

POLITECNICO DI MILANO

Facoltà di Ingegneria dell'Informazione

Corso di Laurea Magistrale in Ingegneria Informatica



**Managing futsal competitions and related events
(Sports Manager)**

Relatore: Prof.ssa Franca Garzotto

Tesi di Laurea di:

Marko Brčić

Matricola n. 764920

Anno Accademico 2013-2014

To my family

Contents

1	Introduction.....	1
1.1	Motivation	1
1.2	Outline	2
2	Background.....	3
2.1	Entity Framework	3
2.1.1	Introduction.....	3
2.1.2	Code first	3
2.1.3	Data Annotations or Fluent API.....	4
2.1.4	Migrations	5
2.1.5	Eager, Lazy and Explicit loading.....	6
2.2	Web API	8
2.2.1	Introduction.....	8
2.2.2	Controllers and basic CRUD operations	8
2.2.3	Defining routes	10
2.2.4	Http Clients.....	10
2.3	Windows Store applications	11
2.3.1	Introduction.....	11
2.3.2	Variety of technologies for building apps	11
2.3.3	Developing for both Windows Phone and Windows store application.....	11
2.3.4	Windows Store	12
2.3.5	Publishing to Windows Store	12
2.3.6	Deploying Enterprise applications	12
3	Sports Manager application	13
3.1	Functional requirements	15
3.1.1	Competition administrator role	16
3.1.2	Match commissioner role	17
3.1.3	Match statistician role.....	18

3.1.4	Match reporter role.....	19
3.2	Non-functional requirements.....	20
3.2.1	Ability to work offline.....	20
3.2.2	Syncing local data.....	20
3.2.3	Touch interface.....	20
3.2.4	Security.....	20
4	User experience and interaction design.....	21
4.1	Menu Bar Highlighting.....	21
4.2	Switch competition view.....	22
4.3	Competition view.....	22
4.4	Switch event view.....	23
4.5	Event view.....	23
5	Architecture.....	24
5.1	Database layer.....	25
5.1.1	Competition core tables.....	27
5.1.2	Competition schedule tables.....	28
5.1.3	Match core tables.....	29
5.1.4	Competition and sport rules tables.....	30
5.1.5	Competition users and mailing lists tables.....	31
5.2	Model layer (Entity Framework).....	32
5.3	Controller layer (Web API).....	34
5.4	View layer (Windows 8.1 application).....	35
6	User manual.....	37
6.1	Choose and manage competition.....	37
6.2	Choose and manage competition event.....	38
7	Conclusion.....	39
8	Appendices.....	40
8.1	Appendix A: Versioning system details.....	40
8.2	Appendix B: Development tools used.....	40
8.3	Appendix C: Visual Studio solution projects.....	41

8.4	Appendix D: List of figures	42
8.5	Appendix E: List of tables.....	42
8.6	Appendix F: List of code examples	43
9	Bibliography.....	44
10	Internet resources	46

Abstract

Tablet devices are becoming more and more popular each day. It's because they are thinner and lighter than laptops while on the other hand their power and performance is catching up with the performance of desktop and laptop computers. Including the touch interface, which was introduced few decades ago but nowadays is common, working with tablets on the go has become so natural and has brought the technology even closer to users as an omnipresent part of their everyday lives.

In this work we will present the application called Sports Manager developed for Windows 8.1 tablet devices. It is a robust application, built with latest Microsoft technologies. The application manages futsal competitions and futsal events. It is built with a later intent to support more sports, but for now futsal is chosen as primary sport for prototype version since futsal is author's primary sport passion.

Microsoft and its operating systems (from version 8.0) developed for tablets maybe stood up on the wrong foot when they first showed up (as version 8.0). It is because they brought up a new experience and a completely new way of thinking to their current windows users, while on the other hand being incomplete and unfinished solution, with basic apps missing a lot of common features and mysterious ways of switching from new Start menu and windows store apps to normal desktop mode (on non RT devices).

With the first update (8.1 version) Microsoft went back to the right track. They improved a lot of things and made transition and learning process more smother for the current desktop and laptop users.

Windows store will, no doubt about it, grow each year, presenting more and more quality applications to the users. It is ahead of us to see if they will manage to compete on the current tablet market with strong players like Google's Android and Apple's iOS operating systems designed for touch devices.

Sommario

Dispositivi tablet stanno diventando sempre più popolare ogni giorno. E' perché sono più sottili e leggeri di computer portatili, mentre d'altra parte il loro potere e le prestazioni sta recuperando terreno con le prestazioni dei computer desktop e laptop. Compreso l'interfaccia touch, che è stato introdotto qualche decennio fa, ma al giorno d'oggi è comune, lavorando con compresse in movimento è diventato così naturale e ha portato la tecnologia ancora più vicino agli utenti come una parte onnipresente della loro vita quotidiana.

In questo lavoro presenteremo l' applicazione denominata Direttore Sportivo sviluppato per Windows 8.1 dispositivi tablet. Si tratta di un'applicazione robusta, costruita con le più recenti tecnologie Microsoft. L' applicazione gestisce le competizioni di futsal e di futsal. E 'costruito con un intento più tardi per sostenere più sport, ma per ora futsal viene scelto lo sport principale per la versione del prototipo in quanto il futsal è la passione sportiva principale dell'autore.

Microsoft ei suoi sistemi operativi (dalla versione 8.0) sviluppati per le tavolette forse si alzò in piedi con il piede sbagliato la prima volta che si presentò (nella versione 8.0). E 'perché hanno portato una nuova esperienza e un modo completamente nuovo di pensare ai propri utenti di Windows correnti, mentre d'altra parte essendo la soluzione incompleta e incompiuta, con applicazioni di base manca un sacco di caratteristiche comuni e modi misteriosi del passaggio dal nuovo avvio menu e Windows Store applicazioni in modalità desktop normale (sui dispositivi non RT).

Con il primo aggiornamento (versione 8.1) di Microsoft è tornato in pista giusta. Hanno migliorato un sacco di cose e fatti di transizione e di apprendimento processo più soffocare per gli attuali utenti di desktop e laptop.

Negozi di Windows, non c'è che dire, crescere ogni anno, presentando sempre più applicazioni di qualità per gli utenti. E ' più avanti di noi per vedere se riusciranno a competere sul mercato tablet attuale con giocatori forti come Android di Google e sistemi operativi iOS di Apple ha progettato per i dispositivi touch.

1 Introduction

1.1 Motivation

With an increase of popularity in using tablets among common people there is a great need for developing applications for tablet platforms. Even if the windows operating system still doesn't hold a significant market share, we believe that it will since a lot of users are familiar with windows operating systems in general. Therefore Windows 8.1 was chosen as targeting platform for the application.

The idea for the application came from the author's participation in the European Universities Championships (EUC) in futsal in which he participated as an assistant technical delegate providing help in organizing the competition in general and also keeping the match records during assigned competition matches. Usually these competitions were, like any common competitions, organized with duration of 2 weeks more or less. They were very intense with the matches and dozen matches were played per day organized in different halls.

The goal of the application is to allow users to manage competitions and related events. The functional requirements arise of the centralized service with whom many clients need to interact. These clients need to be robust so that in case of no connection to the internet, they can still collect data from the competition events and synchronize when the connection to the internet becomes available. With these requirements in mind, instead of building the web application, the proper choice was to build client application for a specific platform. If the idea shows to be good, the client application can be extended to more than one platform.

The choice of technologies and frameworks went in favor of Microsoft and its latest technologies, Entity Framework, Web API, and Windows Store applications. The language used is C#.

1.2 Outline

This work is structured as follows.

In section 2, Background, we will start by introducing the technologies used in developing the application. We will also for every mentioned technology show the example of usage in our application. The technologies are, in order of mentioning in our work, Entity Framework, Web API, Windows Store applications.

In section 3, Sports Manager Application, we will provide functional and non-functional requirements for our application Sports Manager. We will start by providing the designed use cases of the main functionalities in the application and afterwards we'll enumerate non-functional requirements.

In section 4, User experience and interaction design, we will talk about layout of our user interface and good practices that we've used.

In section 5, Architecture, we will describe the architecture of the application, and in detail each layer one by one, from lower layers to upper layers of the architecture. Layers of the application are database layer, model layer, controller layer and view layer.

In section 6, User manual, we will describe briefly two main user scenarios in our application.

In section 7, Conclusion, we will put final word on the whole work, directions and ideas for the improvement of the application.

In section 8, Appendices, we will provide additional short information about the versioning system which hosted the application during development process, also we will name development tools used throughout the project, and finally we will provide short list of figures, tables and code samples for better navigation through our work.

In sections 8 and 9, Bibliography and Internet resources, we will provide a list of resources that helped in making our work.

2 Background

2.1 Entity Framework

2.1.1 Introduction

Entity Framework (EF) is an object-relational mapper that enables .NET developers to work with relational data using domain-specific objects. It eliminates the need for most of the data-access code that developers usually need to write. The latest stable version is 6.1 and this version we used in our application.

There are few basic development workflows that can be chosen depending on the current implementation of the database and the entity framework tools we want to use.

Database state	Model	Code
New database	<u>Model First</u> <ul style="list-style-type: none">- Create model in designer- Database created from model- Classes auto-generated from model	<u>Code first</u> <ul style="list-style-type: none">- Define classes and mapping in code- Database created from model- Use Migrations to evolve database
Existing database	<u>Database first</u> <ul style="list-style-type: none">- Reverse engineer model in designer- Classes auto-generated from model	<u>Code first</u> <ul style="list-style-type: none">- Define classes and mapping in code- Reverse engineer tools available

Table 1. Existing development workflows in Entity Framework

2.1.2 Code first

It is the approach where we write the classes in code first. There is also one class which needs to derive the **DbContext** class and define all the sets of our context. After the classes are written, with automatic or custom migrations we can later update or create the database and the according model.

```

public class Blog
{
    public int BlogId { get; set; }
    public string Name { get; set; }

    public virtual List<Post> Posts { get; set; }
}

public class Post
{
    public int PostId { get; set; }
    public string Title { get; set; }
    public string Content { get; set; }

    public int BlogId { get; set; }
    public virtual Blog Blog { get; set; }
}

public class BloggingContext : DbContext
{
    public DbSet<Blog> Blogs { get; set; }
    public DbSet<Post> Posts { get; set; }
}

```

Code Example 1. Code first Entity Framework workflow

2.1.3 Data Annotations or Fluent API

Data annotations or fluent API are used to define the mappings between the classes defined in DbContext and tables in the database. The two approaches are different in a way that data annotations are defined as annotations in the classes of the entities, while on the other hand fluent API mappings are defined in separate classes.

Both choices have their advantages. Using data annotations, we can see directly in the entity class the defined mappings, while on the other hand, greater readability is accomplished if the details of the mappings between entity classes and database tables we move to a completely separate classes, and therefore we keep entity classes more cleaner and readable.

```

public class Passport
{
    [Key]
    public int PassportNumber { get; set; }
    [Key]
    public string IssuingCountry { get; set; }
    public DateTime Issued { get; set; }
    public DateTime Expires { get; set; }
}

```

Code Example 3. Data annotations example

```

// Configure the primary key for the OfficeAssignment
modelBuilder.Entity<OfficeAssignment>()
    .HasKey(t => t.InstructorID);

// Map one-to-zero or one relationship
modelBuilder.Entity<OfficeAssignment>()
    .IsRequired(t => t.Instructor)
    .WithOptional(t => t.OfficeAssignment);

```

Code Example 2. Fluent API example

2.1.4 Migrations

Migrations are database updates and can be done automatically by data context on its loading. It is sometimes dangerous to enable this setting also in the production environment, so manually writing migrations and running them on a willing occasion is the preferable way.

```

namespace MigrationsDemo.Migrations
{
    using System;
    using System.Data.Entity.Migrations;

    public partial class AddBlogUrl : DbMigration
    {
        public override void Up()
        {
            AddColumn("dbo.Blogs", "Url", c => c.String());
        }

        public override void Down()
        {
            DropColumn("dbo.Blogs", "Url");
        }
    }
}

```

Code Example 4. Database migration example

2.1.5 Eager, Lazy and Explicit loading

Eager loading is the process whereby a query for one type of entity also loads related entities as part of the query. Eager loading is achieved by use of the include method.

```
using (var context = new BloggingContext())
{
    // Load all blogs and related posts
    var blogs1 = context.Blogs
        .Include(b => b.Posts)
        .ToList();

    // Load one blogs and its related posts
    var blog1 = context.Blogs
        .Where(b => b.Name == "ADO.NET Blog")
        .Include(b => b.Posts)
        .FirstOrDefault();
}
```

Code Example 5. Eager loading example

Lazy loading is the process whereby an entity or collection of entities is automatically loaded from the database the first time that a property referring to the entity/entities is accessed. When using POCO entity types, lazy loading is achieved by creating instances of derived proxy types and then overriding virtual properties to add the loading hook. For example, when using the Blog entity class defined below, the related Posts will be loaded the first time the Posts navigation property is accessed:

```
public class Blog
{
    public int BlogId { get; set; }
    public string Name { get; set; }
    public string Url { get; set; }
    public string Tags { get; set; }

    public virtual ICollection<Post> Posts { get; set; }
}
```

Code Example 6. Lazy loading example

Even with lazy loading disabled it is still possible to lazily load related entities, but it must be done with an explicit call (Explicit loading). To do so you use the Load method on the related entity's entry. For example:

```
using (var context = new BloggingContext())
{
    var post = context.Posts.Find(2);

    // Load the blog related to a given post
    context.Entry(post).Reference(p => p.Blog).Load();

    // Load the blog related to a given post using a string
    context.Entry(post).Reference("Blog").Load();
}
```

Code Example 7. Explicit loading example

2.2 Web API

2.2.1 Introduction

Restful services have become very popular. Their main popularity is the accessibility from different platforms. It is crucial in today's development to target as many platforms as possible and at the same time reuse the developed components or layers if possible. Restful services are based on HTTP protocol, and use it as a mean of communication between clients and API. A lot of languages have become popular because they were based on the restful API paradigm, one of the most known development technology is ruby on rails.

