Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004:007
DOI: https://doi.org/10.24867/02OO01Krnic-Otic

REINŽENJERING INFORMACIONOG SISTEMA ZA PODRŠKU PROJEKTNOM BIROU

REENGINEERING OF INFORMATION SYSTEM FOR DESIGN OFFICE SUPPORT

Isidora Krnić-Otić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – INFORMACIONE TEHNOLOGIJE

Kratak sadržaj – *Predmet ovog rada je poređenje i analiza dva arhitekturalna obrasca Web forms i MVC, kao polazna osnova za odluku o izboru jedne od njih za razvoj aplikacije. U okviru rada biće opisan sistem za podršku projektnom birou, prvobitni razvijen Web forms tehnologijom i prošireni sistem razvijen pomoću MVC tehnologije. Cilj rada je da se donese zaključak koji obrazac je povoljniji za konkretno naveden sistem.*

Ključne reči: *ASP.Net Web forms, ASP.Net MVC, Microsoft .Net, MS SQL Server, ADO.Net entity framework, ORM*

Abstract – *Subject of this work is comparison and analysis technologies: Web forms and MVC as beginning point of choosing one of them for developing application. This work will describe system for design office, first developed in Web forms technology and extended, developed with MVC architectural pattern. Purpose of this work is to come to a conclusion witch is more favorable for developing concrete application.*

Key words: *ASP.Net Web forms, ASP.Net MVC, Microsoft .Net, MS SQL Server, ADO.Net entity framework, ORM*

1. UVOD I OSNOVNI POJMOVI

Izrada, vođenje i arhiviranje projektne dokumentacije su ključne aktivnosti projektnog biroa. Upravljanje poslovima jedne ovakve firme bi bilo vođenje evidencije o različitim elementima kao što su podaci o predmetu rada, delovima projekata u okviru tog predmeta rada, evidentiranje podataka o investitorima, inženjerima i tehničkim saradnicima, čuvanje dokumentacije vezane za određeni predmet rada. Pod predmetom rada podrazumeva se objekat za koji se izrađuje projektna dokumentacija ili vrši neka druga delatnost poput nadzora izvođenja radova.

Vođenje evidencije navedenih elemenata može se realizovati: *ručno* (neautomatizovano) - evidentiranje se vodi putem mehanografskih obrazaca, štampanih formi ili putem elektronskih formi sačinjenih pomoću dostupnih programa (MS Word, MS Excell, Adobe Acrobat i sl.), ili *automatiozovano* – koristeći aplikaciju povezanu sa bazom podataka putem koje se čuvaju, čitaju i ažuriraju podaci. Automatizovanim sistemom dobija se mogućnost analize poslova, prispeća rokova, zauzeća radnika i dr.
Projektni biro pored arhiviranja projektne dokumentacije ima potrebu da čuva i sve dokumente koji su prethodili izradi određenog projekta.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Srđan Sladojević, docent.

Tu spadaju dokumenti dobijeni od investitora, ali i skice i crteži, nastali na merenju sa lokacije za koju se izrađuje projekat. Sva navedena dokumentacija se čuva za potencijalne potrebe u budućnosti (adaptacije, rekonstrukcije i drugo). Važan faktor u pronalaženju arhivirane dokumentacije nosi upravo adekvatno upravljanje. Takođe, sa rastom staža firme, raste i potreba za većim fizičkim prostorom za čuvanje navedene dokumentacije.
Posmatrajući samo ovu ulogu, projektni biroi i sa većim i sa manjim obimom posla imaju realnu potrebu za automatizovanim upravljanjem poslova.

2. RAZVOJNI ALATI

2.1. Microsoft SQL Server

Microsoft SQL server sistem za upravljanje relacionim bazama podataka kompatibilan .Net aplikacijama. SQL server omogućava skladištenje i manipulisanje nad podacima sadržanim u bazama. Za razvoj ovog informacionog sistema korišćen je Microstoft SQL server 2005 Express Edition. Za dobavljanje i manipulaciju podatacima na aplikativnom nivou uveden je sloj ADO.Net Entity model.

2.2. ORM i ADO.Net Entity framework

ORM (object relational mapping) je tehnika konverzije podataka između objektnog modela i relacione baze u oba smera. Microsoft .Net Entity framework (EF) je ORM struktura prilagođena .Net aplikacijama.

