ELEKTRONSKA EVIDENCIJA STUDENATA

Bumbak, Ivan

Undergraduate thesis / Završni rad

2016

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Polytechnic of Sibenik / Veleučilište u Šibeniku**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:143:853410

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-07-27**

Repository / Repozitorij:

VUS REPOSITORY - Repozitorij završnih radova Veleučilišta u Šibeniku

VELEUČILIŠTE U ŠIBENIKU

ODJEL MENADŽMENTA

STRUČNI STUDIJ INFORMATIČKI MENADŽMENT

Ivan Bumbak

ELEKTRONSKA EVIDENCIJA STUDENATA

Završni rad

Šibenik, 2016.

VELEUČILIŠTE U ŠIBENIKU

ODJEL MENADŽMENTA

STRUČNI STUDIJ INFORMATIČKI MENADŽMENT

ELEKTRONSKA EVIDENCIJA STUDENATA

Završni rad

Kolegij: Baze podataka

Mentor: dr.sc. Frane Urem

Student: Ivan Bumbak

Matični broj studenta: 0023096262

Šibenik, 2016.

Sadržaj

1. Uvod 1

2. Baze podataka 2

2.1. Povijest baza podataka 3

2.2. SQL 5

3. ER dijagram baze podataka elektronske evidencije studenata 8

4. Baza podataka elektronske evidencije studenata 9

5. MVC 12

6. Kreiranje MVC aplikacije 14

7. Dizajniranje MVC aplikacije 18

8. Kreiranje modela 20

9. Kreiranje kontrolera 29

10. View 34

11. Zaključak 35

Popis literature 36

Popis slika 37

1. Uvod

Obrazovanje je jedan od najvažnijih segmenata ljudskog života. Razvijanje, informatizacija i ulaganje u obrazovanje povećava kvalitetu samog obrazovanja, ali njegovu efikasnost. S boljom kvalitetom i boljom informatizacijom kako učenici i studenti, tako i profesori ulažu veći trud te im je angažman za rad veći. S većim, boljim i produktivnijim radom raste i broj nazočnih studenata na nastavi, stoga je potrebna i informatizacija evidencije studenata, profesora, predmeta, smjerova itd. Elektronskom evidencijom studenata na nastavi skraćuje se vrijeme potrebno za prozivanje i provjeravanje nazočnosti studenata. Osim toga, sprječava se i kolegijalno potpisivanje kolega koji nisu nazočni na nastavi, jer svaki student ima svoj ID, autentifikacijski kod ili će iskoristit svoju studentsku karticu, odnosno x-icu kako bi se mogao elektronski potpisati.

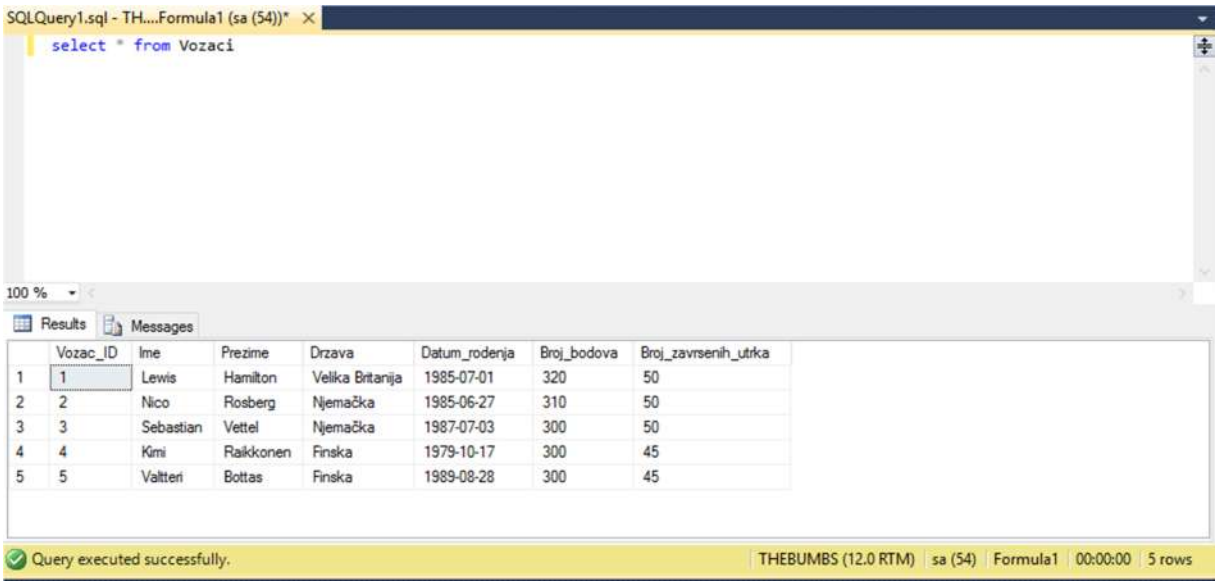
U prvom dijelu rada objašnjavaju se osnove baza podataka, njihov razvoj te gdje i za što se koriste. U ostatku rada se prikazuje izrada aplikacije, za elektronsku evidenciju studenata. Uz navedeni programski kod, prikazuju se alati i metode koje su korišteni pri izradi aplikativnog rješenja.

2. Baze podataka

Baze podataka su sustavno organizirani ili strukturirani repozitorij indeksiranih podataka, obično kao grupa povezanih datoteka podataka koje nam omogućuju jednostavno pronalaženje, praćenje i analizu podataka. Ovakvi podatci su uglavnom spremljeni u računalima u obliku tablica, skripata, grafičkih oblika, teksta i drugo, predstavljajući skoro sve vrste informacija. Većina računalnih programa, poput antivirusnog programa sadrže baze podataka u svojim jezgrama. [1]

Database Management System (DBMS) je računalni software[2] pomoću kojeg korisnik, drugi računalni program ili baza podataka dohvaća i analizira podatke. Osnovna uloga DBMS-ova je omogućivanje korisniku kreiranje, ažuriranje i upravljanje određenom bazom podataka. Primjerice, studentska referada ima pristup bazi podataka svih upisanih studenata na fakultetu, te ju mogu ažurirati, spremati nove podatke ili pak brisati nevažeće podatke o studentima. Najpoznatiji DBMS-ovi su: *MySQL, PostgreSQL, Microsoft SQL Server, Oracle, Sybase, SAP HANA i IBM DB2*. Računalni jezik za izradu, traženje, ažuriranje, dohvaćanje i brisanje podataka iz baza podataka naziva se SQL (*engl. Structured Query Language*).

Slika 1. Primjer dohvaćanje podataka iz baze Formula1



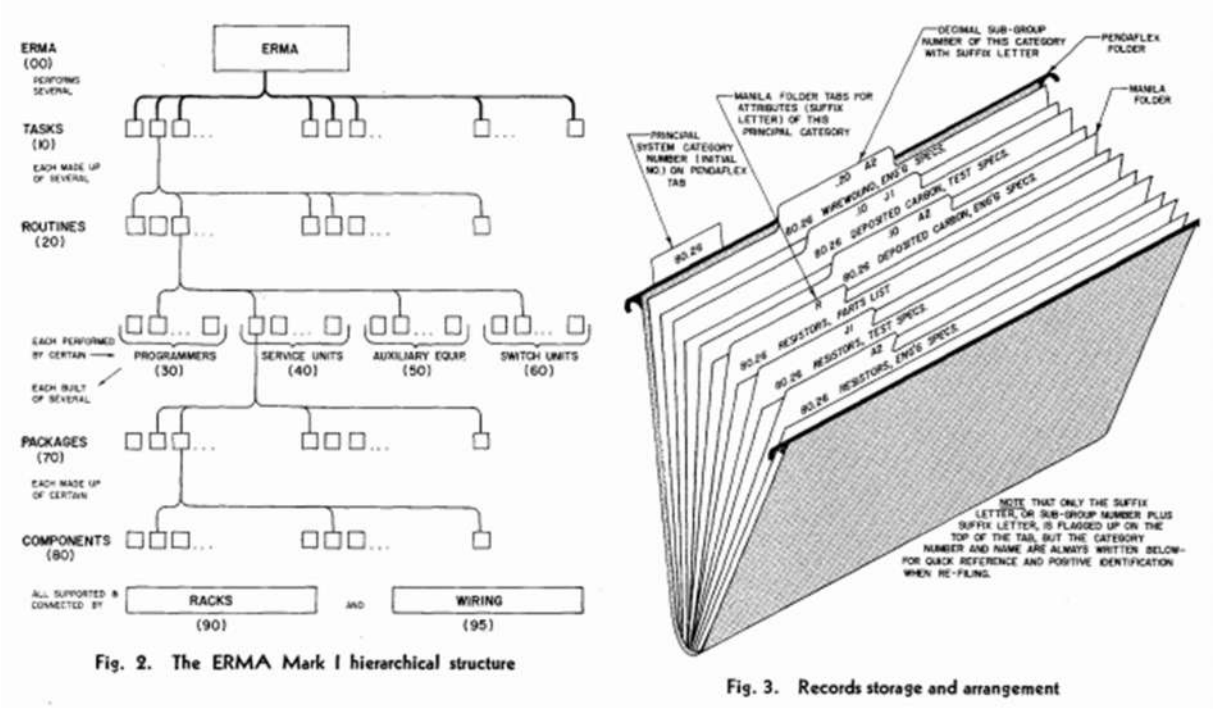
	Vozac_ID	Ime	Prezime	Drzava	Datum_rodenja	Broj_bodova	Broj_zavrsenih_utrka
1	1	Lewis	Hamilton	Velika Britanija	1985-07-01	320	50
2	2	Nico	Rosberg	Njemačka	1985-06-27	310	50
3	3	Sebastian	Vettel	Njemačka	1987-07-03	300	50
4	4	Kimi	Raikkonen	Finska	1979-10-17	300	45
5	5	Valtteri	Bottas	Finska	1989-08-28	300	45

[1] Definicija - http://www.businessdictionary.com/definition/database.html
[2] Software – Dio računalnog sistema koji sadrži enkodirane infomracije ili računalne naredbe

2.1. Povijest baza podataka

Još od davnih dana ljudska bića prikupljaju i spremaju informacije. U davnim vremenima baze podataka nisu bile digitalizirane te su ih kreirali vladini dužnosnici, knjižnice, bolnice, ali i poslovne organizacije. Osnovni principi povijesnih baznih podataka koriste se još i danas.

Slika 2. Organizacija povijesnih baza podataka



Izvor: http://avant.org/project/history-of-databases/#1

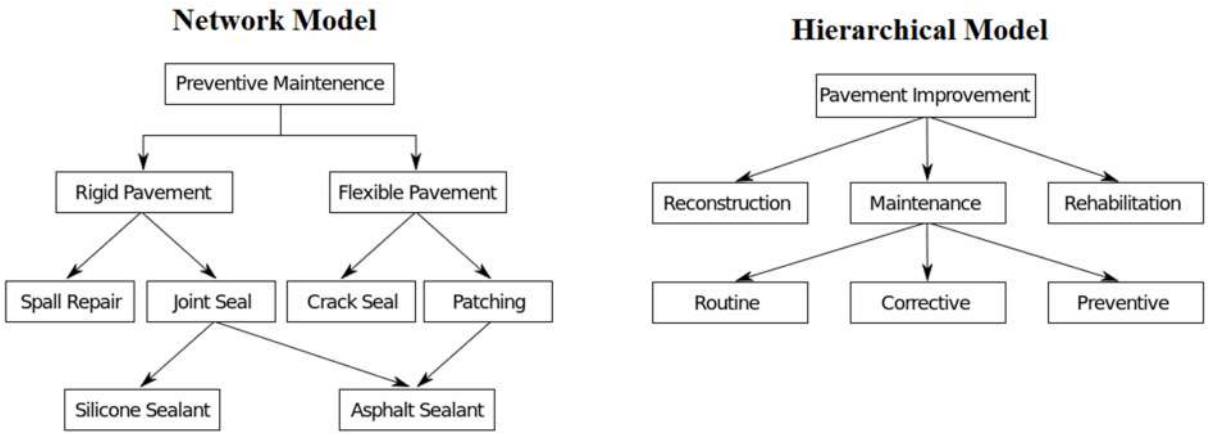
Baze podataka se počinju digitalizirati u 1960.-ima kada upotreba računalima u privatnim organizacijama postaje isplativa. U ovim godinama prevladavaju dva popularna modela baza podataka:

- Mrežni model CODASYL[3] - model podataka koji je zamišljen kao fleksibilan način prikaza objekata i njihovih odnosa
- Hijerarhijski model IMS – model podataka koji je organizirana kao struktura stabla

Osim ova dva modela, SABRE sistem se dokazao kao uspješan komercijalni sistem baza podataka. SABRE sistem kojeg je koristio IBM kako bi pomogao *American Airlines* kompaniji da lakše upravljanju svojim rezervacijama letova.

[3] COSASYL – konferencija za podatkovne sistemske jezike

Slika 3. Prikaz mrežnog modela i hijerarhijskog modela podataka



U 1970.-ima Edgar Frank Codd je objavio rad u kojem preporučuje uporabu relacijskih baza podataka. Relacijska baza podataka je tip baza podataka kod kojeg se organizacija podataka zasniva na relacijskom modelu. Podaci se u ovakvim bazama organiziraju u skup tablica, odnosno entiteta, između kojih se definiraju određene veze. Entiteti su dijelovi stvarnog ili apstraktnog svijeta koji je opisan određenim brojem sredstava koje predstavljamo podatcima. Podatci koji opisuju entitet nazivaju se atributi. Na primjer, entitet „Student“ može imati sljedeće atribute: OIB, ime, prezime, datum rođenja, itd. Da bi se između entiteta mogla ostvariti određena veza svaki od njih treba imati primarni ključ preko kojeg se identificiraju svi podatci u relacijskom entitetu. Razlikujemo 3 vrste relacijskih veza: „1 na 1“, „1 na više“ i „više na više“. Pri relacijskoj vezi više na više nastaje nova među-tablica. U među-tablici primarni ključevi povezanih entiteta nazivaju se strani ključevi (FK). Ovakav model je vrlo brzo prihvaćen te je postao standard svih baza podataka.

Osim relacijskog modela baza podataka, Peter Chen je predstavio tzv. ER (*engl. Entity-Relationship*) model baza podatka. ER model je omogućio kreatorima da se fokusiraju na podatke budućeg aplikativnog rješenja, a ne na logičku strukturu tablica. Tokom 1980.-ih godina relacijske baze podataka postale su komercijalni uspjeh kao i brz rast prodaje računala. Sa sve uspješnijim relacijskim modelom baze podataka, razvio se SQL programski jezik koji je postao standardni jezik za izradu, modificiranje i upravljanje bazama podataka. Također, IBM razvija svoju bazu podataka DB2 koja je postala vodeća svjetska baza podataka, ali i najava IBM-ovih računala. U 1990.-ima razvijaju se novi alati za upravljanje bazama podataka, poput *Oracle Developer-a, PowerBuilder-a, Microsoft Excel-a i Access-a*.

Tokom 1990.-ih godina dolazi do pojave Interneta što je dovelo do eksponencijalnog rasta industrije baze podataka. Prosječni korisnici stolnih računala počeli su koristiti klijentske servere baza podataka kako bi pristupili računalnim sustavima koji su uglavnom sadržavali ostavštine podataka. Osim klijentskih servera, zbog povećanih ulaganja u internetsko poslovanje nastala je potreba za uporabom internetskih baza podataka, poput *Front Page-a, Dream Weaver-a, Enterprise Java Beans-a, Oracle Developer-a 2000,* itd.

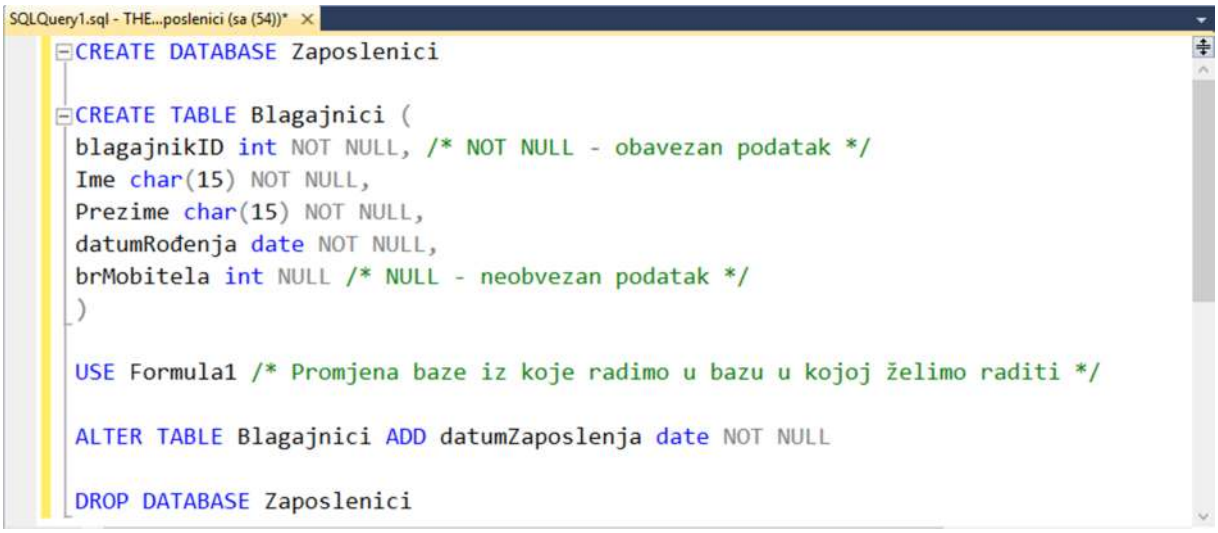
U današnje vrijeme gotovo svaki uređaj sadrži svoju bazu podataka i bez njih je nemoguće poslovati. Razvijene su mnoge aplikacije za PDA-ove, POS aparate i slične uređaje. Najpoznatije i vodeće kompanije koje se bave razvojem baza podataka su Microsoft, IBM i Oracle.

