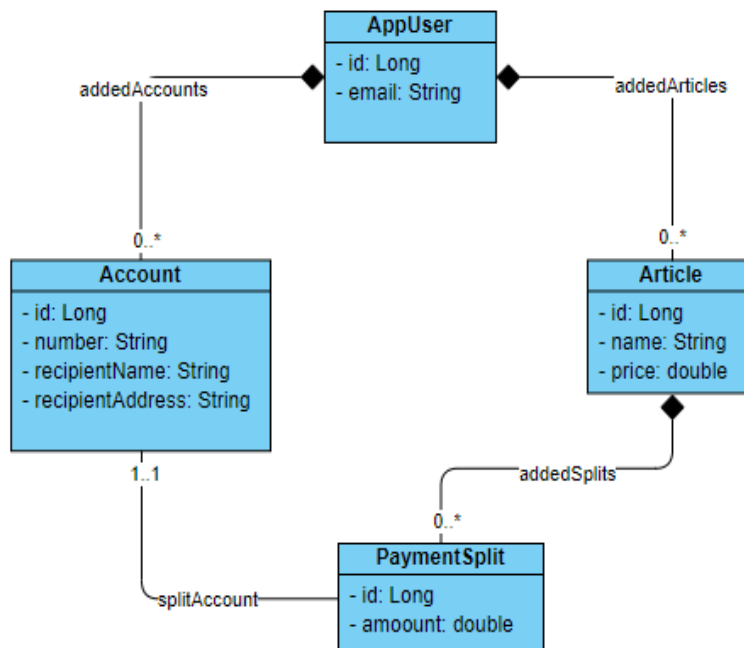
- Objavljen je pod licencom otvorenog koda (*open source*). Dakle, ne plaća se za upotrebu.
- Koristi standardan oblik poznatog *sql* jezika.
- Ima podršku za različite programske jezike poput *php*, *c*, *c++*, *java* i funkcioniše u okviru različitih platformi.
- Radi brzo i dobro funkcioniše sa velikim skupom podataka.
- *MySQL* je prilagodljiv. Licenca otvorenog koda omogućuje programerima da modifikuju softver kako bi ga prilagodili svojim potrebama i uklopili u sklop svojih okruženja.



Slika 5.1.2. – primer relacije između dve tabele u *MySQL* bazi podataka [34].

5.1.3. Dijagram klasa

Na slici 5.1.3 je prikazan dijagram klasa aplikacije. Dijagram klasa ujedno oslikava entitete i atribute sistema, kao i tabele i podatke koji se skladište u *MySql* bazi podataka.



Slika 5.1.3. – dijagram klasa u sklopu projekta.

Svaki entitet je predstavljen plavim pravougaonikom, i čuva se u bazi podataka kao posebna tabela.

Kao što je već pomenuto, entiteti sadrže minimalan skup atributa koji ih opisuju, i prikazani su ispod naziva svakog od entiteta, u sklopu plavih pravougaonika.

Entiteti su međusobno povezani relacijama, koje su predstavljene odgovarajućim strelicama. Za svakog korisnika se čuva kolekcija dodatih artikala i unesenih bankovnih računa, koje korisnik pretražuje prilikom kreiranja podela plaćanja. Za svaki artikal se čuva kolekcija podeljenih plaćanja koje je korisnik definisao. Podeljeno plaćanje ima referencu na bankovni račun na koji će se preneti finansijska sredstva.

