Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.42:004.738.12
DOI: https://doi.org/10.24867/16BE30Pecic

WEB APLIKACIJA U REALNOM VREMENU SA PODRŠKOM SIGNALR BIBLIOTEKE

REAL TIME WEB APPLICATION WITH SIGNALR LIBRARY SUPPORT

Dušan Pećić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – RAČUNARSTVO I AUTOMATIKA

Kratak sadržaj – *Ovaj rad bavi se pregledom web-a u realnom vremenu, kao i svih važnijih tehnika koje pružaju funkcionalnost u realnom vremenu. Takođe, detaljno je objašnjena biblioteka SignalR, koja je jedna od najčešće korišćenja biblioteka za razvoj aplikacija u realnom vremenu. Zatim se prolazi kroz implementaciju aplikacije za chat u realnom vremenu, kao i slanje trenutne lokacije korisnika.*

Ključne reči: *Real-time web, SignalR, ASP.NET Core*

Abstract – *Main goal of this paper is overview of real time web, as if all of the most important tehniques for allowing real time functionality. Also, it is precisely described SignalR library, which is the one of the most used libraries for development of real time web applications. Then we go through implementation of chat application in real time, as well as sending the user curent location.*

Keywords: *Real-time web, SignalR, ASP.NET Core*

1. UVOD

Web je u svojoj ranoj fazi postojanja bio vrlo drugačiji od onoga što danas poznajemo, te je zamišljen za rešavanje specifičnog problema deljenja dokumenata između udaljenih računara. Mogućnost *web*-a koja će se obrađivati u ovom radu su *web* aplikacije koje rade u realnom vremenu. Jedan od nedostataka Web-a je bio taj da je *web* stranicu bilo potrebno ponovo učitati kako bi ona dobila nove podatke od servera. Mogućnost dobijanja podataka bez ponovnog učitavanja stranice otvorila je veliki broj mogućnosti koje su drastično promenile način na koji koristimo *web*.

Danas je ovaj način *web* aplikacija opšte prihvaćen, te se koristi od strane velikog broja *web* stranica, počev od *chat* funkcionalnosti na društvenim mrežama do pretraživanja sadržaja u realnom vremenu na *web* pretraživačima. Razvoj *web* aplikacija u realnom vremenu imao je spor napredak uzrokovan ograničenjima od strane pretraživača, njegovom infrastrukturom i drugim faktorima. Danas postoje standardizovane tehnologije koje podržavaju svi popularniji internet pretraživači, te se koriste u velikom broju javno dostupnih biblioteka i alata za razvoj *web* aplikacija. Internet infrastruktura je takođe napredovala i samim tim više ne predstavlja prepreku u razvoju ovakvih vrsta aplikacija.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Željko Vuković, docent.

Još jedan od razloga jeste to što su korisnici bili izloženi korisničkom iskustvu koje pružaju *web* tehnologije u realnom vremenu i sada zahtevaju tu vrstu iskustva u aplikacijama koje koriste. *Web* tehnologije u realnom vremenu imaju brojne uobičajene upotrebe u svakodnevnom životu, a sve se više otkrivaju nove, inovativne potrebe.

2. RAZUMEVANjE WEB-A U REALNOM VREMENU

Realno vreme (eng. real time), najlakše se može objasniti kao najduže vreme potrebno za izvršenje zadatka u kojem bismo mogli reći da čitav sistem radi prema zadanim specifikacijama. Računanje u realnom vremenu (eng. real time computing), opisuje sisteme koji se temelje na opremi i aplikacijama koji imaju zadato realno vreme. Takve sisteme karakterizuju tri komponente i njihova međuzavisnost, a to su vreme izvršavanja, pouzdanost i okolina sistema. Vreme je najdragoceniji resurs za upravljanje sistemima u realnom vremenu. Zadaci moraju biti dodeljeni i raspoređeni za završetak pre njihovih krajnjih rokova. Pouzdanost se takođe veže na vreme izvršavanja jer svaki deo sistema pretpostavlja da će drugi delovi sistema ispravno izvršiti svoj zadatak u definisanim vremenskim okvirima, jer ukoliko to ne učine to može rezultirati pogrešnim radom sistema. Treća komponenta je okolina sistema i ona predstavlja sve ono što okružuje sistem. tj. spoljašnji delovi kojima se sistem bavi i bez kojih on ne bi imao svrhu [1].

