# Razvoj web aplikacije za evidenciju međunarodne razmjene studenata u programskom jeziku C# i radnom okružju programskog jezika Javascript

**Hat, Luka**

Sveučilište Josipa Jurja Strossmayera u Osijeku

Ekonomski fakultet u Osijeku

Sveučilišni prijediplomski studij (*Poslovna informatika*)

Luka Hat

# Razvoj web aplikacije za evidenciju međunarodne razmjene studenata u programskom jeziku C# i radnom okružju programskog jezika Javascript

Završni rad

Osijek, 2023.

Sveučilište Josipa Jurja Strossmayera u Osijeku

Ekonomski fakultet u Osijeku

Sveučilišni prijediplomski studij ( *Poslovna informatika*)

Luka Hat

# Razvoj web aplikacije za evidenciju međunarodne razmjene studenata u programskom jeziku C# i radnom okružju programskog jezika Javascript

Završni rad

**Kolegij: Razvoj poslovnih aplikacija**

JMBAG: 0010232901

e-mail: lhat@efos.hr

Mentor: doc. dr. sc. Tomislav Jakopec

Komentor: mag. oec. Saša Mitrović

Osijek,2023.

Josip Juraj Strossmayer University of Osijek

Faculty of Economics and Business in Osijek

Undergraduate Study ( *Business informatics*)

Luka Hat

# Web Application Development for the international student exchange menagement in the C# programming language and framework of Javascript programming language

Final paper

Osijek, 2023.

# IZJAVA
# O AKADEMSKOJ ČESTITOSTI, PRAVU PRIJENOSA INTELEKTUALNOG VLASNIŠTVA, SUGLASNOSTI ZA OBJAVU U INSTITUCIJSKIM REPOZITORIJIMAI ISTOVJETNOSTI DIGITALNE I TISKANE VERZIJE RADA

1. Kojom izjavljujem i svojim potpisom potvrđujem da je završni rad isključivo rezultat osobnoga rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu. Potvrđujem poštivanje nepovredivosti autorstva te točno citiranje radova drugih autora i referiranje na njih.
2. Kojom izjavljujem da je Ekonomski fakultet u Osijeku, bez naknade u vremenski i teritorijalno neograničenom opsegu, nositelj svih prava intelektualnoga vlasništva u odnosu na navedeni rad pod licencom *Creative Commons Imenovanje – Nekomercijalno – Dijeli pod istim uvjetima 3.0 Hrvatska*. 
3. Kojom izjavljujem da sam suglasan/suglasna da se trajno pohrani i objavi moj rad u institucijskom digitalnom repozitoriju Ekonomskoga fakulteta u Osijeku, repozitoriju Sveučilišta Josipa Jurja Strossmayera u Osijeku te javno dostupnom repozitoriju Nacionalne i sveučilišne knjižnice u Zagrebu (u skladu s odredbama Zakona o visokom obrazovanju i znanstvenoj djelatnosti, NN 119/2022).
4. izjavljujem da sam autor/autorica predanog rada i da je sadržaj predane elektroničke datoteke u potpunosti istovjetan sa dovršenom tiskanom verzijom rada predanom u svrhu obrane istog.

**Ime i prezime studenta/studentice:** Luka Hat
**JMBAG:** 0010232901
**OIB:**42804642073
**e-mail za kontakt:**luka.hat58@gmail.com
**Naziv studija:** Prijediplomski studij Poslovna informatika
**Naslov rada:** Razvoj web aplikacije za evidenciju međunarodne razmjene studenata u programskom jeziku C# i radnom okružju programskog jezika Javascript
**Mentor/mentorica rada:** doc. dr. sc. Tomislav Jakopec
**Komentor/komentorica rada:** mag. oec. Saša Mitrović

U Osijeku, 13.09.2023. godine

Potpis Luka Hat

# Razvoj web aplikacije za evidenciju međunarodne razmjene studenata u programskom jeziku C# i radnom okružju programskog jezika Javascript

## SAŽETAK

Povećanje u mobilnosti studenata uzrokuje povećane količine informacija o njima te zahtijeva stvaranje centralnog mjesta za skladištenje istih podataka te manipulaciju njima. Iz tog razloga kroz ovaj rad prikazana je izrada web aplikacije za evidenciju međunarodne razmjene studenata.

Kroz rad se prvo pruža teoretska osnova troslojne arhitekture izrade aplikacije, agilnog pristupa razvoju aplikacija te samim tehnologijama korištenim u pozadinskom i klijentskom sloju aplikacije te se kasnije pruža uvid u praktičnu primjenu odabranih tehnologija kroz slike. Aplikacija je također javno dostupna putem web preglednika kao i kod za izradu iste unutar Github platforme.

Aplikacije je izrađena u zasebnim slojevima te je podijeljena na pozadinski sloj koji je napravljen koristeći C# programski jezik unutar Microsoft Visual Studio okruženja za razvoj koristeći .NET radno okružje te klijentski sloj koji je napravljen pomoću Javascript programskog jezika u React radnom okružju unutar Microsoft Visual Studio Code okruženja za razvoj.

Za rad je također bio korišten Git sustav za verzioniranje koda kao i Github platforma za javno objavljivanje istog. Korištena je Sass ekstenzija CSS-a te Gulp za obavljanje repetitivnih zadataka. Razvoj aplikacije grafički i vremenski je praćen kroz Jira program.

Na kraju rada napravljena je usporedba korištenih tehnologija sa aktualnima kao i prijedlozi za unaprjeđenje i daljnji razvoj aplikacije.

Ključne riječi: web aplikacija, pozadinski sloj, klijentski sloj, C#, React

# Web Application Development for the international student exchange menagement in the C# programming language and framework of Javascript programming language

## ABSTRACT

An increase in student mobility is causing an increased amount of information about them and demands a creation of a central space for storing that data and manipulating it. For that reason, this paper shows the creation of a web app for menagment of international exchange students.

Through this paper first there is the theoretical background of three tier architecture of app development, agile aproach to app development and technologies used in backend and frontend part of the app and later it's shown how these technologies were used using pictures.The app is also publicly available through the use of a web browser as is the code for making it on Github.

The app is built in separate layers and it is divided into a backend that uses C# programming language inside Microsoft Visual Studio integrated development enviroment using .NET framework and frontend using the Javascript programming language and its React framework inside the Microsoft Visual Studio Code integrated development enviroment.

For making this paper Git was used as a system for code versioning as well as the Github platform for publicly sharing it. Sass extension of CSS was also used as well as Gulp for repetitive tasks. App development was graphicly and timely tracked through Jira.

At the end of the paper there is a comparison of used technologies with trending ones as well as ideas for improving and further development of the app.

Key words: web application, backend, frontend, C#, React

**SADRŽAJ**

# 1. Uvod

Mobilnost studenata je veća nego ikad, pod okriljem programa poput Erasmus+ studentima diljem Europe omogućeno je studiranje u svim zemljama Europe te se samim time stvara potreba za stvaranjem sustava koji bi evidentirao studente koji su na razmjeni na određenom fakultetu. Ovaj rad će prikazati razvoj web aplikacije koja će korisnicima omogućiti da kroz intuitivno sučelje lakše evidentira studente koji su trenutno na razmjeni na njihovom fakultetu. Aplikacija će biti razvijena koristeći troslojnu arhitekturu te agilni pristup razvoju aplikacija. Za razvoj aplikacije biti će korišten C# programski jezik u pozadinskom sloju te React radno okružje programskog jezika Javascript na klijentskom sloju. Ove tehnologije su odabrane jer omogućavaju jednostavan razvoj intuitivnog korisničkog sučelja i jednostavno povezivanje klijentskog i pozadinskog sloja aplikacije.

## 2. Troslojna arhitektura izrade web aplikacija

Višeslojna arhitektura izrade web aplikacija znači da aplikaciju dijelimo u više slojeva koji imaju svoje pojedine uloge. Postoje različiti pristupi poput dvoslojne, troslojne i n-slojne arhitekture izrade.

Troslojnu arhitekturu prema (IBM) možemo definirati kao „dobro ustanovljenu arhitekturu razvoja softver-a koja organizira aplikacije u 3 logičke i fizičke računalne slojeve: prezentacijski sloj, ili korisničko sučelje; aplikacijski sloj, gdje se procesuiraju podatci; i podatkovni sloj gdje su podatci povezani sa aplikacijom skladišteni i gdje se s njima upravlja“

Za izradu web aplikacije prikazane u ovom radu korištena je troslojna arhitektura izrade koja se naziva MVC odnosno Model-View-Controller arhitektura razvoja. Ona obuhvaća tri sloja:

Prezentacijski sloj („View“) koji ima za zadatak korisniku prikazati sadržaj korisniku kroz grafičko sučelje, kako bi ovaj sloj funkcionirao potrebna je interakcija sa aplikacijskim slojem.

Aplikacijski sloj („Controller“) koji predstavlja srednji sloj koji za zadatak ima primanje zahtjeva za prikaz od prezentacijskoj sloja, dohvaćanje podataka iz podatkovnog sloja te zatim vraćanje podataka prezentacijskom sloju koji ih zatim prikaže krajnjem korisniku.

Podatkovni sloj („Model“) koji predstavlja najniži sloj kojem je glavni zadatak skladištenje aplikacijskih podataka te njihov povrat na zahtjev aplikacijskog sloja.

