eNastavnik

Lučić, Mato

Master's thesis / Diplomski rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:053992

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-24**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

ENASTAVNIK

Diplomski rad

Mato Lučić

Osijek, 2018.

Sadržaj

1. UVOD 1

2. TEHNOLOGIJA IZRADE 2

2.1. C# programski jezik 2

2.1.1. Tipovi podataka 3

2.2. ASP.NET 7

2.2.1. MVC 7

2.2.2. Entity Framework 8

2.2.3. WebAPI 8

2.2.4. Dependency Injection (DI) 8

2.2.5. Objektno-relacijsko mapiranje (ORM) 10

2.3. AngularJS 11

2.4. SQL 11

3. eNASTAVNIK - Backend 12

3.1. Baza podataka 12

3.1.1. Tablica „Korisnik“ 15

3.1.2. Tablica „Predmeti“ 15

3.1.3. Tablica „Ocjene“ 16

3.1.4. Tablica „Kviz“ 17

3.1.5. Tablica „Ucenici“ 17

3.2. DAL (Data Access Layer) 18

3.3. Reporsitory, Generic reporsitory i Service 18

3.4. MVC_WebApi 19

3.5. Token 20

4. eNASTAVNIK – Frontend 21

4.1. Prijava kao uloga „ucitelj“ 24

4.2. Prijava kao uloga „ucenik“ 33
4.3. Prijava kao uloga „roditelj“ 36
5. ZAKLJUČAK 38
LITERATURA 40
SAŽETAK 41
ABSTRACT 42
ŽIVOTOPIS 43
PRILOZI 44
ELEKTRONIČKA VERZIJA DIPLOMSKOG RADA NA CD-u ILI DVD-u 45

1. UVOD

U kontekstu lakšeg pristupa informacijama u školstvu ukazala se potreba za aplikacijom koja će omogućiti evidenciju učenikova napretka u radu. Dakle, potrebno je napraviti interaktivnu aplikaciju namijenjenu nastavnicima, učenicima i roditeljima za vođenje evidencije o uspjehu učenika, što je ujedno i tema ovog diplomskog rada. Nastavnici moraju imati mogućnost prilagodbe aplikacije svojim potrebama i sve mogućnosti administratorskog pristupa. Učenici mogu imati pristup pregledu unesenih podataka koji su njihovi i pristup kvizovima znanja iz pojedinog predmeta. Roditelji mogu pristupiti samo unesenim podatcima u bazu i pregledati ocjene učenika koji im je odobren.

Aplikacija je napravljena u web tehnologiji pomoću programskih jezika i alata: C#, ASP.NET, AngularJS, HTML, JavaScript, CSS, SQL. Korištenje navedenih web tehnologija omogućio je programski paket Visual Studio. Uvid u bazu podataka i manipulaciju podacima olakšao je SQL Server Managment Studio, dok provjeru pozadinskog odvijanja rada s podacima omogućuje programski paket Postman.

U narednim poglavljima bit će objašnjene navedene tehnologije, njihova upotreba u realizaciji web aplikacije. Dakle, u drugom poglavlju objašnjena je tehnologija izrade, njezine primjene u diplomskom radu i općenito. Treće poglavlje govori o Backendu[1] i načinima realizacije problema koji se javio u tom dijelu aplikacije. Unutar Backenda podatak putuje iz baze prema prednjem dijelu vidljivom korisniku gdje se prikaže u onom obliku koji je u aplikaciji potreban. Objašnjena je struktura baze, stvaranje tablica, metode kojima podatak putuje iz modela u model, WebApi i Token. Backend je programiran u ASP.NET okruženju. Dio vidljiv korisniku je objašnjen unutar četvrtog poglavlja i naziva se Frontend[2], a programiran je u JavaScript okviru (u nastavku dokumenta ću koristiti termin framework) – AngularJS framework. Naime, četvrto poglavlje je strukturirano prema ulogama korisnika u sustavu (nastavnik, učenik, roditelj), a unutar tih struktura su objašnjene njihove mogućnosti u aplikaciji. Također, ovaj dio aplikacije omogućuje sučelje korisniku da bi mogao upravljati radnjama aplikacije i predstavlja korisniku vidljivi dio aplikacije.

[1] Backend - pozadinski dio aplikacije. Najčešće uzima podatak iz baze, obavi radnju i prenese u prednji dio koji je vidljiv korisniku (Frontend).
[2] Fontend - dio aplikacije koji je vidljiv korisniku.

2. TEHNOLOGIJA IZRADE

Tehnologija izrade podrazumijeva programske jezike: C# programski jezik, JavaScript, SQL i njihove frameworke za lakšu implementaciju koda: AnguarJS, ASP.NET i dr. Oni su korišteni u rješavanju problema prilikom izrade aplikacije i zbog toga će u ovom poglavlju biti podobnije objašnjene njihove mogućnosti i značajke.

2.1. C# programski jezik

C# je programski jezik koji je dizajniran 2002. godine za izgradnju aplikacija koje se pokreću na .NET platformi. C# je jednostavan, siguran i moćan objektno orijentirani programski jezik. Razvijen je 2000. godine od strane tvrtke Microsoft u laboratoriju pod vodstvom Andersa Hejlsberga, Scotta Wilatmutha i Petera Goldea. C# je potpuno objektno orijentirani jezik što znači da svi elementi unutar njega predstavljaju objekt. Objekt predstavlja strukturu koja sadrži podatkovne elemente, kao i metode i njihove međusobne interakcije. Iduća bitna stavka koja je implementirana u C# programsku jeziku je generičko programiranje. Pojam označava stil programiranja u kojem se pišu algoritmi uz uvjet da se tip podataka specificira prilikom implementacije algoritma ovisno o tipu parametara koji su potrebni algoritmu. Generičko programiranje i generički tipovi podataka su uvedeni da se spriječi dupliciranje koda jer se kreira samo jedna jedinstvena metoda ili klasa koja može biti bilo kojeg tipa prilikom implementacije. Microsoft je od 2014. godine omogućio programerima da do tada korišteni kompajler koji je kompajlirao C# kod, kodnog imena Rosyn preoblikuje u kompajler koji će biti open-source. Taj novi kompajler je dobio ime C# tako da danas C# kod kompajlira njegov kompajler (C#) što uvelike olakšava te ubrzava kompajliranje, pronalaženje greški u kodu, omogućuje brži rad i slično. Kako je rečeno u gornjem tekstu C# je objektno orijentirani programski jezik kojeg prate paradigme poput klasa, objekt, metoda, varijable, enkapsulacije, nasljeđivanja, apstraktne klase, interfejsi, konstruktori. [1]

Klasa je osnovna programska cjelina svakog objektno orijentiranog programskog jezika. Ona predstavlja generalizaciju pojma strukture iz programskog jezika C. Kao i struktura, klasa je samo nacrt po kome se strukturira objekt. Konstruira se ključnom riječju *class*.

Objekt nastaje instanciranjem klase, odnosno stvaranjem varijable koja u sebi drži vrijednost napisane klase te omogućuje pristup svim varijablama i metodama koji su u klasi.

Metoda je procedura dodijeljena klasi objekta. Omogućuje objektu da odradi funkciju koja je potrebna u aplikaciji.

Varijabla je memorijska lokacija čiji se sadržaj može mijenjati tijekom izvođenja programa. Varijable su neizostavan dio svakog programskog jezika i služe kao pomoć u samom procesu programiranja (točno poznavanje memorijske lokacije je zamijenjeno poznavanjem definicije varijable). [2]

Enkapsulacija znači pravo pristupa svojstava i metoda unutar neke klase. Postoji private, public i protected pristup. Private pristup dozvoljava pristup samo unutar klase u kojoj se varijabla ili metoda nalazi. Public omogućuje pristup varijabli, metodi iz bilo koje klase. Protected dozvoljava pristup unutar klase u kojoj se nalazi varijabla ili metoda i klasi koja nasljeđuje tu klasu.

Nasljeđivanje omogućuje ponovnu upotrebljivost koda ovisno o pravu pristupa (enkapsulaciji).

Apstraktne klase služe kao opis ponašanja klase koja ih nasljeđuje. Sadrže metode koje će se implementirati u klasama koje ih nasljeđuju. Kreiranje instance apstraktne klase nije moguće. Apstraktna klasa se kreira ključnom riječju abstract.

Interfejsi su kosturi klasa koje ih nasljeđuju. Unutar interfejsa se definiraju metode i prava pristupa za svaku metodu implementacijom interfejsa potrebno je naslijediti sve metode definirane u interfejsu.

Konstruktor je posebna vrsta metode zadužena za inicijalizaciju objekta njegove klase. Razlikuje se od metode po tome što ima isto ime kao i klasa te nema povratni tip. [3] Ako se stvori vise konstruktora dolazi do pojave koja se naziva preopterećenje.

2.1.1. Tipovi podataka

C# je programski jezik u kojem sve varijable i objekti moraju imati deklariran tip podatka. Varijablama možemo zadati da imaju neki od ugrađenih tipova, poput `int` ili `char` tipa, ili neki korisnički-definirani tip poput klase ili sučelja. [4]

Drugi način podijele je na:

1. vrijednosne tipove (*value type*)
2. referentne tipove (*reference type*)

Varijable koje imaju vrijednosni tip direktno sadržavaju vrijednost. Kad pridružujemo jednu varijablu vrijednosnog tipa drugoj varijabli, vrijednost se kopira. Ovo je različito od slučaja kad su varijable referentnog tipa, tada se kopira samo referenca, ali ne i sam objekt. [4]

Prvi, vrijednosni tipovi dalje se dijele na dvije glavne kategorije:

1. Strukture
    a. Numeričke tipove

U numeričke tipove podataka spadaju cjelobrojni i decimalni tip (realni tip).

Za cjelobrojni tip se koriste sljedeći identifikatori sbyte, byte, short, ushort, int, uint, long i ulong. Razlika između ovih tipova je u veličini broja koji mogu prihvatiti te pohraniti. Identifikatori sbyte, byte, short i ushort se koriste za manje cijele brojeve (do 255), dok se za veće cijele brojeve koriste int i uint. Za jako velike cijele brojeve se koriste long i ulong. [5]

Za realni tip koristimo: float, double i decimal. Razlika između ovih tipova je u veličini koji mogu prihvatiti i preciznosti. [5]

Numeričke vrijednosti se mogu koristiti za izvođenje matematičkih operacija (+ ,-, *, /). Operator '%' prikazuje ostatak poslije cjelobrojnog dijeljenja. Numeričke vrijednosti možemo usporediti pomoću logičkih operatora (==, <, <=, >, >=, !=). Također, nad numeričkim operatorima se mogu izvršavati funkcije (Abs(x), Math.Exp(x), Math. Pow(x,y), Math.Sin(x)). [5]

    b. Integralne tipove

Tab. 2.1. Veličina i raspon integralnih tipova

Tip	Raspon	Veličina
sbyte	-128 to 127	Signed 8-bit integer
byte	0 to 255	Unsigned 8-bit integer
char	U+0000 to U+ffff	Unicode 16-bit character
short	-32,768 to 32,767	Signed 16-bit integer
ushort	0 to 65,535	Unsigned 16-bit integer
int	-2,147,483,648 to 2,147,483,647	Signed 32-bit integer
uint	0 to 4,294,967,295	Unsigned 32-bit integer
long	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	Signed 64-bit integer
ulong	0 to 18,446,744,073,709,551,615	Unsigned 64-bit integer

[4]

c. Tipove sa pomičnim zarezom

Tab. 2.2. Preciznost i raspon tipova sa pomičnim zarezom

Tip	Raspon	Preciznost
float	±1.5e-45 to ±3.4e38	7 digits
double	±5.0e-324 to ±1.7e308	15-16 digits

[4]

d. Decimal

Tab. 2.3. Decimal

Tip	Raspon	Preciznost
decimal	±1.0 × 10e-28 to ±7.9 × 10e28	28-29 significant digits

[4]

Decimal je 128-bitni vrijednosni tip. U usporedbi sa tipom s pomičnim zarezom, decimalni tip ima veću preciznost i manji raspon, što ga čini pogodnijim za financijske i novčane proračune. [4]

e. Bool

Ključna riječ *bool* je alias za System.Boolean tip. Koristi se za deklariranje varijabli koje mogu pohranjivati Boolean vrijednosti true i false. [4]

f. Korisnički definirane strukture

Korisnički definirane strukture se koriste za enkapsulaciju manje grupe povezanih varijabli, poput koordinata pravokutnika ili karakteristika nekog predmeta iz skladišta. Ključna riječ je *struct.* [4]

2. Pobrojani tip

Pobrojani tip ili enumeracija se stvara pomoću enum ključne riječi, a koristi se za definiranje seta konstanti. Svaka enumeracija ima ispod sebe neki integralni tip (osim char tipa). Defaultni tip je int. Po defaultu, prva konstanta ima vrijednost 0, a sve ostale po 1 više od prethodnog. [4]

Drugi, referentni tipovi čuvaju samo reference na stvarne podatke. Razliku u odnosu na vrijednosne tipove smo već spomenuli kod opisa vrijednosnih tipova. Osnovna podjela referentnih tipova je na:

- Klase
- Sučelja
- Delegate [4]

Postoje i dva već ugrađena referentna tipa:

- Object
- String[4]

* U slučaju da vrijednosne tipove želimo koristiti kao objekte koristimo *boxing* i *unboxing* postupak.

