SVEUČILIŠTE U ZAGREBU

**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. 4001

**Web-aplikacija za planiranje putovanja**

Ante Tomić

Zagreb, lipanj 2015.

SVEUČILIŠTE U ZAGREBU

**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. 4001

**Web-aplikacija za planiranje putovanja**

Ante Tomić

Zagreb, lipanj 2015.

Izvornik

Posebno se zahvaljujem mentoru doc. dr. sc. Igoru Mekteroviću na prihvaćanju teme, razumijevanju, strpljenju, idejama i poticajnim sugestijama prilikom izrade ovog završnog rada.

Sadržaj

[1. Uvod 1](#_Toc421186200)

[2. Opis problema i korisnički zahtjevi aplikacije 3](#_Toc421186201)

[2.1. Opis zadatka 3](#_Toc421186202)

[2.2. Analiza slučajeva korištenja aplikacije 4](#_Toc421186203)

[3. Arhitektura i dizajn sustava 6](#_Toc421186204)

[3.1. Web-aplikacija 6](#_Toc421186205)

[3.2. Baza podataka 7](#_Toc421186206)

[4. Pregled korištenih tehnologija u izradi aplikacije 12](#_Toc421186207)

[4.1. .NET Framework (radni okvir) 12](#_Toc421186208)

[4.2. ASP.NET radni okvir 14](#_Toc421186209)

[4.2.1. ASP.NET MVC (Model-View-Controller) 14](#_Toc421186210)

[4.3. Entity Framework 15](#_Toc421186211)

[4.4. JavaScript i jQuery 16](#_Toc421186212)

[4.4.1. JavaScript 16](#_Toc421186213)

[4.4.2. jQuery 16](#_Toc421186214)

[5. Izrada aplikacije i opis rješenja 17](#_Toc421186215)

[6. Zaključak 17](#_Toc421186216)

[7. Literatura i reference 18](#_Toc421186217)

[8. Sažetak 19](#_Toc421186218)

[9. Summary 19](#_Toc421186219)

[10. Ključne riječi 19](#_Toc421186220)

# Uvod

Pojam aplikacije može se interpretirati na više načina, ali je nama najvažnija računalna interpretacija pojma. Ona nam kaže da je aplikacija program ili softver koji je stvoren da radi određeni posao. Postoje mnoge vrste aplikacija te su one podijeljene na obrazovne, simulacijske, zabavne, simulacijske i mnoge druge vrste. Nas najviše zanimaju web-aplikacije bilo kakve vrste koje su zapravo računalne aplikacije kojima se pristupa preko internetskih preglednika pomoću Interneta ili bilo koje druge mreže.

Razvoj web-aplikacija počinje razvojem samog Weba, a nastavlja se tijekom 90-ih godina prošlog stoljeća i traje i do današnjeg dana. One koriste web-dokumente napisane u standardnim formatima poput HTML-a ili JavaScripta koji su podržani od strane većine internetskih preglednika. Razvoj i korištenje tehnologija poput Flasha, Silverlighta, JavaScripta i mnogih drugih omogućuje korištenje posebnih metoda i značajki pojedine tehnologije koje možda inače ne bi bile dostupne.

Web-aplikacije su sve popularnije zbog sve veće prisutnosti internetskih preglednika u različitim platformama što olakšava pristup bilo kojoj web-aplikaciji koja je dostupna na izabranoj mreži. Takve aplikacije također koriste internetski preglednik kao klijent koji se povremeno naziva tanki klijent jer internetski preglednik sadrži samo sloj prezentacijske logike. Glavni razlog popularnosti web-aplikacija je mogućnost korištenja bez distribucije i instalacije bilo kakvih dodatnih aplikacija na mogućim klijentskim računalima što omogućuje podršku na više platformi.

Svaka se aplikacija sastoji od servisa koji su udruženi u rješenje te se definiraju kao dio aplikacijske logike koji uključuje metode za implementaciju operacije, funkcije ili transformacije. Servisi su pridruženi akcijama su ih je moguće pristupiti preko mreže pomoću objavljenog sučelja koje sadrži specifikaciju sučelja. Korisniku nije važna implementacija servisa, nego mu je važnija sposobnost servisa da obavi traženu radnju. Jedan se servis obično sastoji od korisničkih, poslovnih, podatkovnih i sistemskih servisa od kojih svaki dio pruža određenu funkcionalnost.

Servisi su organizirani u rješenje pomoću odabrane aplikacijske arhitekture. Aplikacijska se arhitektura sastoji od definicija, pravila i veza koje formiraju strukturu aplikacije, ali ne sadrži implementacijske detalje. Neke od najkorištenijih aplikacijskih arhitektura su: klijent/server, višeslojna, nepamteća, pričuvna i višeslojna klijent-pričuvna-nepamteća-pričuvna-server arhitektura. Aplikacije su obično podijeljene u logičke cjeline nazvane slojevi od kojih je svakom sloju pridružena određena uloga.

Najkorištenija arhitektura, koja se koristila u izradi ovog završnog zadatka, je višeslojna arhitektura. Ona je napredna verzija klijent/server arhitekture koja se sastoji od hijerarhijskih slojeva u koje su spremljeni odgovarajući razni servisi. Najčešće se koristi troslojna arhitektura koja se sastoji od prezentacijskog, poslovnog i podatkovnog sloja. Slojevi enkapsuliraju servise i štite jedan servis od drugog uz pružanje pojednostavljenog kompleta sučelja za dijeljena sredstva. Prednosti ove arhitekture su poboljšana skalabilnost sustava, visoka sigurnost, mogućnost uravnoteživanja servisa između dostupnih resursa i raznih slojeva te mogućnost implementacije sigurnost na definiranim granicama uz minimalni utjecaj na komunikacije između servisa unutar sistema, dok su neki od nedostataka povećanje općih troškova, kompleksnosti i latentnosti.