Neke od prednosti izrade aplikacije prateći principe troslojne arhitekture izrade web aplikacija prema (IBM) su:

„Brži razvoj aplikacije: Zato što se svaki sloj aplikacije se može razvijati istovremeno od strane različitih timova, organizacija može prije pustiti aplikaciju na tržište te programeri mogu koristiti najnovije i najbolje programske jezike i alate za svaki sloj

Poboljšana skalabilnost: Svaki sloj može biti skaliran neovisno o drugima po potrebi

Poboljšana pouzdanost: Prekid rada jednog sloja ima manju vjerojatnost utjecaja na dostupnost ili funkcionalnosti drugih slojeva

Poboljšana sigurnost: Zato što prezentacijski sloj i podatkovni sloj ne komuniciraju direktno, dobro dizajniran aplikacijski sloj može raditi kao interni firewall, sprječavajući SQL injekcije i ostale maliciozne mane.“ ( https://www.ibm.com/topics/three-tier-architecture)

Za implementaciju navedene odabrane arhitekture izrade bit će korišteni C# programski jezik u .NET radnom okružju kroz implementaciju Entity Framework radnog okružja u podatkovnom sloju odnosno kao „model“ dio aplikacije, C# programski jezik u .NET razvojnom okružju za aplikacijski sloj aplikacije odnosno kao „controller“ te Javascript programski jezik kroz React razvojno okružje.

Prema (IBM) „u razvoju web aplikacija slojevi imaju različita imena no obavljaju slične funkcije:

- Web server je prezentacijski sloj te pruža korisničko sučelje
- Aplikacijski server shodno pripada srednjem sloju, sadrži poslovnu logiku za procesuiranje korisničkog unosa
- Server baza podataka su podatci ili podatkovni sloj web aplikacije“

## 3. Agilni pristup izradi web aplikacija

Kako bismo razmatrali agilni pristup izradi web aplikacija važno je razumjeti semantiku povezanih pojmova. Metodologija predstavlja tijelo metoda i pravila odnosno određene procedure ili setove procedura. Prema (Merriam Webster)

" metodologija : tijelo metoda, pravila, i postulata korištenih od discipline : specifična procedura ili skup procedura

: analiza principa ili procedura upita u specifičnom polju“

A prema (Rakesh Patel)

„Metode razvoja softver-a referiraju se na proces ili slijed procesa korištenih u razvoju softver-a. Namjena im je objasniti kakav bi ciklus razvoja softver-a trebao biti u vašem projektu. To je kolekcija uloga, pravila, i najbolje prakse za razvoj softver-a. Definitivno ne sadrži korištenje tehničkih elemenata ali uključuje pažljivo planiranje za životni ciklus razvoja sofver-a.“

Za razvoj web aplikacija na raspolaganju imamo puno mogućnosti među kojima se nalazi i odabrana – agilna metodologija. Odabir metodologije za izradu aplikacije određuje tijek razvoja iste aplikacije. Različite metodologije pružaju različite prednosti te je potrebno razmotriti zahtjeve aplikacije koju razvijamo kako bi odabrali prikladnu metodologiju koja bi nam omogućila najbolji tijek razvoja aplikacije. Prema izvoru (Plavno) agilna metodologija je najbolja za razvoj softverskih rješenja te je ujedno i jedna od najpopularnijih.

Agilna metodologija ima i svoj proglas prema (Beck, K., Beedle, M., Bennekum. et.al.) te glasi:

“ Proglas o metodi agilnog razvoja softvera

Tražimo bolje načine razvoja softvera razvijajući softver i pomažući drugima pri njegovom razvoju. Takvim radom smo naučili da više cijenimo:Ljude i njihove međusobne odnose nego procese i oruđa Upotrebljiv softver nego iscrpnu dokumentaciju Suradnju s naručiteljem nego pregovaranje oko ugovora Reagiranje na promjenu nego ustrajanje na planu. Drugim riječima, iako cijenimo vrijednosti na desnoj strani, više vjerujemo u one na lijevoj“

Agilna metodologija bazira se na iterativnom razvoju. Primjenom ove metodologije rad se dijeli u zasebne cjeline koje definiramo kao sprintove koji najčešće traju 2 ili više tjedana. Ova metodologija zahtjeva povratne informacije na temelju kojih se zahtjevi stavljaju u takozvani "backlog" iz kojeg se odabiru zadatci odnosno dijelovi softver-a koji se razvijaju tijekom sprinta. Nakon sprinta dobivaju se povratne informacije te se ovaj ciklus nastavlja kroz cijeli životni ciklus aplikacije.

# 4. Zakonska osnova međunarodne razmjene studenata

Web aplikacija koja će biti razvijena za svrhe ovog rada treba omogućavati upravljanje podatcima o studentima na razmjeni. Razmjena studenata ima brojne prednosti za studenta te se broj studenata na razmjeni sve više povećava. Iz tog razloga javlja se potreba za razvijanje web aplikacije koja olakšava proces upravljanja podatcima o razmjeni studenata. Aplikacija će omogućiti pregled podataka o studentu (ime, prezime i jmbag) kao i pregled podataka o razmjeni (student na razmjeni, država iz koje student dolazi, sveučilište s kojeg student dolazi, datum početka razmjene kao i datum završetka iste).

# 5. Pozadinski sloj kroz korištenje C# programskog jezika

Pozadinski sloj aplikacije će biti izrađen korištenjem C# programskog jezika. C# je objektno orijentirani programski jezik koji će biti korišten na .NET platformi. Kroz .NET platformu omogućeno je stvaranje aplikacija za Windows sučelje, Linux sučelje, Mac sučelje, kao i izgradnja mobilnih i web aplikacija. C# će u slučaju ove web aplikacije za međunarodnu razmjenu studenata služiti kao komunikacija između baze podataka i vidljivog korisničkog sučelja u vidu "controllera" te će služiti za upravljanje bazom podataka tako da omogućuje brisanje, pregled, uređivanje te stvaranje podataka. C# će također biti odgovoran za skladištenje podataka. Kroz .NET platformu biti će stvoren API s kojim će komunicirati "frontend" sloj aplikacije. API odnosno „Application Programming Interface" predstavlja kod pomoću kojeg 2 različita programa mogu međusobno komunicirati. Pozadinski sloj aplikacije biti će objavljen putem Azure-a. Azure nam omogućava objavljivanje te usluge "hostanja" baze podataka kao i web stranica. Unutar .NET platforme biti će korišten Entity Framework za olakšano upravljanje bazom podataka. Pozadinski sloj aplikacije biti će postavljen kao „controller" te ujedno i „model" gledano kroz MVC arhitekturu razvoja aplikacija.

# 6. Klijentski sloj kroz korištenje React radnog okruženja

Klijentski sloj aplikacije biti će izrađen na webu. Web sučelja se grade pomoću HTML-a. HTML („Hypertext markup language“) koristi se za definiranje „gradivnih blokova“ web stranice. Korištenjem HTML-a definira se struktura stranice. HTML omogućava izgradnju osnovne strukture no za dodavanje stiliziranja te izgradnje dobrog korisničkog iskustva potrebno je koristiti CSS („Cascading Style Sheets“). CSS nam omogućava da korištenjem raznih selektora dohvatimo HTML elemente te omogućuje manipulaciju istima u vidu promjene zadanog izgleda i pozicije elemenata. Korištenjem ove dvije tehnologije možemo dobiti statičnu web stranicu. No kako bi izgradili web aplikaciju sa dinamičnim elementima i podatcima te omogućili napredne animacije potrebno je koristiti programski jezik Javascipt. Javascript je objektno orijentirani programski jezik koji se najčešće koristi za manipulaciju HTML elemetima te upravljanjem podatcima kroz web sučelje. Korištenjem ovih tehnologija možemo izgraditi napredne web aplikacije sa dobrim korisničkim sučeljem. U doba nastanka ovih tehnologija web stranice i web aplikacije nisu bile kompleksne koliko su u današnje vrijeme te se sa dodavanjem funkcionalnosti web aplikacijama i dodavanjem ogromne količine podataka ubrzo javio problem prevelike količine linija koda. Kao rješenje tog problema javlja se korištenje ranih okružja programskog jezika Javascript, jedno od tih okružja je i React . React nam omogućuje izgradnju web aplikacija pomoću komponenti korištenjem JSX-a. JSX predstavlja sintaktičku ekstenziju Javascripta stvaranjem šablona koje izgledaju kao HTML gradivni blokovi no omogućuju potpuno korištenje Javascript mogućnosti. Ovaj pristup uvelike smanjuje broj potrebnih linija koda jer direktno povezuje elemente HTML-a sa njihovom funkcionalnosti. Također olakšava uklanjanje grešaka u kodu jer je jasnije vidljivo u kojoj komponenti se greška javlja. Što se tiče izgleda stranice omogućeno nam je korištenje CSS-a no korištenje istog povećanjem veličine i kompleksnosti aplikacije također dolazi do prevelikog broja linija koda stoga se javljaju takozvani preprocessori koji omogućavaju sintaktički čitljiviji CSS. Jedan od tih preprocessora je i SASS („Syntactically awesome style sheets“) koji omogućava kreiranje varijabli te predložaka za uklanjanje potrebe dupliciranja koda. Nakon što napravimo sass dokument on će se kompajlirati u običan css dokument koji je čitljiv browseru. Sam postupak kompajliranja obavit će se automatski korištenjem Gulp-a. Gulp predstavlja set alata koji automatiziraju poslove poput minificiranja koda radi ubrzanja, pretvaranje sass dokumenata u css dokumente te prevođenje modernog javascripta u onaj dostupan svim browserima.

