UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Pugelj Sebastjan

PRIMERJAVA TEHNOLOGIJ ASP IN ASP.NET PRI RAZVOJU SPLETNEGA SPREMLJANJA NOGOMETA

DIPLOMSKO DELO NA
VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU
PRVE STOPNJE

MENTOR: višji pred. dr. Igor Rožanc

Ljubljana, 2010

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00049/2010
Datum: 08.11.2010

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **SEBASTJAN PUGELJ**

Naslov: **PRIMERJAVA TEHNOLOGIJ ASP IN ASP.NET PRI RAZVOJU SISTEMA ZA SPLETNO SPREMLJANJE NOGOMETA**

THE COMPARISION OF ASP AND ASP.NET TECHNOLOGIES ON WEB FOOTBALL TRACKING SYSTEM DEVELOPMENT

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

V diplomski nalogi primerjajte tehnologiji ASP in ASP.NET pri izdelavi sistema za spremljanje nogometnih dogodkov preko spleta.

V ta namen najprej predstavite uporabljene tehnologije in orodja, čemur naj sledi sistematičen prikaz razvoja uporabniškega in uredniškega modula sistema na podlagi definiranih zahtev. Pri izvedbi izpostavite predvsem prednosti in slabosti obeh pristopov. Predstavite tudi izgled in uporabo sistema, nalogo pa zaključite s sistematično primerjavo obeh rešitev z vidika uporabljenih tehnologij.

Mentor:

viš. pred. dr. Igor Rožanc

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

IZJAVA O AVTORSTVU
diplomskega dela

Spodaj podpisani Pugelj Sebastjan, z vpisno številko 63020280, sem avtor diplomskega dela z naslovom:

PRIMERJAVA TEHNOLOGIJ ASP IN ASP.NET PRI RAZVOJU SPLETNEGA SPREMLJANJA NOGOMETA

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom dr. Igorja Rožanca,
- so elektronska oblika diplomskega dela, naslov (slov., ang.), povzetek (slov., ang.) ter ključne besede (slov., ang.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 29.10.2010 Podpis avtorja:

ZAHVALA

Zahvaljujem se mentorju diplomske naloge, višjemu predavatelju dr. Igorju Rožancu, za vodenje in prijazno pomoč pri nastanku diplomske naloge.

Posebno bi se zahvalil družini in mojemu dekletu, ki so me v času študija podpirali in vseskozi spodbujali.

KAZALO

KAZALO SLIK

KRATICE, OKRAJŠAVE, SIMBOLI

ASP	Active Server Pages – strežniški skriptni jezik.
ASP.NET	Spletno aplikacijsko ogrodje za izdelavo spletnih aplikacij in spletnih servisov.
BCL	Base Class Library – standardna knjižnica za vse programske jezike, ki uporabljajo .NET Framework.
CLR	Common Language Runtime – jedro v Microsoft .NET Framework ogrodju za izvajanje aplikacij.
CMS	Content Management System – sistem za upravljanje s spletnimi vsebinami (programske aplikacije, ki podpirajo generiranje in urejanje spletnih vsebin).
C#	Microsoftov programski jezik.
FLEX	brezplačno in odprtokodno ogrodje za ustvarjanje spletnih aplikacij.
HTML	HyperText Markup Language – standardni jezik za razvoj spletnih strani. Z jezikom HTML označujemo in določamo lastnosti besedila.
HTTP	HyperText Transfer Protocol – komunikacijski protokol med odjemalci in strežniki. Protokol je namenjen objavljanju in prejemanju informacij na spletu preko HTML strani.
JS	JavaScript – je skriptni jezik, ki omogoča izvajanje določenih operacij na uporabnikovi strani in tako omogoča hitrejše delovanje aplikacije.
OOP	Object oriented programming – objektno programiranje.

RIA	Rich Internet Applications – spletne aplikacije, ki posedujejo veliko lastnosti namiznih aplikacij.
SQL	Structured Query Language – strukturni poizvedovalni jezik.
T-SQL	Transact SQL – serija programskih razširitev za SQL, ki omogoča nadzor nad transakcijami ter rokovanje z napakami.
XML	Extensible Markup Language – programski jezik podoben HTML-ju, ki nam omogoča format za opisovanje strukturiranih podatkov ali arhitektura za prenos podatkov in njihovo izmenjavo med več omrežji.
MXML	Magic extensible Markup Language – označevalni jezik, ki temelji na XML-u ter se ga uporablja v Adobe Flex ogrodju.

POVZETEK

Cilj diplomske naloge je primerjava dveh različnih tehnologij za razvoj spletnih aplikacij. V našem primeru sta obe tehnologiji Microsoftov produkt: ASP in ASP.NET. Pri prvi tehnologiji uporabljamo kot glavni razvojni jezik Visual Basic, pri drugi tehnologiji pa C#. Primerjava je izvedena na primeru izdelave sistema za spremljanje nogometnih dogodkov.

Najboljši način, da naš sistem spozna in uporablja širši krog ljudi, je predstavitev preko spletne strani na svetovnem spletu. V diplomski nalogi smo podrobno predstavili razvoj in delovanje sistema za prikaz nogometnih dogodkov. Sistem je razdeljen na dva modula: predstavitveni modul ter uredniški modul. Primerjava tehnologij je izvedena na uredniškem modulu. Ločeno je predstavljen tudi razvoj predstavitvenega modula, ki je izdelan v tehnologiji FLEX.

ASP je prvo Microsoftovo strežniško-aplikacijsko ogrodje za izdelavo dinamičnih spletnih strani.

ASP.NET je naslednik ogrodja ASP. ASP.NET je tako kot ASP namenjen razvoju dinamičnih spletnih aplikacij in spletnih servisov. ASP.NET je zgrajen na jedru CLR, ki omogoča kodiranje v poljubnem jeziku, ki je podprt z ogrodjem .NET.

Adobe Flex je razvojno okolje, ki ga je razvil Adobe Systems. Namenjen je razvoju medplatformsko obogatenih spletnih aplikacij (ang. Rich Internet Applications), ki temeljijo na platformi Adobe Flash.

Sistem je izdelan po natančnih specifikacijah stranke, zato je bil razvoj usmerjen v strogo izpopolnjevanje zahtev le-teh. V uporabniškem modulu odjemalci pregledujejo vsebino nogometnih dogodkov preko interaktivnih kontrol, uredniški modul pa služi komentatorjem za vnašanje ter spreminjanje podatkov, kateri se prikazujejo odjemalcu.

Za izdelavo aplikacije z uporabljeno infrastrukturo se je za boljšo rešitev izkazala tehnologija ASP.NET. Kot glavno prednost naj navedemo tako lažje kodiranje kot tudi razhroščevanje.

Ključne besede:

ASP, ASP.NET, .NET Framework, CMS, Flex

ABSTRACT

The main aim of the diploma thesis is the comparison of two different technologies for web application development. In our case, both technologies are Microsoft products: ASP and ASP.NET. The first technology uses as main development language the Visual Basic, and the other technology C#. The comparison is done through a development of a system for monitoring the football events.

Web site publications are one of the most efficient ways to present our activity to the general public. The diploma work describes the development and performance system for football events monitoring. The system is divided into two modules: user module and administrator module. The comparison of the technologies is made on the administrator module. Also, presented separately is the user module developed in FLEX technology.

ASP is the first Microsoft server side script engine for dynamically generated web pages.

ASP.NET is the successor of the ASP framework. ASP.NET is intended like ASP for dynamical web application and web service development. ASP.NET is built on the Common Language Runtime (CLR) which allows programmers to write code using any supported .NET language.

Adobe Flex is a software development kit, released by Adobe Systems for the development and deployment of cross-platform rich Internet applications based on the Adobe Flash platform

The system is developed strictly to the customer's demands, pointing the development path in the direction of fulfilling the demands. User module's primary function is monitoring football events by internet users. The administrator module is intended for commentators to insert and edit data relevant to the football events appearing to the user.

The best technology for building applications with given infrastructures has proved to be ASP.NET. Its advantages are both, much easier coding and debugging.

Keywords:

ASP, ASP.NET, .NET Framework, CMS, Flex

1. UVOD

Mediji stremijo k čim večji gledanosti, poslušanosti, branosti, zato je jasno, da se poročanje o športnih dogodkih in vse, kar je povezano z njimi, hitro seli tudi na svetovni splet, ki je najpopularnejši sodobni medij.

Časi čakanja na časopis ali TV oddajo so minili. Ljudje hočejo dobiti informacije takoj. Informacije morajo biti vedno dostopne, kar jim omogoča uporaba spleta. Vendar se zaradi vedno večje medijske raznolikosti informacije širijo na vedno manj pregleden način. Spletni mediji so najbolj uspešni, ko imajo popoln pregled nad dogajanjem, v ključnih trenutkih razpolagajo s pravimi informacijami ter jih predstavijo na privlačen način.

Na podlagi teh smernic smo skupaj z naročnikom, ki upravlja športni (nogometni) medijski portal, razvili celovito rešitev za učinkovito in pregledno spremljanje nogometnih dogodkov ter vseh pripadajočih informacij v živo preko spleta.

Glavni problem pri vzpostavitvi sistema je nastopil pri izbiri ustreznega sistema CMS za upravljanje uredniškega modula. Na začetku smo morali izbrati med izdelavo lastnega CMS-ja ali že izdelanega CMS-ja, pri katerem se odločamo med odprtokodnimi ali komercialnimi rešitvami. Pri komercialnih rešitvah je velikokrat težava visoka cena ter nejasno specificirani stroški vzdrževanja. Pri odprtokodnih rešitvah je problem predvsem neprijaznost sistema, kar zahteva veliko vlaganje lastnega dela in vaje, nihče nam ne zagotavlja delovanje sistema, poleg tega pa razvoj lahko zaide v slepo ulico. Na tržišču je veliko različnih sistemov za upravljanje s spletnimi vsebinami, ki temeljijo na različnih programskih jezikih, uporabljajo različne spletne in aplikacijske strežnike ter različne podatkovne baze. Zaradi naštetih vzrokov smo se odločili izdelati lasten CMS.

Na naslednjem koraku se je pojavilo vprašanje o vrsti tehnologije, ki jo bomo uporabili. Na spletu je podano veliko mnenj na temo izbire tehnologije ASP oziroma ASP.NET. Privrženci prve ali druge podajajo svoja mnenja, pri čemer nobeno mnenje ne deluje dovolj objektivno. Tako smo se tega vprašanja lotili s preizkusom. Sklenili smo razviti sistem v obeh tehnologijah, tako ASP in ASP.NET, ki ju v zaključku primerjamo ter predstavimo njune prednosti in slabosti.

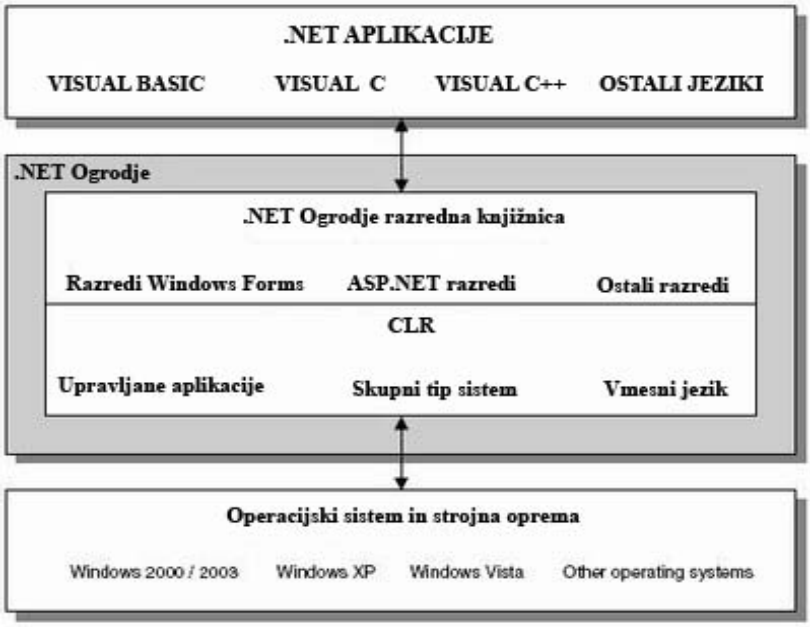
Namen in cilji diplomske naloge so: opisati in predstaviti uporabljeni tehnologiji in orodja, izdelati uporabno spletno aplikacijo za spremljanje nogometnih dogodkov ter primerjati obe tehnologiji pri izdelavi uredniškega modula.

2. UPORABLJENE TEHNOLOGIJE IN ORODJA

Poglavje je namenjeno predstavitvi tehnologij ter orodij, s pomočjo katerih je nastal celoten sistem: modul namenjen urednikom ter predstavitveni modul za odjemalce. Oba modula sta izdelana kot spletni aplikaciji.

2.1. Ogrodje .NET

Ogrodje .NET (ang. .NET Framework) je programska komponenta, ki jo naložimo na Microsoftove operacijske sisteme [1]. Ponuja obširen nabor vnaprej pripravljenih rešitev za različne programske potrebe. Ogrodje (slika 1) je s svojo enostavno uporabo postalo ključni element za razvoj programskih rešitev, ki delujejo na Microsoftovih platformah. Sestavljata ga dva ključna dela: knjižnice in CLR virtualni stroj. Virtualni stroj skrbi za izvajanje programske kode napisane v enem od jezikov, ki so podprti v ogrodju .NET. Ogrodje .NET predstavlja največji premik iz DOS okolja v Windows NT, saj pomeni odhod COM standarda in ga nadomešča izvajanje aplikacij znotraj skupnega izvajalnega okolja CLR.



Slika 1 : Ogrodje arhitekture .NET Framework

Osnovni razredi ogrodja .NET so razvrščeni po imenskih prostorih (ang. namespaces). Pomembni imenski prostori so:

- System – standardni razredi, ki jih uporabljajo skoraj vsi programi,
- System.Data – upravljanje s podatkovnimi bazami (ADO.NET),
- System.IO – ukazi za delo z datotečnim sistemom,
- System.Windows.Forms – razredi za delo z Windows uporabniškim vmesnikom,
- System.Net – mrežno programiranje,
- System.Web.UI – spletne strani ASP.NET,
- System.Collections – razredi za upravljanje s podatkovnimi zbirkami objektov,
- System.Security – razredi za kriptografijo,
- System.Threading – razredi za delo z nitmi.

Največja prednost okolja CLR je v zmožnostih uporabe komponent, razvitih v preostalih programskih jezikih.

Izvajanje v okolju .NET obsega štiri korake:

1. Načrtovanje in pisanje programske kode: prednosti izvajalnega okolja lahko izkoristimo le z uporabo prevajalnika, ki prevaja za CLR.