2.1 Šta su *web* aplikacije u realnom vremenu?

Aplikacije u realnom vremenu postepeno dominiraju internetom jer pružaju savršenu ravnotežu informacija, funkcionalnosti, sadržaja i interaktivnosti što na kraju povećava angažovanje korisnika sa aplikacijom. Nepotrebno je naglašavati da aplikacije u realnom vremenu vide veliki obim prometa i stoga zahtevaju efikasno korišćenje hardverskih sredstava sa kojima raspolažu. Ove aplikacije moraju biti lagane, omogućiti iterativan razvoj i pružati sadržaj na organizovan i struktuiran način [2]. Kao što je već rečeno, pojavom *web* aplikacija dolazi do nove dimenzije razvijanja *web-a*, u smeru dinamičkog učitavanja sadržaja na zahtev klijenta, učitavanja sadržaja sa više servera odjednom na istu stranicu, slanja podataka, obrada i slično. Cilj je da korisnik dobije traženu informaciju koja će biti dinamički učitana, što nije problem, dok god klijent zahteva te informacije. Problem nastaje ukoliko se informacija promeni u međuvremenu ili ukoliko pristignu nove informacije koje su vezane za korisnike ili bitne za njegov rad. Da bi se informacija učitala na strani klijenta potrebno je da postoji zahtev koji klijent podnosi serveru

za učitavanje informacija. Sposobnost da klijent dobije informaciju u istom ili gotovo istom trenutku naziva se „*web* u realnom vremenu“.

2.2 Ajax

Asinhroni *JavaScript* i *XML* - *AJAX* (eng. asynchronous JavaScript and XML) je programska tehnika koja koristi JavaScript i objekat *XMLHTTPRequest* za razmenu podataka između *web* pretraživača i *web* servera.
AJAX se koristi za poboljšanje interaktivnosti *web* stranica i pruža programerima načine na koje se pojedinačni delovi stranice mogu ažurirati u realnom vremenu bez potrebe za ponovnim učitavanjem celokupnog sadržaja stranice. Ranije, ako je bilo potrebno ažurirati određeni deo sadržaja na *web* stranici, cela stranica bi se ponovo učitala sa *web* servera, uzrokujući prenos velike količine dupliranih podataka. Korišćenjem *AJAX*-a, sadržaj unutar *web* stranice može se ažurirati na osnovu radnji korisnika. *AJAX* se može opisati kao skup različitih tehnologija koje se mogu koristiti za implementaciju *web* aplikacije koja komunicira sa serverom u pozadini, bez ometanja trenutnog stanja stranice. Konekcije sa serverom se mogu kreirati asinhrono, što znači da korisnik ne mora čekati da server odgovori.

2.3 Polling

Polling je komunikaciona tehnika *web* aplikacija u realnom vremenu u kojoj klijent šalje zahtev serveru tražeći podatke u fiksnim vremenskim intervalima, nakon što dobije odgovor na prethodno poslat zahtev. Klijent ne zna kada server dobija nove podatke, samim tim on kontinuirano šalje nove zahteve serveru, kako bi što pre bio spreman za primanje novih podataka. Klijent šalje normalne *Http* zahteve ali kontinualno u određenim vremenskim intervalima. Server zatim momentalno šalje odgovor koji sadrži nove podatke ili samo prazan odgovor, ako nije bilo podataka za preuzimanje.

2.4 Long Polling

Long Polling je u suštini efikasniji oblik originalne tehnike anketiranja po parametrima trošenja resursa. Upućivanje ponovnih zahteva serveru troši resurse jer svaka nova dolazna veza mora biti uspostavljena, *Http* zaglavlja moraju biti parsirana, upit za nove podatke je potrebno izvršiti, kao i generisati i isporučiti odgovor od servera (obično bez novih podataka). Veza tada mora biti zatvorena i svi resursi očišćeni. Sve navedene nedostatke rešava *Long Polling*. Umesto velikog broja zahteva, klijent upućuje *Http* zahtev serveru na uobičajen način, sa namerom da zatraži podatke koje još nije primio. Ukoliko još nema novih podataka, server drži konekciju otvorenom dok ne pristignu podaci koje je korisnik zatražio. Ako posle određenog vremenskog perioda novi podaci i dalje nisu dostupni, tada server šalje odgovor pauze (eng. timeout response).

