ELEKTRONSKA EVIDENCIJA GOSTIJU

Brala, Luka

Undergraduate thesis / Završni rad

2019

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:143:797171

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-27**

Repository / Repozitorij:

VUS REPOSITORY - Repozitorij završnih radova Veleučilišta u Šibeniku

zir.nsk.hr

dabar

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

VELEUČILIŠTE U ŠIBENIKU

ODJEL MENADŽMENTA

STRUČNI STUDIJ INFORMATIČKI MENADŽMENT

Luka Brala

ELEKTRONSKA EVIDENCIJA GOSTIJU

Završni rad

Šibenik, 2019.

VELEUČILIŠTE U ŠIBENIKU
ODJEL MENADŽMENTA
STRUČNI STUDIJ INFORMATIČKI MENADŽMENT

ELEKTRONSKA EVIDENCIJA GOSTIJU

Završni rad

Kolegij: Baze podataka
Mentor: mr. sc. Ivan Livaja, v. pred.
Student: Luka Brala
Matični broj studenta: 1219054584

Šibenik, 2019.

TEMELJNA DOKUMENTACIJSKA KARTICA

Veleučilište u Šibeniku Završni rad

Odjel Menadžmenta

Preddiplomski stručni studij Menadžment

ELEKTRONSKA EVIDENCIJA GOSTIJU

LUKA BRALA

Brune Bušića 45A 23242 Posedarje, luka.brala1@gmail.com

Elektronskom evidencijom gostiju korisnici će moći lako voditi brigu o evidenciji svojih gostiju i njihovim podacima. Ubrzavamo način upisa u evidenciju te tako štedimo na svom dragocjenom vremenu. Prezentiraju se metode i uloge, kao i implementacija programskog koda te spajanja baza podataka s web aplikacijom. Podaci koji su pohranjeni u bazi podataka na raspolaganju su korisniku za lakše pretraživanje i obradu. Za obradu i dohvaćanje podataka iz baze podataka zaslužan je računalni program *Database Managment System* (DBMS) koji omogućuje korisniku upravljanje tom bazom podataka. Jedna od značajnijih uloga DBMS programa je ta da omogućuje korisniku stvaranje, ažuriranje i kontroliranje baze podataka koju korisnik koristi. Aplikacija izrađena MVC arhitekturom je idealna za brz razvitak Web Aplikacije zato što se cijeli sustav može razviti uz ne veliko znanje programiranja, HTML-a i CSS-a. MVC arhitektura dopušta da se svaki dio aplikacije razvija zasebno i jednostavno poveže s bazom podataka, što uvelike omogućava brže i efikasnije stvaranje. Ova aplikacija nam daje jednostavnu i čitku strukturu podataka upisanih u bazu podataka te prikazuje kako se može na jasan i efikasan način upravljati podacima. Tako smo omogućili optimalno i brzo rješenje za upis, ažuriranje i brisanje podataka iz baze podataka te tako pokazali njenu primjenu.

(37 stranica / 28 slika / 8 literaturnih navoda / jezik izvornika: hrvatski)

Rad je pohranjen u: Knjižnici Veleučilišta u Šibeniku

Ključne riječi: mvc, sql, evidencija, aplikacija, web, html.

Mentor: mr. sc. Ivan Livaja, v. pred.

Rad je prihvaćen za obranu:

BASIC DOCUMENTATION CARD

Polytechnic of Šibenik

Department of Management

Professional Undergraduate Studies of Management

Final paper

ELECTRONIC EVIDENCE OF THE GUESTS

LUKA BRALA

Brune Bušića 45A 23242 Posedarje, luka.brala1@gmail.com

Electronic guest records will allow users to easily take care of their guests' records and their data. We are speeding up our records and thus saving our valuable time. Methods and roles are presented, as well as the implementation of program code and the merging of databases with a web application. The data stored in the database is available to the user for easy retrieval and processing. The Database Management System (DBMS) computer program, which allows the user to manage that database, is responsible for processing and retrieving data from the database. One of the significant roles of the DBMS program is that it allows the user to create, update and control the database that the user uses. An application created by MVC architecture is ideal for the rapid development of the Web Application because the whole system can be developed with little knowledge of programming, HTML and CSS. MVC architecture allows each part of the application to be developed individually and easily linked to the database, which greatly enables faster and more efficient creation. This application gives us a simple and readable structure of the data entered in the database and shows how it can be managed in a clear and efficient way. In this way, we enabled an optimal and fast solution for entering, updating and deleting data from the database, thus demonstrating its application.

(37 pages / 28 figures / 8 references / original in Croatian language)

Paper deposited in: Library of Polytechnic in Šibenik

Keywords: mvc, sql, records, application, web, html.

Supervisor: mr. sc. Ivan Livaja, v. pred.

Paper accepted:

Sadržaj

1. Uvod 1
2. Baze podataka 2
2.1. Početak baza podataka 3
2.1.1. Relacijski tip baza podataka 4
2.1.2. ER model baza podataka 5
3. Ciljevi baza podataka 6
4. Računalni jezik SQL 7
4.1. Primjena naredbi u bazi podataka 8
5. Primjena baza podataka – evidencija gostiju 11
5.1. Baza podataka evidencija gostiju 12
6. MVC 15
6.1. Značajke MVC-a 17
6.2. Prednosti MVC-a 17
6.3. Nedostaci MVC-a 18
6.4. Stvaranje MVC aplikacije 19
7. Kreiranje i dizajn MVC Aplikacije 21
7.1. Kreiranje Modela 21
7.2. Kreiranje View 25
7.3. Kreiranje Controller 28
8. Web Aplikacija 32
9. Zaključak 34
Literatura 35
Popis slika 36

1. Uvod

Sve veći broj ljudi i sve manje vremena zbog obveza učinilo je informatizaciju, mobilizaciju i pohranu podataka efikasnijom i bolje optimiziranom. Živimo u digitalno doba pa ključnu ulogu imaju informacije dostupne u što kraćem roku te kojima se može upravljati na vrlo jednostavan način. Spoj informatike u turizmu uvelike olakšava produktivnost, efikasnost i brzu primjenu djelovanja. Iz godine u godinu raste broj turista pa je neophodno ubaciti informatiku u poslovanje. Tako će aplikacijom za evidenciju gostiju korisnici moći lako voditi brigu o svojim gostima, njihovim prijavama i odjavama. Ubrzavamo način upisa u evidenciju te tako štedimo na svom dragocjenom vremenu. U prvom dijelu rada ističu se sami počeci baza podataka, njihovo korištenje i razvoj. Isto tako, objašnjava se i zašto se i gdje koriste te koja im je primjena u radu. U drugom dijelu rada, odnosno ostatku rada, prikazat ćemo primjenu baza podataka sa spojem izrade web aplikacije. Prezentiraju se metode i uloge, kao i implementacija programskog koda te spajanja baza podataka s web aplikacijom.

2. Baze podataka

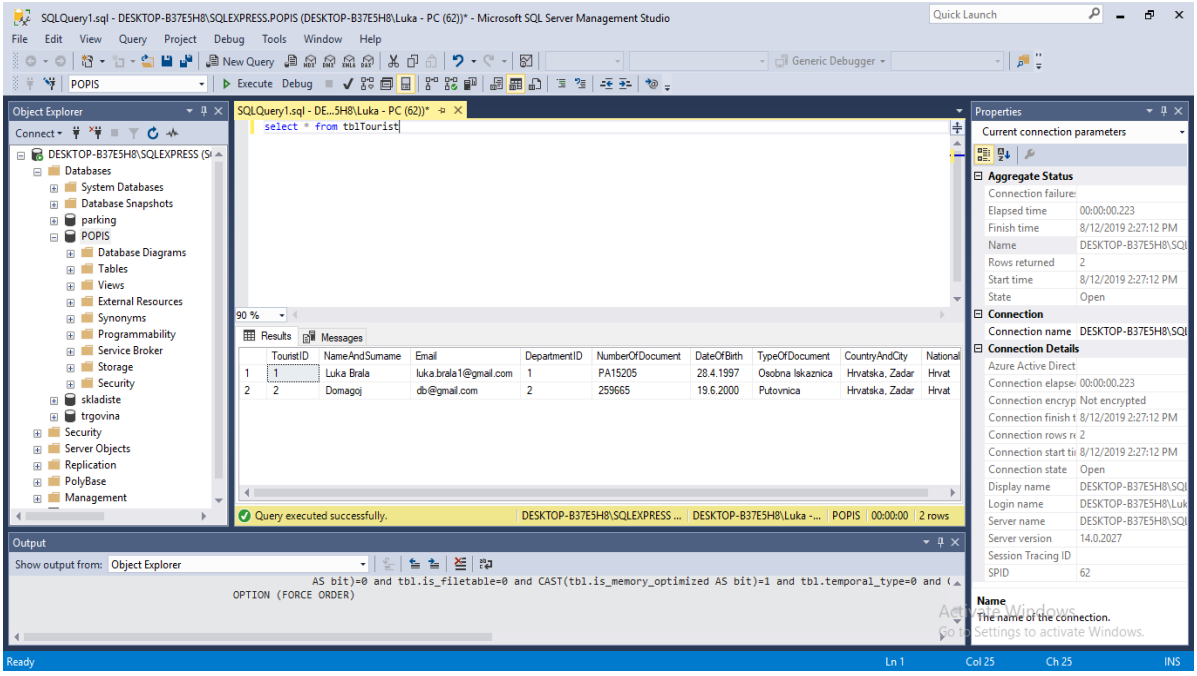
Baza podataka je skup međusobno povezanih podataka pohranjenih na disku. Ti podaci su na raspolaganju korisnicima za pregledavanje, pretraživanje, brisanje, nadopunjavanje i ispravljanje.[1]

Podaci koji su pohranjeni u bazi podataka obično budu u obliku tablica, teksta, skripti ili grafičkih oblika te kao takvi su na raspolaganju korisniku za lakše pretraživanje i obradu.

Za obradu i dohvaćanje podataka iz baze podataka zaslužan je računalni program [2]*Database Managment System* (DBMS) koji omogućuje korisniku upravljanje tom bazom podataka. Jedna od značajnijih uloga DBMS programa je ta da omogućuje korisniku stvaranje, ažuriranje i kontroliranje baze podataka koju korisnik koristi.

Kada govorimo o programu za dohvaćanje podataka iz baze podataka najpoznatiji su Microsoft SQL, MySQL te Oracle program za bazu podataka. Ti programi nam služe za pretraživanje, izlistavanje i brisanje podataka iz navedene baze podataka.

Slika 1. Dohvaćanje i izlistavanje podataka tablice Tourist iz baze podataka POPIS



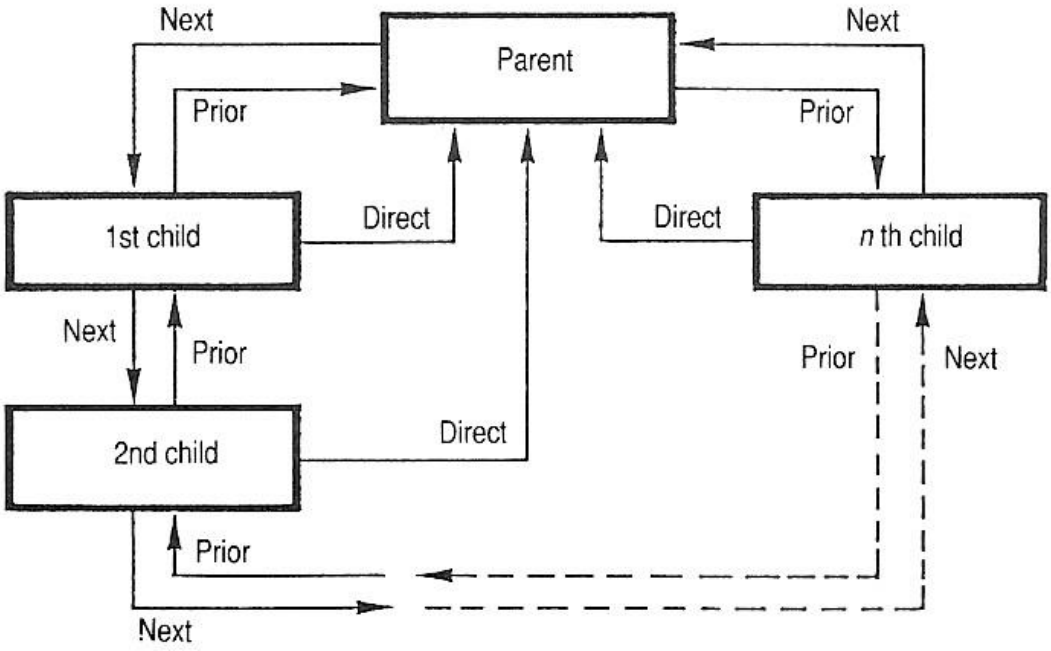
[1] Definicija - http://grdelin.phy.hr/~ivo/Nastava/Baze_podataka/predavanja-2004/01_pred.pdf
[2] Računalni program – struktura naredbi koje su napisane prema pravilima, a koja sadrži informacije za lakše upravljanje podacima

2.1. Početak baza podataka

Još od početka civilizacije ljudi su prikupljali, čuvali i spremali neke podatke, stvari ili predmete. U ne tako davnoj prošlosti ljudi su spremali podatke u određenim spisima. Baze podataka nisu bile u digitalnom obliku, pa su tako ustanove poput knjižnica, bolnica, škola i raznih ustanova skladištili, odnosno spremali podatke u papirnatom obliku.

Baze podataka u prvom obliku kako ih danas poznajemo nalaze svoj početak u 1960.-ima kada velike firme i poduzeća shvaćaju da im se više isplati takav oblik skladištenja podataka. Na samom početku počinju djelovati dva tada popularna modela baza podataka kao što su CODASYL[3] koji je fleksibilan u prikazivanju objekata te njihovog odnosa. Također, tu se javlja i hijerarhijski model nazvan IMS[4] koji je strukturiran po granama.