2. Prevod kode v MSIL: proces prevede izvorno kodo v MSIL in generira zahtevane meta podatke).

3. Proces prevede izvorno kodo: ob času izvajanja JIT prevajalnik prevede kodo MSIL v procesorsko odvisno kodo.

4. Izvedba kode: CLR omogoča izvajanje kode, kot tudi mnogo storitev, ki so na voljo med izvajanjem.

2.2. Spletni strežnik IIS

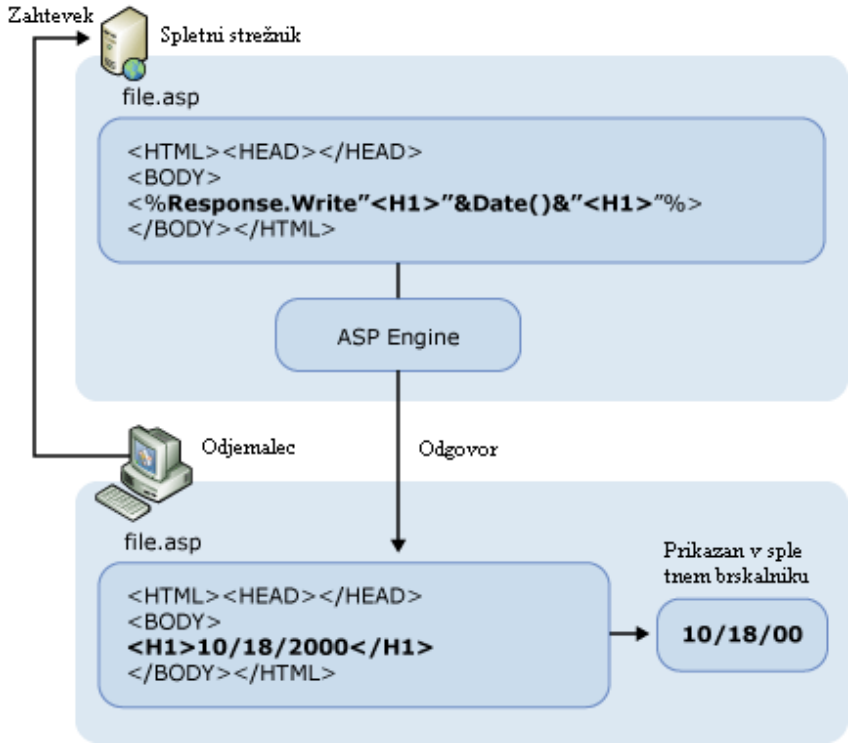
Spletni strežnik IIS (ang. Internet Information Services) je nekdanji niz internetnih storitev za strežnike, ki jih je Microsoft ponujal operacijskemu sistemu Microsoft Windows [10]. Je drugi najbolj priljubljeni spletni strežnik na svetu. Uporabili smo ga predvsem zato, ker je že v osnovni konfiguraciji nastavljen za uporabo Microsoftu lastne ASP.NET tehnologije, tako da smo preprečili težave z nastavitvami ali nezdružljivostjo.

2.3. Spletne tehnologije

V tem poglavju bomo predstavili spletni tehnologiji, ki sta bili uporabljeni za izdelavo uredniškega modula sistema.

2.3.1. Skriptni jezik ASP

Skriptni jezik ASP (ang. Active Server Pages), znan tudi kot klasični ASP, je bil prva Microsoftova strežniška razvojna tehnologija za izdelavo dinamičnih spletnih strani. ASP stran je v osnovi spletna stran (HTML), ki vsebuje skriptne ukaze, katere spletni strežnik obdela preden se stran pošlje brskalniku odjemalca [6]. Osnovni razvojni jezik je VBScript (Visual Basic Script). Primer uporabe ASP strani (slika 2) prikazuje izgled datoteke pred in po obdelavi na strežniku. Zahteva za ASP stran poteka popolnoma enako kot pri običajni HTML strani, vendar zahtevek lahko vsebuje tudi parametre.

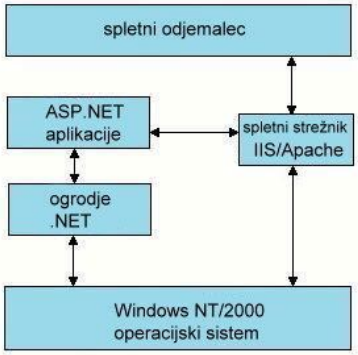


Slika 2 : ObdelavaASP datoteke

Poleg skriptne kode lahko ASP datoteka vsebuje tudi HTML kodo ter klice COM komponent, ki izvedejo vrsto nalog (povezava do baze podatkov, procesiranje poslovne logike, itd.).

2.3.2. Tehnologija ASP.NET

ASP.NET spletna tehnologija je naslednik skriptnega jezika ASP. Tehnologija ni nadgradnja starejše tehnologije, ampak zajema nove, inovativne gradnike za spletno programiranje. ASP.NET tako omogoča razvoj dinamičnih, interaktivnih spletnih aplikacij in spletnih servisov. Ogrodje vsebuje vrsto pripravljenih storitev in kontrol, ki pospešijo in olajšajo razvoj projekta [5]. ASP.NET je del ogrodja .NET Framework (slika 3), izvaja pa ga HTTP strežnik IIS ali Apache. Za izdelavo spletne aplikacije smo uporabili verzijo ASP.NET 2.0.



Slika 3 : Arhitektura ASP.NET

Spletna stran ASP.NET je fizično razdeljena na tri datoteke: na označevalni jezik (HTML), ASP.NET kodo (ang. code behind) in deklaracijo kontrol, ki jih prevajalnik (ang. compiler) združi. Tako je spletna stran razdeljena na tri datoteke s končnicami `.aspx`, `.aspx.cs` in `.aspx.designer.cs`.

Nekateri pomembni imenski prostori so:

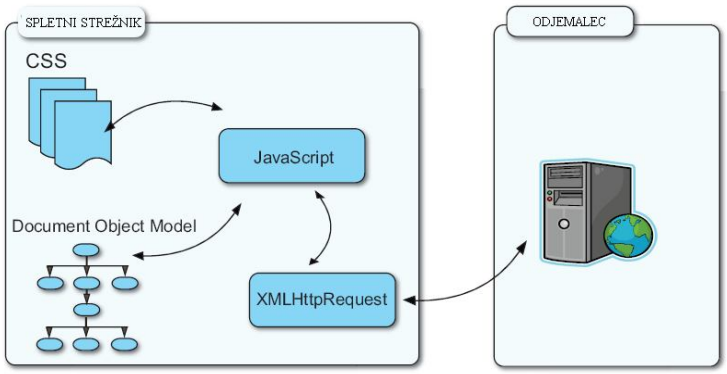
- System.Web
- System.Web.Util
- System.Web.Services
- System.Web.UI

2.3.3. JavaScript

JavaScript je objektno orientiran skriptni jezik, ki deluje na različnih platformah. Namenjen je vključevanju v ostale projekte ter aplikacije, ki so predvsem na odjemalčevi strani. V gostujočem okolju uporabimo JavaScript za povezavo do objektov znotraj okolja samega, s čimer pridobimo nad njimi programsko kontrolo [12].

Jedro JavaScript-a vsebuje več objektov (npr. `Array`, `Date`, `Math`) ter jezikovne elemente (operatorji, strukturne kontrole…). Jedro jezika JavaScript lahko uporabimo za različne namene, če mu podamo dodatne objekte:

- JavaScript na strani odjemalca razširja jedro z dodajanjem objektov, ki nadzorujejo spletni brskalnik (Navigator, IE, Firefox..) in njegov Dokumenti Objektni Model (DOM).
- JavaScript na strani strežnika razširja jedro z dodajanjem objektov, ki so pomembni za delovanje JavaScript-a na strežniku (komunikacija z relacijsko podatkovno bazo podatkov, manipulacija z datotekami na strežniku itd).



Slika 4 : Delovanje JavaScript-a

2.3.4. Ogrodje Adobe Flex

Adobe Flex (krajše Flex) je brezplačno odprtokodno ogrodje za ustvarjanje privlačnih spletnih aplikacij, ki jih je mogoče uporabljati v različnih brskalnikih, namiznih aplikacijah ter operacijskih sistemih z izvajalniki Adobe Flash Player in Adobe AIR. Glavni razlog, da smo uporabili Adobe Flex, je njegova zmožnost izdelave interaktivnih ter odzivnih aplikacij na preprost način [8].

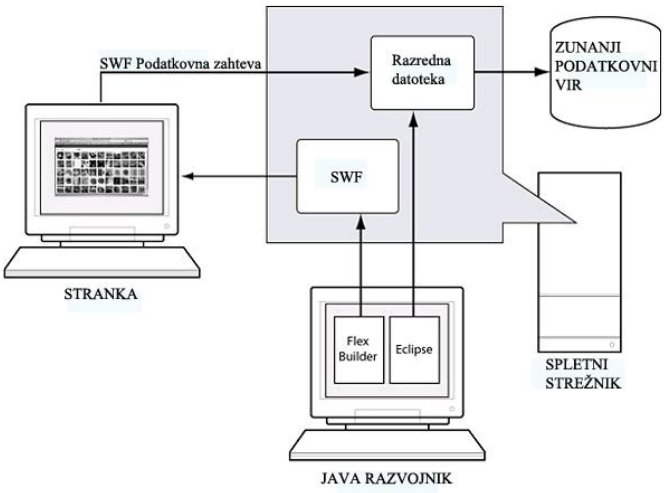
Bistvena lastnost Flex ogrodja je izgradnja spletnih aplikaciji, ki si dinamično izmenjujejo podatke s strežnikom brez potrebe po ponovnem nalaganju spletne strani. Ogrodje Flex ne omogoča direktne povezave do podatkovne baze, ampak uporablja spletni strežnik za pridobivanje ustreznih podatkov. Ena od poglavitnih prednosti ogrodja Flex je možnost uporabe s trenutno priljubljenimi spletnimi tehnologijami. Flex podaja možnost interakcije s strežniki, na katerih delujejo aplikacije, napisane v sledečih jezikih:

- JSP,
- ASP (ASP.NET),
- PHP,
- ColdFusion,
- drugi.

Za načrtovanje postavitve uporabniškega vmesnika uporablja Flex označevalni jezik MXML, ki temelji na XML-ju ter ActionScript-u kot skriptni programski jezik, s katerim lovimo dogodke MXML komponent, pošiljamo zahtevke za podatke ter nalagamo podatke nazaj v MXML komponente.

Flex aplikacije dinamično zahtevajo in pridobivajo podatke preko spletnega strežnika s pomočjo oddaljenih proceduralnih klicev (ang. Remote Procedure Calls).

Arhitektura ogrodja Flex je precej enostavna ter podobna ostalim strežniškim razvojnim ogrodjim. Primer delovanja: spletni odjemalec se poveže na spletno stran, na kateri je povezava do SWF datoteke, ki se nahaja na strežniku. Odjemalec naloži SWF datoteko v svoj Flash prevajalnik. Od te točke naprej aplikacija Flex pridobiva podatke preko prednastavljene spletne storitve, ki teče na spletnem strežniku. Spletni strežnik dobi podatke iz baze podatkov ali datotek ter jih pošlje nazaj aplikaciji Flex. Slika (slika 5) prikazuje sodelovanje med tipičnim spletnim strežnikom in aplikacijo Flex.

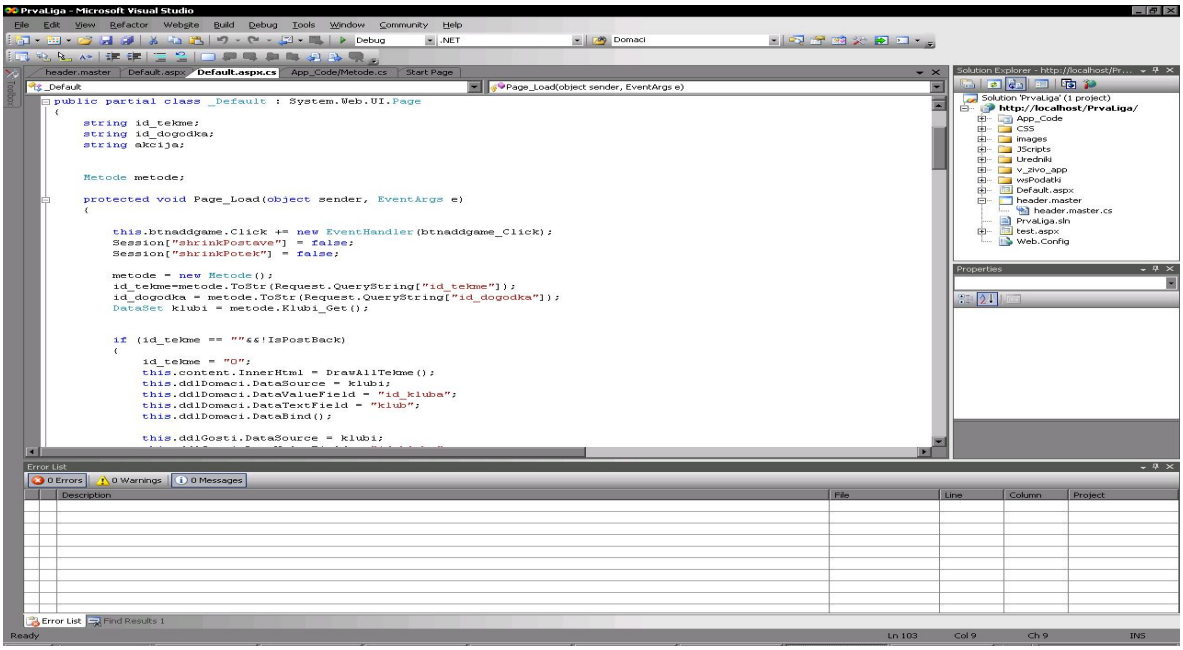


Slika 5 : Sodelovanje med spletnim strežnikom in Flash aplikacijo

2.4. Razvojna orodja

2.4.1. Microsoft Visual studio 2005

Microsoftovo razvojno orodje Visual Studio 2005 je zbirka razvojnih orodij, ki razvijalcem programske opreme pomaga premagovati kompleksne izzive in ustvarjati inovativne rešitve. Vloga zbirke Visual Studio je izboljšati razvojni proces ter omogočiti preprostejši razvoj programskih rešitev za Mirosoftovo okolje. Visual Studio 2005 je primeren za razvoj spletnih aplikacij (ASP.NET, AJAX), Windows, Windows Server, sistem Microsoft Office, SQL Server, igre (tako za PC kot Xbox) in naprave Windows Mobile [7].



Slika 6 : Razvojno okolje Visual Studio 2005

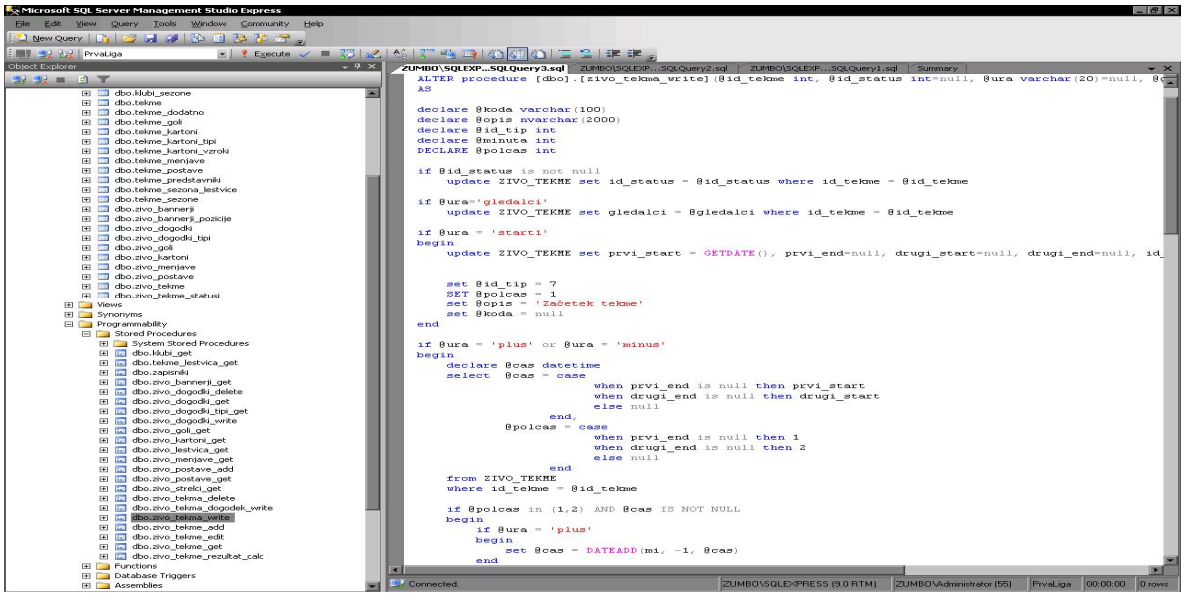
Razvojno okolje (slika 6) je sestavljeno iz urejevalnika, prevajalnika, razhroščevalnika, orodja za dokumentiranje programov ter drugih orodij, ki pomagajo pri pisanju programskih aplikacij. Visual Studio nudi podporo različnim programskim jezikom: Visual C#, Visual C++, Visual Basic in drugim. Aplikacije, ki so razvite v tem okolju, tečejo na vseh platformah, ki podpirajo ogrodje Microsoft.NET.