Boxing je postupak koji je obavlja kad vrijednosni tip "pakiramo" u referentni tip Objekt. To omogućuje vrijednosnom tipu da ga stavimo na gomilu (*heap*) kojom upravlja *garbage collector*. Boxing vrijednosnog tipa alocira instancu objekta na gomili i kopira vrijednost u novi objekt.

Unboxing je eksplicitna konverzija iz tipa objekt u vrijednosni tip. Unboxing operacija se sastoji od:

- provjere da je instanca objekta boxed vrijednost zadanog vrijednosnog tipa
- kopiranja vrijednosti iz instance u varijablu vrijednosnog tipa

[4]

2.2. ASP.NET

ASP.NET je '.NET' (eng. *Dot-Net*) bazirano okruženje, gdje se aplikacije mogu razvijati u bilo kojem .NET kompatibilnom jeziku uključujući C#, VB.NET i JScript.NET. Drugim riječima, cijeli '.NET Framework' je na raspolaganju bilo kojoj ASP.NET aplikaciji. Programeri mogu lako iskoristiti pogodnosti ove tehnologije, koja uključuje CLR (eng. *Common Language Runtime*) okruženje, nasljeđivanje i slično.

ASP.NET je dizajniran da radi s 'WYSIWYG' (eng. *What You See Is What You Get*) HTML editorima i drugim alatima, uključujući Visual Studio .NET koji ne samo da značajno olakšava web programiranje već omogućava i korištenje svih pogodnosti koje on pruža, uključujući razvoj web aplikacija u grafičkom okruženju.

Za razliku od klasičnog ASP-a, ASP.NET programski model je više objektno orijentiran što ga dodatno približava standardnim objektno orijentiranim razvojnim okolinama i jezicima kao što su C++, C#, VB i slično. [6]

ASP.NET se temelji na MVC pristupu te aplikaciji omogućuje olakšani pristup bazi pomoću Entity Frameworka i dohvaćanje podataka pomoću Web API-ja.

2.2.1. MVC

MVC je kratica za generalnu strukturu .NET aplikacije i glasi „Model-View-Controller“.

Model – domena oko koje je sagrađna aplikacija. Zasnovan na stvarnom slijedu i njegovim objektima. Primjer je preslike baze podataka.

View – vizualni prikaz danog modela u samom kontekstu. Realiziran je preko HTML-a, CSS-a, JavaScripta.

Controller – koordinator koji je veza između Viewa i Modela. Realiziran je u C# programskom jeziku. Korisniku omogućuje pristup bazi.

2.2.2. Entity Framework

Entity Framework (EF) je preporučena tehnologija pristupa podacima za aplikacije u .NET-u. To je objektno-relacijski maper koji omogućuje .NET programerima rad s relacijskim podacima pomoću objekata specifičnih za pristup bazi podataka. Programerima omogućuje olakšan pristup bazi podataka s manje pisanja kooda, ulaska u logiku rada sa bazom te prijenosom podataka i slično.

Entity Framework omogućuje tri pristupa podacima i generiranju baze: Database first i Model first.

Model first je takav pristup u kojem se stvaraju modeli kao reprezentativni primjer baze podataka zatim Entity Framework s obzirom na modele stvara bazu podataka. Modelima je potrebno definirati i vezu između modela kako bi tablice mogle dobro funkcionirati i stvorile logički pristup podatcima. U koliko određena klasa treba imati pristup većoj količini podataka druge klase koriste se kolekcije objekata, a ako je potrebno pristupati samo određenom podatku iz klase onda je dovoljno u klasi navesti samo objekt klase kao atribut. Model first pristup je korišten prilikom stvaranja ove aplikacije.

2.2.3. WebAPI

WebAPI okruženje se koristi za izradu RESTful aplikacija. Korištenjem ovog okruženja omogućujemo upravljanje HTTP zahtjevima. WebAPI nam omogućuje komunikaciju između sučelja koji je namijenjen korisniku prilikom upravljanja aplikacije i pozadinskog dijela aplikacije koji manipulira podacima. Pozadinski dio zovemo Backend, a dio namijenjen korisniku za interakciju s bazom zovemo Frontend.

2.2.4. Dependency Injection (DI)

DI je jedan od načina spajanja koda u velikim aplikacijama. Omogućuje lako spajanje, održivost, manje raspisivanja i lakše razumjevanje koda.

```
11 references | 0 exceptions
protected IeUciteljContext Context { get; set; }

0 references | 0 exceptions
public Reporsitory(IeUciteljContext context)//konstruktor da svakim pokretanjem stvori objekt od contexta
{
    this.Context = context;
}
```

Sl. 2.1. *Dependency Injection*

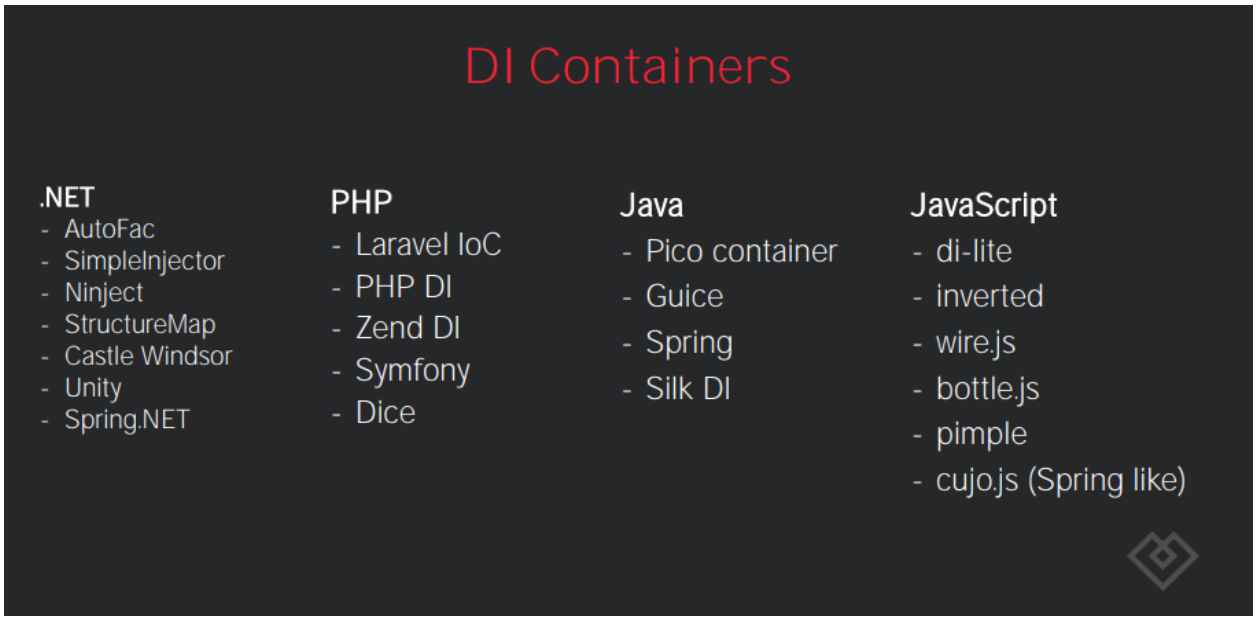
Prema slici **Sl. 2.1.** je vidljivo da konstruktor svakim pokretanjem stvori objekt od klase *IeUciteljContext*. Na ovaj način je stvorena ovisnost klase *IeUciteljContext* o klasi *Reporsitory*. Ovakav način se još zove i *Constructor Injection*.

DI je jedan od najčešćih načina korištenja injectiona. Objekt kase koji je stvoren ovim načinom je moguće koristiti bilo gdje u klasi i omogućuje pristup svim javnim metodama i funkcijama klase iz koje je stvoren objekt više puta bez ikakvih poduplavanja radnji te grešaka u rezultatu.

Alati za korištenje Dependency Injectiona, ovisni o programskom jeziku koji koristimo su nabrojani na slici **Sl. 2.2.** (.NET, PHP, Java, JavaScript). Najčešće korišteni alat u programskom okruženju .NET je *Ninject* i dolazi kao ekstenzija dodana preko NuGet Package Manager[3] alata.

Za rad s klasama nasljeđenim preko interfacea potrebno je napraviti klasu *Resolver.cs* i s funkcijom *Bind< >( ).To< >( )*, naslijeđenom iz *NinjectModule* klase naznačiti koje klase se naslijeđuju iz interfacea.

[3] NuGet Package Manager omogućuje sremanje i korištenje paketa (biblioteka – alati, funkcije, varijable). Centralno spremište za pakete koje koriste većina autora je nuget.org. [7]



Sl. 2.1. *Dependency Injection-a ovisno programskom jeziku* [8]

2.2.5. Objektno-relacijsko mapiranje (ORM)

Objektno-relacijsko mapiranje (ORM) u računalnom softveru je tehnika programiranja za pretvaranje podataka između nekompatibilnih tipova podataka u objektno-orijentiranim programskim jezicima. To stvara, zapravo, "virtualnu objektnu bazu podataka" koja se može koristiti unutar programskog jezika. Tu su i besplatni i komercijalni paketi dostupni da obavljaju objektno-relacijsko mapiranje, iako će se neki programeri odlučiti za stvaranje vlastitih ORM alata. [9]

Za korištenje OMR-a potrebno je dodati AutoMapper alat iz NuGet Pacage Manager biblioteke i stvoriti klasu koja će naslijediti klasu *Profile*. Unutar te klase su stvorene mape koje se inicijaliziraju svakim pokretanjem aplikacije. Na slici **Sl. 2.2.** je prikazan dio mapiranja iz modela koji se nalazi u DAL-u u Domain model.

```
0 references | 0 exceptions
public MappingProfile()
{
    //POCO to Domain
    CreateMap<Korisnik, KorisnikDomainModel>().PreserveReferences().ReverseMap().PreserveReferences();
    //POCO to IDomain
    CreateMap<Korisnik, IKorisnikDomainModel>().PreserveReferences().ReverseMap().PreserveReferences();
    //POCO to IPOCO
    CreateMap<Korisnik, IKorisnik>().PreserveReferences().ReverseMap().PreserveReferences();
```

Sl. 2.2. *Mapiranje*

2.3. AngularJS

AngularJS je JavaScript framework napravljen od strane Google-a i omogućava stvaranje dobro strukturiranih, lakih za testiranje i održavanje Frontend aplikacija. On je framework koji definira brojne koncepte za pravilno organiziranje web aplikacije.

Aplikacija je napravljena uz pomoć AngularJs MVC modula koji su spomenuti u poglavlju 2.2.1.

AngularJS proširuje HTML pružajući direktive oznakama pomoću kojih se može izgraditi moćna i dinamična web aplikacija. Uz HTML, omogućuje i CSS stilske alate za oblikovanje i uređivanje HTML-a. [10]

Kao pomoć prilikom stilskog oblikovanja interfejsa između korisnika i preglednika korišten je framework za CSS stilske alate Bootstrap.

2.4. SQL

SQL (engl. Structured Query Language) je upitni programski jezik posebne namjene, dizajniran specifično za upravljanje podacima relacijskog sustava za upravljanje bazama podataka (RDBMS) te za procesiranje toka podataka relacijskog sustava upravljanja tokom podataka. Opseg SQL-a obuhvaća izradu, traženje, ažuriranje i brisanje podataka iz relacijskih baza podataka. [11]

3. eNASTAVNIK - Backend

Aplikacija je namijenjena nastavnicima, učenicima i roditeljima za vođenje evidencije o učeniku. Pristup aplikaciji je uređen kroz tri uloge: „ucitelj“, „ucenik“, „roditelj“. Ulozi „ucitelj“ su dodijeljena sva administratorska prava te joj je omogućeno da određuje uloge u sustavu, upravlja predmetima, kvizovima i kontrolira sve promjene u bazi. Ulozi „ucenik“ je omogućen pristup samo ocjenama koje mu je dodijelila uloga „ucitelj“ te svim kvizovima. Ulozi „roditelj“ omogućen je pristup ocjenama samo onog profila učenika koji mu je dodijelila uloga „ucitelj“. Ulaskom u aplikaciju uloga se određuje prijavom.

Aplikacija se sastoji od jedanaest projekata unutar kojih su klase. Klase su međusobno povezane te odrađuju funkcionalnost aplikacije. Aplikacija se pokreće na IIS poslužitelju koji se dobije zajedno s Microsoftovim Visual Studio alatom i funkcionira kao poslužitelj koji se nalazi na računalu gdje je aplikacija.

