

Web servis za evidenciju studenata na nastavi

Marinić, Nikolina

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Bjelovar University of Applied Sciences / Veleučilište u Bjelovaru**

Permanent link / Trajna poveznica: <https://um.nsk.hr/um:nbn:hr:144:127432>

Rights / Prava: [In copyright](#)/[Zaštićeno autorskim pravom.](#)

Download date / Datum preuzimanja: **2024-12-21**



Repository / Repozitorij:

[Repository of Bjelovar University of Applied Sciences - Institutional Repository](#)



VELEUČILIŠTE U BJELOVARU
PREDDIPLOMSKI STRUČNI STUDIJ RAČUNARSTVO

**WEB SERVIS ZA EVIDENCIJU STUDENATA NA
NASTAVI**

Završni rad br. 07/RAČ/2020

Nikolina Marinić

Bjelovar, listopad 2020.



Veleučilište u Bjelovaru
Trg E. Kvaternika 4, Bjelovar

1. DEFINIRANJE TEME ZAVRŠNOG RADA I POVJERENSTVA

Kandidat: **Marinić Nikolina** Datum: 31.08.2020. Matični broj: 001863
JMBAG: 0314017954

Kolegij: **C# PROGRAMIRANJE**

Naslov rada (tema): **Web servis za evidenciju studenata na nastavi**

Područje: **Tehničke znanosti** Polje: **Računarstvo**

Grana: **Programsko inženjerstvo**

Mentor: **Krunoslav Husak, dipl.ing.rač.** zvanje: **predavač**

Članovi Povjerenstva za ocjenjivanje i obranu završnog rada:

1. Tomislav Adamović, mag.ing.el., predsjednik
2. Krunoslav Husak, dipl.ing.rač., mentor
3. Ante Javor, struč.spec.ing.comp., član

2. ZADATAK ZAVRŠNOG RADA BROJ: 07/RAČ/2020

U radu je potrebno izraditi Web servis koji služi za identifikaciju i evidenciju studenata na nastavi. Web servis se izrađuje s ciljem da se primjenjuje na Veleučilištu u Bjelovaru kao podrška hardverskom dijelu koji očitava studentske iskaznice. Backend će se izraditi pomoću programskih jezika C# i JavaScript. Za testiranje će se koristiti Postman.

Prijenos podataka će biti izveden pomoću XML i JSON prijenosnih formata. Za pohranu podataka koristit će se Microsoft SQL Server. Izrada aplikacijskog programskog sučelja izvodi se unutar .NET okruženja.

Zadatak uručen: 31.08.2020.

Mentor: **Krunoslav Husak, dipl.ing.rač.**



Zahvala

Zahvaljujem se svom mentoru Krunoslavu Husaku, dipl. ing. rač. na iznimnoj podršci i razumijevanju, jer mi je ukazao povjerenje i omogućio pisanje završnog rada.

SADRŽAJ

1.	UVOD	1
2.	WEB SERVIS	2
2.1.	<i>Primjena Web servisa</i>	3
2.2.	<i>Prednosti i nedostaci Web servisa</i>	3
2.3.	<i>Sigurnost Web servisa</i>	4
2.4.	<i>Vrste Web servisa</i>	5
2.4.1.	<i>SOAP</i>	5
2.4.2.	<i>REST</i>	7
2.5.	<i>Protokoli</i>	9
2.5.1.	<i>HTTP</i>	9
2.5.2.	<i>HTTPS</i>	10
2.6.	<i>Prijenosni formati</i>	11
2.6.1.	<i>XML</i>	11
2.6.2.	<i>JSON</i>	12
3.	KORIŠTENE TEHNOLOGIJE	15
3.1.	C#	15
3.1.1.	<i>.NET</i>	15
4.	PROCES IZRADE	17
4.1.	<i>Microsoft SQL Server Management Studio</i>	17
4.1.1.	<i>Povezivanje na SQL Server</i>	17
4.1.2.	<i>Struktura baze podataka</i>	19
4.2.	Visual Studio	20
4.2.1.	<i>Struktura Web servisa</i>	21
4.2.2.	<i>Izrada Web servisa</i>	22
4.3.	<i>Postman</i>	26
4.3.1.	<i>Upotreba Postmana</i>	27
5.	TESTIRANJE RADA	30
6.	ZAKLJUČAK	33
7.	LITERATURA	34
8.	OZNAKE I KRATICE	36
9.	SAŽETAK	38
10.	ABSTRACT	39
11.	PRILOZI	40

1. UVOD

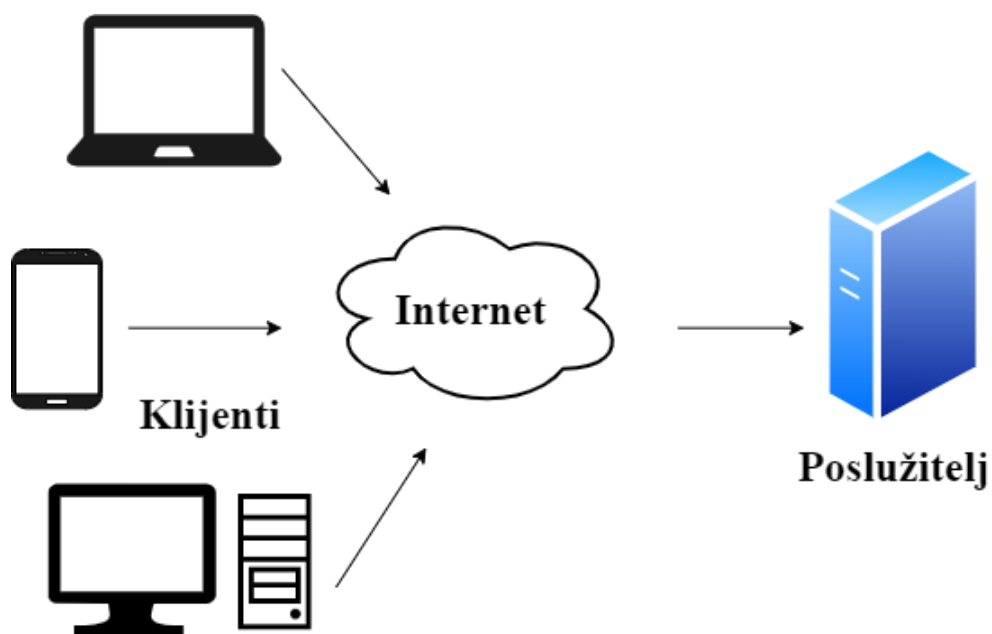
Web servis je standardizirani medij koji širi komunikaciju između klijenta i poslužiteljskih aplikacija. Mogu se tražiti putem mreže i također pozivati u skladu s tim. Kada se pozove, Web servis može pružiti funkcionalnost klijentu koji poziva taj Web servis. Servisi mogu biti privatni, ali i javno dostupni. U današnje vrijeme postoje dvije arhitekture za razvoj Web servisa, a to su: SOAP (*engl. Simple Object Access Protocol*) i REST (*engl. Representational State Transfer*) koji će u nastavku rada biti detaljnije pojašnjeni. Oba tipa arhitekture koriste: HTTP (*engl. Hyper Text Transfer Protocol*) i URI (*engl. Uniform Resource Identifier*) standard za identifikaciju servisa.

Cilj rada je napraviti Web servis za evidenciju studenata i profesora na nastavi koji će omogućiti prikaz, dodavanje, izmjenu i brisanje podataka vezanih za studenta ili profesora. Uz Web servis će biti izrađena baza podataka, a Postman će se koristiti za testiranje HTTP metoda.

Rad je podijeljen u jedanaest poglavlja. Drugo poglavlje opisuje osnovne pojmove koji omogućuju postojanje Web servisa te je opisana povijest SOAP i REST arhitekture. Također će se u drugom poglavlju dati pregled protokola HTTP i HTTPS (*engl. Hyper Text Transfer Protocol Secure*) i prijenosnih formata XML (*engl. Extensible Markup Language*) i JSON (*engl. Java Script Object Notation*). Treće poglavlje opisuje korištene tehnologije i programski jezik korišten prilikom izrade rada. Četvrto poglavlje opisuje proces izrade završnog rada koji se sastoji od nekoliko dijelova. Peto poglavlje omogućuje osvrt na ciljeve koji su postavljeni prilikom izrade rada i postignute rezultate. Posljednja poglavlja donose zaključak i sažetak koji se izvode iz potpoglavlja i odlomaka te literaturu, oznake i priloge.

2. WEB SERVIS

Web servis je usluga koju elektronički uređaj nudi drugom elektroničkom uređaju te međusobno komuniciraju putem WWW-a (*engl. World Wide Weba*). Također je zbirka otvorenih protokola i standarda koji se koriste za prijenos podataka između aplikacije i sustava. Web servisima je moguće pristupiti putem Interneta ili Internet mreže. Glavni protokol za prijenos podataka je HTTP. Formati za prijenos podataka su XML (*engl. Extensible Markup Language*) i JSON (*engl. Java Script Object Notation*). Svaki Web servis ima određeno sučelje preko kojega se vrši komunikacija s okolinom. Najčešće korišteni oblik Web servisa je klijent-poslužitelj koji je u nastavku rada prikazan slikom 2.1. S obzirom na pogodnosti koje pružaju Web servisi su postali obavezni dio u poslovanju gotovo svake organizacije. Povećanjem upotrebe Web servisa u svakodnevnom poslovanju povećava se potreba za metodologijama ispitivanja sigurnosti [1].



Slika 2.1: Princip rada Web servisa

2.1. Primjena Web servisa

Web servisi pružaju razne pogodnosti u svim poslovnim operacijama. Omogućuju raznim aplikacijama da međusobno komuniciraju i dijele podatke između različitih aplikacija ili platformi. Koriste se za osamostaljivanje aplikacijske platforme i tehnologije. Tehnologija izrazito brzo napreduje, a povezivanje i integracija usluga osigurava veće iskustvo za korisnike. Web servisi omogućuju učinkovitu distribuciju tehnologije u cijeloj mreži, postali su kritična okosnica našeg modernog svijeta kojima upravljaju uređaji [2]. Neki od poznatijih javnih Web servis su: Google [2], Facebook [2] i Twitter [2].