Cilj je ovog završnog rada bila izrada web-aplikacije za planiranje putovanja koja bi pokrila sve bitne dijelove aplikacije poput objave sadržaja, spajanja sadržaja u stranicu, pregleda objavljenih sadržaja i stranica, dijeljenja i mnogih drugih dijelova. S obzirom na širinu ovog područja, neki dijelovi nisu toliko opširno uzeti u obzir, ali je aplikacija napravljena što realističnije i detaljnije te je ostavljena mogućnost za buduće nadogradnje sustava.

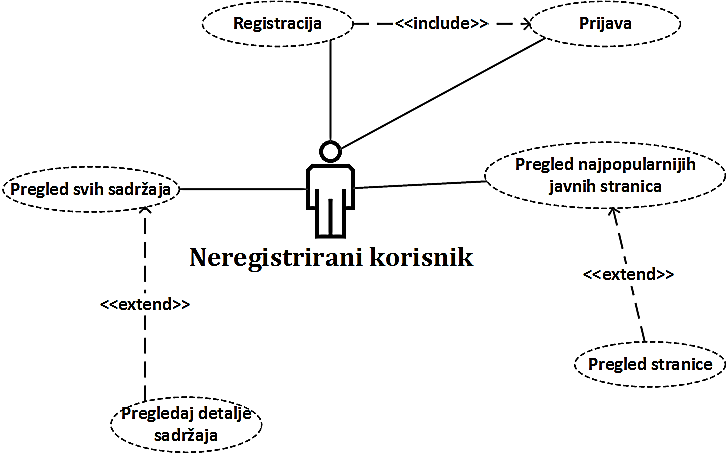
# Opis problema i korisnički zahtjevi aplikacije

## Opis zadatka

Aplikacija za upravljanje putovanjima je web-aplikacija koja korisnicima omogućuje izradu virtualnih kolaža ili fascikala u obliku stranica u kojima bi se nalazili sadržaji koje je korisnik izdvojio ili odabrao. Cilj takvih stranica je isplanirati putovanje slaganjem raznih članaka i drugih vrsta sadržaja u jednu cjelinu podijeljeno po lokacijama čime bi imao sve zanimljive sadržaje sklopljene na jedno mjesto (stranicu). Nakon što je stranica stvorena, korisnik ju može zadržati za sebe, podijeliti sa svim korisnicima i posjetiteljima ili odabranom grupom korisnika.

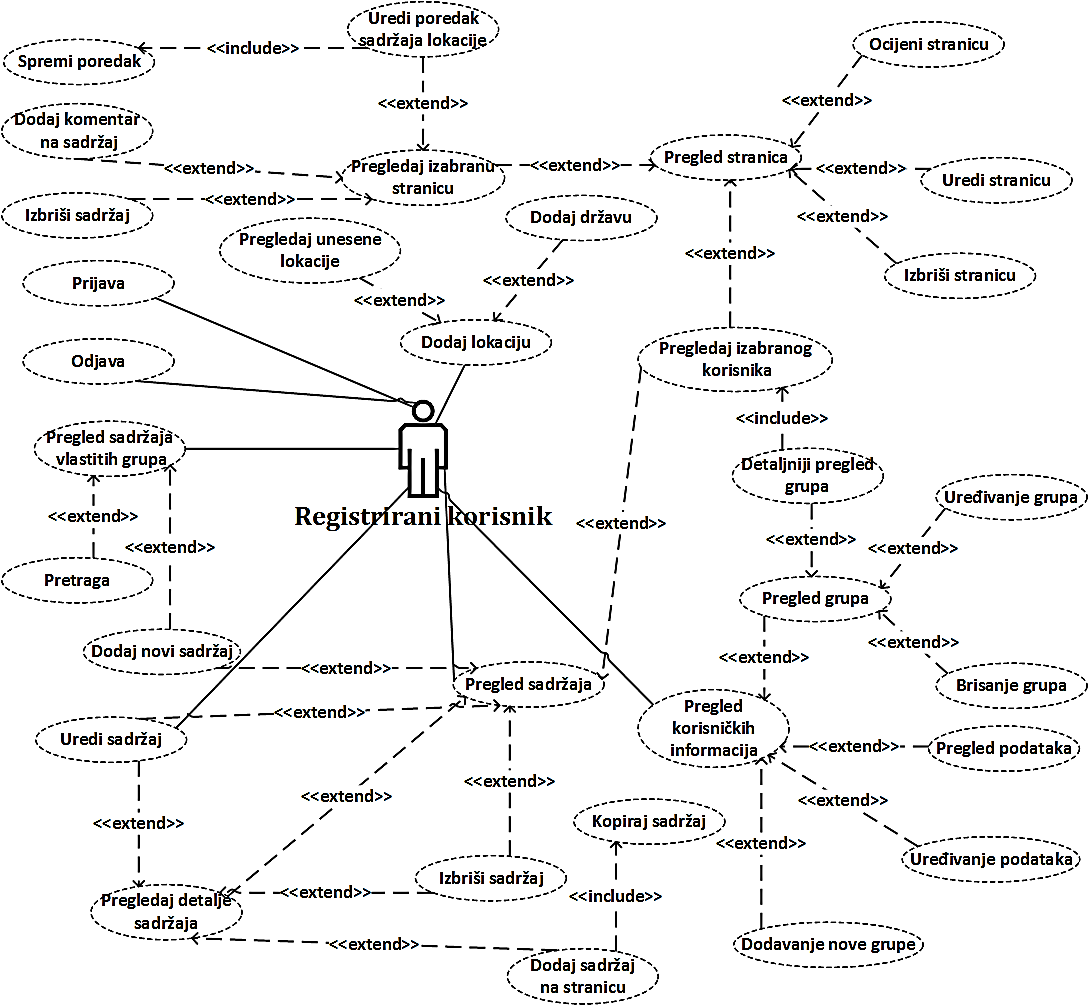
Dolaskom na web-aplikaciju korisnik vidi početnu stranicu s prikazanim svim sadržajima te najpopularnijim i najbolje ocijenjenim stranicama. Klikom na pojedini sadržaj može pregledati sve informacije o sadržaju, a također može pregledati i dostupne top liste. Gostu su dostupne sve stranice koje imaju postavke namještene tako da su vidljive svim korisnicima što u ovom slučaju uključuje i goste. Ako je korisniku stranice potrebno nešto više od onoga što je dostupno bilo kojem gostu, onda se može prijaviti ili registrirati čime dobiva sve mogućnosti koje imaju i registrirani korisnici. Čim se korisnik prijavi, aplikacija ga odmah odvede na početnu stranicu s prikazanim sadržajima članova korisnikovih grupa. Može pregledati njihove objavljene sadržaje te željene dodati na neku od postojećih stranica ili može napraviti novi sadržaj koji autor ili bilo koji drugi registrirani korisnik mogu dodavati na bilo koju stranicu. Korisnik može po želji dodati i novu stranicu na koju može dodavati nove ili postojeće sadržaje koje može uređivati i poredati, dok na postojećim stranicama može isto napraviti.