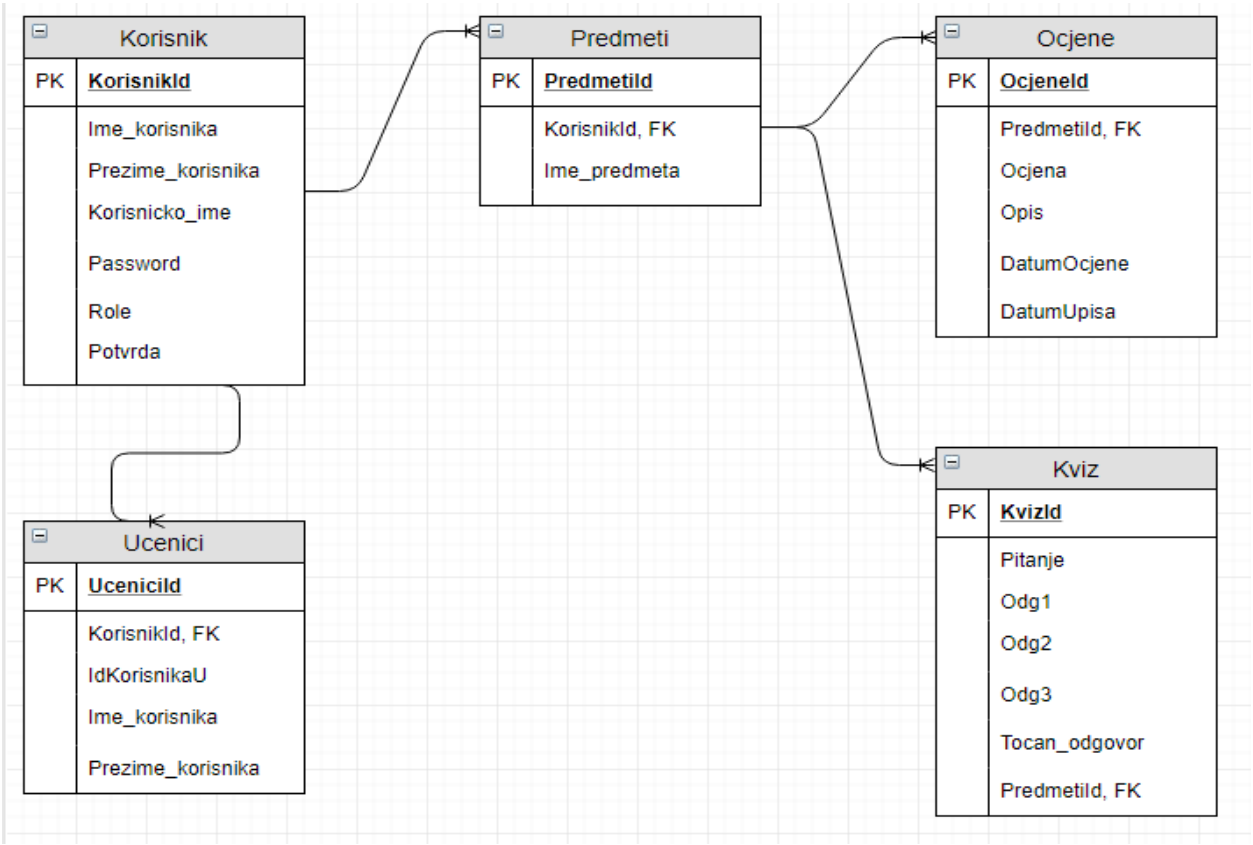
U daljnjem tekstu će biti obrađena baza podatka s metodama koje prenose podatak sve do korisničkog sučelja – Frontend.

3.1. Baza podataka

Baza se sastoji od pet tablica. Svaka tablica ima svoje stupce unutar kojih se pohranjuju podaci. Svaki podatak u tablicama se evidentira pod nekim ID-om. Ti ID-ovi su podatci tipa *Guid*. On je 128-bitni računalno generirani broj kojem je mala vjerojatnost ponavljanja pa je pogodan za predstavljanje redova u bazi podataka zbog svoje autentičnosti.

Svaka tablica ima svoj *Primary key* koji identificira svaki redak u tablici tipa je Guid. Neke tablice sadržavaju *Foreign key* također tipa Guid. Foreign key je jedan ili više redaka u tablici koji se odnose na Primary key u drugoj tablici te se na taj način povezuju dvije tablice.

U daljnjem tekstu ovog podnaslova bit će predstavljena svaka tablica u bazi. Slika **Sl. 3.1.** prikazuje shematski prikaz u .xml dijagramu baze podataka, a slike **Sl. 3.2.**, **Sl. 3.3.** i **Sl. 3.4.** prikazuju korisnike u bazi prikazane preko aplikacije Postman.



Sl. 3.1. *Baza podataka*


```
{
    "KorisnikId": "b1422b8b-681f-459a-934e-57bcee448c4d",
    "Ime_korisnika": "ucitelj",
    "Prezime_korisnika": "ucitelj",
    "Korisnicko_ime": "ucitelj",
    "Password": "2a3a9f8d3e89f38446b33ee2f4b479f5",
    "Potvrda": "Da",
    "Role": "ucitelj",
    "Predmeti": [],
    "Ucenici": []
},
```


Sl. 3.2. *Prikaz korisnika uloge „ucitelj“ u aplikaciji Postman*

```
{
    "KorisnikId": "c5d4790b-efd8-4aa1-a8f7-a8725c8d7ff0",
    "Ime_korisnika": "roditelj",
    "Prezime_korisnika": "roditelj",
    "Korisnicko_ime": "roditelj",
    "Password": "7bcdb890740dd25179e551d842e48ed5",
    "Potvrda": "Da",
    "Role": "roditelj",
    "Predmeti": [],
    "Ucenici": [
        {
            "UceniciId": "5aa9fb10-5f8c-48d4-9acd-3f7d1f7a1efc",
            "KorisnikId": "c5d4790b-efd8-4aa1-a8f7-a8725c8d7ff0",
            "IdKorisnikaU": "08dfd16b-3a94-4e1a-84cf-bc059ec182a2",
            "Ime_korisnika": "ucenik",
            "Prezime_korisnika": "ucenik"
        }
    ]
},
```

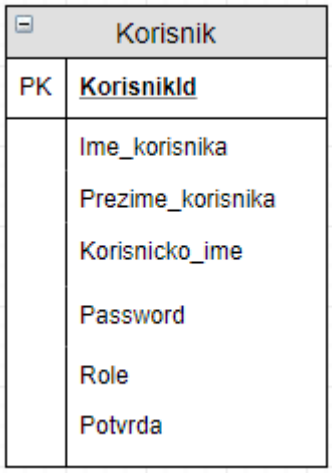

Sl. 3.3. *Prikaz korisnika uloge „roditelj“ u aplikaciji Postman*


```
{
    "KorisnikId": "08dfd16b-3a94-4e1a-84cf-bc059ec182a2",
    "Ime_korisnika": "ucenik",
    "Prezime_korisnika": "ucenik",
    "Korisnicko_ime": "ucenik",
    "Password": "3eca69f4d68e822c331adbbb1177c7c7",
    "Potvrda": "Da",
    "Role": "ucenik",
    "Predmeti": [
        {
            "Ime_predmeta": "Kemija",
            "KorisnikId": "08dfd16b-3a94-4e1a-84cf-bc059ec182a2",
            "PredmetiId": "d9ddebc9-6833-4c74-8fd9-097d8057ad48",
            "Ocjene": [
                {
                    "OcjeneId": "872c2405-d959-4ffa-82c2-e36b3ecb510f",
                    "PredmetiId": "d9ddebc9-6833-4c74-8fd9-097d8057ad48",
                    "Ocjena": 3,
                    "Opis": "Ispit znanja. Broj bodova 19/25.",
                    "DatumOcjene": "2018-03-07T00:00:00",
                    "DatumUpisa": "2018-03-07T00:00:00"
                }
            ],
            "Kviz": [
                {
                    "KvizId": "1245dd60-dcaf-4039-a8d8-6f2da31497da",
                    "PredmetiId": "d9ddebc9-6833-4c74-8fd9-097d8057ad48",
                    "Pitanje": "Simbol za kisik je:",
                    "Odg1": "O",
                    "Odg2": "N",
                    "Odg3": "H",
                    "Tocan_odgovor": "O"
                }
            ]
        }
    ],
    "Ucenici": []
}
```


Sl. 3.4. *Prikaz korisnika uloge „ucenik“ u aplikaciji Postman*

3.1.1. Tablica „Korisnik“

Tab. 3.1. *Korisnik*

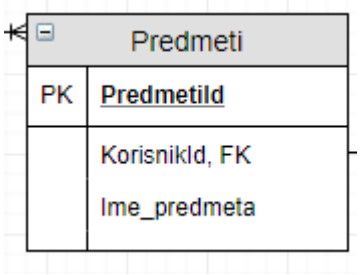


Tablica „Korisnik“ ima sedam stupaca koji pohranjuju ime korisnika, prezime korisnika, korisničko ime, password za pristup, role (uloge), potvrdu te *KorisnikId* koji je ujedno i PK (Primary Key) svakog korisnika.

Ime, prezime, korisničko ime i password predstavljaju osobne podatke korisnika i njih unosi sam korisnik. Rolu i potvrdu određuje korisnik uloge „ucitelj“ koji je i administrator sustava.

3.1.2. Tablica „Predmeti“

Tab 3.2. *Predmeti*

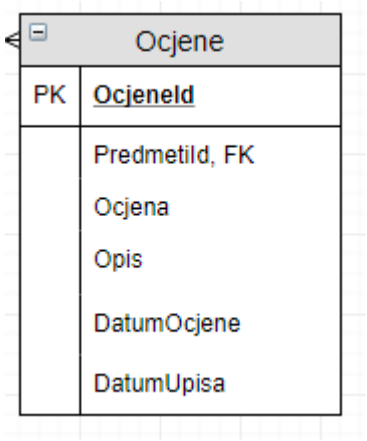


Tablica „Predmeti“ sadrži tri stupca koji pohranjuju ID predmeta koji je ujedno i Primary Key svakog predmeta, ime predmeta i ID korisnika kojem je dodijeljen predmet. Na taj način se

tablica „Predmeti“ referencira na tablicu „Korisnik“ i omogućuje da svaki korisnik može imati više predmeta.

3.1.3. Tablica „Ocjene“

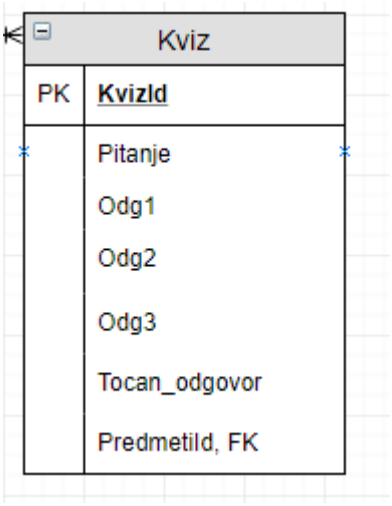
Tab. 3.3. *Ocjene*



Tablica „Ocjene“ sadrži šest stupaca. Primary Key (*OcjeneId*), Foreign Key (*PredmetiId*) služe za identičnost svakog retka u tablici i povezivanje s tablicom „Predmeti“. Na taj način svakom korisniku dodijeljeni su predmeti. Stupci *Ocjena* i *Opis* omogućuje upis ocjene s komentarom i obrazloženjem učitelja o danoj ocjeni. *DatumOcjene* predstavlja datum kada je ocjena zaslužena, a *DatumUpisa* se automatski upisuje i predstavlja stupac za unos datuma kada je učitelj uveo ocjenu u sustav.

3.1.4. Tablica „Kviz“

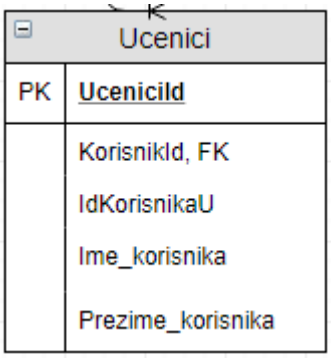
Tab. 3.4. *Kviz*



Tablica „Kviz“ sadrži sedam stupaca. Primary Key (*KvizId*), Foreign Key (*PredmetiId*) služe za identičnost svakog retka u tablici i povezivanje s tablicom „Predmeti“, pa tako za svaki predmet postoji mogućnost stvaranje kviza. U stupac *Pitanje* pohranjuje se pitanje, a u stupce *Odg1*, *Odg2*, *Odg3* i *Tocan_odgovor* se pohranjuju mogući odgovori i točan odgovor.

3.1.5. Tablica „Ucenici“

Tab. 3.5. *Ucenici*



Tablica „Ucenici“ sadrži pet stupaca. Primary Key (*UceniciId*), Foreign Key (*KorisnikId*) služe za identičnost svakog retka u tablici i povezivanje s tablicom „Korisnik“. Tablica omogućuje da uloga „ucitelj“ označi ulogu „ucenik“ kojem je uloga „roditelj“ pristupila sustavu. Na taj način

omogućava se ulozi „roditelj“ uvid u ocjene samo onog učenika kojem je roditelj. U tablicu se prepiše ime, prezime i ID učenika iz tablice „Korisnik“.

3.2. DAL (Data Access Layer)

Dohvaćanje, spremanje i ostala manipulacija podacima u bazi je omogućena preko Entity framework-a, koji sadrži posebne metode za pristup bazi. Navedeni framework je postavljen u DAL projekt u klasu *eUciteljContext.cs*. Klasa sadrži *DbSet* metode koje omogućuju Model first način stvaranja baze podatka gdje se iz danih modela, podešavanjem unutar *web.config*[4] dijela, stvori baza s danim tablicama, stupcima i redcima unutar njih.

Sve klase su preuzete iz interfacea *DAL.Common*. Iz interfacea se također preuzmaju i modeli za stvaranje baze podataka. Svaki model je napisan unutar .cs dokumenta i predstavlja jednu tablicu u bazi. Ulazni argument u funkciju *ForeignKey()* zajedno s funkcijom omogućuje metodi da označi točan Foreign Key na koji se referencira Primary key iz roditeljske tablice.