Web API is Microsoft's answer, for the increased popularity in restful services, developed for .NET platform. It is an extension to MVC paradigm added for ASP.NET technology. It brings restful aspect to the whole idea.

2.2.2 Controllers and basic CRUD operations

Controllers are classes in Web API responsible for managing HTTP requests. Each controller is bound to a URI. Depending on the parameters appended to the URI and the type of HTTP request issued, different methods of controller are called. In the following table we show to relation between CRUD operations and HTTP request type:

CRUD	HTTP request type
Create	POST
Read	GET
Update	PUT
Delete	DELETE

Table 2. Relations between CRUD operations and HTTP request types

In the following table you can see how different URIs can be mapped to different resources of an entity (for example product entity):

Action	HTTP method	Relative URI
Get a list of all products	GET	/api/products
Get a product by ID	GET	/api/products/id
Get a product by category	GET	/api/products?category=category
Create a new product	POST	/api/products
Update a product	PUT	/api/products/id
Delete a product	DELETE	/api/products/id
Get a list of all products	GET	/api/products

Table 3. Example of actions and related URI resources for product entity

How the implementation of products controller looks in code, we will show you in the following example. Take into consideration that repository object is instance of class defined for handling the requests to the data layer. We also excluded the implementation of the Delete method, but from the rest of the example you can get the idea how it would look.

```
public class ProductsController : ApiController
{
    public IEnumerable<Product> GetAllProducts()
    {
        return repository.GetAll();
    }
    public Product GetProduct(int id)
    {
        Product item = repository.Get(id);
        if (item == null)
        {
            throw new HttpResponseException(HttpStatusCode.NotFound);
        }
        return item;
    }
    public IEnumerable<Product> GetProductsByCategory(string category)
    {
        return repository.GetAll().Where(
            p => string.Equals(p.Category, category, StringComparison.OrdinalIgnoreCase));
    }
    public HttpResponseMessage PostProduct(Product item)
    {
        item = repository.Add(item);
        var response = Request.CreateResponse<Product>(HttpStatusCode.Created, item);

        string uri = Url.Link("DefaultApi", new { id = item.Id });
        response.Headers.Location = new Uri(uri);
        return response;
    }
    public void PutProduct(int id, Product product)
    {
        product.Id = id;
        if (!repository.Update(product))
        {
            throw new HttpResponseException(HttpStatusCode.NotFound);
        }
    }
}
```

Code Example 8. Web API controller example

2.2.3 Defining routes

In order to have the proper binding of URIs to controllers, you need to define the routes in `WebApiConfig` static class which you can find in the `App_Start` folder of the project. In there you should define all the routes of your Web API service.

We will show you in the following code sample how the route should look like for our `Products` controller:

```
config.Routes.MapHttpRoute(
    name: "Products",
    routeTemplate: "api/products/{id}",
    defaults: new { controller = "products", id = RouteParameter.Optional }
);
```

Code Example 9. `WebApiConfig` example of defining routes

2.2.4 Http Clients

Since we have a Web API service, now we have to connect to the service from our clients. We can use `HttpClient` class implemented in C# language since we are developing a c# client. In other languages on other platforms we can also assemble an HTTP client for our API service with a great ease, because all languages have pretty strong support for initializing HTTP requests.

Creating a requests to our Web API becomes pretty straightforward. In the following is a sample request from our HTTP client to our Web API service:

```
using (var client = new HttpClient())
{
    client.BaseAddress = new Uri("http://localhost:9000/");
    client.DefaultRequestHeaders.Accept.Clear();
    client.DefaultRequestHeaders.Accept.Add(new
        MediaTypeWithQualityHeaderValue("application/json"));

    // New code:
    HttpResponseMessage response = await client.GetAsync("api/products/1");
    if (response.IsSuccessStatusCode)
    {
        Product product = await response.Content.ReadAsAsync<Product>();
        Console.WriteLine("{0}\t${1}\t{2}", product.Name, product.Price, product.Category);
    }
}
```

Code Example 10. Web API client call example

2.3 Windows Store applications

2.3.1 Introduction

Microsoft's answer for increase in popularity of Application stores is Windows store. Apple joined the store for smartphones and tablets, while Microsoft has separated stores, one for tablet/desktop applications and one for smartphone applications. It is not such a big difference from user's perspective. Probably from the sales point of view joint store is better idea, so we don't know why Microsoft didn't try to make joint store for his applications, including other products as well.

Windows store applications have support in remarkable .NET environment, including support in pretty advanced developing tools offered for years from Microsoft. One step further was made providing the users even wider choice of technologies for building windows store applications.

2.3.2 Variety of technologies for building apps

In standard offerings to Visual Basic, C#, C++, Microsoft had added yet another very popular language nowadays, JavaScript. Let's take a look in the following table which developing technologies are offered to developers:

Programming Language	Layout technology
C#	XAML
C++	XAML
Visual Basic	XAML
JavaScript	HTML

Table 4. Technologies and languages available for developing windows store apps

There are also examples of making runtime components in one of these technologies and using them from another programming language. So as you can see, take whatever language you are most familiar with and make the application using it.

2.3.3 Developing for both Windows Phone and Windows store application

Microsoft also made possible to develop and reuse libraries of code between windows phone and windows store applications. This is a great improvement in development for these platforms. They are called Portable class libraries.

Few days ago they even published that with new Visual Studio 2013 update it will be possible to develop unified experience across Windows Phone 8.1 and Windows 8.1, and that joint template for developing this experience will be offered in development tools.

2.3.4 Windows Store

Windows Store is marketplace for Windows 8.0 and Windows 8.1 applications for desktop and tablet devices. Currently it offers more than 150,000 applications and it's still growing in numbers in Microsoft's pursuit to catch up with prominent stores like Google Play and App Store. If you submit your app to the store, your app will be available to a large population of people situated in more than 200 countries.

2.3.5 Publishing to Windows Store

Before publishing to windows store you can do two things that are rather important:

1. Run your app regularly through the Windows App Certification Kit
2. Reserve the name of the app (reservation lasts for one year)
3. Choose your markets and languages

There is a checklist you need to go through during application's publishing process:

1. Selling details – like price tier, free trial period, release date, choosing markets, choosing category and subcategory
2. Services – like in-app purchases
3. Ratings – age ratings and rating certificates
4. Cryptography
5. Packages
6. Description – in each language that app supports
7. Note to testers – providing info to certification testers

2.3.6 Deploying Enterprise applications

Sometimes you don't want for your app to be available through Windows Store. Reasons can be numerous. Your app connects to some service and you want to make additional configurations or preparations, or contract definitions before installing and enabling your app on client's devices.

With enterprise applications, you also have a greater control of releasing new updates to your clients, and sometimes that is a crucial step when you need to deploy also your web service and you want for your clients to have according version of Windows Store application delivered at the same time. Waiting for store to certificate and publish your app can complicate your plans of delivering the new updates to your clients as soon as possible.

3 Sports Manager application



Figure 1. Sports Manager application logo

The Sports Manager application's greater goal is to be a robust application for managing competitions and competition events for different sports. In scope of this work, it is developed for primary sport, futsal, as a sport on which the whole idea will be tested. After futsal, there are numerous team sports to which the application can be extended like football, basketball, handball, volleyball, rugby, etc.

The idea for the app came to author from repetitive participation in futsal competitions organized as separate European universities championships or as part of European universities games organized by different hosts in coordination with European Universities Sports Association (EUSA). At the beginning, the author went to these events as competitor, but on later occasions also as a match commissioner or assistant technical delegate in futsal which has the responsibilities of to a great extent helping and giving assistance in organizing the detail and events around the whole competitions.

The competitions through which the author participated are:

- 4th European Universities Futsal Championship, Izola (Slovenia), July 23-29, 2007
 - as a competitor from University of Zagreb
- 5th European Universities Futsal Championship, Wroclaw (Poland), July 14-19, 2008
 - as a competitor from University of Zagreb
- 7th European Universities Futsal Championship, Zagreb (Croatia), July 18-25, 2010
 - as a match commissioner from the host's organization side
- 1st European Universities Games, sport futsal, Cordoba (Spain), July 13-23, 2012
 - as assistant technical delegate
- 9th European Universities Futsal Championship, Malaga (Spain), July 21-28, 2013
 - as assistant technical delegate
- 2nd European Universities Games, sport futsal, Rotterdam (Netherlands), July 24 – August 8, 2014
 - as assistant technical delegate

3.1 Functional requirements

In the system four roles will exist in order to distribute work equally and to allow greater separation of concerns. It is better to divide work into more roles because in that case each user with just one role can concentrate more and perform better in the smaller scope of his responsibilities. In the following table we will enumerate the roles and describe each of the roles in our system.

Application Role	Role description
Competition Administrator	The competition administrator is responsible for generally inserting the competition participants and events and has the possibility to print different type of documents or reports for the competition.
Match Commissioner	The match commissioner is responsible for entering all the main information regarding the specific match, match participants, match events, etc. He is also capable of printing the match record.
Match Statistician	The match statistician has the ability to enter additional match events related to additional statistics which can be provided for the match.
Match Reporter	The match reporter has the ability to pull details about a match or competition and report to social media or other stakeholders interested in information about the competition.

Table 5. Application roles and their descriptions

In the next chapters we will graphically display possibilities of each application role through the means of UML use case diagrams. There it will be much easier to distinguish the responsibilities among roles and see the differences in the capabilities of each role. You will also be able to see the separation of concerns in the responsibilities granted.