Visual Studio nam omogoča razvoj različnih vrst projektov, kot so:

- konzolni projekt (brez grafičnega uporabniškega vmesnika),
- projekti z obrazci (ang. Window forms),
- spletne aplikacije,
- spletni servisi (ang. Web Service),
- Windows servisi (ang. Windows Service)

2.4.2. Microsoft SQL Server 2005

Microsoft SQL Server 2005 Express je brezplačna različica SQL Server 2005. Orodje služi za delo z relacijskimi bazami podatkov (ang. Relational Database Management System- RDBMS) [11]. SQL Server Management Studio Express omogoča razvijalcem ter administratorjem pregledno in enostavno upravljanje podatkovnih baz s pomočjo različnih grafičnih orodji (slika 7).

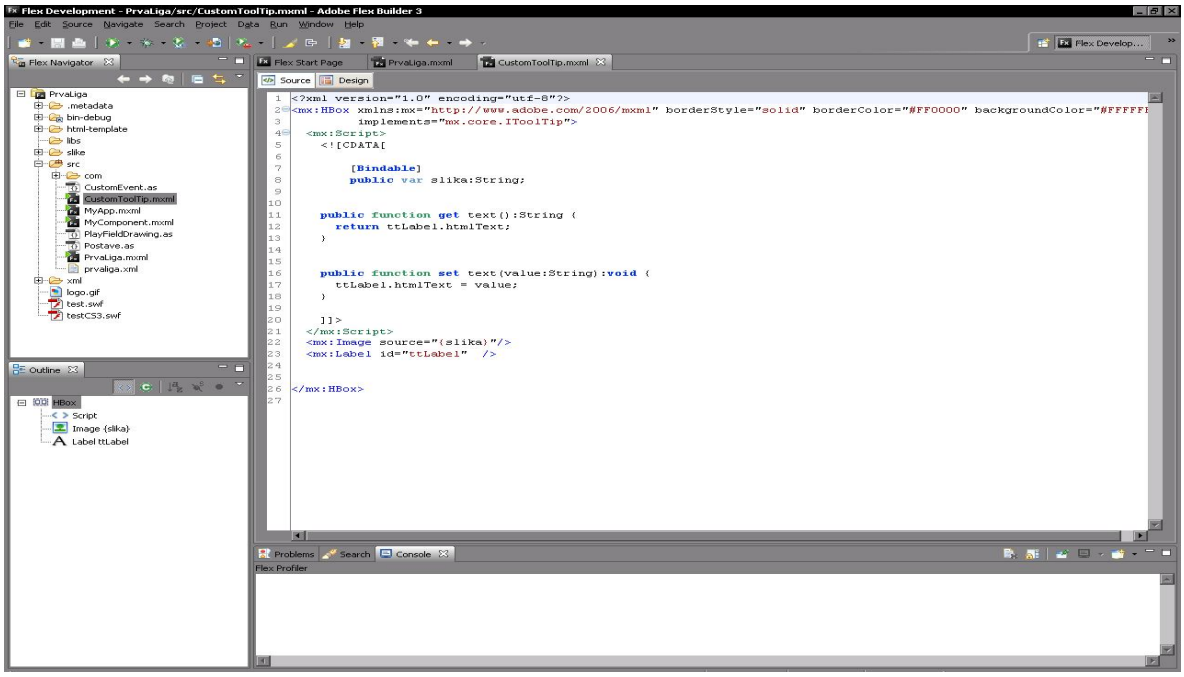


Slika 7 : SQL Management Studio

2.4.3.Adobe Flex Builder

Za razvoj predstavitvenega modula smo uporabili razvojno orodje Flex Builder 3 (slika 8). Orodje je integrirano razvojno okolje (ang. Integrated Development Environment), ki je razvit na platformi Eclipse. Namenjen je za razvoj obogatenih spletnih aplikacij ter namiznih aplikacij s poudarkom na Adobe Flash platformi [8].

Flex Builder poleg urejevalnika kode, prevajalnika ter razhroščevalnika za MXML ter ActionScript programsko kodo vsebuje tudi urejevalnik WYSIWYG (ang. What You See Is What You Get).



Slika 8 : Razvojno okolje Flex Builder 3

3. RAZVITA APLIKACIJA

V prejšnjem poglavju smo se posvetili opisu uporabljenih orodji ter tehnologij, ki so nam služile pri izvedbi diplomske naloge. V tem poglavju bomo podrobneje predstavili celoten postopek razvoja aplikacije od zahtev do končne podobe ter funkcionalnost razvitega sistema.

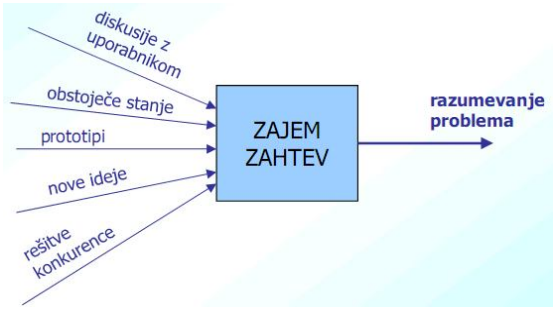
3.1. Pričakovanja naročnika

Celotni sistem temelji na jasno zastavljenih zahtevah naročnika. Osnovna ideja je bila izdelati sistem, ki bo primeren za spremljanje nogometnih dogodkov preko spleta (vendar ne v obliki video posnetka ali prenosa) ter za urejanje komentarjev in rezultatov tekem slovenske klubske in državne lige v spletni obliki.

Sistem naj bi bil preprost, pregleden za komentatorja ter privlačen za gledalca / odjemalca. Pričakovanja naročnika so bila, da bo uredniški modul uporaben za kogarkoli iz njihovih novinarskih vrst, oziroma, da bi vmesnik moral biti več ali manj intuitiven pri njegovi uporabi. Želje naročnika za uredniški del so obsegale več funkcionalnosti; med njimi so bile zahteve, da komentator vpisuje komentarje, kateri se prikazujejo na spletni strani, dodajanje in spreminjanje tekem, dodajanje golov, kartonov itd. Za predstavitveni del je bilo dogovorjeno, da ima odjemalec možnost spremljati tekmo z vsemi pomembnimi podatki ter da bo v barvni kombinaciji glavne spletne strani stranke ter sponzorskih barv.

3.2. Analiza zahtev

Za uspešno izvedbo razvoja aplikacije sta analiza in načrtovanje najpomembnejša dejavnika. Z njima lahko dosežemo boljše razumevanje sistema, ga poenostavimo ter zagotovimo ponovno uporabo, vizualizacijo in nadzor nad arhitekturo sistema [13].



Slika 9 : Zajem zahtev

Najlažji način za zajem zahtev je neposredna komunikacija z udeleženci sistema. V razgovoru neposredno komuniciramo z ljudmi, ki so udeleženi v procesih, katere želimo podpreti. S pomočjo predlogov naročnika ter z našimi lastnimi vprašanji izluščimo vse potrebne funkcije, ki jih mora sistem podpirati.

Poglavje je namenjeno predstavitvi: nefunkcionalne zahteve sistema, funkcionalne zahteve sistema ter idejne rešitve.

3.2.1. Nefunkcionalne zahteve

V sledečem poglavju bomo podali najpomembnejše nefunkcionalne zahteve, ki se nanašajo na tehnične ter ostale ne-vsebinske zahteve celotnega sistema.

1. Strojna in programska oprema

   Glede strojne opreme so zahteve bolj skromne zaradi relativno nezahtevne aplikacije, gledano s tehnične plati. Zadostna strojna oprema je osebni računalnik z dostopom do spleta. V okviru programske opreme pa potrebujemo IIS, .NET FrameWork 2.0, ASP.NET, Microsoft SQL Server ter Adobe Flash Player. Zagotoviti moramo tudi podporo JavaScript za spletni brskalnik.

2. Upravljanje s podatki

   Sistem mora zagotoviti učinkovitost tudi v primeru obravnavanja večje količine podatkov. Zaradi manjše količine podatkov našega sistema v tem oziru zahteva ni predstavljala težav.

3. Zahteve glede zmogljivosti

   Ker je aplikacija namenjena za spremljanje nogometnih tekem v živo, mora sistem omogočati tekoče delovanje zaradi nenehnega osveževanja podatkov. Ostale zmogljivosti sistema so pogojene z zmogljivostjo strežnika ter odjemalčevega osebnega računalnika, vključno s hitrostjo internetne povezave.

4. Varnost

   Sistem mora omogočati varno dostopanje do podatkov na bazi. Vsaka poizvedba v bazi naj bo klic shranjene procedure. Če imamo možnost, sistem razvijemo v večnivojski arhitekturi, tako da so klici shranjenih procedur ločeni od preostale logike sistema in zato težje dostopni.

6. Uporabniški vmesnik

   Uporabniški vmesnik mora biti preprost za uporabo, pregleden, interaktiven ter pomirjujočih barv.

7. Uporaba ustrezne tehnologije

   Tehnologija za predstavitveni modul mora omogočati izdelavo interaktivne ter privlačne uporabniške aplikacije, vendar ne na račun hitrosti in uporabnosti. Uredniški modul mora biti preprost in intuitiven, zato naj bo usmerjen k uporabnosti in hitrosti delovanja.

3.2.2. Funkcionalne zahteve

Funkcionalne zahteve so zahteve, ki se nanašajo na željene funkcionalnosti, ki naj bi jih sistem omogočal. Ker naročnik ni podal popolnega seznama vseh zahtev, smo morali sami zajeti zahteve v razgovoru z bodočimi uporabniki sistema.

Funkcionalne zahteve so zapisane v seznamu, ločeno za posamezen del aplikacije (uredniški ter predstavitveni modul).

Sistem pozna dva akterja:

- Uporabnik / gledalec: obiskovalec spletne strani,
- Urednik / komentator: obiskovalec, prijavljen v uredniški vmesnik z določenimi pravicami.

Zahteve za predstavitveni modul

Uporabniku so omogočene sledeče funkcionalnosti:

1. Uporabnik lahko spremlja tekmo, ki se dogaja v živo, kar vključuje vpogled sledečih podatkov:
   - lokacija tekme,
   - ime sodnika,
   - število gledalcev,
   - trenutni rezultat,
   - komentarji, napisani s strani komentatorja tekme,

- postave,
- statistika tekme.

2. Uporabnik lahko sam osvežuje dogajanje, brez čakanja na osvežitev vseh podatkov.
3. Uporabnik lahko vidi grafični prikaz dogodkov (zadetki, kartoni, menjave, prekrški) na časovnem traku.
4. Uporabnik lahko vidi statistiko strelcev trenutne sezone (ki se sproti osvežuje v skladu s tekmo).
5. Uporabnik lahko vidi lestvico trenutne sezone, ki se ravno tako sproti osvežuje v skladu s tekmo.
6. Uporabnik lahko spremlja več tekem naenkrat (t.i. konferenčni način).

Zahteve za uredniški modul

Uredniku so omogočene sledeče funkcionalnosti:

1. Ker ima uredniški modul omejen dostop, urednika določi administrator celotnega spletnega portala, samostojna registracija pa ni možna. Pri tem mora bodoči urednik podati ime, priimek, uporabniško ime, elektronski naslov ter geslo.
2. Po uspešni registraciji se urednik lahko prijavi v sistem. Za prijavo potrebuje uporabniško ime in geslo.
3. Urednik ima možnost odjave iz sistema. To preprečuje možnost zlorabe njihovih računov.
4. Urednik lahko doda nove tekme v sistem, pri čemer mora vnesti vrsto podatkov (domača ekipa, gostujoča ekipa, sodnik, stadion, kraj, datum…). Podatke lahko ustrezno spreminja tudi na že vnesenih tekmah.
5. Ko je tekma vnesena, lahko urednik spreminja in dodaja vrsto podatkov, ki se tičejo tekme (vsaka sprememba se odraža tudi na predstavitvenem vmesniku):
   - status tekme (pred tekmo, v teku, odigrana),
   - število gledalcev,
   - dodajanje/brisanje minut,
   - dodajanje/brisanje zadetkov, kartonov, menjav…,
   - pisanje/brisanje/popravljanje poteka tekme
   - shranjevanje ter nalaganje zapisnika v datoteko tipa .ZAP

3.2.3. Idejna rešitev

Celoten sistem naj bi sestavljala dva ločena modula. Prvi modul naj bi bil namenjen uredniku, drugi pa odjemalcu. Uredniški modul naj ne bi bil pretirano tehnološko zahteven, saj je nabor podatkov za obdelavo relativno preprost in ne pretirano obsežen. Uredniška aplikacija predstavlja del večjega sistema za upravljanje s spletnimi stranmi**.** Glavni sistem je razvit v klasični tehnologiji ASP, njegov podatkovni del pa je razvit s pomočjo orodja SQL Server in tehnologije za podatkovne transakcije (ang. Transact SQL).

Uporabljena tehnologija glavnega sistema je botrovala k izboru tehnologije za razvoj uredniškega dela. Razlog leži predvsem v zahtevani medsebojni tehnološki uskaljenosti modula in glavnega sistema. Pomemben razlog so tudi izkušnje z vrsto razvitih aplikacij v klasičnem ASP-ju s podobnim načinom dela. Celoten vmesnik smo si zamislili kot razširjeni ali ustrezno spremenjen sistem za upravljanje s spletnimi stranmi: namesto upravljanja spletnih dokumentov bi sistem služil upravljanju komentarjev, dogodkov tekme, itd.

Ker je večina dogodkov v tekmi vezanih na določenega igralca in bi bilo pri vsakem vnosu dogodka zelo zamudo ročno zapisovati ime in priimek igralca, je bilo potrebno poenostaviti vnos. Nastala je zamisel o podatkovni mreži (ang. Data Grid) z igralci in gumbi, ki so vezani na njih. Vsak gumb predstavlja eno vrsto dogodka. Nadaljne olajšave za delo komentatorja obsegajo tudi shranjevanje ter uvoz zapisnika (shranijo se vsi podatki komentiranja).

Naslednji izziv nam je predstavljal izgled ter način prikazovanja predstavitvenega modula. Potrebno je bilo izdelati spletno aplikacijo, ki bi bila čimbolj prenosljiva brez večjih predelav, kompatibilna s čim večjim naborom različnih brskalnikov ter predvsem privlačna in interaktivna.

Čeprav veliko spletnih tehnologij nudi možnost izdelave takšnih aplikacij, se nam je zdel najprimernejši Adobe Flash (bolj natančno ogrodje Adobe Flex). Ogrodje Adobe Flex že vsebuje velik nabor interaktivnih kontrol s privlačnim izgledom ter vrsto nastavljivih vizualnih efektov. Poleg tega je tudi izvedba za programerja prijaznejša od ostalih, nabor funkcij pa večji. Glavni razlog za izbor ogrodja Adobe Flex je bila kompatibilnost s skoraj vsemi spletnimi brskalniki, ki zahtevajo le nameščeno aplikacijo Adobe Flash Player (predvajalnik aplikacij, izdelanih v ogrodjih Adobe Flex in Flash). Adobe Flash Player je večinoma že vgrajen v večino brskalnikov.

Aplikacija SWF (datoteka, ki je produkt ogrodja Adobe Flex) je prenosljiva med spletnimi aplikacijami brez večjih težav in sprememb pri namestitvi. Aplikaciji je potrebno le podati vir podatkov, če ga le-ta potrebuje.