## Analiza slučajeva korištenja aplikacije

Korisnike stranice možemo podijeliti u dva dijela: goste (neregistrirane ili neprijavljene korisnike) i registrirane korisnike. Neregistrirani gost se može prijaviti, registrirati, pregledavati sve sadržaje, detalje o odabranom sadržaju ili pregledavati najpopularnije javne stranice iz top lista. Na slici 2.2.. je prikazan dijagram obrazaca uporabe za neregistriranog korisnika (gosta) za sve opisane ili spomenute obrasce uporabe.

Registracijom ili prijavom neregistrirani korisnik postoje registrirani korisnik. Osim svega što je dostupno neregistriranom korisniku, registrirani korisnik prijavom sa svojim registriranim korisničkim imenom i lozinkom dobiva sve mogućnosti koje ima svaki registrirani korisnik. Njemu se omogućuju kronološki pregled svih sadržaja članova grupa koje je on napravio, pretraživanje sadržaja prema odabranim kriterijima, pregled svih vlastitih sadržaja, pregled stranica na kojima je korisnik jedan od uređivača i koje može vidjeti po pravilima privatnostima postavljenim za svaku moguću stranicu, pregled grupa s pridruženim članovima koje je korisnik napravio, dodavanje sadržaja, stranica i grupa, pregled i uređivanje korisničkih podataka, dodavanje lokacija i države kad god je to potrebno (ZAVRŠITI)

Korisnik za bilo koji sadržaj može pregledavati detalje i dodavati odabrani sadržaj na jednu od vlastitih stranica, dok za bilo koji vlastiti sadržaj može uređivati sve stavke pridružene sadržaju i brisati objavljeni sadržaj. Za bilo koju stranicu korisnik može ocjenjivati stranicu i ostavljati komentare za pojedini sadržaj, dok za stranice gdje je korisnik jedan od sudionika može uređivati podatke o stranici, pregledavati detalje gdje može uređivati poredak sadržaja po lokaciji i uređivati poredak i brisati stranicu, dok za ostale stranice može samo pregledavati detalje. Unutar grupa korisnik može pregledavati članove grupa te njihove sadržaje i stranice, dok kod stranica može vidjeti samo one gdje je sudionik ili gdje mu je prema pravilima privatnosti omogućeno da može vidjeti odabranu stranicu. Na slici 2.2.. je prikazan dijagram obrazaca uporabe za registriranog korisnika za sve opisane ili spomenute obrasce uporabe.