2.5 Server Sent Events

Server Sent Events je push tehnologija servera koja omogućava klijentu da prima automatski ažurirane podatke sa servera putem *Http* veze i opisuje kako serveri mogu započeti prenos podataka prema klijentima nakon uspostavljanja inicijalne konekcije. *SSE* je zasnovan na nečemu što se zove *Server Sent DOM Events*. Ideja je jednostavna, klijent (*web* pretraživač) se može pretplatiti na tok događaja koje generiše server, primajući ažurirane podatke kad se dogodi novi događaj. To je dovelo do kreiranja popularnog interfejsa *Event Source* koji prihvata *Http* vezu i održava vezu otvorenom dok iz nje preuzima dostupne podatke [3]. *SSE* je dizajniran da koristi *JavaScript EventSource API* (Programski interfejs aplikacije) u cilju pretplate na tok događaja u bilo kom popularnijem pretraživaču. Ukratko, *Server Sent Events* je tehnologija gde se ažurirani podaci guraju (umesto da se povlače ili traže od strane klijenta) sa servera u pretraživač.

2.6 Web Sockets

Pre nego što objasnim šta je *WebSockets* protokol i kako funkcioniše, osvrnućemo se na neke izazove koje ima za cilj da reši. Današnje aplikacije zahtevaju razmenu poruka sa malim kašnjenjem u realnom vremenu. Bez obzira da li pravite *web* ili mobilnu aplikaciju, korisnici očekuju mogućnost interakcije sa podacima što je moguće bliže realnom vremenu uz minimiziranje uticaja na njihovo celokupno korisničko iskustvo. Danas postoji mnogo aplikacija sa kojima ste verovatno u interakciji koje zahtevaju povezivanje preko interneta, a ipak pružaju korisničko iskustvo u realnom ili skoro realnom vremenu. Ako ste bili u interakciji sa *Twitter*-om ili *Facebook*-om, doživeli ste da se tokovi aktivnosti u stvarnom vremenu stalno ažuriraju po celom *web*-*u*, na mobilnom uređaju ili u pretraživaču, bez potrebe da pritisnete dugme za osvežavanje. *WebSockets* je mrežni protokol koji rešava većinu navedenih problema, omogućavajući brzu, sigurnu, dvosmernu komunikaciju između klijenta i servera bez oslanjanja na više *HTTP* veza. Za razliku od *HTTP*-a koji koristi zahtev-odgovor obrazac, *WebSockets* mogu slati poruke u bilo kom smeru u bilo kom trenutku. Najbolje od svega, *WebSockets* je interoperabilan i na više platformi na nivou pretraživača, izvorno podržava portove 80/443, kao i konekcije između domena

3. Pregled ASP.NET Core SignalR-a

Život se dešava u realnom vremenu i razmena informacija treba da se odvija na isti način. To je lakše reći nego učiniti, a izazovi nadilaze gomilu tehnologije. Međutim, kako komunikacija između servera/klijenata postaje sve pametnija, programeri su uspeli da iskoriste sofisticirane tehnike za izgradnju komunikacije u realnom vremenu između aplikativnih platformi. Sa modernim okvirima otvorenog koda, poput *SignalR*-a, koji apstraktuje složenost mrežnog steka, programeri se mogu usredsrediti na funkcionalnosti aplikacija u realnom vremenu koje omogućavaju moćna rešenja. *SignalR* je jedna od najčešće korišćenih biblioteka za rezvoj *web* aplikacija u realnom vremenu. Stvorena je 2011. godine od strane David Fowlera i Damian Edwardsa, koji sada igraju ključne uloge u razvoju *ASP.NET*-a. Da bi postigli razmenu poruka u realnom vremenu za *web*, programeri su koristili neefikasne tehnike, poput *AJAX*-a i long polling-a, kao i tehnologije poput *Server Sent Events*-a koje pretraživači nisu često primenjivali. *SignalR* je predodređen da reši ovaj problem i obezbedi laku podršku za funkcionalnosti u realnom vremenu na *ASP.NET* steku stvaranjem biblioteka na strani servera i klijenta koje uklanjaju

komplikacije ovih tehnologija [4]. Iza kulisa, *SignalR* pregovara o najboljem protokolu za određenu konekciju na osnovu onoga što podržavaju i server i klijent. Zatim pruža dosledan *API* za slanje i primanje poruka u realnom vremenu.