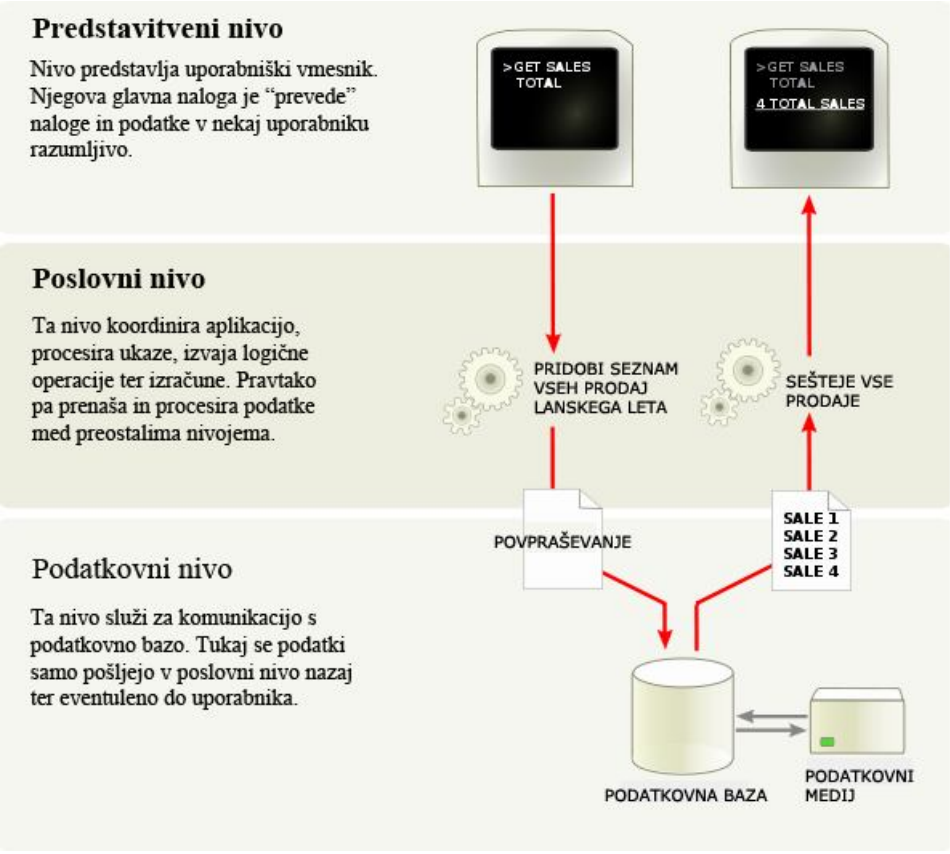
Pri izgledu modula smo se odločili, da se zgledujemo po obstoječem sistemu LiveTicker (spletna aplikacije za spremljanje nogometnih dogodkov nemške lige) [14]. Sistem je bil prav tako izdelan s tehnologijo Adobe Flex in njegove lastnosti so več ali manj ustrezale našim zahtevam (uporabljene kontrole, razvrstitev, način prikazovanja, itd). Celoten koncept sistema smo kljub temu predelali in mu dodali lastne rešitve, tako da je izpopolnjeval vse zahteve.

3.3. Uredniški modul

Modul v celoti obsega:

- uredniško aplikacijo za urejanje podatkov in
- aplikacijo za pretvarjanje podatkov v ustrezno obliko za predstavitveno aplikacijo,

V skladu z zahtevami naloge je bil določen del aplikacije, t.j. uredniški modul, razvit tudi v novejši tehnologiji ASP.NET. Tehnologija nam je omogočala bolj strukturiran razvoj kot uporaba klasične tehnologije ASP, saj smo uporabili pristop trinivojske arhitekture (slika 10), kjer so podatkovni, poslovni ter predstavitveni deli aplikacije ločeni po nivojih.



Slika 10 : Trinivojska arhitektura

Predstavitveni nivo

Prvi nivo arhitekture je predstavitveni nivo, ki predstavlja uporabniški vmesnik. Objekti se preko posebnih razredov prenašajo iz poslovnega nivoja v uporabniški vmesnik, kjer so kot podatki prikazani na uporabniku prijazen način, s tabelaričnimi ter vnosnimi kontrolami.

Poslovni nivo

Poslovni nivo predstavlja najpomembnejši del arhitekture, ki poleg ostalih nalog opravlja tudi funkcijo povezovalnega člena med predstavitvenim in podatkovnim nivojem. Poslovni nivo vsebuje vso poslovno logiko, pretvorbo podatkov med obliko za prikaz in obliko za shranjevanje oz. pretvorbo v obratni smeri, ocenjevanje usklajenosti podatkov itd.

Podatkovni nivo

Podatkovni nivo prevzema celotno funkcionalnost dostopa do podatkovnega vira. Služi za vzpostavljanje povezav na podatkovno bazo in izvajanje dejanskih SQL ukazov. Proces poteka preko t.i. shranjenih procedur (ang. stored procedures), kar v veliki meri pospeši delovanje podatkovnega strežnika. Pri pridobivanju podatkov pošljemo strežniku le ime shranjene procedure, ne pa celotne SQL poizvedbe, kar zmanjša količino prenesenih podatkov.

Shranjena procedura je na strežniku sintaktično pravilna, prevedena in poizvedbeni plan je že pripravljen. Tako mora strežnik v trenutku, ko dobi zahtevo, le izvesti že pripravljeni izvedbeni plan in vrniti rezultat poizvedbe. Hitrost izvajanja takega stavka SQL je odvisna od same zahtevnosti poizvedbe.

Ločen podatkovni nivo poenostavi marsikaj, saj so vsi stavki SQL-a in celotna logika dostopa do baze na enem preglednem mestu. Plast podatkovnega dostopa opravlja sledeče naloge:

- operacije za dodajanje, branje, shranjevanje in brisanje podatkov,
- zaklepanje tabel in nadzor nad hkratnim ažuriranjem zapisov,
- nadzor nad preverjanjem istovetnosti identitete,
- upravljanje s transakcijami,
- pomnjenje že prebranih podatkov in preverjanje, kdaj jih je treba ponovno prebrati,
- posredovanje morebitnih napak.

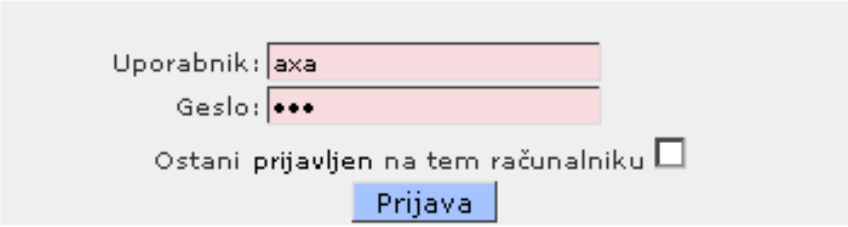
3.3.1. Izgled in delovanje

V tem razdelku bomo podrobneje predstavili glavni element uredniškega modula, ki je namenjen urejanju podatkov s strani urednika. Del uredniškega modula, namenjen pretvorbi podatkov, nima uporabniškega vmesnika, zato je prikaz njegovega izgleda nesmiseln, samo delovanje pa je omejeno na pretvorbo podatkov v zahtevano obliko.

Prijava

Za prijavo v sistem (slika 11) je potreben vnos novega uporabnika s strani administratorja. Uporabnik je določen kot urednik modula za urejanje in komentiranje nogometnih tekem.

Pri prijavi v sistem potrebujemo uporabniško ime in geslo. Imamo tudi možnost izbrati polje `Ostani prijavljen na tem računalniku`, pri čemer se ob prijavi pošlje uporabniku piškotek, z namenom, da se ob naslednji prijavi iz njega preberejo zahtevani podatki. S tem izničimo potrebo po ponovnem vnašanju uporabniškega imena ter gesla, vendar to velja le do izteka seje (20 minut).

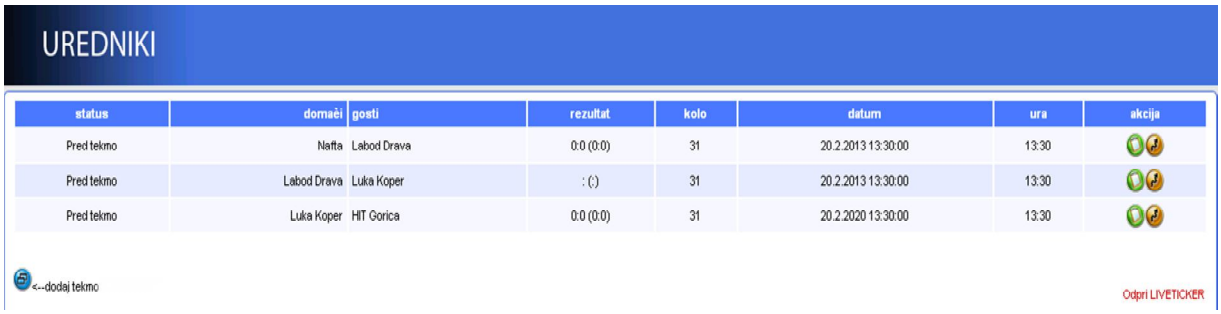


Slika 11 : Prijava v uredniški vmesnik

Prijava je uspešna, ko obstaja natanko eno uporabniško ime in geslo, ki ga je administrator dodal v podatkovno bazo.

Osnovni vmesnik

Po uspešni prijavi v sistem se uporabniku izriše prvi, preprostejši del delovnega okolja (slika 12). Vmesnik je namenjen prikazu ter upravljanju vnešenih tekem.



Slika 12 : Prvi del delovnega vmesnika

Ob kliku na gumb `dodaj tekmo` se nam odpre dodatni nabor možnosti (slika 13), ki so namenjene dodajanju novih tekem. Vnos od nas zahteva določene obvezne podatke (imena obeh ekip, datum, ura, kraj, stadion ter glavni sodnik).

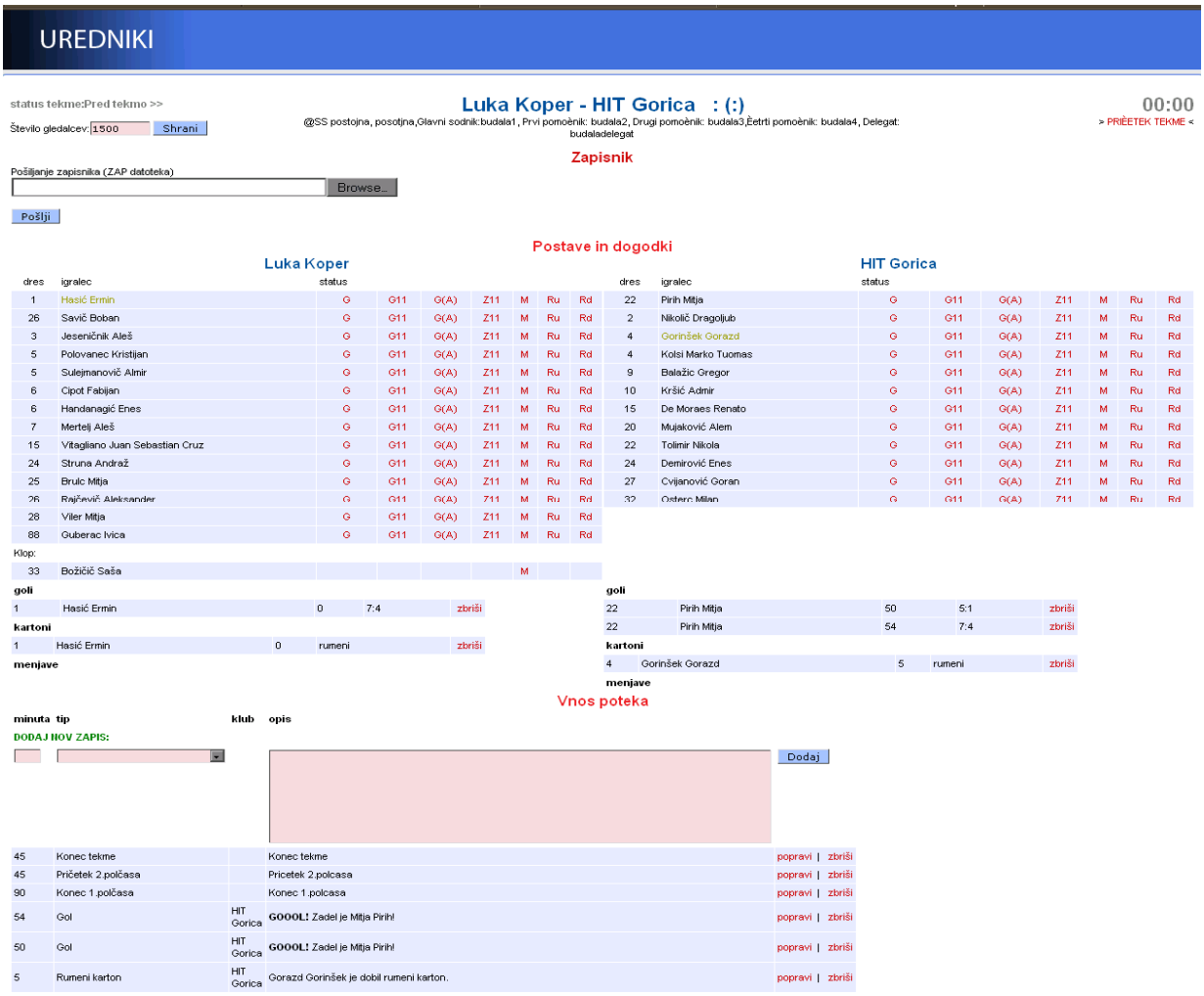
Slika 13 : Drugi del delovnega vmesnika

Vsaka tekma ima dva posebna gumba (slika 14). S pritiskom na gumb `komentiraj` se nam prikaže komentatorski vmesnik, s klikom na gumb `uredi tekmo` pa lahko z uporabo istega nabora možnosti (slika 13) spreminjamo podatke izbrane tekme.

Slika 14 : Gumbi za urejanje in komentiranje

Komentatorski vmesnik

S pritiskom na gumb za urejanje tekme preidemo na drugi del delovnega okolja (slika 15), ki je namenjen komentiranju izbrane tekme v živo. Vmesnik je razdeljen na tri elemente: **Zapisnik**, **Postave in dogodki** ter **Vnos poteka**.

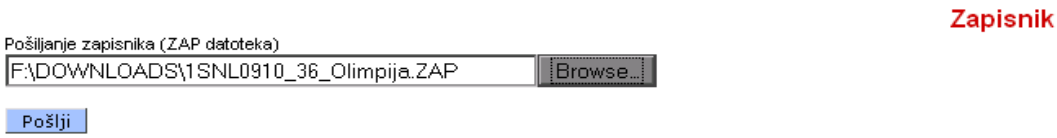


Slika 15 : Komentatorski vmesnik

V osrednjem zgornjem delu so osnovni podatki tekme, v levem zgornjem kotu pa je možno spreminjati status tekme (pred tekmo, v teku, odigrana) ter število gledalcev. V zgornjem desnem kotu je večje število kontrol, ki nadzorujejo časovni potek tekme. Z njimi je možno spremljati čas odvijanja tekme (v slučaju napake komentatorja se lahko doda ali odbije kako minuto) ali pa sproži začetek tekme. Potek tekme je razdeljen na tri dele (pričetek tekme, prvi polčas, drugi polčas), katere lahko komentator nastavlja glede na potek tekme.