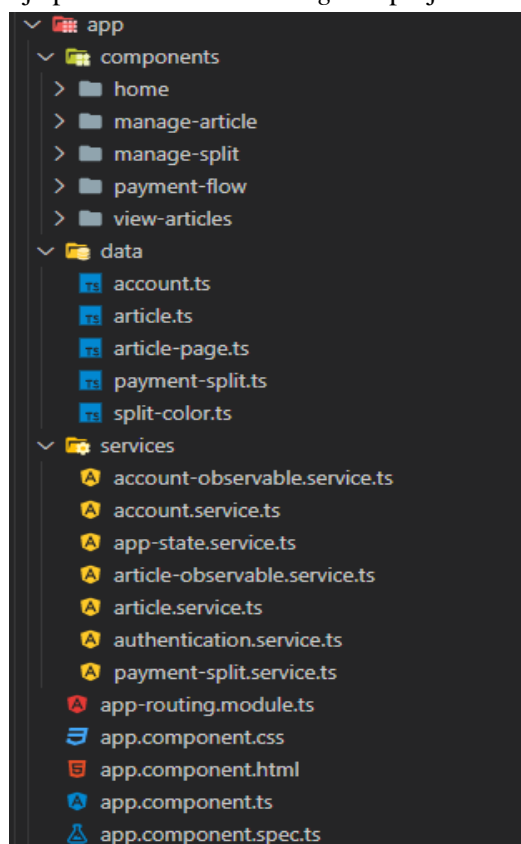
5.2. Frontend

Najveći obim posla prilikom implementacije je bio posvećen izradi klijentskog dela aplikacije. Implementacija je izvršena koristeći *Angular framework*. Cilj je bio napraviti upotrebljivo i intuitivno rešenje za različite dimenzije ekrana kako bi aplikacija bila rasprostranjena i dostupna korisniku sa bilo kog uređaja koji ima pristup internetu.

Početna ideja je bila da se napravi isto rešenje upotrebljivo za obe vrste ekrana. Međutim, u tom slučaju je aplikacija gubila na upotrebljivosti, tako da je rešenje najpreglednije i najpraktičnije za desktop uređaje. Aplikacija se uspešno skalira na manje ekrane, ali se gubi na preglednosti.

5.2.1. Struktura projekta

Na slici 5.2.1 je prikazana struktura *Angular* projekta.



Slika 5.2.1. – struktura *frontend* projekta.

U sklopu *Angular* projekta se izdvajaju sledeće celine:

- *Components* (komponente) – sadrže delove veb stranice koji se mogu višestruko upotrebiti unutar aplikacije. Osnovna svrha im je da omoguće korisniku prikaz, obradu i manipulaciju nad određenim entitetom.
- *Data* (podaci) – podaci su mahom ekvivalentni transfer objektima koji se dobavljaju sa *backend* servera. Ti podaci se mapiraju, koriste unutar aplikacije i šalju na server. Potencijalno sadrže i određene podatke koji se koriste isključivo u sklopu *frontend*-a, poput klase *split-color*.
- *Services* (servisi) – služe za komunikaciju sa *backend* serverom. Pozivaju odgovarajuće metode iz kontrolera, na osnovu *URL*-a i na taj način iniciraju dobavljanje, kreiranje, modifikaciju i brisanje objekata. Pored toga, servisi označeni rečju *observable* služe za komunikaciju između *Angular* komponenti. *App-state* servis služi za utvrđivanje dimenzija ekrana i pomaže u skaliranju aplikacije za različite uređaje.

U nastavku sledi kratak opis komponenti:

- *Home* – početna strana. Jedina komponenta koja ne vrši nikakvu logiku niti manipuliše nekim od entiteta. Početna strana je prva stvar koju korisnik vidi kad pristupi aplikaciji.
- *Manage-article* – komponenta koja služi za kreiranje i modifikaciju artikala. Sadrži formu sa poljima koja predstavljaju attribute objekta *article*.
- *Manage-split* – komponenta koja služi za kreiranje i modifikaciju podela plaćanja. Sadrži formu sa poljima koja predstavljaju attribute objekta *payment-split*. Ovoj komponenti se prosleđuje i objekat *split-color* kako bi obojila dugme za podnošenje forme u odgovarajuću boju, čime bi korisnik stekao bolji utisak nad kojomodelom plaćanja manipuliše.
- *Payment-flow* – komponenta u sklopu koje se prikazuju *manage-split* komponenta i pita dijagram koji služi za prikaz dodatnih podela plaćanja u sklopu artikla. Objedinjuje logiku za prikaz i logiku za dodavanje i izmenu podela plaćanja.
- *View-articles* – komponenta koja služi za izlistavanje artikala korisnika. Funkcionalnosti koje uključuje su pretraga artikala po nazivu, prikaz artikala po stranicama u zavisnosti od broja artikala (paginacija) i brisanje artikla.

5.2.2. Način implementacije

U načelu, tokom razvoja se mogu izdvojiti dve verzije aplikacije. Način implementacije zahteva i korišćene komponente su isti za obe verzije, jedina razlika je u prostornom rasporedu komponenti.

Implementirane su funkcionalnosti navedene u sklopu poglavlja „Opis sistema“. Najproblematičniji zahtevi su bili prikaz i manipulacija nad podelama plaćanja kao i skaliranje aplikacije na ekrane različitih dimenzija.