# 7. Praktična izrada aplikacije

Za početak praktične izrade aplikacije potrebno je odabrati putem kojeg IDE –a („Integrated development enviroment“) ćemo pisati kod. Za pozadinski sloj aplikacije odabran je Microsoft Visual Studio koji nam omogućuje napredne funkcije poput testiranja koda, integriranog procesa „debugginga“ odnosno uklanjanja grešaka. Također nam je ubrzan proces pisanja koda za pozadinski sloj na način da imamo predloške koda te predloške strukture koda za određenu potrebu, u slučaju ove aplikacije korišten je predložak za kreiranje web api-a. Za klijentski sloj korišten je Microsoft Visual Studio Code koji također ima integriranog asistenta koji predlaže linije koda na temelju napisanog teksta te ima mogućnost dodavanja brojnih ekstenzija koji ubrzavaju i olakšavaju proces kreiranja aplikacije. Za objavljivanje aplikacije odabran je Azure koji omogućuje lagano intuitivno upravljanje resursima koje smo objavili te kreiranje centralnog mjesta za sve resurse koji su nam potrebni u projektu. Kako bi implementirali agilni način izrade aplikacije korišten je jira software. Konačno za verzioniranje koda korišten je Git u kombinaciji sa Githubom kako bi kod aplikacije bio javno dostupan.

## 7.1.Izrada baze podataka

Baza podataka aplikacije biti će izrađena kroz korištenje „Entity Framework“ paketa koji će kroz .NET platformu biti dodan u projekt. Omogućiti će nam stvaranje baze podatka bez potrebe za strukturnim upitnim jezicima kroz korištenje LINQ-a („Language Integrated Query“) koji simulira korištenje upitnog jezika kroz sličnu sintaksu.

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace Test_API.Models
{
    21 references
    public class Student
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        9 references
        public int Id { get; set; }
        4 references
        public string Ime { get; set; }
        3 references
        public string Prezime { get; set; }
        3 references
        public string JMBAG { get; set; }
    }
}
```

*Slika 1: Klasa Student*

```
using System.ComponentModel.DataAnnotations.Schema;

namespace Test_API.Models
{
    14 references
    public class Razmjena
    {
        4 references
        public int Id { get; set; }
        [ForeignKey("Student")]
        0 references
        public int StudentId { get; set; }
        0 references
        public Student Student { get; set; }

        0 references
        public string Drzava { get; set; }
        0 references
        public string Sveuciliste { get; set; }
        0 references
        public DateTime DatumOd { get; set; }
        0 references
        public DateTime DatumDo { get; set; }
    }
}
```

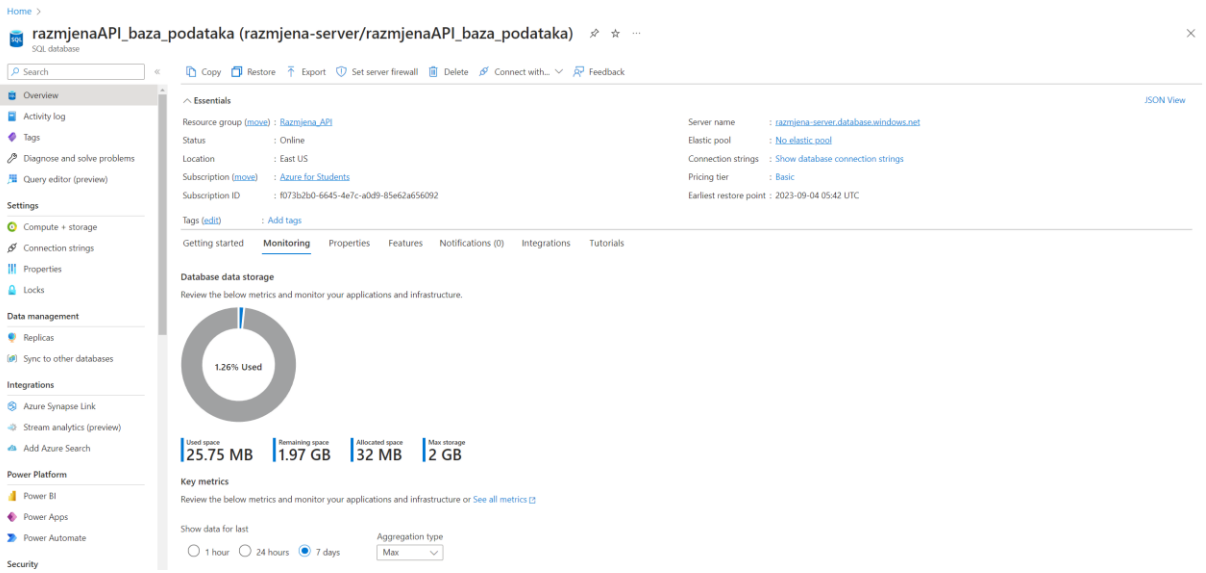
*Slika 2: Klasa Razmjena*

Na slici 1 i slici 2 prikazan je kod koji je korišten za kreiranje klase „Student“ te klase „Razmjena“. Na taj način je stvoren model prema kojem se podatci unose u bazu podataka. Varijable unutar ovih klasa predstavljaju nazive stupaca u tablicama unutar baze podataka . Također je vidljivo spajanje ovih tablica putem stranog ključa na način da se „id“ unutar klase “Student“ predaje klasi „Razmjena“ kako bi bilo moguće pregledati koje studente razmjena sadrži. Bazu podataka zatim smo koristeći .NET platformu ispunili početnim podatcima što je vidljivo u slici 3: „Dodavanje početnih vrijednosti u tablicu 'Students'“.

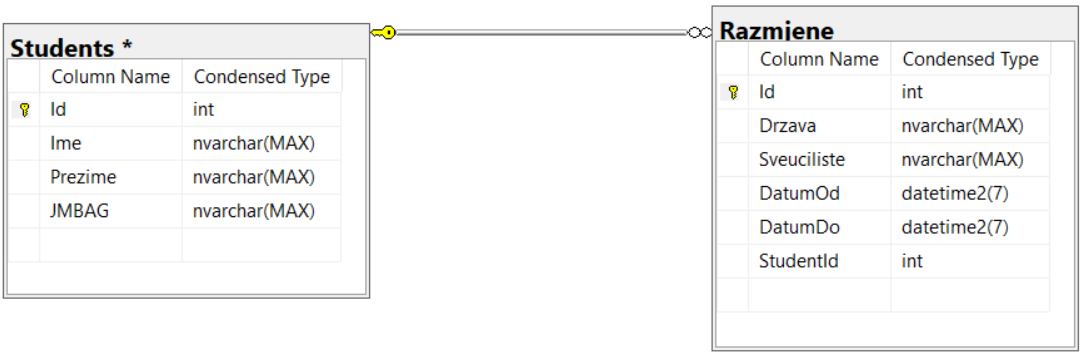
```
using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

#pragma warning disable CA1814 // Prefer jagged arrays over multidimensional

namespace Test_API.Migrations
{
    /// <inheritdoc />
    1 reference
    public partial class SeedStudentTable : Migration
    {
        /// <inheritdoc />
        0 references
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.InsertData(
                table: "Students",
                columns: new[] { "Id", "Ime", "JMBAG", "Prezime" },
                values: new object[,]
                {
                    { 1, "Tea", "0010232001", "Babić" },
                    { 2, "Luka", "0010232901", "Hat" },
                    { 3, "Tomislav", "0010232483", "Kvesić" }
                });
        }

        /// <inheritdoc />
        0 references
        protected override void Down(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.DeleteData(
                table: "Students",
                keyColumn: "Id",
                keyValue: 1);