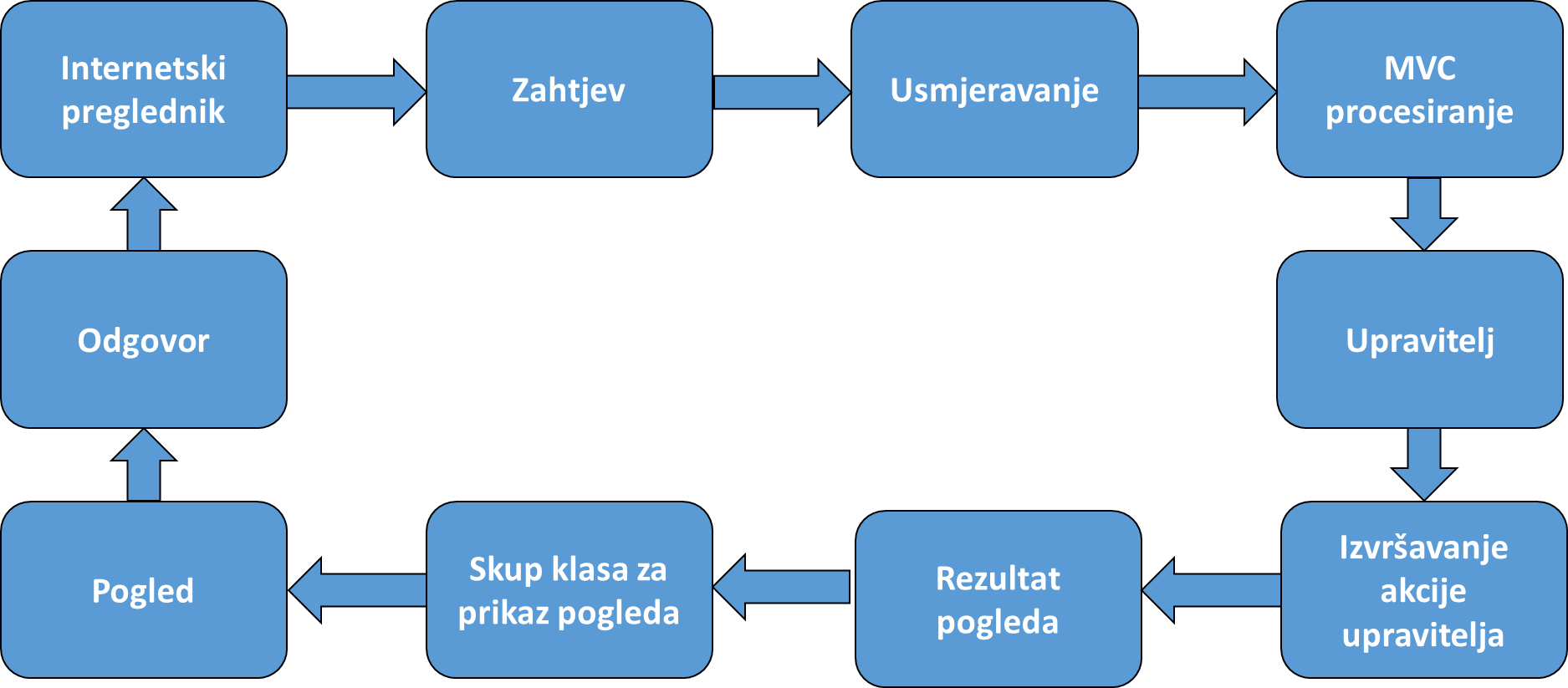
# Arhitektura i dizajn sustava

## Web-aplikacija

Pisanje web-aplikacija je pojednostavljeno zahvaljujući aplikacijama otvorenog koda koji su također i radni okviri te su neki od najpopularnijih Django, Drupal, Ruby on Rails i Symfony. Radni okviri potiču brzinski razvoj aplikacije tako da omoguće programskom timu da se više posvete važnijim i jedinstvenim dijelovima aplikacije bez potrebe za mučenjem s osnovnim dijelovima aplikacije poput uređivanja korisnika. Iako je većina radnih okvira otvorenog koda, uvjet za izradu web-aplikacije nije radni okvir otvorenog koda.

U ovoj se aplikaciji koristi ASP.NET MVC (*Model-View-Controller*) te je za pokretanje web-aplikacije dovoljno računalo s internetskom vezom i instaliranim internetskim preglednikom. Životni ciklus zahtjeva pri otvaranju odabrane web-stranice se sastoji od sedam glavnih koraka, a to su:

1. Usmjeravanje koje služi za spajanje poveznice zahtjeva s poveznicom registriranom u tablici usmjeravanja. Ako je odgovarajuća poveznica pronađena, algoritam za usmjeravanje prosljeđuje zahtjev na odgovarajući *IRouteHandler* koji će procesirati poslani zahtjev. Ako ne postoji poveznica u tablici usmjeravanja, vratit će *404 HTTP Status*[[1]](#footnote-1)grešku.
2. *MvcHandler* je zaslužen za inicijaliziranje pravog procesiranja unutar ASP.NET MVC-a. MVC procesiranje inicijalizira *IHttpHandler* sučelje i dalje procesira zahtjev koristeći *ProcessRequest* metodu.
3. *MvcHandler* koristi *IControllerFactory* instancu i pokušava dobiti *IController* instancu. Ako uspije u tome, zove se *Execute* metoda koja izvršava zahtjev putem upravitelja. *IControllerFactory* može biti osnovna tvornički upravitelj ili osnovna tvornica inicijalizirana putem *Application\_Start* događaja.
4. Kad je upravitelj inicijaliziran, njegov *ActionInvoker* određuje koju akciju treba upotrijebiti iz upravitelja. Akcija koja se treba izvesti je određena na temelju atributa *ActionNameSelectorAttribute* (od osnovne metode koja može imati isto ime kad je akcija izabrana) i *ActionMethodSelectorAttribute* (ako je više metoda pronađeno, bira se točna metoda pomoću ovog atributa).
5. Akcija dobiva ulaz, priprema odgovarajući odgovor te se onda izvršava nakon čega vraća rezultat putem *return* kao tip rezultata. Tip rezultata može biti *ViewResult*, *RedirectToRouteResult*, *RedirectResult*, *ContentResult*, *JsonResult*, *FileResult*, i *EmptyResult*.
6. Prvi korak u izvršavanju rezultata pogleda je izbor prikladnog skupa klasa za prikaz pogleda koji će oblikovati rezultat pogleda. To se određuje putem *IViewEngine* sučelja, a ASP.NET MVC najviše koristi *WebForm* i *Razor* načine prikaza.
7. Pogled vraća HTML stranicu pomoću izabranog skupa klasa za prikaz pogleda i rezultata akcije.