Slika 1. ORM šema

Prednosti u korišćenju ORM-a su programiranje rasterećeno vođenja računa o slojevima koji se tiču baze podataka kao što su konekcija i upiti ka bazi. Programiranje se podiže na apstraktniji nivo i time smanjuje količina pisanog koda. EF kreira objektne modele koje koristi pri upitima i unosima u bazu podataka. Upiti ka bazi su mogući pomoću LINQ upita. EF izvodi automatski transakciju prilikom dobavljanja ili čuvanja podataka. Takođe koristi keširanje učestalih upita smanjujući time dobavljanje podataka direktno iz baze.[1]

2.3. Web forms tehnologija

ASP.NET Web forms je jedan od četiri programska modela za kreiranje ASP.NET web aplikacija. Ostale tri

su ASP.NET MVC, ASP.NET Web Pages, and ASP.NET Single Page Applications.

Osnovni pojmovi Web forms tehnologije su:

Postback akcija – inicira se sa klijenta (preglednik), najčešće događajem neke kontrole. Stanje te kontrole zajedno sa svim ostalim kontrolama na stranici se šalje kroz postback komponentu serveru.

View state – čuva poslednje stanje na serveru svih kontrola.

Životni vek stranice – započinje zahtevom sa klijenta i završava vraćenim odgovorom sa servera. Tu je uključen niz koraka obrade: inicijalizacija, instanciranje kontrola, vraćanje ili održavanje stanja, izvršavanje događaja i konačno iscrtavanje stranice. Faze životnog veka stranice su redom:

Zahtev – započinje pre početka životnog veka stranice. Nakon što je klijent poslao zahtev, ASP.NET određuje da li je potrebno parsirati i kompajlirati stranicu, znači započeti životni vek ili je moguće vratiti keširanu stranicu kao odgovor.

Start – setuje se IsPostBack property.

Initialization – kontrole su dostupne.

Load – ukoliko je zahtev postback , kontrole se učitavaju sa podacima koje čuva view state i control state.

Postback događaji - ako je zahtev postback, pozivaju se metode za obradu događaja.

Rendering – pre samog iscrtavanja, čuva se view state svih kontrola i same stranice. Stranica poziva metodu Render za svaku kontrolu putem OutputStream objekta.

Unload – pokreće se nakon što je stranica u potpunosti učitana.

Karakteristično za Web forms model je što je razvoj sličan razvoju desktop aplikacije. Web forms koristi Page Controller obrazac, dostupne su serverske kontrole koje prevlačenjem generišu kod za izgled određene stranice. Jedna stranica, odnosno web forma je jedna jedinica sačinjena od prezentacione klase (.aspx) i serverske klase (.aspx.cs). Drugim rečima, svaka prezentaciona klasa je vezana za serversku klasu pozadinskog koda gde se definiše ponašanje i/ili izgled kontrola. Ovako koncipirana platforma oslobađa programera za upoznavanjem html i javascript jezika i ubrzava razvoj, a poznat je kao Rapid Application Development (RAD).

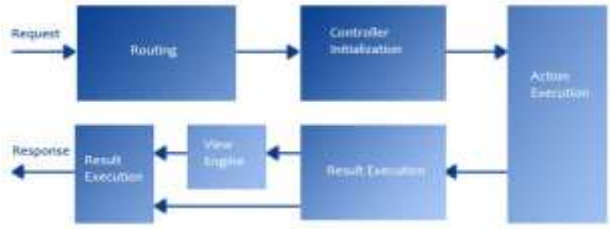
Kada preglednik zahteva stranicu, na serveru se kod prezentacione klase dinamički interpretira u html i zatim šalje klijentu. Važno je istaći, pri ovom procesu, nakon što su podaci za prikaz stranice poslati pregledniku, server ih briše iz memorije (eng. stateless). To znači da pri svakom zahtevu za stranicu (može biti i ista), server ponavlja prethodne instrukcije [2].

Slika 2. ASP.NET interpreter

2.4. MVC arhitekturalni obrazac

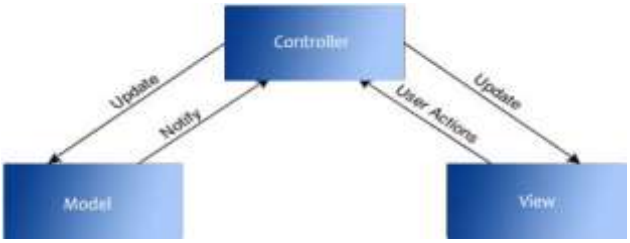
Za razliku od Web forms, .NET MVC isključuje pojam životnog veka stranice.