2.2. SQL

SQL je najpopularniji računalni jezik za izradu, traženje, ažuriranje i brisanje podataka iz relacijskih baza podataka. SQL možemo razdvojiti u dva glavna podjezika. *Data Definition Language* (DDL) koji sadrži naredbe koje se koriste za kreiranje i uništavanje baza podataka i njihovih objekata. Učestale naredbe koje se definiraju DDL podjezikom su:

- CREATE DATABASE *imeBaze* – naredba koja kreira bazu podataka pod nazivom *imeBaze*
- CREATE TABLE *imeTablice* (*nazivStupca1* int, *nazivStupca2* char(30)) – naredba koja kreira tablicu, u postojećoj bazi *imeBaze* pod nazivom *imeTablice* sa dva podatka (stupca): *nazivStupca1* koji sadrži integer tip podatka i *nazivStupca2* koji sadrži character tip podatka koji može spremiti do 30 znakova
- USE – naredba koja omogućava korisniku vrlo jednostavno promjene specifične baze podataka na kojoj već vrši promjene u bazu podataka na kojoj želi vršiti promjene
- ALTER – naredba koja omogućava promjene na već izrađenoj tablici ili stupcu bez da dotičnu tablicu ili stupac brišemo ili ponovo izrađujemo
- DROP – naredba koja omogućava brisanje cjelokupnih objekata određene baze podataka, ili cijele baze iz DBMS-a kojeg koristimo

Slika 4. Primjer naredbi definiranih DDL podjezikom



Nakon što je struktura baze podataka definirana sa DDL podjezikom, administratori ili korisnici koriste drugi podjezik SQL-a, *Data Manipulation Language* (DML). Korisnici DML podjezik koriste kako bi mogli unijeti podatke, ažurirati ih ili modificirati.

Učestale naredbe koje se definiraju DML podjezikom su:

- INSERT – naredba koja omogućava korisniku upisivanje podataka u već postojeću tablicu, primjerice kada u bazi *Zaposlenici* želimo dodati novog radnika u tablicu *Blagajnici*
- SELECT – naredba koja omogućava korisniku baze podataka da na vrlo brz i jednostavan način dohvati potrebne podatke, također ona je jedna od najčešće korištenih SQL naredba
- UPDATE – naredba koju korisnik koristi kada želi modificirati nekakav podataka iz neke postojeće tablice, primjerice, ako se zaposlenicima svake godine plaća povisuje za 3%
- DELETE – naredba kojom korisnici brišu podatke iz tablica

Slika 5. Primjer naredbi definiranih DML podjezikom

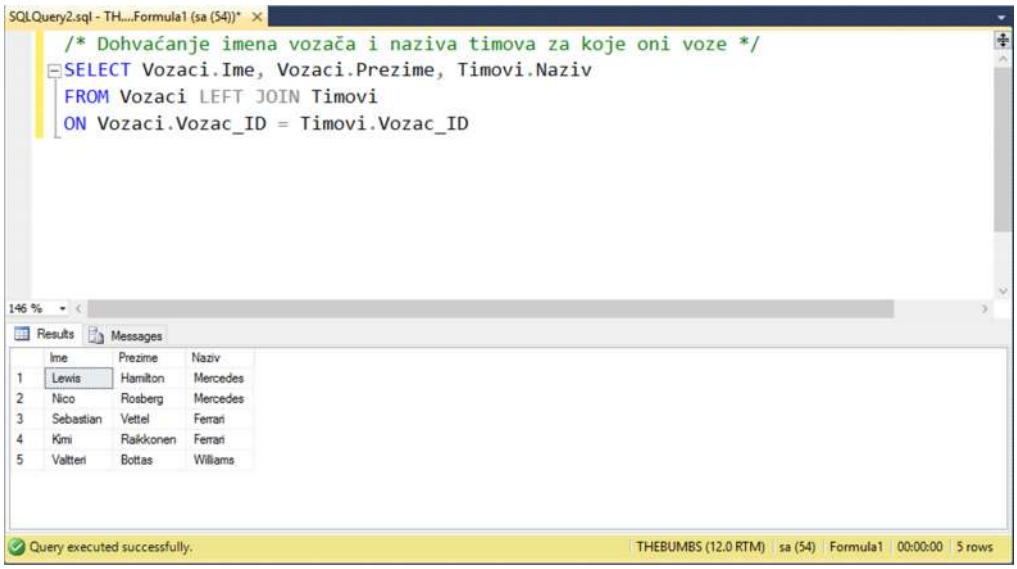
```
SQLQuery2.sql - THE...poslenici (sa (54))*
/* Dohvaćanje svih podataka iz tablice Blagajnici */
SELECT * FROM Blagajnici

/* blagajnikID, Ime, Prezime, datumRođenja, brMobitela, datumZaposlenja, Plaća  */
INSERT INTO Blagajnici
VALUES (1234, 'Pero', 'Perić', '1980/01/01', 0901234567, '2000/04/01', $2500)

/* Promjena broja mobitela blagajnika */
UPDATE Blagajnici
SET brMobitela = 0909876543
WHERE brMobitela = 0901234567

/* Brisanje podatka iz tablice Blagajnici čija je šifra/ID 1234 */
DELETE FROM Blagajnici
WHERE blagajnikID = 1234
```

Osim navedenih osnovnih naredbi postoji i napredna naredba JOIN. Pomoću naredbe JOIN korisnici mogu povezivati podatke iz dvije ili više tablica u jedan jedinstveni podatak. Postoje 4 vrste JOIN naredbe:

- INNER JOIN – dohvaća sve podatke kada postoji barem jedan spoj u obje tablice
- LEFT JOIN – dohvaća sve podatke iz lijeve tablice i njihove parove iz desne tablice
- RIGHT JOIN – dohvaća sve podatke iz desne tablice i njihove parove iz lijeve tablice
- FULL JOIN – dohvaća sve podatke kada postoji par u jednoj od tablica

Kombinirajući osnovne naredbe DML podjezika i naredbu JOIN na vrlo jednostavan način se može doći do bilo kojeg podatka.

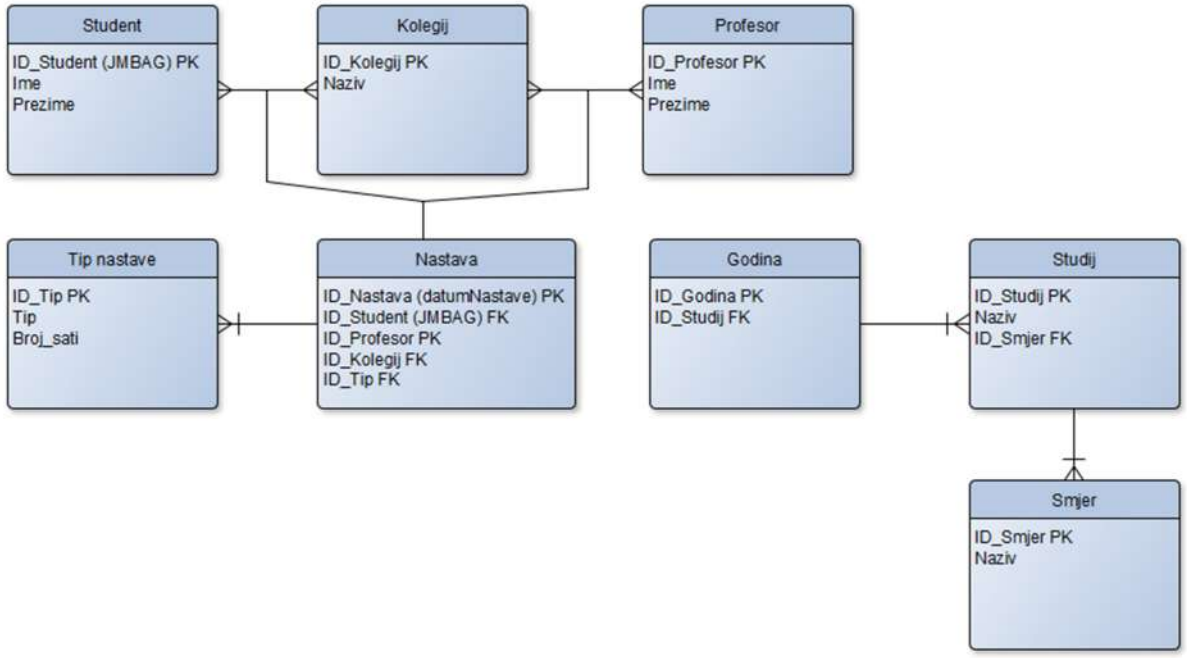
Slika 6. Primjer JOIN naredbe



3. ER dijagram baze podataka elektronske evidencije studenata

Prije same izrade baze podataka za aplikaciju elektronske evidencije studenata preporučljivo je izraditi ER dijagram pridružene baze podataka. Za izradu ER dijagrama korišten je program *yED Grpah Editor* proizveden od tvrtke *yWorks* koja se specijalizirala u izradi programskih rješenja za jednostavnu vizualizaciju dijagrama i mreža. Uredan i pravilno izrađen dijagram daje pregledan uvid u sve entitete, atribute te same relacijske veze između entiteta.

Slika 7. ER dijagram baze podataka informatizirane evidencije

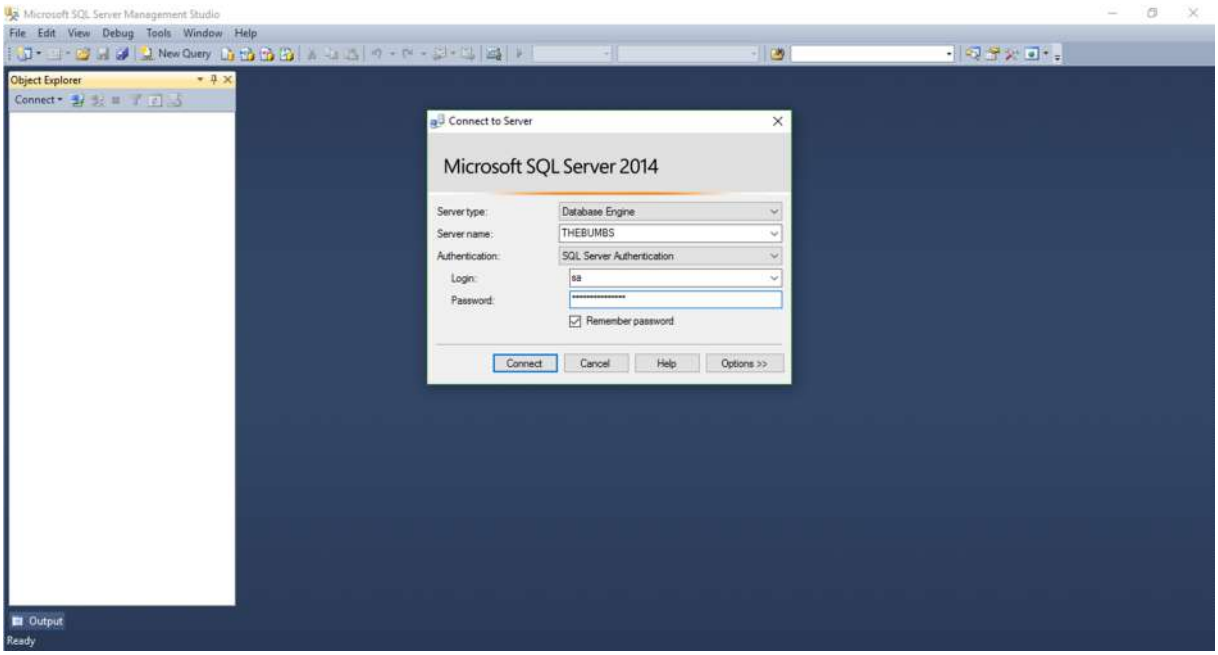


Na slici 7. je prikazan ER dijagram baze podataka informatizirane evidencije studenata, profesora, kolegija itd. Iz dijagrama se može zaključiti kako je za problematiku elektronske evidencije potrebno 8 entiteta: *Student, Kolegij, Profesor, Tip nastave, Nastava, Godina, Studij* i *Smjer*. Svaki entitet je opisan svojim atributima. Između pojedinih entiteta se mogu iščitati njihove relacijske veze. Tako je npr. entitet *Nastava* rezultat relacijskih veza „više na više“ između entiteta *Student, Kolegij* i *Profesor*. Svaki od ta tri entiteta ima svoj primarni ključ (PK) preko kojeg se povezuje na entitet *Nastava*. Entitet *Nastava* koji predstavlja međutablicu sastoji se od stranih ključeva (FK) povezanih entiteta. Entitet *Godina* povezan je na entitet *Studij* relacijom „1 na više“, jer 1 godina može imati više studija. Situacija je slična i kod relacijske veze između entiteta *Studij* i *Smjer*. Jedan studij može sadržavati više smjerova.

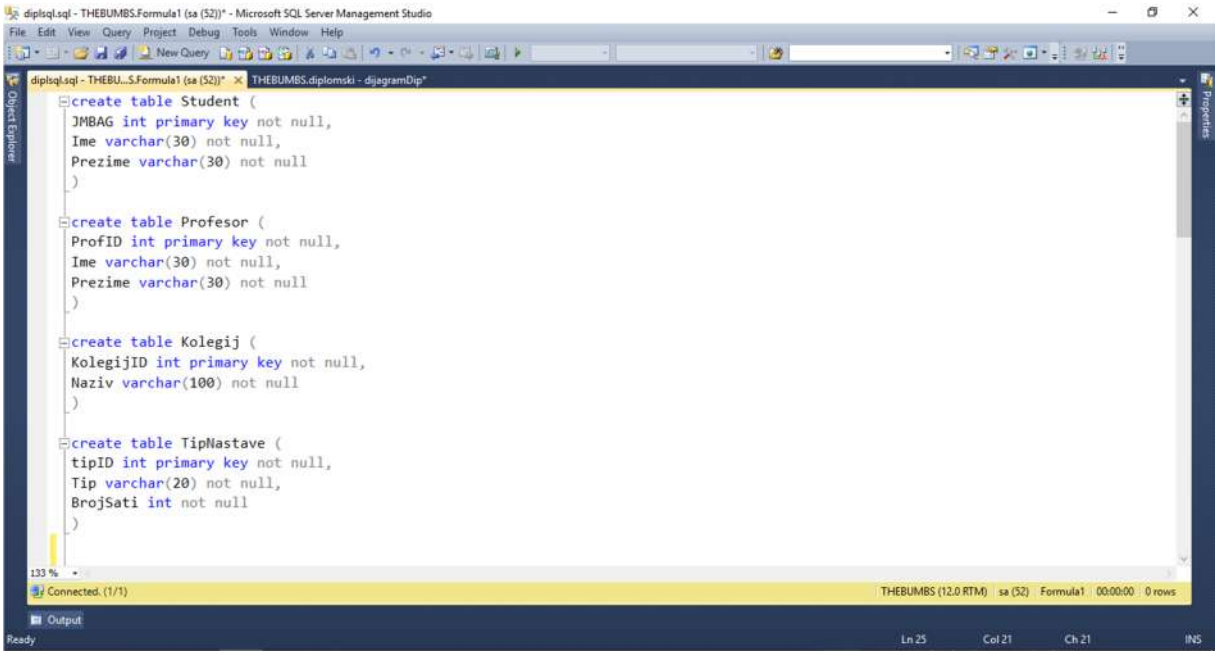
4. Baza podataka elektronske evidencije studenata

Nakon uspješne izrade ER dijagrama moguće je izraditi fizički model baze podataka. Opisana baza podataka elektronske evidencije studenata je izrađena u programu *Microsoft SQL Server Management Studio 2014.*