Slika 2. Primjer CODASYL modela baza podataka



Izvor: https://upload.wikimedia.org/wikipedia/commons/d/d6/CodasylB.png

U svojoj povijesti baza podataka još možemo spomenuti i SABRE[5] sustav kojeg je koristio i IBM.

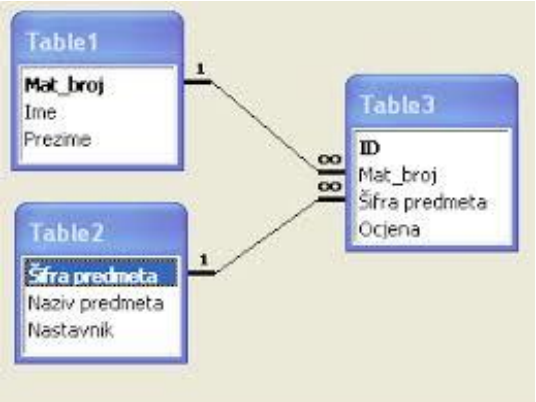
[3] Konferencija za sistemske jezike podataka, nastala 1959. godine.
[4] IMS baza podataka je velika, centralizirana zbirka podataka koja sadrži jednu ili više fizičkih datoteka
[5] Sustav baza podataka kojeg je koristio IBM za upravljanje rezervacijama letova za *American Airlines*

Kada govorimo o povijesti baza podataka ključnu ulogu je odigrao Edgar Frank Codd koji je objavio rad u kojem jasno daje do znanja da je relacijski sustav baza podataka efikasniji.

2.1.1. Relacijski tip baza podataka

Relacijski tip baza podataka je sustav u kojem se podaci organiziraju po relacijskom modelu, što znači da se podaci sažimaju u tablice, odnosno u skup tablica, entiteta te između njih definiramo veze, a to znači na koji način povezujemo tablice. Entiteti su objekti, stvari koji nas zanimaju, tj. objekt kojeg izdvajamo iz realnog ili apstraktnog svijeta kojeg opisujemo svojstvima ili atributima. Za entitet možemo reći i da je on skup atributa, jer recimo, entitet Turist može imati više atributa kao što su osobni broj, ime, prezime, datum rođenja, nacionalnost , grad rođenja i slično. Da bi povezali, tj. da bi se stvorila određena veza između entiteta, svaki entitet mora imati primarni ključ. Primarni ključ znači da se preko njega definiraju svi drugi podaci u modelu. Tako recimo imamo tri vrste relacijskih veza, a to su “1 : N”, “M : N” i “1 : 1”. Kada smo spomenuli relacijsku vezu “M : N”, obavezno mora nastati nova među-tablica. U toj novoj tablici koju nazivamo među – tablica ključevi koje smo zadali kao primarne nastaju strani ključevi pomoću kojih spajamo te tablice. Takav način se koristi i danas te je jako prihvaćen i usvojen.

Slika 3. Relacijski model, povezivanje tablica s među-tablicom.



Izvor: http://genderi.org/rad-s-bazom-podataka-u-net-okruenju-model-relacijske-baze-poda.html

2.1.2. ER model baza podataka

Osim relacijskog modela postoji i ER (*Entity – Relationship)* model baza podataka koji je omogućio programerima da se baziraju na podatke aplikativne namjene. ER model je jednostavan toliko da ga ljudi koji i nisu školovani u ovoj domeni mogu razumjeti. Ovaj model baza podataka zato služi za komunikaciju stvoritelja baza podataka i budućih korisnika, pa čak i u početku razvoja baze podataka.

Brz rast i razvoj prodaje računala uvelike je utjecao na primjenu baza podataka tijekom 1980.-ih te je tako relacijski model utjecao i na komercijalni uspjeh. Tijekom usavršavanja, razmatranja i razvoja relacijskog modela pojavio se i razvio SQL programski jezik te je tako s vremenom postao standard u izradi baza podataka. Tih godina IBM igra ključnu ulogu na tržištu razvoja baza podataka i izrada računala, pa tako IBM stvara svoju bazu podataka pod nazivom DB2.

Godine 1990. počinju se razvijati novi programi, odnosno alati koji pomažu i olakšavaju upravljanje bazama podataka, a neki od njih su Microsoft SQL, MySQL te Oracle program za bazu podataka, Microsoft Excel, Microsoft Access i drugi.

3. Ciljevi baza podataka

Baze podataka znače višu razinu u radu i obradi podataka u odnosu na standardne obične programe. Smatra se da čine višu razinu zato što nastoje ispuniti neke od ciljeva:

- Fizička neovisnost podataka – promjenom fizičke strukture ne zahtjeva promjene u postojećim aplikacijama.
- Logička neovisnost podataka – promjenom logičke definicije ne zahtijevaju promjene u zadanoj aplikaciji, što znači da lokalna logička mreža se svodi na jednostavne promjene tih elemenata.
- Fleksibilnost pristupa podacima – sa starim načinom i starom hijerarhijom podaci su bili definirani unaprijed, dok je korisnik imao mogućnost pretraživanja samo onim redom koji je bio unaprijed definiran izradom baze. U današnje vrijeme korisnik može bilo kojim redom pretraživati podatke.
- Pristup do podataka – omogućuje većem broju korisnika korištenje podataka istovremeno.
- Čuvanje integriteta – korektnost i očuvanost podataka mora se vršiti automatski, pa čak i kad postoje pogreške u programu.
- Sigurnost od neovlaštene uporabe – postoji sigurnost da se korisnicima ograniče mogućnosti upravljanja u nekim dijelovima baze podataka zbog neovlaštenog korištenja.

4. Računalni jezik SQL

Računalni jezik SQL je jedan od najrasprostranjenijih računalnih jezika za izradu i brisanje, ažuriranje i pretraživanje podataka. Podaci su temeljni dio mnogih web aplikacija i mobilnih aplikacija. Na primjer, aplikacija poput Facebooka sadrži podatke o profilu korisnika, uključujući podatke o njihovim prijateljima i postove. Za držanje tih podataka koristi se sustav baze podataka. SQL (strukturirani jezik upita) je programski jezik koji programerima omogućuje rad s tim podacima.[6]

Kao i drugi programski jezici, SQL ima svoj način pisanja koda. Zbog toga programer mora naučiti programski jezik SQL-a prije nego što ga može učinkovito koristiti.

Osim označavanja, još jedna značajka jedinstvena za programiranje baze podataka je koncept tablica. Baza podataka može biti predstavljena kao broj tablica. Svaka tablica ima svoj broj stupaca i redaka i predstavlja skup podataka.

Kao primjer možemo dati knjižnicu i razvrstavanje podataka o knjigama. Mogli bismo stvoriti bazu podataka koja pohranjuje podatke o knjigama u knjižnici.

Slika 4. Primjer tablice za pohranjivanje podataka o knjigama u knjižnici

Books	SQL Data Type
ID (book)	integer
Title	Char (200)
Author	Char (200)
Genre	Char (100)
Year	Date
Language	Char (50)

Izvor: https://learntocodewith.me/posts/sql-guide/

[6] Izvor: https://learntocodewith.me/posts/sql-guide/

4.1. Primjena naredbi u bazi podataka

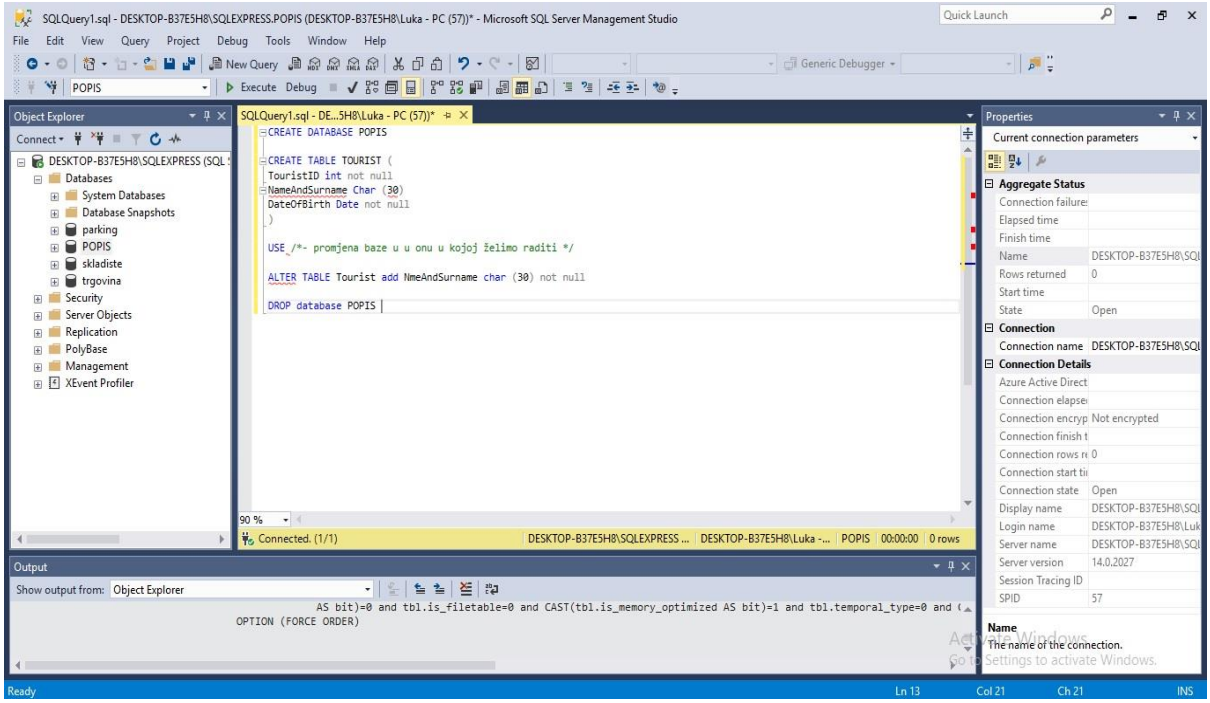
Postoji nekoliko često korištenih SQL naredbi s kojima biste trebali biti upoznati za rad na bazi podataka. U radu s bazama podataka, programer može pisati naredbe kao što su:

- CREATE DATABASE – s ovom naredbom programer stvara novu bazu podataka u kojoj će dalje izraditi tablice, definirati entitete, veze među tablicama i slično.
- CREATE TABLE – s ovom naredbom programer kreira tablicu u zadanoj bazi podataka koju je prethodno kreirao. Tablicu će definirati s dva podatka, tj. stupca.
- ALTER – ova naredba omogućuje promjene u već izrađenoj tablici. Ova naredba neće utjecati na postojeću tablicu, neće izbrisati ni poništiti tablicu.
- DROP – ovo je naredba koja omogućuje brisanje ukupnih podataka, tj. objekata iz zadane baze podataka.
- USE – koristi se kada korisnik mijenja podatke na kojima želi izvršiti promjene.

DDL (*Data Definition Language)*[7] je sintaksa slična računalnom programskom jeziku za definiranje struktura podataka, posebno shema baza podataka. DDL izrazi stvaraju, mijenjaju i uklanjaju objekte baze podataka poput tablica, indeksa i korisnika. Uobičajeni DDL izrazi su CREATE, ALTER i DROP

[7] Definicija - https://en.wikipedia.org/wiki/Data_definition_language

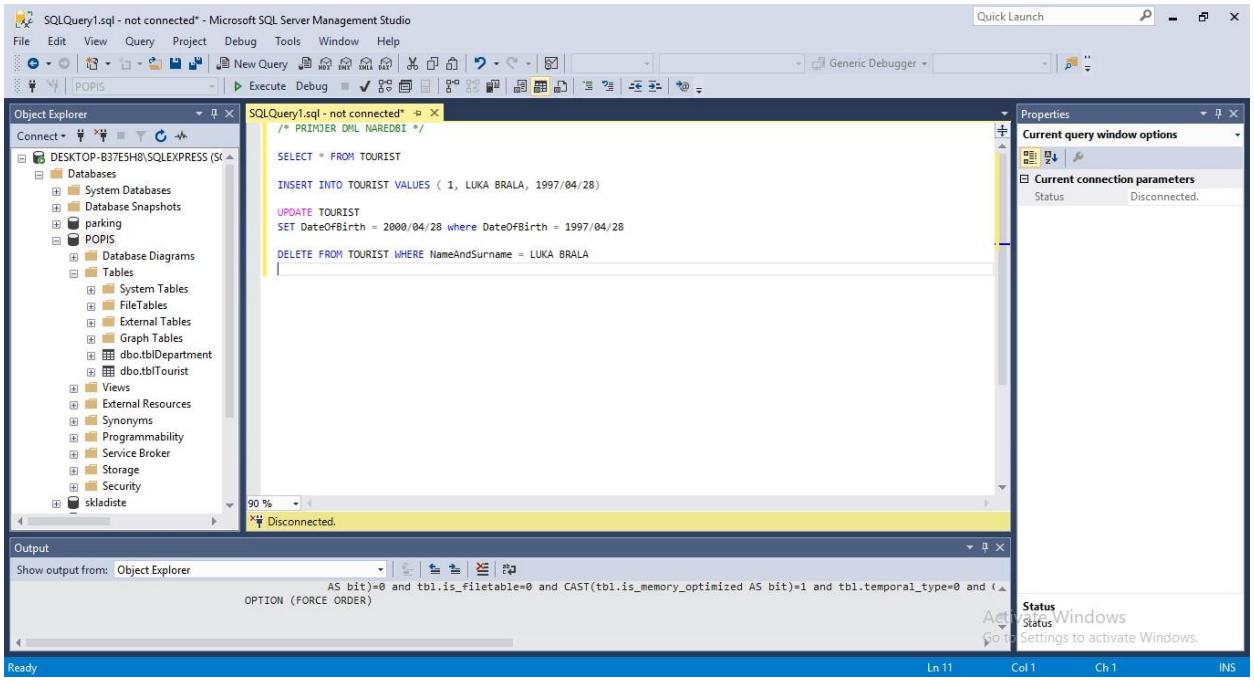
Slika 5. Primjer DDL naredbi



Kada je struktura baze podataka definirana DDL jezikom, korisnici koji žele unijeti podatke koriste se DML podjezikom, a najčešće naredbe koje se koriste u DML podjeziku su:

- SELECT - za pronalazak / izdvajanje nekih podataka iz baze podataka.
- UPDATE - prilagodite i uredite podatke.
- INSERT – naredba se koristi kada želimo ubaciti određene podatke u tablice i slično.
- DELETE – koristi se da bi se izbrisao određeni sadržaj.