3.1.1 Competition administrator role

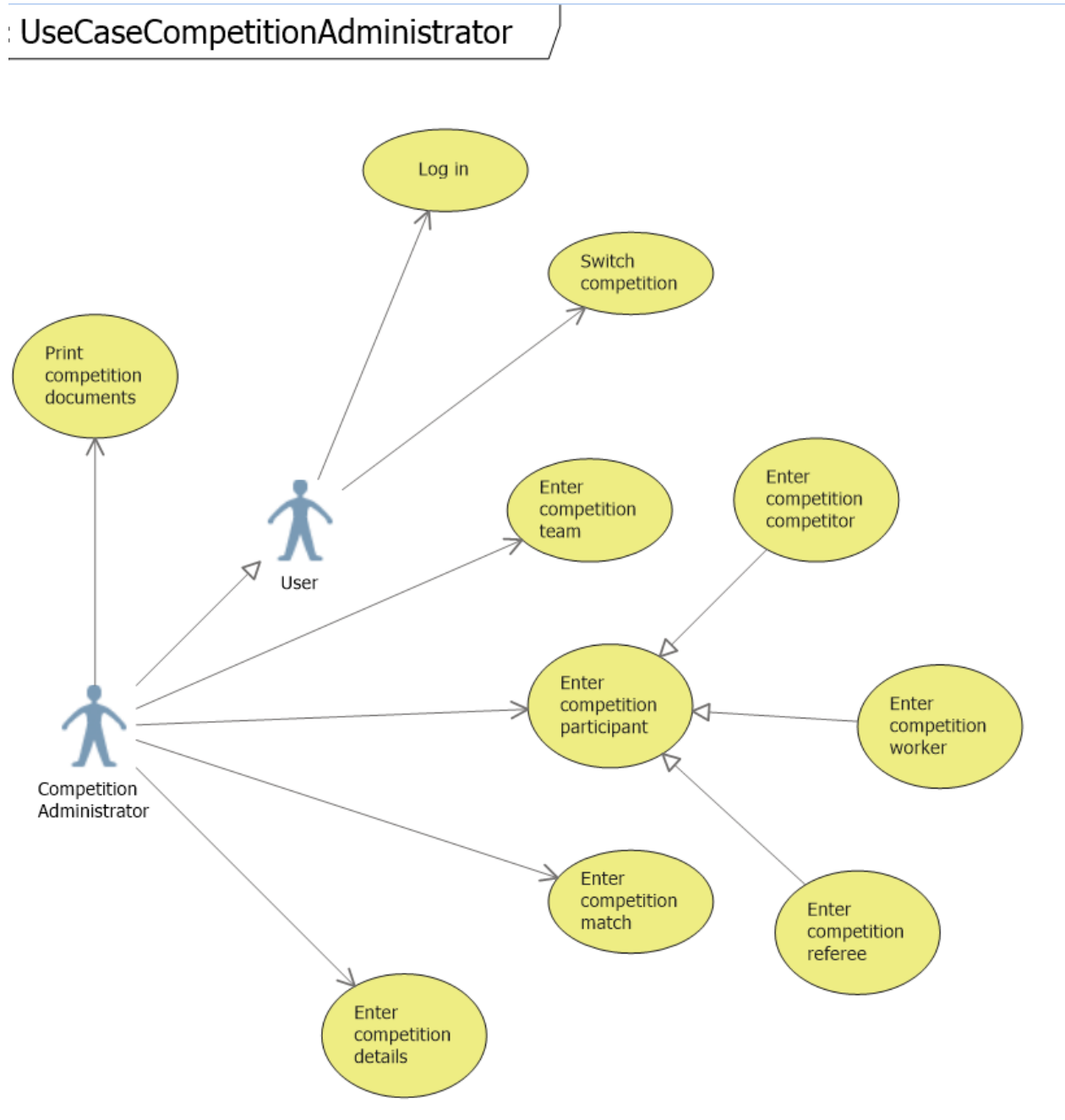


Figure 2. Competition administrator role use case

3.1.2 Match commissioner role

UseCaseMatchCommissioner

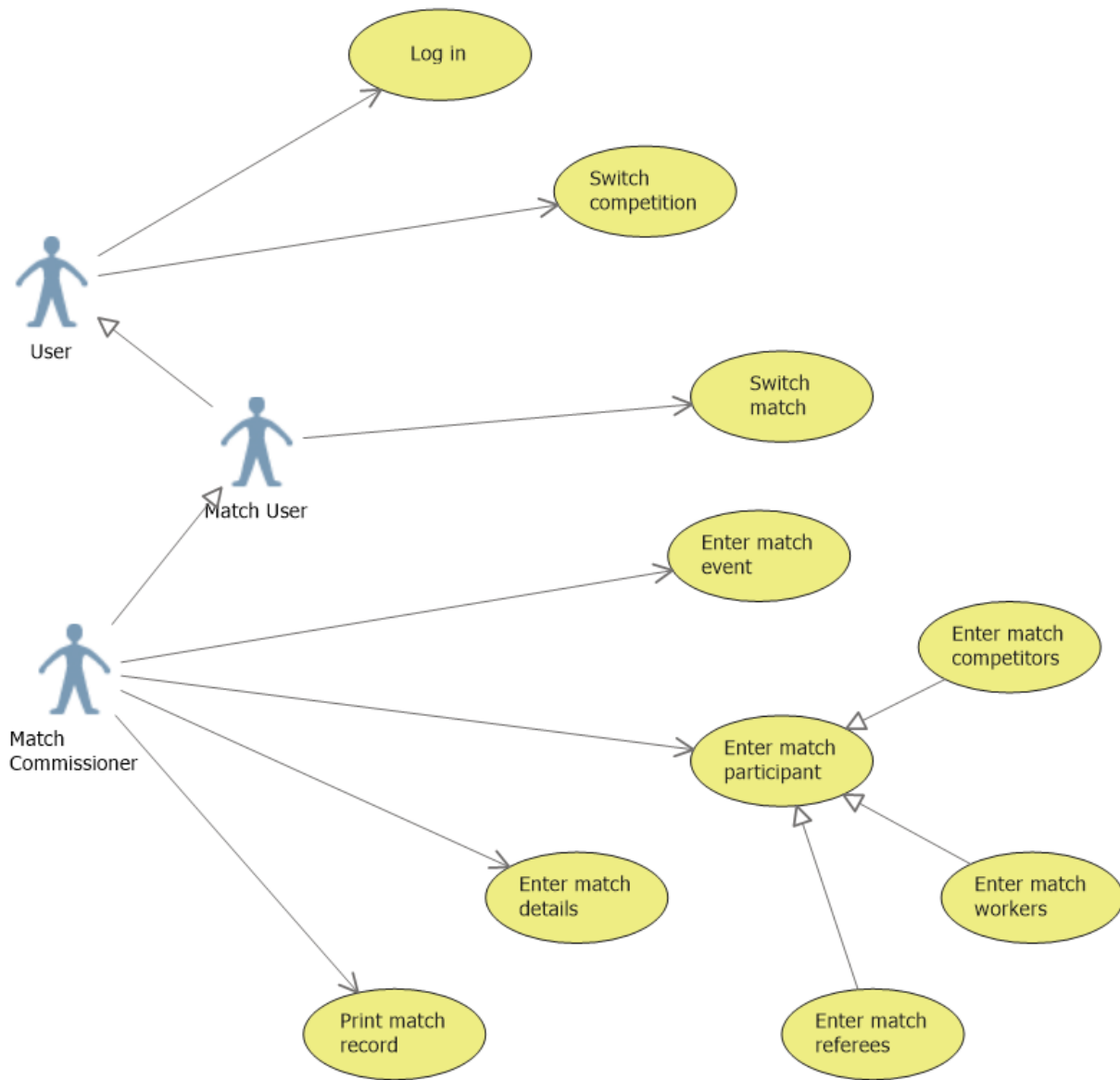


Figure 3. Match commissioner role use case

3.1.3 Match statistician role

UseCaseMatchStatistician

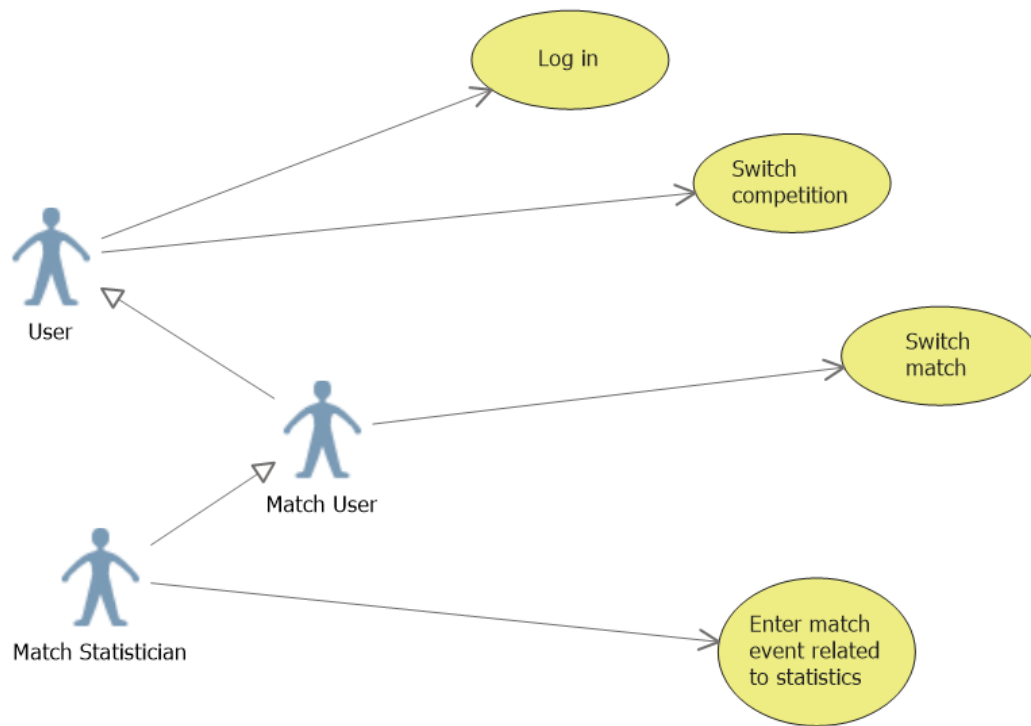


Figure 4. Match statistician role use case

3.1.4 Match reporter role

uc UseCaseMatchReporter

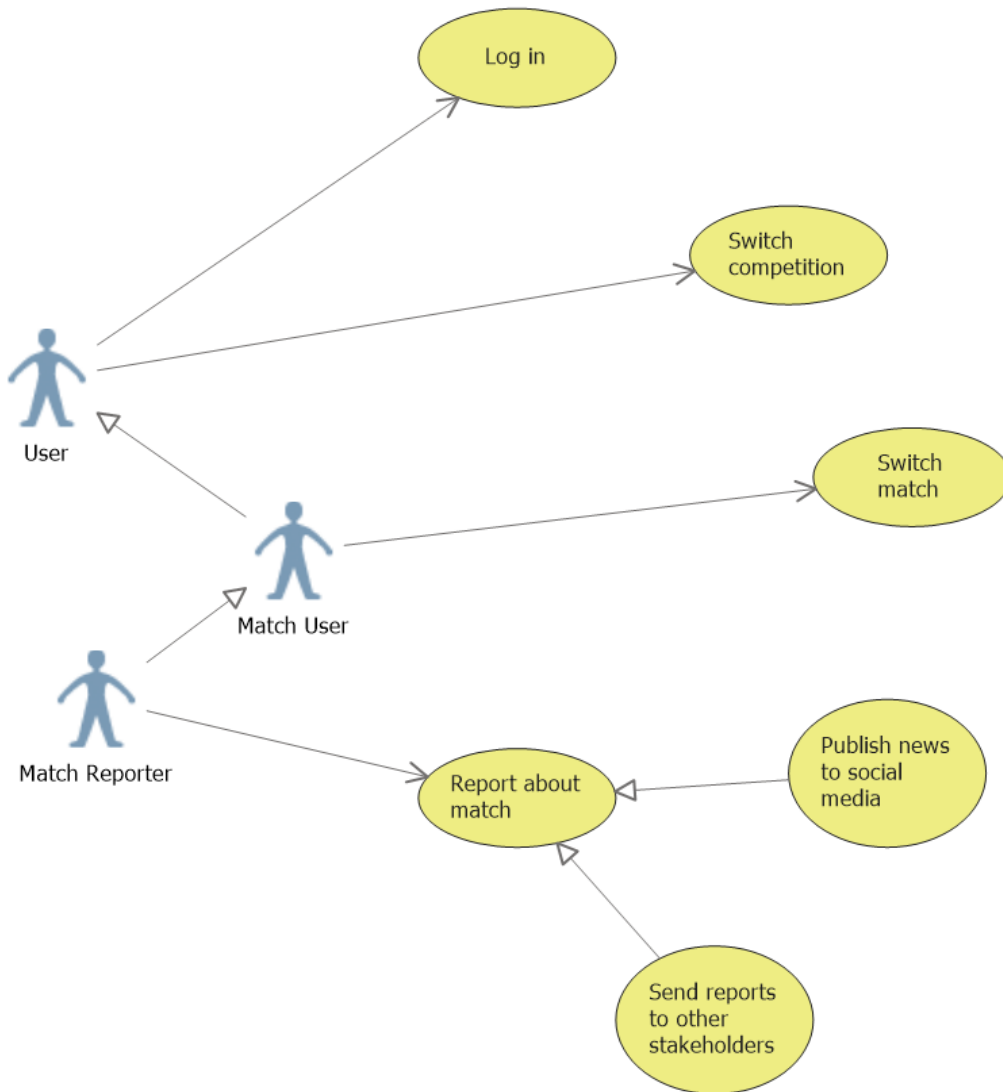


Figure 5. Match reporter role use case

3.2 Non-functional requirements

3.2.1 Ability to work offline

The application will be used in the environment where more than few thousand people including spectators can be situated in few hundred meters. In those conditions, if the devices will be connected on the mobile operator networks, they can often be without enough signal or connection to send data to server in real time. Either if devices are connected to wireless access points, the chances of constant connection and quality signal are extremely poor.