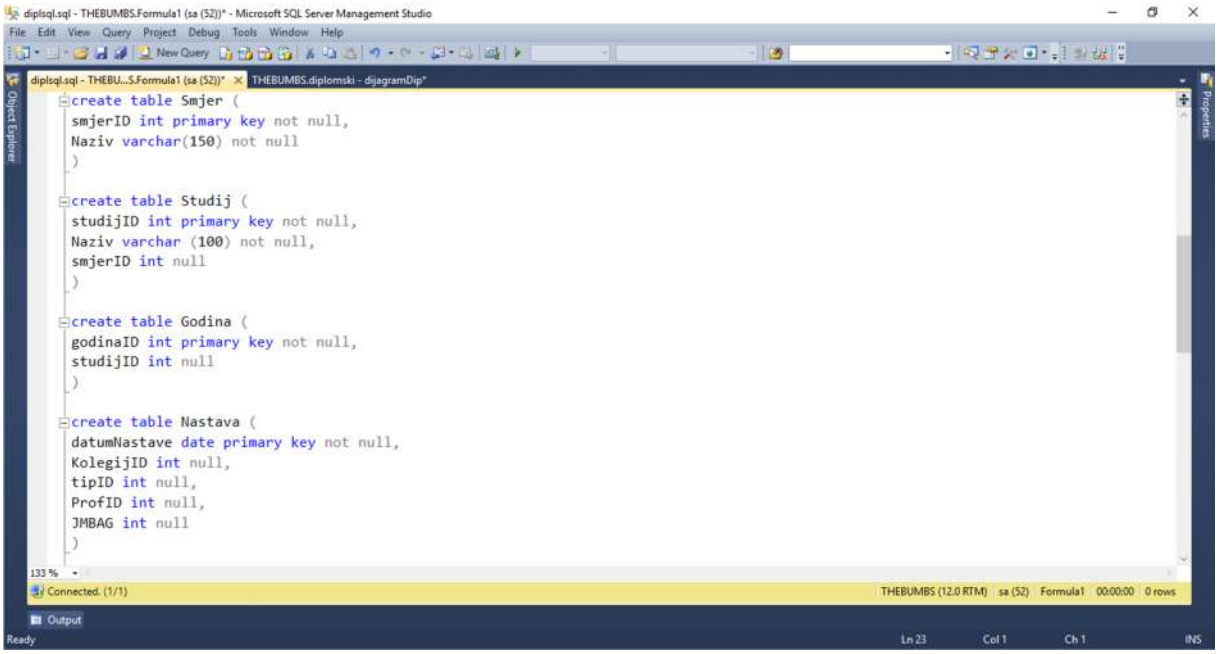
Slika 8. Prikaz spajanja na lokalni Microsoft SQL Server 2014



Slika 9. Kreiranje tablica

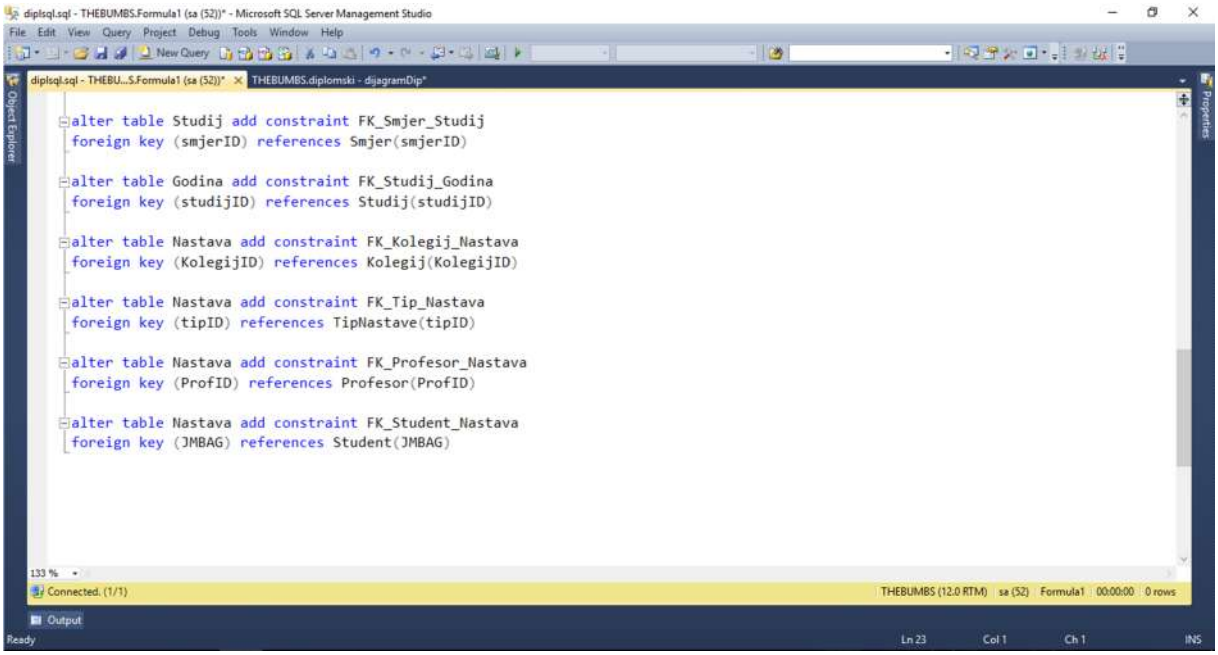


Slika 10. Kreiranje tablica



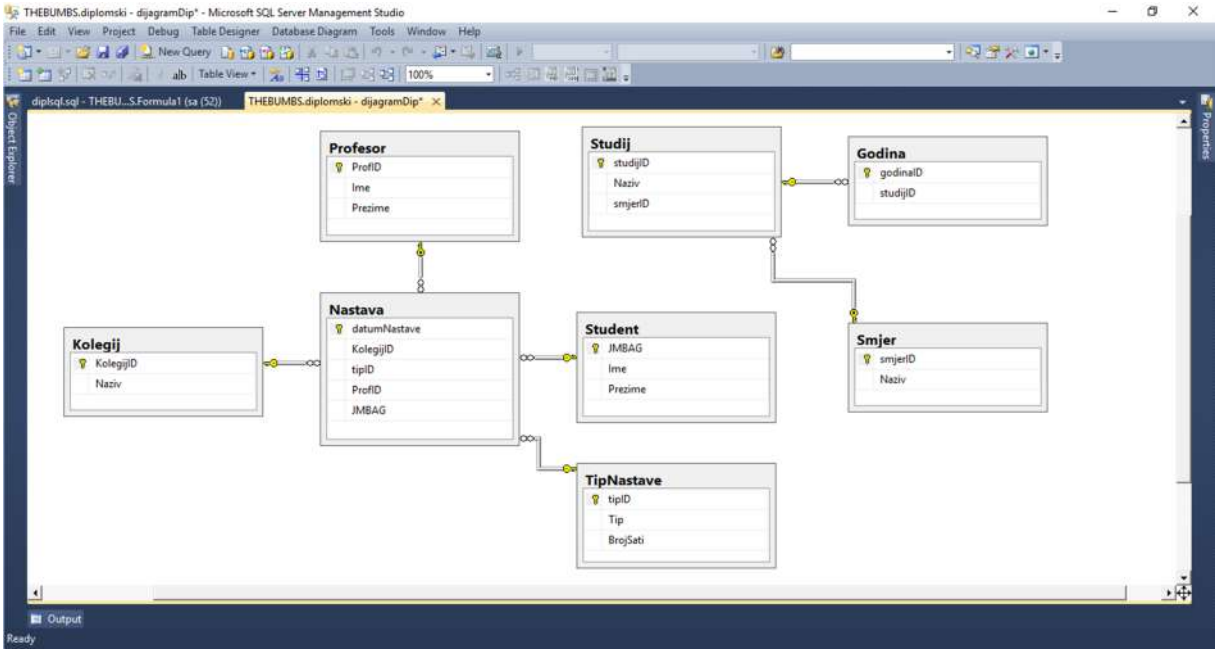
Na slici 9. i na slici 10. prikazan je SQL programski kod kojim su izrađene tablice u bazi podataka. Tablice u bazi podataka predstavljaju entitete koji su prethodno opisani ER dijagramom u prethodnom poglavlju. Može se uočiti kako svaka naredba CREATE TABLE sadrži stupce koji predstavljaju atribute svakog entiteta. Također, svaki entitet sadrži primarni ključ. Nakon izrade svih tablica potrebno je stvoriti strane ključeve kako bi se tablice mogle relacijski povezati. Slika 11. prikazuje SQL kod kojim se kreiraju strani ključevi na potrebnim mjestima.

Slika 11. Kreiranje stranih ključeva



Nakon što smo stvorili primarne ključeve možemo generirati dijagram baze podataka. Na slici 12. je prikazan dijagram kojeg će *Microsoft SQL Server Management Studio* po našem zahtjevu i odabiru tablica automatski generirati. Iz grafa se može iščitati koja tablica i preko kojeg ključa je povezana na drugu tablicu.

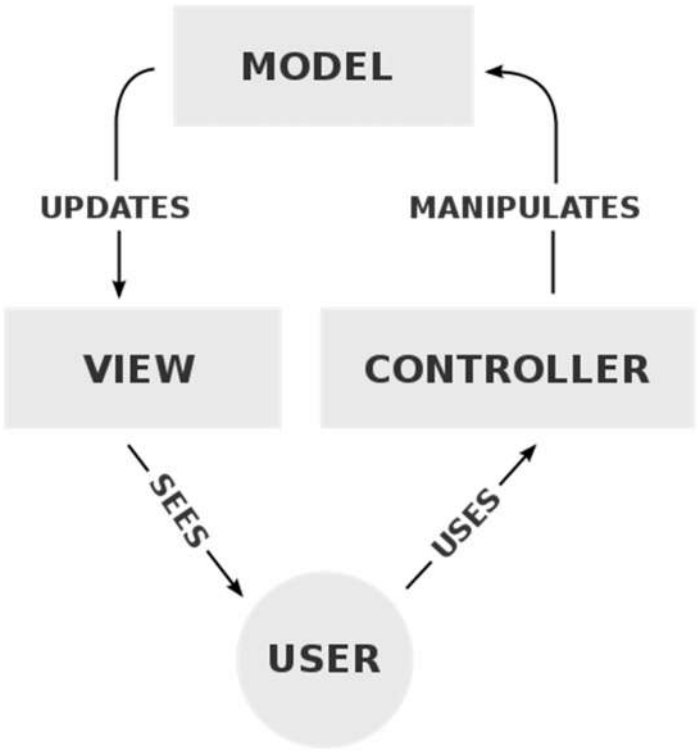
Slika 12. Dijagram baze podataka informatizirane evidencije



5. MVC[4]

MVC (*engl. Model-View-Controller*) je obrazac softverske arhitekture, pomoću koje je moguće na vrlo jednostavan način izraditi web sjedište koje pristupa bazama podataka. Osnovna ideja MVC arhitekture je odvajanje pojedinih dijelova aplikacije prema njihovoj namjeni. Osnovni MVC-a su: *Model, View* i *Controller*.

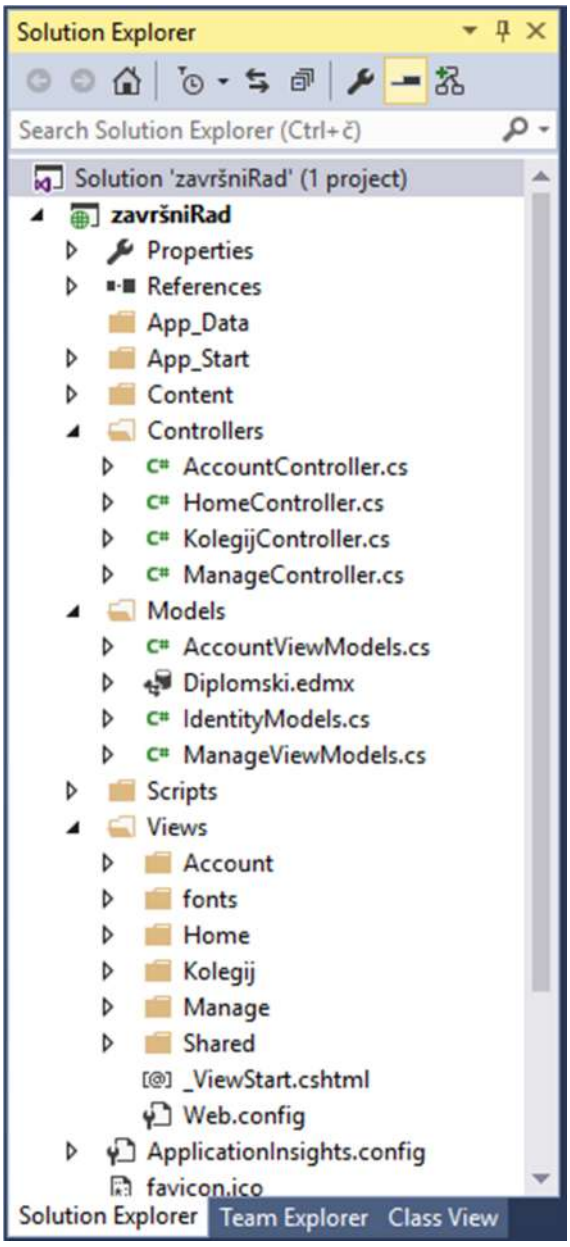
Slika 13. Osnovni dijelovi MVC-a



Model u MVC-u predstavlja podatke te poslovnu logiku aplikacije. Putem njih se pohranjuje i upravlja sa svim podatcima koji se nalaze u aplikaciji. Podatke iz modela moguće je prikazati pomoću *Grid.Mvc* opcije koja podatke slaže u tablice i omogućuje vrlo jednostavno upravljanje, ažuriranje ili brisanje podataka iz tablica. *View* je dio MVC aplikacije koji korisnicima web sadržaja ili aplikacije prikazuje vizualnu sliku HTML (engl. *HyperText Markup Language)*[5] datoteka koje se nalaze u njegovoj mapi. Osim HTML datoteka u mapi *View* se nalaze mape za svaki kreirani *Controller*. *Controller* u MVC aplikaciji je odgovoran za upravljanje korisničkim zahtjevima aplikacije.

[4] MVC – software za implementaciju aplikativnog sučelja na računalima
[5] HTML – standardni programski jezik koji se koristi za kreiranje web sadržaja

Slika 14. Primjer Solution Exproler-a MVC aplikacije

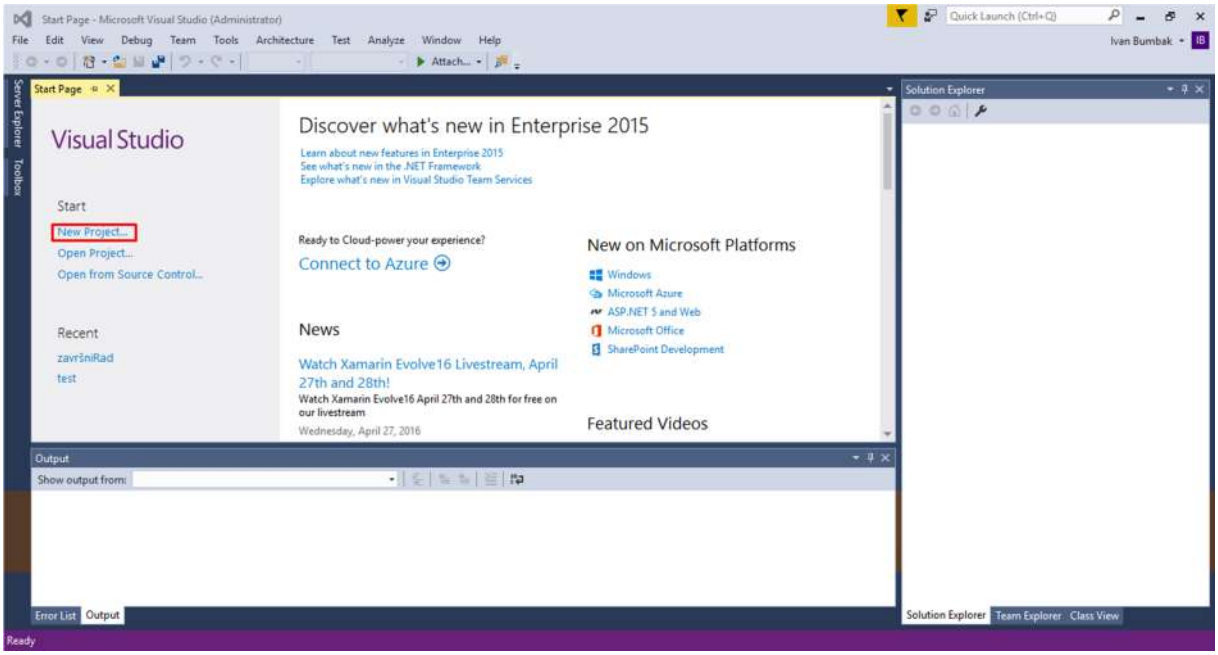


Slika 14. prikazuje *Solution Exproler* aplikacije razvijene za završni rad. Na slici je vidljivo kako se svaki *Controller*: *AccountController.cs, HomeController.cs, KolegijController.cs*, nalazi u mapi *Views*, preko čega se svaki od njih vizualno prikazuje korisnicima web sadržaja ili aplikacije. Također, vidljivo je kako se u mapi *Models* nalazi baza podataka *Diplomski.edmx* u kojoj će se pohraniti svi podatci vezani za studente, profesore, kolegije itd.