3.3. Reporsitory, Generic reporsitory i Service

Kreiranjem klase *Reporsitory.cs* te korištenjem Dependency Injectiona za dohvat podataka iz interfacea *IeUciteljContext* omogućeno je stvaranje generalizirane klase za manipulaciju elementima baze podataka. Metode dohvaćaju, spremaju, brišu i mijenjaju podatke u bazi. Na taj način je podacima omogućen tok od baze prema korisniku i obrnuto. Sve metode su asinkrone[5].

Za svaku tablicu kreirana je klasa koja alatom Dependency Injection stvara objekt interfacea klase *Reporsitory.cs*. Preko objekta pristupa se metodama klase što omogućuje da svaka tablica u bazi ima metode za upravljanje tokom podataka. Sve metode su asinkrone i omogućavaju nadzor nad bazom podataka te sve radnje koje su potrebne za bazu. Na ovom dijelu aplikacije, vrši se mapiranje iz standardnih modela koji su korišteni u *eUciteljContextu* za stvaranje baze podatka u *Domain modele*[6].

[4]Web.config je xml dokument koji omogućuje podešavanje svih aplikacija na serveru pisanih u ASP.NET tehnologiji.
[5]Asinkrone metode - metode koje se ne izvršavaju u isto vrijeme već jedna metoda čeka kraj izvršenja druge metode. U .Net tehnologiji se realiziraju putam ključnih riječi *async* i *await*.
[6] Domain modeli su modeli koji se koriste u Service klasi, a služe za business logiku aplikacije.

Unutar *Service* projekta stvarani su objekti klasa interfaceova *GenericReporsitoryja* putem Dependency Injectiona. Ti objekti su omogućili sve kompliciranije zahvate nad podatcima iz tablice (business logika).

3.4. MVC_WebApi

MVC_WebApi je glavni projekt aplikacije. U njemu se nalaze controlleri, modeli i view-ovi svake tablice. Inicijalizacijom klase unutar controllera stvori se objekt Service-a koji omogućuje preko ostalih povezanih projekata pristup bazi koja je na poslužitelju. Svaki controller nasljeđuje *ApiController* koji definira svojstva i metode za API controllere. Ovaj način se koristi u *RESTFul* aplikacijama. Kreira se *Task*, on odradi Http zahtjev koji se šalje prema poslužitelju, a poslužitelj vrati rezultat i odgovor kao status realizacije (**Sl. 3.5**.).

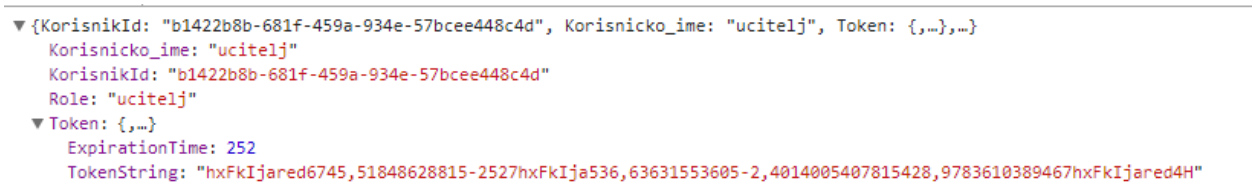
```
public class KorisnikController : ApiController
{
    10 references | 0 exceptions
    protected IKorisnikService KorisnikService { get; set; }

    0 references | 0 exceptions
    public KorisnikController(IKorisnikService korisnikService)
    {
        this.KorisnikService = korisnikService;
    }

    [HttpGet]
    [Route("getAllK")]
    0 references | 0 requests | 0 exceptions
    public async Task<HttpResponseMessage> GetAllKorisnik()
    {
        try
        {
            var response = Mapper.Map<IEnumerable<KorisnikViewModel>>(await KorisnikService.GetAll());
            return Request.CreateResponse(HttpStatusCode.OK, response);
        }
        catch (Exception e)
        {
            return Request.CreateErrorResponse(HttpStatusCode.InternalServerError, e);
        }
    }
```

Sl. 3.5. *Metoda za dohvaćanje svih korisnika*

Unutar metode koja se izvršava, potrebno je mapirati iz Domain modela u View model. View model je zadnji model u hijerarhiji. Omogućuje da se elementi prikažu u Frontendu.

3.5. Token

Token je mali servis koji obavlja funkciju stvaranja vrijednosti stringa koji se dodjeli prilikom prijave u aplikaciju. Token radi preko četiri klase od kojih jedna generira slučajni string određene težinske vrijednosti. Ostale klase služe kao modeli za stvaranje tokena. Token se poziva prijavom u aplikaciju, dodjeljuje se korisniku te se sprema u *Local storage*[7] aplikacije. Zajedno s tokenom se sprema *ExpirationTime(Vrijeme isteka),* korisničko ime i ID prijavljene osobe **(Sl 3.6.)**. ExpirationTime je varijabla koja broji vrijeme isteka prijavljenog člana. Nakon isteka vremena spremljenog u varijablu pohrana se automatski briše.



Sl. 3.6. *Korisnik spremljen u Local storage prilikom prijave u aplikaciju*

[7] Local Storage je pohrana podatka na korisnikovoj strani, unutar pretraživača.

4. eNASTAVNIK – Frontend

Zadatke Frontend dijela odrađuje programski jezik JavaScript s AngularJS frameworkom. AngularJS također funkcionira kao MVC (Modela, Viewa i Controllera). Takav način funkcioniranja pokazuje širinu AngularJS-a, gdje se unutar Controllera piše JavaScript kod koji je potpuno ne vezan za HTML (View) dio. Kod u ovoj aplikaciji je podijeljen u pet dijelova: „Controller, View, js, Services i app.js“.

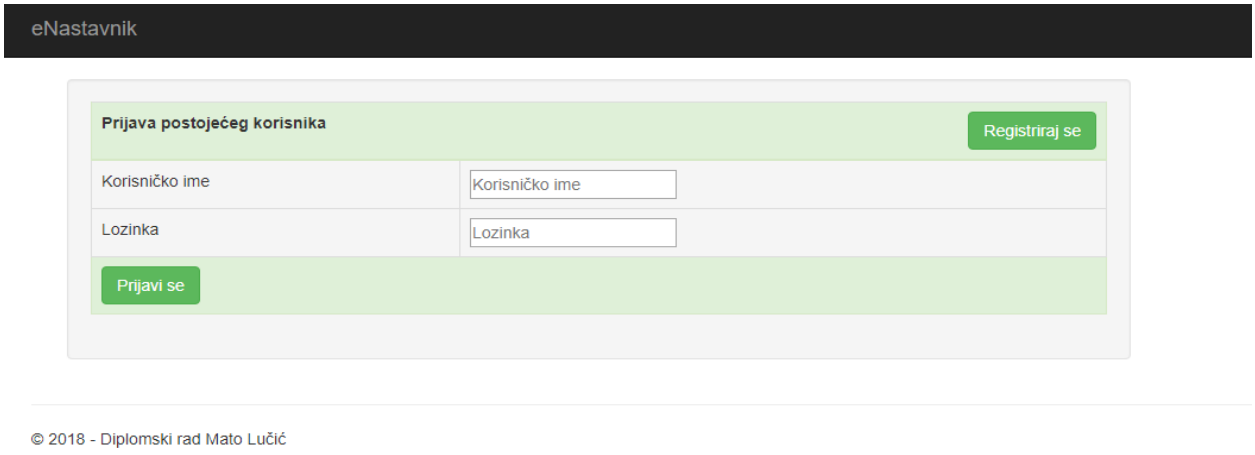
Controller sadržava metode i funkcije za manipulaciju podatcima u bazi. Svaka tablica u bazi ima svoje skripte koje pomoću *$http* metode imaju pristup .NET metodama s njihovim mogućnostima definiranim u podnaslovu eNASTAVNIK – Backend. Preko *$scope* metode se šalje obrađen podatak iz controllera u view. View se sastoji od .html dokumenata unutar kojih je HTML kod te tvori sve vidljivo u aplikaciji. View je podijeljen po tablicama i korišten ovisno o potrebi i zahtjevima aplikacije. Js sadržava različite filtere, a unutar Services-a su pohranjeni servisi za pomoć prilikom realizacije funkcija aplikacije. U aplikaciji su korišteni gotovi servisi npr. *AuthenticationService.js,* korišten kao pomoć za prijavu u aplikaciju. Taj servis preko $http metode poziva token prilikom prijave u sustav. App.js je modul za stvaranje ruta, tipično za AngularJS. Povezuje controller i view te ga stavlja u određenu rutu koja je zabilježena url-om. Url u ovom slučaju služi za pristup ruti (**Sl 4.1.**).

Aplikacija generalno funkcionira kao *Single page application (SPA)*. To je način dinamičkog učitavanja samo zahtijevane stranice s poslužitelja umjesto da učita sve stranice svakom promjenom u viewu. U AngularJS-u je SPA riješen *Data-binding*-om. Data-binding je automatski način promjene view-a svakom promjenom u modelu ili zahtijevom korisnika.

```
}).state('register', {
    url: '/korisnik/registracija',
    controller: 'RegisterController',
    views: {
        "root": {
            templateUrl: 'app/Views/Korisnik/Register.html'
        }
    }
```

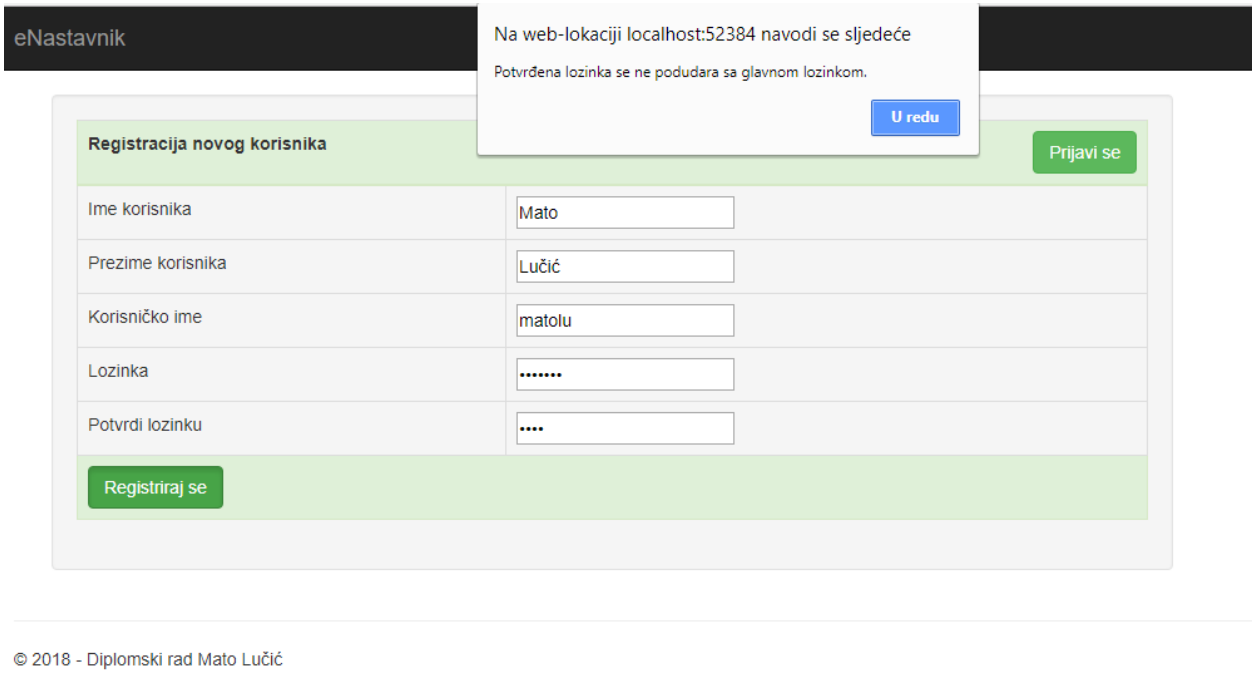
Sl. 4.1. *Ruta „register“*

Prva stranica koja se pojavljuje ulaskom u aplikaciju „eNastavnik“ zahtjeva prijavu u sustav (**Sl. 4.2.**). Aplikacija se temelji na tri ulog koje imaju drugačije mogućnosti te je zbog toga prijava u sustav najvažnija.



Sl 4.2. *Stranica za prijavu ako je korsinik registriran*

U slučaju da korisnik nije registriran potrebno je obaviti registraciju klikom na gumb „Registriraj se“. Prilikom registracije potrebno je unijeti ime, prezime korisnika, korisničko ime, lozinku te potvrditi lozinku da ne bi došlo do greške prilikom utipkavanja. Nakon unošenja svih podataka i klikom na gumb „Registriraj se“ provjeravaju se uneseni podatci. Glavna provjera je dali uneseno korisničko ime već postoji u bazi podatka. U slučaju da ime postoji ispisuje se poruka upozorenja. Druga važna provjera je dali se lozinka i potvrđena lozinka podudaraju. Ako uvjet nije ispunjen ispisuje se poruka upozorenja (**Sl 4.3.**).



Sl 4.3. *Stranica za registraciju i prikaz poruke o pogrešci*

Ako je uvjet ispunjen, korisnik se sprema u bazu te čeka potvrdu od administratora (rola „ucitelj“). Ulaskom korisnika koji nije potvrđen u sustav, na alatnoj traci mu je dozvoljeno vidjeti samo opće informacije o sustavu (**Sl 4.4.**).