Slika 6. Primjer DML naredbi



U primjeni baza podataka postoji još naredbi, a definiraju se pomoću naredbe JOIN. Preko naredbe JOIN korisniku je dozvoljeno podatke iz više tablica sažeti u jedno. Tako ćemo pomoću naredbe JOIN dobiti jednu vrstu podataka iz više tablica.

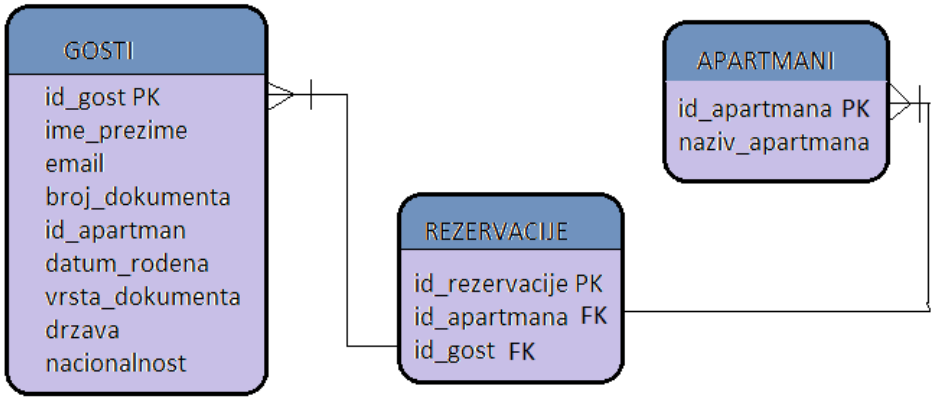
Postoje četiri dijela naredbi JOIN:

- INNER JOIN – ova naredba će dohvatiti sve podatke koji su strukturirani u tablici onda kada postoji minimalno jedna veza u obje tablice.
- LEFT JOIN – kako i samo ime kaže, ova naredba dohvaća podatke s lijeve strane i parove podataka s desne tablice.
- RIGHT JOIN – suprotno od LEFT JOIN, naredba dohvaća podatke iz desne tablice i parove iz lijeve.
- FULL JOIN – kada je postojan par, tada ova naredba sažima sve podatke u tablici.

5. Primjena baza podataka – evidencija gostiju

Prije same izrade baze podataka za jednostavnu evidenciju gostiju poželjno je sve potkrijepiti ER dijagramom. S ER dijagramom ćemo bolje utvrditi vezu između tablica, dat će nam jasan pregled veza i odnosa između entiteta i slično.

Slika 7. Primjer ER dijagrama baze podataka evidencije gostiju



Iz prethodne slike možemo lako vidjeti da nam za jednostavnu evidenciju gostiju trebaju 3 entiteta, a to su: *GOSTI, APARTMANI i REZERVACIJE*. Kao što smo već ranije spominjali na početku rada, svaki entitet je opisan svojim atributima. Recimo, entitet *GOSTI* je opisan atributima kao što su: *id_gost, ime_prezime, email, broj_dokumenta* itd.

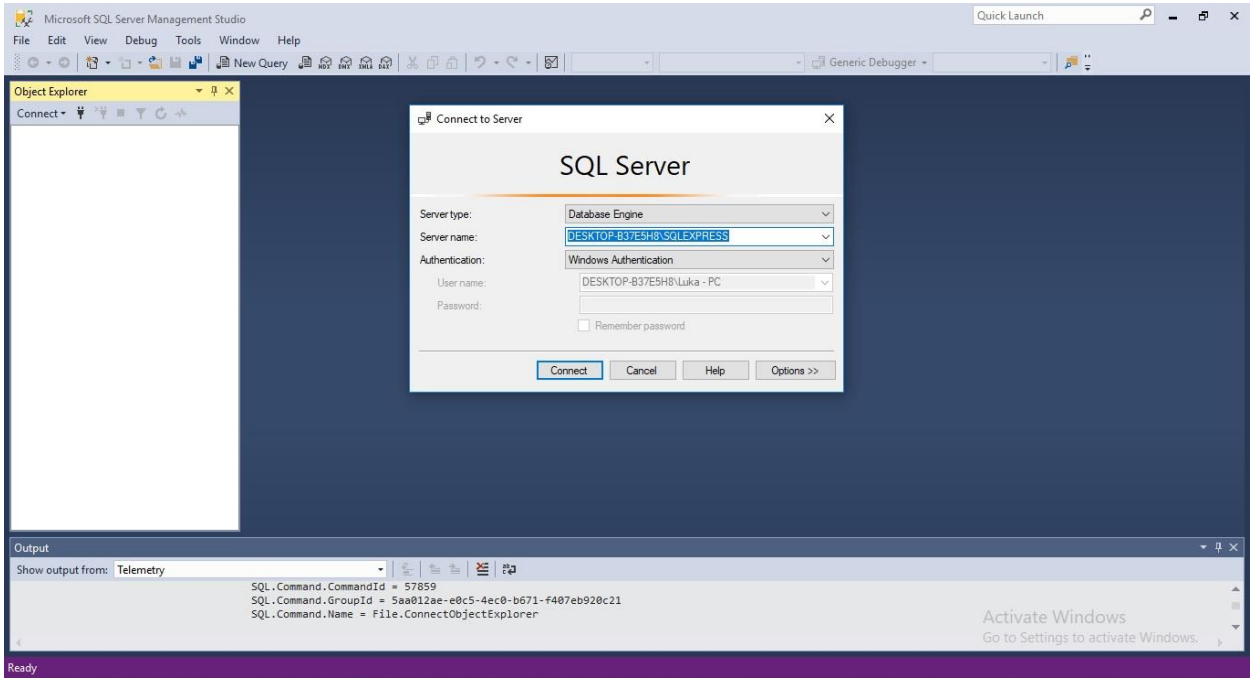
Za primjer uzmimo ova dva entiteta, *GOSTI* i *APARTMANI*, svaki od njih ima svoj primarni ključ- Pomoću relacijskih veza „M : N“ dolazimo do jedinstvenog spajanja tablica u relacijski vezu pomoću među tablice pod nazivom *REZERVACIJE*. Primarnim ključem nazivamo onaj atribut koji jedinstveno opisuje navedeni entitet te nam on služi za vezu s drugom tablicom, što znači da se on spaja na njihove strane ključeve. U našem slučaju, primarni ključ tablice *GOSTI* je *id_gosti* koji se spaja na strani ključ među tablice *REZERVACIJE*. Isto tako primarni ključ tablice *APARTMANI* nam daje mogućnost spajanja na među tablicu pomoću stranog ključa pod nazivom *id_apartmana*.

5.1. Baza podataka evidencija gostiju

Postupak nakon izrade ER dijagrama je izrada baze podataka, odnosno, potrebno je izraditi fizički model. Primjer ćemo pokazati u programu *Microsoft SQL Server*.

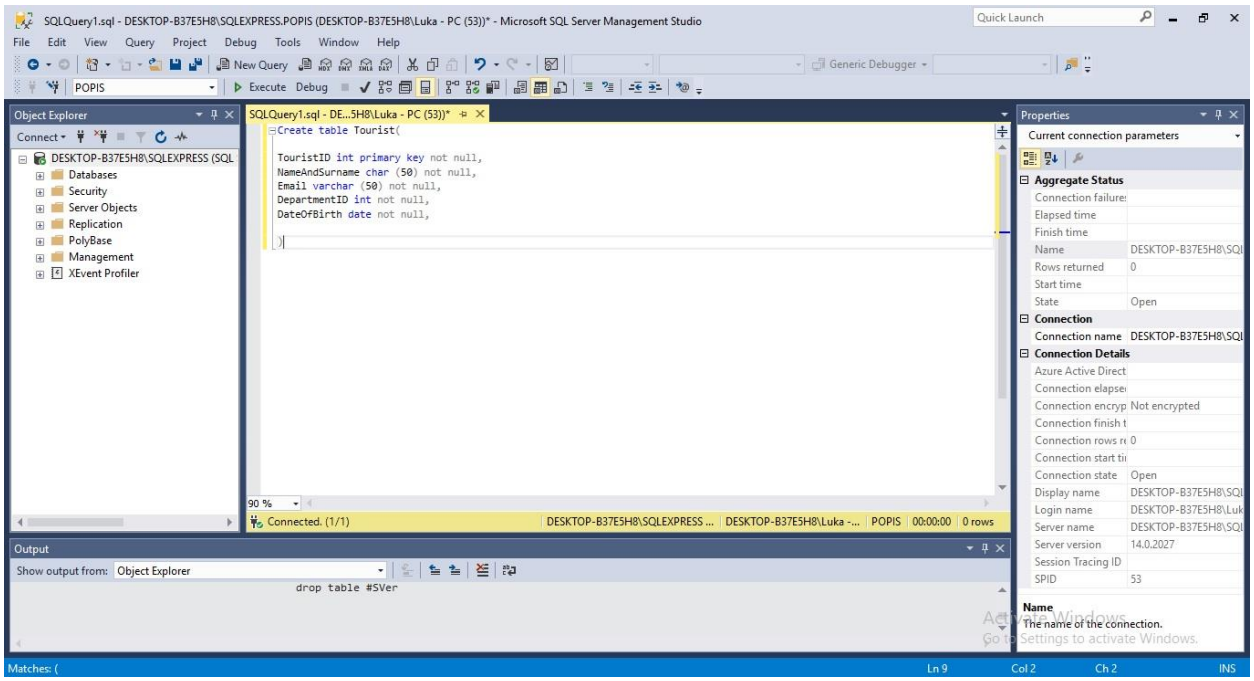
Prvi postupak nam je taj da ćemo pri pokretanju programa spojiti se na lokalni server i otvoriti novu bazu podataka, odnosno *New Query* gdje ćemo u prvom koraku kreirati novu bazu podataka.

Slika 8. Pokretanje programa Microsoft SQL Server



Nakon spajanja na server i kreiranja baze podataka s naredbom CREATE DATABASE krećemo na izradu tablica.

Slika 9. Primjer kreiranja tablice

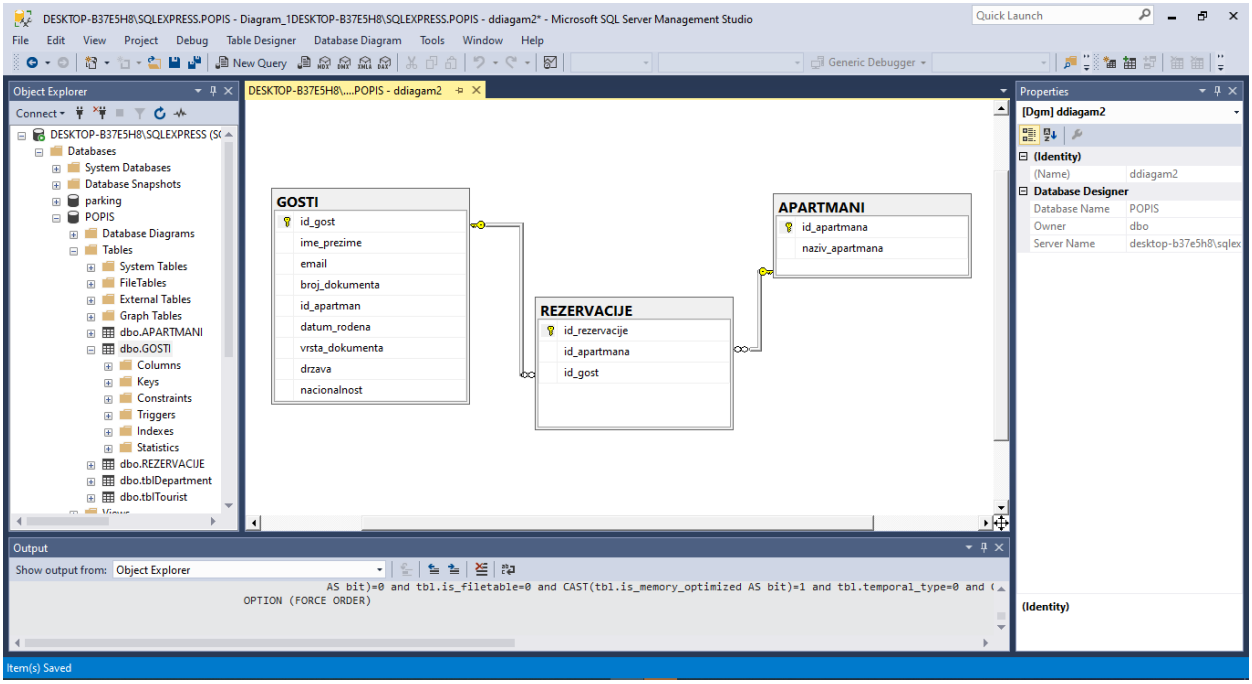


Kreiranje tablice započinje jednostavnom naredbom CREATE TABLE *ime_tablice*. Tablice su postavljene tako da predstavljaju entitete koje smo prije toga zadali ER dijagramom. Isto tako, možemo primijetiti da svaka tablica, odnosno svaki entitet sadrži primarni ključ. Nakon toga, u povezivanju tablica, kreiramo među-tablice gdje kreiramo strane ključeve za povezivanje dvaju tablica.