6. Kreiranje MVC aplikacije

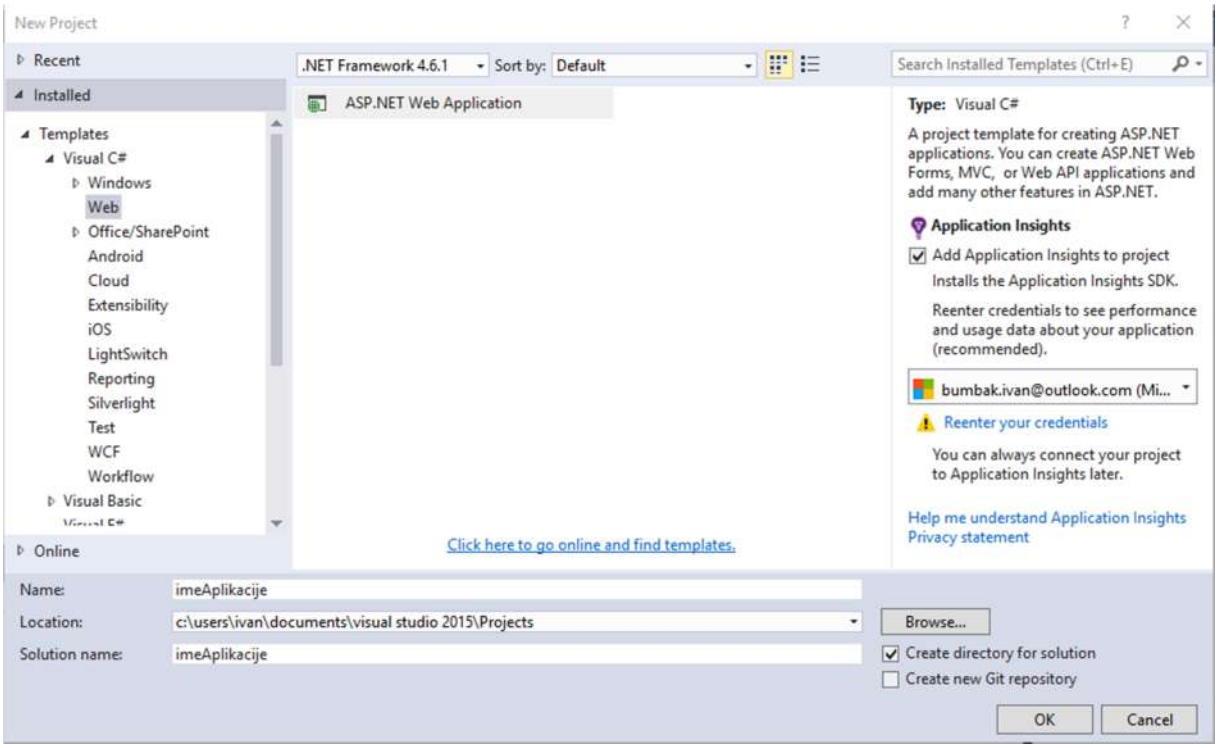
Za kreiranje MVC aplikacije korišten je program *Microsoft Visual Studio Enterprise 2015*.

Slika 15. Prikaz početnog zaslona Microsoft Visual Studio Entreprise 2015



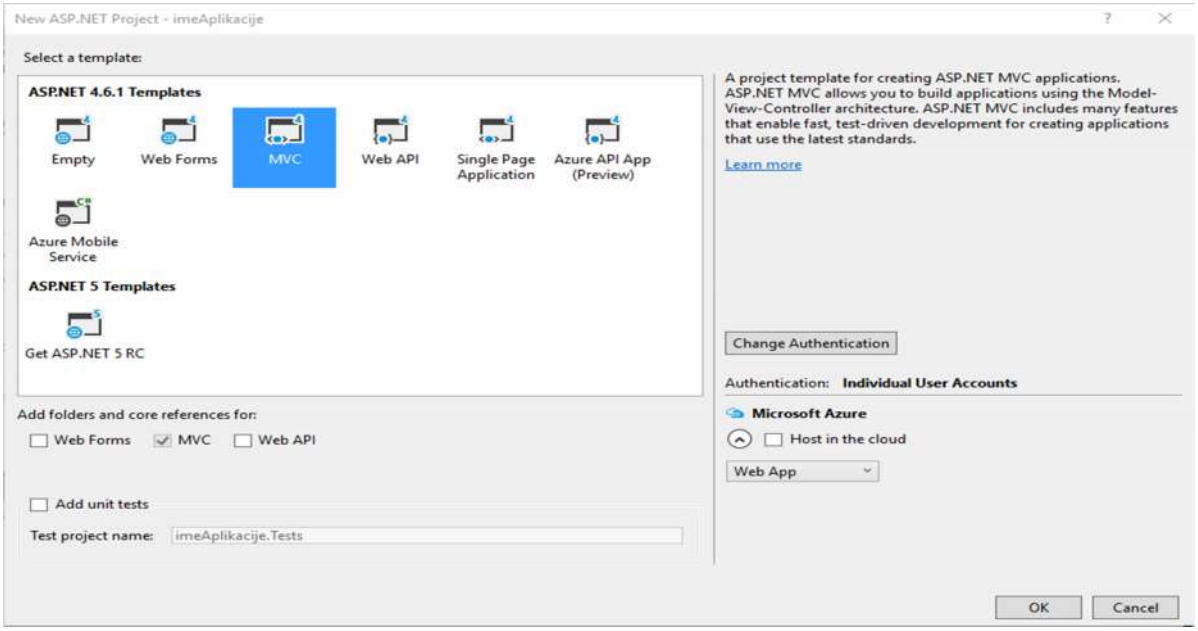
Kako bi se izradio novi projekt potrebno je kliknuti na poveznicu *New Project…* koja se nalazi na lijevoj strani zaslona. Klikom na poveznicu otvara se novi prozor gdje je potrebno odabrati tip aplikacije. U lijevom izborniku odabere se opcija *Web*, a u desnom izborniku odabere se *ASP.NET Web Aplication*. Na dnu prozora upiše se naziv projekta. (slika 16.).

Slika 16. Prikaz odabira ASP.NET Web aplikacije



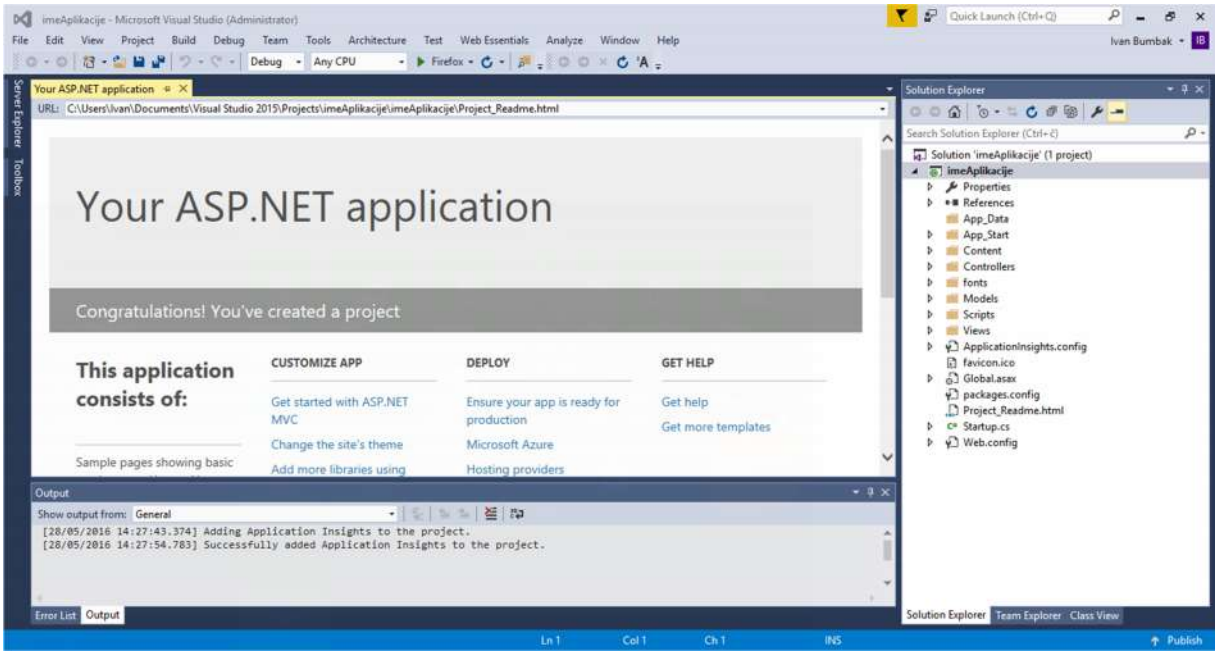
Nakon odabira *ASP.NET Web Application* i određivanja imena projekta otvara se novi prozor sa izborom raznih *Web* aplikacija. Za završni radi odabran je MVC. Prije razvijanja aplikacije, na desnoj strani pritiskom na poveznicu *Change Authentication* može se i odabrati tip autentikacije: *bez autentikacije, individualna autentikacija, poslovna ili školska autentikacija* te *Windows autentikacija*. (slika 17.)

Slika 17. Prikaz odabira MVC aplikacije



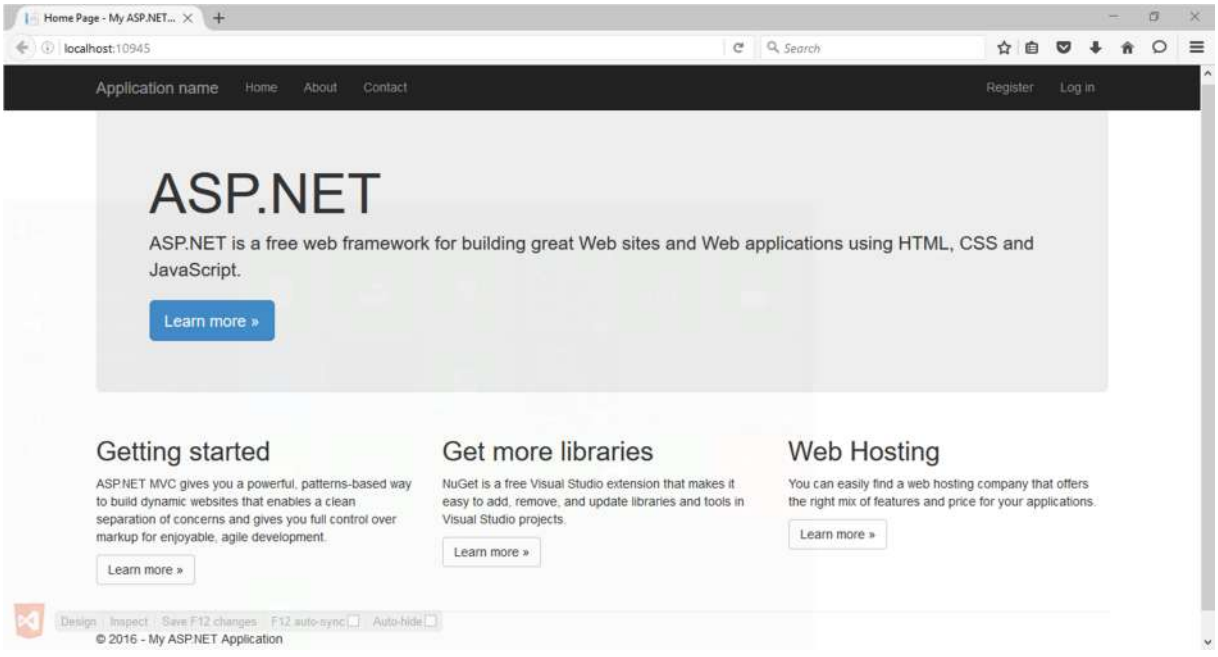
Nakon što softver sam izvrši sve svoje provjere te kreira MVC aplikaciju, otvara se prozor u kojem su zapisane informacije o potvrdi razvijanja i funkcionalnosti MVC aplikacije. Dakle, u nekoliko jednostavnih koraka kreirana je MVC aplikacija.

Slika 18. Prikaz dolaznog prozora MVC aplikacije

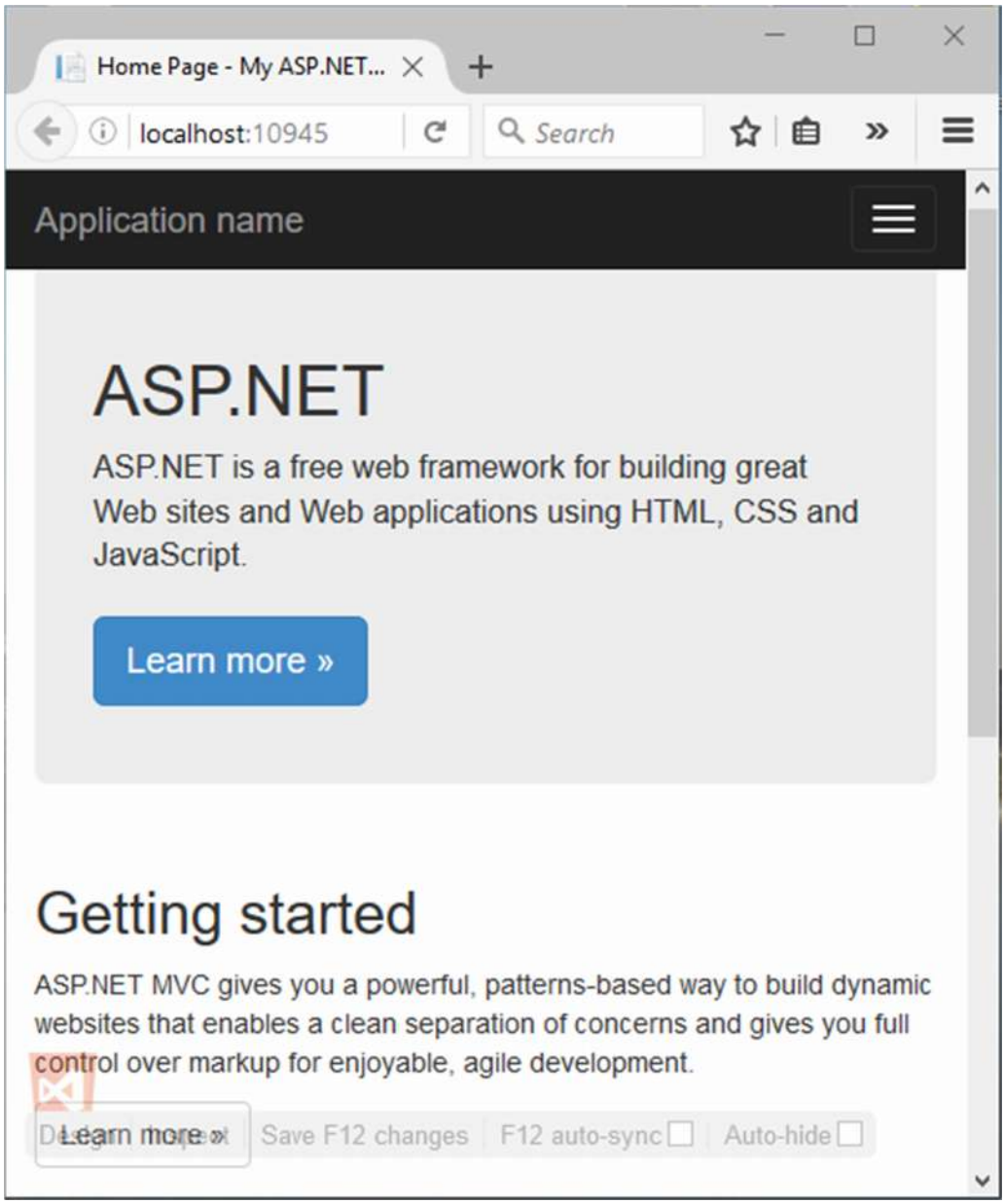


Aplikaciju je moguće pokrenuti u nekom od instaliranih pretraživača na računalu. Aplikacija je pokrenuta u *Google Chrome* pretraživaču. Osim što je aplikacija razvijena na vrlo jednostavan način, automatski je i kreiran programski kod koji omogućava jednostavnost i preglednost aplikacije na svim postojećim zaslonima. (slika 19. i slika 20.)