2.2. Prednosti i nedostaci Web servisa

Prednosti korištenja Web servisa:

1. Web servisi pomažu u razmjeni podataka između različitih aplikacija i platformi,
2. Brža komunikacija,
3. Pomažu u rješavanju problema interoperabilnosti (karakteristika proizvoda ili sustava),
4. Omogućuju aplikacijama međusobnu komunikaciju, razmjenu podataka i zajedničke usluge,
5. Omogućuju programerima upotrebu preferiranih programskih jezika,
6. Svaka usluga postoji neovisno o ostalim uslugama,
7. Posebno su dizajnirani kako bi se koristili kao zahtjev za Web stranice i pomažu prilikom primanja podataka,
8. Jeftiniji za upotrebu [3].

Nedostaci korištenja Web servisa:

1. HTTP protokol nije pouzdan, ne nudi nikakvu sigurnost prilikom isporuke,
2. Dostupnost,
3. Nepromjenjiva sučelja,
4. Prilikom stvaranju usluge za rad s kupcem, postoje specijalizirani zahtjevi koji moraju biti ispunjeni [5].

2.3. *Sigurnost Web servisa*

Sigurnost Web servisa je upitna zato što niti jedna metoda ispitivanja Web servisa nije stopostotna zaštita. Međutim, niti SOAP niti REST arhitekture ne postavljaju nikakve sigurnosne zahtjeve ili provjeru autentičnosti. Postoje dvije točke izvora ranjivosti. Te točke su: komunikacijski kanal koji služi za razmjenu poruka i domena posluživanja [6].

Sigurnosni problemi su:

1. *Povjerljivost*

Web servis se može sastojati od nekoliko povezanih aplikacija. Poruke prilikom slanja moraju biti šifrirane na svakom čvoru. Svaki čvor predstavlja potencijalnu slabu točku u lancu. W3C XML je standard koji pruža okvir za šifriranje i dešifriranje cijelih dokumenata ili samo točno određenih dijelova dokumenta. Komunikacija se može šifrirati putem SSL-a (*engl. Secure Socket Layer-a*) [6].

2. *Sigurnost mreže*

„Sigurnost mreže je izrazito bitna, ali isto tako i veliki problem. U ovom trenutku nema jednostavnog odgovora na ovaj problem, a o njemu se mnogo raspravlja. Ako postoji potreba za filtriranjem SOAP ili REST poruke, jedina mogućnost je filtrirati sve HTTP POST zahtjeve koji postavljaju vrstu sadržaja na *text/xml*“ [6].

3. *Autentifikacija*

Mogu se razmotriti sljedeće mogućnosti:

- HTTP – sadrži ugrađenu podršku za osnovnu provjeru autentičnosti te se usluge mogu približno zaštititi kao što su zaštićeni HTML dokumenti,
- Organizacija za unapređenje strukturiranih informacijskih standarda (OASIS) radi na razvoju standarda za sigurnost, Internet stvari, energetike i drugim područjima,

- SOAP digitalni potpis koristi kriptografiju javnog ključa koji služi za digitalno potpisivanje poruka. Klijentu ili poslužitelju omogućuje provjeru identiteta druge strane [6].

Ako dođe do napada prilikom prijenosa podataka ti podatci moraju biti zaštićeni. Štitimo ih uz pomoć metoda za zaštitu Web servisa. Metode su: cjelovitost, autentifikacija, povjerljivost, ne poricanje, raspoloživost i kontrola pristupa [6].

2.4. Vrste Web servisa

Web servise potrebno je implementirati na razne načine. Postoje dvije vrste Web servisa, a to su SOAP (*engl. Simple Object Access Protocol*) i REST (*engl. Representational State Transfer*) Web servisi [7].

2.4.1. SOAP

SOAP je 1998. godine dizajniran u svrhu pristupa objektima. Dizajnerali su ga Dave Winer, Don Box, Bob Atkinson i Mohsen Ali- Ghosein za Microsoft. Servisi su ranije koristili CORBA (*engl. Common Object Request Broker Architecture*) i DCOM (*engl. Distributed Component Object Model*) tehnologije. Glavna ideja stvaranja SOAP-a je osigurati programima koji su izrađeni na različitim platformama i programskim jezicima, da mogu sigurno razmjenjivati podatke. Nakon prvog predstavljanja postao je temeljni sloj složenijeg skupa Web servisa. SOAP je definiran kao protokol uz pomoć kojeg se jednostavno pristupa objektima. Ovaj protokol Web servisa razmjenjuje podatke koristeći XML i općenito HTTP za prijenos. SOAP protokol je zasnovan na XML-u koji služi za prijenos poruka [8].

SOAP poruka je XML dokument koji mora zadovoljiti nekoliko pravila:

- Poruka mora biti ispravno formatirani XML dokument i potrebno je koristiti SOAP oмотnicu i SOAP prostor imena prilikom prijenosa podataka,

- Ne smije zadržavati DTD (*engl. Document Type Definition*) referencu niti naredbe za procesiranje XML dokumenta [9].

SOAP poruka se sastoji od nekoliko elemenata:

- **Omotnica** (*engl. Envelope*) – zadaća omotnice je identificirati XML dokument kao SOAP poruku. Omotnica koristi *xmlns:soap* područje imena (*engl. Namespace*) i uvijek mora imati vrijednost koja definira omotnicu kao SOAP omotnicu.
- **Zaglavlje** (*engl. Header*) – unutar zaglavlja se nalaze informacije o primjeni poruke, identifikatoru prijenosa, pošiljatelju i primatelju poruke. Sadržaj i format podataka ovisi o vrsti SOAP poruke.
- **Tijelo** (*engl. Body*) – glavni dio poruke koji sadrži podatke o pozivu, odgovoru i poruke o greškama koje su se dogodile tijekom obrade.
- **WSDL** (*engl. Web Service Description Language*) – jezik baziran na XML-u koji omogućuje opis Web servisa i sučelja za njihovo korištenje. Nalazi se unutar tijela poruke [9].



Slika 2.3: Elementi SOAP poruke

Omogućuje klijentima da pozivaju Web usluge i primaju odgovore neovisno o jeziku i platformama. SOAP Web servisi imaju standarde za sigurnost i adresiranje. Prednosti SOAP Web servisa su: jednostavan za upotrebu, neutralan (koristi više standarda) i neovisan. Nedostaci su: teška postavka, teže se razvija i zamršenije kodiranje. SOAP omogućuje programerima povezivanje procesa koji se izvode na različitim operativnim sustavima Windows, Linux i MacOS za autentifikaciju i autorizaciju [8].

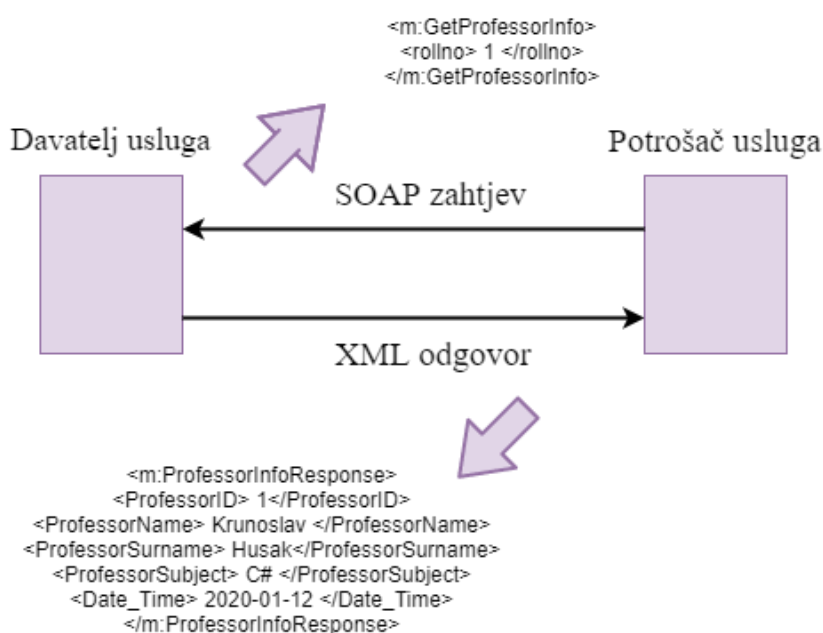
2.4.2. REST

REST je definirao Roy Fielding u svojoj doktorskoj disertaciji 2000. godine. Razvijao je REST paralelno s HTTP-om 1.1. od 1996 do 1999. godine na temelju postojećeg HTTP-a 1.0. REST definira skup ograničenja koji će se koristiti za stvaranje Web servisa. Web servisi koji odgovaraju arhitektonskom stilu REST nazivaju se RESTful Web servisi. RESTful Web servisi pružaju interoperabilnost između računalnih sustava na Internetu. Posebno je dizajniran za rad s datotekama ili objektima na određenim hardverskim uređajima. REST nije skup standarda ili pravila nego je stil softverske arhitekture. REST locira resurse na temelju URL-a (*engl. Uniform Resource Locator-a*) i izvodi radnju na temelju metode transportne radnje. Formate koje koristi su: HTML, XML i JSON. Podatci se najčešće prebacuju u JSON format. Može biti izvršen na bilo kom

serveru ili klijentu koji sadrži HTTP/HTTPS podršku. RESTful intenzivno koristi HTTP metode kako bi odredili operaciju koja se treba izvršiti [10].

Metode koje se najčešće primjenjuju su:

- **GET** – metoda koja se koristi za dohvaćanje svih podataka identificiranih pomoću *Request-URI*. Koristi se za čitanje i dohvat resursa. Smatraju se sigurnima i ne mijenjaju stanje resursa. *GET* je jedna od najčešćih HTTP metoda. Ne smije se koristiti prilikom rada s osjetljivim podacima [11]. Prikazana je slikom 2.2.



Slika 2.4: GET metoda

- **POST** – metoda *POST* koristi se za slanje podataka na poslužitelj. Podatci poslani na poslužitelj pohranjuju se u tijelu HTTP zahtjeva. *POST* zahtjevi se nikada ne pohranjuju u pred memoriju te ne ostaju u povijesti preglednika [11].
- **PUT** - se koristi za slanje podataka na poslužitelj i ažuriranje resursa. URI definira resurs koji želimo mijenjati. Uspješni zahtjev s ključnom riječi *PUT* trebao bi biti neovisan o broju izvršavanja tog zahtjeva. Razlika između *POST* i *PUT* metode je u tome što će *PUT* uvijek dati isti rezultat, a *POST* zahtjev drugačiji rezultat [11].

- **DELETE** – je idempotentna metoda, što znači da će njegov rezultat uvijek biti isti bez obzira koliko se puta pozivao. Metoda **DELETE** briše navedeni resurs [11].