Therefore the ability to store data locally is of great importance. Because of that, the choice of robust client is chosen before the option of standard web client that runs in browser.

3.2.2 Syncing local data

Because the data are stored locally, it is important that the syncing feature is available at predefined periods, or on the events of restored network connection. The framework for syncing the locally stored data needs to be built.

3.2.3 Touch interface

The simplicity of use and the option of having mobile device which can be easily transferred to a place of better overview of the pitch or the competition event is mandatory so the choice of tablet devices with touch interface came as first choice.

3.2.4 Security

The application needs the authentication system and authorization system to make possible only for authorized personnel to insert and change the data related to competition and related events. The option of auditing the changed data is a possibility at later stage of application development because it will greatly improve the identification and sanctioning of security breaches if and when they happen.

4 User experience and interaction design

While designing the interaction in our application we tried to put main effort to the following features:

- Maintain consistency throughout the system
- Interactive behavior on user interface events
- Make the interface intuitive
- Highlight the features and information that are important to user

4.1 Menu Bar Highlighting



Figure 6. Non highlighted example of menu button



Figure 7. Highlighted example of menu button

In the following four sections we will provide figures of simple but intuitive design of our pages where you can on real examples see the consistency in the design throughout the system and also how important information needs to be highlighted and noticeable to user.

4.2 Switch competition view

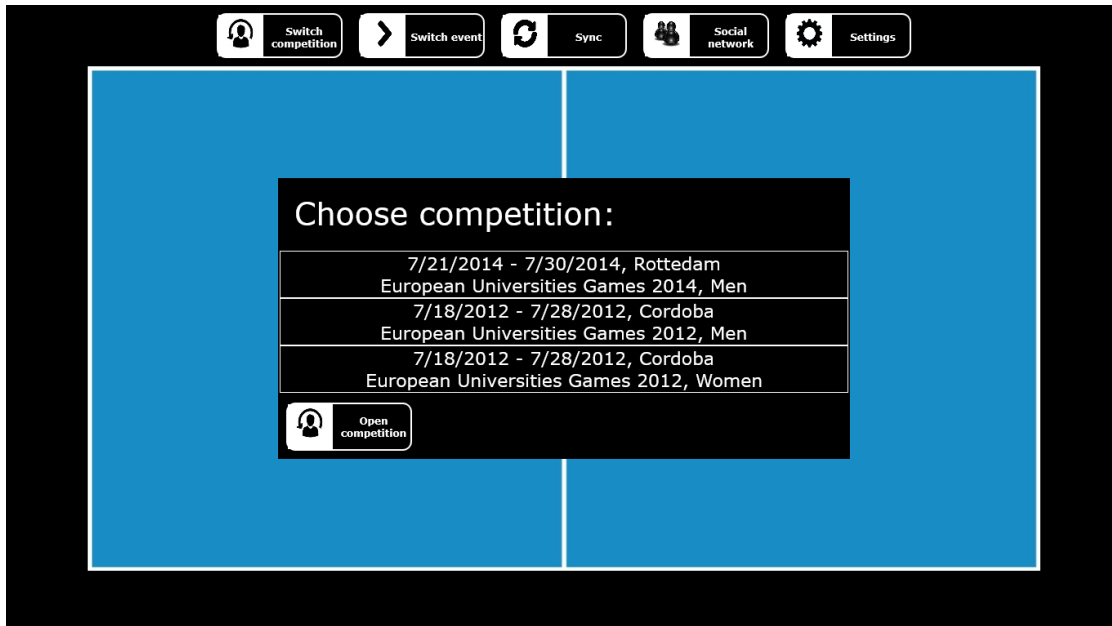


Figure 8. Switch competition view

4.3 Competition view

Teams	Referees	Workers
Vern University	Name180 Surname180 Eufa	Name200 Surname200 MatchCommissioner
Ljubljana University	Name179 Surname179 National	Name199 Surname199 Announcer
Politecnico di Milano	Name178 Surname178 Fifa	Name198 Surname198 Commentator
	Name177 Surname177 Eufa	Name197 Surname197 HallManager
	Name176 Surname176 National	Name196 Surname196 TechnicalDelegate
	Name175 Surname175 Fifa	Name195 Surname195 MatchCommissioner
	Name174 Surname174 Eufa	Name194 Surname194 Announcer
	Name173 Surname173 National	Name193 Surname193 Commentator
	Name172 Surname172 Fifa	Name192 Surname192 HallManager
	Name171 Surname171 Eufa	Name191 Surname191 TechnicalDelegate
	Name170 Surname170 National	Name190 Surname190 MatchCommissioner
	Name169 Surname169 Fifa	Name189 Surname189 Announcer
	Name168 Surname168 Eufa	Name188 Surname188 Commentator
	Name167 Surname167 National	Name187 Surname187 HallManager
	Name166 Surname166 Fifa	Name186 Surname186 TechnicalDelegate
	Name165 Surname165 Eufa	Name185 Surname185 MatchCommissioner