Kada postavimo i kreiramo primarne ključeve, tada smo spremni za generiranje dijagrama baze podataka. Ako smo sve dobro postavili i isprogramirali, povezali tablice, tada će MSQL (*Microsoft SQL server Managment Studio*) automatski stvoriti dijagrame po našem zadanom kodu.

Iz slike 10. možemo vidjeti jednostavnu primjenu dijagrama koji se automatski kreirao generiranjem koda, spajanjem tablica. Odnosno, povezivanjem tablica putem stranih i primarnih ključeva. U ovom slučaju mi smo spojene tablice prikazali u *Microsoft SQL Server Managment Studiu*. Zatim ćemo nakon toga povezati našu bazu podataka s Microsoft Visual Studijom te nastaviti s izradom aplikacije.

Slika 10. Dijagram baze podataka generiran spajanjem tablica



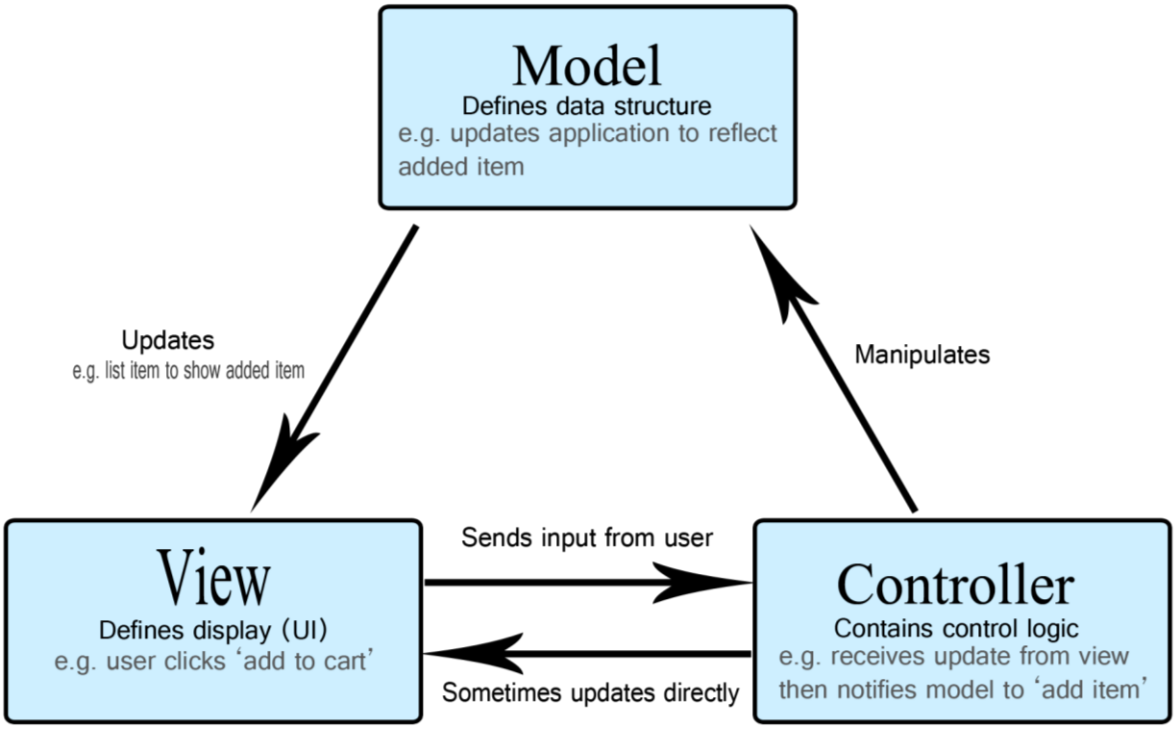
Glazbene aplikacije poput Spotify i Pandora također čine baze podataka s intenzivnom uporabom. Između ostalog, baze podataka pomažu tim aplikacijama da pohrane ogromne biblioteke glazbenih datoteka i albuma raznih izvođača, upravljaju tim podacima da bi pronašli ono što korisnik traži, pohranili podatke o korisnicima i njihovim preferencijama itd.

6. MVC

MVC (*Model View Controller*) je kratica za Model, View i Controller. MVC je popularan način organiziranja vašeg koda. Velika ideja koja stoji iza MVC-a je da svaki odjeljak vašeg koda ima svrhu, a te svrhe su različite. Neki od vašeg koda sadrže podatke vaše aplikacije, neki od vašeg koda čine vašu aplikaciju lijepom, a neki vaš kôd kontrolira kako vaša aplikacija funkcionira.

MVC[8] je način organiziranja osnovnih funkcija koda u svoje uredno organizirane okvire. To olakšava i čistije razmišljanje o vašoj aplikaciji, pregled njezine aplikacije i dijeljenje aplikacije s drugima.

Slika 11. Dijelovi Model-Viev-Controller



Izvor: https://developer.mozilla.org/en-US/docs/Glossary/MVC

[8] Izvor: https://www.codecademy.com/articles/mvc

Model-View-Controller [9]ili skraćeno MVC je aplikacijski model dizajniran od tri međusobno povezana dijela. Uključuju model (podatke), prikaz (korisničko sučelje) i kontroler (procesi koji upravljaju ulazom).

MVC model ili "uzorak" obično se koristi za razvoj modernih korisničkih sučelja. Pruža temeljne dijelove za dizajn programa za radne površine ili mobilne uređaje, kao i web aplikacije. Odlično funkcionira s objektno orijentiranim programiranjem, budući da se različiti modeli, pogledi i kontroleri mogu tretirati kao objekti i ponovno se koristiti u aplikaciji.

MVC se sastoji od tri dijela:

1. Model

Model su podaci koji koristi program. To može biti baza podataka, datoteka ili jednostavan objekt, poput ikone ili znaka u videoigri.

2. Pogled

Pogled je način prikazivanja predmeta u aplikaciji. Primjeri uključuju prikazivanje prozora ili gumba ili teksta unutar prozora. To uključuje sve što korisnik može vidjeti.

3. Kontroler

Kontroler ažurira i modele i prikaze. Ono prihvaća unos i vrši odgovarajuće ažuriranje. Na primjer, kontroler može ažurirati model mijenjanjem atributa lika u videoigri. Može mijenjati prikaz prikazom ažuriranog znaka u igri.

Tri dijela MVC-a međusobno su povezana (vidi dijagram). Pogled prikazuje model za korisnika. Upravljač prihvaća korisnički unos i ažurira model te ga u skladu s tim prikazuje. Iako MVC nije potreban u dizajnu aplikacija, mnogi programski jezici i IDE[10] podržavaju MVC arhitekturu, što ga čini uobičajenim izborom za programere.

[9] Izvor: https://techterms.com/definition/mvc
[10]Integrirano razvojno okruženje (IDE) softverska je aplikacija koja računalnim programerima pruža sveobuhvatne pogodnosti za razvoj softvera.

6.1. Značajke MVC-a

- Jednostavan, visoko ispitivan, proširiv i prilagodljiv okvir
- Nudi potpunu kontrolu nad HTML-om kao i nad vašim URL-ovima
- Koristite postojeće značajke koje pružaju ASP.NET, JSP, Django, itd.
- Jasno razdvajanje logike: model, prikaz, kontroler. Odvajanje zadataka aplikacije, tj. poslovna logika, Ul logika i ulazna logika
- Usmjeravanje URL-ova za SEO prijateljske URL-ove. Snažno mapiranje URL-ova za razumljive i pretražive URL-ove
- Podrška za razvoj testnog pokreta (TDD)

Kao primjer MVC-a da bi ga lakše razumjeli možemo navesti mehanizam vožnje automobila. Svaki se automobil sastoji od tri glavna dijela.

1. Pogled = korisničko sučelje: (poluga zupčanika, paneli, volan, kočnica itd.)
2. Mehanizam kontrolera (motor)
3. Model - Skladištenje (Spremnik za benzin ili Diesel)

Automobil vozi iz motora, a gorivo donosi iz skladišta, ali radi samo na spomenutim uređajima korisničkog sučelja.

6.2. Prednosti MVC-a

- Jednostavno održavanje koda lako se proširiti i rasti
- Komponenta MVC modela može se testirati odvojeno od korisnika
- Lakša podrška za nove vrste klijenata
- Razvoj različitih komponenti može se izvoditi paralelno.
- Pomaže da izbjegnete složenost dijeljenjem aplikacije u tri jedinice. Model, pogled i kontroler.

- Koristi se samo uzorak prednjeg kontrolera koji obrađuje zahtjeve web aplikacija putem jednog kontrolera.
- Nudi najbolju podršku za testni razvoj.
- Odlično djeluje za web aplikacije koje podržavaju veliki timovi web dizajnera i programera.
- Pruža čisto razdvajanje problema (SoC).
- Prijateljska optimizacija za tražilice (SEO).
- Svi razvrstani i objekti su međusobno neovisni, tako da ih možete zasebno testirati.
- MVC omogućuje logično grupiranje povezanih radnji na kontroleru zajedno.

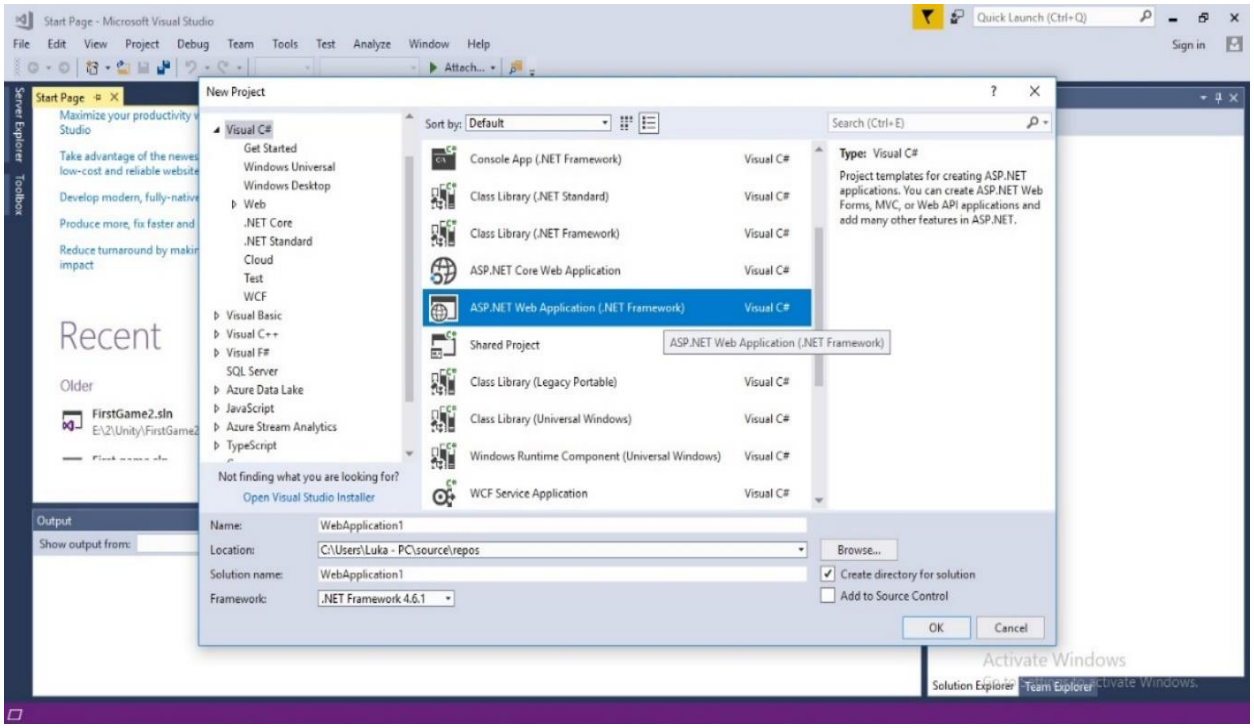
6.3. Nedostaci MVC-a

- Teško je čitati, mijenjati, testirati jedinice i ponovo koristiti ovaj model
- Okvirna navigacija može vremenski biti složena jer uvodi nove slojeve apstrakcije zbog kojih se korisnici moraju prilagoditi kriterijima dekompozicije MVC.
- Nema formalne potpore za potvrdu
- Povećana složenost i neefikasnost podataka
- Teškoća korištenja MVC-a s modernim korisničkim sučeljem
- Potrebno je više programera za paralelno programiranje.
- Potrebno je poznavanje više tehnologija.
- Održavanje puno koda u kontroleru

6.4. Stvaranje MVC aplikacije

Da bi stvorili MVC aplikaciju upotrijebljen je program *Microsoft Visual Studio*. U *Microsoft Visual Studiu* je potrebno pokrenuti novi projekt i odabrati ime projekta. Nakon odabiranja imena projekta potrebno je odabrati tip aplikacije. U našem slušaju odabiremo ASP.NET [11]Web Aplication.