            migrationBuilder.DeleteData(
                table: "Students",
                keyColumn: "Id",
                keyValue: 2);
```

*Slika 3: Dodavanje početnih vrijednosti u tablicu "Students"*

Baza podataka putem „Microsoft Azure“ platforme dodana je na Internet te omogućuje praćenje dinamičnih promjena nad podatcima unutar tablica kao i pregled stanja same baze.




*Slika 4: Baza podataka unutar Microsoft Azure platforme*




*Slika 5: ERA dijagram baze podataka*

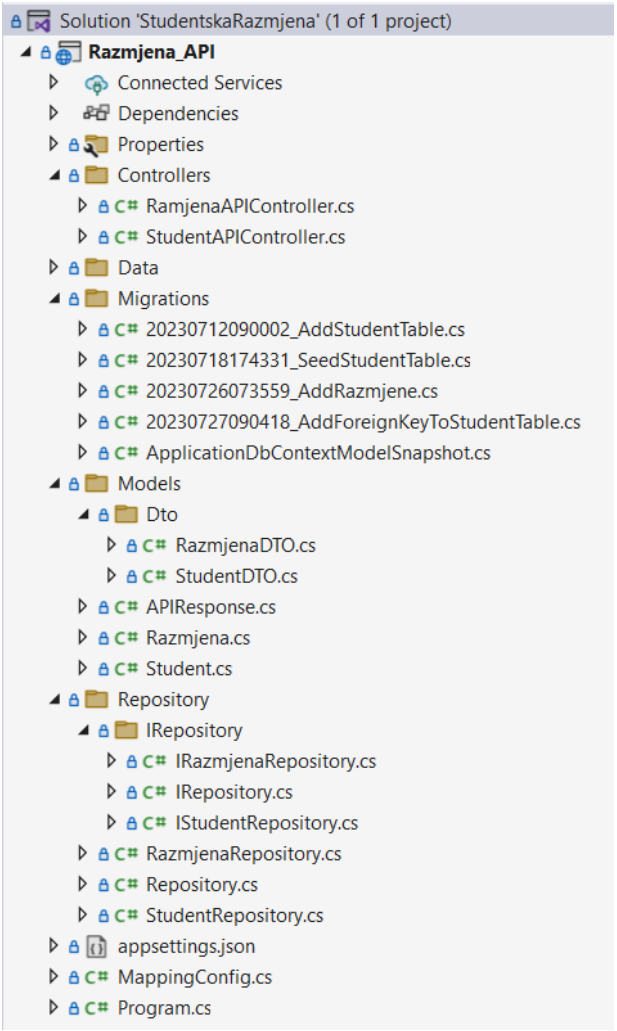
Kao što je vidljivo iz slike 4 Azure platforma nam omogućava i olakšano upravljanje naprednim značajkama poput sinkronizacije s ostalim bazama podataka, postavljanje vatrozida, postavljanje ograničenja pristupa, kao i određivanje „connection string-ova“ čijom integracijom u kod se spajamo sa pojedinom bazom podataka.

Na slici 5 imamo prikazan ERA dijagram baze podataka koja sadrži 2 tablice: „Student“ tablicu te „Razmjena“ tablicu. Vidljiva je povezanost tablica, konkretno u ovom slučaju

tablica „Razmjene“ može sadržavati jednog ili više studenta. Tablice se su povezane koristeći strani ključ. Također su vidljivi tipovi podataka koji se koriste u kreiranju podataka unutar tablice.

### 7.2.Izrada pozadinskog sloja

Na početku izrade pozadinskog sloja aplikacije unutar Microsoft Visual Studio programa ponuđene su razne šablone za olakšan početak rada. U slučaju ove aplikacije odabrana je šablona za kreiranje „ASP.NET Core Web API“ aplikacije. Ovim načinom dobivamo početne mape kao i dokumente koji će nam biti potrebni za stvaranje API aplikacije. Struktura završene aplikacije vidljiva je na slici 5.




*Slika 6: Struktura pozadinskog sloja*

Unutar pozadinskog sloja aplikacije izrađena je funkcionalnost za brisanje, unos, pregled te uređivanje podataka iz baze podataka koja je prethodno stvorena. API koji je stvoren sadrži 2

controller-a, jedan za studente te jedan za razmjene. Unutar tih controller-a definirane su metode za interakciju sa podatcima unutar  tablice koja sadrži studente te unutar tablice koja sadrži razmjene.

```
[HttpGet]
[ProducesResponseType(StatusCodes.Status200OK)]
0 references
public async Task<ActionResult<APIResponse>> GetStudents()
{
    try
    {
        IEnumerable<Student> studentList = await _dbStudent.GetAllAsync();
        _response.Result = _mapper.Map<List<StudentDTO>>(studentList);
        _response.StatusCode = HttpStatusCode.OK;
        return Ok(_response);
    }
    catch (Exception ex)
    {
        _response.IsSuccess = false;
        _response.ErrorMessages = new List<string>() { ex.ToString() };
    }
    return _response;
}
```

*Slika 7: "Get" metoda unutar Student Controller-a*

„HTTP definira  set metoda zahtjeva koji indiciraju na željenu akciju koja će biti obavljena na danom resursu.“( Mozilla Developer Network, n.d.)

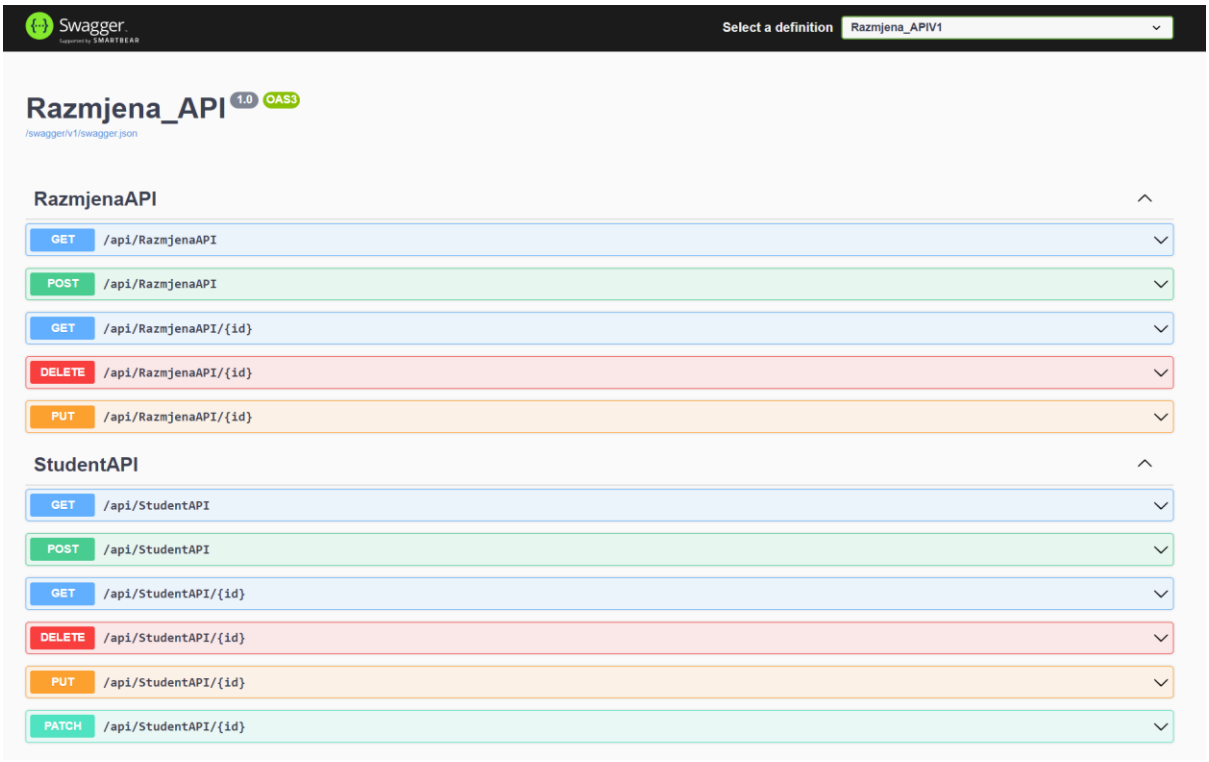
U ovom slučaju Get metoda koristi se za dohvaćanje svih podataka. Također je stvorena funkcionalnost za metodu „Post“ koja omogućuje kreiranje, metodu „Delete“ koja omogućuje brisanje te metodu „Put“ koja omogućuje ažuriranje podataka.  Na slici 6 vidljiva je implementacija povrata statusnih kodova, u ovom slučaju statusni kod 200 predstavlja uspješno izvršenje zahtjeva te uspješan povrat željenih podataka.  Kod za funkcionalnost stavljen je u „try catch block“, ukoliko je sve uspješno izvršiti će se samo dio koda unutar „try“ dijela unutar kojeg je sama funkcionalnost, odnosno pozivanje metode za asinkrono dohvaćanje svih podataka iz tablice „Students“, povrat statusnog koda 200 te konačno povrat podataka. Ukoliko postoji greška prilikom pozivanja ove metode izvršava se „catch“ dio koda, odnosno povratno pratimo kako zahtjev nije uspješan te vraćamo tekst greške koja se dogodila. Ovakav način razvoja aplikacije uvelike olakšava testiranje kao i proces otklanjanja grešaka. Funkcija „GetAllAsync“ koja omogućuje asinkrono dohvaćanje definirana je unutar baznog repozitorija čije metode nasljeđuje student repozitorij te je vidljiva na slici 7.