Figure 9. Competition view

4.4 Switch event view

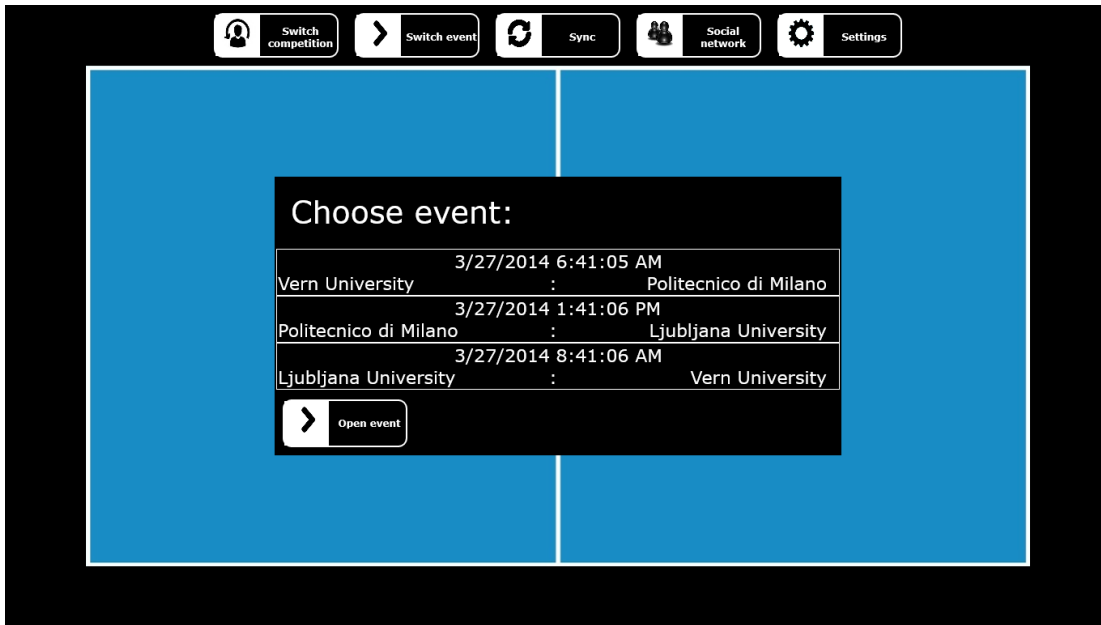


Figure 10. Switch event view

4.5 Event view

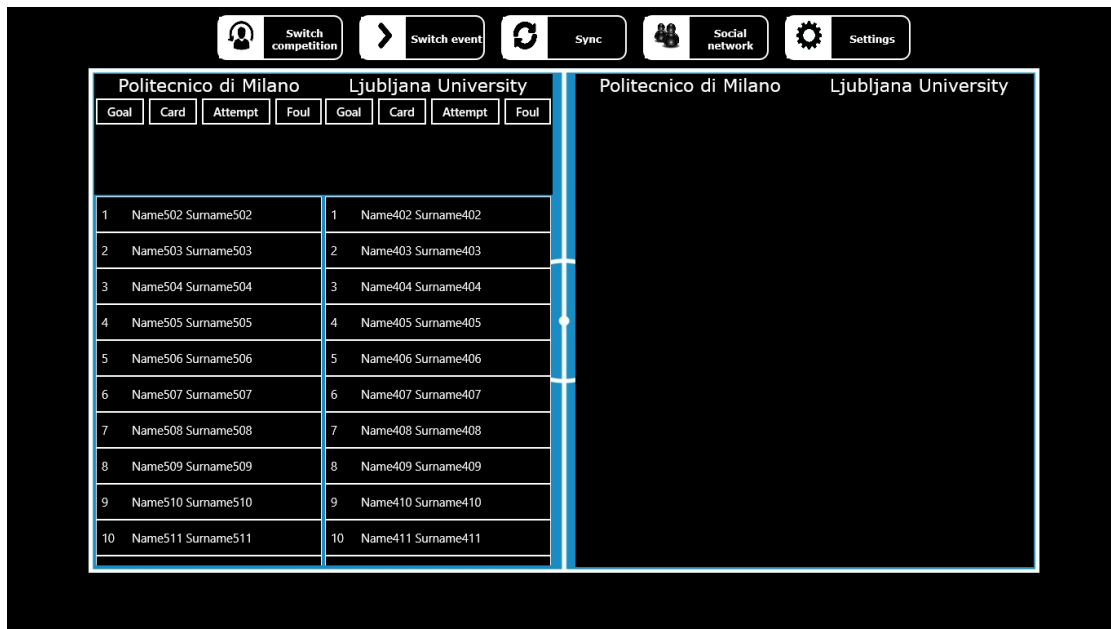


Figure 11. Event view

5 Architecture

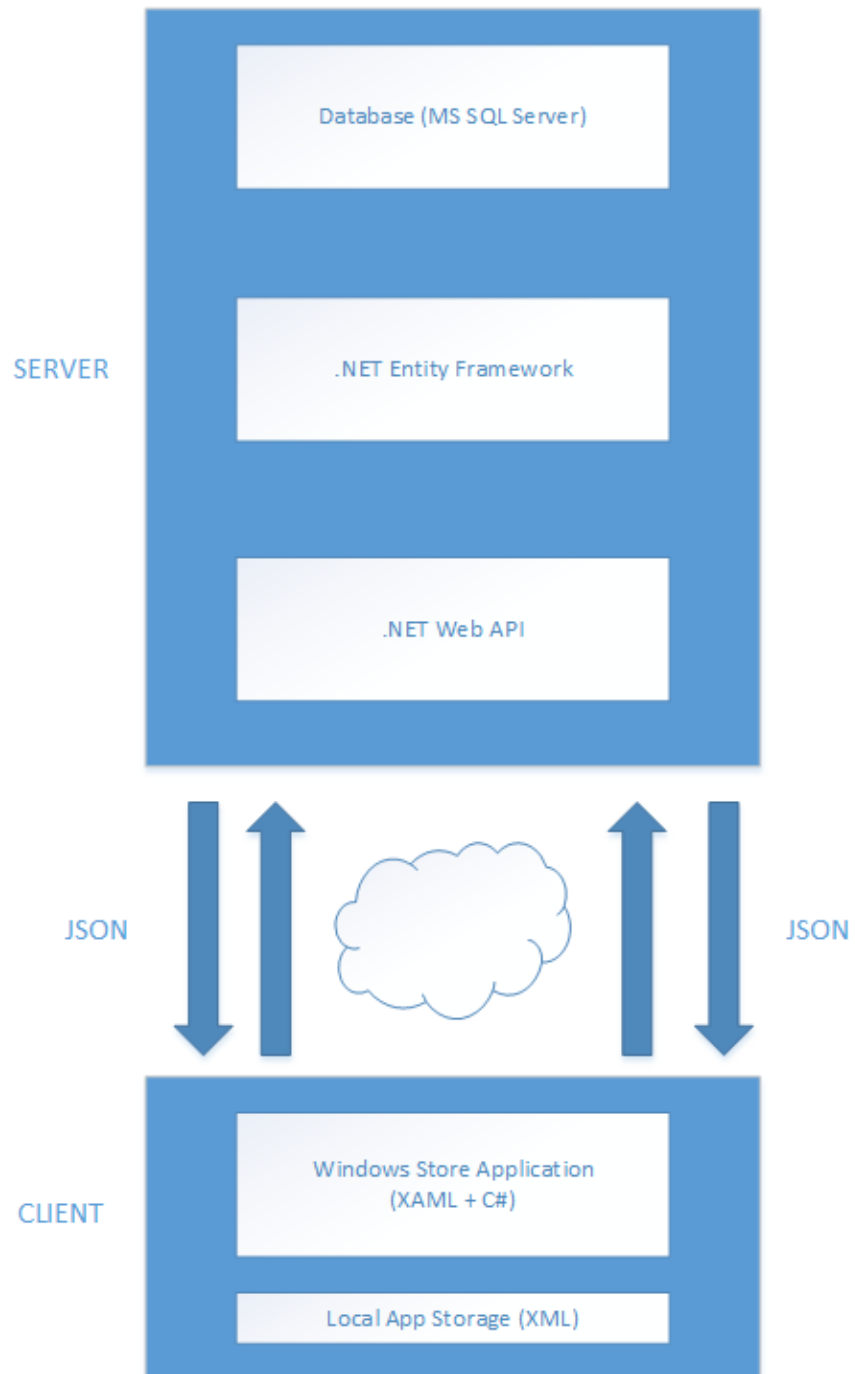


Figure 12. Application's overall architecture

5.1 Database layer

The database is implemented in MS SQL Server 2012 system. The initial architecture was created in SQL Server Management Studio's designer view. Afterwards the updates of the database were managed through code.

The database was created with future expansion for other sports in mind. Therefore, the database might look overly complicated just for the scope of futsal as a sport, but take into consideration the perspective of storing data for other sports too.

In the following we will enumerate all the tables in the database and briefly describe for each of them the data stored:

Database table	Table description
CompetitionFacilities	Facilities and related data like name, latitude, longitude, etc.
CompetitionMailingLists	Mailing lists for specific competitions on which the competition reports need to be sent
CompetitionPhases	Phases of the competition
Competitions	Competitions and related basic data
CompetitionTeams	Teams related to specific competition
Competitors	Competitors related to specific competition and specific team
Countries	Countries, their logos and shortcuts
DatabaseUpdates	Database management related data and the description of database updates
DrawPots	Pots related to draws
DrawRankings	Ranking of the teams participating in a draw
Draws	Draws of a specific competition
GroupsCrossings	Crossings defined for groups of a competition phase
GroupsPlayingFormats	Playing formats for a group
GroupTeams	Competition teams situated in a specific competition phase group
MailingListEmails	Emails related to a mailing list
MatchCompetitors	Roles and data related to specific match for a competitor
MatchEquipment	Equipment assigned to each of the teams for a specific match
Matches	Matches and all basic data for match
MatchEventCompetitors	Competitors related to a specific match event
MatchEvents	Events related to a match like points, cautions,

	statistics, etc.
MatchReferees	Referees and their roles and grades for a specific match
MatchTeams	Teams involved in a match
MatchWorkers	Workers involved in a match, like commentator, timekeeper, delegate, etc.
Organizations	Organizations related to competitions
Persons	All the basic data for a person to which all other tables like referees, workers, competitors are related to
PhaseGroups	Groups of a specific competition phase
PhasePlayingDates	Dates related to a competition phase
PlayingDateTimes	Times for a specific date of a competition phase
PotTeams	Teams situated to a pot of a specific competition phase draw
Referees	Referees and all basic data related to referees involved in a competition
Sports	Sports for which the competitions can be maid
TeamEquipment	Team equipment for a specific competition
Teams	Teams and basic data for each team
UserCompetitions	Users privileges for accessing a specific competition
Users	All users
ValidCompetitionRules	Rules valid for a specific competition that override the general rules defined for sport
ValidRules	Rule types
ValidSportRules	Rules valid for a specific sport which are effective in the competition if the rule was not defined for that competition specifically
Workers	Workers and all basic data related to workers involved in a competition