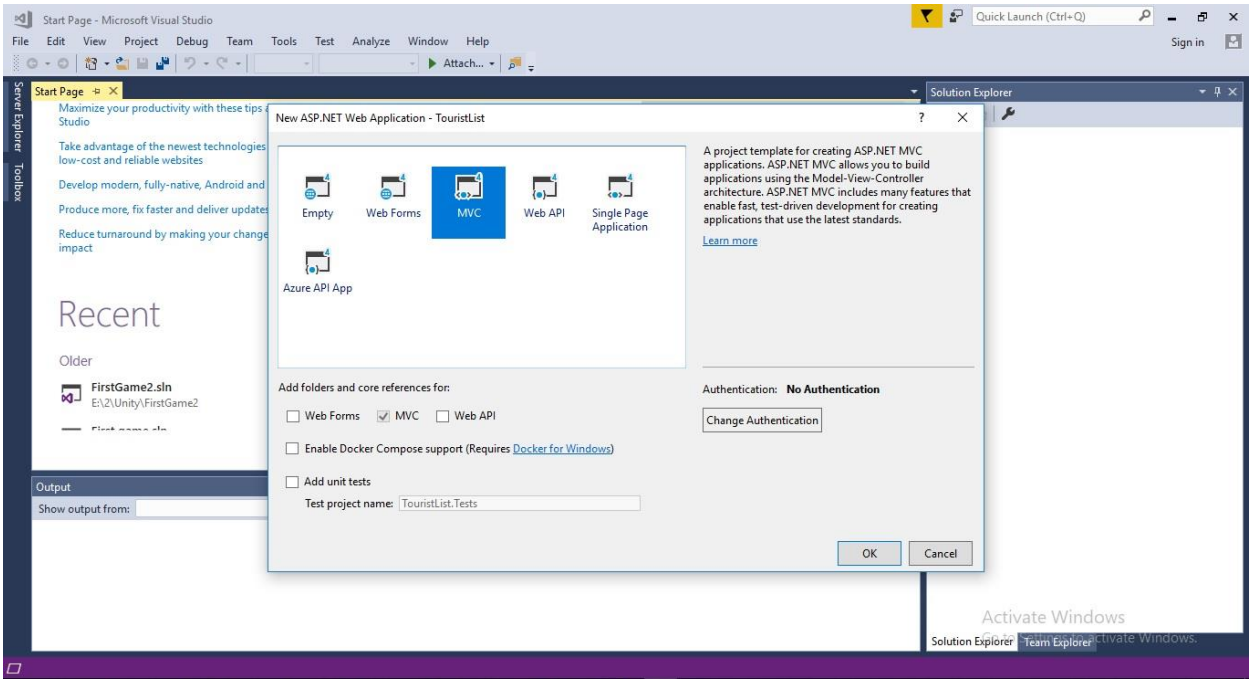
Slika 12. Prikaz odabiranja tipa aplikacije u programu *Microsoft Visual Studio*



Nakon što odaberemo ASP.NET Web Application i nakon što odredimo ime aplikacije otvara se novi prozor u kojem će nam *Microsoft Visual Studio* ponuditi više vrsta Web aplikacija, a mi odabiremo MVC.

[11] ASP.NET - platforma za razvojne programere koja se sastoji od alata, programskih jezika i knjižnica za izgradnju mnogih različitih vrsta aplikacija.

Slika 13. Odabir MVC aplikacije



Nakon nekoliko trenutaka što program učita i izvrši sva učitavanja, kreira se naše MVC sučelje u kojem ćemo krenuti dizajnirati našu web aplikaciju.

Aplikacija se pokreće u nekim od pretraživača, a najčešće je to zadani Google Chrome. Reklo bi se da je MVC aplikacija razvijena na čak najjednostavniji način od bilo kojih drugih aplikacija. Programski kod se automatski generira i zato omogućava jednostavan pogled na sučelje i na pojedine zaslone.

7. Kreiranje i dizajn MVC Aplikacije

U prethodnom koraku aplikacija je stvorena, kod je automatski generiran i bez znanja programskog jezika, što predstavlja jednostavnost i lako korištenje MVC arhitekture, pa se tako lako može krenuti na kreiranje i dizajniranje MVC aplikacije.

U dizajniranju aplikacije potrebna su minimalna znanja HTML[12]-a i CSS[13]-a. U našem programu koji smo kreirali da bi vidjeli primjenu baza podataka koristili smo se najjednostavnijim tehnikama da bi motivirali i programera početnika, mladog entuzijasta, da krene s primjenom, učenjem i implementacijom sustava. Pa smo tako koristili najjednostavnije kodove u izradi web stranice.

Pomoću HTML-a i CSS-a je moguće promijeniti ili napraviti potpuno novi izgled aplikacije. Postupak nakon dizajnirane aplikacije je stvaranje modela i kontrolera programa. To znači da će se krenuti s izradom ključnih postupaka da bi aplikacija funkcionirala, što znači da izrađujemo *Model, View i Controller* (MVC).

7.1. Kreiranje Modela

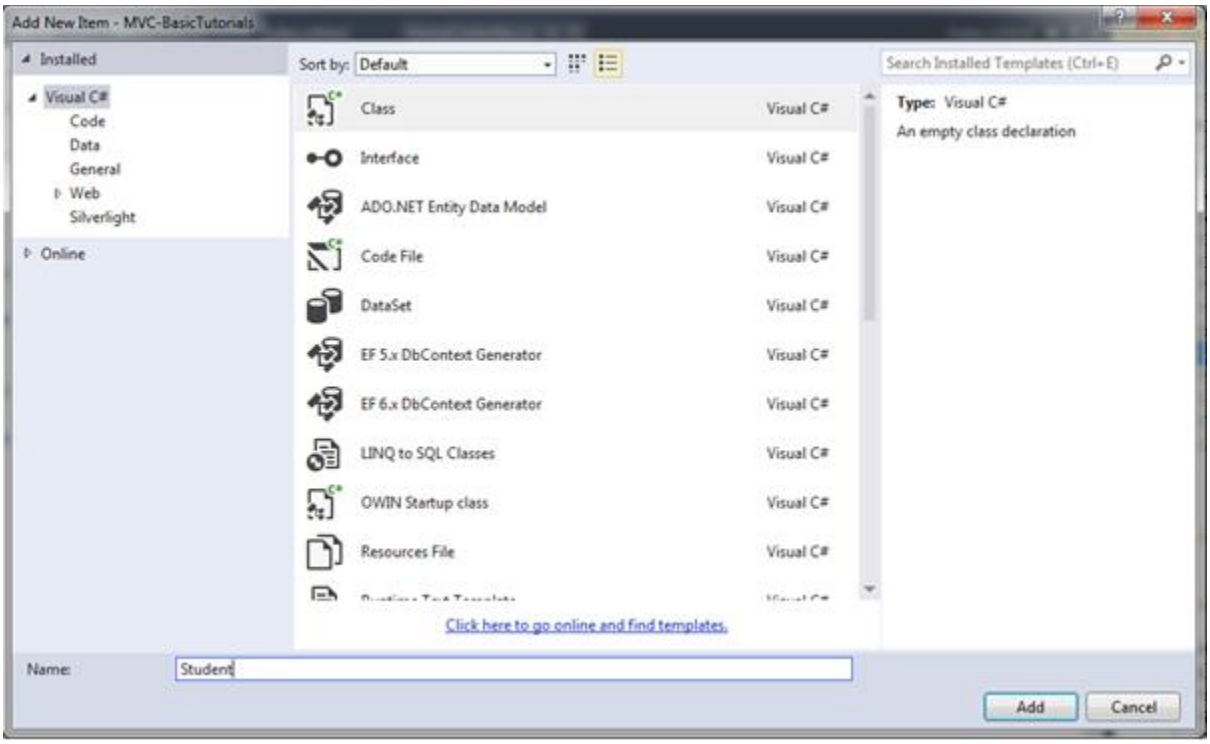
Model predstavlja podatke vezane uz domenu i poslovnu logiku u MVC arhitekturi. Održava podatke aplikacije. Objekti modela dohvaćaju i pohranjuju stanje modela u spremnost trajnosti poput baze podataka.

Model klasa sadrži podatke u javnim objektima. Sve klase modela nalaze se u mapi mape u strukturi mapa MVC.

[12] HTML - prezentacijski jezik za izradu web stranica
[13] CSS – koristi se za opis prezentacije dokumenta napisanog pomoću markup (HTML) jezika

Pogledajmo kako dodati klasu modela u ASP.NET MVC.

Slika 14. Kreiranje modela

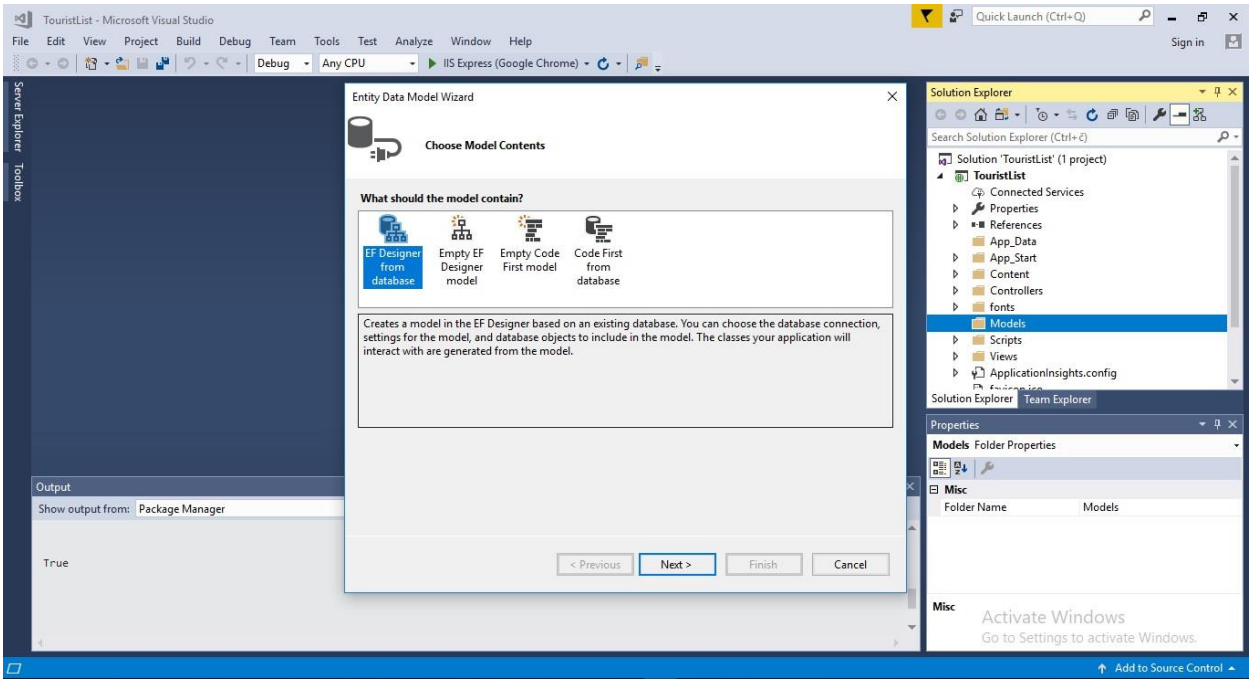


Izvor: https://www.tutorialsteacher.com/mvc/mvc-model

Otvorite naš prvi MVC projekt kreiran u prethodnom koraku u Visual Studiu. Desnim klikom na mapu Model -> Add -> kliknite na Class. U dijaloškom okviru Add New Item unesite naziv klase „Student“ kao što je prikazano na slici ili u našem slučaju mi smo upisali ime „Tourist“ kliknite Add.

Kada odaberemo tip i vrstu modela prikazuje se dijaloški okvir gdje se izabiru podaci koje taj model može svrstati. Postoje četiri opcije,a to su *Code First from database, Empty code First model, Empty EF designer model* te *Ef Designer from database*. Mi odabiremo EF Designer from database iz razloga što povlačimo sve podatke s lokalne baze podataka i cilj nam je da se one iščitaju vizualno u web aplikaciji.

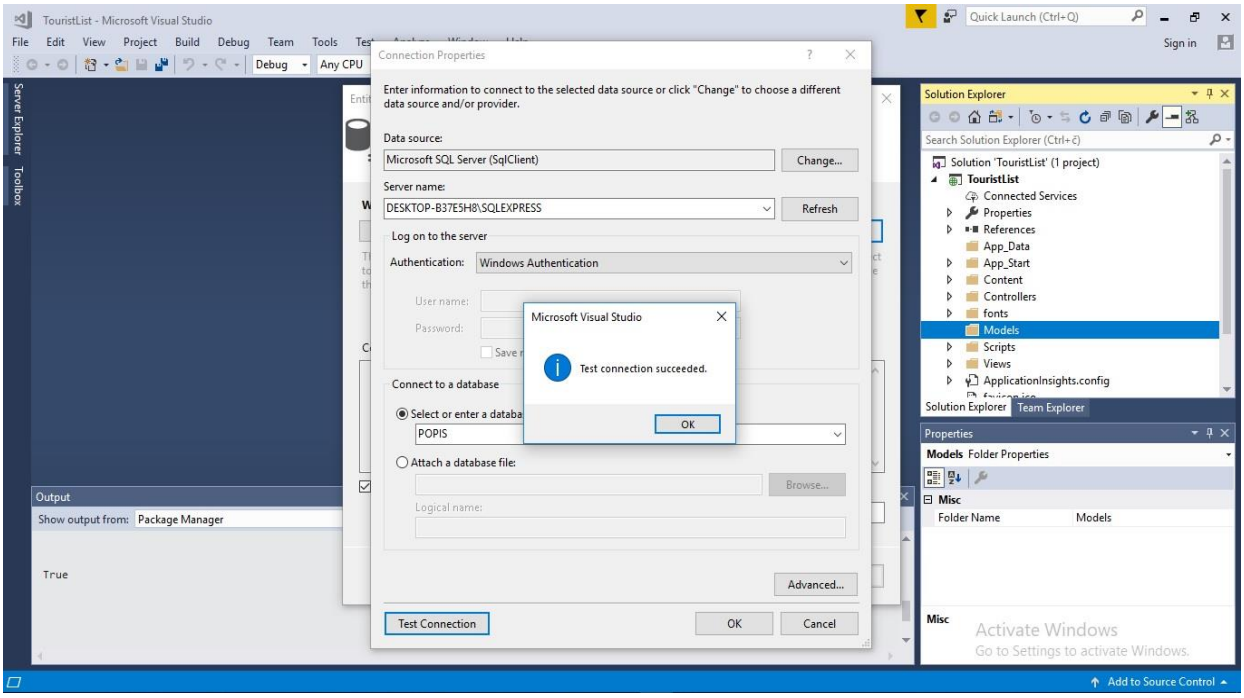
Slika 15. Primjer odabira EF Designer modela



Kada smo odabrali sadržaj modela otvorit će se dijaloški okvir pomoću kojeg ćemo povezati dosadašnju bazu podataka koju smo već ranije izradili. Klikom odabira na *New Connection* otvara se novi prozorčić i odabiremo Microsoft SQL Server za *Data Source*. Nakon toga klikom na *Server Name* otvorit će se lista već stvorenih baza podataka. Mi odabiremo naš server na kojem je izrađena baza podataka koju ćemo koristiti s Web Aplikacijom.