```
3 references
public async Task<List<T>> GetAllAsync(Expression<Func<T, bool>>? filter = null)
{
    IQueryable<T> query = dbSet;
    if (filter != null)
    {
        query = query.Where(filter);
    }
    return await query.ToListAsync();
}
```

*Slika 8: GetAllAsync metoda*

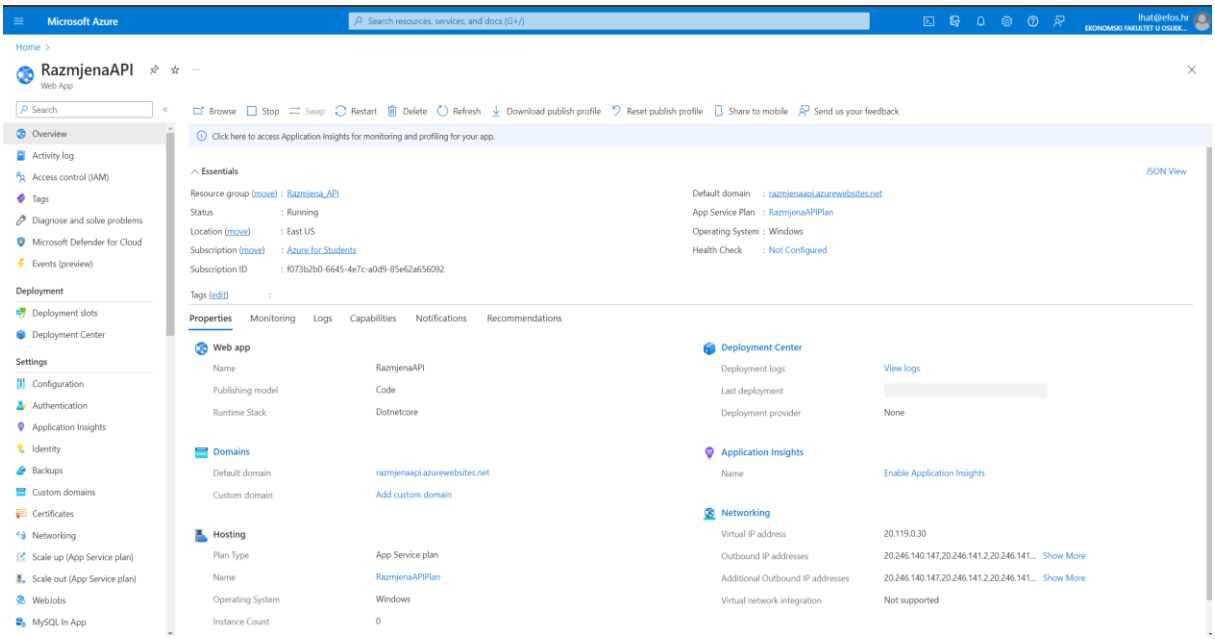
Ova metoda je asinkrona što znači da se ne čeka izvršavanje ove metode prije nego što se nastavi program već se ona poziva te u pozadini čeka izvršenje te se nakon toga prikazuju vraća rezultat iste. Ključna riječ „await“ predstavlja koji dio koda se čeka kako bi se vratio rezultat. Na ovaj način ne moramo čekati izvršenje svih metoda te povrat svih podataka kako bi se ostatak koda primijenio već se duži zadatci poput ovoga odvijaju u pozadini te prikazuju korisniku kada su spremni.Ova metoda ima generični tip „T“ što znači da može raditi s bilo kojim tipom entiteta kako bi mogla biti korištena unutar repozitorija za studente kao i repozitorija za razmjene što smanjuje repeticiju. Stvara se upit prema bazi podataka te metoda provjerava je li proslijeđen filter za upit, ukoliko je primjenjuje se na upit pomoću „Where“ metode. Na kraju dobivamo rezultat upita koji je možda prošao kroz filter te se vraća kao lista. Koristeći ove principe također su kreirane metode za brisanje pod nazivom „RemoveAsync“, metoda za spremanje promjena „SaveAsync“ te metoda za ažuriranje podataka pod nazivom „UpdateAsync“.

Kada je aplikacija stvorena, funkcionalnost iste može se isprobati na više načina, u slučaju ove aplikacije korišten je „Swagger“. To je alat koji nam omogućuje da definiramo strukturu našeg API za strojeve. Također nam omogućuje pregled metoda za koje smo napravili funkcionalnost te korištenje istih na brz i pregledan način.




*Slika 9: Swagger alat*

Kako bi mogli dohvatiti ovaj pozadinski sloj u klijentskom sloju ovaj API mora biti javno dostupan. API je objavljen na Azure-u koji pruža neke napredne funkcije za olakšano upravljanje ovom aplikacijom te obavlja usluge poslužitelja odnosno „hosting“.




*Slika 10: Pozadinski sloj na Azure-u*

Koristeći Azure također smo omogućili CORS odnosno „Cross-Origin Resource Sharing“ koji omogućava Javascript kodu našeg klijentskog sloja da komunicira sa našim pozadinskim slojem.

### 7.3.Izrada klijentskog sloja

Za izradu klijentskog sloja aplikacije u radnom okružju React korišten je Visual Studio Code. Unutar Visual Studio Code-a omogućen je pristup „Teminal-u“ pomoću kojega se dolazi do direktorija unutar kojeg se projekt želi stvoriti te se naredbom *npx create-react-app {naziv aplikacije}* stvara aplikacija sa zadanim dokumentima koji ubrzavaju proces stvaranja projekta. Za početak rada potrebno je spojiti Javascript sa HTML elementom unutar kojeg će se prikazati.

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
import { BrowserRouter } from "react-router-dom";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>
);
```

*Slika 11: Inicijalno spajanje s HTML-om*

Ovo spajanje napravljeno je unutar indeks.js dokumenta te sadrži App komponentu koja predstavlja temeljnu komponentu ove aplikacije.

```jsx
import React from "react";
import { Route, Routes } from "react-router-dom";
import { Home } from "./Home";
import { Students } from "./Students";
import { Razmjene } from "./Razmjene";
import { ErrorPage } from "./ErrorPage";
import { Header } from "./Header";
import { DeleteRazmjenaForm } from "./DeleteRazmjenaForm";
import { DeleteStudentForm } from "./DeleteStudentForm";
import { EditStudentForm } from "./EditStudentForm";
import { EditRazmjenaForm } from "./EditRazmjenaForm";
import { CreateRazmjenaForm } from "./CreateRazmjenaForm";
import { CreateStudentForm } from "./CreateStudentForm";
//styles
import "./dist/css/styles.min.css";

const App = () => {
  return (
    <>
      <Header />
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/studenti" element={<Students />} />
        <Route path="/razmjene" element={<Razmjene />} />
        <Route path="/edit/razmjena" element={<EditRazmjenaForm />} />
        <Route path="/edit/student" element={<EditStudentForm />} />
        <Route path="/create/student" element={<CreateStudentForm />} />
        <Route path="/create/razmjena" element={<CreateRazmjenaForm />} />
        <Route path="/delete/student" element={<DeleteStudentForm />} />
        <Route path="/delete/razmjena" element={<DeleteRazmjenaForm />} />
        <Route path="*" element={<ErrorPage />}></Route>
      </Routes>
    </>
  );
};

export default App;
```

*Slika 12: App komponenta*

Unutar App komponente uvezli smo sve ostale komponente te napravili osnovne rute aplikacije. Kako bi rute unutar aplikacije bile moguće pomoću terminala u projekt je dodan „React Router“, radno okruženje Javascript jezika koje omogućuje funkciju usmjeravanja. Također je iz slike 11 vidljivo dodavanje „Header“ komponente koja se prikazuje neovisno o ruti na kojoj se nalazimo te prikazuje korisniku poveznice na temeljne komponente.

```jsx
import React from "react";
import { Link } from "react-router-dom";

export const Header = () => {
  return (
    <>
      <nav>
        <ul className="header-nav">
          <li>
            <Link to="/">
              <h2>Evidencija Razmjena</h2>
            </Link>
          </li>
          <li className="main-nav">
            <Link to="/studenti">Studenti</Link>
            <Link to="/razmjene">Razmjene</Link>
          </li>
        </ul>
      </nav>
    </>
  );
};
```

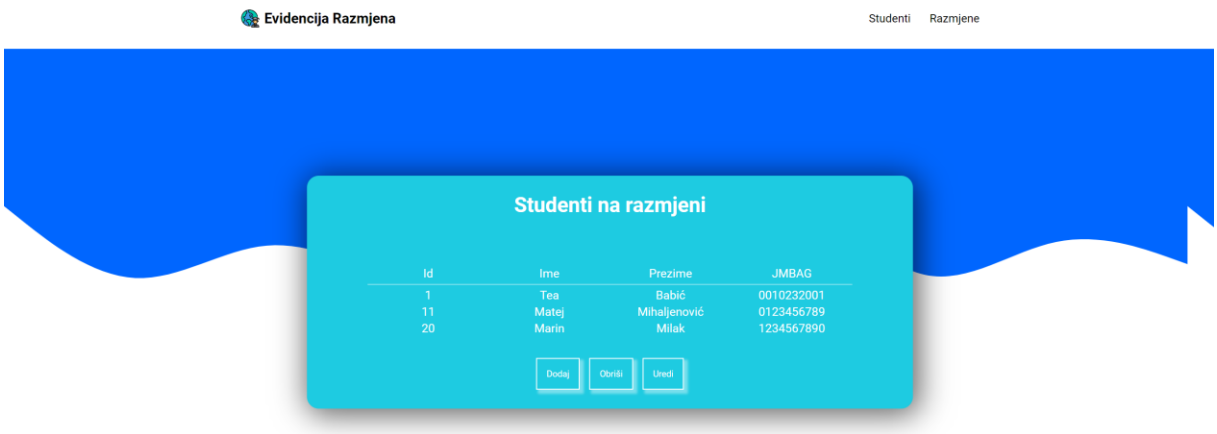
*Slika 13: Header komponenta*

Iz slike 12 vidljivo je kako je svaka komponenta zapravo svojevrsna funkcija koja kao vrijednost koju vraća ima JSX elemente koji predstavljaju nadogradnju na HTML te olakšanu integraciju Javascript programskog jezika u osnovni HTML kod. Kako bi funkcionalnost definirana u pozadinskom sloju bila dostupna potrebno je koristiti poveznicu na kreirani Web API. Na slici 13 tu poveznicu vidimo kao vrijednost varijable getStudentsUrl. Budući da su studenti promjenjivi (može ih se brisati, dodavati i sl.) potrebno je koristiti useState „Hook“. React dolazi sa ugrađenim „hook-ovima“ koji olakšavaju rad sa komponentama, no također je moguće stvarati vlastite „hook-ove“. Najprije studentima dodijelimo vrijednost praznog niza kojeg kasnije punimo podatcima. Kao što je vidljivo na slici 13, definiramo metodu „fetchStudents“ koja je asinkrona te sadržava „try-catch blok“. Ukoliko nema grešaka prilikom izvršenja metode najprije spremamo podatke koje smo dobili prateći poveznicu koristeći „axios.get“ metodu u varijablu response1 dolazimo do podataka koji nam trebaju iz odgovora unutar varijable studentData nakon čega postavljamo vrijednost te varijable varijabli „students“ koristeći „setStudent“ opciju „useState hook-a“. Axios predstavlja klijent