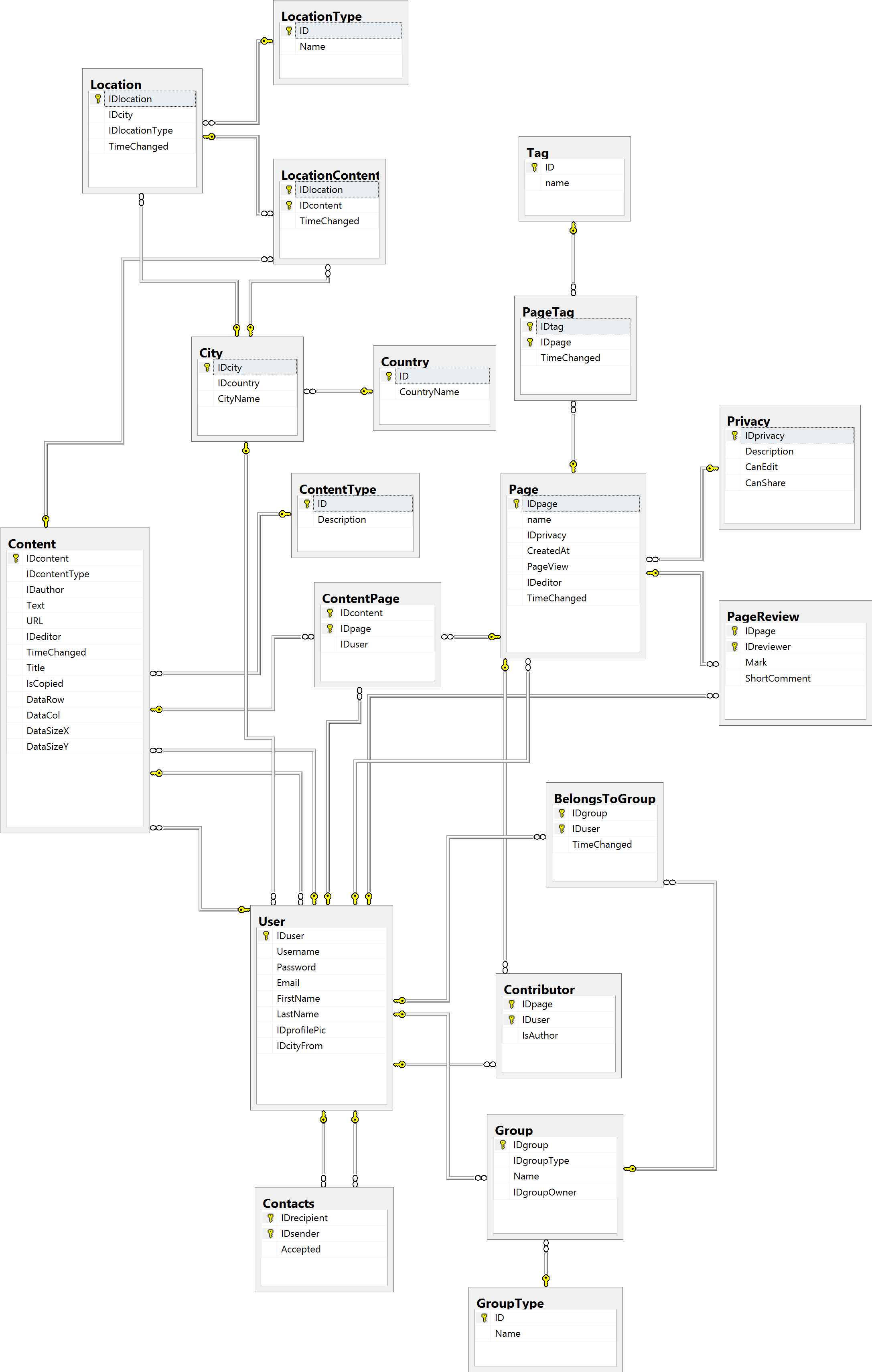


Slika 3.1.1. – ASP.NET MVC životni ciklus zahtjeva

## Baza podataka

U bazu podataka se pohranjuju podatci koje je unio korisnik i svi podatci generirani na temelju odabranih postavki. Baza podataka je ostvarena koristeći Microsoft SQL Server 2012 i njegov primarni jezik za upite T-SQL[[2]](#footnote-2). Zbog teme i potrebe za spremanjem različitih podataka, radi se o velikoj bazi podataka s (UNESI KONAČAN BROJ) entiteta.

Većina entiteta ima numerički primarni ključ oblika IDimeEntiteta ili ID koji je postavljen tako da se njegova vrijednost automatski povećava kako dodajemo redove podataka u odabrani entitet. Taj numerički primarni ključ se neće navoditi kod entiteta gdje se koristi, već će se podrazumijevati da postoji, dok će numerički primarni ključevi koji su drugačije postavljeni unutar odabranog entiteta biti navedeni i definirani.



**User:**

Glavni entitet, bez kojeg ne mogu postojati ostali entiteti jer ovise o njemu, je korisnički entitet *User*. On se sastoji od jedinstvenog korisničkog imena *Username*, odabrane lozinke za prijavu na web-aplikaciju *Password* te pripadajuće korisničke e-mail adrese *Email*. Osim osnovnih podataka za prijavu, entitet *User* se također sastoji i od dodatnih korisničkih informacija poput imena *FirstName* i prezimena korisnika *LastName* te mjesta podrijetla *IDcityFrom* koji služe za bolju identifikaciju korisnika na stranici i prepoznavanje korisnika od strane drugih korisnika.

**Content:**

Jedan od najvažnijih entiteta baze podataka je i entitet *Content* koji sadrži sve objavljene sadržaje koji se mogu pridružiti odabranim stranicama. On se sastoji od pridružene vrste sadržaja *IDcontentType*, autora sadržaja *IDauthor*, teksta sadržaja *Text* i naslova sadržaja *Title*. Sadržaj se može uređivati te se u tom slučaju spremaju korisnik *IDeditor* koji je odabrani sadržaj uredio i vrijeme uređivanja sadržaja *TimeChanged*. Postoji i zastavica *IsCopied* koja služi za prikaz samo sadržaja koji nisu kopirani kad su bili dodani na neku stranicu. Prisutni su i dodatni numerički atributi koja predstavljaju poziciju sadržaja unutar stupca *DataRow*, poziciju sadržaja unutar retka *DataCol*, širinu sadržaja *DataSizeX* i dužinu sadržaja *DataSizeY*. Ova četiri atributa služe za namještavanje rasporeda sadržaja koji se prikazuje unutar odabrane stranice.

**ContentType:**

Entitet koji definira vrstu sadržaja i koji služi kao šifrarnik je entitet *ContentType* te se on sastoji od samo opisa odabranog tipa sadržaja *Description*.

**Page:**

Još jedan od važnih entiteta je entitet za stranice *Page*. Ovaj se entitet sastoji od imena stranice *name*, pridružene postavke privatnosti za stranicu *IDprivacy*, vremena stvaranja stranice *CreatedAt* i od broja pregleda odabrane stranice *PageView*. Ako je autor ili jedan od sudionika stranice uređivao nešto na stranici, ime sudionika stranice koji je mijenjao nešto se sprema u *IDeditor*, dok se zadnje vrijeme promjene sprema u *TimeChanged*.

**Contributor:**

Među najvažnijim entitetima bez kojega nema pridruženih sudionika odabranoj stranici je entitet *Contributor*. On se sastoji od primarnih ključeva stranice *IDpage* i sudionika stranice *IDuser* te se još dodatno sastoji od zastavice *IsAuthor* pomoću koje znamo je li sudionik stranice koji je pridružen odabranoj stranici autor te stranice.