Nadalje, kada smo odabrali našu bazu podataka, u ovom slučaju koristimo bazu podataka imena „POPIS“, kliknemo na OK te označimo polje *Yes* i završili smo spajanje.

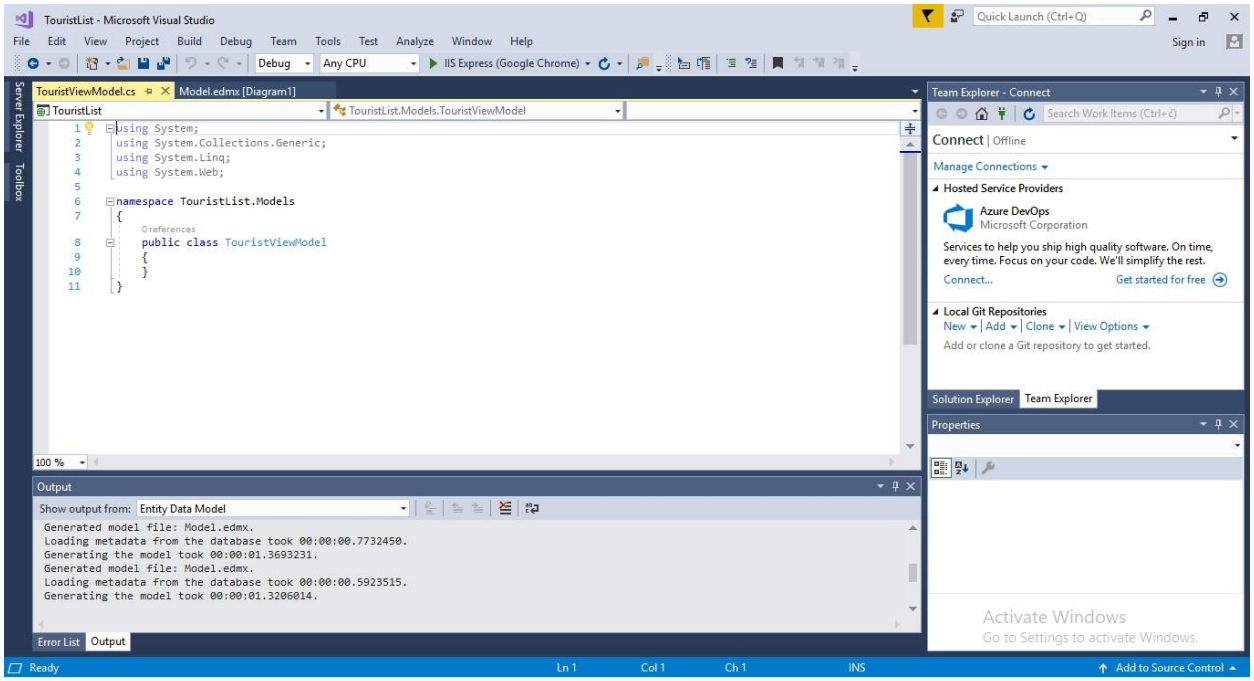
Slika 16. Spajanje baze podataka



Kada smo spojili bazu podataka automatski se generira dijagram baze podataka, u našem slučaju za Završni rad smo koristili najjednostavniju vezu, da bi svima omogućili što jasniji prikaz i funkcionalnost primjene baze podataka u stvarnom životu.

Otvorite naš prvi MVC projekt kreiran u prethodnom koraku u Visual Studiu. Desnim klikom na mapu Model -> Add -> kliknite na Class. U dijaloškom okviru Add New Item unesite naziv klase „Tourist“ kao što je prikazano na slici ili u našem slučaju mi smo upisali ime „Tourist“ kliknite Add.

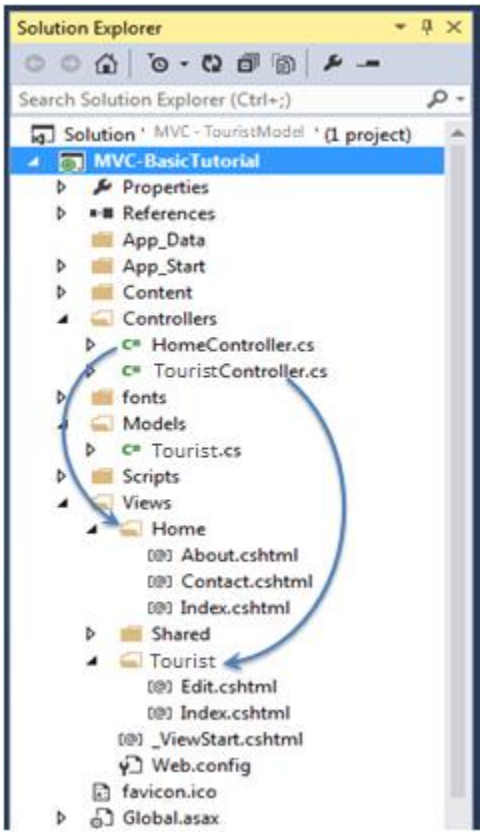
Slika 17. Primjer model view klase



7.2. Kreiranje View

View je korisničko sučelje. Prikaz prikazuje podatke s modela korisniku i omogućuje im da izmijene podatke. ASP.NET MVC prikazi pohranjuju se u mapu Views. Različite metode djelovanja jedne klase kontrolera mogu donijeti različite poglede, tako da mapa Views sadrži zasebnu mapu za svaki kontroler s istim nazivom kao kontroler, kako bi se uklopilo više pregleda. Na primjer, prikazi koji će biti prikazani bilo kojom od metoda akcije HomeController nalaze se u mapi Views> Home. Na isti način, prikazi koji će biti prikazani iz TouristController-a nalazit će se u mapi Views> Tourist mapa kao što je prikazano u nastavku.

Slika 18. Prikaz View mape za kontrolere

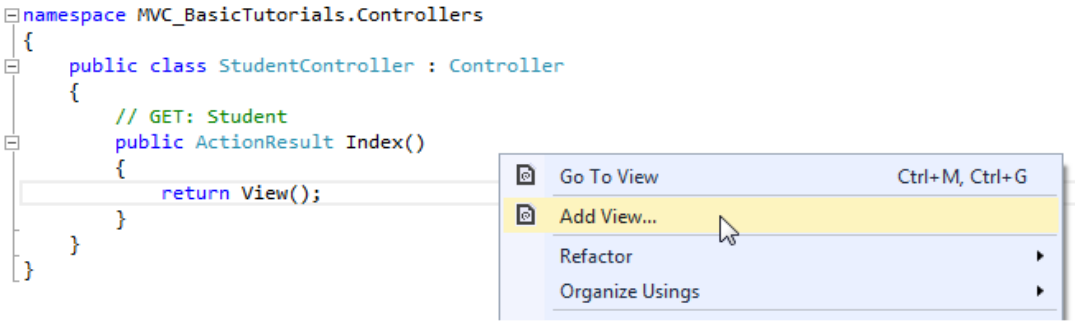


Izvor: https://www.tutorialsteacher.com/mvc/mvc-view

Već smo stvorili TouristController i studentsko model u prethodnom odjeljku. Sada ćemo stvoriti Studentski prikaz i razumjeti kako koristiti model u *View*.

Stvorit ćemo prikaz koji će biti prikazan indeksnom metodom TouristContollera. Dakle, otvorite klasu TouristController -> desnom tipkom miša kliknite unutar Index metode -> kliknite Add.

Slika 19. Kreiranje novog View-a

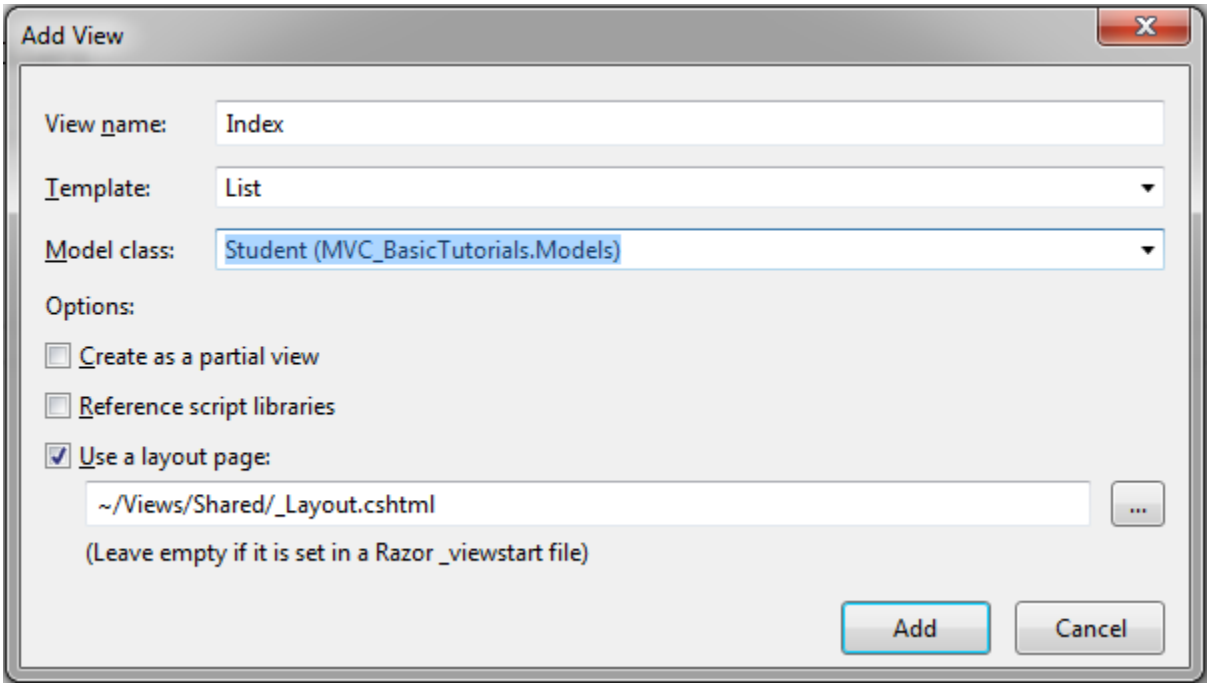


Izvor: https://www.tutorialsteacher.com/mvc/mvc-view

U dijaloškom okviru Dodaj prikaz zadržite naziv prikaza kao Indeks. Dobra je praksa da naziv vlasničkog pregleda ostane isti kao i naziv metode radnje kako ne biste morali izričito navesti naziv prikaza u akcijskoj metodi dok vraćate prikaz.

Odaberite predložak skela. Na padajućem izborniku predloška prikazat će se zadane predloške dostupne za prikaz Stvori, Obriši, Pojedinosti, Uredi, Popis ili Prazan prikaz. Odaberite predložak "Popis", jer želimo prikazati popis učenika u prikazu.

Slika 20. View

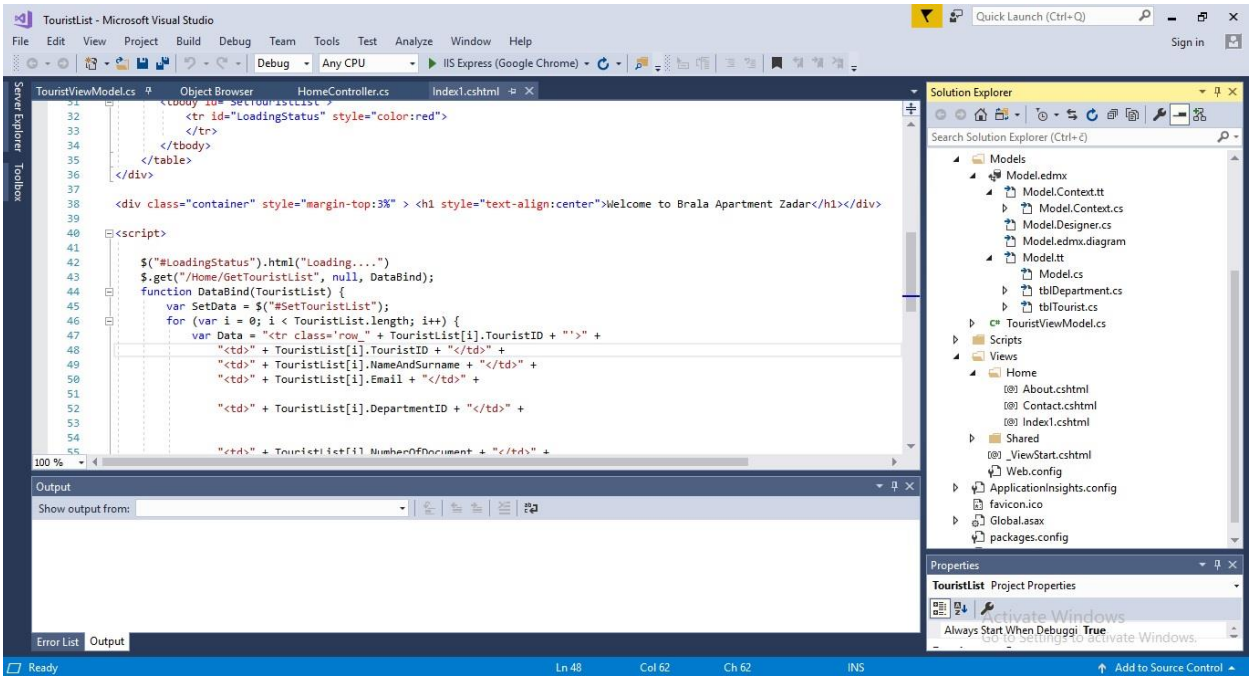


Izvor: https://www.tutorialsteacher.com/mvc/mvc-view

Označite potvrdni okvir "*Use a layout page*" i odaberite Layout.cshtml stranicu za ovaj prikaz, a zatim kliknite gumb *Add*. Kasnije ćemo vidjeti što je stranica za izgled, ali za sada ćemo to zadati za glavnu stranicu u MVC-u. Taj način će stvoriti indeksni prikaz pod View -> Tourist

Na sljedećoj slici ćemo vidjeti isječak koda Index.cshtml koji smo stvorili na način opisan u prethodnom tekstu.