Skaliranje aplikacije na različite vrste uređaja je postignuto upotrebom *bootstrap*-ovog mrežnog sistema. Dodavanjem odgovarajućih klasnih oznaka na redove i kolone, dolazi do izmeštanja položaja komponenti na ekranu. Prostorni raspored komponenti se razlikuje u zavisnosti od dimenzija ekrana. Konkretno, postoje tri upečatljiva rasporeda. To su rasporedi za desktop ekrane, za ekrane srednje veličine (npr. kod tableta) i za male ekrane (ekrani mobilnih uređaja).

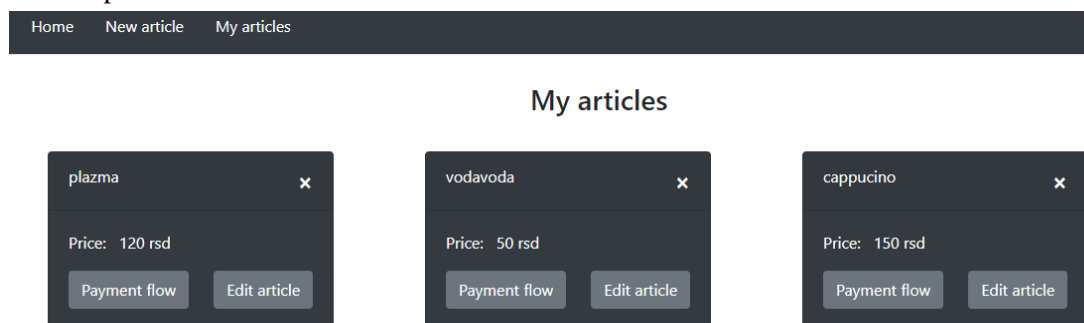
Podele plaćanja se prikazuju unutar pita dijagrama, iz *Chart.js* biblioteke. Pita dijagram je pogodan za reprezentaciju doprinosa različitih kategorija ukupnoj sumi. Koristi se za prikaz procentualnih ili proporcionalnih podataka. Jedini nedostatak je što se prilikom gomilanja kategorija gubi na preglednosti [35]. Uz pretpostavku da će korisnici za svaki artikal kreirati do 10 podela plaćanja (npr. za plaćanje računa, kirije, poreza, odgovarajuće donacije i za svoj profit), nedostatak preglednosti usled prevelikog broja kategorija ne bi trebalo da predstavlja problem.

Svaka podela plaćanja sadrži referencu na bankovni račun. Kako korisnik ne bi morao svaki put da unosi podatke o računu, računi koje unosi se pamte. Prilikom ukucavanja broja računa se vrši pretraga unesenih računa, i rezultati se prikazuju u vidu padajućeg menija. Klikom na neki od postojećih računa se automatski ispunjavaju podaci o računu u formi. Nakon što unese odgovarajuću sumu koju prenosi i klikne na dugme za podnošenje zahteva, nova podela plaćanja se dodaje u pita dijagram. Za svaku podelu se nasumično generiše boja prilikom dodavanja novih ili dobavljanja postojećih podela plaćanja. Postoji mogućnost da se pojave podele sa sličnim nijansama boje. Verovatnoća za tu situaciju je mala, ali boje nisu fiksne pa se mogu promeniti prilikom osvežavanja stranice.

Klikom na kategoriju, odnosno podelu plaćanja, u sklopu pita dijagrama je omogućena izmena ili brisanje postojećih podela. Pri tome, korisnik može modifikovati količinu novca koju prenosi, naziv primaoca za račun u banci i adresu primaoca. Nakon što izmeni podatke ili ukloni podelu, veličina i oznaka odgovarajućeg odsečka u sklopu pite se promeni ili potpuno nestane.

5.2.3. Početna verzija

Ideja početne verzije je bila da isto rešenje bude jednako upotrebljivo i za telefon i za *desktop* uređaj. U skladu sa tim, artikli su se izlistavali na posebnoj stranici, kao što se vidi na slici 5.2.2. Klikom na *payment flow* je izvršena redirekcija na odeljak za konfigurisanje podela plaćanja, a klikom na *edit article* je izvršena redirekcija na odeljak za izmenu podataka o artiklu.



Slika 5.2.2. – prikaz artikala u početnoj verziji aplikacije.