Slika 19. Prikaz MVC aplikacije na stolnom računalu



Slika 20. Prikaz MVC aplikacije na mobilnom uređaju



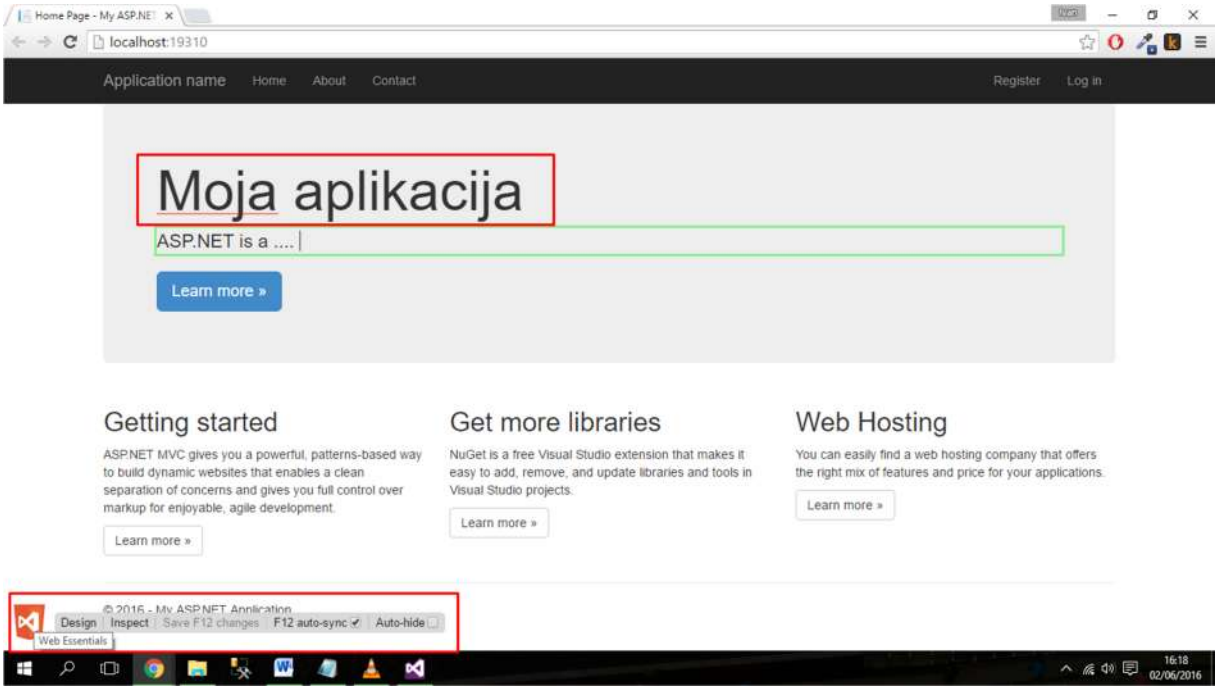
Dakle, na vrlo jednostavan način i u samo nekoliko koraka je prikazana prednost korištenja MVC arhitekture u kreiranju aplikacija. Na ovaj način može se kreirati aplikacija bez znanja izrade programskog koda i njezine implementacije na lokalni server[6].

[6] Server koji se generira na korisnikovom računalu

7. Dizajniranje MVC aplikacije

Nakon što je aplikacija kreirana moguće je malo urediti, odnosno prikazati je specifičnim izgledom. MVC pruža razne mogućnosti, pa tako i alate pomoću kojih se jednostavno može promijeniti izgled aplikacije. Ipak, pri uređivanju aplikacije potrebno je minimalno znanje HTML i CSS (*engl. Cascading Style Sheet*)[7] programskog jezika. Za jednostavnu promjenu bilo kojeg teksta, naslova ili svih poveznica unutar aplikacije korišten je alat *Web Essentials*.

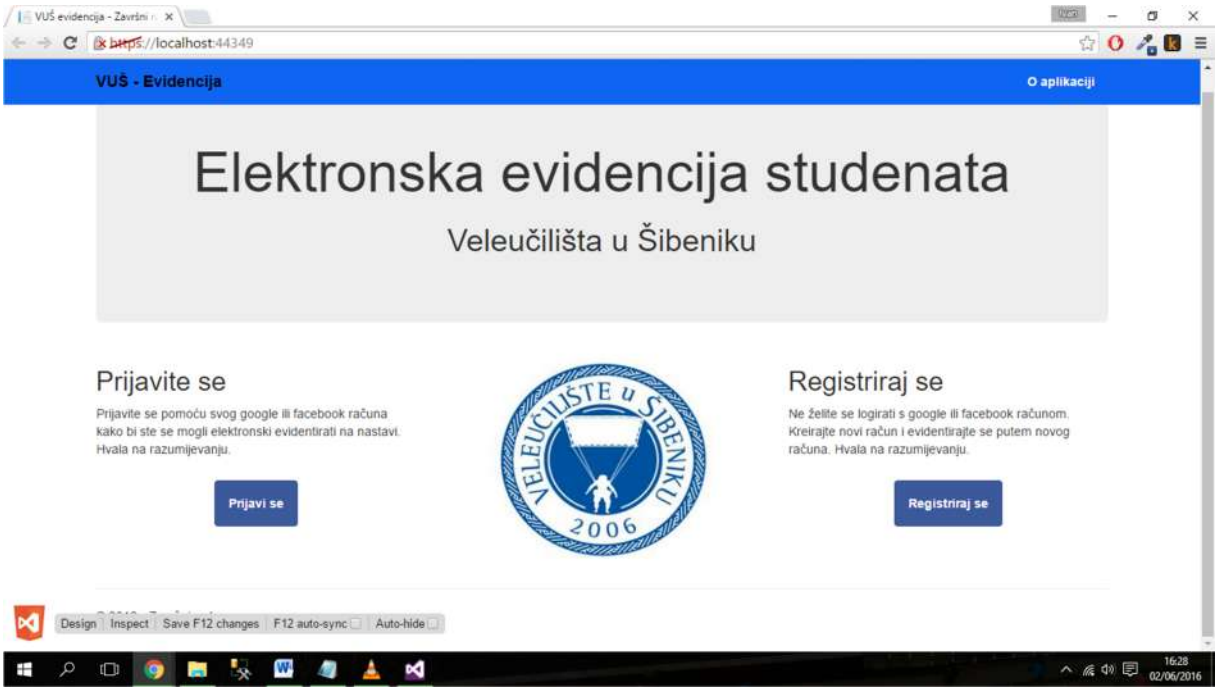
Slika 21. Prikaz uporabe Web Essentials



Na slici 21. može se primijetiti da je pomoću alata *Web Essentials* promijenjen naziv *ASP.NET* u *Moja aplikacija* (crveni pravokutnik) klikom na *Design* u donjem lijevom uglu te odabirom mjesta gdje se taj naslov nalazio. Poznavanjem CSS i HTML programskoj jezika moguće je kompletno promijeniti vizualni izgled razvijene MVC aplikacije. (slika 22.).

[7] CSS – opisuje kako elementi HTML koda trebaju biti prikazani na ekranu

Slika 22. Prikaz MVC aplikacije završnog rada

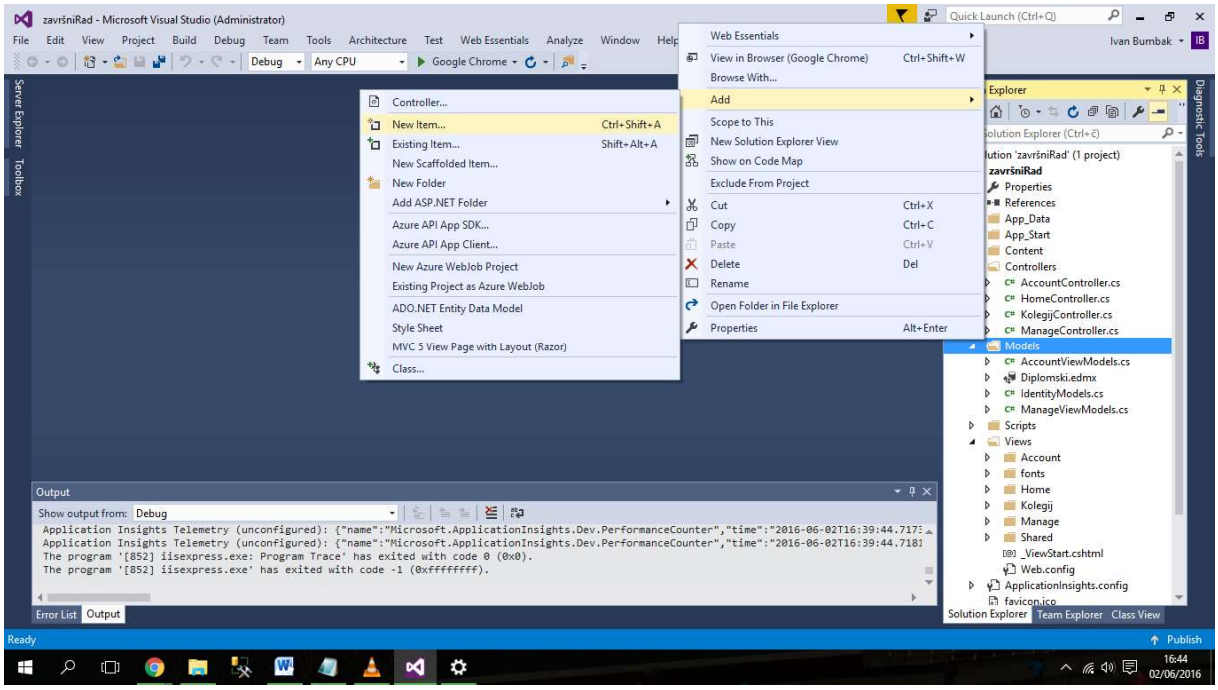


Nakon što je aplikacija dizajnirana može se krenuti na kreiranje modela i kontrolera MVC aplikacije. U praksi je zamišljeno da svaki tim radi svoj dio MVC arhitekture. Dakle jedan tim izrađuje *Model*, drugi tim *View*, odnosno izrađuje dizajn, a treći *Controller*, odnosno logiku aplikacije.

8. Kreiranje modela

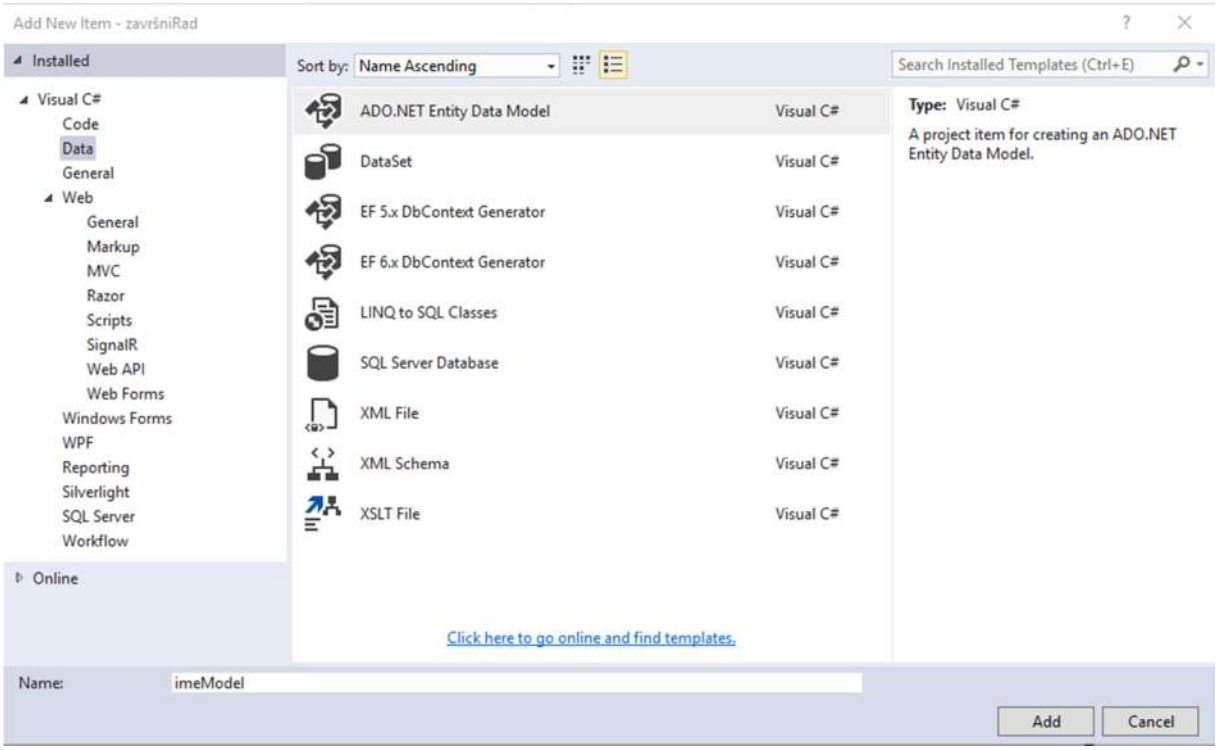
Kao što je kreirana aplikacija, na sličan način se kreira i model. Uz nekoliko klikova moguće je stvoriti model baze podataka potrebne za elektronsku evidenciju studenata. U *Solution Explorer* se označi mapa *Models* i desnim tipkom miša odabere *Add*, a zatim *New Item*….

Slika 23. Prikaz kreiranja novog modela



Pritiskom na *New Item …* otvara se prozor u kojem se odabire tip i vrsta modela. U lijevom izborniku odabire se tip modela: *web, visual C#, SQL Server* ili neki drugi ponuđeni tip modela. Model baze podataka moguće je izraditi odabirom *Data* opcije u lijevom izborniku, dok je u desnom izborniku potrebno odabrati *ADO.NET Entity Data Model*. Na dnu prozora upisuje se ime modela, dok se model izrađuje pritiskom na *Add*. (slika 24.)

Slika 24. Prikaz odabira vrste i tipa modela

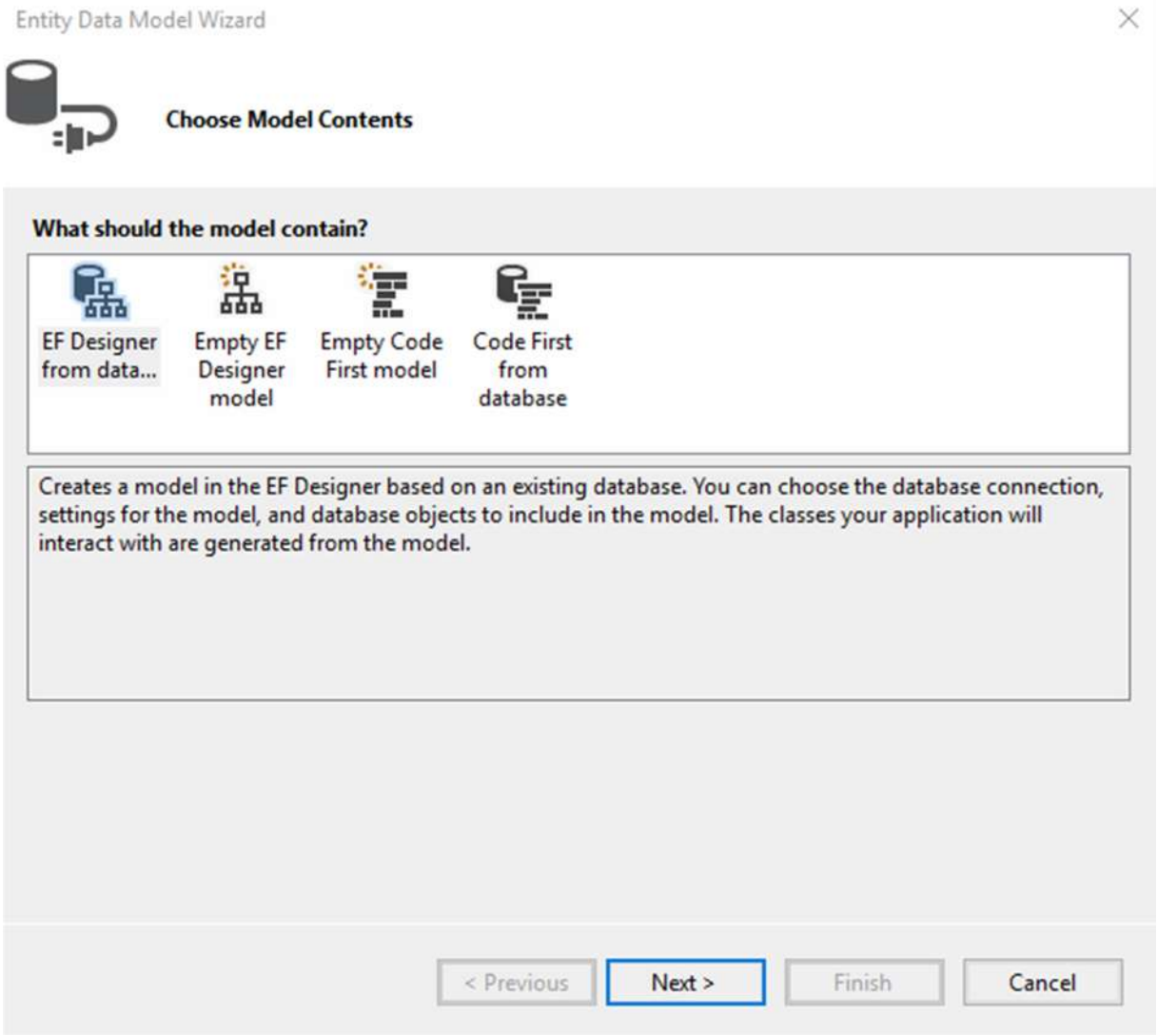


Nakon što je odabran tip i vrsta modela otvara se prozor u kojem se odabiru podaci koje model treba sadržavati. Može se odabrati između 4 opcije:

- *EF Designer from database* – kreira model baze podataka iz već postojeće baze podataka
- *Empty EF Designer model* – kreira prazni model u kojem je početni korak vizualno dizajniranje modela baze podataka
- *Empty Code First model* – kreira prazni model u kojem je početni korak kreiranje modela koristeći programski kod
- *Code First from database* – kreira programski kod iz postojeće baze podataka

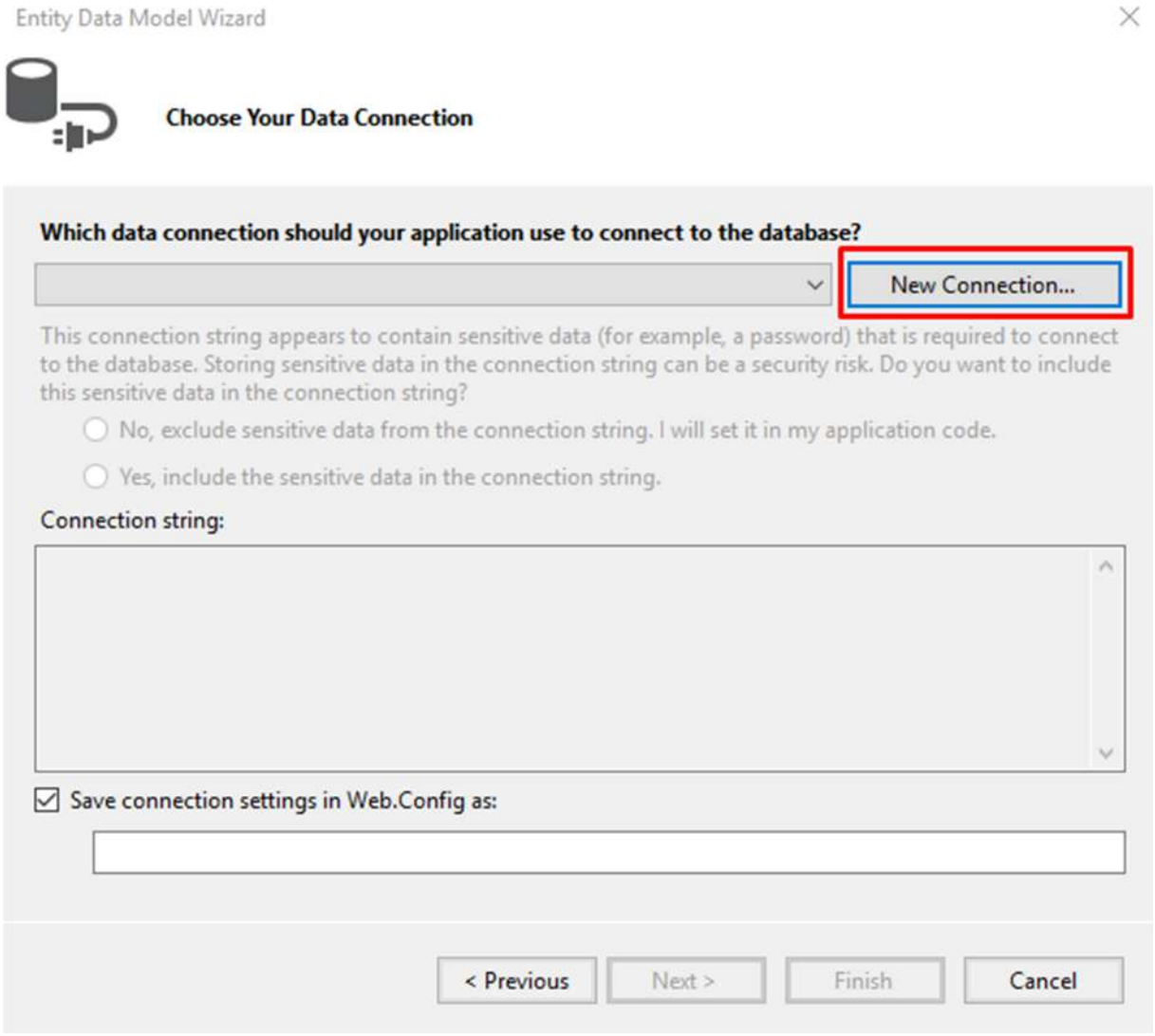
U svrhu završnog rada odabran je model *EF Designer from database*, jer se uvodi već postojeća baza podataka. (slika 25.)

Slika 25. Prikaz odabira EF Designer from database modela

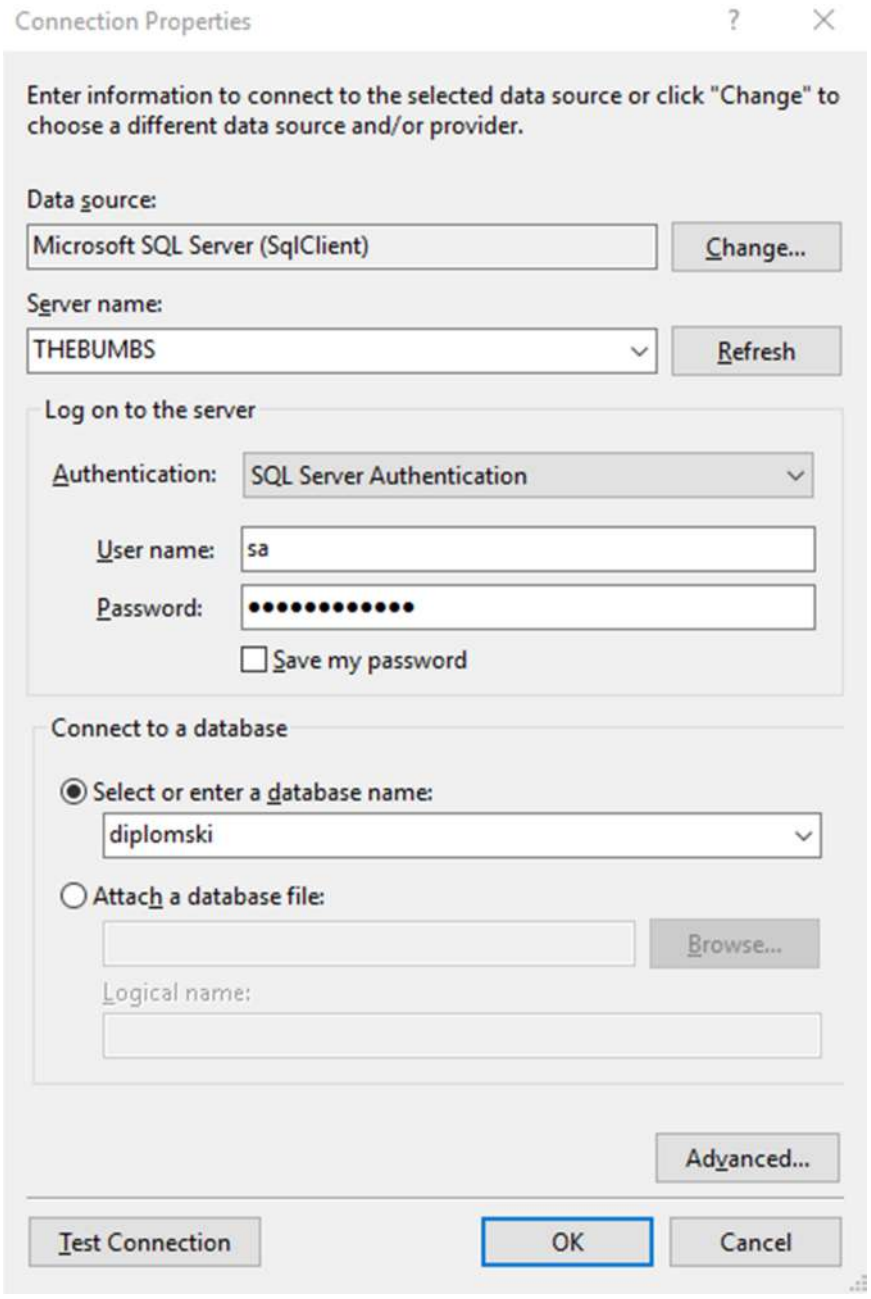


Nakon odabira sadržaja modela otvara se prozor u kojem će se aplikacija povezati sa ranije izrađenom bazom podataka (slika 26.). Klikom na *New Connection* te u novo otvorenom prozoru za *Data Source* je odabran *Microsoft SQL Server (SQL Client)*. Nadalje pritiskom na *Server Name* otvara se lista svih postojećih servera. Odabire se server na kojem je izrađena baza, te se prijavljuje na server sa SQL administratorskim računom. U trećem koraku odabire se potrebna baza podataka. Za završni rad odabrana je baza *diplomski*. (slika 27.). Klikom na tipku OK vraća se početni prozor stvaranja nove konekcije (slika 26.) sa upisanim podatcima. Prije nego što se krene na sljedeći korak preporučljivo je označiti opciju *Yes, include the sensitive dana in the connection string*. (slika 28.)