Table 6. Database tables

5.1.1 Competition core tables

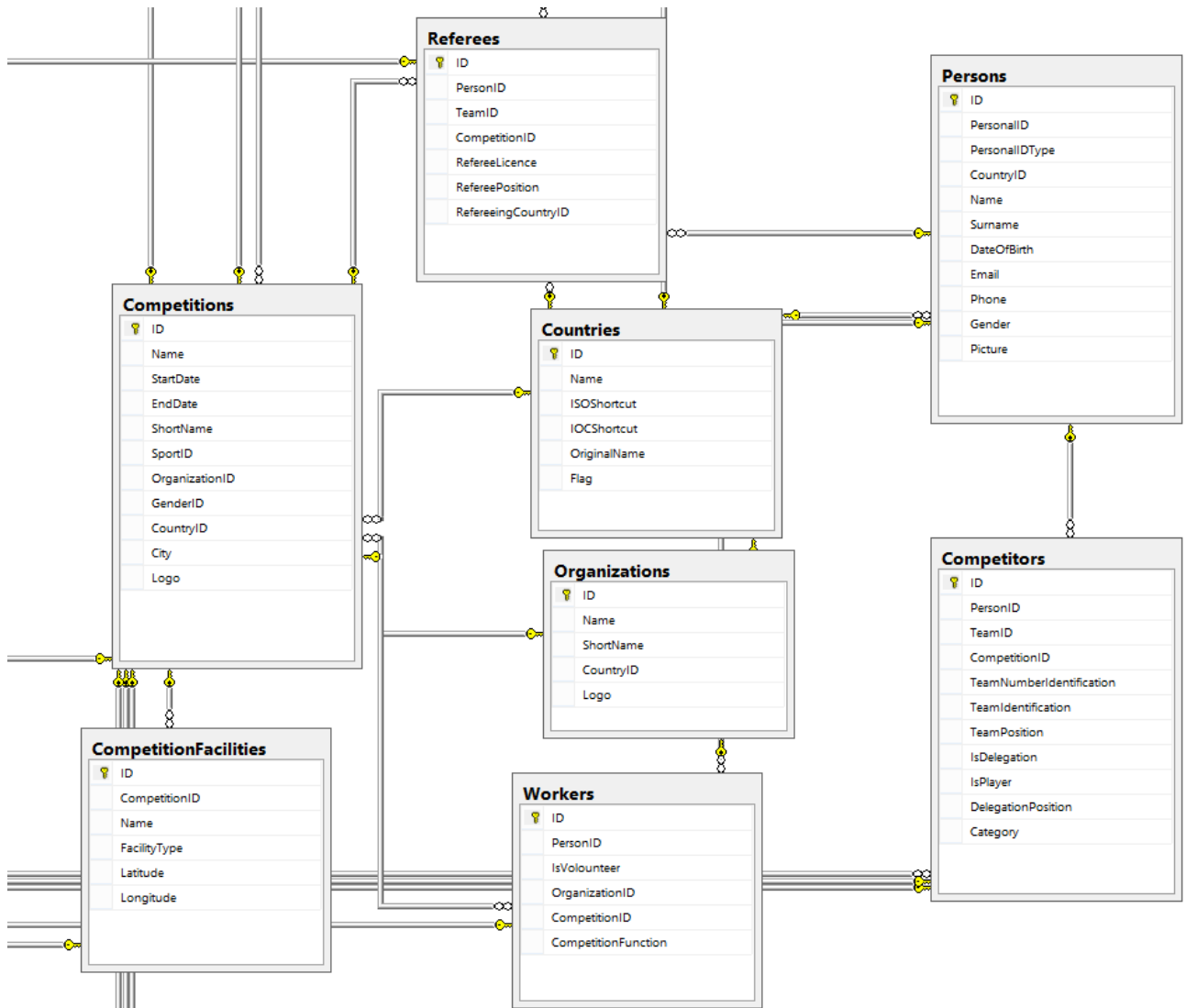


Figure 13. Competition core tables

5.1.2 Competition schedule tables

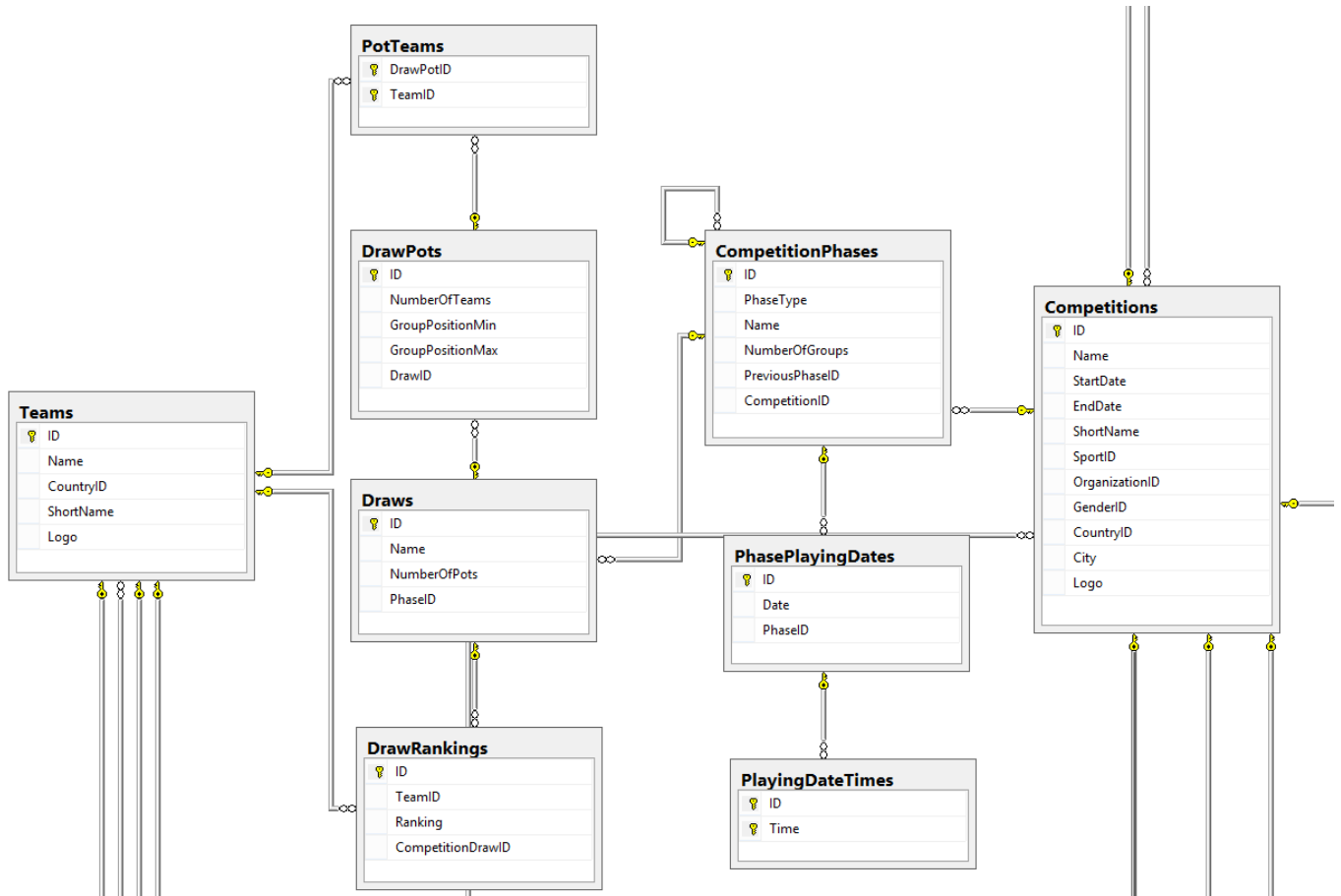


Figure 14. Competition schedule tables

5.1.3 Match core tables

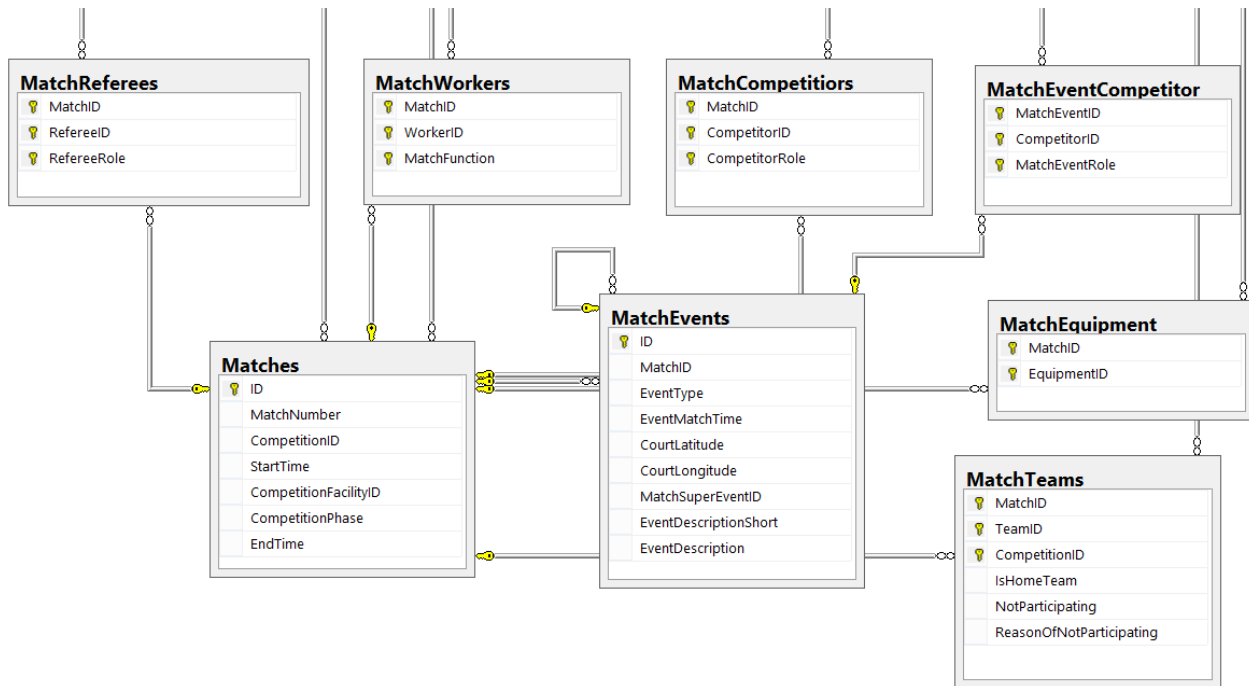


Figure 15. Match core tables

5.1.4 Competition and sport rules tables

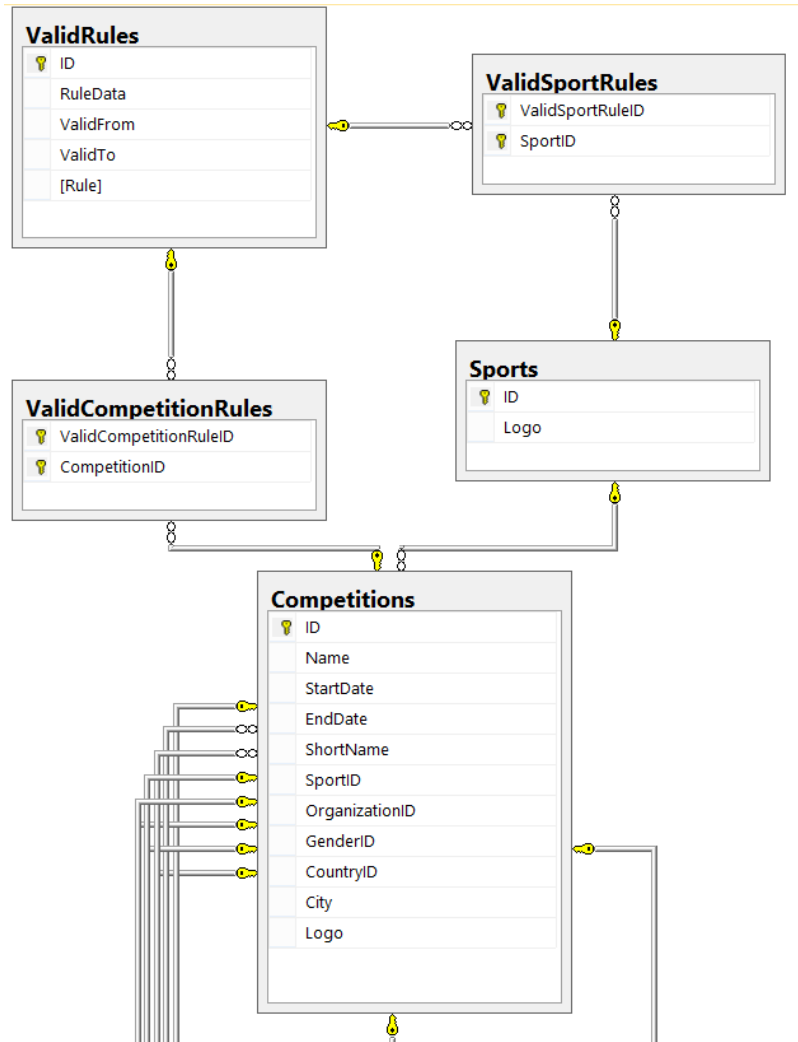


Figure 16. Competition and sport rules tables

5.1.5 Competition users and mailing lists tables

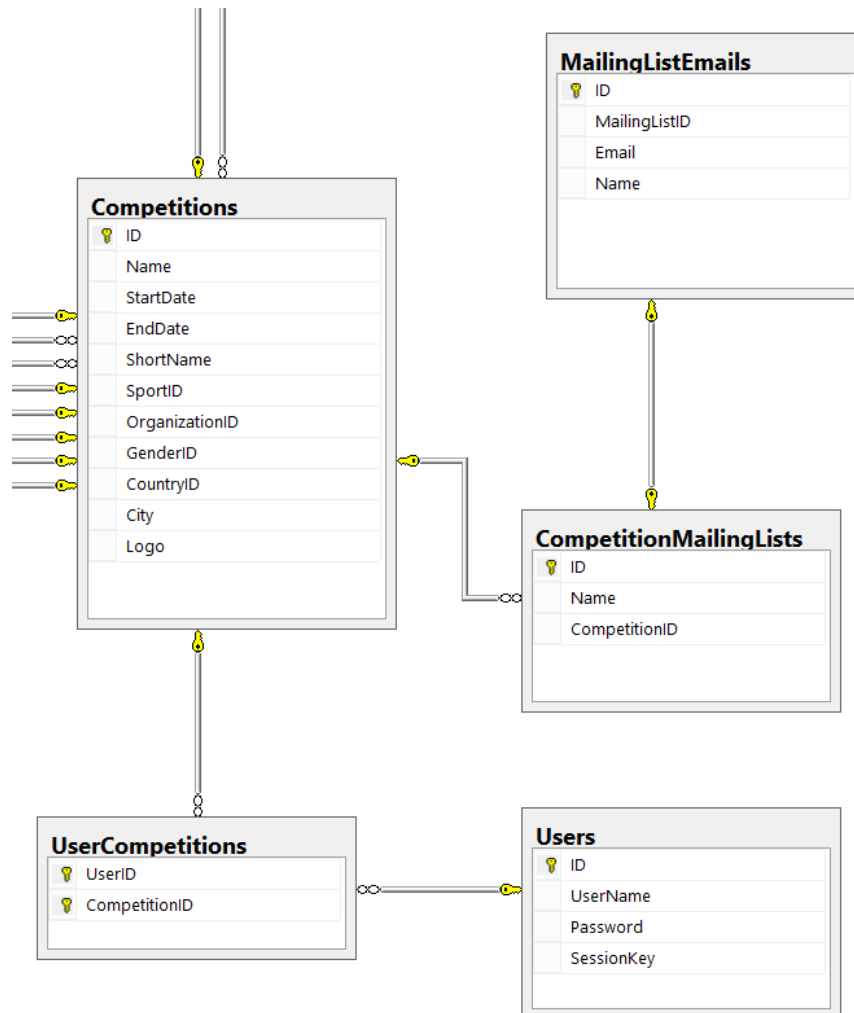


Figure 17. Competition users and mailing list tables

5.2 Model layer (Entity Framework)

The code first workflow was used in the Entity Framework. From the existing database with reverse engineering the entity framework classes were extracted. In that process, the significant role had the Entity Framework Power Tools extension for Visual Studio.

With the help of T4 templates, entity, context and mapping classes were generated directly from database. The things left to do were to make the queries, and insert, update and delete method for needed entities on top of entity framework.

In the following examples we will show the examples of queries for two main entities of the application:

```
public CompetitionDC GetCompetition(int competitionID)
{
    return smContext.Competitions
        .Where(c => c.ID == competitionID)
        .Include(c => c.Sport.ValidRules)
        .Include(c => c.Organization)
        .Include(c => c.Country)
        .Include(c => c.Workers.Select(cw => cw.Person))
        .Include(c => c.Referees.Select(cr => cr.Person))
        .Include(c => c.CompetitionTeams.Select(ct => ct.Team))
        .Include(c => c.CompetitionTeams.Select(ct => ct.Competitors))
        .Include(c => c.ValidRules)
        .Include(c => c.CompetitionFacilities)
        .Include(c => c.CompetitionMailingLists)
        .First();
}
```

Code Example 11. Query for retrieving competition and all related data

```
return smContext.Competitions
    .Include(c => c.Sport)
    .Include(c => c.Organization)
    .Include(c => c.Country);
```

Code Example 12. Query for retrieving all competitions

```

public MatchDC GetMatch(long matchID)
{
    return smContext.Matches
        .Where(m => m.ID == matchID)
        .Include(m => m.MatchCompetitors.Select(mc => mc.Competitor))
        .Include(m => m.MatchEvents)
        .Include(m => m.MatchReferees.Select(mr => mr.Referee))
        .Include(m => m.MatchTeams)
        .Include(m => m.MatchTeams.Select(x => x.CompetitionTeam.Team))
        .Include(m => m.MatchTeams.Select(x => x.CompetitionTeam.Competitors.Select(y => y.Person)))
        .Include(m => m.MatchWorkers.Select(mw => mw.Worker.Person))
        .Include(m => m.TeamEquipments)
        .Include(m => m.CompetitionFacility)
        .Include(m => m.Competition)
        .Include(m => m.Competition.Organization)
        .Include(m => m.Competition.Sport.ValidRules)
        .Include(m => m.Competition.Workers.Select(cw => cw.Person))
        .Include(m => m.Competition.Referees.Select(cr => cr.Person))
        .Include(m => m.Competition.Country)
        .Include(m => m.Competition.ValidRules)
        .Single();
}

```

Code Example 13. Query for retrieving competition event and all related data

```

return smContext.Matches
    .Where(m => m.CompetitionID == competitionID)
    .Include(m => m.MatchTeams.Select(mt => mt.CompetitionTeam.Team));

```

Code Example 14. Query for retrieving all matches of a competition

5.3 Controller layer (Web API)

Web API is the controller layer of our application. For each important entity or a relation we have the according controller. In the following table we will show the controllers and related resources for the few main entities in our application:

Controller	Resources
CompetitionsController	GET /api/competitions/
	GET /api/competitions/{id}
	POST /api/competitions/
	PUT /api/competitions/{id}
MatchesController	GET /api/matches/{id}
	PUT /api/matches/{id}
	POST /api/matches/
CompetitionMatchesController	GET /api/competitions/{id}/matches/{date}
	GET /api/competitions/{id}/matches/
MatchEventsController	GET /api/matches/{matchID}/matchevents/
	GET /api/matches/{matchID}/matchevents/{id}
	PUT /api/matches/{matchID}/matchevents/{id}
	POST /api/matches/{matchID}/matchevents/

Table 7. Web API controllers for main entities

5.4 View layer (Windows 8.1 application)

The client application was made as a Windows 8.1 application. The technologies used were c# language and XAML for layout. The application was made with a master page layout. One page with a frame and standard button bar on top was made as master page and all the other pages just defined the layout of the inner frame.

The communication with the server went through http communication and the format exchanged was JSON because of the smallest size that it has compared to xml or some other format of serialized data.

In the following example we will show the code and layout of the master page.