Zapisnik

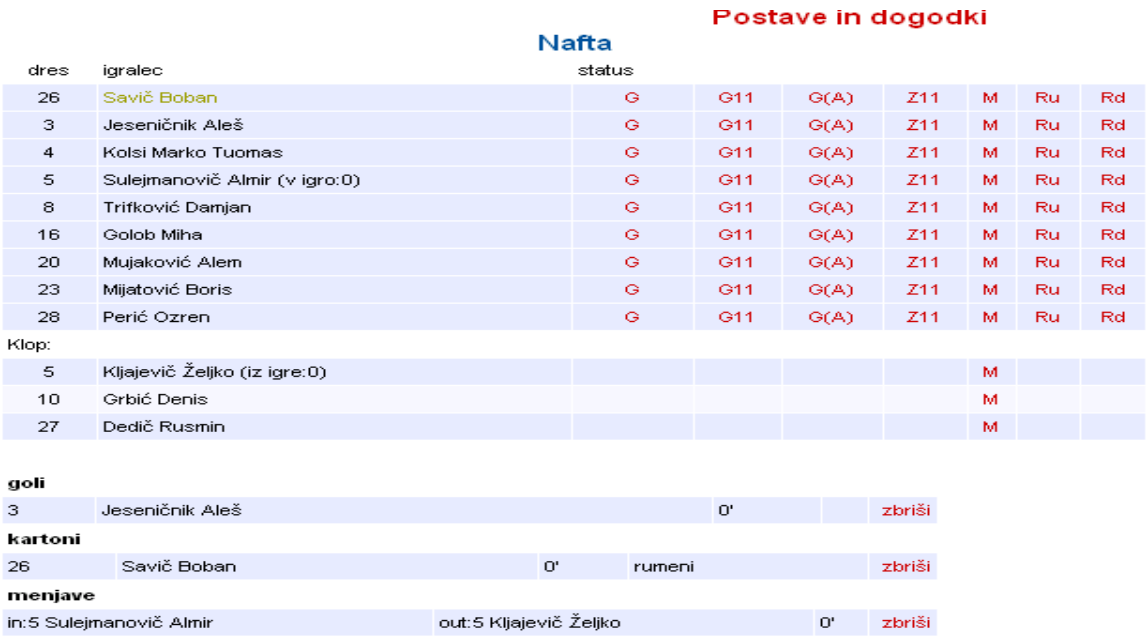
Prvi ločeni element vmesnika (slika 16) je namenjen shranjevanju vseh trenutnih podatkov tekme (postave, dogodki, komentarji…). Shranjeni zapisnik tekme služi kot varnostna kopija (ang. backup) v slučaju napak. Potrebno je določiti ime datoteke ter s pritiskom na gumb `pošlji` podatke shranimo v datoteko tipa .ZAP (navadna tekstovna datoteka, ki ima spremenjeno končnico zaradi lažjega ločevanja od preostalih datotek). Datoteko lahko administrator po potrebi uporabi za obnovitev vseh podatkov zapisnika.



Slika 16 : Shranjevanje zapisnika

Postave in dogodki

Osrednji element (slika 17) predstavlja glavni del celotnega vmesnika.



Slika 17 : Vmesnik postave in dogodki

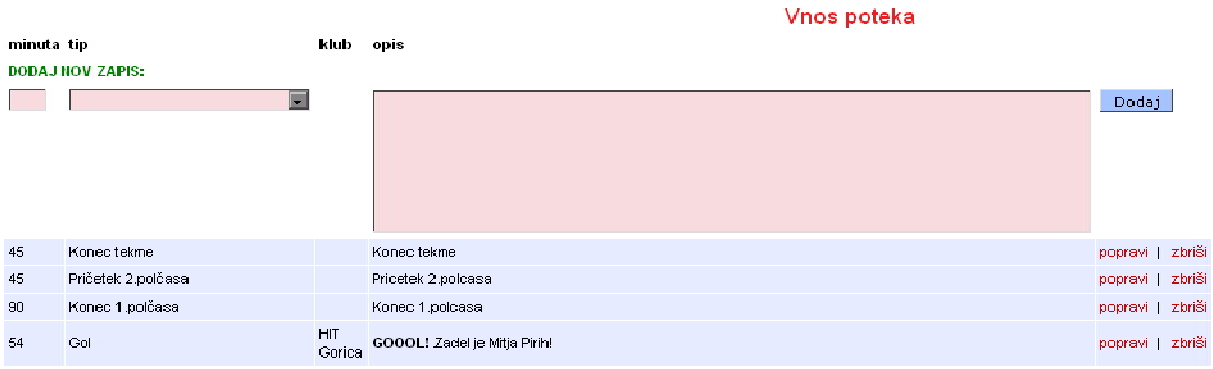
V njem komentator vnaša dogodke, vezane na določenega igralca. Tudi ta del je sestavljen iz več elementov:

- postave (prikazane so postave obeh ekip v igri),
- klop (igralci, ki niso v igri),
- dogodki (doseženi zadetki, dodeljene kazni, menjave).

Vsak dogodek (gol, avtogol, enajstmetrovka, menjava, rumeni karton, rdeči karton) je vezan na svoj gumb, ta pa na določenega igralca. Primer: s klikom na gumb za gol se doda zapis pod delom **goli**. Zapis je sestavljen iz zapisa minute, v kateri je bil dosežen zadetek, iz imena igralca, ki je dosegel gol, ter iz gumba za izbris omenjenega zadetka. Za lajšanje dela komentatorjem se hkrati avtomatsko generira nov komentar v vnosu poteka tekme, s čimer se izognemo ročnemu vpisovanju ponavljajočih se dogodkov. Na isti način delujejo tudi vnosi preostalih dogodkov.

Vnos poteka

Poslednji element vmesnika je vnos poteka tekme (slika 18). Vnos novega komentarja vsebuje zapis minute dogodka, tip dogodka ter besedilo komentarja. Po vnosu vseh potrebnih podatkov dodamo komentar v tabelo vseh komentarjev s pritiskom na gumb `dodaj`. Vsak zapis je sestavljen iz zapisa minute, tipa dogodka, ekipe, na katere je vezan dogodek (vendar ne nujno), iz komentarja ter gumba `popravi` in `briši`. Kot lahko sklepamo iz njunega poimenovanja, sta gumba namenjana spreminjanju ter brisanju izbranega komentarja v seznamu.



Slika 18 : Vnos poteka

Na koncu naj omenimo tudi, da se vsi vnosi ali spremembe v uredniški aplikaciji odražajo tudi v aplikaciji znotraj uredniškega modula, ki služi kot vir podatkov predstavitvenemu modulu.

3.3.2. Izvedba

Izvedbo obeh različič uredniškega modula in primerjavo med njima je podrobneje predstavljeno v 4. poglavju.

3.4. Predstavitveni modul

Aplikacija omogoča pregled dogajanj slovenske nogometne lige. Ker je bilo potrebno zagotoviti odjemalcem čimbolj prijazno in dostopno mesto za pregled tekem, je spletni nogometni portal predstavljal idealno rešitev. Spletna aplikacija dostopa do podatkov preko elementa uredniškega modula, ki generira podatke v XML strukturi. Podatki se pri tem preberejo iz podatkovne baze na lokalnem strežniku.

3.4.1. Izgled in delovanje

Sama aplikacija ne zahteva prijave ali registracije. Odprta je vsem uporabnikom, ki spremljajo portal, na katerem se nahaja. Vmesnik je razdeljen na štiri dele. Vsak del omogoča pregled določenih podatkov. Ti deli so sledeči:

- okno z ekipami ter rezultatom,
- časovni trak dogodkov,
- različni seznami (tekme, lestvica, strelci),
- glavno okno (komentarji, statistika tekme, postave, gumb za ročno osveževanje ter izbira filtra dogodkov).

Predstavili bomo poglavitne dele vmesnika (slika 19), ki se nanašajo na spremljanje tekem v živo.

Slika 19 : Celotni vmesnik predstavitvene aplikacije

Okno, ki se v celotni postavitvi nahaja skrajno levo, je namenjeno prikazu obeh ekip ter rezultatu tekme (slika 21). Ko kliknemo na gumb `VSE TEKME` (slika 20), se odpre posebni način prikazovanja (t.i. konferenčni pregled), ki omogoča pregled vseh tekem sezone naenkrat.

Slika 20 : Normalni pregled

Slika 21 : Konferenčni pregled

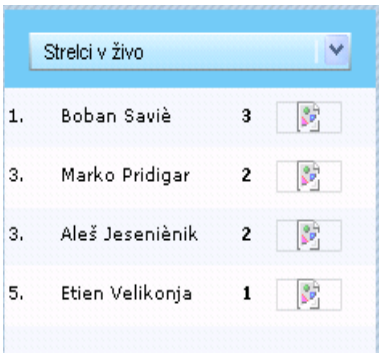
Časovni trak v desni zgornji legi ima funkcijo vizualnega prikazovanja dogodkov, ki jih vnaša komentator v skladu s potekom tekme. Vsak dogodek ima na traku narisano ustrezno sliko ter minuto, ko se je dogodek zgodil. V sredini se nahaja trenutna minuta igranja izbrane tekme.

V levem spodnjem delu je kontrola, ki opravlja tri funkcije:

- prikaz tekem,
- prikaz trenutne lestvice sezone,
- prikaz najboljših strelcev sezone.

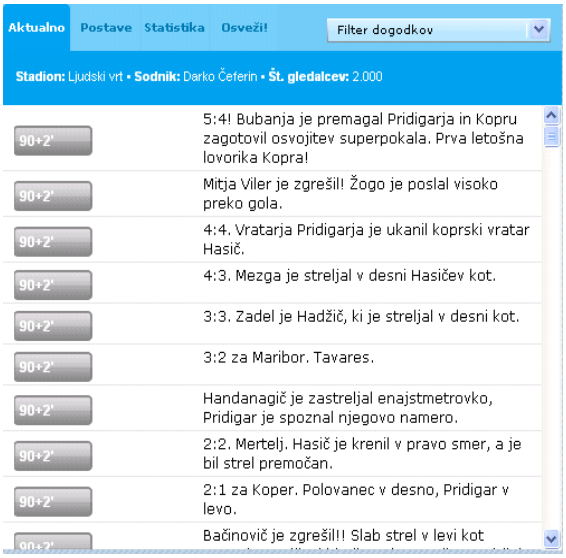
Prikaz tekem je sestavljen iz obeh ekip, ki sodelujeta v tekmi. V sredini se nahaja rezultat ali datum, ko se bo tekma odvijala. Prikaz lestvice (slika 23) ter prikaz strelcev (slika 24) omogoča pregled trenutne razvrstitve v sezoni.

Slika 22 : Lestvica ekip



Slika 23 : Strelci sezone

Večino vmesnika zaseda osrednji komentatorski del, ki je poglavitni element celotnega modula. Vidni so različni podatki, ki pripadajo izbrani tekmi. Tudi ta del je razdeljen na tri segmente (**Aktualno**, **Postave**, **Statistika**) ter ločeni prikaz osnovnih podatkov tekme (kraj, sodnik, število gledalcev). Vsi podatki segmentov se preberejo iz podatkovne baze, ki jo uporablja tudi uredniški modul. Vsak gumb sproži svoj dogodek (ang. event), ob katerem se iz baze prebere ustrezen podatke, ki jih določeni dogodek zahteva. Ob tem se spremeni barva izbranega gumba. Gumbi, poimenovani enako kot segmenti, so: `Aktualno`, `Postave` in `Statistika`.



Slika 24 : Aktualni dogodki na tekmi

Slika 25 : Postave na tekmi

Slika 26 : Statistika tekme

Opis gumbov:

- `Aktualno` (vsi komentarji ter dogodki tekme, ki jih vnese komentator. Sestavljeni so iz minute dogodka, ekipe dogodeka ter besedila, slika 25),
- `Postave` (igralci tekme po ekipah, poleg se pa nahajajo tudi vsi dogodki, slika 26),
- `Statistika` (statistika vrsto podatkov vezanih na izbrano tekme, slika 27).

Komentarje dogodkov je možno tudi filtrirati glede na tip dogodka. Uporabnik ima možnost ročno osvežiti vse podatke s pritiskom na gumb `Osveži`.

3.4.2. Izvedba

V tem poglavju bomo na kratko opisali poglavitne elemente razvoja predstavitvenega modula. Celotni modul je razvit v okolju Adobe Flex s pomočjo razvojnega orodja Flex Builder 3. Orodje podpira avtomatsko dokončanje kode (dokončevanje klica funkcije) tako kot večina preostalih razvojnih orodij IDE (ang. Integrated Development Environment). Flex Builder 3 omogoča tudi vpogled v izgled strani, ki je ločen od kode in kjer je možno postavljati nove kontrole s preprostim sistemom »povleci in spusti«. Ob tem se ustrezno generira tudi koda kontrole.

Aplikacija nudi s svojim bogatim naborom kontrol pregleden ter privlačen način spremljanja nogometnih dogodkov preko spleta. Prednost teh kontrol pred kontrolami preostalih razvojnih ogrodij je predvsem v velikosti izbora ter v nastavljanju vizualnih učinkov, ki delujejo na njih. Pri ostalih ogrodjih je praviloma večino vizualnih učinkov potrebno ročno ustvariti. V našem

primeru po tem ni potrebe, ker je veliko število vizualnih učinkov že vgrajenih. Delo z njimi je preprosto, saj potrebujemo le ID (identifikacijo) kontrole, na katero se efekt nanaša (priloga A).

Kakor je razvidno iz kode (priloga A), smo kontroli določili, da se ob sprožitvi dogodka »skrije« ali »prikaže« z izbranimi učinki, ki imajo določen čas trajanja ter njim lastne parametre. Učinke je možno klicati znotraj metod in jih zaganjati ročno (priloga B).

Učinek zaženemo z ukazom `play` ter končamo z ukazom `end`. Predstavitveni modul ni sestavljen iz večih nivojev kakor uredniški, vse deklaracije kontrol in metod so v isti datoteki. Aplikacija pridobiva podatke (priloga C) iz virov, ki ji niso lastni, ker to ogrodje ne podpira neposredne komunikacije s podatkovno bazo. V našem primeru ji podatke dostavlja uredniška aplikacija, napisana v različno tehnologiji kot predstavitveni modul.

Bistvena prednost omenjenega načina dostopa do podatkov je preglednost aplikacije. Ker naša aplikacija zahteva nenehno osveževanje zaradi narave podatkov (podatki se spreminjajo v živo), bi izbira druge tehnologije botrovala k temu, da bi se spletna stran odjemalcu stalno osveževala. V našem primeru pa se podatki obnavljajo brez motečega osveževanja celotne spletne strani.

Ogrodje podpira objektni pristop, zato je možno pisanje lastnih razredov ter znotraj njih metode. Klicanje teh objektov poteka podobno kot v drugih ogrodjih, ki podpirajo OOP (priloga Č).

3.4.2.1. Polnjenje podatkov ob zagonu aplikacije

1. Razlaga kode polnjenja podatkov (priloga F):

- `[Bindable] public var LestvicaXML:XML = new XML();`

  Z deklaracijo kreiramo objekt tipa XML, v katerega kasneje shranimo prebrano XML strukturo.

  Obvezno moramo dodati metapodatkovno oznako `[Bindable]`, ki pove prevajalniku, da bomo na spremenljivko vezali podatke, ki jih je kasneje možno spreminjati v določenih dogodkih. Spremenljivka je globalna.

- `LINKSERVER = Application.application.parameters.link;`

  S kodo podamo spremenljivki `LINKSERVER` vrednost parametra (v tem primeru povezavo do vira podatkov), ki smo ga podali ob klicu datoteke .SWF znotraj datoteke HTML (`<param name="link" value="http://localhost/prva-liga/wsPodatki/" />`).

- `URL = new URLRequest(XML_URL);`
  `Loader = new URLLoader();StrelciLoader.load(StrelciURL);`

  Razred `URLRequest` nam »ujame« vse podatke v enem HTTP zahtevku. Njegovi objekti so nato poslani metodi `load()` razreda `URLLoader`, ki prebere podatke v tekstovni, binarni ali URL-kodirani obliki.

3.4.2.2. Risanje elementov na časovni trak

1. Razlaga kode risanja elementov (priloga G):

- `DrawObjects()`

  Funkcija izvaja risanje dogodkov na časovni trak glede na ekipo ter minuto igre. Pred vsakim izvajanjem najprej preveri, če narisani elementi že obstajajo. Če obstajajo, jih pobriše ter ponovno izriše vse obstoječe elemente skupaj z novimi.