U ovom obrascu postoji životni vek koji se odnosi na kompletnu aplikaciju i životni vek zahteva. Životni vek aplikacije počinje startovanjem web servisa i završava se njegovim zaustavljanjem. Životni vek jednog zahteva čini niz događaja koji se izvršavaju pri svakom zahtevu. Polazna tačka svakog zahteva je rutiranje. Modul za rutiranje je odgovoran za pronalaženje adekvatne URL adrese. Podaci iz ovog modula se šalju prethodno kreiranom adekvatnom kontroleru putem Action invoker komponente. Ova komponenta poziva action metodu u samom kontroleru. Pošto je pripremljen Action result, sledi faza Result Execution. Ova komponenta je konkretni odgovor http zahtevu.



Slika 3. Životni vek zahteva

Važno je napomenuti da .Net MVC platforma i MVC arhitekturalni obrazac nisu isto. .Net MVC platforma je implementirala MVC obrazac prilagođen tako da odgovara web aplikacijama. MVC (akronim od Model-View-Controller) je arhitekturalni obrazac koji je u širokoj primeni u različitim tehnologijama. Ovaj obrazac podrazumeva grupisanje koda u tri nezavisna dela (slika 4.)



Slika 4. MVC arhitektura

Model sadrži klase čije instance čuvaju podatke sa kojima aplikacija manipuliše.

View definiše izgled korisničkog interfejsa. Pored osnovnog html-a, brine se o događajima, osvežavanjima i drugim relevantnim pojmovima u vezi sa izgledom stranice.

Controller čine klase koje sadrže logiku aplikacije i preko kojih se vrši komunikacija između modela i vew-a. Na ovaj način se čuva konzistentnost podataka. Nezavisan view se ne oslanja na specifičnu implementaciju modela i kao takav može da se koristi za prezentovanje podataka nekog drugog modela.

Ideja ovog obrasca je da odvoji manipulaciju podacima od korisničkog interfejsa [3].

3. KOMPARATIVNA ANALIZA

3.1. Page/Controller i Front Controller

- *Web forms* koristi *Page/Controller obrazac* za generisanje stranice. U ovom pristupu svaka stranica ima svoj kontroler koja procesuira zahtev. [4][6]

- *.Net MVC* koristi *Front Controller* obrazac. Praktično to je najčešće jedan kontroler za sve stranice koje se tiču istog modela (tabele u bazi). [4] Moguće je uneti dodatnu apstrakciju kreiranjem baznog kontrolera kojeg će ostali naslediti a preko kojeg se unosi logika za sve stranice (npr. Autorizacija prema rolama i druge zaštite). Centralizacija kontrolera daje prednost MVC platformi.

3.2. Međusobna sprega

- *Web forms* karakteriše *jaka sprega* između kontrolera i prezentacionog dela, što je mana pri razvoju većih aplikacija. Kod većih aplikacija, gde automatsko testiranje igra važnu ulogu, zbog ove osobine je veoma otežano [4][6].
- *.Net MVC* odvaja odgovornosti komponenti, izgled i kontroler se jasno odvajaju. TDD (test driven development) je jednostavan sa ovom platformom [4][6].

3.3. View state

- Da bi se programerima dalo iskustvo slično razvoju desktop aplikacije, *Web forms* uvodi view state čime kontroliše konzistentnost podataka[3]. View state je skrivena kontrola na stranici i zbog toga problem nastaje kod izuzetno velikog view state-a jer utiče na veličinu stranice, a samim tim i na brzinu učitavanja.
- *.Net MVC* kao i druge web platforme ne uključuje koncept view state-a [4][6].

3.4. Životni vek stranice i životni vek zahteva

- *Web forms* model prati *životni vek stranice*.
- *.Net MVC* koristi jednostavan životni vek zahteva (request cycle), što čini intuitivniji pristup za programera [4][6].

3.5. RAD model i Scaffolded item

- *Web forms* je razvio *serverske kontrole* slično kao za desktop aplikacije sa namerom da minimalizuje upotrebu html-a, javascript-a i css-a, što u nekim slučajevima može biti nedostatak. Drag n' drop princip programiranja je pozitivna strana posmatrajući iz ugla brzine razvoja.
- *.Net MVC* kao odgovor brzini razvoja uvodi mogućnost kreiranja komponenti pomoću opcije *scaffolded item*. Na ovaj način povezujući se sa modelom iz baze veoma brzo generiše html kod za prikaz stranice. Pored toga uveo je tzv. Html helpere pomoću kojih se slično web forms serverskim kontrolama generiše html kod. Omogućena je kompletna kontrola nad html-om, javascript-om i css-om.

Uzimajući u obzir da Web forms i sa drag and drop pristupom zahteva izradu svake pojedinačne stranice, dok MVC platforma automatski generiše kod, svrstava u ovom pogledu MVC ispred Web forms.