koji olakšava korištenje HTTP metoda te skraćuje potrebnu količinu koda za obavljanje istih. Zatim unutar „useEffect hook-a“ pozivamo metodu „fetchStudents“ koja se poziva kada se stranica inicijalno učita. Naposljetku unutar JSX-a pozivamo Javascript metodu „map“ kojoj predajemo kao parametar studenta te za svakog studenta destrukturiramo vrijednosti koje sadrži te ih odvajamo za prikaz u željenim JSX elementima. Također unutar ove komponente imamo definirane poveznice na komponente za brisanje, uređivanje te stvaranje studenata.

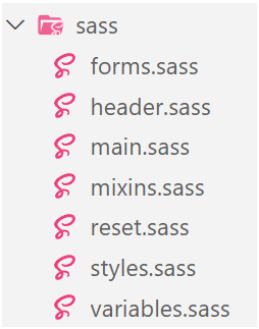
```
const getStudentUrl = "https://razmjenaapi.azurewebsites.net/api/StudentAPI";

export const Students = () => {
  const [students, setStudents] = useState([]);
  const fetchStudents = async () => {
    try {
      const response1 = await axios(getStudentUrl);
      const studentData = response1.data.result;
      setStudents(studentData);
    } catch (error) {
      console.log(error.response1);
    }
  };
  useEffect(() => {
    fetchStudents();
  }, []);
  return (
    <>
      <main>
        <h1>Studenti na razmjeni</h1>
        <ul className="student-list">
          <li className="student row-titles">
            <span>Id</span>
            <span>Ime</span>
            <span>Prezime</span>
            <span>JMBAG</span>
          </li>
          {students.map((student) => {
            const { id, ime, prezime, jmbag } = student;
            return (
              <li key={id} className="student">
                <h3>{id}</h3>
                <h3>{ime}</h3>
                <h3>{prezime}</h3>
                <h3>{jmbag}</h3>
              </li>
            );
          })}
          <li className="edit-create-buttons">
            <Link to="/create/student">
              <button>Dodaj</button>
            </Link>
            <Link to="/delete/student">
              <button>Obriši</button>
            </Link>
            <Link to={"/edit/student"}>
              <button>Uredi</button>
            </Link>
          </li>
        </ul>
      </main>
    </>
  );
};
```

*Slika 14: Students komponenta*




*Slika 15: Izgled Students komponente*

Izgled JSX elemenata unutar Students komponente definiran je koristeći Sass. Sass je korišten na način da su u zasebnim dokumentima definirani stilovi za zasebne komponente te varijable i mixin-ovi.



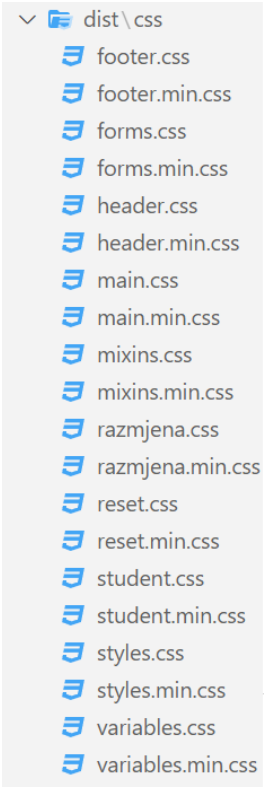

*Slika 16: Struktura Sass dokumenata*

Sass je po sintaksi drugačiji od CSS-a na način da se elementi mogu hijerarhijski selektirati te njihov odnos odražavamo indentacijom.

```
small
    @include displayFlex
    flex-direction: row
    gap: 20%
    height: 100%
    a
        color: $white
        cursor: pointer
        &:hover
            color: $grey
```

*Slika 17: Primjer indentacije*

Nakon završene Sass strukture aplikacije korišten je gulp koji je Sass dokumente transpilirao u CSS dokumente te je CSS dokumente minificirao odnosno uklonio sve razmake kako bi se postigla veća brzina pri učitavanju.



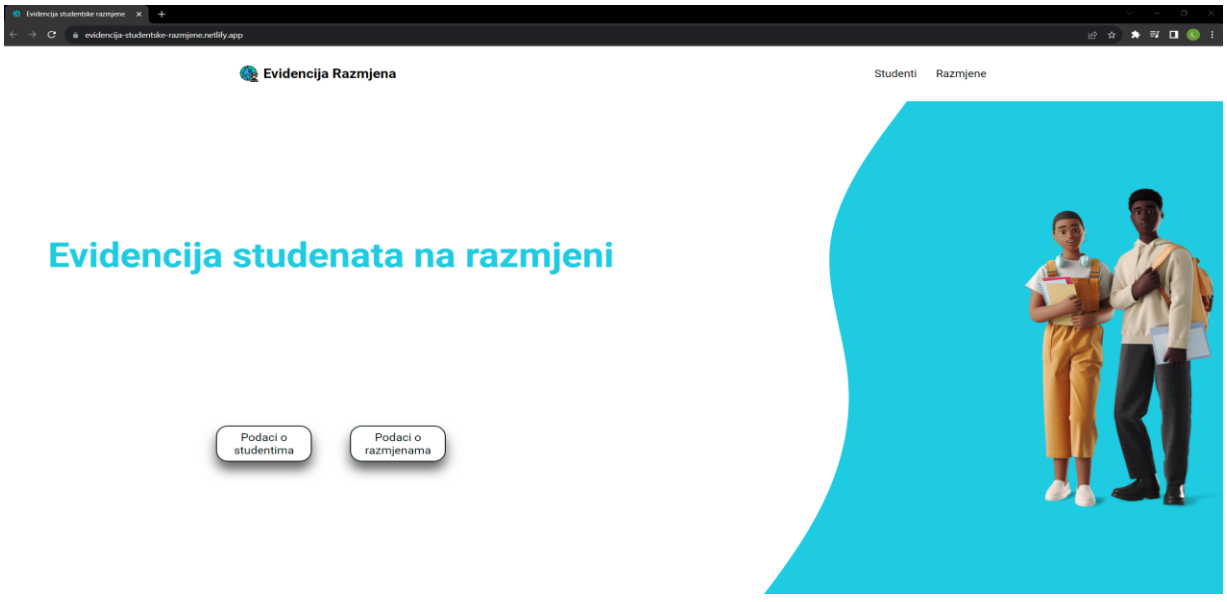

*Slika 18: Nastali CSS i minificirani dokumenti*

Za obavljanje operacija poput brisanja, uređivanja pregleda te ažuriranja kreirane su zasebne komponente koje u obliku formi primaju podatke te koristeći axios zahtjev šalju pozadinskom sloju te ponovno od pozadinskog sloja traže ažurirane podatke.

```jsx
<>
  <main>
    <h1>Unesite ID studenta kojeg želite izbrisati</h1>
    <form onSubmit={(e) => e.preventDefault()}>
      <div>
        <label htmlFor="student_id">ID:</label>
        <input
          type="number"
          name="student_id"
          id="student_id"
          placeholder="ID Studenta"
          value={id}
          onChange={handleInput}
        />
        <button type="button" onClick={handleDelete}>
          Izbriši
        </button>
      </div>
      <Link to="/studenti">
        <button onClick={fetchStudents}>Popis studenata</button>
      </Link>
    </form>
  </main>
</>
```

*Slika 19: JSX za brisanje studenata*

Nakon završetka razvoja klijentskog sloja aplikacije korištena je naredba *npm run build* koja kreira mapu unutar koje su svi dokumenti potrebni za objavu projekta. Za objavu projekta korišten je Netlify kao jednostavno i besplatno rješenje te je aplikacija javno objavljenja te dostupna na poveznici [https://evidencija-razmjena-studenata.netlify.app/](https://evidencija-razmjena-studenata.netlify.app/).