Slika 21. Prikaz koda Index.cshtml



U gornjem prikazu indeksa, on sadrži i HTML kodove i britve. Izraz britve u liniji počinje sa @ simbolom, dok je @Html pomoćna klasa za generiranje HTML kontrola.

7.3. Kreiranje Controller

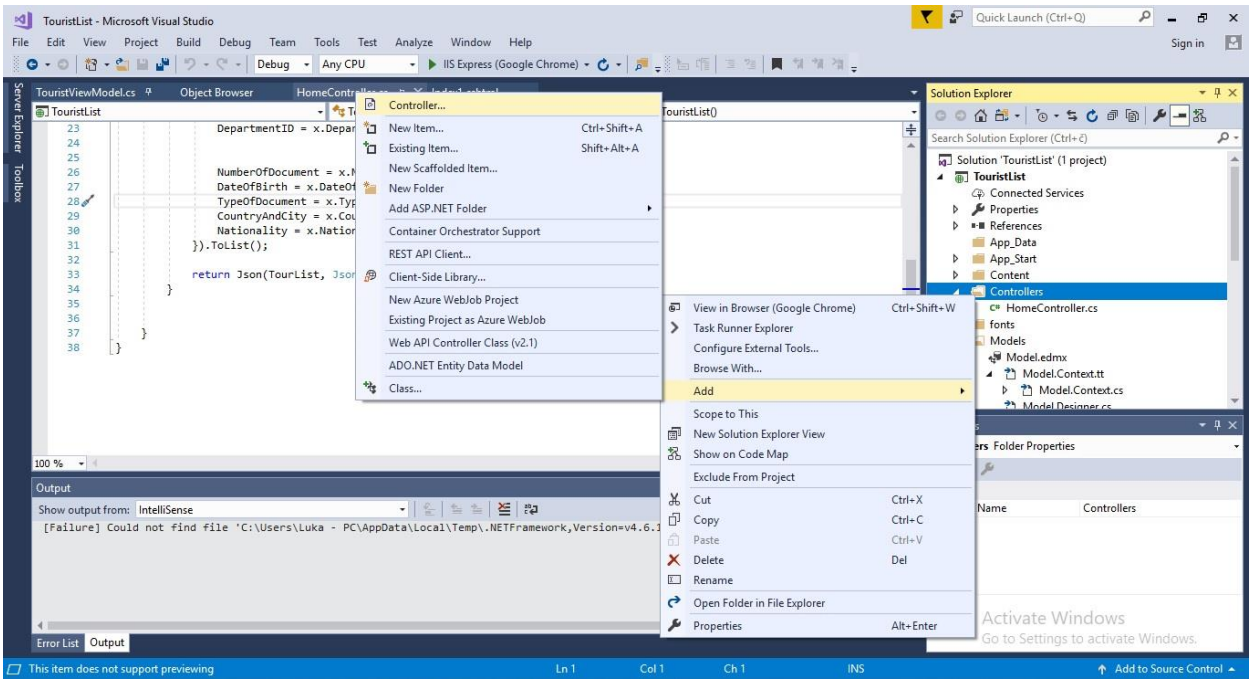
Nakon izrade modela izrađuje se kontroler, on se izrađuje tako što se u prozoru *Solution Explorer* klikne n mapu *Controllers* te se desnom tipkom miša klikne na *Add* pa zatim na *Controller* kao što ćemo objasniti i prikazati u sljedećem tekstu

Kontroler u MVC arhitekturi obrađuje sve dolazne URL zahtjeve. Controller je klasa, izvedena iz osnovne klase System.Web.Mvc.Controller. Klasa kontrolera sadrži javne metode koje se nazivaju Akcijske metode. Kontroler i njegova radna metoda obrađuje dolazne zahtjeve preglednika, dohvaća potrebne podatke modela i vraća odgovarajuće odgovore.

U ASP.NET MVC ime svake klase kontrolera mora se završiti riječju "Controller". Na primjer, kontroler za početnu stranicu mora biti HomeController, a kontroler za turista mora biti TouristController. Također, svaka klasa kontrolera mora biti smještena u mapi Controller strukture MVC mape. MVC će izbaciti pogrešku "Resurs se ne može pronaći" ako ne dodate "Controller" nazivu klase kontrolera.

Kada želimo dodati novi kontroler, u Visual Studio desnom tipkom miša kliknite mapu Controller -> odaberite Add -> kliknite na Controller .

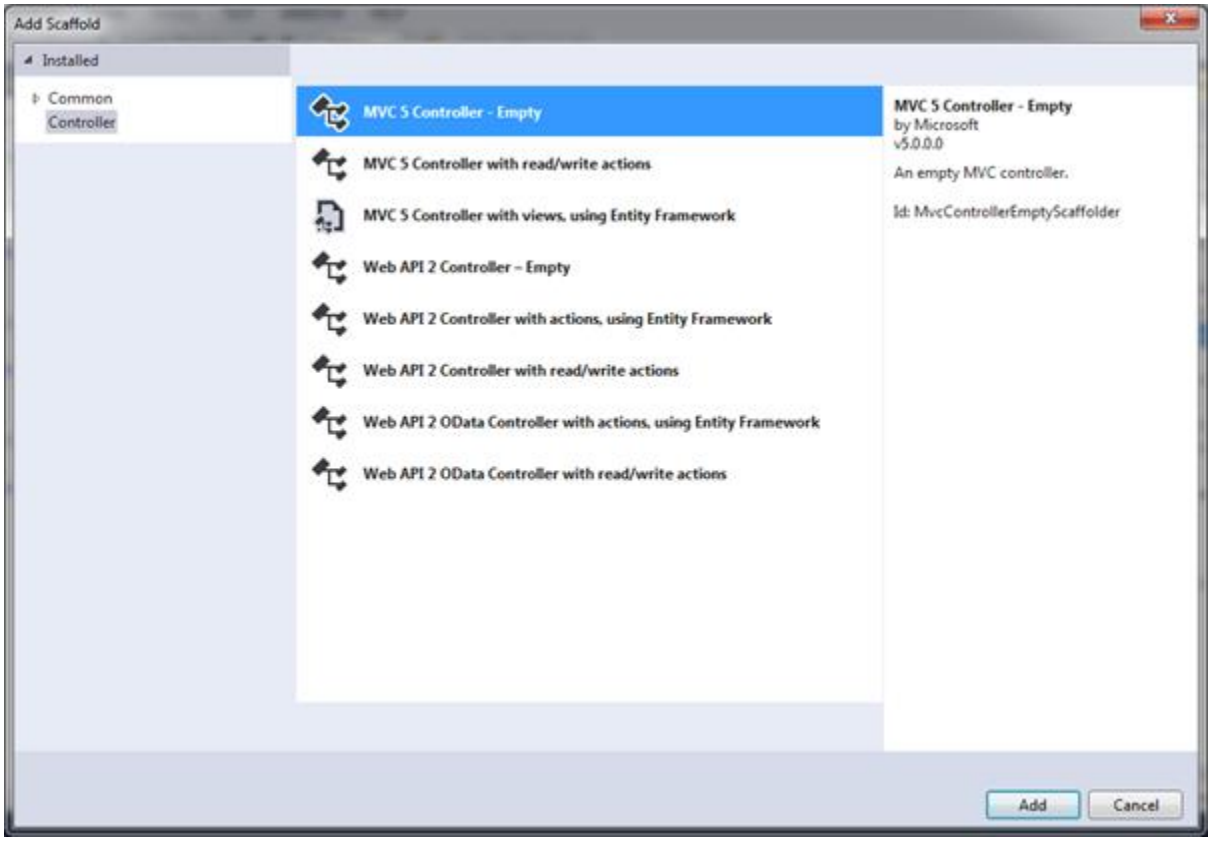
Slika 22. Dodavanje kontrolera



Nakon što smo kliknuli desni klik na *Controllers*, pa zatim na *ADD* i *Controller* program nam otvara jedan poseban dijaloški okvir „Add Scaffold[14]“ kao što je prikazano u nastavku.

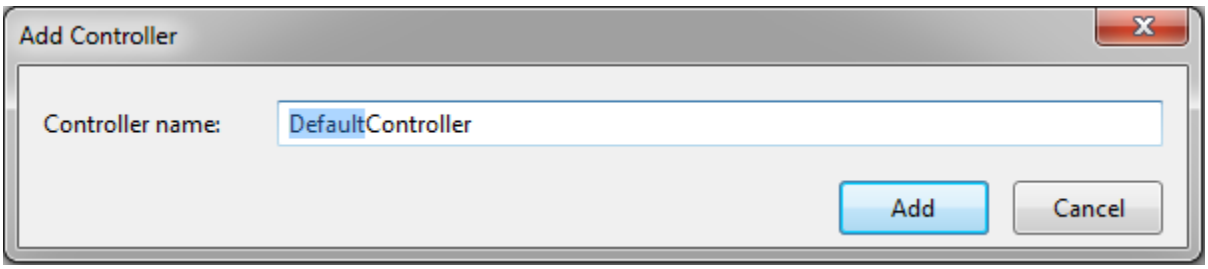
[14] Scaffold su okvir za automatsko generiranje koda za ASP.NET web aplikacije. Također, smanjuju vrijeme potrebno za izradu regulatora, prikaza itd.

Slika 23. Odabiranje novog kontrolera



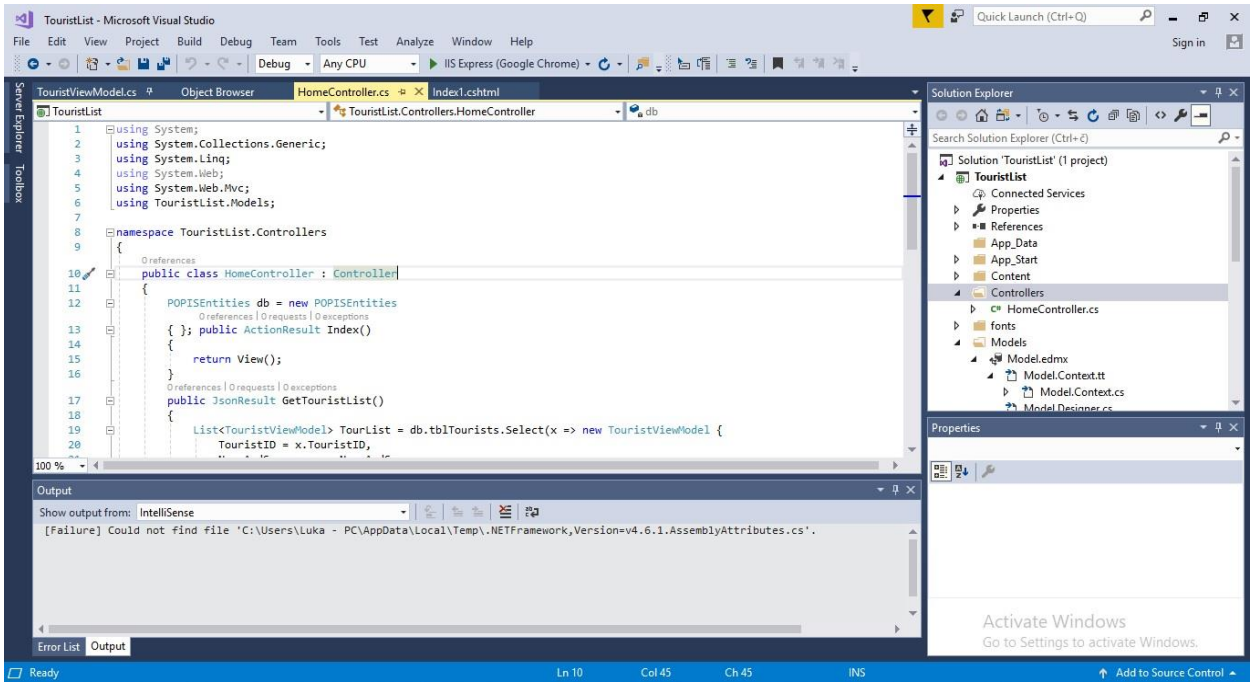
Dijaloški okvir sadrži različite predloške za stvaranje novog kontrolera, odaberite "MVC 5 kontroler - prazan" i kliknite Dodaj. Otvorit će se dijaloški okvir Add Controller kao što je prikazano u nastavku i promijenimo ime u željeno ime kontrolera, ali ne smijemo zaboraviti da ssvaki kontroler mora završavati s *Controller* na kraju imena.