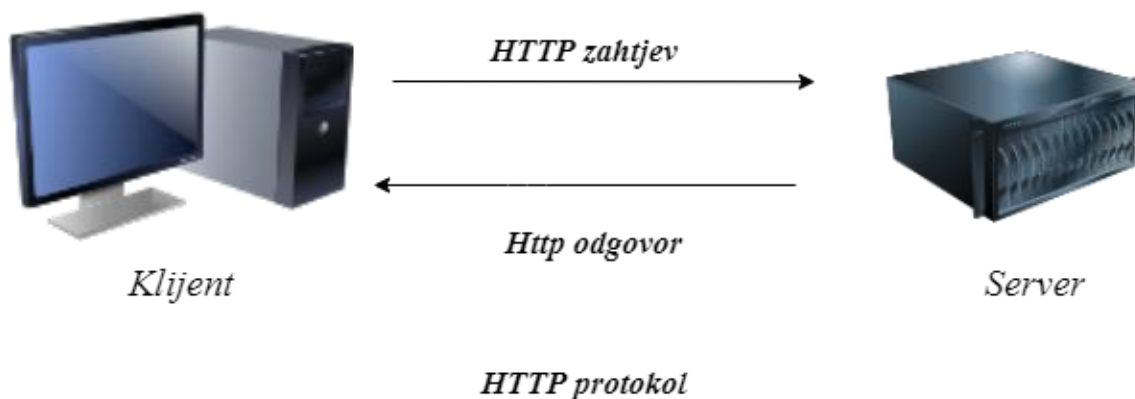
REST Web servis nije protokol kao SOAP već je koncept baziran na promjeni stanja klijenta. REST arhitektura se sastoji od klijenta i servera. Klijent daje zahtjev serveru, server procesira zahtjev i vraća odgovor klijentu. Prednosti REST Web servisa su: lagan, čitljiv ljudima i jednostavan za izradu. Nedostatak REST Web servisa je nedostatak standarda. REST također podržava pred memoriranje. RESTful Web servisi dominiraju iz više razloga, a REST se široko primjenjuje u razvoju API-ja. REST je preferiraniji od SOAP-a [10].

2.5. Protokoli

Protokol je jezik ili skup pravila uz pomoć kojeg računala u mreži međusobno komuniciraju. Temeljni su aspekt digitalne komunikacije. Rade u pozadini kako bi olakšali korištenje digitalne komunikacije i sakrili tehničke detalje od krajnjih korisnika. Protokole uspostavljaju međunarodne organizacije [13]. Postoje protokoli za razne namjene. Među najvažnijim protokolima se nalaze TCP (*engl. Transmission Control Protocol*), HTTP, HTTPS i drugi. U nastavku rada s pojašnjenjem će biti prikazani HTTP i HTTPS protokoli.

2.5.1. HTTP

HTTP se primjenjuje od 1990. godine. Originalno ga je osmislio Tim Berners Lee. Zasnovan je na TCP/IP-u, koji se koristi za isporuku podataka. Standardni protokol aplikacijskog sloja koji služi za razmjenu datoteka na World Wide Webu. HTTP je protokol zahtjeva/odgovora koji se temelji na arhitekturi zasnovanoj na klijentu/poslužitelju. Komunikacija između klijenta/poslužitelja se vrši zahtjevima/odgovorima tako da klijent šalje HTTP zahtjev na Web, Web poslužitelj prima zahtjev, poslužitelj pokreće aplikaciju za obradu zahtjeva, poslužitelj vraća HTTP odgovor na preglednik i za sam kraj klijent prima odgovor [14]. Komunikacija klijent/poslužitelj prikazana je slikom 2.5.



Slika 2.5: Komunikacija HTTP protokola

HTTP koristi određene metode zahtjeva za izvršavanje različitih zadataka. Primjeri metoda su: *GET, HEAD, PUT, POST, DELETE, TRACE, OPTIONS, CONNECTION* i *PATCH*. *GET* i *HEAD* su metode koje koriste svi HTTP poslužitelji, a ostale metode nisu podržane od strane svih poslužitelja [15]. HTTP protokol se razlikuje od ostalih TCP protokola, po tome što se konekcija i komunikacija sa serverom automatski prekida nakon izvršenog zahtjeva klijenta. Ova karakteristika HTTP protokola stvara probleme Web dizajnerima, s obzirom na to da nedostatak konekcije s poslužiteljem moraju nadoknaditi upotrebom drugih metoda [14].

2.5.2. HTTPS

Netscape Communications je stvorio 1994. godine. HTTPS je Internetski protokol nastao kombinacijom HTTP-a i SSL/TLS-a, izvorno se koristio s SSL protokolom. Koristi se za slanje podataka između Web preglednika i Web stranice. Šifriran je što omogućuje sigurniji prijenos podataka. To je osobito važno prilikom prijenosa osjetljivih podataka, poput prijave na bankovni račun. Za šifriranje komunikacije koristi protokol šifriranja. Protokol se naziva TLS (*engl. Transport Layer Security*) [16].

Ova vrsta sustava koristi dva različita ključa za šifriranje komunikacije:

- **Privatni ključ** – ovim ključem upravlja samo vlasnik Web stranice. Ključ „živi“ na Web poslužitelju i koristi se za dešifriranje podataka šifriranih javnim ključem [17].
- **Javni ključ** - je dostupan svima koji žele interakciju s poslužiteljem. Informacije koje je šifrirao javni ključ može dešifrirati samo privatni ključ [17].

HTTPS sprječava Web stranicama emitiranje informacija. Promet je šifriran i ako paketi budu otkriveni nije moguće ništa napraviti jer će biti prikazani nasumični znakovi. Trenutna verzija HTTPS-a je formalno određena u svibnju 2000. godine od strane RFC-a (*engl. Request For Comments*) [16].

2.6. Prijenosni formati

Prilikom komunikacije sustava izrazito je bitno postaviti određene standarde kako bi se mogla odvijati komunikacija između servisa/servisa ili između klijenta/servisa. U suprotnom bi svaka tvrtka kreirala svoje standarde te bi klijenti morali razvijati posebnu programsku podršku kako bi mogli obavljati svoj posao. U svrhu prijenosa podataka između servisa/servisa ili klijenta/servisa koriste se različiti prijenosni formati kao što su: XML, JSON, SMILE, CSV, BSON i drugi. Od navedenih, najširu upotrebu imaju XML i JSON formati.

2.6.1. XML

XML je označeni jezik koji definira skup pravila za kodiranje dokumenata, identificira podatke, pohranjuje ih i organizira. XML se ne klasificira kao programski jezik zato što ne izvodi proračune ili algoritme. Najčešće se pohranjuje u tekstualnu datoteku. Može ga koristiti bilo koji pojedinac, grupa pojedinaca ili tvrtka koja želi dijeliti informacije. Osnovni blok XML dokumenta je definiran oznakama. Sadrži oznake za početak i kraj. Podržava ugniježdene elemente ili elemente unutar elemenata. Elementi unutar XML dokumenta moraju biti pravilno ugniježdene te pravilno zatvoreni. Prilikom zatvaranja dokumenta pravilo nalaže da se koristi isto ime koje je korišteno za otvaranje elemenata. Atributi unutar XML dokumenta moraju biti pisani malim slovima i moraju imati pridruženu vrijednost.

Tri važne karakteristike:

1. XML je proširiv- omogućuje stvaranje vlastitih oznaka ili jezika koji odgovaraju aplikaciji,
2. XML je javni standard – razvijen od strane organizacije W3C (*engl. World Wide Web Consortium*) i dostupan je svima,
3. XML nosi podatke, ne predstavlja ih – omogućuje pohranu podataka bez obzira na koji su način predstavljeni [18].

XML se može koristiti za razmjenu podataka između organizacija i sustava te se može koristiti za pojednostavljivanje HTML dokumenta za velike Web stranice. Pretjerana upotreba XML-a je umanjila njegovu stvarnu vrijednost. XML-ova snaga leži u jednostavnosti [18].

```
1  <?xml version="1.0" encoding="UTF-8" ?>
2  <Student>
3      <StudentID>1</StudentID>
4      <StudentJMBAG>1233456781</StudentJMBAG>
5      <StudentName>Nikolina</StudentName>
6      <StudentSurname>Marinić</StudentSurname>
7      <StudentGender>Ž</StudentGender>
8      <StudentCity>Slavonski Brod</StudentCity>
9      <StudentDirection>Računarstvo</StudentDirection>
10     <YearOfCollage>3</YearOfCollage>
11 </Student>
```

Slika 2.6: Primjer XML dokumenta

2.6.2. JSON

JSON je nastao iz potrebe za komunikacijom protokola poslužitelj/preglednik. Douglas Crockford prvi je popularizirao JSON format. Temelji se na JavaScript-u. Prvi put je standardiziran 2013. godine kao ECMA-404 [19]. JSON je tekstualni format koji ne

ovisi o programskom jeziku i računalnoj platformi. Imena JSON datoteka koriste ekstenziju *.json*. Služi za pohranu podataka na organiziran, lako pristupačan način. Preferiran je format za kreiranje Web servisa [20].

Osnovni tipovi podataka JSON-a su:

- **Broj** (*engl. Number*) – podržava cijele (*Int*) i decimalne (*Float*) brojeve, s tim da su decimalni brojevi odvojeni točkom,
- **Tekst** (*engl. String*) – tekstovi su razdvojeni s dvostrukim navodnicima,
- **Logički tip podataka** (*engl. Boolean*) – sadrži dvije vrijednosti istina/laž,
- **Polje** (*engl. Array*) - sortirani niz vrijednosti koji mogu biti bilo kojeg tipa podataka. Mogu sadržavati druga polja. Zatvoreni su unutar uglatih zagrada, vrijednost se odvaja zarezom,
- **Objekt** (*engl. Object*) – ne sortirani niz oblika ključ/vrijednost. Svaki objekt se nalazi unutar vitičastih zagrada, nakon svakog ključa se postavlja dvotočka, a ključ/vrijednost su odvojeni zarezom,
- **Null vrijednost** – predstavlja praznu vrijednost [21].