- `DoesExist(xcoor:int,ycoor:int):Boolean`

  Funkcija preverja, če na podanem mestu že obstaja element. Če obstaja, funkcija zamakne njegovo koordinato na ordinatni osi, da se elementa ne prekrivata.

- `AddItemToCanvas(minuta:int,item:String,ekipa:int,igralec_in:String,igralec_out:String,vzrok:String,rezultat:String,minutaT:String,avtogol:int);`

  Funkcija nariše vsak element posamično na časovni trak. V skladu s tipom dogodka se nastavi ustrezna slika, ki predstavlja tip dogodka.

3.4.2.3. Časovno osveževanje tekme v živo

1. Razlaga kode časovnega osveževanja (priloga H):

- `TimerTekmaHandler(event:TimerEvent)`

  Funkcija se izvede ob vsakem »tick-u« časovnega merilnika (ang. Timer). Ob vsakem klicu osvežuje vse potrebne podatke. Kliče se vsaki dve sekundi.

- `LoadPotek();`

  Funkcija za nalaganje podatkov o poteku tekme (komentarji).

- `LoadZapisnik(ForceRefresh:int)`

  Funkcija je podobna prejšnji funkciji, le da osvežuje drugi tip podatkov (časovni trak, statistika...). Če funkcija zazna, da je način prikazovanja konferenčni način, ne osveži podatkov, ker je v tem načinu zapisnik neaktiven.

4. PRIMERJAVA TEHNOLOGIJ IN RAZLIKE V RAZVOJU

V skladu z zahtevami diplomske naloge je uredniški modul razvit tako v klasičnem ASP-ju kot v ASP.NET različici. Ne gre le za nadgradnjo določenih elementov aplikacije, ampak za dve popolnoma ločeni aplikaciji, pri čemer je vsaka bila razvita v svoji tehnologiji. (koda je priložena v prilogi).

4.1. Primerjava tehnologij

V tem poglavju bomo navedli nekatere pomembnejše razlike, ki ločujejo omenjeni tehnologiji. Tehnologiji smo primerjali na podlagi zapletenosti sintakse, omejitev ter težavnosti pri razvoju in zmožnosti orodij, ki se uporabljajo za delo z njima. Pomemben dejavnik je bila tudi urejenost ter preglednost kode.

ASP nima mehanizma za poganjanje na ne-Microsoftih tehnoloških platformah, kot so recimo spletni strežnik Apache. Po drugi strani pa je ASP.NET možno zagnati tudi na omenjenih platformah.

Pisanje skript v ASP-ju je možno samo v jezikih VBScript in JavaScript, medtem ko za ASP.NET ni takšne omejitve. Z ASP.NET lahko uporabljamo vsak skladni jezik .NET, ki vključuje C# ter VB.NET (oba jezika sta usmerjena za delo na strežniku).

Primer prve vrstice v datoteki `.aspx`:

```
<%@Page Language="C#"AutoEventWireup="true"
   CodeFile="Default.aspx.cs" Inherits="_Default"
   MasterPageFile="~/header.master" %>
```

Prva vrstica nam pove, da uporabljamo jezik C#, imamo vklopljene dogodke, da se koda, ki se izvaja v ozadju, nahaja v razredu (datoteki) `Default.aspx.cs` ter da stran deduje od strani `_Default`. Navedena je tudi glavna stran (`master page`), kar pove, da je to vsebovana stran (`content page`).

Primer enostavne ASP skripte:

```
<html><body><%Response.Write('Hello World') %></body> </html>
```

Skriptni blok ASP-ja se začne z oznako `<%` in konča z `%>`. Na zgornjem primeru ASP skripta pošlje brskalniku besedilo »Hello World«, katerega brskalnik izpiše.

ASP.NET nam z dogodki (ang. events) prihrani veliko dela, ki bi ga porabili s pisanjem JavaScript kode. Slabost pisanja kode v tehnologiji ASP.NET pa je, da se vsa programska koda izvaja na strežniku in tako tudi ko ne potrebujemo podatkov iz podatkovne baze po nepotrebnem obremenjujemo strežnik.

Ker ASP ni dogodkovno usmerjen, je vsak dogodek potrebno »uloviti« ter ustrezno ukrepati s pomočjo skripte JavaScript.

4.1.1. Shranjevanje lastnosti kontrol

ASP.NET ob vsakem dogodku `POST` shrani lastnosti kontrol v skrito polje `_VIEWSTATE`. S tem omogoči ob ponovni osvežitvi strani (ang. postback event) osvežitev samo želenih kontrol in vrednosti, izris ostalih kontrol z vrednostmi pa ostane enak. To lastnost lahko vklopimo ali izklopimo, tako na nivoju celotne strani, kot za vsako kontrolo ločeno. ASP podobne funkcionalnosti nima, zato jo mora programer, če jo potrebuje, razviti sam,.

4.1.2. Kontrole

ASP razvojna orodja ne vsebujejo kontrol. Za izdelavo osnovne forme uredniškega vmesnika v različici ASP nismo uporabili nobene kontrole, s katero lahko prikažemo več spletnih strani v oknu brskalnika. Uporabili smo le ukaz `INCLUDE`, s katerim vnesemo vsebino druge strani (glava, noga, funkcije, itd.) na prikazano stran, preden se stran prebere na strežniku.

V tehnologiji ASP.NET smo osnovno formo izdelali s pomočjo glavne spletne strani (ang. Master Page).

Glavna spletna stran je v osnovi kontrola, ki omogoča, da na enem mestu definiramo izgled in obnašanje spletne strani, skupno večim spletnim stranem. Prednosti uporabe glavne spletne strani

pridejo do izraza šele z njeno uporabo skupaj z vsebovanimi stranmi (ang. content pages). Na prvi pogled se uporaba glavne spletne strani zdi enaka uporabi tehnologije HTML frameset. Pri tem pa obstaja bistvena razlika: ob osvežitvi katerekoli vsebovane strani se še enkrat naloži tudi celotna spletna stran t.j. glavna spletna stran s podstranmi.

Vsakokratno osveževanje glavne spletne strani je njena slabost. Uporaba glavne spletne strani je zato primerno le za pretežno statične strani, kjer se ob spremembi podstrani glavna spletna stran ne spreminja.

4.1.3. Namestitev

Z namestitvijo tehnologij ASP ter ASP.NET in pripadajočih orodij ni bilo težav.

4.1.4. Sintaksa

Sintaksa je pri ASP.NET-u lažja za pisanje, kar prispeva k hitrejšemu razvijanju. Težava pri ASP-ju je v slabi preglednosti kode, ker koda ni ločena od preostale HTML kode, ter v neuporabnosti oklepajev za zaključevanje funkcij, zank, itd. Prednost in hkrati slabost ASP-ja je v nepotrebnosti definiranja tipa pri deklaraciji spremenljivke. Razvoj je po eni strani hitrejši, pri obširnejših funkcijah pa lahko po drugi strani pride do problemov, ker ni znano, kaj spremenljivka hrani.

Deklaracijo spremenljivke brez tipa omogoča tudi tehnologija .NET. V tako spremenljivko lahko shranimo kakršenkoli tip ali objekt, vendar pa tovrstna spremenljivka generira vedno tipizirano referenco (ang. strongly typed reference). Ob prevajanju kode se tip spremenljivke določi glede na prvo inicializacijo spremenljivke.

Poglejmo si nekaj primerov sintakse:

ASP.NET deklaracija spremenljivk:

```
int stevilo = 1;
```

ASP deklaracija spremenljivk:

```
dim stevilo = 1;
```

ASP.NET klicanje tvorbe objekta ter klic metode objetka

```
objekt ImeObjekta=new objekt();
ImeObjekta.metoda(parametri);
```

ASP klicanje tvorbe objekta ter klic metode objekta

```
set spremenljivka=Server.CreateObject(”objekt”);
spremenljivka.metode parametri
```

4.1.5. Prevajanje (ang. compilation)

ASP

Pri ASP-ju prevajanja kode ni, ker se HTML in vsebovana skripta ASP interpretira, kadar je spletna stran zahtevana. Strežnik naloži kodo v spomin ter skuša poiskati način, kako kodo pognati.

ASP.NET

Pri zahtevi strani `.aspx` se ukaz pošlje v obdelavo ASP.NET-u (procesiranje). Ob prvem zahtevku strani ASP.NET prevede stran v jezik MSIL (ang. Microsoft Intermediate Language). CLR (ang. Common language runtime) kodo predela v strojno kodo in zahteva je pognana z uporabo prevedene kode. Naslednje zahteve so izvedene iz iste strojne kode s predpostavko, da stran ni bila spremenjena.

4.1.6. Razhroščevanje

V ASP je razhroščevanje (ang. debugging) težka naloga, saj nima lastnega razhroščevalnika, ki kot pri ASP.NET v času prevajanja kode (ang. Compiling) odkrije ter javi večino napak. Pri ASP-ju odkrivamo napake iz skopega števila sporočil, ko je aplikacija že v delovanju.

Razvojni orodji Visual Studio in Visual Web Developer Express imata vgrajeno orodje za razhroščevanje. Zaradi predhodnega programiranja v Visual Studiu nam je uporaba teh razvojnih orodij enostavna in intuitivna.

Čas, namenjen izbiri ustreznega razvojnega okolja, namestitvi in učenju uporabe, sem (zaradi obvladljive obsežnosti aplikacije) namenil razhroščevanju na klasičen način z uporabo komentarjev v kodi. Na ta način sem izpisoval vrednosti spremenljivk in položaj v skripti. Težave so mi povzročale sintaktične napake, ki jih je bilo včasih težko odkriti, saj se je pri prikazovanju ASP strani vedno prikazalo isto sporočilo.

4.1.7. Avtomatsko generiranje kode

Za razvoj aplikacije v ASP-ju smo uporabili orodje Visual Studio, ki pomaga pri kodiranju z avtomatskim dokončavanjem klicu funkcije. Podobno pomoč ponujata tudi Notepad++ in Visual Web Developer Expess, kjer nam Microsoftovo orodje za avtomatsko dokončanje (ang. Intellisense) ponuja precej obširnejše možnosti.

Pri pisanju kode v ASP.NET-u se avtomatsko generira veliko kode. To je lahko prednost, ker pohitri programiranje, lahko pa tudi slabost, ker se tako generirajo nepotrebni deli kode in je koda bolj zahtevna.

4.2. Izvedbe in nekatere razlike v razvoju

Obe različici, ASP in ASP.NET, shranjujeta in berete podatke iz podatkovne baze z enakim podatkovnim modelom. Vendar smo v različici ASP.NET uporabili posebni razred `DBInteract` (koda je zaradi večje obsežnosti priložena v prilogi E), namenjen le za komunikacijo s podatkovno bazo, medtem ko je v različici ASP celotna komunikacija prepletena znotraj datotek HTML brez uporabe za to ločenih razredov. Sporazumevanje s predstavitvenim modulom obe različici izvajata preko datotek .XML.

Pri dostopu do podatkov bomo prikazali razlike med obema tehnologijama, saj ASP uporablja tehnologijo ADO, ASP.NET pa tehnologijo ADO.NET.

1. Razlaga kode ASP različice dostopa do podatkov (priloga I):

- `dim ConnectionStringA="Provider=SQLOLEDB.1;SERVER=AXA\SQLEXPRESS; DATABASE=PrvaLiga;"RuntimeUserName ="sp";RuntimePassword = "1234"`

  Način je enakovreden povezavi `ConnectionString` iz ASP.NET-a, razloženi v naslednji točki pri različici ASP.NET. `ConnectionString` predstavlja niz, v katerem so podatki, potrezni za povezavo do podatkovne baze (krajše povezovalni niz).

- `set DBConn = Server.CreateObject("ADODB.Connection")`
  `DBConn.Open ConnectionStringA, RuntimeUserName, RuntimePassword`

  Funkcija ustvari objekt tipa `ADODB.Connection`, potreben za vzpostavitev povezave s podatkovno bazo. Z metodo `Open()` odpre povezavo s pomočjo prej navedenih podatkov v `ConnectionStringA`.

- `Set RS = DBConn.execute("tekma_write@id_tekme="&SQLNumber(id_tekme)& ",@ura=" & SQLString(Request("tip")))`

  Metoda `execute()` je podobna ADO.NET metodi `ExecuteNonQuery()`, vendar pri tej metodi navedemo parametre kar znotraj klica metode same.

2. Razlaga kode ASP.NET različice dostopa do podatkov (priloga E):

- `SqlConnection(ConfigurationManager.ConnectionStrings["DBConnect"].ConnectionString);)`

  Za vzpostavitev povezave smo uporabili razred `SqlConnection`. Razred vsebuje vse potrebne metode za povezavo do podatkovne baze. V tem primeru smo podatke o povezavi prebrali iz datoteke `Web.Config` (kot pove ime samo, je to konfiguracijska datoteka spletne strani), kjer so shranjeni različni podatki, ki so stalno v uporabi.

- `SqlCommand comm = new SqlCommand("procN", conn);`
  `comm.CommandType = CommandType.StoredProcedure;`
  `comm.Parameters.Add(new SqlParameter("@ime_igralca", _ime));)`

  Razred `SqlCommand` predstavlja poizvedbo, ki jo bomo izvedli na podatkovni bazi. Ukaz je tipa shranjena procedura ali celotna SQL poizvedba. Metodo `Parameters.Add()` uporabimo pri shranjenih procedurah, ki zahtevajo določene lastne parametre za uspešno izvršitev transakcije.

- `conn.Open();comm.ExecuteNonQuery();conn.Close();`

  Metoda `Open()` odpre povezavo do podatkovne baze s pomočjo nastavitev iz povezovalnega niza. Metoda `ExecuteNonQuery()` izvede T-SQL ukaz na podatkovni bazi ter vrne število vrstic, na katere je ukaz vplival.

Obe različici aplikacije vračata podatke predstavitvenemu modulu na podoben način. Različica ASP uporablja samo datoteko `default.asp`, v kateri je napisana vsa koda, medtem ko ima različica ASP.NET kodo strukturirano razdeljeno na več datotek. Delovanje je podobno, saj se v obeh različicah najprej preberejo poizvedbe (ang. QueryStrings) iz HTML povezave, nato pa se v skladu z njimi pridobi ustrezne podatke ter se jih vstavi v obliko XML (primer strukture: priloga D).

V nasprotju s tem poteka vnašanje ter spreminjanje tekem pri ASP.NET precej drugače kot pri ASP-ju, saj ASP.NET uporablja vgrajene kontrole, s katerimi je možno nadzorovati vnos (s pomočjo kontrole `RequiredFieldValidator)` ter uporabljati dogodke, vezane na njih.

3. Razlaga kode ASP različice vnosa nove tekme (priloga J):

- `addgame_Check()`

  Celotna kontrola vnosa poteka s pomočjo funkcij JavaScript. Poleg večje zamudnosti pri razvoju se preverjanje podatkov izvaja na osebnem računalniku odjemalca in ne na strežniku kot pri ASP.NET, kar je seveda slabše. Za pošiljanje podatkov uporabimo formo z metodo GET ter gumb tipa `Submit`.

- Po kliku na gumb `dodaj` s pomočjo kode JavaScript in HTML povezave (ponovno naložimo stran, vendar njeni povezavi dodamo parameter `akcija=add_tekma)` pričnemo postopek vnašanja podatkov. Podatke iz forme prenesemo s pomočjo metode GET.
- Popravljanje podatkov poteka znova s pomočjo HTML povezave (ponovno naložimo stran, vendar njeni povezavi dodamo parameter `akcija=edit_tekma` ter `id_tekme_edit=1`).

4. Razlaga kode ASP.NET različice vnosa nove tekme (priloga K):

- `btnaddgame_Click(object sender, EventArgs e)`

  Dogodek se proži po vnosu potrebnih podatkov ob kliku na gumb `dodaj`. Najprej se preveri, če so vneseni vsi obvezni podatki, nato pa se preberejo preostali podatki. Na koncu se vse vpiše v podatkovno bazo. V nasprotnem primeru se nam prikaže opozorilo, da nismo izpolnili obveznih polj.