*Slika 20: Aplikacija unutar web preglednika*

### 7.4.Korištenje sustava za verzioniranje koda

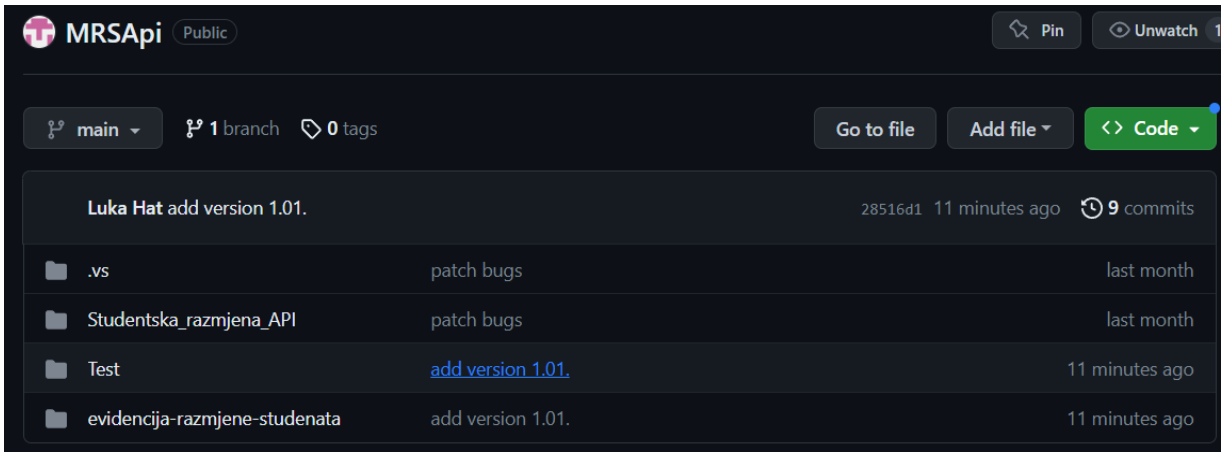
Za verzioniranje koda korišten je Git Bash. Ova aplikacija koristi se kroz tekstualno sučelje koristeći naredbe. Koristeći boje odvajaju se dijelovi sintakse kako bi korisničko iskustvo bilo ugodnije za korisnika. Git nam omogućuje razne prednosti poput povratka koda na starije verzije, pregleda dodanog koda, stvaranje različitih grana putem kojih možemo isprobati funkcije aplikacije prije implementacije te spajanje grana kako bi implementirali kod sa različitih grana. Git se prilikom izrade ove aplikacije koristio u kombinaciji sa GitHubom. Github je web platforma za dijeljenje sadržaja, najčešće putem Git-a te najčešće koda iako ima brojne mogućnosti. Putem Githuba podijeljen je kod aplikacije te su promjene koje su dodane u Git-u objavljene na GitHub-u. Git se koristio kako bi se pratile prethodne verzije aplikacije te uspoređivale promjene u odnosnu na prethodne verzije kao i dodavanje promjena na javno dostupni repozitorij na GitHub-u

```
PC@DESKTOP-OJMI992 MINGW64 /c/Medunarodna razmjena studenata (main)
$ git log
commit 28516d13449e0e433b94b0b91e1faef68e928f88 (HEAD -> main, origin/main)
Author: Luka Hat <luka.hat58@gmail.com>
Date:   Mon Sep 11 14:02:08 2023 +0200

    add version 1.01.

commit edd11822c53cf2299bb25bb885bd5daebd3c25fb
Author: Luka Hat <luka.hat58@gmail.com>
Date:   Fri Sep 8 03:55:52 2023 +0200

    finish version 1

commit cd64213db37acebdfbf9d0287849a2b72a9732d9
Author: Luka Hat <luka.hat58@gmail.com>
Date:   Fri Sep 8 00:02:59 2023 +0200

    add functionality

commit de1aa24425ff749d0af1c2129a2f6dff839227a2
Author: Luka Hat <luka.hat58@gmail.com>
Date:   Tue Sep 5 19:17:19 2023 +0200

    add basic structure and styling

commit 61bb3e628d416af986d79a7848b721e1e1083160
Author: Luka Hat <luka.hat58@gmail.com>
Date:   Wed Aug 16 21:50:44 2023 +0200

    patch bugs

commit fc9d122eaa0c16108e38706327b01c1e27d6750e
Author: Luka Hat <luka.hat58@gmail.com>
Date:   Sun Jul 2 22:27:31 2023 +0200

    update project

commit b9bf01249e3f8569c9724aad6a057dc00a56f78f
Author: Luka Hat <luka.hat58@gmail.com>
Date:   Sat Jul 1 23:32:16 2023 +0200

    deletion

commit 586fcfeafa8786af8848443cd770851426ab3c01
Author: Luka Hat <luka.hat58@gmail.com>
Date:   Fri Jun 30 18:05:34 2023 +0200

    remove default files

commit f3500b4be3a0551d46711bd32bae2fa06263b35c
Author: Luka Hat <luka.hat58@gmail.com>
Date:   Fri Jun 30 17:56:23 2023 +0200

    initial commit
```

*Slika 21: Povijest git commit-ova*

Na slici 20 imamo pregled prethodnih git commit-ova te unutar Git Bash sučelja možemo navigirati do verzija dokumenata prije ili poslije nekih promjena.

Sav kod za izradu ovog rada javno je dostupan na GitHub-u na poveznici https://github.com/LukaHat/MRSApi

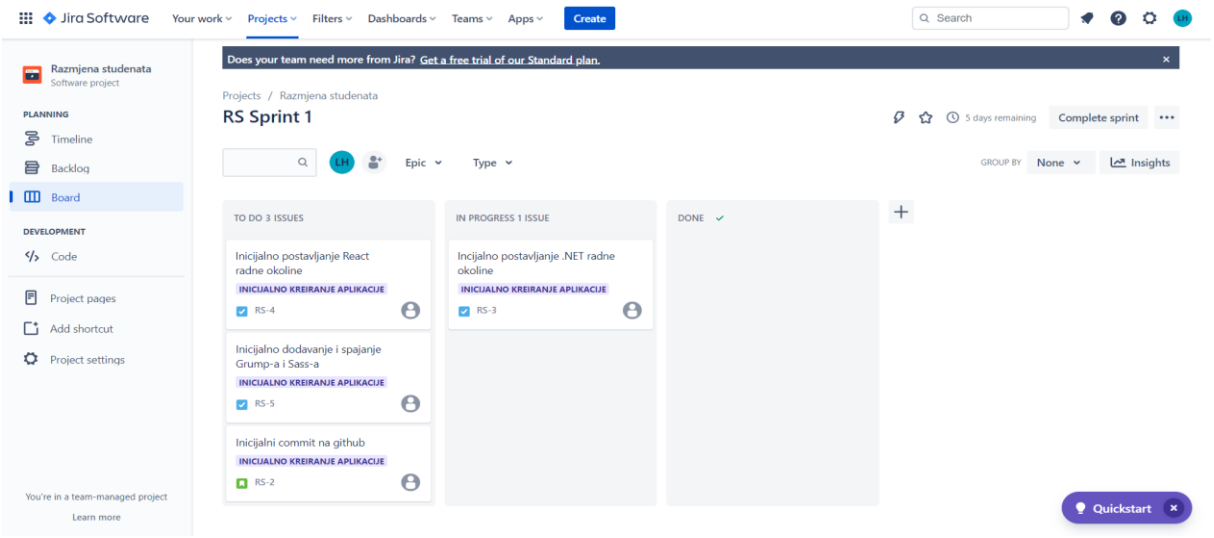



*Slika 22: GitHub pregled strukture aplikacije*

### 7.5.Korištenje JIRE

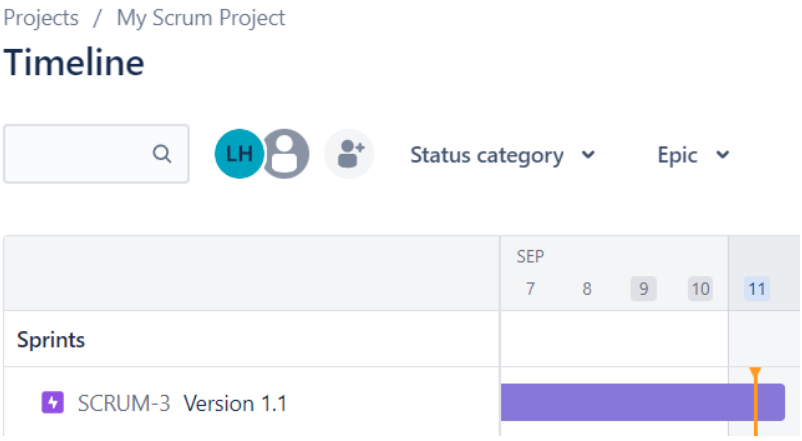
JIRA[1] je alat koji se koristi pri agilnom razvoju aplikacije. Pomoću JIRE možemo vizualno prikazati sprintove odnosno njegove faze. JIRA omogućuje dodavanje zadataka ili dijelova zadataka koji se zatim mogu dodijeliti u početnu fazu. JIRA unutar sprinta dijeli zadatke na „To do“ odnosno dio zadata koji se još mora napraviti, „In progress“ odnosno dio zadataka koji trenutno obavljamo te „Done“ sekciju za zadatke koje smo napravili.

[1] https://www.atlassian.com/software/jira




*Slika 23: Početni sprint*

Iz slike 22 vidljivo je kako su sekcije grafički odvojene i pregledne te se klikom na svaki zadatak mogu dobiti dodatne infromacije. Pomoću „Timeline“ dijela Jira alata vremenski je praćen razvoj aplikacije te grafički prikazani zadatci kroz njihov vremenski raspon.