Koristi se za serijalizaciju i prijenos podataka putem mreže. XML koristi oznake dok JSON ne što ga čini lakšim za pisanje i razumnijim za čitanje. Može se koristiti s modernim programskim jezicima. Ima podršku za stvaranje, dekodiranje i čitanje. JSON format sve više zamjenjuje XML format [20].

```
{  
  "Student":{  
    "StudentID": 1,  
    "StudentJMBAG": 1233456781,  
    "StudentName": "Nikolina",  
    "StudentSurname": "Marinić",  
    "StudentGender": "Ž",  
    "StudentCity": "Slavonski Brod",  
    "StudentDirection": "Računarstvo",  
    "YearOfCollage": 3  
  }  
}
```

Slika 2.7: Primjer JSON dokumenta

3. KORIŠTENE TEHNOLOGIJE

3.1. C#

Anders Hejlsberg je glavni dizajner i arhitekt C# u Microsoftu. Microsoft je prvi put upotrijebio naziv C# 1988. godine. Odobren je od strane Europskog udruženja proizvođača računala (ECMA) i Međunarodne organizacije za standardizaciju (ISO). C# je suvremeni i objektno orijentirani programski jezik. Dizajniran je za tvrtke koje grade sve vrste softvera uz pomoć jednog programskog jezika. Sintaksa C# vrlo je izražajna, ali isto tako jednostavna i laka za učenje. Podržava generičke metode i tipove podataka, podržava koncept enkapsulacije, nasljeđivanja i polimorfizma. Varijable i metode su uvrštene u definiciju klase. Klasa se može izravno naslijediti od jedne obiteljske klase, ali također može implementirati i bilo koji broj sučelja. C# je brz u usporedbi s nekim drugim programskim jezicima visoke razine. Ne dopušta pretvorbe tipova podataka koji mogu dovesti do gubitaka istih. Baziran je na pisanju efikasnog koda. Može se koristiti za stvaranje Windows klijentskih aplikacija, XML Web servisa, aplikacija klijent/poslužitelj i još mnogo toga. Najnovija verzija je 8.0, koja je objavljena 2019. godine [22].

3.1.1. .NET

.NET je dizajniran i razvijen od strane Microsoft-a. Prva verzija je izašla 2002. godine. Podržava više od 60 programskih jezika od kojih je 11 razvijeno od strane Microsoft-a. Programski jezici koje je dizajnirao Microsoft su: C#.NET, C++.NET, VB.NET, JSCRIPT.NET, IRON PYTHON i drugi. C# programi rade na .NET platformi. .NET je integrirana komponenta sustava Windows koja uključuje CLR (*engl. Common Language Runtime*). CLR je virtualna mašina na kojoj se izvršava asemblerski kod i omogućuje pristup raznim servisima. Pruža i druge usluge kao što su rukovanje iznimkama i upravljanje resursima. Kod koji je izvršen od strane CLR-a ponekad se naziva „upravljivi kod“. Interoperabilnost ključna je značajka .NET-a. .NET uključuje opsežnu i organiziranu biblioteku od preko 4000 klasa koje pružaju široku lepezu korisnih funkcionalnosti. Također omogućuje rad s kolekcijama, mrežom, kriptografiju i još mnogo toga. Omogućuje da različite aplikacije imaju identične dijelove koda, koje se koriste kao

komponente i tako omogućuju višestruku primjenu istog koda što štedi vrijeme i novac [23].

4. PROCES IZRADE

4.1. *Microsoft SQL Server Management Studio*

SSMS je softverska aplikacija koja je prvi put pokrenuta 2005. godine. Koristi se za upravljanje, konfiguriranje i administraciju svih komponenti. Smatra se relacijskom bazom podataka. SQL Server je baza podataka koja se ugnijezdila između manjih i srednjih baza. Omogućuje izvršavanje zadataka bilo programski ili putem GUI-a. Radnje koje može izvoditi su: stvaranje, izmjena i brisanje podataka i objekata, stvaranje i održavanje korisničkih računa i uloga, uvoz i izvoz podataka iz jedne baze u drugu bazu, održavanje sigurnih kopija i ispitivanje svoje baze podataka. Podržava većinu administratorskih zadataka i održava integrirano okruženje za upravljanje i autorizaciju *SQL Server Database Enginea*. Microsoft je 2015. godine objavio da buduće verzije SSMS-a budu bile objavljene bez obzira na izdanja SQL Servera [24].

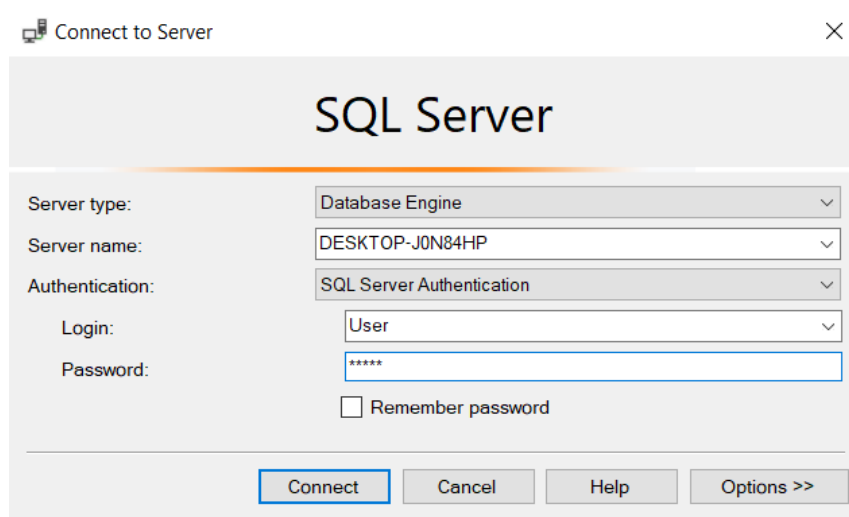


Microsoft SQL Server Management Studio 18

Slika 4.1: Logo SQL Server Management-a

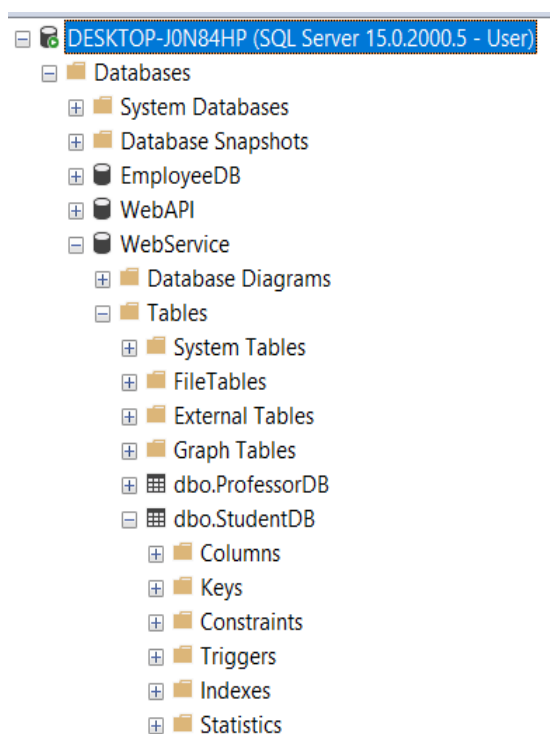
4.1.1. *Povezivanje na SQL Server*

U početku rada SQL Server Management Studio zatražit će povezivanje. Kako bi se povezao sa SQL Serverom i tako mogao obavljati ostale zadatke koji su potrebni za izradu rada. Konekcija je u nastavku prikazana slikom 4.2.



Slika 4.2: SQL Server konekcija

Nakon povezivanja kreirat će se baza podataka *New Database*. U prikazanom slučaju je kreirana baza koja se naziva *WebService*. Unutar *WebService*-a su kreirane dvije tablice *StudentDB* i *ProfessorDB*. Što je moguće vidjeti na slici 4.3.

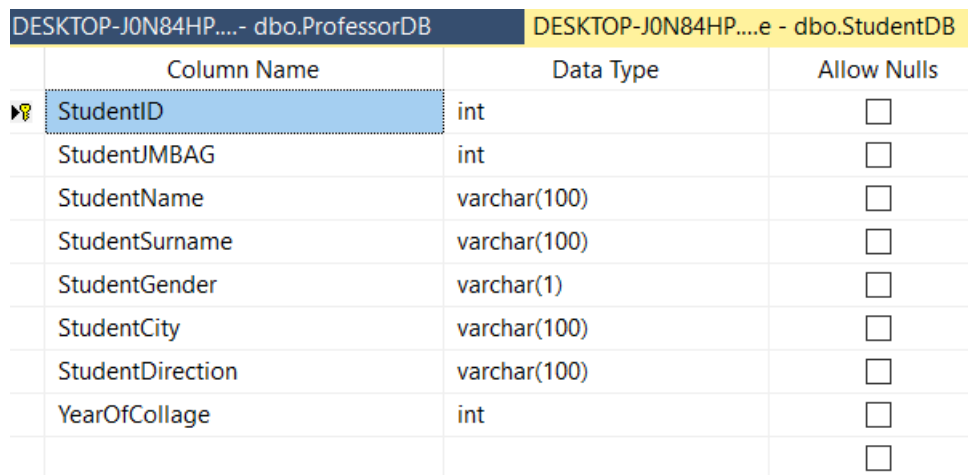


Slika 4.3: Baza i kreirane tablice unutar baze

4.1.2. Struktura baze podataka

Potrebno je izraditi shemu podataka s kojima će Web servis raspolagati u budućnosti. *WebService* sastoji se od dvije tablice:

1. **Tablica StudentDB** – sadrži sve podatke koji su bitni za identifikaciju studenta na predavanju, laboratorijskim vježbama ili ispitu. Tablica se može iskoristiti za pohranu podataka koji će se unositi. Podatak ID u tablici *StudentDB* je primarni ključ. Uz pomoć ID-a i JMBAG-a će se identificirati studenti. Što je moguće vidjeti na slici 4.4.



Column Name	Data Type	Allow Nulls
StudentID	int	<input type="checkbox"/>
StudentJMBAG	int	<input type="checkbox"/>
StudentName	varchar(100)	<input type="checkbox"/>
StudentSurname	varchar(100)	<input type="checkbox"/>
StudentGender	varchar(1)	<input type="checkbox"/>
StudentCity	varchar(100)	<input type="checkbox"/>
StudentDirection	varchar(100)	<input type="checkbox"/>
YearOfCollage	int	<input type="checkbox"/>

Slika 4.4: Tablica StudentDB

2. **Tablica ProfessorDB** – je druga tablica unutar koje su pohranjeni podatci vezani za profesora koji će u tom trenutku održati predavanje, laboratorijske vježbe ili ispit. ID je primarni ključ i služi za identifikaciju profesora. Što je moguće vidjeti na slici 4.5.