**ContentPage:**

Jedan od važnijih entiteta koji služi za pridruživanje sadržaja odabranoj stranici je *ContentPage*. On se sastoji od primarnih ključeva sadržaja *IDcontent* i stranice *IDpage* te se još dodatno sastoji od pridruženog imena korisnika *IDuser* koji je dodao sadržaj na odabranu stranicu.

**Privacy:**

Entitet *Privacy* je također jedan od važnijih entiteta jer definira postavke privatnosti za odabranu grupu. Sastoji se od opisa odabrane privatnosti *Description*, zastavice *CanEdit* koja definira je li odabrana grupa može uređivati odabranu stranicu i zastavice *CanShare* koja definira je li odabrana grupa može vidjeti odabranu stranicu.

**PageReview:**

Ovaj entitet služi za ocjenjivanje stranice. On se sastoji od primarnih ključeva stranice *IDpage* koja se ocjenjuje i korisnika ocjenjivača *IDreviewer* te od brojčanog atributa *Mark* u koji se sprema ocjena stranice.

**Location:**

Jedan od entiteta koji je veoma bitan za sortiranje sadržaja po lokaciji je entitet *Location*. On se sastoji od pridruženog imena lokacije *IDcity*, pridruženog tipa lokacije *IDlocationType* i eventualnog vremena promjene nekog od zapisa *TimeChanged*.

**LocationType:**

Entitet i šifrarnik *LocationType* definira vrste lokacija te se sastoji od samo imena vrste lokacije *Name*.

**City:**

Entitet koji se veže na *Location* je *City*. On se sastoji od pridružene države *IDcountry* gdje se nalazi odabrana lokacija i imena lokacije *CityName*.

**Country:**

Ovaj entitet ima ulogu šifrarnika te služi za spremanje imena država pa se zbog toga sastoji od samo imena države *CountryName*.

**LocationContent:**

Entitet koji pridružuje sadržaj odabranoj lokaciji je *LocationContent* te se on sastoji od primarnih ključeva lokacije *IDlocation* i sadržaja kojem se pridružuje lokacija *IDcontent* te od dodatnog atributa *TimeChanged* koji služi za eventualne promjene jednog od atributa.

**Group:**

Ovaj entitet služi za definiranje korisničkih grupa te se sastoji od pridruženog tipa grupe *IDgroupType*, izabranog imena grupa *Name* i pridruženog vlasnika grupe *IDgroupOwner*.

**GroupType:**

Entitet *GroupType* je šifrarnik koji definira sve vrste grupa te se sastoji od samo imena vrste grupe *Name*.

**BelongsToGroup:**

Među entitetima koji označuju pripadnost se nalazi i entitet *BelongsToGroup* koji označava pripadnost odabranog korisnika jednoj od grupa. On se sastoji od primarnih ključeva grupe *IDgroup* i korisnika kojeg pridružujemo nekoj grupi *IDuser* te od dodatnog atributa *TimeChanged* koji služi za eventualne promjene jednog od atributa.

**Tag:**

Ovaj entitet ima ulogu definiranja ključnih riječi koje služe za bliže opisivanje odabrane stranice te se entitet *Tag* sastoji od samo pridružene ključne riječi *Description*.

**PageTag:**

Entitet *PageTag* služi za pridruživanje odabrane ključne riječi stranici te se on sastoji od primarnih ključeva ključne riječi *IDtag* i stranice *IDpage* kojoj se treba pridružiti te od dodatnog atributa *TimeChanged* koji služi za eventualne promjene jednog od atributa.

# Pregled korištenih tehnologija u izradi aplikacije

## .NET Framework (radni okvir)

.NET radni okvir je Microsoftov softverski radni okvir koji se uglavnom izvodi na Microsoft Windows platformama. Sastoji se od velike biblioteke klasa *Framework Class Library (FCL)* koja se koristi unutar nekoliko programskih jezika te pruža interoperabilnost jezika zahvaljujući čemu svaki programski jezik može koristiti kod pisan u drugom jeziku. FCL pruža korisničko sučelje, pristup podatcima, mogućnost povezivanja s bazama podataka, kriptografiju, razvoj web-aplikacija, numeričke komunikaciju i mrežnu komunikaciju. Programeri razvijaju aplikaciju kombiniranjem vlastitog koda s .NET radnim okvirom i drugim bibliotekama. Najčešće se koristi *Visual Studio*, Microsoftova integrirana razvojna okolina za .NET aplikacije.



Slika 4.1. – Model .NET Frameworka

CLR (*Common Language Runtime*) je komponenta virtualne mašine u .NET razvojnom okviru koja upravlja kodom i izvršavanjem .NET aplikacija. Kod svih programa u .NET-u je tzv. upravljani kod te se sad kod u jednom trenu prevodi u stogovni međujezik CIL (*Common Intermediate Language*). CLR provodi JIT (*just-in-time*) prevođenje gdje se program prvo prevodi u stogovni međujezik te se prilikom prvog izvršavanja programa CIL kod pretvori u strojni kod platforme što se naziva JIT prevođenje. Prilikom svakog sljedećeg izvršavanje aplikacije se uzima strojni kod iz pričuvne (*cache*) memorije čime se postiže ubrzanje rada programa. Od mogućnosti CLR pruža i oslobađanje resursa putem korištenja sakupljača smeća koji briše objekte koji se više ne koriste, višejezično okruženje koje koristi CTS (*Common Type System*) gdje se nalaze tipovi podataka koji su zajednički svim jezicima u .NET-u, strukturirano hvatanje iznimki i pakiranje CIL koda u asemblije, najmanje cjelina izvršnog koda u .NET-u koje su zapravo komponente koje opisuju same sebe.