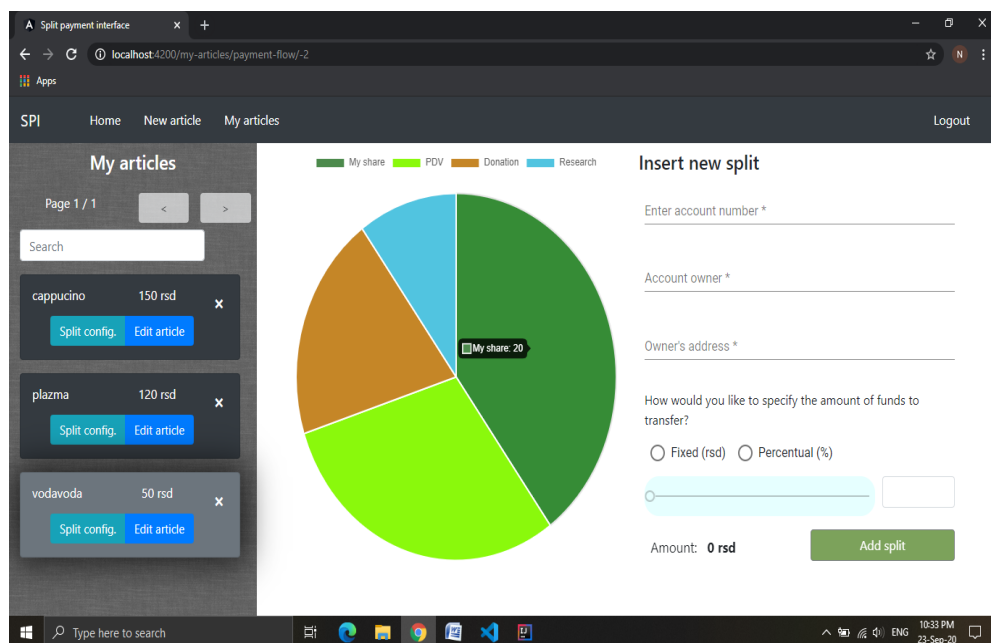
Odeljak za konfigurisanje podela plaćanja je tada u zaglavlju prikazao naziv i cenu artikla, da bi korisnik imao predstavu na koji artikal je kliknuo. Za male ekrane je ovo predstavljalo dobro rešenje, ali je kod velikih ekrana dovodilo do neiskorišćenosti u prostoru, preširoke forme za unos podataka i korisnik nije imao istovremeni pogled na navigacionu traku i na dugme za podnošenje zahteva, kao što je prikazano na slici 5.2.3.

Slika 5.2.3. – konfigurisanje podela plaćanja u početnoj, *desktop* verziji aplikacije.

5.2.4. Konačna verzija

U konačnoj verziji je došlo do poboljšanja upotrebljivosti aplikacije za *desktop* uređaje. Umesto posebne stranice za prikaz artikala, artikli se u konačnoj verziji prikazuju u navigacionoj traci koja je pozicionirana bočno na levoj strani ekrana. Pored artikala, traka sadrži polje za pretragu i dugmiće za navigaciju do odgovarajućih stranice u slučaju većeg broja artikala. Odabirom nekog artikla se promeni boja kartice, kako bi korisnik znao nad kojim artiklom sprovodi odgovarajuću operaciju.