Column Name	Data Type	Allow Nulls
ProfessorID	int	<input type="checkbox"/>
ProfessorName	varchar(100)	<input type="checkbox"/>
ProfessorSurname	varchar(100)	<input type="checkbox"/>
ProfessorSubject	varchar(100)	<input type="checkbox"/>
Date_Time	date	<input type="checkbox"/>
		<input type="checkbox"/>

Slika 4.5: Elementi tablice ProfessorDB

Visual Studio je potrebno povezati s bazom podataka kako bi se podatci pohranili. Baza podataka je povezana s Web Servisom tako da se unutar *Web.config-a* dodaje *connectionString* unutar kojeg se nalaze svi bitni podatci vezani za *WebService*. Što je prikazano slikom 4.6.

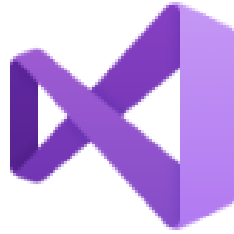
```
<connectionStrings>
  <add name="StudentAppDB" connectionString="Data Source=.;Initial Catalog=WebService;Integrated Security=True" providerName="System.Data.SqlClient" />
</connectionStrings>
```

Slika 4.6: Konekcija baze podataka i Visual Studia

4.2. Visual Studio

Prva verzija Visual Studija bila je Visual Studio 97. Visual Studio je integrirano razvojno okruženje. Integrirano razvojno okruženje je program s raznim značajkama koji se može koristiti za mnoge aspekte razvoja softvera. Razvijen je od strane Microsofta. Prva inačica Visual Studija je izdana 1997. godine. Podržava 36 programskih jezika, a jedni od njih su C, C++, C#, Visual Basic, XML, HTML, CSS i mnogi drugi. Dostupan je kao podrška i za druge jezike poput Pytona, Node.js-a i Ruby-a. Koristi se za razvoj računalnih programa, Web aplikacija, Web stranica i mobilnih aplikacija. Posjeduje uređivač koda, program za ispravljanje pogrešaka i alat za dizajn GUI-a. Dostupan je za Windows i Mac. Postoje tri izdanja Visual Studija: Professional, Community i Enterprise. Najnovije izdanje

Visual Studija „Community“ dostupno je besplatno. Visual Studio Community orijentiran je prema individualnim programerima i malim timovima. Najnovija podržana verzija Visual Studija je 2019 [25].

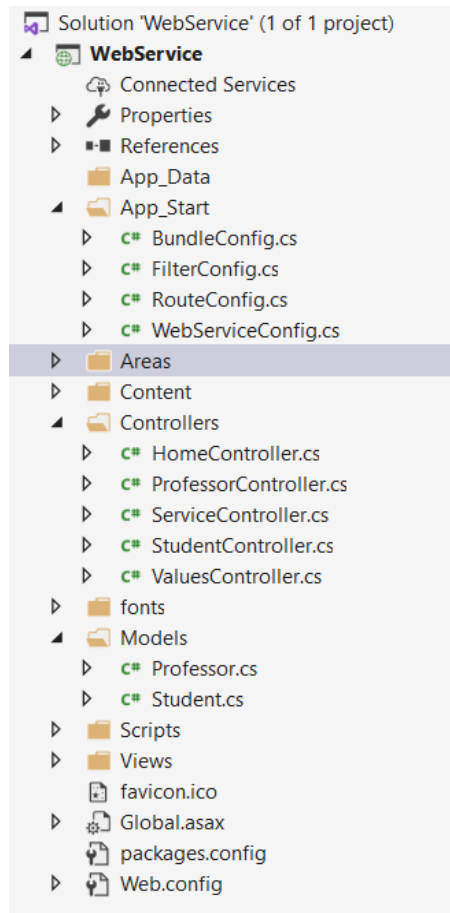


Visual Studio 2019

Slika 4.7: Microsoft Visual Studio logo

4.2.1. Struktura Web servisa

Za početak kreiranja projekta potrebno je izabrati predložak (*engl. Template*) za danji rad. Izabrani predložak je ASP.NET Web Application (*.NET Framework*). Web servis sastojat će se od više programskih komponenti. Prilikom izrade projekta nisu korištene napredne tehnologije. Sama izrada Web servisa nije bila pretjerano teška, ali to ne umanjuje kvalitetu izrađenog projekta. Projekt se sastoji od više datoteka koje su prikazane slikom 4.8. Neke od bitnijih su: mapa *Controllers* unutar koje se nalaze svi potrebni kontroleri za danji rad *HomeController*, *ProfessorController*, *StudentController* i *ValuesController*, mapa *Models* sadrži klase *Professor* i *Student*, mapa *App_Start* unutar koje se nalazi klasa za konfiguraciju te je za sam kraj bitno spomenuti datoteku *Web.config* unutar koje su smještene informacije vezane za bazu podataka. Bitno je napomenuti da se radi o RESTful modelu Web servisa.



Slika 4.8: Komponente Web servisa

4.2.2. Izrada Web servisa

Prilikom izrade Web servisa korišteni su XML i JSON prijenosni formati. Nakon istraživanja vezanog za Web servise JSON format je format koji se sve više koristi. Iz tog razloga su XML podatci pretvoreni u JSON. Prilikom pretvorbe podataka će biti korišten *JsonFormatter* koji je definiran unutar *WebServiceConfiguration-a*. Nakon toga unutar mape *Model* kreirat će se dvije klase:

1. ***Student.cs*** – u klasi se nalaze podatci vezani za identifikaciju studenta koji će se unositi prilikom predavanja, laboratorijskih vježbi ili ispita. Svi uneseni podatci predstavljaju stupce u tablici baze podataka *StudentDB*. Podatci od kojih se klasa sastoji su: ID koji se stvara automatski prilikom unosa studenta, JMBAG, Ime, Prezime, Spol, Grad, Smjer i godina obrazovanja. Što će biti prikazano slikom 4.9. Na osnovu tih podataka kreirat će se *StudentController* koji sadrži HTTP metode.

```

namespace WebService.Models
{
    public class Student
    {
        1 reference
        public int StudentID { get; set; }
        2 references
        public int StudentJMBAG { get; set; }
        2 references
        public string StudentName { get; set; }
        2 references
        public string StudentSurname { get; set; }

        2 references
        public string StudentGender { get; set; }

        2 references
        public string StudentCity { get; set; }
        2 references
        public string StudentDirection { get; set; }

        2 references
        public int YearOfCollage { get; set; }
    }
}

```

Slika 4.9: Model podataka za studente

2. **Professor.cs** – u klasi se nalaze podatci vezani za identifikaciju profesora koji u tom trenutku održava predavanje, laboratorijske vježbe ili ispit. Svi uneseni podatci predstavljaju stupce u tablici baze podataka *ProfessorDB*. Podatci od kojih se klasa *Professor* sastoji su: ID koji se stvara automatski prilikom unosa profesora, Ime, Prezime i Datum. Što će biti prikazano slikom 4.10. Na osnovu tih podataka će biti kreiran *ProfessorController* čija svrha će biti objašnjena u nastavku.

```

namespace WebService.Models
{
    2 references
    public class Professor
    {
        1 reference
        public int ProfessorID { get; set; }
        2 references
        public string ProfessorName { get; set; }
        2 references
        public string ProfessorSurname { get; set; }
        2 references
        public string ProfessorSubject { get; set; }
        2 references
        public DateTime Date_Time { get; set; }
    }
}

```

Slika 4.10: Model podataka za profesore

Kreirane klase će biti od pomoći prilikom stvaranja kontrolera. Stvorena su dva nova kontrolera *StudentController* i *ProfessorController* koja podržavaju HTTP metode *GET*, *POST*, *PUT* i *DELETE*. *POST* metoda bit će definirana tako da se moraju unijeti sve vrijednosti vezane za studenta ili profesora kako bi zahtjev bio odobren, ako sve vrijednosti nisu unesene zahtjev neće biti odobren i osoba neće biti pohranjena u bazu. Jako je bitno napomenuti da su vrijednosti ograničene i da ne postoji mogućnost unosa podataka izvan tih okvira. Što je moguće vidjeti uz pomoć slike 4.11.

```

// POST: service/Student
0 references
public string POST(Student student)
{
    try
    {
        DataTable table = new DataTable();
        string query = @"

            INSERT INTO dbo.StudentDB (StudentJMBAG,StudentName,StudentSurname,StudentGender,StudentCity,StudentDirection,YearOfCollage)

            VALUES

            (" + student.StudentJMBAG + @"'
            ,'" + student.StudentName + @"'
            ,'" + student.StudentSurname + @"'
            ,'" + student.StudentGender + @"'
            ,'" + student.StudentCity + @"'
            ,'" + student.StudentDirection + @"'
            ,'" + student.YearOfCollage + @"'

            )";

        using (var con = new SqlConnection(ConfigurationManager.ConnectionStrings["StudentAppDB"].ConnectionString))
        using (var cmd = new SqlCommand(query, con))
        using (var da = new SqlDataAdapter(cmd))

        {
            cmd.CommandType = CommandType.Text;
            da.Fill(table);
        }

        return "Record Inserted Successfully!";
    }
    catch (Exception ex)
    {
        return "Error while inserting record!";
    }
}

```

Slika 4.11: POST metoda za studenta

Uz pomoć *GET* metode svi uneseni podatci će biti prikazani unutar Web servisa. Podatci su prikazani u JSON formatu koji će kasnije biti iskorišten u svrhu Postman-a. Prilikom pokretanja Web servisa korisniku se nudi mogućnost da vidi sve unesene studente ili profesore. *PUT* metoda služi za izmjenu i manipulaciju s podacima. Metoda je kreirana na način da nije moguće promijeniti ID studenta ili profesora i JMBAG studenta. Prilikom izmjene podataka ako je izmjena uspješna profesor ili student bit će pohranjeni u bazu, ali ako je zahtjev odbijen dobit će se povratna informacija u obliku poruke da nešto nije uredu. Za sam kraj ostaje *DELETE* metoda koja služi za brisanje studenata ili profesora tako da se unese ID i ako osoba postoji s tim ID-om bit će uklonjena iz baze podataka. Što je moguće vidjeti na slici 4.12.

```

// DELETE: service/Professor
0 references
public string DELETE(int ID)
{
    try
    {
        DataTable table = new DataTable();

        string query = @"
            DELETE FROM dbo.ProfessorDB WHERE ProfessorID =" + ID;

        using (var con = new SqlConnection(ConfigurationManager.ConnectionStrings["StudentAppDB"].ConnectionString))
        using (var cmd = new SqlCommand(query, con))
        using (var da = new SqlDataAdapter(cmd))
        {
            cmd.CommandType = CommandType.Text;
            da.Fill(table);
        }

        return "Record Delete Successfully!";
    }
    catch (Exception)
    {
        return "Error while deleting record!";
    }
}
}
}