Sl 4.4. *Mogućnosti nepotvrđenog korisnika u sustavu*

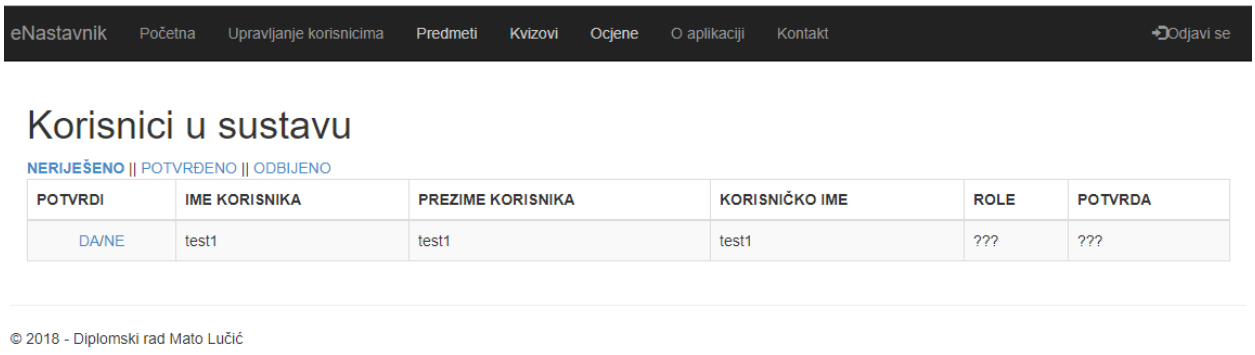
Prva poveznica na alatnoj traci „Početna“ je početna stranica sa slikom i pozdravnom porukom. Druga poveznica na alatnoj traci „O aplikaciji“ je uputstvo za korištenje aplikacije. Uputstvo je tehnička dokumentacija aplikacije. Na trećoj poveznici alatne trake „Kontakt“ navedene su odgovorne osobe kojima se može obratiti u slučaju greške ili ne mogućnosti korištenja aplikacije. Zadnja poveznica „Odjavi se“ automatski briše sve informacije pohranjene u Local storage memorije i korisnika prebacuje na početnu stranicu za prijavu novog korisnika. Sve tri uloge u sustavu imaju gore navedene poveznice.

4.1. Prijava kao uloga „ucitelj“

Uloga „ucitelj“ je ujedno i administrator sustava. U sustavu je pohranjen jedan sigurnosni korisnik koji ima ulogu „ucitelj“ te se kao takav ne može izmijeniti. On prvi administrira aplikaciju, prije prvog registriranja nastavnika u sustav. Također, taj pohranjeni sigurnosni

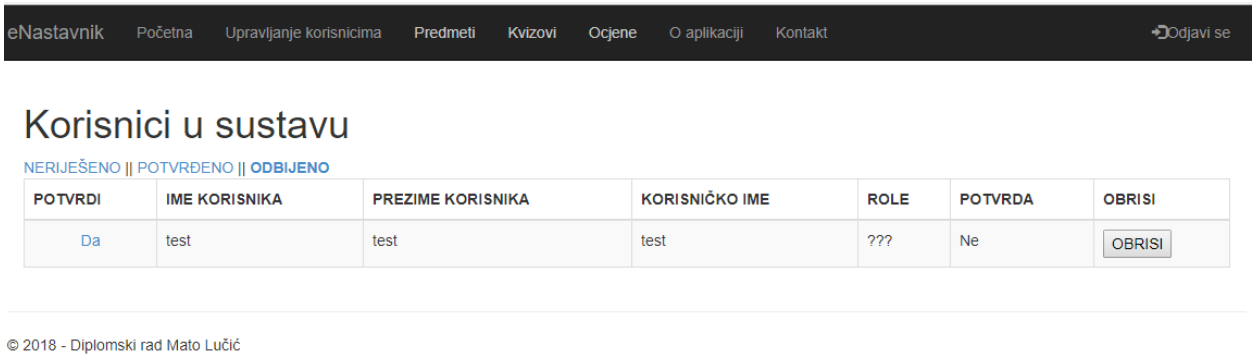
korisnik služi kao dodatna sigurnost za nastavnika u slučaju da se njegov profil obriše, ili da dođe do bilo kakve greške s njegovim profilom. Korisnik uloge „ucitelj“ dobiva podatke sigurnosnog profila (korisničko ime i lozinku) od osoba koji su razvili sustav. Nastavnik preko sigurnosnog profila jednom administrira svoj profil te daljnje korištenje istog je samo u slučaju nužde.

Uloga „ucitelj“ ima svu ovlast nad sustavom i svim ostalim registriranim korisnicima. Na slici **Sl. 4.5.** prikazane su mogućnosti korisnika koji se registrirao i ima ulogu „ucitelj“, a u daljnjem tekstu su objašnjene sve njegove mogućnosti.



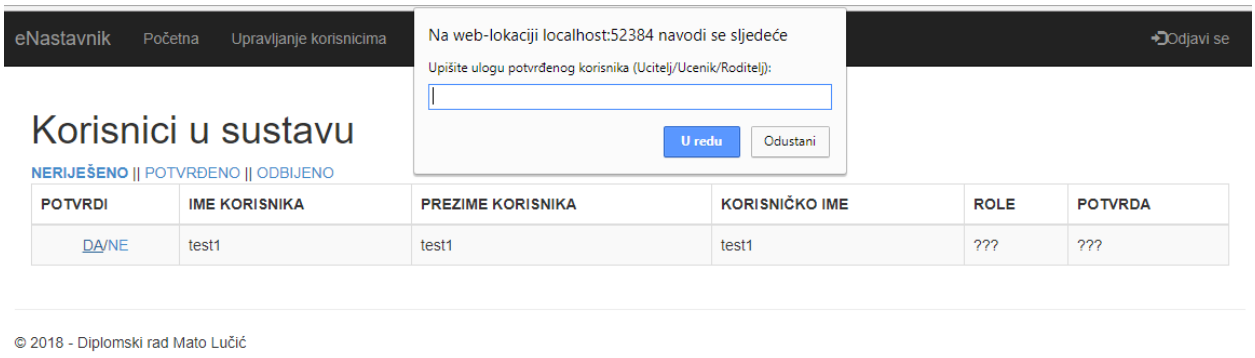
Sl. 4.5. *Korisnik prijavljen pod ulogom „ucitelj“*

Glavna mogućnost uloge „ucitelj“ je upravljanje korisnicima koji žele pristupiti sustavu. To je moguće u poveznici „Upravljanje korisinicima“, u alatnoj traci gdje su korisnici podijeljeni na tri dijela. Korisnici se dijele na: „NERIJEŠENO, POTVRĐENO, ODBIJENO“, s obzirom na stupac „POTVRDA“ u tablici. Poveznica „NERIJEŠENO“ ispisuje sve korisnike kojima su stupci „POTVRDA“ i „ROLE“ (uloga) jednake: „???“ (na slici **Sl. 4.5.** je prikazan korisnik koji je trenutno u kategoriji „NERIJEŠENO“). Administratoru (korisniku uloge „ucitelj“) je preko stupca „POTVRDI“ u tablici omogućeno mijenjanje potvrde. U slučaju da administrator odabere „NE“, korisnik odlazi u tablicu „ODBIJENO“. Stupac „POTVRDA“ mu se mijenja u „Ne“, a stupac „ROLE“ ostaje „???“. **(Sl. 4.6.)** Klikom na „Da“ u stupcu „POTVRDI“ korisnik će prijeći u kategoriju „POTVRĐENO“ čim administrator obavi sve potrebne radnje s obzirom na ulogu koja je dodijeljena korisniku. Korisnik koji je odbijen se može obrisati iz sustava klikom na „OBRIŠI“ u tablici te se trajno uklanja iz sustava.

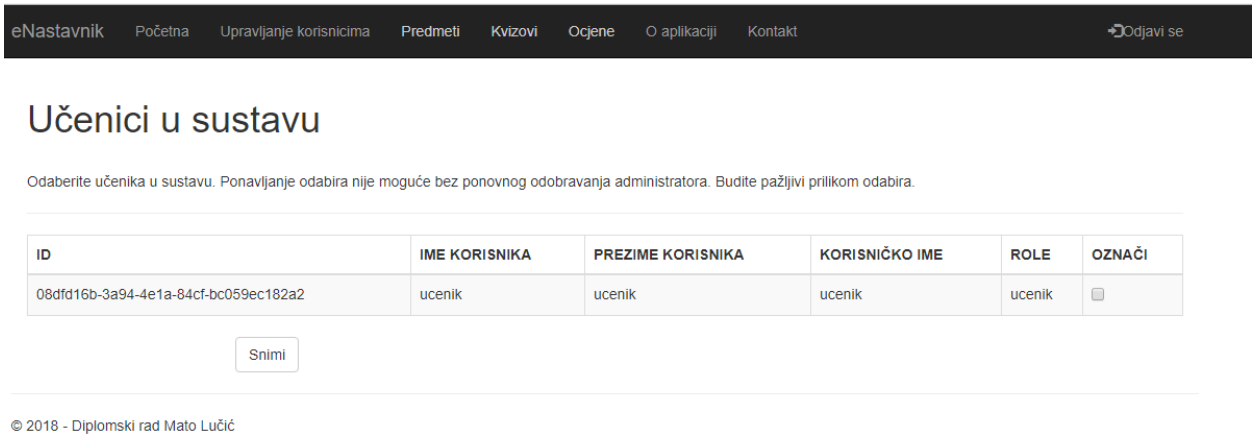


Sl 4.6. *Odbijeni korisnik*

Drugim riječima, ako administrator odabere „DA“, stupac „POTVRDA“ prelazi u „Da“ te korisnik prelazi u kategoriju „POTVRĐENO“. Prije ulaska u kategoriju „POTVRĐENO“ potrebno je dodijeliti ulogu korisniku (**Sl. 4.7.**). Ako se unese uloga „roditelj“ potrebno je roditelju označiti koje korisnike uloge „ucenici“ oni mogu pratiti tj. vidjeti im ocjene (**Sl. 4.8.**).

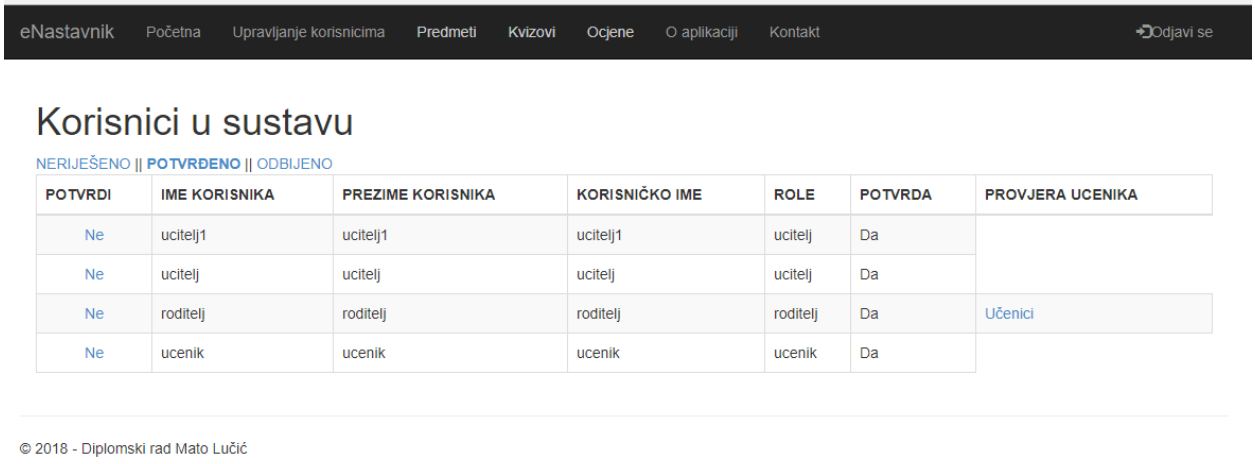


Sl 4.7. *Administrator dodjeljuje ulogu korisniku*



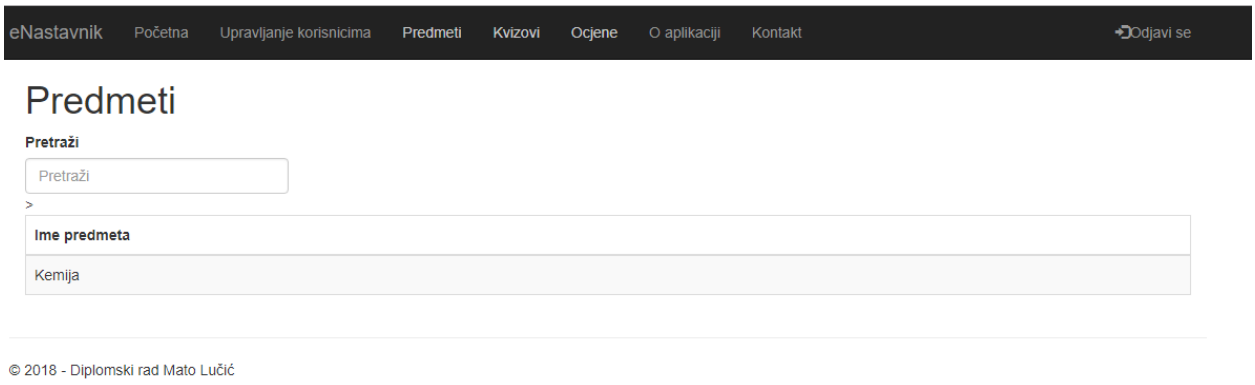
Sl. 4.8. *Administrator dodjeljuje korisnike ROLE – „ucenik“, korisniku ROLE-„roditelj“*

Korisnik koji ima određenu ulogu i potvrdu označenu s „DA“ može se prijaviti u sustav i koristiti sustav s obzirom na ulogu koja mu je dodijeljena. Ako je korisnik u kategoriji „POTVRĐENO“ i u stupcu „ROLE“ mu je „roditelj“ on ima poseban stupac u tablici „PROVJERA UCENIKA“ (**Sl. 4.9.**). Klikom na „Učenici“ mogu se vidjeti svi korisnici kojima on može vidjeti ocjene.