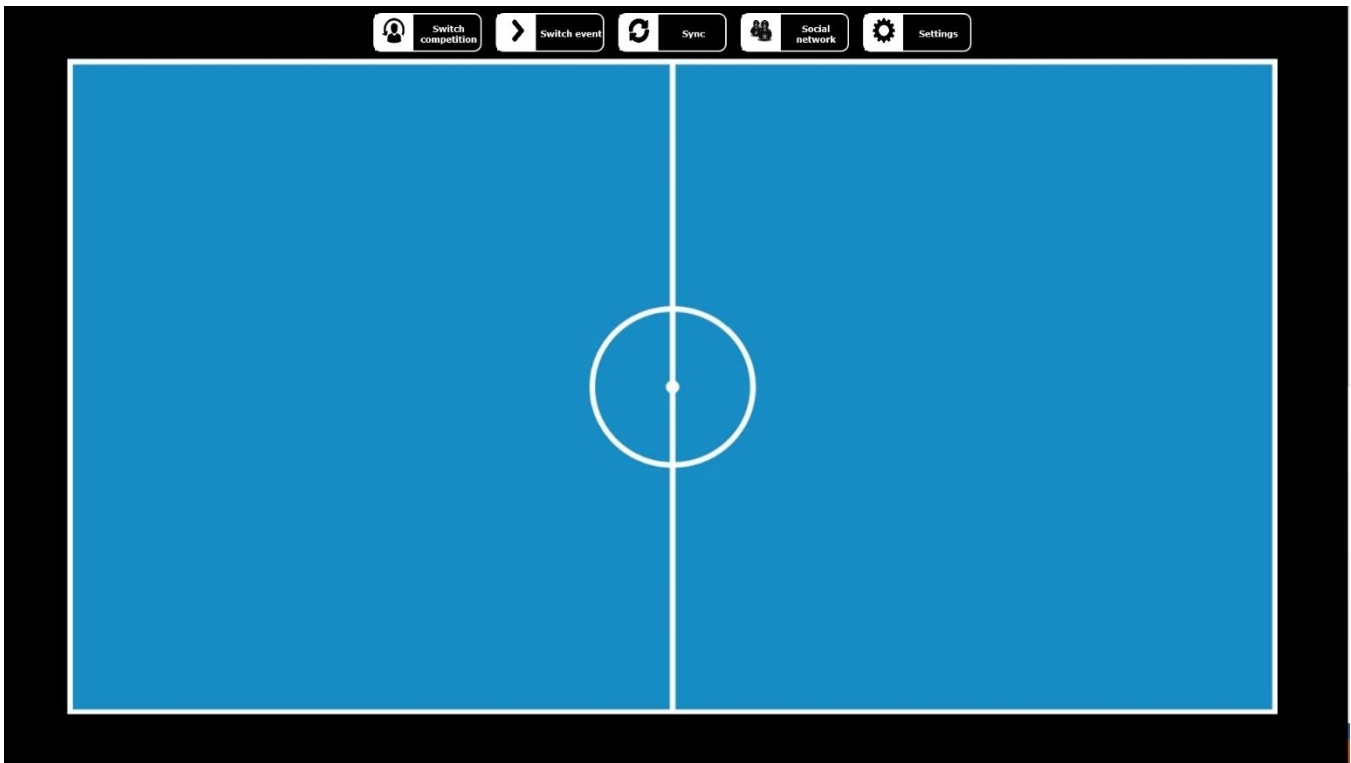


Figure 18. Master page layout

```

<Page
  x:Class="SportsManager.MasterPage"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:local="using:SportsManager"
  xmlns:cc="using:SportsManager.CustomControls"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  mc:Ignorable="d">

  <Grid x:Name="MasterPageGrid" Background="black">
    <Grid.RowDefinitions>
      <RowDefinition Height="75"/>
      <RowDefinition MaxHeight="1500"/>
      <RowDefinition Height="75"/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
      <ColumnDefinition Width="100" />
      <ColumnDefinition MaxWidth="2000" />
      <ColumnDefinition MaxWidth="2000" />
      <ColumnDefinition Width="100" />
    </Grid.ColumnDefinitions>

    <Frame x:Name="frameBody" Grid.Row="1" Grid.Column="1" Grid.ColumnSpan="2">
      <Frame.Background>
        <ImageBrush Stretch="Fill" ImageSource="Assets/Backgrounds/futsal_court2.jpg" AlignmentY="Center"
  AlignmentX="Center"/>
      </Frame.Background>
    </Frame>

    <StackPanel x:Name="panelRightButtons" Grid.Column="1" Grid.ColumnSpan="2" Grid.Row="0" Orientation="Horizontal"
  Height="75" HorizontalAlignment="Center" >
      <TextBlock x:Uid="test.Content"></TextBlock>
      <cc:SMBUTTON Name="btnSwitchCompetitionUser" Click="btnSwitchCompetition_Click"
        ImagePath="/Assets/Buttons/changeUser.png"
        PressedImagePath="/Assets/Buttons/changeUser_press.png" >
        Switch competition
      </cc:SMBUTTON>
      <cc:SMBUTTON Name="btnSwitchEvent" Click="btnSwitchEvent_Click"
        ImagePath="/Assets/Buttons/switch.png"
        PressedImagePath="/Assets/Buttons/switch_press.png">
        Switch event
      </cc:SMBUTTON>
      <cc:SMBUTTON Name="btnSync"
        ImagePath="/Assets/Buttons/sync.png"
        PressedImagePath="/Assets/Buttons/sync_press.png">
        Sync
      </cc:SMBUTTON>
      <cc:SMBUTTON Name="btnSocial"
        ImagePath="/Assets/Buttons/social.png"
        PressedImagePath="/Assets/Buttons/social_press.png">
        Social network
      </cc:SMBUTTON>
      <cc:SMBUTTON Name="btnSettings"
        ImagePath="/Assets/Buttons/settings.png"
        PressedImagePath="/Assets/Buttons/settings_press.png">
        Settings
      </cc:SMBUTTON>
    </StackPanel>
  </Grid>
</Page>

```

Code Example 15. Master page layout in XAML

6 User manual

6.1 Choose and manage competition

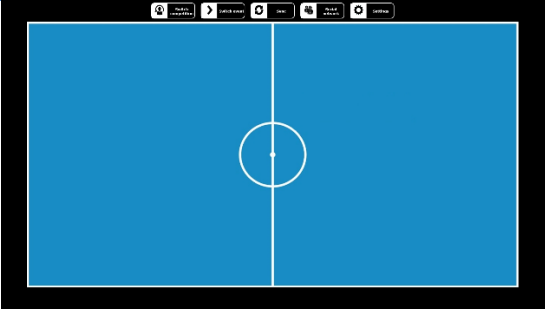
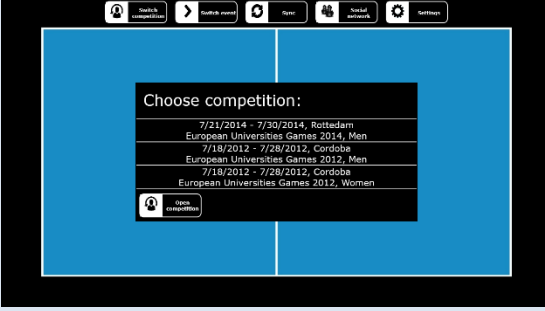

View	Actions																																																			
	<ol style="list-style-type: none"> 1. Click on the most left button in the menu bar on top 2. Then the window in the next step will be shown 																																																			
	<ol style="list-style-type: none"> 1. Click on one of the rows representing the competition that we want to choose 2. The row will be highlighted 3. Click "Open competition" situated on the bottom of list of competitions 4. Wait for few seconds and the view in the next step will be opened 																																																			
 <table border="1"> <thead> <tr> <th>Teams</th> <th>Referees</th> <th>Workers</th> </tr> </thead> <tbody> <tr> <td>Vrije Universiteit</td> <td>Name101 Suriname101, Iudith</td> <td>Name200 Suriname200, MaitheCompassion</td> </tr> <tr> <td>Utrecht University</td> <td>Name129 Suriname129, National</td> <td>Name199 Suriname199, Antwanance</td> </tr> <tr> <td>Northwood @ Miami</td> <td>Name178 Suriname178, Eifa</td> <td>Name198 Suriname198, Communication</td> </tr> <tr> <td></td> <td>Name177 Suriname177, Iudith</td> <td>Name197 Suriname197, HRManager</td> </tr> <tr> <td></td> <td>Name176 Suriname176, National</td> <td>Name196 Suriname196, HRManager</td> </tr> <tr> <td></td> <td>Name175 Suriname175, Eifa</td> <td>Name195 Suriname195, MediaCommunication</td> </tr> <tr> <td></td> <td>Name174 Suriname174, Iudith</td> <td>Name194 Suriname194, Antwanance</td> </tr> <tr> <td></td> <td>Name173 Suriname173, National</td> <td>Name193 Suriname193, Communication</td> </tr> <tr> <td></td> <td>Name172 Suriname172, Eifa</td> <td>Name192 Suriname192, HRManager</td> </tr> <tr> <td></td> <td>Name171 Suriname171, Iudith</td> <td>Name191 Suriname191, Antwanance</td> </tr> <tr> <td></td> <td>Name170 Suriname170, National</td> <td>Name190 Suriname190, MaitheCompassion</td> </tr> <tr> <td></td> <td>Name169 Suriname169, Eifa</td> <td>Name189 Suriname189, Antwanance</td> </tr> <tr> <td></td> <td>Name168 Suriname168, Iudith</td> <td>Name188 Suriname188, Communication</td> </tr> <tr> <td></td> <td>Name167 Suriname167, National</td> <td>Name187 Suriname187, HRManager</td> </tr> <tr> <td></td> <td>Name166 Suriname166, Iudith</td> <td>Name186 Suriname186, Antwanance</td> </tr> <tr> <td></td> <td>Name165 Suriname165, Eifa</td> <td>Name185 Suriname185, MaitheCompassion</td> </tr> </tbody> </table>	Teams	Referees	Workers	Vrije Universiteit	Name101 Suriname101, Iudith	Name200 Suriname200, MaitheCompassion	Utrecht University	Name129 Suriname129, National	Name199 Suriname199, Antwanance	Northwood @ Miami	Name178 Suriname178, Eifa	Name198 Suriname198, Communication		Name177 Suriname177, Iudith	Name197 Suriname197, HRManager		Name176 Suriname176, National	Name196 Suriname196, HRManager		Name175 Suriname175, Eifa	Name195 Suriname195, MediaCommunication		Name174 Suriname174, Iudith	Name194 Suriname194, Antwanance		Name173 Suriname173, National	Name193 Suriname193, Communication		Name172 Suriname172, Eifa	Name192 Suriname192, HRManager		Name171 Suriname171, Iudith	Name191 Suriname191, Antwanance		Name170 Suriname170, National	Name190 Suriname190, MaitheCompassion		Name169 Suriname169, Eifa	Name189 Suriname189, Antwanance		Name168 Suriname168, Iudith	Name188 Suriname188, Communication		Name167 Suriname167, National	Name187 Suriname187, HRManager		Name166 Suriname166, Iudith	Name186 Suriname186, Antwanance		Name165 Suriname165, Eifa	Name185 Suriname185, MaitheCompassion	<ol style="list-style-type: none"> 1. There will be three columns on the screen, one representing teams, one referees and one workers 2. Each of the entities can be chosen and edited
Teams	Referees	Workers																																																		
Vrije Universiteit	Name101 Suriname101, Iudith	Name200 Suriname200, MaitheCompassion																																																		
Utrecht University	Name129 Suriname129, National	Name199 Suriname199, Antwanance																																																		
Northwood @ Miami	Name178 Suriname178, Eifa	Name198 Suriname198, Communication																																																		
	Name177 Suriname177, Iudith	Name197 Suriname197, HRManager																																																		
	Name176 Suriname176, National	Name196 Suriname196, HRManager																																																		
	Name175 Suriname175, Eifa	Name195 Suriname195, MediaCommunication																																																		
	Name174 Suriname174, Iudith	Name194 Suriname194, Antwanance																																																		
	Name173 Suriname173, National	Name193 Suriname193, Communication																																																		
	Name172 Suriname172, Eifa	Name192 Suriname192, HRManager																																																		
	Name171 Suriname171, Iudith	Name191 Suriname191, Antwanance																																																		
	Name170 Suriname170, National	Name190 Suriname190, MaitheCompassion																																																		
	Name169 Suriname169, Eifa	Name189 Suriname189, Antwanance																																																		
	Name168 Suriname168, Iudith	Name188 Suriname188, Communication																																																		
	Name167 Suriname167, National	Name187 Suriname187, HRManager																																																		
	Name166 Suriname166, Iudith	Name186 Suriname186, Antwanance																																																		
	Name165 Suriname165, Eifa	Name185 Suriname185, MaitheCompassion																																																		