```

Slika 4.12: DELETE metodu za profesora

4.3. Postman

Postman je API (*engl. Application Programming Interface*) razvojno okruženje koje pomaže pri testiranju, dokumentiranju i nadgledanju. Nudi elegantno programsko sučelje pomoću kojeg se mogu poslati razni zahtjevi. Smatra se komercijalnom aplikacijom. Na samom početku je razvijen kao dodatak za Web preglednik Google Chrome, ali s vremenom je prerastao u samostalnu aplikaciju i samo korištenje Postman-a kao dodatka Web pregledniku više nije podržano. Unutar Postman-a je ugrađena gotovo svaka funkcionalnost koja bi mogla biti potrebna programerima. Omogućava izradu različitih vrsta HTTP zahtjeva, amortizaciju, spremanje srodnih upita u kolekcije, spremanje podataka za kasniju upotrebu i još mnogo toga. Pomoću njega je moguće saznati gdje se nalazi greška i jesu li HTTP upiti koje smo napravili ispravni. Jedan je od najpoznatijih alata za razvoj API-ja. Postman je vrlo jednostavan i prilagodljiv, što ga čini

pogodnim za sigurno testiranje. Dostupan je za Windows, Linux, Mac i također kao aplikacija za Chrome [26].

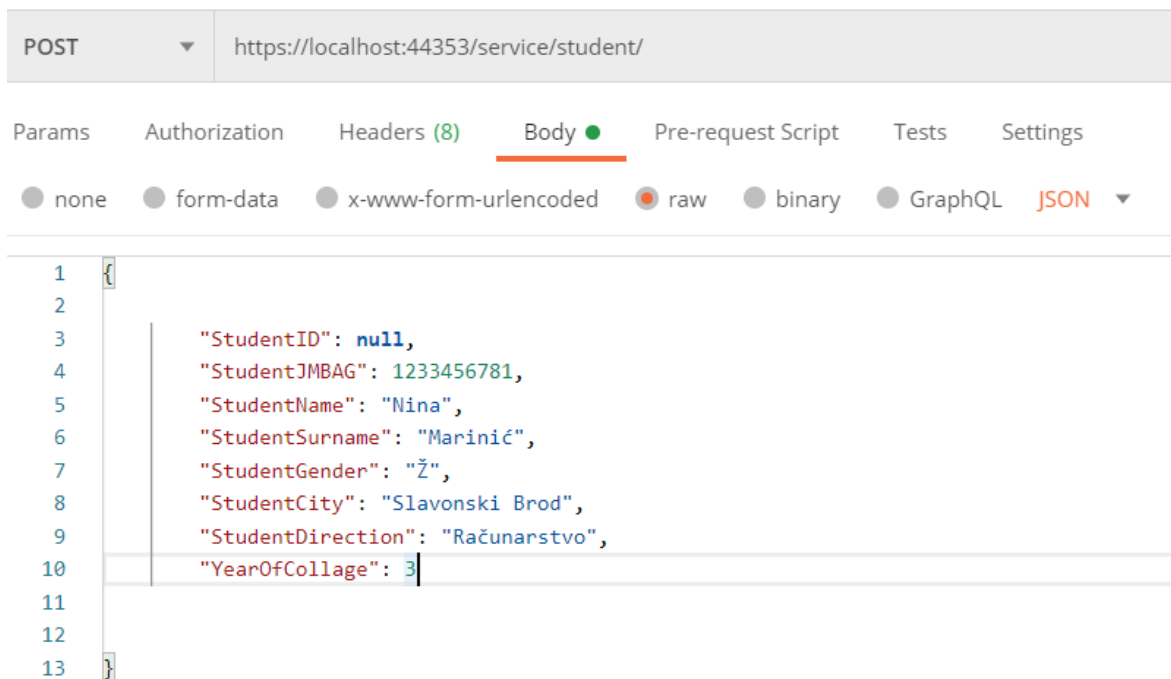


Postman

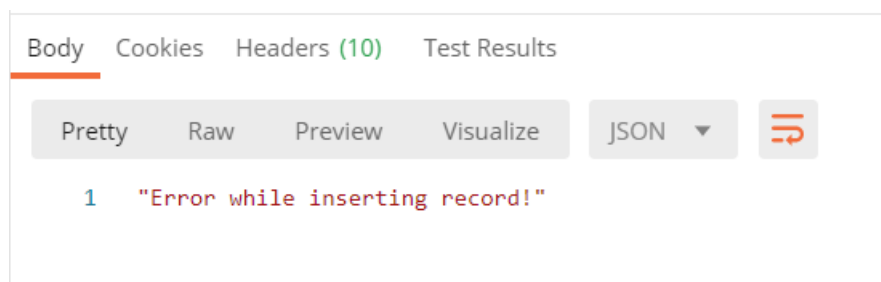
Slika 4.13: Postman logo

4.3.1. Upotreba Postmana

Zadaća Postman-a je testiranje HTTP metoda *GET*, *POST*, *PUT* i *DELETE*. Odnosno jesu li metode ispravne. Prilikom pokretanja zahtjeva potrebno je znati URL, metodu i druge vrijednosti kao što su *Auth* i parametri. Svaka krajnja točka dostupna je na određenom URL-u. Za početak će biti potrebno pokrenuti Web servis i kopirati URL koji će se koristiti prilikom testiranja unutar Postman-a. Nakon kopiranog URL-a odabrat će se metoda *POST*, otići će se do stupca *Body*, ispod njega izabrat će se *Raw*, JSON kao prijenosni format i unutar *Raw-a* unijeti će se novi student ili profesor. Nakon što se svi parametri ispoštuju poslat će se zahtjev i čekat će se odgovor na zahtjev. Zahtjev i odgovor na zahtjev moguće je vidjeti na slikama 4.14 i 4.15.



Slika 4.14: Zahtjev POST metode



Slika 4.15: Odgovor na zahtjev POST metode

Nakon *POST* metode na red dolazi *GET* metoda koja će prikazati sve dodatne studente ili profesore ako je zahtjev uspješan. *PUT* metoda služi za izmjenu podataka. Slična je kao *POST* metoda, ali prilikom unosa parametara potrebno je unijeti točno odgovarajući ID u suprotnom zahtjev neće proći. Za kraj je ostala *DELETE* metoda koja služi za brisanje studenata ili profesora iz baze podataka. Prilikom slanja zahtjeva unutar URL-a potrebno je unijeti ID parametar kako bi se utvrdilo postojanje studenta ili profesora. Ako postoji odgovarajući ID zahtjev će biti odobren i osoba će biti uklonjena iz baze, ali ako ID ne postoji zahtjev je odbijen. Što je moguće vidjeti na slici 4.16.


DELETE ▼ https://localhost:44353/service/student/6

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON** ▼

```
1 {
2
3   "StudentID": null,
4   "StudentJMBAG": 1233456782,
5   "StudentName": "Nina",
6   "StudentSurname": "Marinić",
7   "StudentGender": "Ž",
8   "StudentCity": "Slavonski Brod",
9   "StudentDirection": "Računarstvo",
10  "YearOfCollage": 3
11
12 }
13
```

Body Cookies Headers (10) Test Results

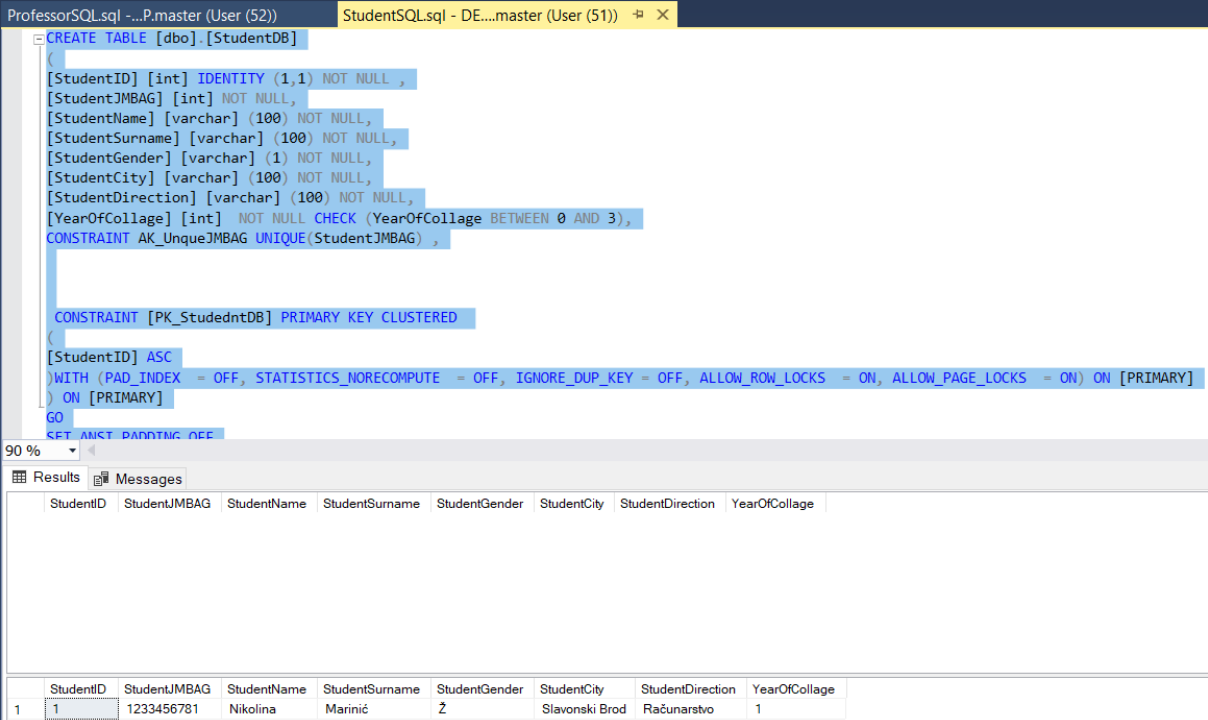
Pretty Raw Preview Visualize JSON ▼ 

```
1 "Record Delete Successfully!"
```

Slika 4.16: Zahtjev i odgovor DELETE metode

5. TESTIRANJE RADA

Kreiranje Web servisa započeto je uz pomoć baze podataka, a završilo se u Visual Studiju. Kada se baza podataka i Visual Studio povežu, završni rezultat je Web servis za unos, izmjenu, ispis i brisanje svih podataka vezanih za studenata ili profesora s fakulteta. Bitno je napomenuti da je riječ o RESTful Web servisu. Web servis za evidenciju studenata na nastavi ja nastao zato što mnogi studenti lažiraju svoju prisutnost. Iz tog razloga će svaki student biti unesen i pohranjen u bazu podataka u trenutku kada je to potrebno. Studenti će se identificirati uz pomoć ID-a i JMBAG-a, a ostali podatci su radi formalnosti. Osim toga korišten je i Postman u svrhu testiranja HTTP metoda koje se primjenjuju unutar Web servisa.