- `Page.Validate("dodajigro");`

  Funkcija preverja določeno izpolnjenost za skupino kontrol (v tem primeru kontrole, ki pripadajo skupini `dodajigro`). Če katera od kontrol ni izpolnjena, nam vrne negativen odgovor ter s tem onemogoči pošiljanje nepopolnih podatkov na strežnik.

- `FillData()`

  Funkcija se izvede, ko želimo spremeniti podatke izbrane tekme. Vse kontrole napolni s podatki ter postavi parameter `Session["edit_game"]` na 1. S tem spremeni namembnost dogodka ob dodajanju, saj sedaj dogodek ob proženju uporabi funkcijo `Zivo_Tekme_Edit()` namesto funkcije `Zivo_Tekme_Add()`.

Na novo dodana tekma je po uspešnem vnosu vidna v uredniškem ter predstavitvenem modulu.

4.3. Ugotovitve

Zadnji del poglavja je namenjen predstavitvi ugotovitev. Ocenili bomo orodja ter podali primerjavo obeh tehnologij, ki smo jih uporabili pri izdelavi.

4.3.1. Ocena uporabljenih orodij

Večina uporabljenih orodij, ki so nam precej olajšala razvoj aplikacije, se ni pretirano razlikovala med izvedbo ASP.NET in izvedbo ASP.

V obeh primerih smo uporabili:

- za podatkovno bazo in upravljanje s podatki: Microsoft SQL Server 2005,
- za grafiko: PhotoShop Cs2,
- spletni strežnik: IIS.

Za obe različici smo kot razvojno okolje uporabljali Visual Studio 2005. Uporabljeno orodje se je izkazalo za pravilno in učinkovito izbiro.

4.3.2. Prednosti in slabosti tehnologij

Obe tehnologiji lahko uporabimo za razvoj obširnejših spletnih aplikacij. Če se odločamo med eno in drugo tehnologijo, je dobro poznati namen in obseg uporabe ter določiti infrastrukturo aplikacije.

Prednosti tehnologije ASP:

- brezplačna uporaba,
- veliko literature in dokumentacije ter pomoči na internetu,
- hitro popravljanje kode, ker je vse znotraj ene datoteke,
- preprosto razvojno okolje,
- porabi manj strežniških virov.

Slabosti tehnologije ASP:

- skromna podpora OOP,
- podpora le dvem skriptnim jezikom (VBScript, JavaScript),
- nezmožnost poganjanja na ne-Microsoftih platformah,

- zamudno odkrivanje napak v kodi,
- slaba podpora avtomatskega generiranja kode.

Prednosti tehnologije ASP.NET:

- brezplačna uporaba,
- veliko literature in dokumentacije ter pomoči na spletu,
- dobro razvojno okolje s podporo večih programskih jezikov,
- enostavno razhroščevanje in dober opis napak,
- dober sistem za avtomatsko dokončanje kode,
- dobra podpora za objektno usmerjeno programiranje,
- shranjevanje lastnosti kontrol in
- bogata paleta kontrol.

Slabosti tehnologije ASP.NET:

- deluje samo na Microsoftovih operacijskih sistemih,
- dogodkovni razvoj je za začetnike težje razumljiv,
- podpirajo ga le Microsoftovi in Apache strežniki.

Po zaključenih primerjavah in preizkušnjah se je ASP.NET izkazal kot boljši pristop k razvoju dinamičnih spletnih strani. Kljub temu ne smemo delati krivice ASP-ju. ASP ni pravo razvojno okolje kot ASP.NET je zgolj skriptni jezik. Poleg tega je ASP.NET sodobnejša nadgradnja ASP-ja, kar botruje k njegovi boljši razvitosti od predhodnika.

5. ZAKLJUČEK

Cilj diplomske naloge je bila, poleg primerjave obeh programskih jezikov oziroma tehnologij, predstavitev razvitega sistema. Ta po eni strani omogoča uredniku vnos, obdelavo in pregled podatkov, po drugi strani pa je sistem namenjen odjemalcu za spremljanje nogometnih dogodkov preko spleta. Predstavili smo posamezne dele sistema, opisali uporabljene tehnologije in pokazali, kam sodijo v arhitekturi sistema.

Obe uporabljeni tehnologiji sta bili kos nalogi, vendar je bil razvoj v tehnologiji ASP.NET bistveno lažji, predvsem zaradi enostavnega razhroščevanja, dobrih opisov napak ter obširnejšega nabora uporabnih kontrol. Ogrodje Microsoft .NET je nudilo boljšo rešitev za vsako večjo težavo, na katero smo naleteli. Tudi starejša tehnologija ASP se je pokazala za dobro prvo izbiro, saj je bila kljub njeni tehnološki zastarelosti uspešna pri razvoju uredniškega modula, ki deluje brez kakršnihkoli težav.

Med razvojem aplikacije smo ohranjali stalno komunikacijo z naročnikom, kar se je izkazalo za uspešno potezo. V aplikacijo smo vključili tudi tiste funkcionalnosti, ki so bile pri prvotnemu zajemu zahtev precej nerazčiščene ter ne najbolj premišljene.

Razvoj aplikacije (z uredniškim modulom v klasičnem ASP-ju) je trajal približno 2 meseca. Temu je botrovalo predvsem učenje in prilagajanje novi tehnologiji Adobe Flex, ki smo jo uporabili za razvoj predstavitvenega modula. Izbira omenjene tehnologije se je izkazala za odlično, saj so že v fazi testiranja uporabniki pokazali veliko zanimanja. Ugotovili so, da je modul preprost za uporabo ter podaja vse ustrezne podatke na pregleden in vizualno privlačen način.

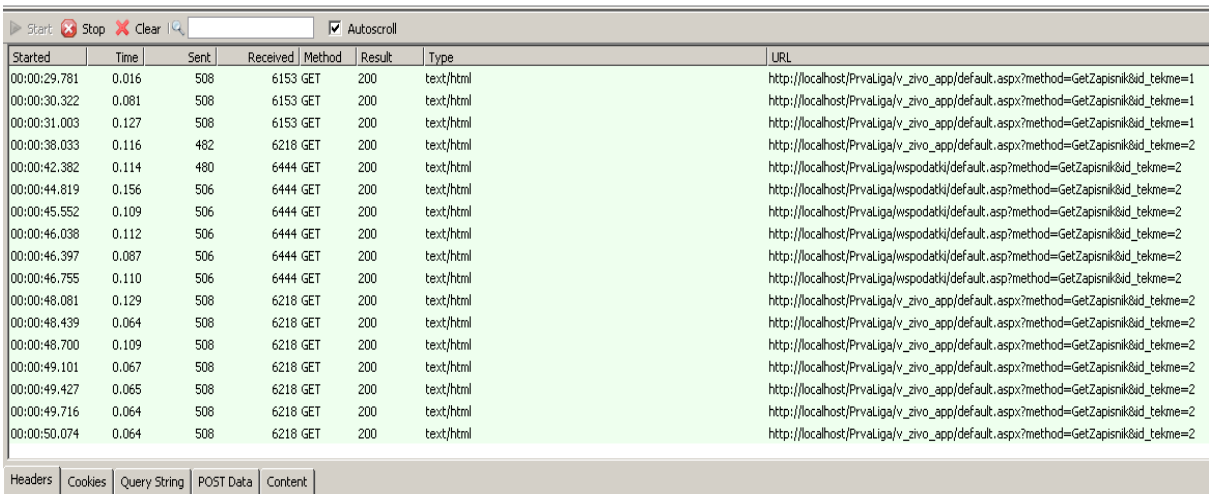
5.1. Analiza rešitve

Z naročnikom smo si bili enotni, da aplikacija izpolnjuje zahteve v celoti in jih celo preseže, saj smo med razvojem dodali nekaj lastnih izboljšav. Za te smo bili mnenja, da bodo uporabnikom nudile boljšo in celovitejšo izkušnjo.

Zmogljivosti uredniškega modula so odvisne predvsem od hitrosti strežnika, vendar je zaradi manjše količine podatkov analiza hitrosti relativno nepomembna, ker ne prihaja do velikih hitrostnih odstopanj pri osveževanju le-teh. Naj vseeno omenimo, da aplikacija (uredniški ter

predstavitveni del) dosega več kot zadovoljive hitrostne postavke, saj se podatki osvežijo praktično v nekaj stotinkah.

Ker je uredniški modul razvit v dveh tehnologijah, smo naredili manjši primerjalni preizkus med tehnologijama s pomočjo orodja HTTPAnalyzer [16]. Z njim lahko izmerimo čas odziva (slika 28), ko stran zahteva podatke iz podatkovne baze. Preizkus smo izvedli na lokalnem razvojnem strežniku.



Slika 27 : HTTP Analyzer

Pokazalo se je, da je bil kljub majhni količini podatkov preizkusa ASP.NET modul hitrejši. Modul v ASP-ju je dosegel povprečje 0.105 sekunde za osveževanje podatkov, medtem ko je modul v ASP.NET-u porabil le 0.064 sekunde. Razlika je minimalna, vendar smo mnenja, da bi se povečevala z naraščanjem količine podatkov.

Oba modula sta bila razvita s poudarkom na praktičnosti in intuitivnosti vmesnika. Ker so njune naloge preproste narave, sta bila modula razvita v skladu z njimi. Oba vmesnika bi težko razvili za še lažjo uporabo, edina izboljšava bi po našem mnenju bila uporaba pristopa »primi in spusti« k dodajanju komentarjev.

Gledano s funkcionalnega stališča je primerjava s podobnimi sistemi težka, saj je trenutno veljavna praksa, da se tekme spremlja v živo preko video prenosa in ne preko statične spletne aplikacije. Uredniški modul bi lahko postavili ob bok vrsti CMS sistemov, vendar je naš modul usmerjen preveč ozko za ustrezno primerjavo.

Ostalo je še nekaj odprtih možnosti za izboljšanje sistema. Večjo pozornost bi namenili predvsem predstavitvenemu modulu, saj smo mnenja, da je zaradi pomanjkanja video prenosa izkušnja bolj suhoparna. Pri spremljanju nogometnih tekem igra veliko vlogo vzdušje. Samo prikazovanje podatkov brez vpogleda dogajanja v video ali slikovni obliki ni najbolj zanimivo, zato bi morali dodati video prenos ali vsaj slikovno gradivo s tekme, da bi ustvarili boljše vzdušje. To bi verjetno doprineslo veliko k temu, da bi našo storitev uporabljalo večje število odjemalcev.

Sistem trenutno deluje že nekaj časa. V tem času smo zabeležili, da je predstavitveni modul v času tekme uporabljajo največ 203 uporabnikov naenkrat. Glede na to, da sistem ne uporablja video prenosa, nas je rezultat pozitivno presenetil. Ugotovitev je bila glavni razlog za načrtovanje določene razširitve vmesnika, ki bodo odjemalcu ponudile še boljši in bogatejši prikaz. Uredniško aplikacijo so v tem času komentatorji uspešno uporabljali brez potrebe po večjih prilagoditvah ali popravkih.

Za konec naj dodam, da se je v času razvoja sistema začela širiti nova Microsoftova tehnologija: ASP.NET MVC. Tehnologija uporablja arhitekturo Model–Pogled–Kontroler (ang. Model–View–Controller), ki loči domensko logiko od uporabniškega vmesnika in hkrati dovoljuje ločen razvoj ter preizkušanje obeh [15]. S to tehnologijo je razvoj dinamičnih spletnih strani storil še korak naprej, kar bomo zagotovo izkoriščali pri razvijanju spletnih aplikacij naslednje generacije.

PRILOGE

A) Efekti na kontrolo:

```
<mx:Fade id="fadeOut" target="banner2" duration="500" alphaFrom="1.0" alphaTo="0.0"/>
<mx:Fade id="fadeIn" target="banner2" duration="500" alphaFrom="0.0" alphaTo="1.0"/>
<mx:Resize id="expandBanner" target="{ banner2}" widthTo="231" heightTo="318"/>


<mx:Canvas x="270" y="26" hideEffect="{fadeOut}" showEffect="{fadeIn}"
            width="490" height="120" backgroundColor="#B2C7D6" id="banner2"
            backgroundAlpha="0.0" borderColor="#ADC3D3">
</mx:Canvas>
```

B) Ročno proženje efekta:

```
private function moveLestvicaDol():void
      {
            rezultattekme.visible=true;
            moveTabeleCBDol();

            contractLestvica.end();
            contractLestvica.play();
      }
```

C) Primer pridobivanja podatkov strelcih sezone:

```
public var LINKSERVER:String;
   public var StrelciXML:XML = new XML();
   public var XML_Strelci_URL:String;
   public var StrelciURL:URLRequest;
   public var StrelciLoader:URLLoader;


   public function preinitializeApp():void
       {
         LINKSERVER = Application.application.parameters.link;

         if (LINKSERVER == null)
            LINKSERVER = "http://localhost/PrvaLiga/v_zivo_app/Default.aspx";
         XML_Strelci_URL = new String(LINKSERVER+"?method=GetStrelci");
         StrelciURL = new URLRequest(XML_Strelci_URL);
         StrelciLoader = new URLLoader();

     }
```

Č) Klicanje objektov Adobe Flex (OOP):

```
package
```

```
{
  public class Postave
  {
         public var StGolDomaci:int=0;
         public var StGolGosti:int=0;

         public function Postave()
         {
    public function GetMenjaveKartoniZaIgralcaDomaci(id_igralca:String,Zapisnik:XML):String
    {
          var RR:String=new String();
            for each(var node:XML in Zapisnik.Menjave.Domaci.Menjava)
            {
              if(node.id_igralec_out==id_igralca)
              {
                 RR="v ("+node.minuta+").minuti:"+node.igralec_in;
              }
            } return RR;
     }

}
```

```
<mx:Script>
  <![CDATA[
     import Postave;//uvoz razred

    var FunkcijePostave:Postave=new Postave();//deklariranje objekta

    PostaveDomaciXML=new XML(FunkcijePostave.PostaveDomaci(Zapisnik));//klic metode

</mx:Script>
```

D) XML struktura za lestvico:

```
<Lestvica>
     <Klub>
      <mesto>1</mesto>
       <id_kluba>2</id_kluba>
       <klub>HIT Gorica</klub>
       <grb>http://www.prvaliga.si/klubi/grbi/nkgoPrva.gif</grb>
       <st_tekem>2</st_tekem>
       <tocke>3</tocke>
  </Klub>

     <Klub>
     <mesto>2</mesto>
     <id_kluba>3</id_kluba>
     <klub>Labod Drava</klub>
     <grb>http://www.prvaliga.si/klubi/grbi/nkdrPrva.gif</grb>
     <st_tekem>1</st_tekem>
     <tocke>1</tocke>
    </Klub>
</Lestvica>
```

E) Razred DBInteract:

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Collections.Generic;
using System.Data.SqlClient;

public class DBInteract
{
  public Dictionary<string, object> parameters;
  public SqlConnection conn;

  protected DataSet dset;

  #region Constructor

  public DBInteract()
  {
    parameters = new Dictionary<string, object>();
    conn = new
SqlConnection(ConfigurationManager.ConnectionStrings["DBConnect"].ConnectionString);

  }

  public DBInteract(string connC)
  {
    parameters = new Dictionary<string, object>();
    conn = new SqlConnection(ConfigurationManager.ConnectionStrings[connC].ConnectionString);
  }
  #endregion

  #region Methods

  public DataSet DBInteractOneparameter(string paramName, object paramValue, string procName)
  {
    parameters.Add(paramName, paramValue);

    DataSet dset = ExecuteDBinteractStoredProcedure(procName);

    return dset;

  }
  public DataSet ExecuteDBinteractStoredProcedure(string procN)
  {

    #region SQL

    SqlCommand comm = new SqlCommand(procN, conn);
    comm.CommandType = CommandType.StoredProcedure;

    if (parameters.Count > 0)
    {
      string parameterName = String.Empty;

      foreach (string paramName in parameters.Keys)
      {
        parameterName = paramName;

        if (string.Compare(parameterName.Substring(0, 1), "@") != 0)
        {
          parameterName = parameterName.Insert(0, "@");
        }
        comm.Parameters.Add(new SqlParameter(parameterName, parameters[paramName]));
      }
    }