3.1 Šta je SignalR?

ASP.NET Core SignalR je biblioteka otvorenog koda koja pojednostavljuje dodavanje *web* funkcionalnosti u realnom vremenu aplikacijama. *Web* funkcionalnost u realnom vremenu omogućava serverskoj strani da dostupan sadržaj istog trenutka prosledi povezanim klijentima. *SignalR* se može koristiti za dodavanje bilo koje vrste *web* funkcionalnosti u „realnom vremenu“ u vašu *ASP.NET* aplikaciju. *SignalR* pruža jednostavan *API* za kreiranje udaljenih procedura od servera do klijenta (RPC) koje pozivaju JavaScript funkcije u klijentskim pretraživačima (i drugim klijentskim platformama) iz *.NET Core* koda na strani servera. *SignalR* takođe uključuje *API* za upravljanje vezama (na primer, događaje povezivanja i isključivanja) i grupisanje veza.
Pomoću *SignalR*-a možete slati poruke svim povezanim klijentima istovremeno ili ciljati određenog klijenta ili grupu klijenata [5]. Veza između klijenta i servera je trajna, za razliku od klasične *HTTP* veze, koja se ponovo uspostavlja za svaku komunikaciju. *SignalR* podržava funkciju "server push", u kojoj kod na serverskoj strani može da pozove klijentski kod u pretraživaču pomoću udaljenih poziva procedura (eng. Remote Procedure Calls), umesto zahtev-odgovor modela koji se i danas koristi na *web*-u. *SignalR* aplikacije mogu se proširiti na hiljade klijenata pomoću ugrađenih i nezavisnih provajdera za proširenje.

3.2 Razlike u odnosu na .NET SignalR verziju

Između *ASP.NET SignalR*-a i *ASP.NET Core SignalR*-a bilo je nekoliko značajnih promena, ovo su neke od njih [4] :

- Klijentska *JavaScript* biblioteka

U pretraživaču je najveća promena uklanjanje zavisnosti od *jQuery*-ja. Klijentska *JavaScript*/*TypeScript* biblioteka sada se može koristiti bez pozivanja na *jQuery*, što joj omogućava da se koristi sa okvirima kao što su *Angular*, *React* i *Vue* bez problema.

- Ugrađeni i prilagođeni protokoli

ASP.NET Core SignalR se isporučuje sa novim *JSON* protokolom za poruke koji nije kompatibilan sa ranijim verzijama *SignalR*-a. Osim toga, takođe ima drugi ugrađeni protokol zasnovan na *MessagePack*-u, binarnom protokolu koji ima manje korisnog opterećenja od *JSON*-a baziranom na tekstu.

- Dependency Injection

ASP.NET Core SignalR jednostavno koristi ugrađeni okvir za ubrizgavanje zavisnosti. Čvorovi u *ASP.NET Core SignalR*-u sada podržavaju konstruktor *Dependency Injection* bez dodatne konfiguracije, baš kao što to rade *ASP.NET Core* kontroleri ili razor pages.

- Rekonekcija

ASP.NET Core SignalR ne podržava automatsko ponovno povezivanje ili automatsko skladištenje poruka.

3.3 The Azure SignalR Service

Azure SignalR Service pojednostavljuje proces dodavanja *web* funkcionalnosti u realnom vremenu aplikacijama preko *HTTP*-a. Ova funkcionalnost u realnom vremenu omogućava servisu da prosleđuje ažuriran sadržaj povezanim klijentima, poput single page *web* stranice ili mobilne aplikacije. Kao rezultat toga, klijenti se ažuriraju bez potrebe za pozivanjem servera ili podnošenjem novih *HTTP* zahteva za ažuriran sadržaj [6].
Za šta se koristi *Azure SignalR Service*? Svaki scenario koji zahteva guranje (eng. pushing) podataka sa servera na klijenta u realnom vremenu može koristiti *Azure SignalR Service*. Tradicionalne funkcije u realnom vremenu koje često zahtevaju polling sa servera takođe mogu da koriste *Azure SignalR Service*.