*Slika 24: Timeline*

## Rasprava

Koristeći odabrane tehnologije postignut je željeni rezultat no potrebno je proučiti sve mogućnosti koje imamo na raspolaganju. Aplikaciju je bilo moguće napraviti i kroz MVC šablonu unutar Microsoft Visual Studio okruženja što bi značilo da na jednom mjestu imamo pregled sve funkcionalnosti u vidu dohvaćanja podataka kao i mogućnost izrade i manipulacije izgledom stranice no React je odabran zbog svoje jednostavnosti kao i odvajanja elemenata u komponente što smanjuje odgovornost pojedinog dijela koda te lakši proces otklanjanja grešaka kao i dizajniranja aplikacije. No React nije jedino radno okružje programskog jezika Javascript. Kao najveći konkurenti pojavljuju se Svelte, Vue.js te Angular. Svelte omogućuje najbrže aplikacije no radi se o novijem radnom okružju koje početno izlazi 2016. Godine što znači da nema zajedicu veličine React-a Vue-a te Angulara. Angular zahtjeva korištenje Typescript-a te ostale napredne značajke koje osiguravaju „čišći kod“ no otežavaju početak rada za početnike. Vue nudi slične značajke kao React te je lakši za početnike no također mu nedostaje velika zajednica i popularnost kakvu React ima. Na mjestu Sass-a možemo koristiti LESS koji omogućuje kreiranje određenih obrazaca dizajna elemenata koji se primjenjuju kada se neki uvjet zadovolji. Također smo mogli koristiti Bootstrap, CSS okvir koji omogućuje korištenje velikog broja unaprijed definiranih klasa koje sadrže određeni dizajn, slično tome postoji i opcija Tailwind-a koji također definira dizajn unutar klasa. Sass nam ipak nudi najveću slobodu pri kreiranju izgleda te dodaje funkcionalnost CSS-u za uvelike olakšan, ubrzan te manje repetitivan rad. Na mjestu Gulp-a mogao je biti korišten i Grump koji nudi istu funkcionalnost postignutu na drugačiji način no Gulp ipak zadržava prednost u brzini. Ovom usporedbom nekih od korištenih tehnologija te ostalim dostupnim opcijama vidljivo je kako odabrane tehnologije ipak zadržavaju prednost u nekim aspektima. Aplikacija nudi mogućnost kreiranja, brisanja, pregleda te ažuriranja podataka o studentima te razmjenama no moguće je dodati dodanu funkcionalnost. Budući da se radi o aplikaciji koja je namijenjena za korištenje administrativnom osoblju fakulteta moguće je dodati početni korak autentifikacije kako bi se potvrdio ovlašteni pristup. Kada bi to bilo implementirano moguće je dodati mogućnost prijave za studente koji žele ići na međunarodnu razmjenu te bi im bile pružene informacije o fakultetu te i potrebnim koracima za ostvarenje razmjene. Također je potrebno aplikaciju napraviti responzivnijom kako bi se jednako mogla koristiti neovisno o tome s kojeg uređaja joj se pristupa

# Zaključak

Kroz ovaj rad istražen je i opisan proces izrade web aplikacije za međunarodnu razmjenu studenata. Problem koji ova aplikacija nastoji riješiti je povećanje mobilnosti studenata što otežava evidenciju podataka o njima te manipulaciju nad tim podatcima. Kroz teorijsku osnovu danu u radu promotrena je troslojna arhitektura izrade aplikacije, agilni pristup izradi aplikacije te podjela aplikacije na podatkovni, pozadinski te klijentski dio.

Pozadinski sloj aplikacije kreiran je pomoću C# programskog jezika u .NET radnom okružju te je omogućio jednostavno kreiranje podatkovnog sloja bez potrebe za upitnim jezikom. Također je koristeći iste tehnologije kreiran pozadinski sloj koji nam omogućava jednostavne operacije nad podatcima te njihovo pozivanje u klijentskom sloju aplikacije. Aplikacija u klijentskom sloju na jednostavan način osigurava da korisnici imaju pregled svih podataka o studentima na razmjeni kao i o razmjenama te osigurava pozivanje metode za pregled, brisanje, ažuriranje te kreiranje istih.

Unutar Azure platforme na jednostavan način je javno objavljen pozadinski sloj aplikacije dok je klijentski objavljen pomoću Netlify platforme. Obje platforme nam pružaju brz i lagan pregled objavljenih resursa te neke napredne značajke.

Kao zaključak ova aplikacija doprinijela je rješenju problema povećane količine podataka uslijed povećanja studentske mobilnosti. Nekoliko načina na koji su riješeni ovi problemi su:

- Olakšan pregled podataka – aplikacija korisnicima pruža centralno web mjesto s kojega se mogu pregledati svi podatci o studentima i razmjena koji su potrebni
- Manipulacija podatcima – aplikacija pruža korisnicima mogućnost unosa,brisanja te uređivanja podataka o studentima i razmjenama
- Prilagodljivost – aplikacija se može jednostavno prilagoditi za korištenje velikom broju fakulteta te se može lako urediti vrsta potrebnih podataka
- Smanjenje troškova – aplikacija smanjuje administrativne troškove fakulteta koji ju koriste

Kroz pregled ovih prednosti možemo vidjeti kako je razvoj ove aplikacije uspješno riješio postavljen problem.

Aplikacija je razvijena na način raspoređene odgovornosti prema slojevima što omogućuje olakšan proces uklanjanja grešaka ukoliko se iste pojave. Također je unutar klijentskog sloja odgovornost podijeljena u komponente te se bilo kakvi vizualni ili funkcionalni problemi lako rješavaju i detektiraju. Aplikacija je također podložna promjenama i poboljšanjima u budućnosti čija bi implementacija bila brza i jednostavna.

U aplikaciji se u budućnosti očekuje nekakav oblik implementacije autentifikacije koja bi aplikaciju razdvojila na verziju dostupnu fakultetskom osoblju kao i verziju dostupnu studentima koji žele sudjelovati u međunarodnoj razmjeni. Aplikacija bi podržavala više jezika kako bi se prilagodila studentima raznih zemalja kao i otvorila za korištenje globalnom tržištu. Također je moguće implementirati analitiku i izvješća fakultetima koji koriste aplikaciju ovisno o njihovim potrebama. Aplikacija će se razviti responzivno kako bi pokrila mogućnost korištenja putem svih medija.

## Literatura


1. Merriam-Webster. "*Methodology*." Bez datuma. Dostupno na: https://www.merriam-webster.com/dictionary/methodology [Pristupljeno: 11.srpnja.2023.].
2. Steve Smith. "*ASP.NET Core MVC Overview*." 27.lipnja.2022. Dostupno na: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?WT.mc_id=dotnet-35129-website&view=aspnetcore-7.0 [Pristupljeno: 11.srpnja.2023.].
3. IBM. "*Three-Tier Architecture*." Bez datuma. Dostupno na: https://www.ibm.com/topics/three-tier-architecture [Pristupljeno: 2.kolovoza.2023.].
4. Kate Brush. "*Agile Software Development*." 10.studeni.2022.. Dostupno na: https://www.techtarget.com/searchsoftwarequality/definition/agile-software-development [Pristupljeno: 2.kolovoza.2023.].
5. Europska komisija. "*Opportunities for Individuals - Studying Abroad*." Bez datuma. Dostupno na: https://erasmus-plus.ec.europa.eu/hr/opportunities/opportunities-for-individuals/students/studying-abroad [Pristupljeno: 3.kolovoza.2023.].
6. Microsoft. "*C# Programming Language*." Bez datuma. Dostupno na: https://learn.microsoft.com/en-us/dotnet/csharp/ [Pristupljeno: 20.svibnja 2023.].
7. Git. "*Git - Distributed Version Control System*." Bez datuma. Dostupno na: https://git-scm.com/ [Pristupljeno: 20.svibnja 2023.].
8. React. "*React - A JavaScript Library for Building User Interfaces*." Bez datuma. Dostupno na: https://react.dev/ [Pristupljeno: 20.svibnja.2023.].
9. Mozilla Developer Network. "*HTTP Methods*." 10.travnja 2023. Dostupno na: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods [Pristupljeno: 20.kolovoza.2023.].
10. Swagger. "*What is Swagger?*" Bez datuma. Dostupno na: https://swagger.io/docs/specification/2-0/what-is-swagger/ [Pristupljeno: 20.svibnja 2023.].
11. React Router. "*React Router - Declarative Routing for React.*" Bez datuma. Dostupno na: https://reactrouter.com/en/main [Pristupljeno: 10.srpnja 2023.].
12. Sass. "*Sass: Syntactically Awesome Style Sheets*." 2006. Dostupno na: https://sass-lang.com/ [Pristupljeno: 7.srpnja 2023.].
13. Gulp. "*Gulp.js - A Toolkit for Automating Painful or Time-Consuming Tasks in Your Development Workflow.*" 2013. Dostupno na: https://gulpjs.com/ [Pristupljeno: 7.sprnja.2023.].

14. Plavno. "*Software Development Methodologies in 2023*." 24.siječnja,2023.. Dostupno na: https://plavno.io/company/blog/software-development-methodologies-2023 [Pristupljeno: 11. srpnja 2023.].
15. K.Beck,M.Beedle, A. van Bennekum "*The Agile Manifesto.*" Bez datuma. Dostupno na: https://agilemanifesto.org/iso/en/manifesto.html [Pristupljeno: 11. srpnja 2023.].

## Popis slika