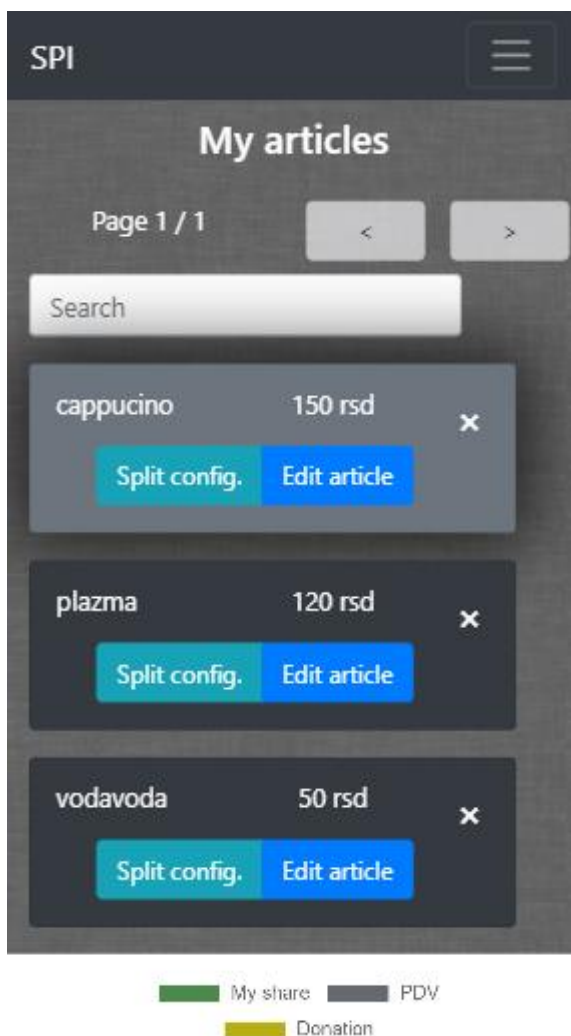
Na ovaj način je iskorišćen prazan prostor i korisnik ima dobru preglednost nad artiklima, pita dijagramom i formom za upravljanje podelom plaćanja. Takođe, dijagram je prebačen na levu stranu, ne bi li korisniku privukao veću pažnju. Boja dugmeta za slanje zahteva sugerise koji artikal se modifikuje, ili u slučaju dodavanja nove podele, sugerise boju kojom će se prikazati u dijagramu. Primer je prikazan na slici 4.2.4.



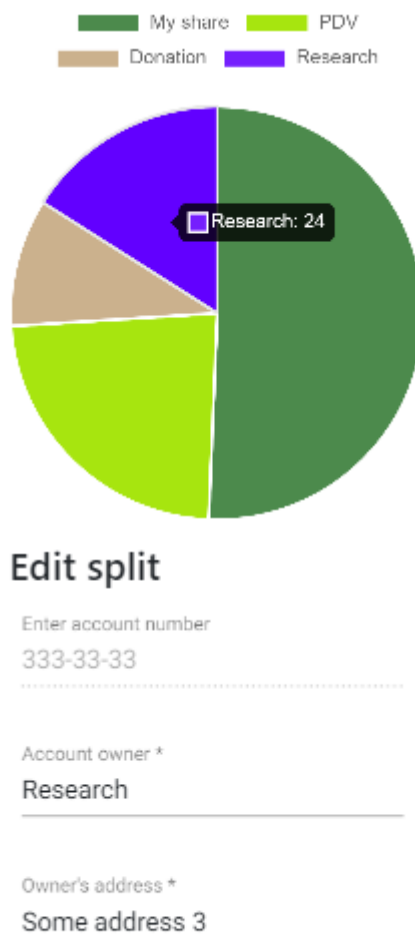
Slika 5.2.4. – pregled artikala i konfiguracija podela plaćanja u konačnoj, *desktop* verziji aplikacije.

Preglednost koja se postiže na *desktop* uređajima ne može da se prenese na uređaje manjih ekrana. Komponente za pregled artikala, prikaz dijagrama i forme za upravljanje podelom plaćanja u tom slučaju ne mogu da stanu jedna pored druge, tako da se komponente nižu jedna ispod druge.

U jednom trenutku korisnik posmatra jednu od navedene tri komponente, tako da je neophodno *scroll*-ovanje. Slike 5.2.5. i 5.2.6. obe prikazuju skaliranu verziju aplikacije za mobilni telefon. Na slici 5.2.5. je prikazan pregled artikala, dok na slici 5.2.6. se vidi da korisnik ne može istovremeno da sagleda celu formu sa prikazanim podacima i dijagram.



Slika 5.2.5. – pregled artikala u konačnoj verziji aplikacije skaliranoj na mobilni telefon.



How would you like to specify

Slika 5.2.6. – prikaz pita dijagrama i forme za upravljanje podelom plaćanja u konačnoj verziji aplikacije skaliranoj na mobili telefon.

Klikom na artikal se automatski vrši *scroll* do podnožja ekrana, da ne bi korisnik došao u situaciju gde misli da ništa nije postigao klikom. Prilikom kreiranja, modifikacije ili brisanja podele plaćanja korisnik ne vidi dijagram u potpunosti, tako da se, pored promena koje se odražavaju na dijagramu, ispisuju i odgovarajuće poruke kako bi korisnik znao da li je operacija izvršena uspešno ili neuspešno.