BCL (*Base Class Library*) je podskup FCL-a i odnosi se na cijelu klasu razreda koji dolaze uz .NET razvojni okvir. Neki od najpoznatijih biblioteka iz BCL-a su: Windows forme, ADO.NET, ASP.NET, *Language Integrated Query* (LINQ), *Windows Presentation Foundation* (WPF), *Windows Communication Foundation* (WCF) i *Workflow Fountation* (WF). Razredi su grupirani u prostore imena (*namespace*) poput *System*, *System.Collections*, *System.Data*, *System.IO*, *System.Text*, *System.Threading* i mnogih drugih.

## ASP.NET radni okvir

ASP.NET radni okvir je aplikacijski radni okvir otvorenog koda koji radi na serverskoj strani te je izrađen za web-razvoj radi izrade dinamičkih web-stranica. Izrađen je od strane Microsofta s ciljem izrade dinamičkih web-stranica, web-aplikacija i web-servisa. On je danas u procesu reimplementacije u moderni i modularni internetski radni okvir MVC 6 koji bi se sastojao od radnih okvira ASP.NET MVC, ASP.NET Web API i ASP.NET Web Pages (Razor).

ASP.NET web-stranice, koje se službeno zovu web-forme (*Web Forms*), glavni su dio razvoja aplikacije u ASP.NET-u.

### ASP.NET MVC (Model-View-Controller)

Slika 4.2. – Model-View-Controller koncept. Puna crta prikazuje direktnu povezanost, dok isprekidana prikazuje indirektnu povezanost kod promatrača.



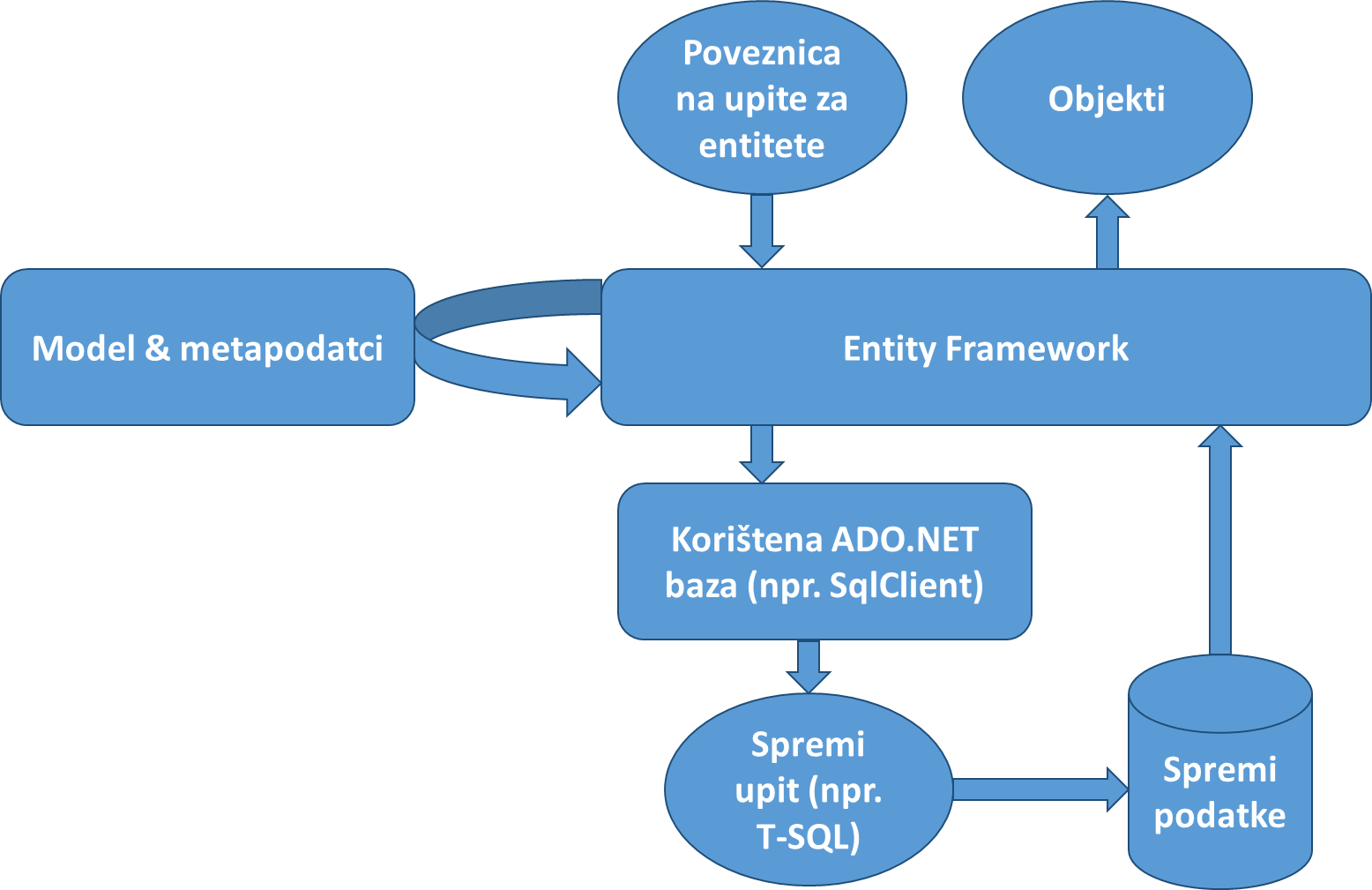
ASP.NET MVC je aplikacijski radni okvir otvorenog koda koji je temeljen na *Model-View-Controller* softverskoj arhitekturi. MVC je uzorak za razvoj aplikacija koje su dobro dizajnirane, koje se mogu testirati i koje se lako održavaju, a sastoji se od tri cjeline: modela (*Model*) koji sadrži klase kojima se modeliraju podatci i ostvaruje validacija, pogleda (*View*) koji prikazuje prethodno modelirane podatke i upravitelja (*Controller*) koji upravlja korisničkim zahtjevima tako da hvata zahtjeve preglednika, dohvaća podatke iz modela i prosljeđuje ih za prikaz u pogled. MVC uzorak nam pomaže u stvaranju aplikacija koje odvajaju različite dijelove aplikacije uz pružanje barem nekakve veze između stvorenih elemenata.