4. PROŠIRENJE APLIKACIJE

Aplikacija za podršku projektnom birou, inicijalno je razvijena u ASP.Net Web forms tehnologiji, da bi se pri potrebi za njenim proširenjem razmatrala opravdanost njene migracije u ASP.Net MVC tehnologiju. Razlozi za proširenjem aplikacije (baze podataka) su redukovanje unosa redundantnih podataka i podizanje nivoa detaljnosti kako bi se omogućilo izveštavanje o trenutnom stanju.

4.1. Početno stanje

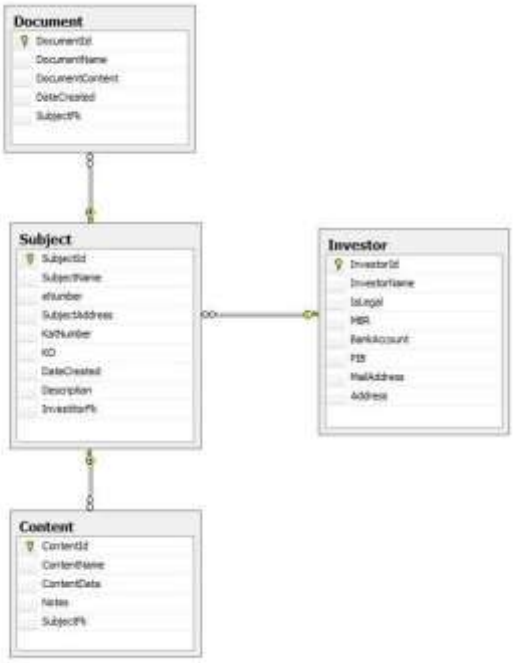
4.1.1. Opis baze

Baza podataka sastoji se iz četiri tabele. Tabela Subject je nosilac podataka o projektu za određeni objekat. Tabela Investor čuva podatke o investitoru. Investitor može naručiti više projekata, dok se jedan projekat izrađuje za jednog investitora. Svaki projekat vodi evidenciju o dokumentima putem tabele Document. Tabela Document čuva elektronski sadržaj svakog dokumenta. Odnosi se na skenirane dokumente dobijene od investitora ili ručne crteže sačinjene na predmetnoj lokaciji. Korišćen je ADO.NET Entity Data Model za povezivanje aplikacije sa bazom.

4.1.2. Opis aplikacije

Prvobitna manja aplikacija vodi evidenciju o projektima (Projects), njihovim investitorima (Investors), dokumentima (Documents) i evidentira sadržaj svakog projekta (Contents). Aplikacija je koncipirana da se na Home stranici izlistavaju svi projekti. Izmena i veze sa određenim projektom se vrši njegovom selekcijom i izborom dugmeta za manipulaciju nad projektom ili dugmeta za pregled drugih tabela u vezi sa njim.

Slika 5. Home stranica



Slika 6. Šema baze

Razvoj uz pomoć Web forms tehnologije zahteva izradu svake stranice pojedinačno, što podrazumeva izradu prezentacione klase za generisanje html-a i serverske klase za kontrolu nad elementima pripadajuće prezentacione klase. U ovom slučaju, pored generalnih listing stranica kojih ima koliko i tabela, potrebno je bilo implementirati za svaku od njih još po jednu stranicu za dodavanje ili ažuriranje i dodatno stranice koje prikazuju vezu između dve tabele.

4.2. Proširena aplikacija

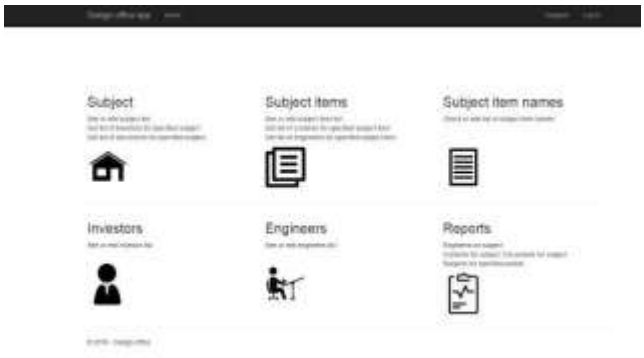
Proširenje aplikacije je proisteklo iz potrebe za detaljnijim vođenjem evidencije nad svim entitetima koji se pojavljuju u procesu vođenja projektnog biroa. Takođe u prvobitnoj aplikaciji za jedan objekat u tabeli Subject je bilo potrebno čuvati nove unose ukoliko je radjeno više od jednog projekta za određeni objekat. U tom smislu je rađeno na poboljšanju veza između tabela, kao i dodavanje novih.