    SqlDataAdapter dokument = new SqlDataAdapter(comm);
```

```
  dset = new DataSet();

  try
  {
    conn.Open();
    dokument.Fill(dset);
    conn.Close();

    parameters.Clear();
    dokument.Dispose();
  }
  catch(Exception x)
  {
    dset = null;
  }

  #endregion

  return dset;

}
public void ExecuteDBinteractNoDataSet(string procN)
{
  SqlCommand comm2 = new SqlCommand(procN, conn);
  comm2.CommandType = CommandType.StoredProcedure;

  if (parameters.Count > 0)
  {
    string parameterName = String.Empty;

    foreach (string paramName in parameters.Keys)
    {
      parameterName = paramName;

      if (string.Compare(parameterName.Substring(0, 1), "@") != 0)
      {
        parameterName = parameterName.Insert(0, "@");
      }
      comm2.Parameters.Add(new SqlParameter(parameterName, parameters[paramName]));
    }
  }

  try
  {
    conn.Close();
    conn.Open();
    comm2.ExecuteNonQuery();
    conn.Close();
    comm2.Dispose();
    parameters.Clear();


  }
  catch(Exception e)
  {
    string exception = e.ToString();
  }

}
public void ExecuteDBinteract(string Sqlcommand)
{

  SqlCommand comm = new SqlCommand(Sqlcommand, conn);

  try
  {
    conn.Open();
    comm.ExecuteNonQuery();
    conn.Close();
    comm.Dispose();
```

```
    }
    catch
    {

    }

  }

  #endregion
}
```

F) Polnjene podatkov za predstavitveni modul

```
//////////////////PODATKI OD TEKEM//////////////////////////
[Bindable]
public var GamesData:XML = new XML();
public var XML_GAMES_URL:String;
public var GamesDataURL:URLRequest;
public var GamesLoader:URLLoader;


/////////LESTVICA/////////////////////////////////////////////
[Bindable]
public var LestvicaXML:XML = new XML();
public var XML_Lestvica_URL:String;
public var LestvicaURL:URLRequest;
public var LestvicaLoader:URLLoader;
//////////////////////////////////////////////////////////////

/////////STRELCI/////////////////////////////////////////////
[Bindable]
public var StrelciXML:XML = new XML();
public var XML_Strelci_URL:String;
public var StrelciURL:URLRequest;
public var StrelciLoader:URLLoader;



//POLNJENJE PODATKOV NA SWF LOAD
public function preinitializeApp():void
{
LINKSERVER = Application.application.parameters.link;
if (LINKSERVER == null)
LINKSERVER = "http://enterprise/prva-liga/wsPodatki/";
//LINKSERVER = "http://www.prvaliga.si/wsPodatki/";


Banner_URL = new String(LINKSERVER+"?method=GetBanners");
BannerURLR = new URLRequest(Banner_URL);
BannerLoader = new URLLoader();

XML_GAMES_URL = LINKSERVER+"?method=GetTekme";
GamesDataURL = new URLRequest(XML_GAMES_URL);
GamesLoader = new URLLoader();

XML_Lestvica_URL = new String(LINKSERVER+"?method=GetLestvica");
LestvicaURL = new URLRequest(XML_Lestvica_URL);
LestvicaLoader = new URLLoader();

XML_Strelci_URL = new String(LINKSERVER+"?method=GetStrelci");
StrelciURL = new URLRequest(XML_Strelci_URL);
StrelciLoader = new URLLoader();
}
```

G) Risanje elementov na časovni trak

```
    private function DrawObjects():void
    {
       var velikost:int=igrisce.getChildren().length;
       var i:int;

      //brisanje dodanih elementov(ne label za ime skupine in minute)
      if(velikost>5)
      {
        for(i=velikost-1;i>=3;i--)
        {
         igrisce.removeChild(igrisce.getChildAt(i));
        }
      }
      //MENJAVE
      for each (var item1:XML in Zapisnik.Menjave.Domaci.Menjava)
       {

AddItemToCanvas(item1.minuta,"menjava",2,item1.igralec_in,item1.igralec_out,"","",item1.minuta,0)
;
       }
      for each (var item2:XML in Zapisnik.Menjave.Gosti.Menjava)
       {

AddItemToCanvas(item2.minuta,"menjava",1,item2.igralec_in,item2.igralec_out,"","",item2.minuta,0)
;
       }
      //KAZNI
      var tipK:String;
      for each (var item:XML in Zapisnik.Kazni.Domaci.Igralec)
       {
        if(item.tip=="1")
        {
         tipK="rumeni karton";
        }
        else if(item.tip=="2")
        {
         tipK="rdeci karton";
        }
        else if(item.tip=="3")
        {
         tipK="rumeno/rdeči karton";
        }
            AddItemToCanvas(item.minuta,tipK,2,item.Igralec,"",item.vzrok,"",item.minuta,0);
       }
      for each (var item3:XML in Zapisnik.Kazni.Gosti.Igralec)
       {
          if(item3.tip=="1")
        {
         tipK="rumeni karton";
        }
        else if(item3.tip=="2")
        {
         tipK="rdeci karton";
        }
        else if(item3.tip=="3")
        {
         tipK="rumeno/rdeči karton";
        }
            AddItemToCanvas(item3.minuta,tipK,1,item3.Igralec,"",item3.vzrok,"",item3.minuta,0);
       }
      //ZADETKI
       for each (var item4:XML in Zapisnik.Zadetki.Domaci.Igralec)
       {
```

```
AddItemToCanvas(item4.minuta,"gol",2,item4.Igralec,"","",item4.rezultat,item4.minuta,item4.avtogo
l);
        }
       for each (var item5:XML in Zapisnik.Zadetki.Gosti.Igralec)
        {

AddItemToCanvas(item5.minuta,"gol",1,item5.Igralec,"","",item5.rezultat,item5.minuta,item5.avtogo
l);
        }
     }

     //CE ZE OBSTAJA ELEMENT NA PODANI POZICIJI V CANVASU
     private function DoesExist(xcoor:int,ycoor:int):Boolean
     {
      var isThere:Boolean=false;

      var x:int=0;

                 for (var i:String in this.igrisce.getChildren())
                 {
                        if(igrisce.getChildAt(x).y==ycoor&&igrisce.getChildAt(x).x==xcoor)
       {
         isThere=true;
       }
       x++;
      }

      return isThere;
      }
     //RISANJE ELEMENTOV V "CANVAS" IGRISCE
     public function
AddItemToCanvas(minuta:int,item:String,ekipa:int,igralec_in:String,igralec_out:String,vzrok:Strin
g,rezultat:String,minutaT:String,avtogol:int):void
      {

      var slika:Image=new Image();
      var pika:Image=new Image();
      var minutaL:Label=new Label();
      pika.source="slike/pika.gif"
      minutaL.text=minuta.toString();

      slika.buttonMode=true;
      slika.mouseChildren=false;
      slika.useHandCursor=true;
      slika.addEventListener(ToolTipEvent.TOOL_TIP_CREATE, function(event:ToolTipEvent):void
      {
       var parm1:String = new String();

       if(item=="gol")
       {
          parm1="slike/zoga.gif";
       }
       else if(item=="rdeci karton")
       {
          parm1="slike/rdecikarton.gif";
       }
       else if(item=="rumeni karton")
       {
          parm1="slike/rumenikarton.gif";
       }
        else if(item=="menjava")
       {
          parm1="slike/menjava.gif";
       }
       var ctt:CustomToolTip=new CustomToolTip();
           ctt.slika=parm1;
       event.toolTip = ctt;
      });

       if(item=="gol")
```

```
{
    slikaP="slike/zoga.gif";
    if(avtogol==1)
    {
    slika.toolTip=minutaT+".minuta(AG):"+igralec_in;
    }
    else
    {
    slika.toolTip=minutaT+".minuta:"+igralec_in;
    }
}
else if(item=="rdeci karton")
{
   slikaP="slike/rdecikarton.gif";
     slika.toolTip=minutaT+".minuta:"+igralec_in;
}
else if(item=="rumeni karton")
{
    slikaP="slike/rumenikarton.gif";
    slika.toolTip=minutaT+".minuta:"+igralec_in;
}
 else if(item=="menjava")
{
   slikaP="slike/menjava.gif";
   slika.toolTip=minutaT+".minuta: ven "+igralec_out+" -noter "+igralec_in;
}

slika.source=slikaP;
slika.width=15;
           slika.height=15;

              if(minuta<90)
                                  {
                                        slika.x=5.45*minuta;
                    }
          else
                    {
                  slika.x=5.40*90;
                    }

   if(ekipa==1)
     {
       if(avtogol==1)
       {
          slika.y=8;
       }
       else
       {
        slika.y=45;
       }

     }
   else if(ekipa==2)
     {
       if(avtogol==1)
       {
         slika.y=45;
       }
       else
       {
        slika.y=8;
       }

     }
           igrisce.addChild(slika);
           pika.x = slika.x+4;
           minutaL.x = slika.x-1;

           if (slika.y==8)
                 {
                   pika.y = 0;
```

```
                    minutaL.y = 23;
                }
                else
                {
                    pika.y = 62;
                    minutaL.y = 33;
                }

            minutaL.setStyle("color", 0xFFFFFF);
            minutaL.setStyle("fontSize", 8);

            igrisce.addChild(pika);
            igrisce.addChild(minutaL);
    }
```

H) Časovno osveževanja

```
public var TimerTekma:Timer = new Timer(2000, 0);  // timer za refresh tekme,
ki ima status zivo

 //////////////TIMER FUNKCIJA KO SE ZAZENE TEKMA////////////
   public function TimerTekmaHandler(event:TimerEvent):void
   // Kliče se, ko je status tekme = ZIVO
   {
        // Ali se je tekma končala? Če se je, ugasni timer!
        var i:int=0;
        for(i=0;i<=this.poljeIDtekem.length-1;i++)
        {
            if(ID_tekme==int(this.poljeIDtekem[i]))
            {
                if(int(this.poljeStatusTekem[i])==3)
                {
                 trace("Tekma se je zaključila!");
                 this.TimerTekma.stop();
                }
            }
        }
     LoadPotek();
     LoadZapisnik(0);
   }

private function LoadZapisnik(ForceRefresh:int):void
            {
                if (ID_tekme != 0 || ForceRefresh==1)
                // Če prikazujemo VSE TEKME, ni potrebno refreshati zapisnika
                {
                    ZapisnikForceRefresh = ForceRefresh;
                    ZapisnikLoader.load(new
                    URLRequest(LINKSERVER+"?method=GetZapisnik&id_tekme="+ID_tekme));
                    ZapisnikLoader.addEventListener(Event.COMPLETE,
                    LoadZapisnikHandler);
                }

      }
```

I) Dostopanje do podatkov ASP

```
<%
dim Portal_Conn_ConnectionString
dim Portal_Conn_RuntimeUserName
dim Portal_Conn_RuntimePassword

Portal_Conn_ConnectionString = "Provider=SQLOLEDB.1;SERVER=ZUMBO\SQLEXPRESS; DATABASE=PrvaLiga;"
Portal_Conn_RuntimeUserName = "sp"
Portal_Conn_RuntimePassword = "1234"
```

```
DIM DBConn
set DBConn = Server.CreateObject("ADODB.Connection")
        DBConn.ConnectionTimeout = 15
        DBConn.CommandTimeout = 30
        DBConn.CursorLocation = 3
        DBConn.Open Portal_Conn_ConnectionString, Portal_Conn_RuntimeUserName,
Portal_Conn_RuntimePassword
%>
```

J) Dodajanje tekme ASP

```
        if Request("akcija")="add_game_new" then
     var dat=GetaData();
                set RS=DBConn.execute("zivo_tekma_write @id_tekme=" & SQLNumber(id_tekme) &
",@ura=" & SQLString(Request("tip")))

                Response.Redirect "default.asp?id_tekme=" & id_tekme
        end if
```

K) Dodajanje tekme ASP.NET

```
void btnaddgame_Click(object sender, EventArgs e)
  {
    Page.Validate("dodajigro");

    if (IsValid)
    {
      int id_domaci = metode.ToInt(this.ddlDomaci.SelectedValue);
      int id_gosti = metode.ToInt(this.ddlGosti.SelectedValue);
      DateTime datum = RetDate(this.txtDatum.Text,this.txtUra.Text);
      string kraj=this.txtKraj.Text;
      string ura = this.txtUra.Text;
      string stadion = this.txtStadion.Text;
      int gledalci = metode.ToInt(this.txtGledalci.Text);
      string s_glavni = this.txtGlavni.Text;
      string s_prvi = this.txtPrvi.Text;
      string s_drugi = this.txtDrugi.Text;
      string s_cetrti = this.txtCetrti.Text;
      string s_delegat = this.txtDelegat.Text;

      if (metode.ToInt(Session["edit_game"]) == 1)
      {
        metode.Zivo_Tekme_Edit(metode.ToInt(Request.QueryString["id_tekme_edit"]), datum, kraj,
ura, stadion, gledalci, s_glavni, s_prvi, s_drugi, s_cetrti, s_delegat);
      }
      else
      {
         metode.Zivo_Tekme_Add(id_domaci, id_gosti, datum, kraj, ura, stadion, gledalci,
s_glavni, s_prvi, s_drugi, s_cetrti, s_delegat);
      }


      Clear();


    }
    else
    {
      string script = "<script language=javascript>alert('prosimo, vnesite vse potrebne
podatke');</script>";
      Page.ClientScript.RegisterStartupScript(Page.GetType(), "nonvalid", script);
    }
    }
```

VIRI

[1] Tom Archer, Andrew Whitechapel: INSIDE C#. Redmond: Microsoft press, 2002

[2] Microsoft.com, »ADO.NET«
http://msdn.microsoft.com/en-us/library/27y4ybxw(VS.71).aspx

[3] Migrating from ASP do ASP.NET
http://msdn.microsoft.com/en-us/library/dddsc60w%28VS.71%29.aspx

[4] Bill Evjen, Scott Hanselman, Farhan Muhammad, Srinivasa Sivakumar, Devin Rader: Professional ASP.NET 2.0. Indianapolis: Wiley Publishing, Inc., 2006. ISBN-13: 978-0-7645-7610-2

[5] ASP.NET
http://en.wikipedia.org/wiki/ASP.NET

[6] David Buser, Chris Ullman, Jon Duckett, John Kauffman, Juan T. Llibre, David Sussman, Brian Francis**:** Beginning Active Server Pages 3.0 **:** Wiley Publishing, Inc., 2006. ISBN:0-7645-4363-6

[7] Matej Govek: Primerjava razvoja sistema za upravljanje spletnih vsebin v tehnologijah ASP.NET in PHP, diplomsko delo, FRI, 2010

[8] Adobe Flex
http://en.wikipedia.org/wiki/Adobe_Flex

[9] PhotoShop
http://en.wikipedia.org/wiki/Adobe_Photoshop

[10] Internet Information Services
http://en.wikipedia.org/wiki/Internet_Information_Services

[11] SQL Server 2005 Express
http://www.microsoft.com/Sqlserver/2005/en/us/express.aspx

[12] Patrick Carey, Frank Canovatchel: New perspective on JavaScript: Course Technology, 2003. ISBN-10: 0-619-26797-6

[13] Informacijski sistemi: mag. Andrej Bregar, univ. dipl. inž., 2008.
http://lisa.uni-mb.si/~bregar/Predavanja/04-zajem_zahtev.pdf

[14] LiveTicker
http://www.fussball-liveticker.eu/

[15] ASP.NET MVC
http://en.wikipedia.org/wiki/ASP.NET_MVC_Framework

[16] HTTPAnalyzer
http://www.ieinspector.com/httpanalyzer/