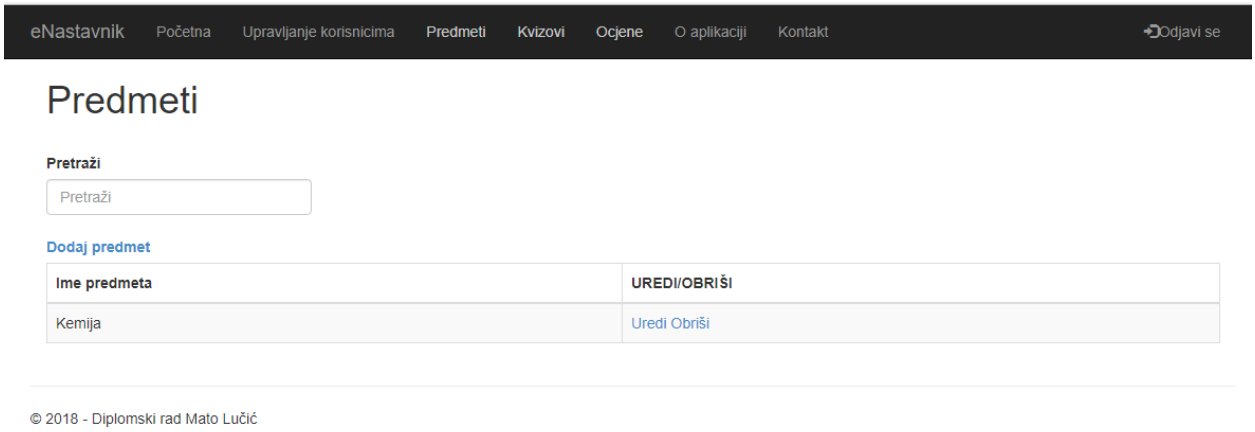
Sl. 4.9. *Korisnici u sustavu sa „POTVRDOM“ – „Da“ i određenom „ROLE“*

Korisnici uloge „ucitelj“, osim mogućnosti upravljanja korisnicima, upravljaju i predmetima (omogućen im je unos predmeta). Na alatnoj traci u padajućem izborniku „Predmeti“ nalaze se dva linka „Pregled predmeta“ i „Upravljanje predmetima“. Kikom na „Pregled predmeta“ aplikacija otvara stranicu kao na slici **Sl. 4.10.**

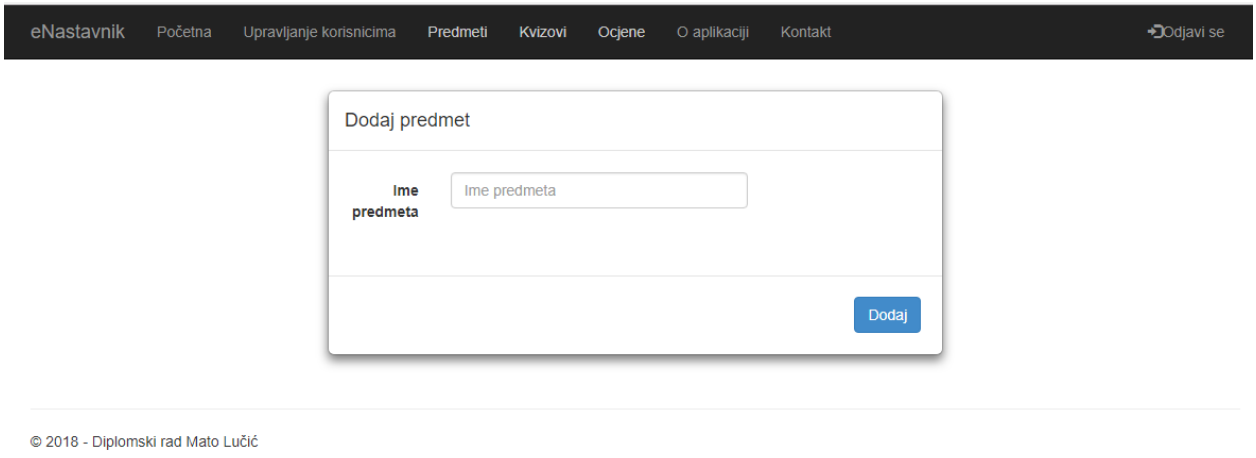


Sl. 4.10. *Pregled predmeta*

Ovaj prozor omogućuje samo uvid u sve predmete u sustavu. Klikom na „Upravljanje predmetima“ (**Sl. 4.11.**) otvara se mogućnost dodavanja predmeta u sustav (**Sl. 4.12.**).



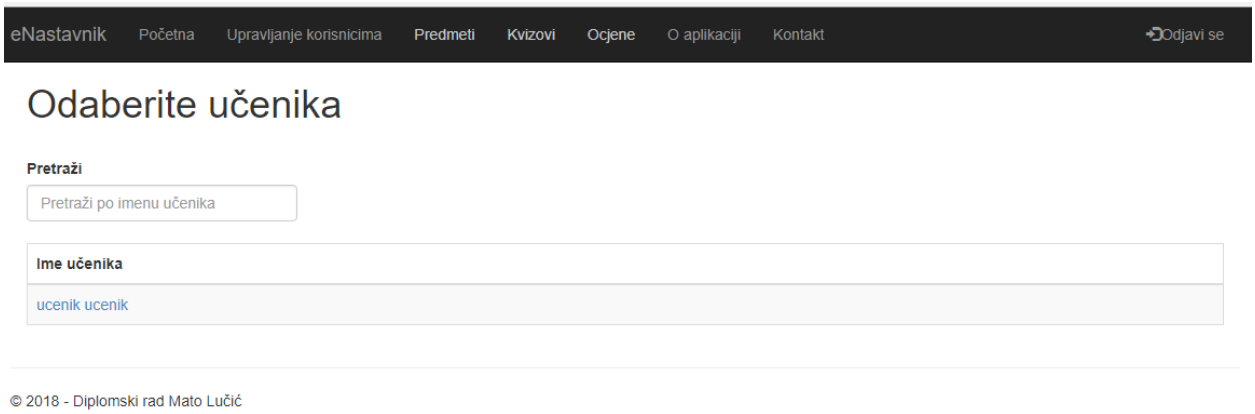
Sl. 4.11. *Upravljanje predmetima*



Sl. 4.12. *Dodavanje predmeta*

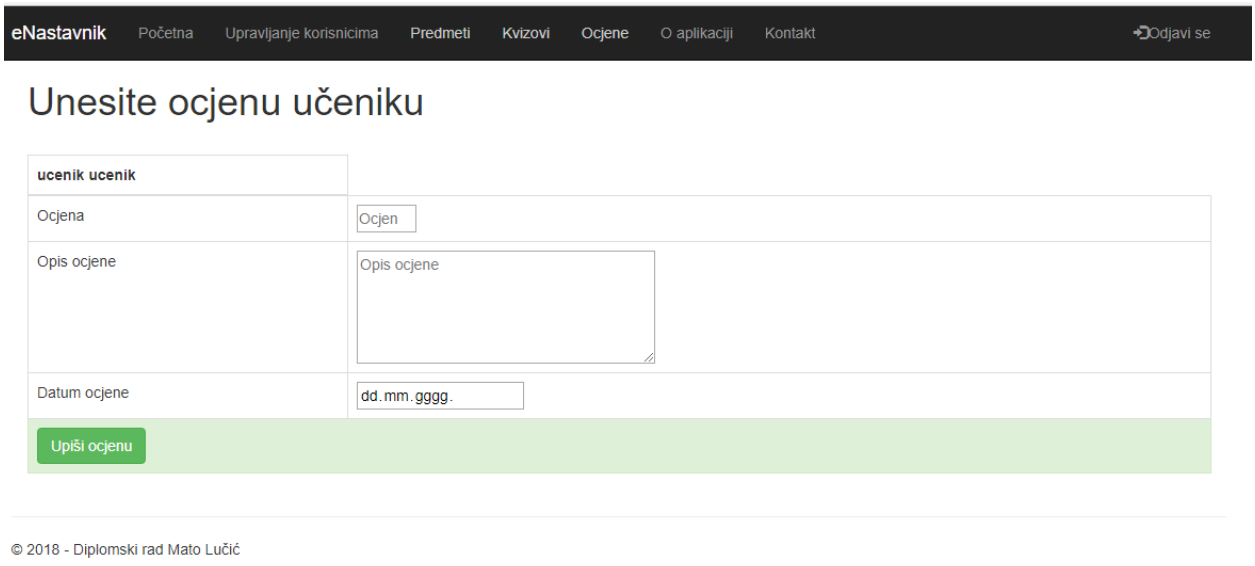
U rubriku „Ime predmeta“ upisuju se predmeti koji se žele dodati u sustav. Klikom na „Dodaj“, predmet se sprema u bazu. Unutar stupca „UREDI/OBRIŠI“ (**Sl. 4.11.**), klikom na „Uredi“, omogućen je ponovni unos imena predmeta u slučaju da je predmet pogrešno nazvan. Ako je potrebno trajno ukloniti predmet iz baze, radnja se izvršava klikom na gumb „Obriši“, koji se nalazi u tablici pored svakog predmeta. Nakon svake radnje ispisuje se poruka o uspješnosti.

Administriranjem učenika i dodavanjem predmeta omogućen je unos ocjena preko alatne trake klikom na „Ocjene“. U padajućem izborniku moguće je odabrati linkove „Dodaj ocjenu učeniku“, „Pregled ocjena“, „Obriši unos“. Klikom na „Dodaj ocjenu učeniku“ otvara se tablica sa svim učenicima u sustavu (**Sl. 4.13.**). Odabirom učenika otvara se tablica za odabir predmeta (**Sl. 4.14.**), a odabirom predmeta otvara se stranica za unos ocjene učeniku (**Sl. 4.15.**).



Sl 4.13. *Učenici u sustavu*

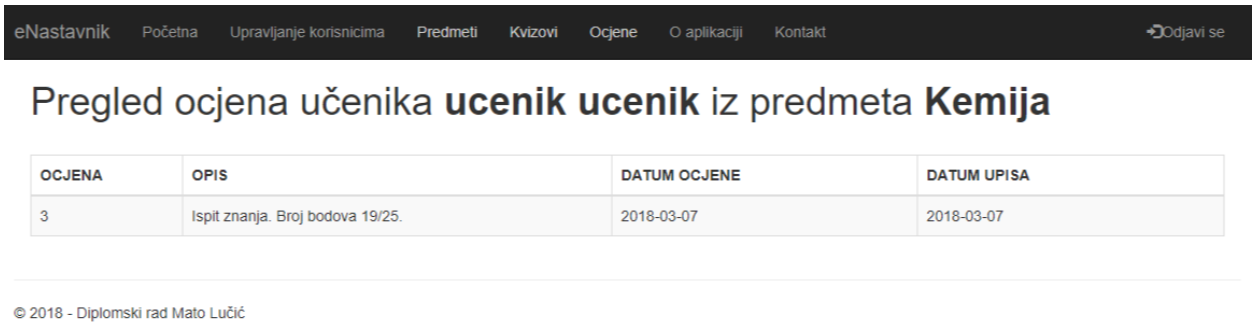
Sl 4.14. *Predmeti u sustavu*



Sl. 4.15. *Forma za unos ocjene*

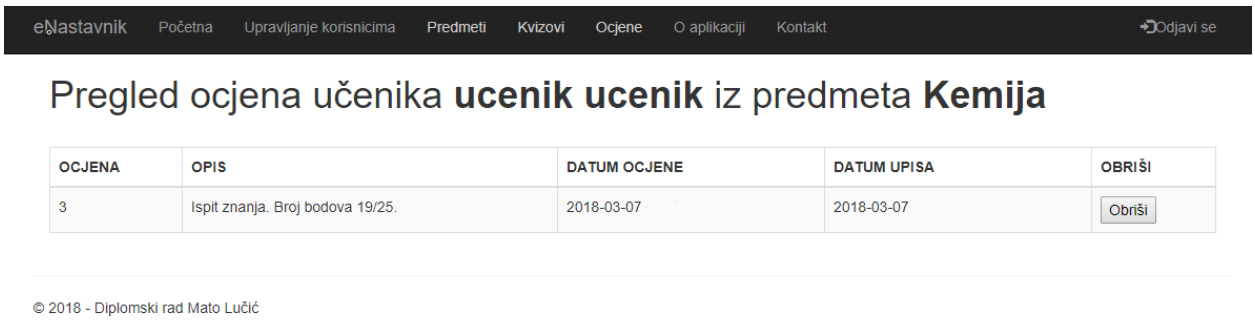
U rubriku „Ocjena“ unosimo ocjenu u brojčanom obliku u rasponu od 1-5. Niti jedan drugi unos nije moguć. Iduća rubrika je „Opis ocjene“, unutar koje se unosi tekst s objašnjenjem ocjene. Rubrika „Datum ocjene“, prikazuje datum zaslužene ocjene. Klikom na „Upiši ocjenu“ svi podatci se spremaju u bazu popraćeni porukom o uspješnosti.