Slika 24. Dodavanje imena kontrolera



Ovo će stvoriti klasu TouristController metodom indeksa u datoteci TouristController.cs u mapi Kontroleri, kao što je prikazano u nastavku.

Slika 25. Prikaz klase Tourist



Kao što možete vidjeti gore, klasa TouristController potječe iz klase Controller. Svaki kontroler u MVC-u mora poticati iz ove apstraktne klase kontrolera. Ova osnovna klasa kontrolera sadrži pomoćne metode koje se mogu koristiti u razne svrhe.

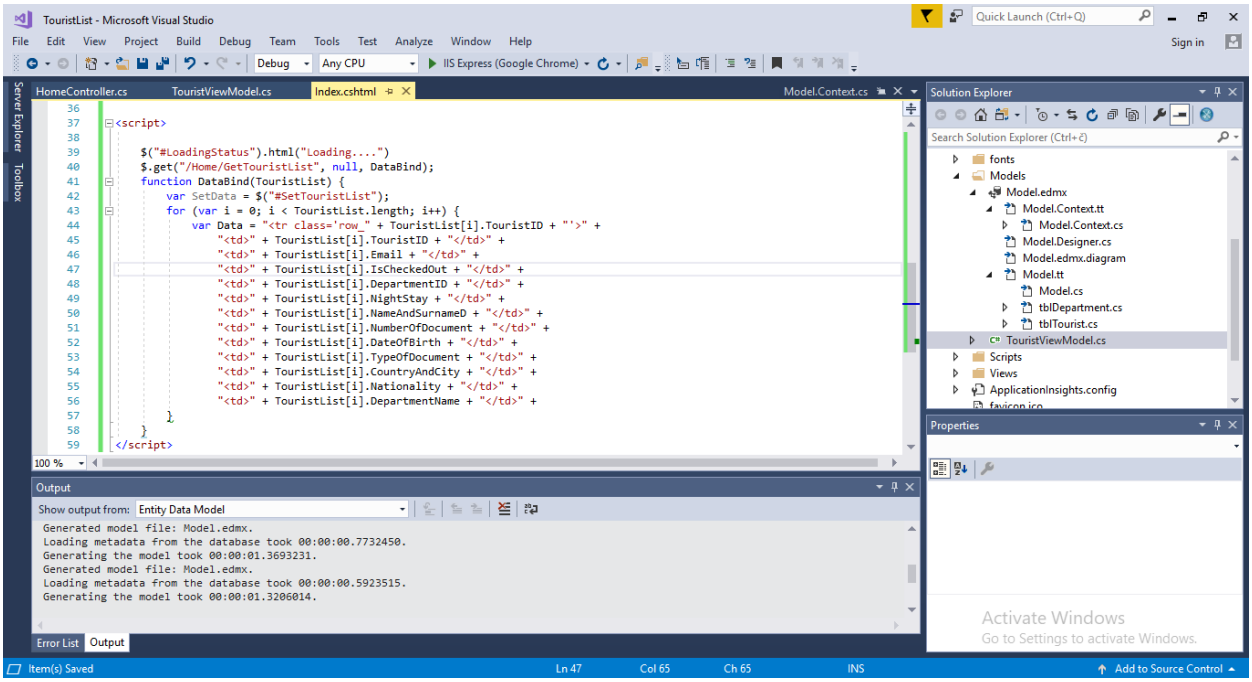
Kontroler obrađuje dolazne zahtjeve za URL-om te MVC usmjeravanjem šalje zahtjev odgovarajućem kontroleru i metodi djelovanja na temelju URL-a i konfiguriranih ruta. Sve javne metode u klasi Controller nazivaju se *Action* metode. Klasa kontrolera mora biti izvedena iz klase System.Web.Mvc.Controller te naziv klase kontrolera mora završiti s "*Controller*". Novi kontroler može se stvoriti pomoću različitih predložaka, a isto tako možete stvoriti i prilagođeni predložak *scaffold-a*.

8. Web Aplikacija

Nakon što sve zbrojimo, uparimo i kada sve profunkcionira, *View* dio MVC arhitekture stvara gotovi vizualni pogled koji gledamo na ekranu, *Controllers* dio upravlja s kodom i *Model* dijelom MVC arhitekture koji daje funkcionalnu Web Aplikaciju. S kreiranom bazom podataka na početku, kreiranja tablica, stvaranja dijagrama, definiranjem entiteta i njihovih atributa, pa sve do kraja stvaranja aplikacije MVC arhitekturom dajemo sliku primjene baza podataka.

U sučelje aplikacije zadali smo HTML - jezikom izgled stranice, definirali tablicu te spojili entitete s bazom podataka, odnosno povezali entitete baze podataka s web aplikacijom. Tako smo dobili jasnu suradnju između baza podataka i web aplikacije.

Slika 26. Povezivanje entiteta baze podataka s Web Aplikacijom

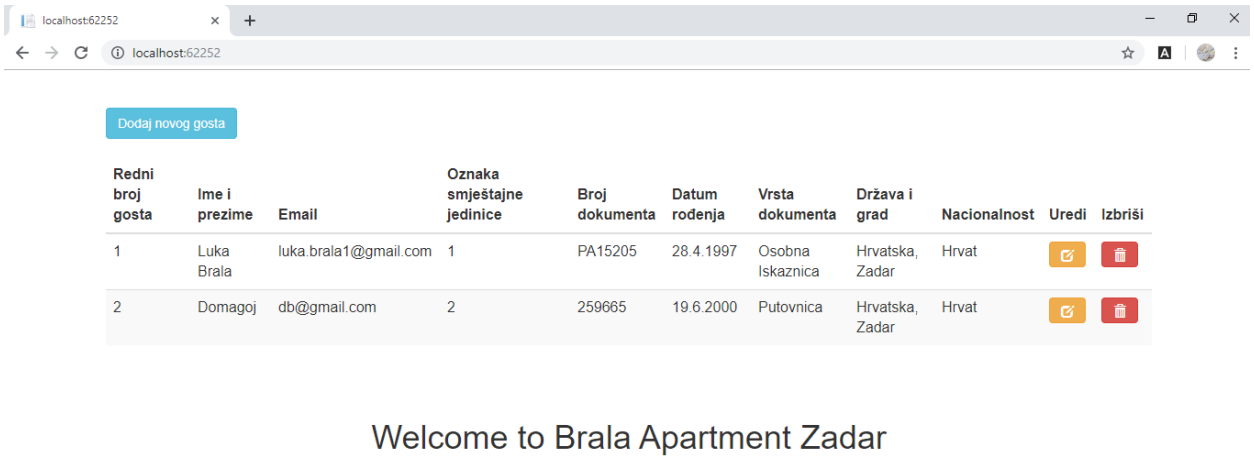


Pokrenemo li web aplikaciju u nekom od preglednika on će dati jasnu sliku podataka koji su u toj bazi podataka. Konkretno za Završni rad *Primjena baza podataka* sam odlučio na jednostavan način prikazati popis gostiju u nekom objektu.

Klikom na *Dodaj novog gosta* dodajemo nove podatke u bazu podataka. S tim načinom se upisuju novi podaci i spremaju u bazu podataka. Također, klikom na ikonu *Uredi* dali smo korisniku mogućnost uređivanja podataka koje je upisao u bazu podataka, a također jednostavnim klikom na

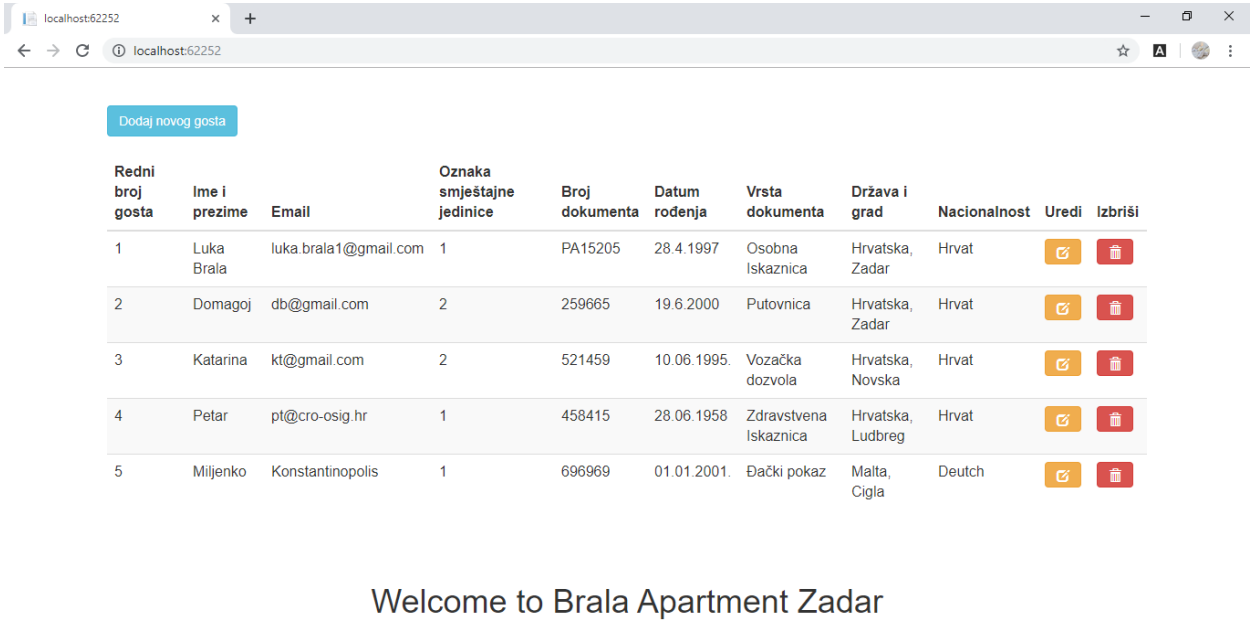
ikonu *Izbriši* omogućili smo također kontrolu nad bazom podataka te će tako izbrisati podatak upisan pod tim redom.

Slika 27. Web Aplikacij



Klikom na link *Dodaj novog gosta* smo omogućili lak upis novih podataka u bazu podataka.

Slika 28. Ažurirana Web Aplikacija



9. Zaključak

U današnje vrijeme bez aplikacija, programa i web sadržaja život je ne zamisliv. S obzirom na to da je život sve više informatiziran i ubrzan neophodno je ubrzati cijeli sustav. Aplikacija izrađena MVC arhitekturom je idealna za brz razvitak Web Aplikacije zato što se cijeli sustav može razviti uz ne veliko znanje programiranja, HTML-a i CSS-a. MVC arhitektura dopušta da se svaki dio aplikacije razvija zasebno, što uvelike omogućava brže i efikasnije stvaranje. Isto tako, u Završnom radu je prikazana primjena baza podataka u stvarnom životu u spoju sa MVC arhitekturom i izradom Web Stranice koja međusobno komunicira s bazom. Ova aplikacija nam daje jednostavnu i čitku strukturu podataka upisanih u bazu podataka te prikazuje kako se može na jasan i efikasan način upravljati podacima. Tako smo omogućili optimalno i brzo rješenje za upis, ažuriranje i brisanje podataka iz baze podataka te tako pokazali njenu primjenu.

Literatura

1. http://genderi.org/rad-s-bazom-podataka-u-net-okruenju-model-relacijske-baze-poda.html (Pristup: 03.07.2019.)
2. https://learntocodewith.me/posts/sql-guide/ (Pristup: 10.07.2019.)
3. www.tutorialsteacher.com/mvc (Pristup: 11.07.2019.)
4. https://techterms.com/definition/mvc (Pristup: 11.07.2019.)
5. CET. 2013. *Microsoft SQL server 2012 programiranje*
6. Hamilton B. 2006. *Programiranje SQL Server 2005*. Zagreb: Dobar plan
7. Liberty, J. 2007. *Programiranje na jeziku C#.* Zagreb: Mikro knjiga.
8. Vujnović, R. 1995. *SQL i relacijski model podataka*. Zagreb: Znak

Popis slika

1. Slika 1. Dohvaćanje i izlistavanje podataka tablice Tourist iz baze podataka POPIS
2. Slika 2. Primjer CODASYL modela baza podataka
3. Slika 3. Relacijski model, povezivanje tablica s među-tablicom.
4. Slika 4. Primjer tablice za pohranjivanje podataka o knjigama u knjižnici
5. Slika 5. Primjer DDL naredbi
6. Slika 6. Primjer DML naredbi
7. Slika 7. Primjer ER dijagrama baze podataka evidencije gostiju
8. Slika 8. Pokretanje programa Microsoft SQL Server
9. Slika 9. Primjer kreiranja tablice
10. Slika 10. Dijagram baze podataka generiran spajanjem tablica
11. Slika 11. Dijelovi Model-Viev-Controller
12. Slika 12. Prikaz odabiranja tipa aplikacije u programu Microsoft Visual Studio
13. Slika 13. Odabir MVC aplikacije
14. Slika 14. Kreiranje modela
15. Slika 15. Primjer odabira EF Designer modela
16. Slika 16. Spajanje baze podataka
17. Slika 17. Primjer model view klase
18. Slika 18. Prikaz View mape za kontrolere
19. Slika 19. Kreiranje novog View-a
20. Slika 20. View
21. Slika 21. Prikaz koda Index.cshtml
22. Slika 22. Dodavanje kontrolera
23. Slika 23. Odabiranje novog kontrolera
24. Slika 24. Dodavanje imena kontrolera
25. Slika 25. Prikaz klase Tourist
26. Slika 26. Povezivanje entiteta baze podataka s Web Aplikacijom
27. Slika 27. Web Aplikacija

28. Slika 28. Ažurirana Web Aplikacija