4. IMPLEMENTACIJA *CHAT* APLIKACIJE

4.1 Specifikacija aplikacije

Glavni zadatak ovog projekta jeste *web* aplikacija *Chat* koja predstavlja upravo aplikaciju za ćaskanje preko interneta između različitih korisnika. Da bi korisnik imao mogućnost korišćenja aplikacije potrebno je da napravi svoj nalog, nakon čega može da može da se loguje i koristi aplikaciju. Pored klasične razmene poruka, korisnik ima mogućnost i slanja svoje lokacije.

4.2 Opis korišćenih alata i tehnologija

Aplikacija se sastoji od serverskog i klijentskog dela. Za serverski deo aplikacije je korišćena *ASP.NET.Core* tehnologija i *SQLite*, kao i *Microsoft Visual Studio* kao alat. Za klijentski deo aplikacije je korišćen *Blazor WebAssembly*, *Google maps*, dok je od alata takođe korišćen *Microsoft Visual Studio*.

4.2.1 Alati

Microsoft Visual Studio je integrisano razvojno okruženje, koje je korišćeno za sam razvoj aplikacije. *IDE* (Integrisano razvojno okruženje) je program bogat funkcijama koji se može koristiti za mnoge aspekte razvoja softvera.

4.2.2 Tehnologije

ASP.NET Core je *open-source* i *cloud* optimizovan *web* okvir za razvoj savremenih *web* aplikacija koje se mogu razvijati i pokretati na *Windows*, *Linux* i *Mac* operativnim sistemima.

Blazor je besplatan open-source *web* okvir, razvijen od strane *Microsoft*-a, koji omogućava kreiranje *web* aplikacija korišćenjem *C#* i *HTML*-a. *Blazor WebAssembly* je *single-page* okvir za kreiranje interaktivnih *web* aplikacija na strani klijenta, zasnovanog na *.NET*-u.

SQLite je *ACID*-kompatibilan ugrađen sistem za upravljanje bazama podataka sadržan u relativno maloj *C* programskoj biblioteci. Za razliku od klijent-server sistema za upravljanje bazama podataka, jezgro *SQLite*-a nije samostalan proces sa kojim aplikacija komunicira. Umesto toga, *SQLite* biblioteka je uvezana i postaje sastavni deo aplikacije.

HTML (engl. Hyper Text Markup Language) je opisni jezik specijalno namenjen opisu *web* stranica. Pomoću njega se jednostavno mogu odvojiti elementi kao što su naslovi, paragrafi, citati i slično.

CSS (engl. Cascading Style Sheets) je jezik formatiranja pomoću kog se definiše izgled elemenata *web* -stranice.

4.3 Implementacija *chat* funkcionalnosti pomoću SignalR-a

Blazor WebAssembly aplikacija je kreirana pomoću *Microsoft Visual Studio*-a. *Blazor WebAssembly* aplikacije rade na klijentskoj strani u pretraživaču na *.NET runtime* zasnovanom na *WebAssembly*-ju. *Blazor* aplikacija se izvršava direktno na niti korisničkog interfejsa pretraživača. Ažuriranje korisničkog interfejsa i rukovanje događajima odvija se u okviru istog procesa. Aplikacija je hostovana, što znači da je kreirana sa pozadinskom aplikacijom *ASP.NET Core* za opsluživanje datoteka. Dakle, aplikacija se sastoji iz klijentskog i serverskog dela. Hostovana klijentska aplikacija komunicira sa pozadinskom server aplikacijom preko mreže pomoću *web API*-ja ili *SignalR*-a. Glavni zadatak master rada je kreiranje *Chat* aplikacije koja će omogućiti razmenu poruka između dva autentifikovana korisnika u realnom vremenu. Da bi neautentifikovani korisnik postao autentifikovan, potrebno je da napravi svoj nalog u aplikaciji, nakon čega dobija mogućnost ćaskanja sa drugim korisnicima koji imaju nalog u ovoj aplikaciji. Za postizanje *Chat* funkcionalnosti klijent će biti hostovan na pretraživaču, dok će serverska stranom biti hostovana na računaru. Klijenti su povezani na server, kako bi mogli da primaju i šalju podatke na server.