The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows the SQL script for creating the `StudentDB` table. The script includes the following details:

- Table name: `[dbo].[StudentDB]`
- Columns: `[StudentID]` (int, IDENTITY (1,1) NOT NULL), `[StudentJMBAG]` (int, NOT NULL), `[StudentName]` (varchar (100) NOT NULL), `[StudentSurname]` (varchar (100) NOT NULL), `[StudentGender]` (varchar (1) NOT NULL), `[StudentCity]` (varchar (100) NOT NULL), `[StudentDirection]` (varchar (100) NOT NULL), and `[YearOfCollage]` (int, NOT NULL CHECK (YearOfCollage BETWEEN 0 AND 3)).
- Constraints: `CONSTRAINT AK_UnqueJMBAG UNIQUE (StudentJMBAG)` and `CONSTRAINT [PK_StudedntDB] PRIMARY KEY CLUSTERED ([StudentID] ASC) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]`.

The bottom pane shows the results of the query, displaying a table with the following data:

StudentID	StudentJMBAG	StudentName	StudentSurname	StudentGender	StudentCity	StudentDirection	YearOfCollage
1	1233456781	Nikolina	Marinić	Ž	Slavonski Brod	Računarstvo	1

Slika 5.1: Tablica *StudentDB*

Slika 5.1. prikazuje sve parametre unutar tablice *StudentDB*. Za početak se može vidjeti da u tablici nema korisnika, ali nakon što je korisnik dodan uspješno je i pohranjen u bazu podataka.

The screenshot shows a Postman interface for a PUT request. The URL is `https://localhost:44353/service/student/`. The request body is a JSON object with the following fields:

```

1 {
2   "StudentID": 1,
3   "StudentJMBAG": 1233456781,
4   "StudentName": "Nina",
5   "StudentSurname": "Marinić",
6   "StudentGender": "Ž",
7   "StudentCity": "Slavonski Brod",
8   "StudentDirection": "Računarstvo",
9   "YearOfCollage": 3
10 }
11
12
13

```

The response body is:

```

1 "Record Update Successfully!"

```

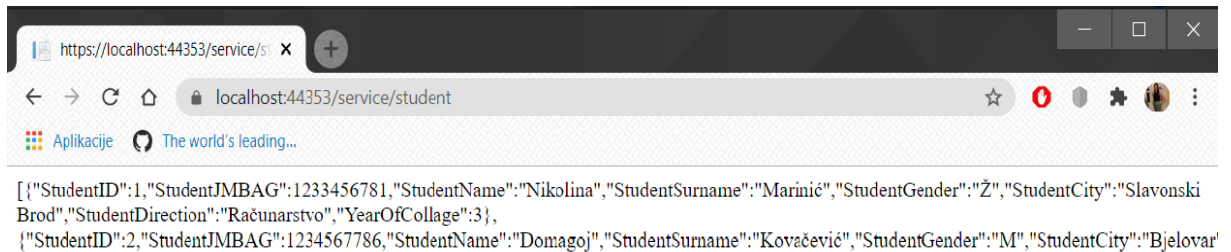
Slika 5.2: Zahtjev i odgovor na zahtjev PUT metode

Postman služi za testiranje HTTP metoda. Što je moguće vidjeti na slici 5.2. U prikazanom primjeru se koristi *PUT* metoda zato što je potrebno izmijeniti neke podatke kod dodanog studenta. Rezultat izmjene podataka je uspješan što je vidljivo na slici 5.3.

StudentID	StudentJM...	StudentNa...	StudentSur...	StudentGe...	StudentCity	StudentDir...	YearOfColl...
1	1233456781	Nikolina	Marinić	Ž	Slavonski Br...	Računarstvo	3
2	1234567786	Domagoj	Kovačević	M	Bjelovar	Računarstvo	3

Slika 5.3: Pohranjeni podatci unutar baze

Slika 5.3. prikazuje izmjenu kod studenta koji sadrži Ime {Nina}. Cilj je bio izmijeniti ime studentice. Kao što je prikazano izmjena je uspješna. Nakon izmjene podataka dodan je još jedan korisnik.