Dvoklikom ili klikom na odgovarajuću komponentu u dijagramu dva puta, vrši se automatsko *scroll*-ovanje do podnožja ekrana gde se nalazi

forma za upravljanje podelom plaćanja. Ova opcija nije podržana kod *desktop* uređaja pošto se istovremeno vide i dijagram i forma za upravljanje podelama plaćanja, pa nema potrebe za tim.

Prilikom skaliranja aplikacije na ekran srednje veličine komponente su takođe prikazane jedna ispod druge. Glavna razlika u odnosu na verziju za mobilni telefon, pored većeg dijagrama i forme za konfigurisanje podela, je način prikazivanja artikala. Kao što se može primetiti na slici 5.2.7. prilikom pregleda artikala na ekranu srednje veličine, artikli su raspoređeni u dve kolone. Na taj način se može delimično videti i dijagram, tako da korisnik klikom na artikal može uočiti da je klik miša registrovan i da je došlo do odgovarajuće promene. Po stranici se izlistavaju 4 artikla, tako da će korisnik uvek moći, makar delimično da vidi dijagram.



Slika 5.2.7. – pregled artikala u konačnoj verziji aplikacije skaliranoj na ekran srednje veličine.

6. POKRETANJE APLIKACIJE

Za pokretanje veb aplikacije, neophodno je pokrenuti *frontend* (*Angular*), *backend* (*spring boot*) i *MySql* servere.

6.1. Pokretanje *spring boot* servera

Za uspešno pokretanje servera, neophodno je uraditi sledeće:

- Instalirati programski jezik *java*, verziju 11.
- Instalirati *MySql* server za bazu podataka i pokrenuti ga.
- U *application.properties* fajlu, u okviru *resources* foldera treba zameniti *datasource* polja *username* i *password*, korisničkim imenom i lozinkom napravljenim prilikom instalacije *MySql* servera.

U načelu, *spring boot* aplikacija se može pokrenuti na dva načina:

- Preko razvojnog okruženja namenjenog za *java* programski jezik.
- Preko komandne linije.

Kao razvojno okruženje za javu mogu poslužiti *IntelliJ* ili *Eclipse*. U tom slučaju treba pronaći i otvoriti klasu pod nazivom *SpiBackendApplication* i pokrenuti glavnu, *main* metodu.

Kako bi se pokrenuo server preko komandne linije, neophodno je izvršiti instalaciju *maven*-a i odgovarajućih paketa. Ovo se postiže pozicioniranjem u folder *spi-backend* i pokretanjem sledeće komande:

- Za *Windows* operativni sistem: „*mvnw spring-boot:run*“
- Za *Linux* i *MAC* operativne sisteme se na početak komande doda *./*.

U zavisnosti od performansi računara, server bi trebao da se pokrene u roku od jednog minuta. Ukoliko nije došlo do greške, ispis bi trebao da ukazuje na uspešno kreiranje tabela u bazi podataka i uspešno pokretanje servera. Primer takvog ispisa je prikazan na slici 6.1.

```

create table account (id bigint not null auto_increment, number varchar(255) not null, recipient_address varchar(255) not null, recipient_name
create table app_user (id bigint not null auto_increment, email varchar(255) not null, password varchar(255) not null, primary key (id)) engine
create table article (id bigint not null auto_increment, name varchar(255) not null, price double precision not null, user_id bigint, primary k
create table payment_split (id bigint not null auto_increment, amount double precision not null, account_id bigint, article_id bigint, primary
alter table app_user add constraint UK_1j9d9a06i600gd43uu3km82jw unique (email)
alter table account add constraint FKf6hes5gpr1l00ww9xwi24vbq6 foreign key (creator_id) references app_user (id)
alter table article add constraint FK6w0tr7ts5bqd4j52s8oa126yf foreign key (user_id) references app_user (id)
alter table payment_split add constraint FK6uxl4gx43kk9era8i59oiumgl foreign key (account_id) references account (id)
alter table payment_split add constraint FK4k8a6n1a591i5lncbhf0l0aww foreign key (article_id) references article (id)
14:47:36,705 INFO [task-1] org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator: HHH000490: Using JtaPlatform implement
14:47:36,713 INFO [task-1] org.springframework.orm.jpa.AbstractEntityManagerFactoryBean: Initialized JPA EntityManagerFactory for persistence
14:47:37,352 WARN [main] org.springframework.boot.autoconfigure.orm.jpa.JpaBaseConfiguration$JpaWebConfiguration: spring.jpa.open-in-view is e
14:47:38,255 INFO [main] org.apache.juli.logging.DirectJDKLog: Starting ProtocolHandler ["http-nio-8080"]
14:47:38,294 INFO [main] org.springframework.boot.web.embedded.tomcat.TomcatWebServer: Tomcat started on port(s): 8080 (http) with context pat
14:47:38,295 INFO [main] org.springframework.data.repository.config.DeferredRepositoryInitializationListener: Triggering deferred initializati
14:47:38,526 INFO [main] org.springframework.data.repository.config.DeferredRepositoryInitializationListener: Spring Data repositories initial
14:47:38,536 INFO [main] org.springframework.boot.StartupInfoLogger: Started SpiBackendApplication in 9.525 seconds (JVM running for 10.697)