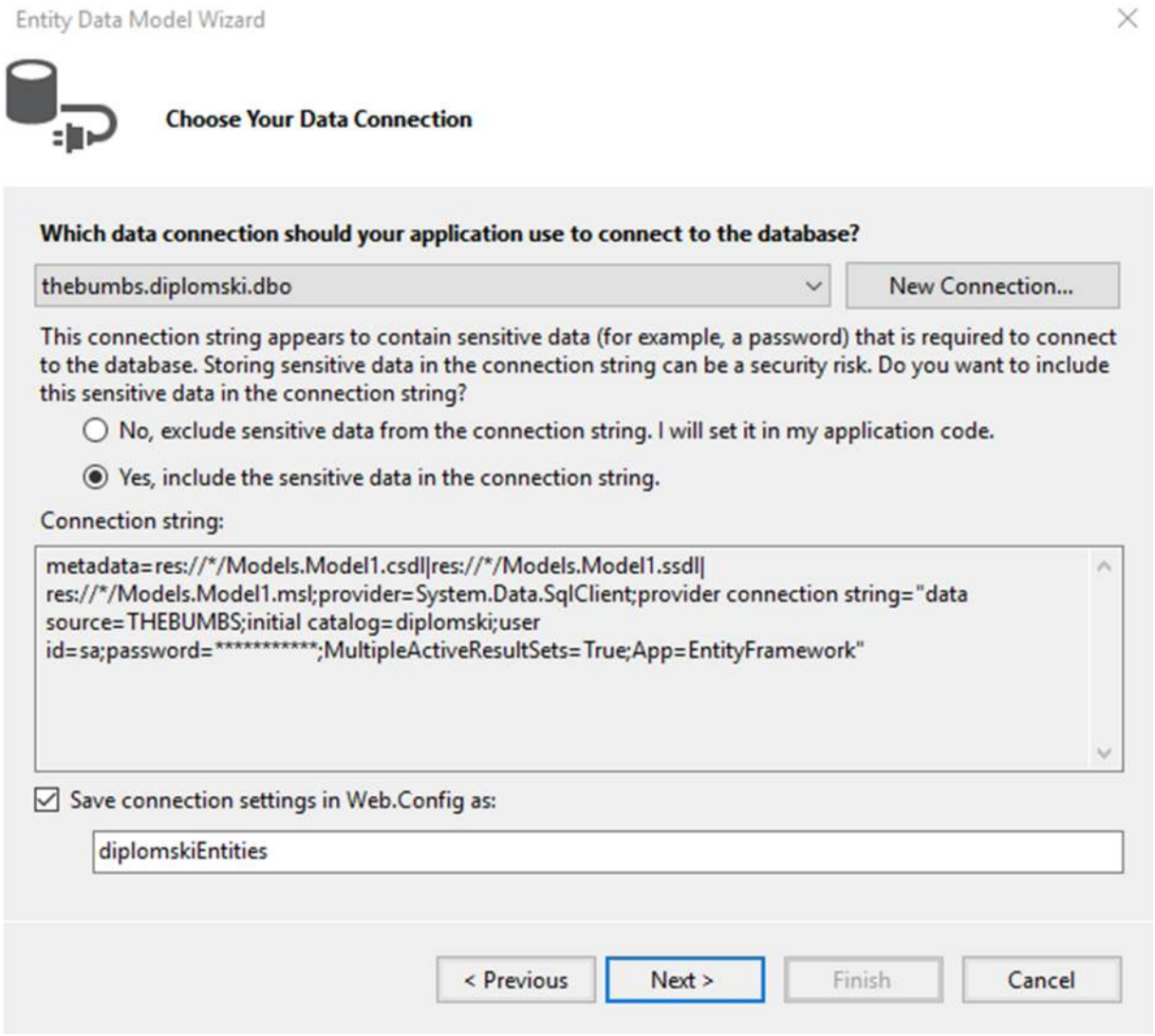
Slika 26. Povezivanje s bazom podataka



Slika 27. Prikaz spajanja na bazu diplomski

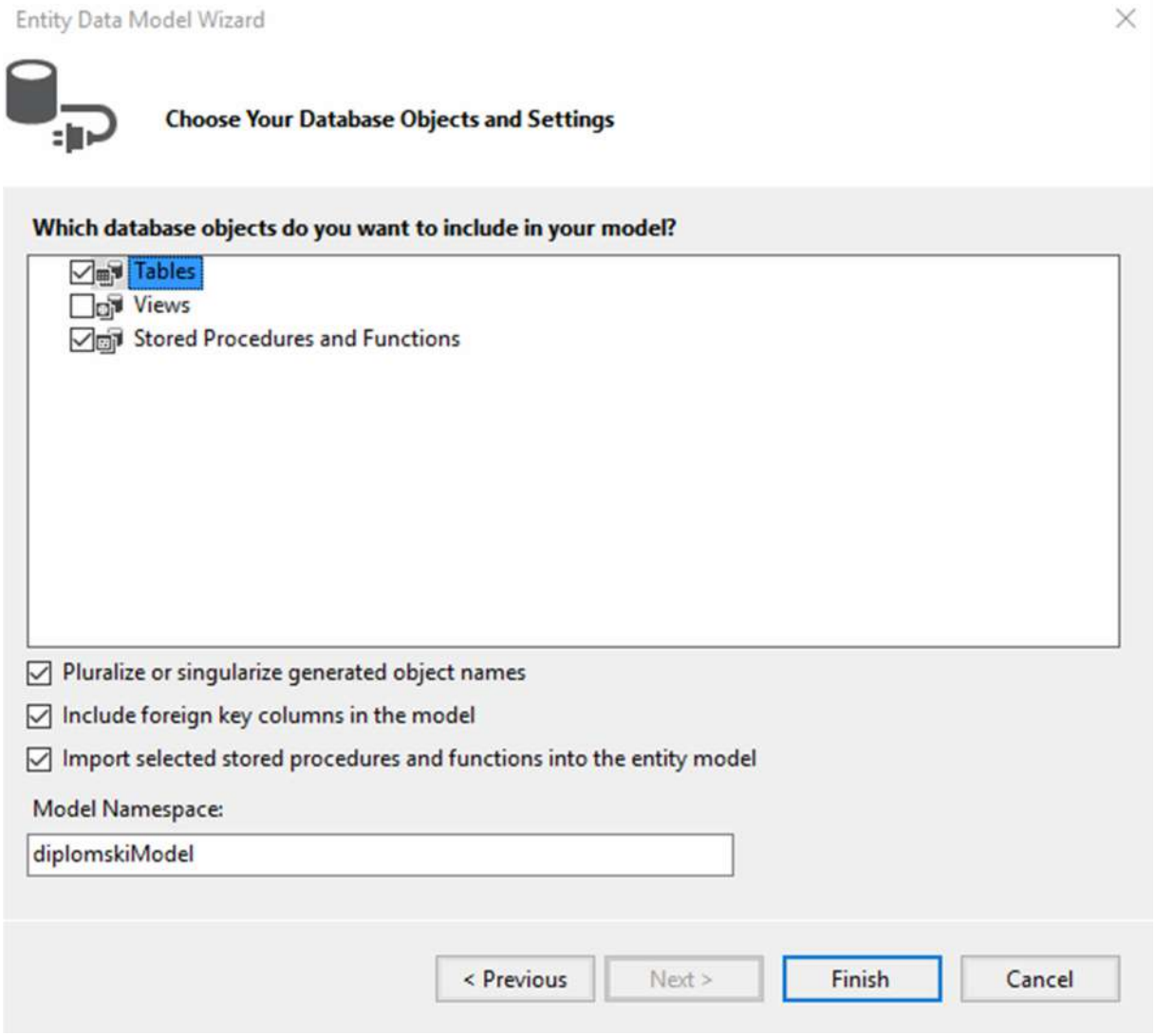


Slika 28. Prikaz podataka konekcije



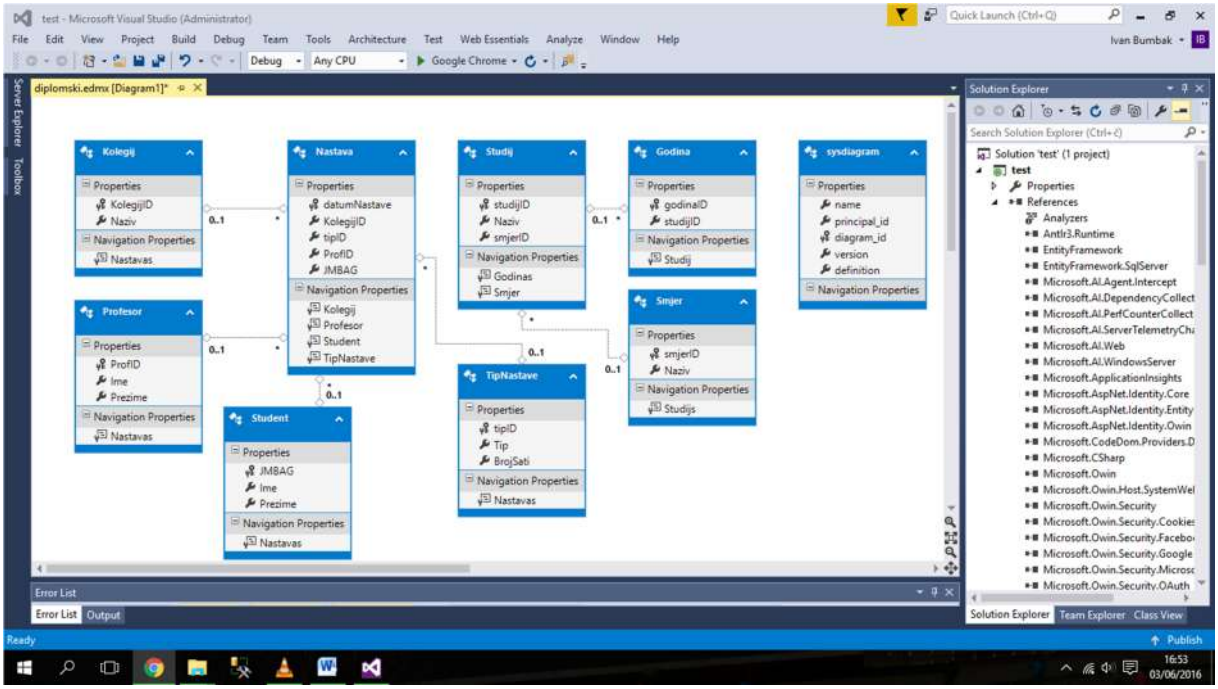
Klikom na tipku *Next* otvara se prozor u kojem je moguće odabrati koje objekte izabrane baze podataka se želi prikazati u modelu. Za završni rad *Elektronske evidencije studenata* odabrani su svi objekti koji su izrađeni prilikom razvijanja baze podataka. Moguće je promijeniti ime modela, ali u ovom slučaju ostavljeno je već zadano ime *diplomskiModel*. (slika 29.)

Slika 29. Odabir objekata baze podataka



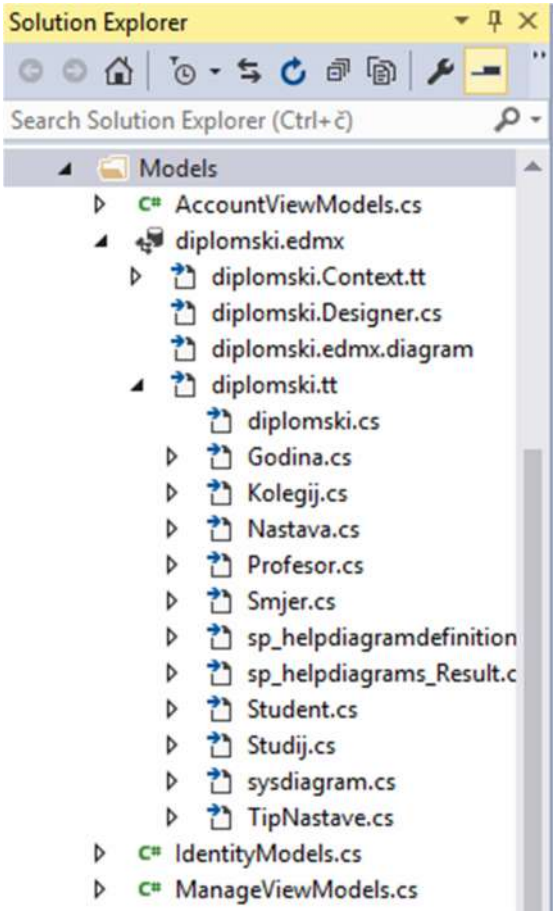
Pritiskom na tipku *Finish* automatski se generira dijagram modela baze podataka koji prikazuje sve definirane veze i relacije između tablica koje su prethodno kreirane. (slika 30.)

Slika 30. Dijagram modela baze podataka



Na desnoj strani u *Solution Explorer* prozoru potraži se mapa *Models* u kojoj se mogu pronaći svi objekti izabrane baze podataka.

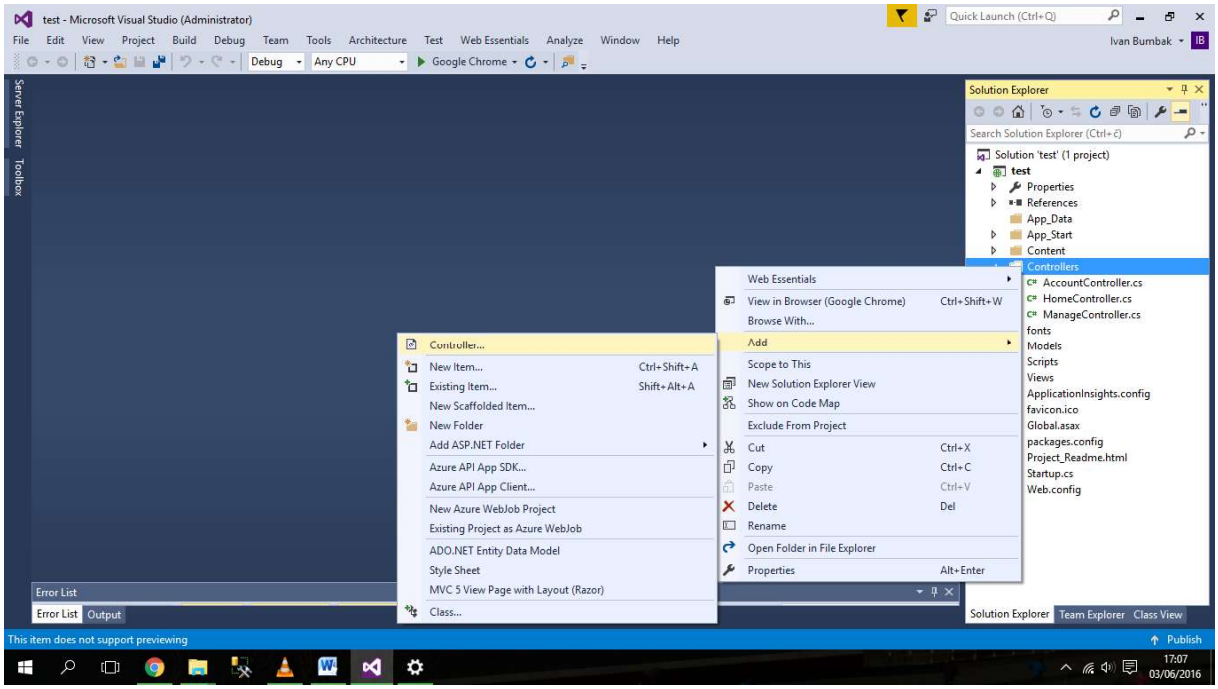
Slika 31. Prikaz svih objekata baze podataka u mapi Models



9. Kreiranje kontrolera

Nakon što je izrađen model, može se izraditi kontroler. Kontroler se izrađuje na način da se u prozoru *Solution Explorer* pronađe mapa *Controllers* i pritiskom na desnu tipku miša odabere *Add*, i u novo otvorenom prozoru *Controller*…. (slika 32.)

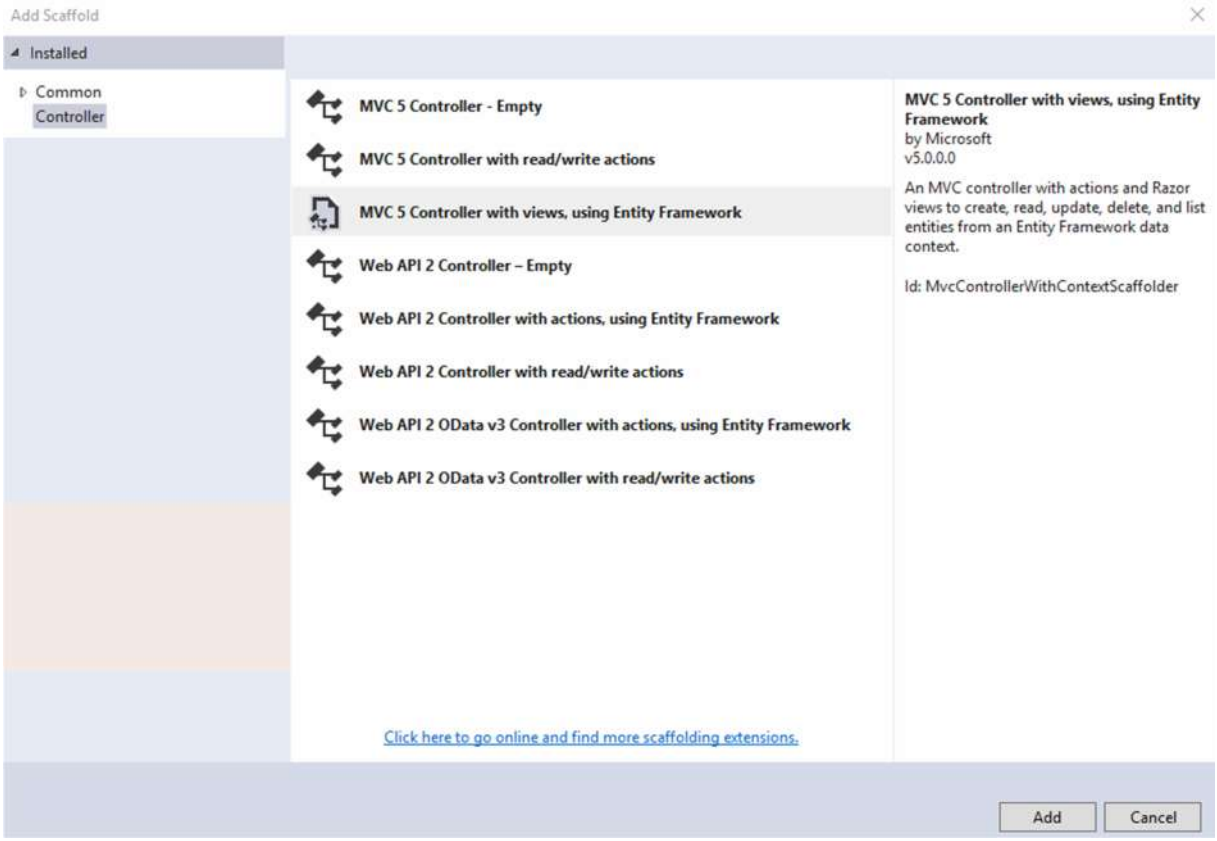
Slika 32. Kreiranje novog kontrolera



Klikom na *Controller* otvara se prozor u kojem se odabire tip *Scaffold-a*.[8] U svrhu završnog rada odabran je *MVC 5 Controller with views, using Entity Framework*, koji će automatski kreirati i potrebni *View* kojeg se može dizajnirati kao i cijelu aplikaciju. Nakon odabira na željeni *scaffold* klikne se na tipku *Add*. (slika 32.)

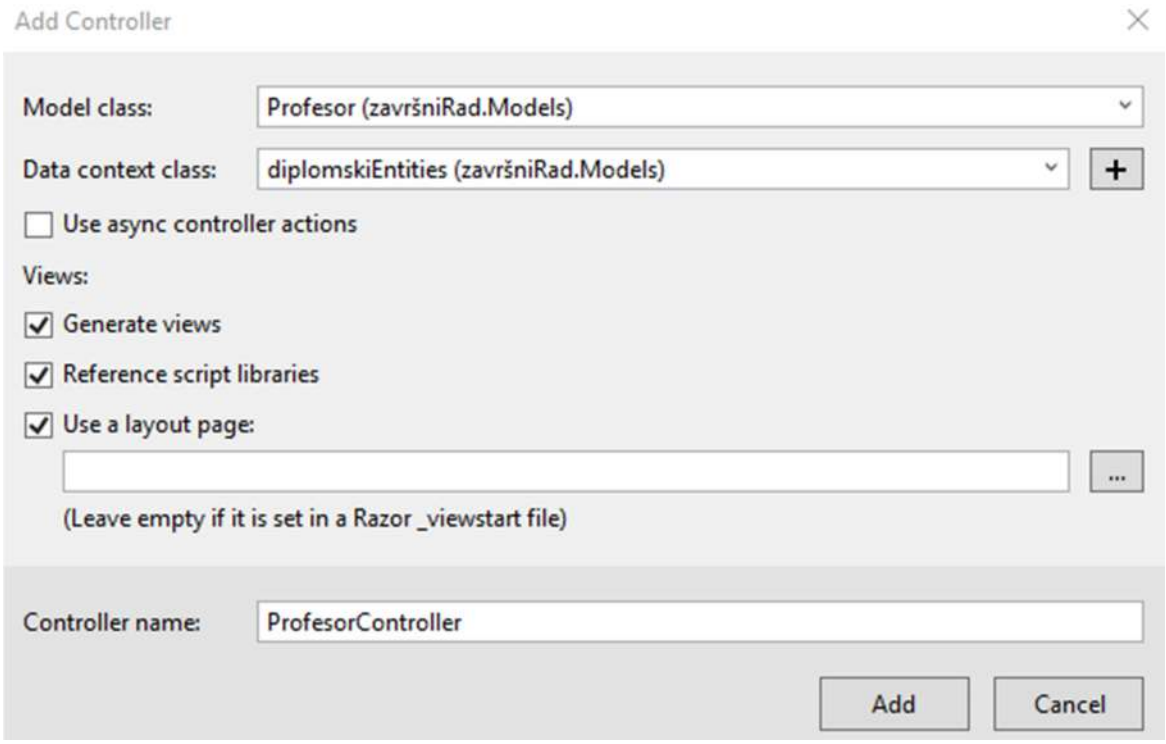
[8] Scaffold/Scaffolding – engl. konstrukcija koju koriste radnici prilikom izrade neke građevine

Slika 33. Odabir scaffold-a



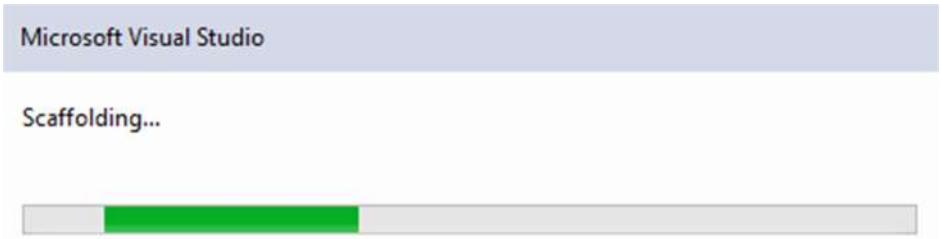
Klikom na tipku *Add* otvara se prozor *Add Controller* u kojem će se dodati novi kontroler. Prvi korak je odabir model razreda baze podataka koji je kreiran u prethodnom poglavlju, odnosno odabiremo jedan od objekata baze podataka kojeg se želi prikazati u aplikaciji te bazu podataka na koju je taj objekt povezan. Na dnu prozora automatski će se generirati ime kontrolera kojeg je preporučljivo nazvati kao objekt kontrolera koji se izrađuje. (slika 33.)