Slika 5.4: Podatci Web servisa u JSON formatu

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<DataSet xmlns="http://tempuri.org/">
  <xs:schema xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" id="NewDataSet">
    ...
  </xs:schema>
  <diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">
    <NewDataSet xmlns="">
      <Table diffgr:id="Table1" msdata:rowOrder="0">
        <StudentJMBAG>1233456781</StudentJMBAG>
        <StudentName>Nikolina</StudentName>
        <StudentSurname>Marinić</StudentSurname>
        <StudentGender>Ž</StudentGender>
        <StudentCity>Slavonski Brod</StudentCity>
        <StudentDirection>Računarstvo</StudentDirection>
        <YearOfCollage>3</YearOfCollage>
      </Table>
    </NewDataSet>
  </diffgr:diffgram>
</DataSet>
```

Slika 5.5: Podatci Web servisa u XML formatu

Na slikama 5.4 i 5.5 je moguće vidjeti ispis podataka unutar Web servisa. Podatci su ispisani u JSON i XML prijenosnom formatu. Također se na slikama može vidjeti da je izmjena podataka uspješna. Prilikom testiranja rada zadani cilj je postignut. Web servis ispunjava sve parametre koji su potrebni za njegovu izradu i spreman je za rad.

6. ZAKLJUČAK

Web servis je standardizirani medij koji širi komunikaciju između klijenta i poslužiteljskih aplikacija. Postoje dvije arhitekture za razvoj Web servisa, a to su SOAP i REST. Međusobnu komunikaciju vrše putem *World Wide Weba*. Glavni protokol za prijenos podataka je HTTP. Web servis koristi XML i JSON kao prijenosne formate. Najčešće korišteni oblik Web servisa je klijent-poslužitelj. Web servisi omogućuju učinkovitu distribuciju tehnologije u cijeloj mreži. Postali su kritična okosnica našeg modernog svijeta kojim upravljaju uređaji. Web servisi su obavezni dio u poslovanju gotovo svake organizacije. Povećanjem upotrebe Web servisa u svakodnevnom poslovanju povećava se potreba za metodologijama ispitivanja sigurnosti. Web servis je izrađen s ciljem da se primjenjuje na Veleučilištu u Bjelovaru. Omogućuje prikaz, dodavanje, izmjenu i brisanje podataka vezanih za studenta ili profesora. Rad je uspješno testiran, ispunjava sve dogovorene parametre i može se koristiti u budućnosti.

7. LITERATURA

- [1] Cleo. What Are Web Services? Easy-to-Learn Concepts with Examples [Online]. 2018. Dostupno na: <https://www.cleo.com/blog/knowledge-base-web-services>. (13.6.2018.)
- [2] Web Programiranje. Web servisi (osnove) [Online]. 2019. Dostupno na: <https://www.webprogramiranje.org/web-servisi-osnove/>. (11.9.2019.)
- [3] Guru99. What are Web Services? Architecture, Types, Example [Online]. 2020. Dostupno na : <https://www.guru99.com/web-service-architecture.html>. (23.7.2020.)
- [4] Microsoft. Advantages & Disadvantages of Webservices [Online]. 2007. Dostupno na: <https://social.msdn.microsoft.com/Forums/en-US/435f43a9-ee17-4700-8c9d-d9c3ba57b5ef/advantages-amp-disadvantages-of-webservices?forum=asmxandxml>. (19.11.2007.)
- [5] informIT. Disadvantages and Pitfalls of Web Services [Online]. 2003. Dostupno na: <https://www.informit.com/articles/article.aspx?p=31729>. (16.5.2003.)
- [6] Tutuorialspoint. Web Services- Security [Online]. 2002. Dostupno na: https://www.tutorialspoint.com/webservices/web_services_security.htm. (1.7.2002.)
- [7] java T point. What is Web Service [Online]. 2019. Dostupno na: <https://www.javatpoint.com/what-is-web-service>. (11.6.2019.)
- [8] informIT. Simple Object Access Protocol (SOAP) [Online]. 2002. Dostupno na: <https://www.informit.com/articles/article.aspx?p=26666&seqNum=3#>. (10.5.2002.)
- [9] Tutotialspoint. SOAP- Message Structure [Online]. 2016. Dostupno na: https://www.tutorialspoint.com/soap/soap_message_structure.htm. (14.9.2016.)
- [10] COMPUTERWORLD. Representational State Transfer (REST) [Online]. 2007. Dostupno na: <https://www.computerworld.com/article/2552929/representational-state-transfer--rest-.html>. (6.8.2007.)
- [11] w3schools.com. HTTP Request Methods [Online]. 2012. Dostupno na : https://www.w3schools.com/tags/ref_httpmethods.asp. (18.10.2012.)
- [12] Tech Terms. Protocol [Online]. 2019. Dostupno na: <https://techterms.com/definition/protocol>. (29.3.2019.)
- [13] CompTIA. What Is a Network Protocol, and How Does It Work? [Online]. 2019. Dostupno na: <https://www.comptia.org/content/guides/what-is-a-network-protocol>. (10.10.2019.)

- [14] Network Encyclopedia. Hypertext Transfer Protocol (HTTP) [Online]. 2015. Dostupno na: <https://networkencyclopedia.com/hypertext-transfer-protocol-http/#history-of-http>. (15.5.2015.)
- [15] ExtraHop. Hypertext Transfer Protocol (HTTP) [Online]. 2020. Dostupno na: <https://www.extrahop.com/resources/protocols/http/>. (15.5.2020.)
- [16] Kaufman J. History of HTTPS Usage [Online]. 2018. Dostupno na: <https://www.jefftk.com/p/history-of-https-usage>. (8.3.2018.)
- [17] CLOUDFLARE. What Is HTTPS? [Online]. 2018. Dostupno na: <https://www.cloudflare.com/learning/ssl/what-is-https/>. (21.6.2018.)
- [18] Tutorialspoint. XML- Overview [Online]. 2014. Dostupno na: https://www.tutorialspoint.com/xml/xml_overview.htm. (5.8.2014.)
- [19] Json. Introducing JSON [Online]. 2019. Dostupno na: <https://www.json.org/json-en.html>. (28.5.2019.)
- [20] WhoIsHostingThis?. Learn JSON: Get Started with Portable Dana Transportation [Online]. 2018. Dostupno na: <https://www.whoishostingthis.com/resources/json-resource/>. (12.12.2018.)
- [21] Web Programiranje. JSON [Online]. 2019. Dostupno na: <https://www.webprogramiranje.org/json/>. (25.1.2019.)
- [22] {C#}STATION. When Was C# Created? A Brief History [Online]. 2020. Dostupno na: <https://csharp-station.com/when-was-c-sharp-created-a-brief-history/>. (18.3.2020.)
- [23] Microsoft. Get started with .NET Framework [Online]. 2019. Dostupno na: <https://docs.microsoft.com/en-us/dotnet/framework/get-started/>. (4.2.2019.)
- [24] BYTESCOUT. MS SQL SERVER HISTORY AND ADVANTAGES [Online]. 2014. Dostupno na: <https://bytescout.com/blog/2014/09/ms-sql-server-history-and-advantages.html>. (1.10.2014.)
- [25] CODE MAGAZINE. History of the VS IDE [Online]. 2007. Dostupno na: <https://www.codemag.com/Article/0710022/History-of-the-VS-IDE>. (27.9.2007.)
- [26] digitalcraft. What is Postman, and Why Should I Use It? [Online]. 2017. Dostupno na: <https://www.digitalcrafts.com/blog/student-blog-what-postman-and-why-use-it>. (25.4.2017.)

8. OZNAKE I KRATICE

API (*engl. Application Programming Interface*), skup određenih pravila i specifikacija koje programeri slijede tako da se mogu služiti uslugama operacijskog sustava.

CORBA (*engl. Common Object Request Broker Architecture*), standard za olakšanu komunikaciju sustava.

CLR (*engl. Common Language Runtime*), virtualna strojna komponenta Microsoftovog .NET okvira.

DCOM (*engl. Distributed Component Object Model*), tehnologija za komunikaciju softverskih komponenti na umreženom računalu.

DTD (*engl. Document Type Definition*), stariji način određivanja pravila strukture XML dokumenta.

ECMA (*engl. European Computer Manufacturers Association*), Europsko udruženje proizvođača računala.

HTTP (*engl. Hyper Text Transfer Protocol*), internetski protokol.

HTTPS (*engl. Hyper Text Transfer Protocol*), internetski protokol

ISO (*engl. International Organization for Standardization*), Međunarodna organizacija za standardizaciju.

JSON (*engl. Java Script Object Notation*), prijenosni format.

RFC (*engl. Request For Comments*), dokument koji opisuje metode, ponašanja, istraživanja ili inovacije primjenjive na Internetu.

REST (*engl. Representational State Transfer*), vrsta Web servisa.

SSL (*engl. Secure Socket Layer*), kriptografski protokol dizajniran za pružanje komunikacijske sigurnosti preko računalne mreže.

SOAP (*engl. Simple Object Access Protocol*), vrsta Web servisa.

TLS (*engl. Transport Layer Security*), kriptografski protokol dizajniran za pružanje komunikacijske sigurnosti preko računalne mreže.

TCP (*engl. Transmission Control Protocol*), vrsta protokola.

URI (*engl. Uniform Resource Identifier*), jedinstveni identifikator resursa.

URL (*engl. Uniform Resource Locator*), putanja do određenog sadržaja na Internetu.

WWW (*engl. World Wide Web*), najkorištenija usluga Interneta koja omogućava pregled hipertekstualnih dokumenata.

W3C (*engl. World Wide Web Consortium*), organizacija koja se bavi standardizacijom tehnologija korištenih na webu.

WSDL (*engl. Web Service Description Language*), jezik baziran na XML-u koji omogućuje opis Web servisa i sučelja.

XML (*engl. Extensible Markup Language*), prijenosni format.

9. SAŽETAK

Naslov : Izrada Web servisa za evidenciju studenata na nastavi

Ovaj rad prikazuje izradu Web servisa koji služi za evidenciju studenata i profesora na nastavi. Web servis je izrađen s ciljem da se primjenjuje na Veleučilištu u Bjelovaru. Unutar Web servisa ovlaštena osoba (profesor) ima mogućnost pribilježiti studente koji se nalaze u tom trenutku na nastavi, laboratorijskim vježbama ili ispitu i unijeti svoje podatke. Cijela aplikacija, back-end izrađena je pomoću programskog jezika : C#-a. Na Backend-u se radi obrada primljenih podataka i vrši se priprema podataka za slanje prema korisničkom sučelju. Za prijenos podataka korišteni su : XML i JSON prijenosni formati. Svi uneseni podatci su pohranjeni unutar Microsoft SQL Server Management Studia. U bazi podataka su smješteni svi podatci vezani za studenta ili profesora. Ti podatci sadrže privatne informacije studenta ili profesora. Također se koristi i Postman koji služi za testiranje HTTP metoda. Kreiranje aplikacijskog programskog sučelja izvedeno je unutar .NET okruženja.

Ključne riječi: Web servis, XML, JSON, SQL Server, Postman, .NET.

10. ABSTRACT

Title: Web service for student attendance evidention

This paper showcases the creation process of a Web service for student and professor class attendance evidention. The Web service was created with usage at the University in Bjelovar in mind. Within Web service, an authorized person (i.e. professor) can record students currently in class, on laboratory exercises or exams, and input their data. The whole application, back-end, is made in the C# programming language. The back-end processes the received data and prepares it for the user interface. For the data transfer, XML and JSON transfer formats were used. All inputted data is stored within the Microsoft SQL Server Management Studio. The database contains all student and professor related data. This data contains the private information of both the students and the professors. Postman is also being used for testing HTTP methods. The application's programming interface was created within the .NET environment.

Key words: Web service, XML, JSON, SQL Server, Postman, .NET.

11. PRILOZI

U prilogu se nalaze slike koje prikazuju kreiranu stranicu uz pomoć React JS-a. Stranica je povezana s Web servisom i tako omogućava jednostavniji unos, izmjenu, prikaz i brisanje podataka vezanih za studenta ili profesora. Stranica podržava sve HTTP metode.

```
cmd Select npm
```

```
Your environment has been set up for using Node.js 12.18.3 (x64) and npm.
```

```
C:\Users\Nikolina>cd C:\Users\Nikolina\source\repos\student-app
```

```
C:\Users\Nikolina\source\repos\student-app> code .
```

```
C:\Users\Nikolina\source\repos\student-app> npm start
```

```
> student-app@0.1.0 start C:\Users\Nikolina\source\repos\student-app
```

```
> react-scripts start
```

```
i @wds: Project is running at http://192.168.1.7/
```

```
i @wds: webpack output is served from
```

```
i @wds: Content not from webpack is served from C:\Users\Nikolina\source\repos\student-app\public
```

```
i @wds: 404s will fallback to /
```

```
Starting the development server...
```

```
Compiled with warnings.
```

VUB Home Professor Student



Welcome!

Student Web Service

ID	JMBAG	Name	Surname	Gender	City	Direction	Year of Collage	Option
1	1233456781	Nikolina	Marinić	Ž	Slavonski Brod	Računarstvo	3	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
2	1234567786	Domagoj	Kovačević	M	Bjelovar	Računarstvo	3	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

Profesor

×

ID:

1

Name:

Krunoslav

Record Update Successfully!

×

Surname:

Husak

Subject:

C#

Date:

2020-01-12

localhost:3000/student

Na web-lokaciji localhost:3000 navodi se sljedeće
Are you sure?

U redu Odustani

ID	JMBAG	Name	Surname	Gender	City	Direction	Year of Collage	Option
1	1233456781	Nikolina	Marinić	Ž	Slavonski Brod	Računarstvo	3	Edit Delete
2	1234567786	Domagoj	Kovačević	M	Bjelovar	Računarstvo	3	Edit Delete

[Add Student](#)

Profesor



Name:

Tomislav

Surname:

Adamović

Subject:

C

Date:

2020-01-01

[Add Professor](#)

Record Inserted Successfully!



[Close](#)

IZJAVA O AUTORSTVU ZAVRŠNOG RADA

Pod punom odgovornošću izjavljujem da sam ovaj rad izradio/la samostalno, poštujući načela akademske čestitosti, pravila struke te pravila i norme standardnog hrvatskog jezika. Rad je moje autorsko djelo i svi su preuzeti citati i parafraze u njemu primjereno označeni.

Mjesto i datum	Ime i prezime studenta/ice	Potpis studenta/ice
U Bjelovaru, <u>22. 10. 2020.</u>	Nikolina Marinčić	Marinčić Nikolina

Prema Odluci Veleučilišta u Bjelovaru, a u skladu sa Zakonom o znanstvenoj djelatnosti i visokom obrazovanju, elektroničke inačice završnih radova studenata Veleučilišta u Bjelovaru bit će pohranjene i javno dostupne u internetskoj bazi Nacionalne i sveučilišne knjižnice u Zagrebu. Ukoliko ste suglasni da tekst Vašeg završnog rada u cijelosti bude javno objavljen, molimo Vas da to potvrdite potpisom.

Suglasnost za objavljivanje elektroničke inačice završnog rada u javno dostupnom nacionalnom repozitoriju

Nikolina Marinio

ime i prezime studenta/ice

Dajem suglasnost da se radi promicanja otvorenog i slobodnog pristupa znanju i informacijama cjeloviti tekst mojeg završnog rada pohrani u repozitorij Nacionalne i sveučilišne knjižnice u Zagrebu i time učini javno dostupnim.

Svojim potpisom potvrđujem istovjetnost tiskane i elektroničke inačice završnog rada.

U Bjelovaru, 22. 10. 2020.

Marinio Nikolina
potpis studenta/ice