```

Slika 6.1 – ispis u konzoli prilikom uspešnog pokretanja *spring* aplikacije.

6.2. Pokretanje *Angular* servera

Za uspešno pokretanje servera, neophodno je instalirati sledeće:

- *Node.js* – *javascript* radno okruženje.
- *Npm package manager* – trebao bi se automatski instalirati prilikom instalacije *Node.js* okruženja.
- *Angular CLI* - alat za interfejs komandne linije koji omogućuje inicijalizaciju, razvoj, pokretanje i održavanje *Angular* aplikacija direktno iz komandne linije.

Nakon instalacije prethodnih paketa, treba se preko komandne linije pozicionirati u *spi-frontend* folder i pokrenuti sledeću komandu:

- *npm start*

Nakon uspešnog pokretanja servera, otvoriti pretraživač i izvršiti navigaciju do sledeće putanje:

- localhost:4200

Ukoliko je server uspešno pokrenut, trebala bi se prikazati početna strana aplikacije.

7. ZAKLJUČAK

U ovom radu je predstavljen prototip za interfejs sa podrškom za podeljeno plaćanje - aplikaciju koja omogućuje korisniku da jednostavno i efikasno za svoje artikle konfiguriše prenos finansijskih sredstava na određene bankovne račune, kreiranjem tzv. podela plaćanja. Podele bi omogućile korisniku da prilikom prodaje svakog artikla odredi koliko novca kome prenosi kako bi na kraju meseca uspeo automatski da namiri odgovarajuće dažbine.

U realnosti, implementacija potpuno funkcionalnog sistema predstavljenog u radu bi bila izuzetno zahtevna. Pored funkcionalnosti koje obezbeđuju prenos novca na račune i izvršenje odgovarajućih transakcija, treba osmisлити i dizajn koji bi bio jednostavan i razumljiv za korisnike aplikacije i upotrebljiv na različitim veličinama ekrana. Upravo to je motivacija i cilj ovog rada i najviše vremena je posvećeno tome.

Predstavljeno rešenje je najpreglednije, samim tim i najupotrebljivije za velike ekrane, odnosno *desktop* uređaje. Korisnik ima dobru preglednost nad svim elementima kojima rukuje, tako da može brzo i efikasno da izvršava odgovarajuće operacije. Kod ekrana manjih dimenzija, kaokod tableta ili mobilnih uređaja su elementi naslagani jedni ispod drugih. Korisnik je u tom slučaju primoran da „šeta“ gore-dole po ekranu. Radi unapređivanja iskustva na manjim ekranima, dodate su funkcionalnosti koje pozicioniraju korisnika na odgovarajući deo ekrana izvršavanjem odgovarajućih klikova, kako ne bi došao u situaciju gde misli da se ništa nije dogodilo klikom na neki element.

Za dalje unapređenje projekta bi bilo poželjno sprovesti anketu i ispitati korisnike koji dizajn im najviše odgovara, i šta bi dodali ili oduzeli. Pored toga, bilo bi korisno napraviti posebnu aplikaciju za mobilni telefon, kako bi se napravio poseban dizajn isključivo za ovu vrstu ekrana koji bi omogućio bolju preglednost nad elementima aplikacije i efikasnije sprovođenje osnovnih funkcionalnosti aplikacije. Pored toga, trebalo bi ispitati načine integrisanja projekta u realni sistem gde stvarno dolazi do autentifikacije korisnika, plaćanja u različitim valutama, prenosa novca i izvršavanja odgovarajućih transakcija.