4.2.1. Opis baze

Nova baza sadrži deset tabela. Tabela Subject je remodelovana da bi čuvala podatke o više projekata za jedan objekat. Ona evidentira podatke o samom objektu kao i u prethodnom slučaju, pri čemu je veza sa Content tabelom remodelovana. Budući da je moguće imati više investitora i ova veza Subject-Investor je izmenjena. Tabela Document je proširena kao bi se u njoj pored inicijalnih mogli evidentirati i administrativni dokumenti (ponude, ugovori, fakture). Baza je proširena sa tabelom koja evidentira inženjere i saradnike. Konačno preko šifrarnik tabele SubjectItemNames napravljena je veza sa Content tabelom kako bi se automatizovalo izlistavanje potrebne dokumentacije koju određeni projekat treba da sadrži.

4.2.2. Opis aplikacije

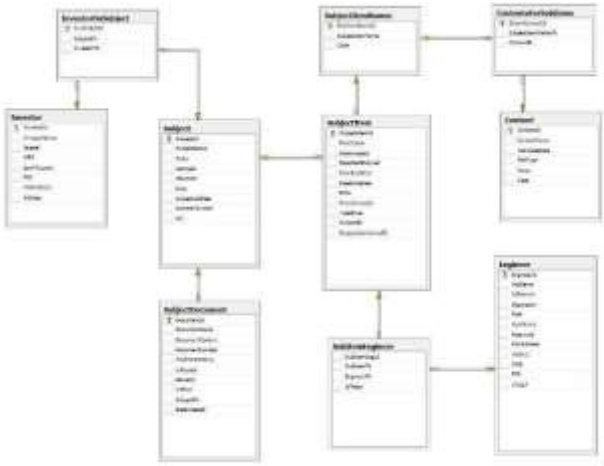
Nova aplikacija je razvijena pomoću .Net MVC platforme, uzevši u obzir njene, prethodno predstavljene, prednosti nad Web forms platformom.



Slika 7. Design office home stranica

Kao za prvobitnu aplikaciju i ovde je korišćen ADO.Net Entity Model za rad sa bazom. Automatsko generisanje koda znatno ubrzava razvoj i smanjuje frustraciju pri radu sa dugotrajnom izradom osnovnog izgleda aplikacije. U ovom slučaju aplikacija je izgenerisana uz pomoć nekoliko klikova. Uz pomoć skaffolded item opcije izgenerisane su za svaku tabelu mvc komponente uvezane sa Entity modelom. Potencijalno dodavanje novih pogleda ili tabela kao i njihovo proširivanje ne zahteva veliki trud da bi se aplikacija ažurirala.

Ono što preostaje da se razvija su specifičnosti same aplikacije, odnosno poslovna logika.



Slika 8. Design office šema baze

5. ZAKLJUČAK

Kako su obe posmatrane aplikacije nekompleksne, izabrani su faktori koji se mogu jednako posmatrati i u slučaju da jesu kompleksne. Akcenat je stavljen na faktor zadovoljstva programera pri razvoju kao i faktor brzine razvoja.

Automatsko generisanje koda koje utiče na poboljšanje zadovoljstva programera jer ne troši energiju na kucanje ponavljajućeg koda, svrstava MVC platformu ispred Web forms.

Treba još napomenuti da su Web forms i .Net MVC platforma nastale odvojeno i bez namere da se prva zameni drugom. Kod većih aplikacija prema ovoj analizi prednost bi se dala MVC platformi, dok kod manjih, presudnu ulogu za izbor jednog od dva bi mogli biti afiniteti programera.

6. LITERATURA

[1] http://www.agiledata.org/essays/mappingObjects.html (online), posl.pr. 20.08.18

[2] https://docs.microsoft.com/en-us/aspnet/web-forms/what-is-web-forms (online), posl.pr. 20.08.18

[3] https://www.tutorialspoint.com/asp.net_mvc/asp.net_mvc_pattern.htm (online). Posl.pr. 20.08.18

[4] https://www.codeproject.com/Articles/668182/Difference-betweeen-ASP-NET-WebForms-and-ASP-NET-M. (online), posl.pr. 20.08.18

[5] http://www.entityframeworktutorial.net/what-is-entityframework.aspx (online), posl.pr. 20.08.18

[6] Adam Freeman, "Pro ASP.NET MVC 5", 2013.

[7] Jon Galloway, Brad Wilson, K.Scott Allen, David Matson, Professional ASP.NET MVC 5, 2014.

Kratka biografija:

Isidora Krnić-Otić rođena je u Novom Sadu 1977. godine. Master rad na Fakultetu tehničkih nauka iz oblasti Informacione tehnologije brani u 2018. godini.