4.3.1 Slanje lokacije pomoću *Chat* aplikacije

Dodatna funkcionalnost koja je implementirana u *Chat* aplikaciju predstavlja slanje lokacije drugom korisniku. Dobavljanje trenutne geografske lokacije se vrši pomoću *Geolocation API*-ja. Uobičajeni izvori informacija o lokaciji uključuju *GPS* i lokaciju izvedenu iz mrežnih signala, kao što su *IP* adresa, *WiFi*, *Bluetooth* itd. Kako bismo dobili trenutnu lokaciju pokreće se asinhroni zahtev za detekciju lokacije korisnika i traži najnovije informacije od hardvera za pozicioniranje. Kada se odredi položaj korisnika izvršava se definisana funkcija povratnog poziva. Uređajima za *GPS*-om može biti potrebno do nekoliko minuta za dobijanje potpuno tačne lokacije, zato manje precizni podaci (*IP* lokacija ili *WiFi*) mogu biti vraćeni. Nakon što korisnik odobri dozvolu, pomoću *Geolocation API*-ja dobijamo informacije o geografskoj širini i dužini, nakon čega je potrebno te podatke prikazati na mapi. Za prikaz lokacije u chat-u korisnika prvo je potrebno dodati *GoogleMaps API* skriptu.

5. ZAKLJUČAK

Ovim poglavljem završavamo sa pregledom razvoja *web* aplikacija u realnom vremenu. Kroz ovaj rad smo se na kratko osvrnuli na istoriju *web-a* i interneta, gde su pojašnjeni razlozi nastanka i razvoja *web* aplikacija u realnom vremenu. Takođe je objašnjen i sam pojam realnog vremena, njegov način računanja i vrste sistema koji funkcionišu u realnom vremenu. Prošli smo kroz obrazloženje aplikacija u realnom vremenu i njihovih karakteristika, gde smo se dotakli svih važnijih tehnika za postizanje realnog vremena sa prednostima i manama svake od njih. Detaljno je opisana *SignalR* biblioteka koja je korišćena pri implementaciji *Chat* aplikacije, njena istorija nastanka, transporti koje koristi, opis komunikacije čvora sa klijentima i prednosti koje su postignute sa ovom bibliotekom.

Završnim primerom prikazan je jedan mogući slučaj korišćenja ovakve vrste aplikacija, koja je izrađena uz pomoć pomenute *SignalR* biblioteke, jedne od najkvalitetnijih biblioteka za izradu *web* aplikacija u realnom vremenu današnjeg doba. *Web* se razvija iz dana u dan, a *web* aplikacije su već neko vreme deo naše svakodnevnice. Responzivne i kvalitetne aplikacije definitivno su tome doprinele, dok je napredak infrastrukture interneta povećao mogućnosti u izradi ovih aplikacija. Razvojem biblioteka poput one opisane u ovom radu, kao i razvojem *web-a* i interneta, doprinosi se tome da ne postoji razlog zašto različite aplikacije ne bi iskoristile neke od mogućnosti i prednosti koje pružaju *web* aplikacije u realnom vremenu. Na taj način će korisničko iskustvo, uspeh na tržištu, kao i sam razvoj ovakvih aplikacija biti podignut na jedan viši nivo.

6. LITERATURA

[1] Shin.K.G. & Ramanathan, P. (1994). Real-Time Computing: A New Discipline of Computer Science and Engineering

[2] https://www.clariontech.com/blog/key-considerations-while-developing-a-real-time-web-application

[3] https://developer.mozilla.org/en-US/docs/Web/API/EventSource

[4] https://www.codemag.com/article/1807061/Build-Real-time-Applications-with-ASP.NET-Core-SignalR

[5] https://docs.microsoft.com/en-us/aspnet/signalr/overview/getting-started/introduction-to-signalr

[6] https://docs.microsoft.com/en-us/azure/azure-signalr/signalr-overview

Kratka biografija:

Dušan Pećić rođen je 08.12. 1995. godine u Novom Sadu gde i dalje živi. Osnovnu školu „Jovan Jovanović Zmaj“ u Sr. Kamenici završio je 2010. godine kao nosilac Vukove diplome. Elektrotehničku školu „Mihajlo Pupin“ u Novom Sadu završio je 2014. godine. Iste godine upisuje Fakultet tehničkih nauka, smer Elektroenergetski softverski inženjering. U oktobru 2019. godine završava osnovne studije, nakon čega upisuje master studije takođe na Fakultetu tehničkih nauka, smer Računarstvo i Automatika, podsmer – Elektronsko poslovanje.