Slika 34. Odabir objekta kontrolera



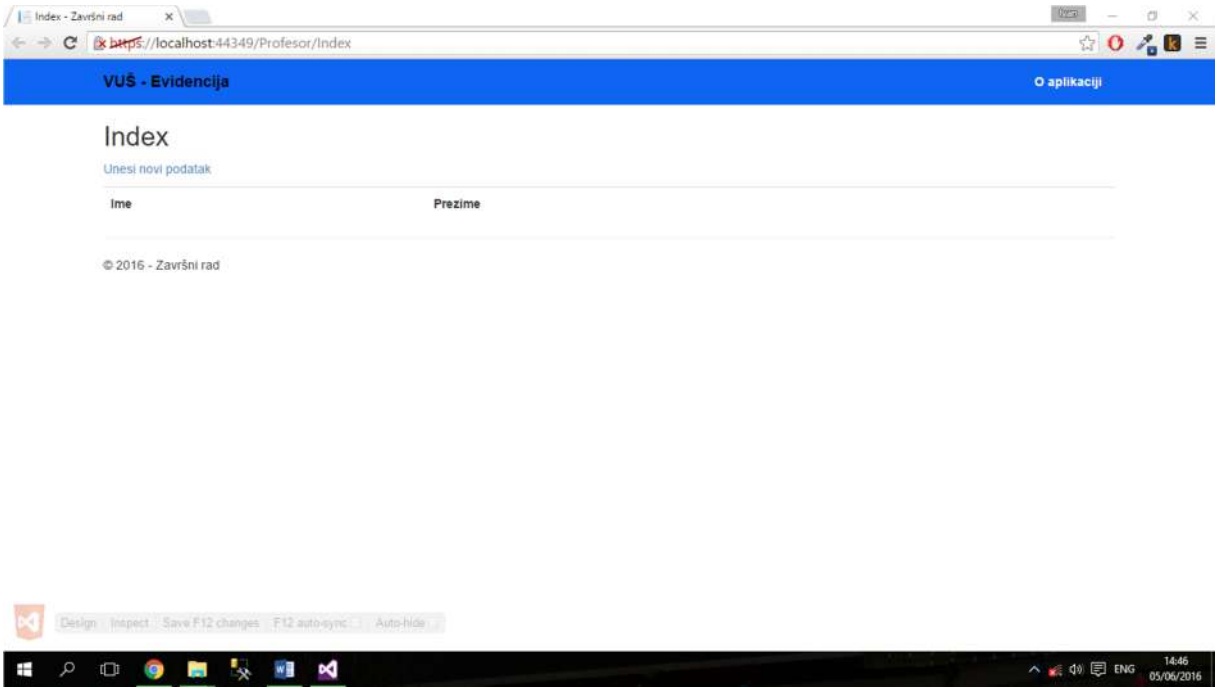
Nakon što je odabran željeni objekt, npr. *Profesor*, klikom na tipku *Add* se prikazuje obavijest *Scaffolding* što bi u doslovnom prijevodu značilo da se oko aplikacije nalaze skele preko kojih se ona uređuje, odnosno izgrađuje, poput gradnje nove građevine ili obnavljanjem postojeće građevine. (slika 34.)

Slika 35. Scaffolding …



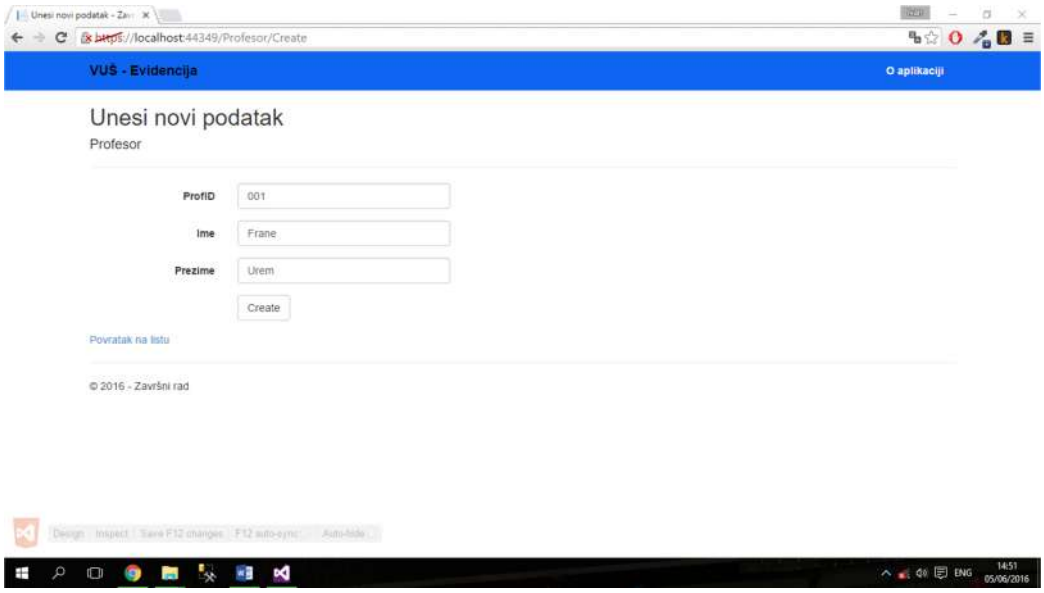
Nakon što se izvrši *scaffold* generira se programski kod koji preko *View* dijela MVC arhitekture korisnicima prikazuje sadržaj kojeg programski kod opisuje. Također, *View* će automatski izraditi mapu pod nazivom *Profesor*. Pokrene li se aplikacija u jednom od pretraživača i na već postojeći lokalni link aplikacije dopiše naziv odabranog objekta otvorit će se stranica na kojoj je moguće dodavati nove podatke, uređivati ili brisati postojeće podatke. (slika 35.)

Slika 36. Vizualni prikaz kontrolera Profesor

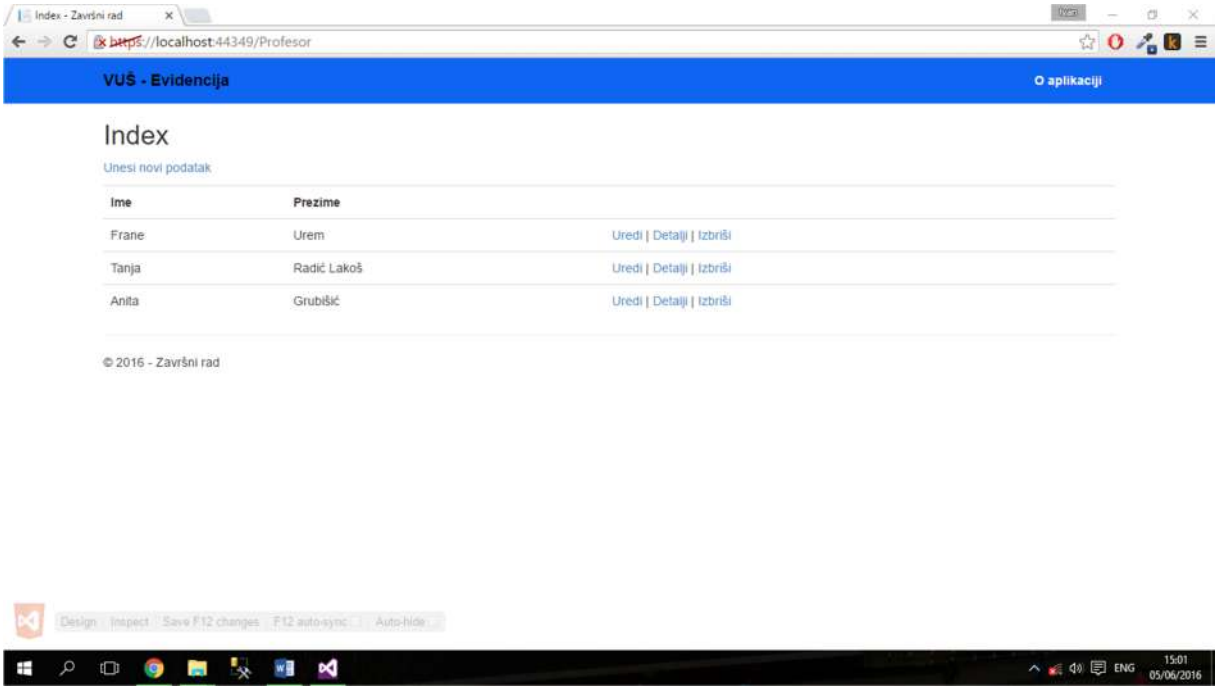


Klikom na *Unesi novi podatak* otvara se vrlo jednostavno sučelje u kojeg se upisuju potrebni podatci: *ID profesora*, *ime* i *prezime* profesora. Nakon što se popune sve prikazane ćelije klikne se na tipku *Create* koja će listu svih postojećih upisanih podataka u objektu *Profesor* upisati upravo ispunjene podatke (slika 36.). Na desnoj strani, svakog pojedinog upisanog podatka prikazati će se i naredbe *uredi, detalji* i *briši*. Naredbom uredi se uređuje odabran podatak, naredba detalji dohvaća detalje upisanog podatka, a naredba obriši briše odabrani podatak iz objekta *Profesor*. (slika 37.)

Slika 37. Prikaz unosa novog podatka u objekt Profesor



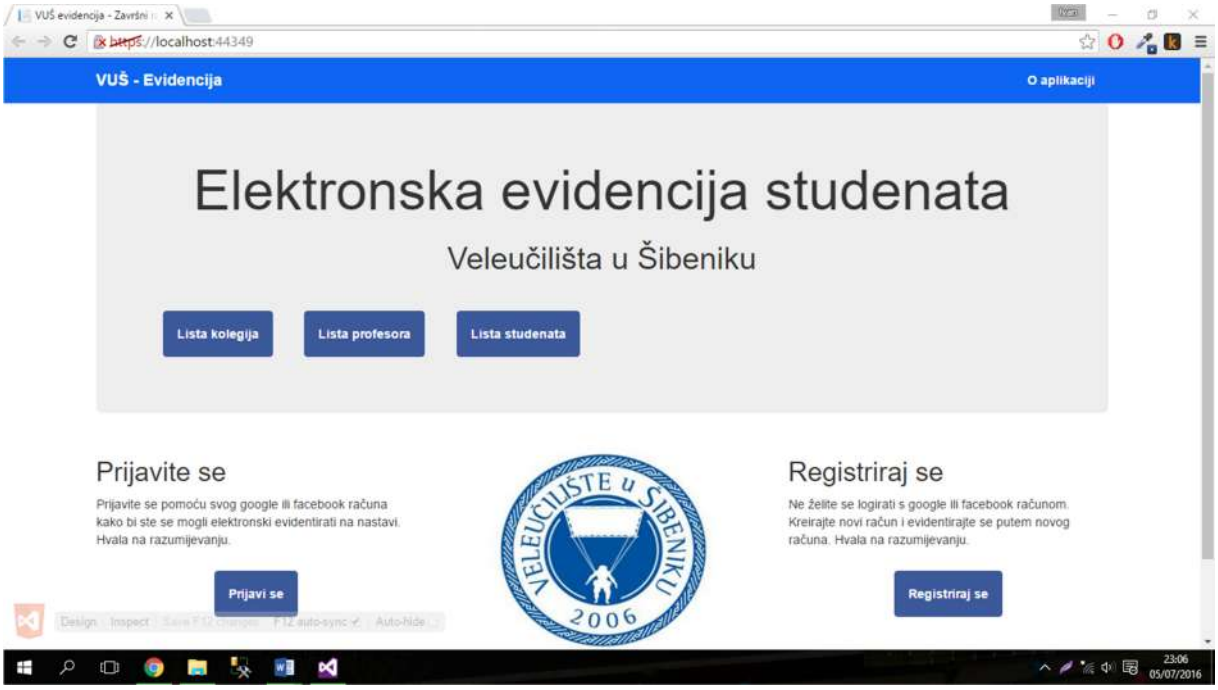
Slika 38. Lista upisanih podataka u objektu Profesor



10. View

View dio MVC arhitekture generira vizualnu sliku napisanog programskog koda. Većina *View-ova* generira se automatski nakon kreiranja potrebnih kontrolera. Uz malo poznavanja HTML i CSS programskog koda web aplikacija se može dizajnirati po želji. (slika 38.)

Slika 39. Prikaz poveznica na kreirane liste podataka



11. Zaključak

U 21. stoljeću sve više raste trend za obrazovanjem, ali i općenito informatizacijom poslovnih sustava. Korištenjem novih tehnologija, poput MVC arhitekture u vrlo kratkom vremenu, svega nekoliko minuta, s minimalnim znanjem programiranja i funkcionalnosti programskog koda može se dignuti vlastiti web sadržaj. Osim toga, MVC arhitektura omogućuje da se svaki segment web sadržaja izrađuje posebno, tako da svaki tim koji izrađuje web sadržaj može izrađivati samo ono u čemu je najbolji. Također, u radu je pokazan i jednostavan način pridruživanja postojeće baze podataka na web sjedište. Pridruživanjem baze podataka na web sjedište, kreira se aplikativno rješenje, pomoću kojeg se može dohvaćati, uređivati i dodavati potrebne podatke. U ovom aplikativnom rješenju prikazano je kako se na vrlo jednostavan način može dodavati studente, profesore, studije i ostale važne informacije preko web sučelja u samo nekoliko klikova.

Literatura

1. http://databases.about.com/od/sql/a/sqlfundamentals_2.htm (pristup: 07.05.2016.)

2. http://avant.org/project/history-of-databases/#1 (pristup: 07.05.2016.)

3. http://quickbase.intuit.com/articles/timeline-of-database-history (pristup: 25.05.2016)

4. http://www.unizd.hr/portals/1/primjena_rac/brodostrojarstvo/predavanje_4.pdf (pristup: 05.06.2016.)

Popis slika

1. Slika 1. Primjer dohvaćanje podataka iz baze Formula1, vlastit izvor, str. 2.

2. Slika 2. Organizacija povijesnih baza podataka, izvor: http://avant.org/project/history-of-databases/#1 , str. 3.

3. Slika 3. Prikaz mrežnog i hijerarhijskog modela baza podataka, vlastiti izvor, str. 4.

4. Slika 4. Primjena naredbi definirana DDL podjezikom, vlastiti izvor, str. 6.

5. Slika 5. Primjer naredbi definiranih DML podjezikom, vlastiti izvor, str. 7.

6. Slika 6. Primjer JOIN naredbe, vlastiti izvor, str 7.

7. Slika 7. ER dijagram baze podataka informatizirane evidencije, vlastiti izvor, str. 8.

8. Slika 8. Prikaz spajanja na lokalni Microsoft SQL Server 2014, vlastiti izvor, str. 9.

9. Slika 9. Kreiranje tablica, vlastiti izvor, str. 9.

10. Slika 10. Kreiranje tablica, vlastit izvor str. 10.

11. Slika 11. Kreiranje stranih ključeva, vlastiti izvor, str. 10.

12. Slika 12. Dijagram baze podataka informatizirane evidencije, vlastiti izvor, str. 11.

13. Slika 13. Komponente MVC-a, vlastiti izvor, str. 12.

14. Slika 14. Primjer Solution Exproler-a MVC aplikacije, vlastiti izvor, str. 13.

15. Slika 15. Prikaz početnog zaslona Microsoft Visual Studio Entreprise 2015 software-a, vlastiti izvor, str. 14.

16. Slika 16. Prikaz odabira ASP.NET Web aplikacije, vlastiti izvor, str. 15.

17. Slika 17. Prikaz odabira MVC aplikacije, vlastiti izvor, str. 15.

18. Slika 18. Prikaz dolaznog prozora MVC aplikacije, vlastiti izvor, str. 16.

19. Slika 19. Prikaz MVC aplikacije na stolnom računalu, vlastiti izvor, str. 16.

20. Slika 20. Prikaz MVC aplikacije na mobilnom uređaju, vlastiti izvor, str. 17.

21. Slika 21. Prikaz uporabe Web Essentials-a, vlastiti izvor, str. 18.

22. Slika 22. Prikaz MVC aplikacije završnog rada, vlastiti izvor, str. 19.

23. Slika 23. Prikaz kreiranja novog modela, vlastiti izvor, str. 20.

24. Slika 24. Prikaz odabira vrste i tipa modela, vlastiti izvor, str. 21.

25. Slika 25. Prikaz odabira EF Designer from database modela, vlastiti izvor, str. 22.

26. Slika 26. Povezivanje s bazom podataka, vlastiti izvor, str. 23.

27. Slika 27. Prikaz spajanja na bazu diplomski, vlastiti izvor, str. 24.

28. Slika 28. Prikaz podataka konekcije, vlastiti izvor, str. 25.

29. Slika 29. Odabir objekata baze podataka, vlastiti izvor, str. 26.

30. Slika 30. Dijagram modela baze podataka, vlastiti izvor, str. 27.

31. Slika 31. Prikaz svih objekata baze podataka u mapi Models, vlastiti izvor, str. 28.

32. Slika 32. Kreiranje novog kontrolera, vlastiti izvor, str. 29.

33. Slika 33. Odabir scaffold-a, vlastiti izvor, str. 30.

34. Slika 34. Odabir objekta kontrolera, vlastiti izvor, str. 31.

35. Slika 35. Scaffolding, vlastiti izvor, str. 31.

36. Slika 36. Vizualni prikaz kontrolera Profesor, vlastiti izvor, str. 32.

37. Slika 37. Prikaz unosa novog podatka u objekt Profesor, vlastiti izvor, str. 33.

38. Slika 38. Lista upisanih podataka u objektu Profesor, vlastiti izvor, str. 33.

39. Slika 39. Prikaz poveznica na kreirane liste podataka, vlastiti izvor, str. 34.