Neke od glavnih značajki ASP.NET MVC-a su: razdvajanje aplikacijskih zadataka (slojeva), mogućnost ispitivanja, ispitivanje čitavog sustava (funkcijsko ispitivanje i ispitivanje performansi), proširivost, podrška za ASP.NET usmjeravanje, podrška za korištenje označavanja u postojećim ASP.NET stranicama, korisničkim kontrolama i stranicama zaglavlja (*master pages*) te podrška za dostupne ASP.NET značajke.

## Entity Framework

Entity Framework (u daljnjem tekstu: EF) je ADO.NET radni okvir za objektno-relacijsko preslikavanje koji je izvorno bio dio .NET radnog okvira, dok je trenutno paket otvorenog koda (*Open Source*) čiji je izvorni kod dostupan javnosti na uvid, korištenje, izmjene i daljnji razvoj ili izmjenu funkcionalnosti odabrane aplikacije. Arhitekti i programeri podatkovno-orijentiranih aplikacijama su se mučili s ostvarivanjem dva različita cilja jer moraju modelirati entitete, veze i logiku poslovnih problema koje rješavaju te također moraju raditi s bazama podataka koje koriste za spremanje i dohvaćanje podataka. Podatci mogu biti spremljeni na više mjesta ili baza podataka od kojih svaka može imati svoj vlastiti protokol.

EF omogućuje programerima da rade s podatcima u obliku objekata i svojstava karakterističnih samo za određenu domenu bez brige o osnovnim entitetima i atributima gdje su dohvaćeni podatci spremljeni. Mogu raditi na većoj razini apstrakcije gdje mogu razviti aplikacije s manje koda temeljene na podatcima dohvaćenim pomoću EF-a.



Slika 4.3. – Entity Framework i procesiranje rezultata

## JavaScript i jQuery

### JavaScript

JavaScript je dinamički programski jezik uglavnom korišten kao dio internetskih preglednika čija implementacija omogućava klijentskim skriptama da komuniciraju s korisnikom, kontroliraju preglednik, komuniciraju asinkrono i mijenjaju prikazani sadržaj dokumenta. Također je korišten u okolinama poput Node.js-a, okolinama za razvoj igara, stolnih i mobilnih aplikacija.

Glavni cilj JavaScripta je omogućiti dinamičku interaktivnost bilo koje web-stranice čime se postiže mogućnost dodavanja bilo kakvih dodatnih mogućnosti. Jezik je dosta kompaktan, ali vrlo fleksibilan te su programeri zahvaljujući dome napisali mnoge alate koji omogućuju korištenje JavaScripta i otključavanje velikog broja dodatnih funkcionalnosti uz malo truda. Neki od njih su: osnovni API-ji (Application Programming Interfaces ili sučelje za programiranje aplikacija) koji se ugrađuju u internetske preglednike te pružaju mnoge dinamičke funkcionalnosti, specifični API-ji s programskim paketima posebne namjene koji omogućuju programerima da ugrade funkcionalnosti s drugih web-stranica i drugi radni okviri i biblioteke.

Danas za JavaScript postoje mnoge biblioteke, a jedna od najkorištenijih i najpopularnijih je jQuery.

### jQuery

jQuery je višeplatformna JavaScript biblioteka stvorena da olakša skriptiranje HTML-a s klijentske strane. On je besplatna aplikacija otvorenog koda licencirana putem MIT licence.

Sintaksa jQueryja je napravljena tako da olakša navigaciju kroz dokument, izbor DOM[[3]](#footnote-3) elemenata, izradu animacija, izvršavanje akcija prilikom obrade izabranog događaja i razvoj AJAX aplikacija.

Neke od prednosti jQueryja su: veće razdvajanje JavaScripta i HTML-a zahvaljujući događajima, kratkoća i jasnost, uklanjanje nekompatibilnosti na više internetskih preglednika i proširivost.

jQuery biblioteka se sastoji od JavaScript datoteke sa svim potrebnim funkcijama te se inicijalizira tako da se u web-stranicu doda poveznica na lokaciju biblioteke.

# Izrada aplikacije i opis rješenja

# Zaključak

# Literatura i reference

SREDITI KASNIJE

http://en.wikipedia.org/wiki/Web\_application

K. Harding, Analyzing requirements and defining Microsoft .net solution architectures. Microsoft Press, 2003.

http://en.wikipedia.org/wiki/.NET\_Framework

http://en.wikipedia.org/wiki/ASP.NET

https://msdn.microsoft.com/en-us/library/dd381412%28v=vs.108%29.aspx

<https://msdn.microsoft.com/en-us/library/vstudio/bb399567%28v=vs.100%29.aspx>

<https://developer.mozilla.org/en-US/Learn/Getting_started_with_the_web/JavaScript_basics>

<http://en.wikipedia.org/wiki/JQuery>

http://www.dotnet-tricks.com/Tutorial/mvc/TbR0041112-Asp.net-MVC-Request-Life-Cycle.html

# Sažetak

# Summary

# Ključne riječi

1. 404 ili Not Found pogreške je HTTP-ov standardni odgovor koji znači da je klijent uspio ostvariti komunikaciju sa željenim serverom, ali da server nije uspio pronaći ono što je bilo zahtijevano. [↑](#footnote-ref-1)
2. T-SQL (Transact-SQL) je Microsoftov i Skybaseov vlasnički dodatak SQL-u koji se najviše koristi u Microsoft SQL Serveru. On proširuje SQL standard tako da koristi proceduralno programiranje, procesiranje podataka, matematičke izraze, promjene u DELETE i UPDATE izrazima i mnoge druge promjene. [↑](#footnote-ref-2)
3. DOM (*Document Object Model*) je stablasta reprezentacija svih elemenata web-stranice koja omogućuje jQueryju jednostavno traženje, biranje i manipuliranje DOM elementima. [↑](#footnote-ref-3)