Odabirom linka „Pregled ocjena“ u padajućem izborniku „Ocjene“ princip je sličan kao na linku „Dodaj ocjenu učeniku“. Dakle, odabere se učenik, zatim predmet koji želimo pogledati, zatim uvid u ocjene (**Sl. 4.16.**).



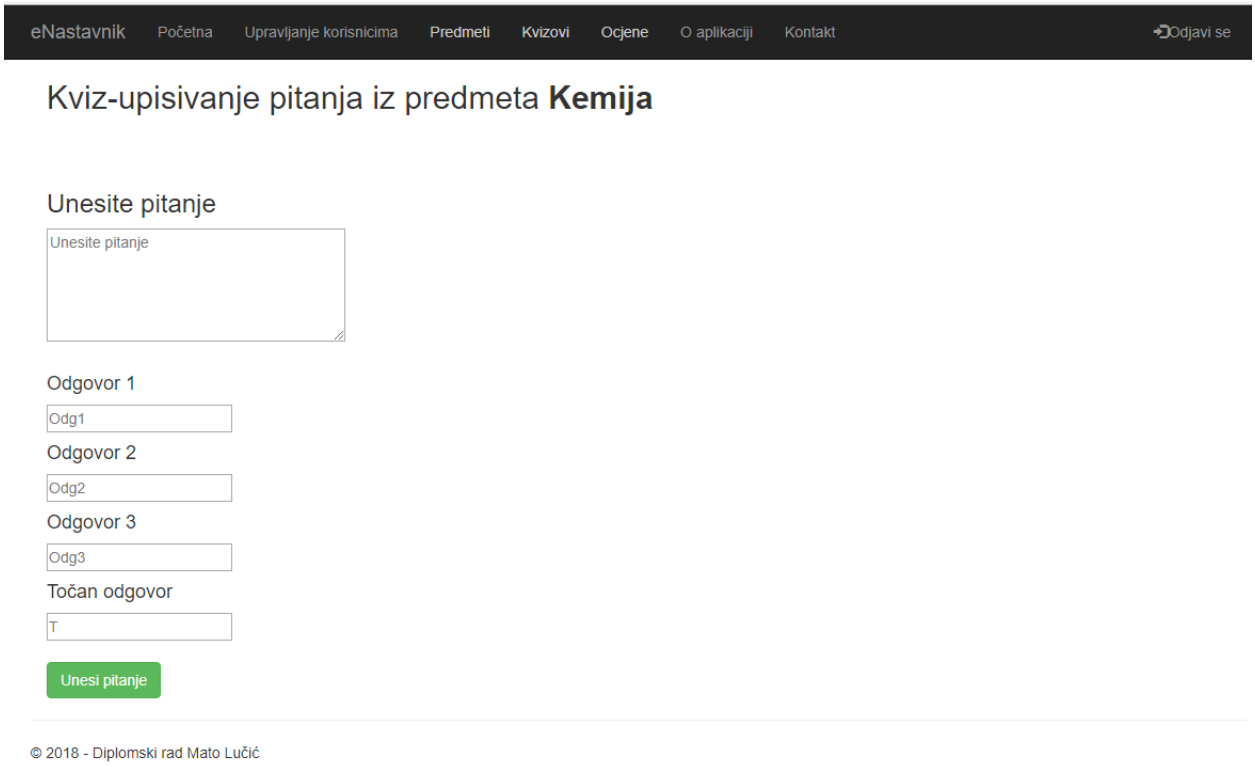
Sl. 4.16. *Pregled ocjena*

Odabirom linka „Obriši unos“ u padajućem izborniku „Ocjene“, korisnik uloge „ucitelj“ može obrisati ocjenu korisniku uloge „ucenik“. Postupak je isti kao na prijašnjem linku, prvo odabere učenika,a zatim predmet. Dakle, na toj stranici ispisane su i vidljive sve ocjene koje odabrani učenik ima iz određenog predmeta. U slučaju da dođe do pogrešnog unosa, u petom stupcu tablice je gumb „Obriši“. Klikom na taj gumb, uklanja se podatak pohranjen u bazu (**Sl. 4.17.**).



Sl. 4.17. *Brisanje ocjena*

Također, postoji još jedna mogućnost korisnika uloge „ucitelj“. Ona se odnosi na unos kvizova, koji se unose preko alatne trake klikom na padajući izbornik „Kvizovi“. Klikom na izbornik, otvaraju se dvije mogućnosti: „Dodaj kviz“ i „Pregled i upravljanje kvizovima“. Link „Dodaj kviz“ otvara stranicu na kojoj je popis svih predmeta, a odabirom predmeta otvara se forma za unos pitanja (**Sl. 4.18.**).



Sl. 4.17. *Forma za unos kviza*

U prvu rubriku u formi, unosi se pitanje iz odabranog predmeta. Ostale rubrike su odgovori („Odgovor 1“, „Odgovor 2“ i „Odgovor 3“) i sadržavaju jedan točan odgovor i dva pogrešna. Zadnja rubrika je ponovljena kao točan odgovor na pitanje.

Klikom na gumb „Unesi pitanje“, sprema se pitanje u bazu i brišu unosi u formi za drugo pitanje, i tako redom. S druge strane, klikom na link „Pregled i upravljanje kvizovima“, otvara se stranica za odabir predmeta iz kojeg želimo pogledati kvizove. Nadalje, odabirom predmeta ispisuje se tablica s kvizom iz predmeta (**Sl. 4.18.**).

Zadnji stupac je nazvan „POPRAVI/OBRIŠI“. Tu nalazimo mogućnost popravka pitanja i odgovora u slučaju da je krivo uneseno, ali i mogućnost brisanja pitanja iz baze podataka.

Sl. 4.17. *Pregled pitanja kviza odabranog predmeta*

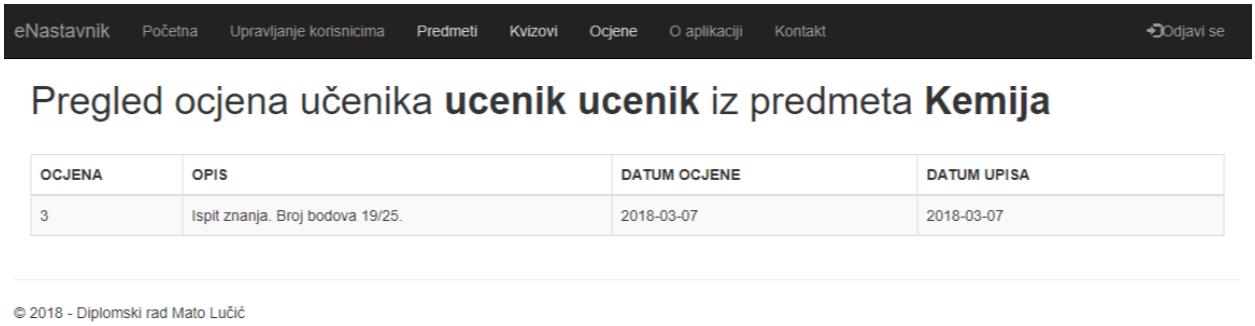
4.2. Prijava kao uloga „ucenik“

Prijavom korisnika u ulozi „ucenik“, korisniku je omogućen pregled ocjena i rješavanje kvizova, koje je omogućio korisnik uloge „ucitelj“ (**Sl. 4.18.**).

Sl. 4.18. *Početna stranica korisnika uloge „ucenik“*

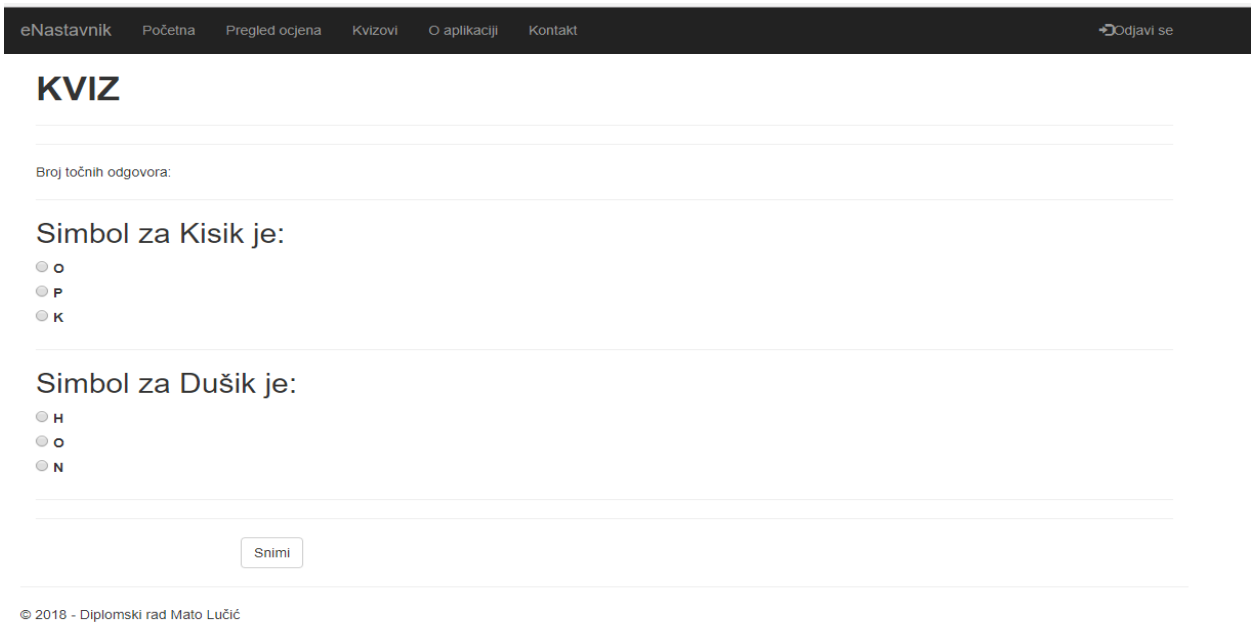
Klikom na poveznicu „Pregled ocjena“ na alatnoj traci, korisniku je omogućen pregled samo vlastitih ocjena, koje mu je unio korisnik uloge „ucitelj“. Također, klikom na nju izlistavaju se

svi predmeti korisnika, a odabirom predmeta za pregled ocjena, moguće je izlistati sve ocjene iz odabranog predmeta. Na slici **Sl. 4.19.** prikazan je pregled ocjena učenika „ucenik ucenik“ iz predmeta „Kemija“.



Sl. 4.18. *Pregled ocjena*

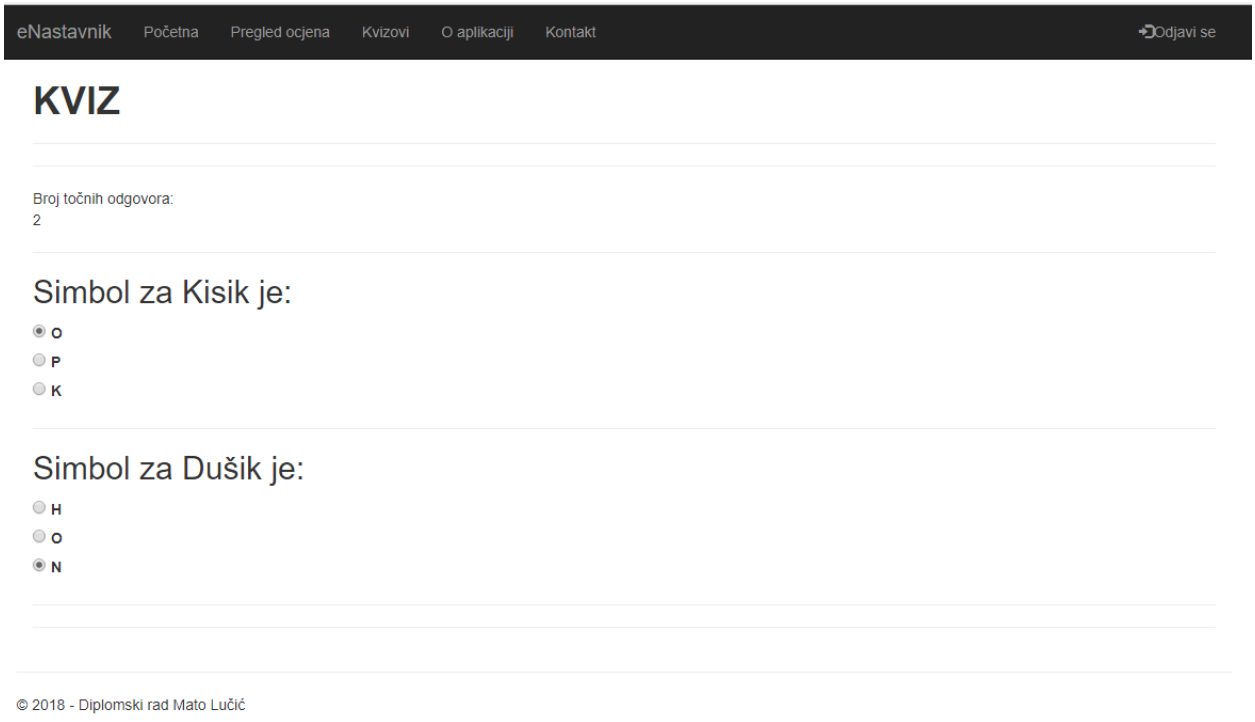
Klikom na poveznicu „Kvizovi“, korisniku je omogućeno rješavanje kvizova. Prvo je potrebno odabrati predmet iz kojeg se kviz rješava, a zatim rješavanje može započeti. Na slici **Sl. 4.19.** je prikazan kviz iz predmeta „Kemija“.