LITERATURA

- [1] <https://www.websitesnmore.com.au/blog/marketplace-payments-split-payments-amongst-multiple-receivers-options/> - preuzeto Septembra 2020.
- [2] <https://www.investopedia.com/terms/p/paypal.asp> - preuzeto Septembra 2020.
- [3] <https://developer.paypal.com/docs/archive/adaptive-payments/integration-guide/APIIntro/> - preuzeto Septembra 2020.
- [4] <https://www.mulesoft.com/resources/api/what-is-an-api> - preuzeto Septembra 2020.
- [5] <https://developers.braintreepayments.com/guides/braintree-marketplace/overview> - preuzeto Septembra 2020.
- [6] <https://webkul.com/blog/magento2-marketplace-braintree-payment-gateway/> - preuzeto Septembra 2020.
- [7] https://en.wikipedia.org/wiki/Web_application - preuzeto Septembra 2020.
- [8] https://www.tutorialspoint.com/spring_boot/spring_boot_introduction.htm - preuzeto Septembra 2020.
- [9] <https://www.jrebel.com/blog/2020-java-technology-report> - preuzeto Septembra 2020.
- [10] <https://learn.onemonth.com/frontend-vs-backend-developers/> - preuzeto Septembra 2020.
- [11] <https://techterms.com/definition/framework> - preuzeto Septembra 2020.
- [12] <https://techterms.com/definition/platform> - preuzeto Septembra 2020.
- [13] <https://docs.oracle.com/javaee/6/firstcup/doc/gkhoy.html> - preuzeto Septembra 2020.
- [14] <https://www.baeldung.com/spring-vs-spring-boot> - preuzeto Septembra 2020.
- [15] <https://techterms.com/definition/standalone> - preuzeto Septembra 2020.
- [16] https://www.webopedia.com/TERM/P/production_grade.html - preuzeto Septembra 2020.

- [17] <https://angular.io/guide/architecture> - preuzeto Septembra 2020.
- [18] <https://www.programmableweb.com/news/what-angular-framework-and-why-should-developers-use-it/analysis/2017/03/06> - preuzeto Septembra 2020.
- [19] <https://opensource.com/resources/what-open-source> - preuzeto Septembra 2020.
- [20] <https://sr.wikipedia.org/sr/Mikroservisi> - preuzeto Septembra 2020.
- [21] <https://www.bgit.rs/blog/single-page-applications/> - preuzeto Septembra 2020.
- [22] <https://www.chartjs.org/> - preuzeto Septembra 2020.
- [23] <https://www.creativebloq.com/how-to/use-chartjs-to-turn-data-into-interactive-diagrams> - preuzeto Septembra 2020.
- [24] https://en.wikipedia.org/wiki/MIT_License - preuzeto Septembra 2020.
- [25] https://www.tutorialspoint.com/angular_material/angular_material_quick_guide.htm - preuzeto Septembra 2020.
- [26] <https://github.com/devversion/material-components> - preuzeto Septembra 2020.
- [27] <https://careerfoundry.com/en/blog/web-development/what-is-bootstrap-a-beginners-guide> - preuzeto Septembra 2020.
- [28] <https://www.slant.co/options/504/~bootstrap-review> - preuzeto Septembra 2020.
- [29] <https://dzone.com/articles/working-with-bootstrap-4-grid-system-for-creating> - preuzeto Septembra 2020.
- [30] <https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#repositories> - preuzeto Septembra 2020.
- [31] <https://sh.wikipedia.org/wiki/HTTP> - preuzeto Septembra 2020.
- [32] <https://www.webprogramiranje.org/web-servisi-osnove/> - preuzeto Septembra 2020.
- [33] <https://www.tutorialspoint.com/mysql/mysql-introduction.htm> - preuzeto Septembra 2020.
- [34] <https://code.tutsplus.com/articles/sql-for-beginners-part-3-database-relationships--net-8561> - preuzeto Septembra 2020.
- [35] https://en.wikipedia.org/wiki/Pie_chart - preuzeto Septembra 2020.

BIOGRAFIJA

Bojan Vlasonjić je rođen 15.06.1997. godine u Novom Sadu. Osnovnu školu „Jovan Dučić“ u Petrovaradinu završio je 2012. godine. Gimnaziju „Isidora Sekulić“ u Novom Sadu završio je 2016. godine. Iste godine upisao se na Fakultet tehničkih nauka, odsek Softversko inženjerstvo i informacione tehnologije. Položio je sve ispite predviđene planom i programom, i studije završava 2020. godine.