Sl. 4.18. *Rješavanje kviza*

Kviz je koncipiran tako da se na svako pitanje podijeljeno crtama postavi točkica pored točnog odgovora. Sve dok nije stisnut gumb „Snimi“, postoji mogućnost promjene u odgovoru. Kada se

stisne gumb „Snimi“, kviz se zaključi, gumb nestane, pa se ispiše broj točnih odgovora u rubriku iznad pitanja. (**Sl. 4.19.**)



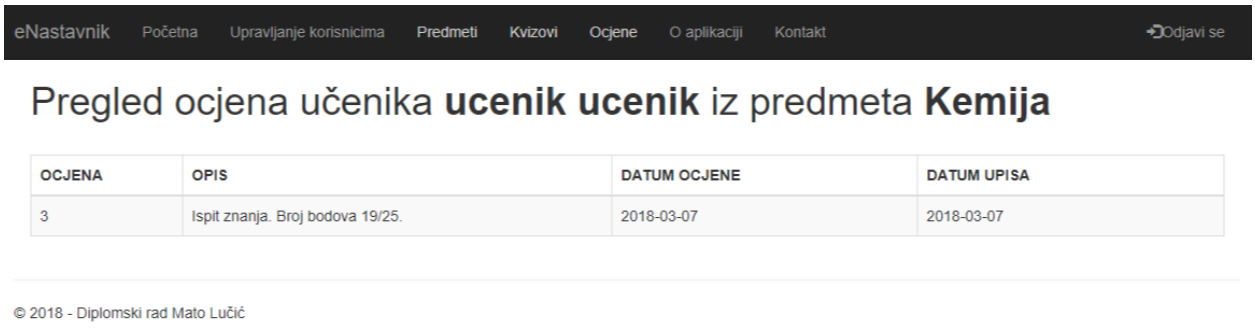
Sl. 4.19. *Klikom na gumb „Spremi“ provjere se odgovori te ispiše broj točnih odgovor*

4.3. Prijava kao uloga „roditelj“

Prijavom uloge „roditelj“, u sustavu je omogućen samo pregled ocjena korisnika uloge „ucenik“, koje mu je korisnik uloge „ucitelj“ dodijelio (**Sl. 4.20.**).

Sl. 4.20. *Početna stranica korisnika uloge „roditelj“*

Zbog mogućnosti da jedan korisnik uloge „roditelj“ ima više djece u sustavu, klikom na „Pregled ocjena učenika“ u alatnoj traci otvara se popis učenika kojima ima pristup i može pogledati ocjene. Odabirom učenika izlistavaju se svi predmeti, a odabirom predmeta ispisuju se ocjene (**Sl. 4.21.**).



Sl. 4.21. *Pregled ocjena*

5. ZAKLJUČAK

Sve većim napretkom tehnologije javlja se potreba, odnosno navika pristupa informacijama na bilo kojem mjestu i u bilo koje vrijeme. Isto tako, kako bi se održao takav standard, potrebno je „uhvatiti korak s vremenom“. U tom smislu došlo je do potrebe za izradom aplikacije „eNastavnik“ koja bi služila kao alat, odnosno kao pomoć u školstvu. Tu se, prije svega, misli na pomoć nastavniku pri obavljanju svakodnevnih obaveza vezanih uz evidenciju i provjeru znanja, ali i na mogućnost da učenik i njegovi roditelji imaju konstantan uvid u njegov napredak.

Za izradu spomenute aplikacije, bilo je potrebno proučiti programske jezike i alate: C#, ASP.NET, AngularJS, HTML, JavaScript, CSS i SQL. Također, za implementaciju navedenih alata, bilo je potrebno koristiti razvojno okruženje Visual Studio, SQL Managment Studio namijenjen za bazu podataka, te Postman i druge aplikacije koje su služile za implementaciju koda, kontroliranje toka podataka i sl. Isto tako, kako bi aplikacija bila stilski što jednostavnija i smislenija krajnjem korisniku, za većinu uređivanja je korišten framework Bootstrap. Drugim riječima, cijela logika aplikacije odrađena je u dva dijela: Backend i Frontend. Backend komunicira s bazom i prenosi podatke na Frontend, koji preuzima podatke te ih šalje na korisničko sučelje.

Aplikacija „eNastavnik“ je razvijena u web tehnologijama i funkcionira kao web stranica postavljena na server, kojoj je omogućen pristup samo onim korisnicima koji polažu prava na nju. Pokretanjem aplikacije otvara se pogled za prijavu u nju. U slučaju da korisnik nije ranije pristupao aplikaciji, mora se registrirati. Stiskom na gumb „Registriraj se“, prebačen je na pogled za registraciju na kojem je forma koja zahtjeva unos određenih podataka. Nadalje, nastavnik je glavni administrator i njemu su dane sve ovlasti nad aplikacijom, odnosno dodijeljena mu je uloga „ucitelj“. Uloga „ucitelj“ može upravljati svim ostalim ulogama u sustavu (može ih obrisati ili promijeniti), kao i dodavati predmete, kvizove i ocjene. Osim njega pristup aplikaciji u manjoj mjeri ima i učenik, preko dodijeljene uloge „ucenik“, koji, kada i ako mu to administrator odobri, može vidjeti ocjene iz predmeta koji mu je dodan i rješavati kvizove da provjeri stečeno znanje. Isto tako, pristup ima i roditelj, preko dodijeljene uloge „roditelj“, kojem je omogućen uvid u ocjene samo onog korisnika uloge „ucenik“ koji mu je dodijelila uloga „ucitelj“.

Sve u svemu, aplikacija „eNastavnik“ je realizirana kroz tri uloge u sustavu i nastala je kako bi olakšali nastavniku vođenje evidencije o učenikovom uspjehu i napretku kroz razdoblje

njegovog školovanja, ali i kako bi učenik mogao rješavati kvizove, odnosno provjerio i uvježbao gradivo, ali i zajedno roditeljima imao uvid u vlastiti uspjeh.

Na kraju, razvoj ove aplikacije je moguć i može se ostvariti dodavanjem novih mogućnosti. Primjerice, učenicima se može omogućiti uvid u povijest riješenih kvizova, kao i evidencije bodova koje su ostvarivali rješavajući kvizove. Također, u nekom daljnjem nadograđivanju aplikacije, može se postaviti da nastavnik može vidjeti sve predmete, ali uređivati samo one predmete koji su mu dozvoljeni, kao i dodavati ocjene samo onim učenicima koji su mu određeni. To su samo neki od mogućih primjera koji neće dopustit aplikaciji da zastari, nego ostane u „toku s vremenom“, odnosno s potrebama sadašnjega, kao i budućeg društva.

LITERATURA

[1] http://www.manuelradovanovic.com/2015/11/uvod-u-c-programski-jezik.html, datum pristupa 8.3.2018.

[2] http://www.tutorijali.net/teorija-programiranja/varijable, datum pristupa 15.3.2018

[3] https://web.math.pmf.unizg.hr/nastava/rp2/pred3/pred3.html , datum pristupa: 8.3.2018.

[4] http://web.zpr.fer.hr/ergonomija/2004/bosnjakm/tipovi.html , datum pristupa: 8.3.2018.

[5] https://abcsharp.wordpress.com/2016/03/15/tipovi-podataka-u-programskom-jeziku-c/ , datum pristupa: 8.3.2018.

[6] https://podrska.avalon.hr/knowledgebase/144/Sto-je-ASPNET.html , datum pristupa: 8.3.2018.

[7] https://marketplace.visualstudio.com/items?itemName=NuGetTeam.NuGetPackageManager , datum pristupa: 8.3.2018.]

[8] http://www.mono.hr/Pdf/Dependency-Injection-in-practice-CodeCAMP.pdf , datum pristupa: 8.3.2018.

[9] https://www.info-novitas.hr/tehnologija/orm/ , datum pristupa: 8.3.2018

[10] https://www.popwebdesign.net/popart_blog/2015/05/angularjs-framework/ , datum pristupa: 8.3.2018.

[11] http://app.eva-sms.com/claroline/claroline/backends/download.php?url=L3NlbWluYXJza2lfcmFkb3ZpLzEyX1NRTF9pX3N0YW5kYXJkaV96YV9wcmlzdHVwX2JhemFtYV9wb2RhdGFrYV9HcmFzaWNaZWxpbWlyLnBkZg%3D%3D&cidReset=true&cidReq=MP1516 , datum pristupa: 8.3.2018.

SAŽETAK

Suvremeno doba obilježeno je brzim protokom informacija, koji sa sobom nosi sve veći napredak u svim sferama života, a poglavito u onim koje se tiču tehnoloških ostvarenja. To podrazumijeva konstantnu potrebu za novim informacijama, ali isto tako i kontinuirani uvid u iste. U ovom diplomskom radu prikazana je razrada i implementacija aplikacije „eNastavnik“, odnosno aplikacije koja se nudi kao jedno od rješenja na spomenute, nove zahtjeve društva. Naime, aplikacija „eNastavnik“ razvijena je kao web rješenje koje olakšava posao nastavniku tako da zamjenjuje „ručni“ način vođenja evidencije o uspjehu i pruža mogućnost kreiranja kvizova koji imaju odliku provjere usvojenog gradiva. U tom kontekstu, odnosno kako bi se održao određeni današnji standard, u izradi ove aplikacije korištene su suvremene moderne programske tehnologije. Glavni jezici korišteni za izradu su: C# i JavaScript sa svojim framework nadopunama, dok sama aplikacija za spremanje podataka koristi SQL bazu podataka. Logika aplikacije je odrađena u dva dijela: Backend i Frontend. Drugim riječima, u ovom radu je opisana izrada i reprezentacija aplikacije koja omogućava kontinuiran uvid u napredak svakog učenika u smislu pristupa aplikaciji kroz dodijeljene/podijeljene tri uloge: „ucenik“, „roditelj“ i „ucitelj“. Svaka od uloga ima različitu mogućnost pristupa određenim podacima od kojih se ponajviše ističe uloga „ucitelj“ koja ima administratorska prava u sustavu.

KLJUČNE RIJEČI

eNastavnik, uloga, web aplikacija, ucenik, nastavnik, roditelj, ocjena, kviz.

TITLE

eTEACHER

ABSTRACT

The modern era is marked by a rapid flow of information, which carries ever greater progress in all spheres (aspects) of life, especially in terms of technological achievements. This implies a constant need for new information, but also a continuous insight in the same. This thesis shows the elaboration and implementation of the “eNastavnik” application, i.e. the application that is offered as one of the solutions to the mentioned, new demands of the society. Namely, the “eNastavnik” application has been developed as a web solution that makes the job of a teacher easier by replacing a "manual" way of keeping track of success and providing the possibility to create quizzes that have a feature to check the learnt content. In this context, i.e. in order to maintain a certain contemporary standard, modern software technologies have been used in creating this application. The main programming languages used for creating are: C # and JavaScript with their framework additions, while the application itself uses the SQL database to store the data.The application logic is done in two parts: Backend and Frontend. In other words, this thesis describes the creation and representation of the application that provides a continuous insight into the progress of each student in terms of accessing the application through assigned / divided three roles: "student", "parent" and "teacher". Each of the roles has different access to certain data, most of which is attributed to the "teacher" role that has administrator permission in the system.

KEY WORDS:

eTeacher, role, web application, student, teacher, parent, grade, kviz.

ŽIVOTOPIS

Mato Lučić je rođen u Slavonskom Brodu, živi u Gundincima u ulici Stjepana Radića 68. Osnovnu školu je završio u Gundincima te se odlučio za daljnje školovanje u Slavonskom Brodu, upisavši Gimnaziju „Matija Mesić“. Završio je preddiplomski studij računarstva na Fakultetu elektrotehnike računarstva i informacijskih tehnologija u Osijeku. Nakon preddiplomskog studija upisuje diplomski studij računarstva, izborni blok Programsko inžinjerstvo. Radnog iskustva nema. Dobar je poznavalac engleskog jezika. U slobodno vrijeme bavi se izradom aplikacija i glazbom.

_____

MATO LUČIĆ

PRILOZI

Na DVD-u koji se prilaže uz dokumentaciju nalaze se:

Dokumenti:

eNastavnik_dokumentacija_Lučić_Mato.doc

eNastavnik_dokumentacija_Lučić_Mato.pdf

Datoteke:

Cjelokupni projekt s kodom

Slike

ELEKTRONIČKA VERZIJA DIPLOMSKOG RADA NA CD-u ILI DVD-u