Table 8. User actions for managing competitions

6.2 Choose and manage competition event

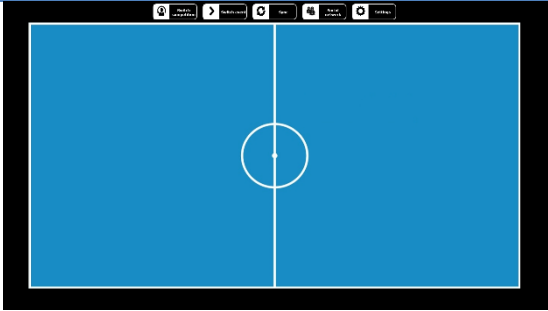
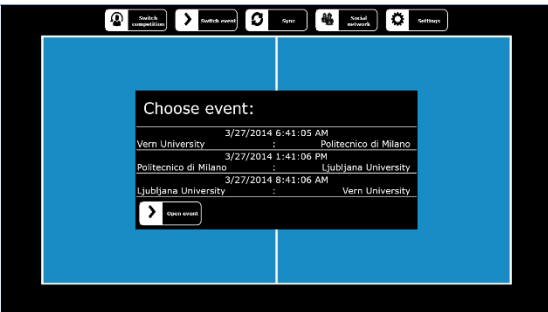
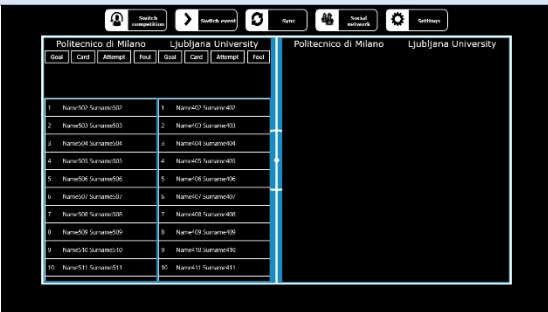
View	Actions
	<ol style="list-style-type: none"> 1. Click on the second from the left button in the menu bar on top 2. Then the window in the next step will be shown
	<ol style="list-style-type: none"> 1. Click on one of the rows representing the event that we want to choose 2. The row will be highlighted 3. Click "Open event" situated on the bottom of list of events 4. Wait for few seconds and the view in the next step will be opened
	<ol style="list-style-type: none"> 1. There will be four columns on the screen, first and third for home team, and second and fourth for away team 2. In the left two columns, in the top part, there are buttons for creating match events, goals, cards, fouls, etc. 3. It is enough to choose and mark the player from the list and click on event, and the event will be created which will be manifested in the third or fourth column

Table 9. User actions for managing competition events

7 Conclusion

We have briefly showed the technologies we've used in the project. They are all latest state of the art technologies on Microsoft's .NET platform. We also need to mention that we used the latest stable versions of all these technologies. Regarding the application we developed, it is a stable program for managing futsal competitions and futsal competition events. Developed for windows 8.1 operating system, the application keeps the data locally in an xml files, and connects to sync with service developed as Web API. Never the less, the application still needs to pass the exhaustive testing phase, and the decision needs to be made whether to distribute the application as an enterprise application only to customers that purchase it or through windows store to make it available to everyone.

To conclude, the application idea is solid. There are already potential customers interested in the idea, like European Universities Sports Association (EUSA) and International University Sports Federation (FISU). Other potential customers for sure exist somewhere out there, because there are millions of competitions in dozens of sports organized throughout the world. Our mission would be to create top quality product with a lot of distinguished features and to find the customers hiding somewhere in the real world.

Plans for the extension of the application are numerous. Firstly, the idea would be to extend the application to support different sports, mainly team sports, but other individual sports are not excluded. Other big milestone would be to extend the application on different platforms, firstly Android, than if need arises, iOS. Afterwards, there are ideas for the extension to offer to the clients to create their web site that can easily be built to use the existing web service and underlying layers of our solution According to that idea, a Content managements system (CMS) can be built therefore allowing us to more quickly provide the web portals as additional product to our clients.

8 Appendices

8.1 Appendix A: Versioning system details

Detail	Value
Type	Team Foundation Server
Uri	webko.visualstudio.com
Owner	marko.brcic@outlook.com
Team project	Sports Manager

Table 10. Versioning system details

8.2 Appendix B: Development tools used

Development tool	Description
Visual Studio 2013	Microsoft's popular and robust IDE
MS SQL Server 2012	SQL Server solution from Microsoft, including Management Studio for connecting to databases
Fiddler	Traffic sniffer and requests composer from Telerik used for testing Web API
Notepad++	Simple editor for viewing xml and json strings

Table 11. Development tools used

Visual Studio 2013 Extension	Description
T4 editor	Editor for t4 templates
RockMargin	Code highlighter
Productivity Power Tools 2013	Set of extra commands added to Visual studio
Nuget Package Manager	Package manager
Entity Framework Power Tools Beta 4	Set of extra commands for reverse engineering existing database into entity framework code first model

Table 12. Visual Studio 2013 extensions used

8.3 Appendix C: Visual Studio solution projects

Project name	Project description
SportsManager	Windows store 8.1 application project
SportsManager.Tests	Windows store 8.1 unit tests project
SportsManager.UITests	Windows store 8.1 user interface tests project
SportsManagerAPI	Web API project
SportsManagerAPI.Tests	Web API unit tests project
SportsManagerEF	Entity Framework project
SportsManagerModel	Model and Http Client project for different .NET project types and platforms
SportsManagerDocumentation	Project for UML documentation
DatabaseClearAndFill	Project for filling database environment with test data

Table 13. SportsManager solution projects

8.4 Appendix D: List of figures

Figure 1. Sports Manager application logo	13
Figure 2. Competition administrator role use case.....	16
Figure 3. Match commissioner role use case	17
Figure 4. Match statistician role use case	18
Figure 5. Match reporter role use case	19
Figure 6. Non highlighted example of menu button	21
Figure 7. Highlighted example of menu button	21
Figure 8. Switch competition view	22
Figure 9. Competition view	22
Figure 10. Switch event view.....	23
Figure 11. Event view	23
Figure 12. Application's overall architecture	24
Figure 13. Competition core tables.....	27
Figure 14. Competition schedule tables.....	28
Figure 15. Match core tables.....	29
Figure 16. Competition and sport rules tables.....	30
Figure 17. Competition users and mailing list tables	31
Figure 18. Master page layout	35

8.5 Appendix E: List of tables

Table 1. Existing development workflows in Entity Framework.....	3
Table 2. Relations between CRUD operations and HTTP request types.....	8
Table 3. Example of actions and related URI resources for product entity	8
Table 4. Technologies and languages available for developing windows store apps.....	11
Table 5. Application roles and their descriptions.....	15
Table 6. Database tables	26
Table 7. Web API controllers for main entities	34
Table 8. User actions for managing competitions	37
Table 9. User actions for managing competition events	38
Table 10. Versioning system details	40
Table 11. Development tools used.....	40
Table 12. Visual Studio 2013 extensions used	40
Table 13. SportsManager solution projects	41

8.6 Appendix F: List of code examples

Code Example 1. Code first Entity Framework workflow	4
Code Example 2. Fluent API example.....	5
Code Example 3. Data annotations example	5
Code Example 4. Database migration example	5
Code Example 5. Eager loading example	6
Code Example 6. Lazy loading example	6
Code Example 7. Explicit loading example	7
Code Example 8. Web API controller example	9
Code Example 9. WebApiConfig example of defining routes	10
Code Example 10. Web API client call example	10
Code Example 11. Query for retrieving competition and all related data.....	32
Code Example 12. Query for retrieving all competitions.....	32
Code Example 13. Query for retrieving competition event and all related data	33
Code Example 14. Query for retrieving all matches of a competition.....	33
Code Example 15. Master page layout in XAML	36

9 Bibliography

- [1] Brown, Pete. Windows Store App Development (2013), *C# and XAML*.
- [2] Moemeka, Elizabeth and Edward. Real World Windows 8 App Development with Javascript (2013), *Create Great Windows Store Apps*
- [3] Falafel Software. Pro Windows phone App Development, *3rd edition*(2013)
- [4] Lee, Henry and Eugene Chuvyrov. Beginning Windows Phone App Development (2012)
- [5] Lee, Henry and Eugene Chuvyrov. Beginning Windows Phone 7 Development , *2nd edition*(2011)
- [6] Isaacs, Scott and Burns, Kyle. Beginning Windows Store App Development(2013), *HTML and JavaScript edition*
- [7] Maitra, Sumit. Building a Windows 8 Store App (2013), *End-to-End Windows 8 Store Application development using C# and XAML*
- [8] Freeman Adam. The Definitive Guide to HTML5 (2011)
- [9] Freeman, Adam. Pro ASP.NET MVC 4 (2013)
- [10] Troelsten, Andrew. Pro C# 5.0 and the .NET 4.5 Framework (2012) (*Expert's Voice in .NET*)
- [11] Jamie, Kurtz. ASP.NET MVC 4 and the Web API (2013) : *Building a REST Service from Start to Finish*
- [12] Millett, Scott. Professional ASP.NET Design Patterns (2010)
- [13] Badrinarayanan Lakshmiraghavan. Pro ASP.NET Web API Security (2013) : *Securing ASP.NET Web API*
- [14] Ugurlu, Tugberk. Pro ASP.NET API (2013) : *HTTP Web Services in ASP.NET*
- [15] Lerman, Julia and Miller, Rowan. Programming Entity Framework: Code First (2011)
- [16] Albahari, Joseph and Ben. C# 5.0 in a Nutshell (2012)
- [17] Lerman, Julia. Programming Entity Framework: *Building Data Centric Apps with the ADO.NET Entity Framework*(2010)
- [18] Lerman, Julia. Programming Entity Framework: DbContext (2012)
- [19] Chadwick, Jess and Snyder, Todd and Panda, Hrusikesh. Programming ASP.NET MVC 4: *Developing Real-World Web Applications with ASP.NET MVC* (2012)
- [20] Skeet, Jon. C# in Depth, 3rd edition (2013)
- [21] Badrinarayanan Lakshmiraghavan. Practical ASP.NET Web API (2013)
- [22] Freeman, Adam. Pro JavaScript for Web Apps (2012)
- [23] Collins, Mark. Pro HTML5 with Visual Studio 2012 (2012)

- [24] Sowell, Eric. Mobile ASP.NET MVC 5 (2013)
- [25] Freeman,Adam. Pro ASP.NET MVC 5 (2013)
- [26] Bewis, Tony. C# Design Pattern Essentials (2012)
- [27] Johnson, Bruce. Professional Visual Studio 2012 (2012)
- [28] István Novák ,Zoltan Arvai, György Balássy, David Fulop .Beginning Windows 8 Application Development (2012)
- [29] Garland,John. Windows Store Apps Succinctly
- [30] Baharestani Daniel. Mastering Ninject for Dependency Injection(2013)
- [31] Mark, Seemann. Dependency Injection in .NET(2011)
- [32] Osherove, Roy. The Art of Unit Testing: with examples in C# (2013)
- [33] MacDonald, Matthew. Beginning ASP.NET in C# (2012)
- [34] Brian Driscoll, Nitin Gupta, Robert Vettor and Zeeshan Hirani. Entity Framework 6 Recipes (2013)

10 Internet resources

[35] App features from start to finish (XAML). Retrieved from

<http://msdn.microsoft.com/en-us/library/windows/apps/xaml/dn632431.aspx>

[36] App features from start to finish (HTML). Retrieved from

<http://msdn.microsoft.com/en-us/library/windows/apps/dn263202.aspx>

[37] Installing apps on PC. Retrieved from

<http://windows.microsoft.com/en-us/windows-8/apps-windows-store-tutorial>

[38] Windows Store DirectX game programming environment. Retrieved from

<http://msdn.microsoft.com/library/windows/apps/dn166876.aspx>

[39] Publishing an app to the Windows Store. Retrieved from

<http://msdn.microsoft.com/en-us/library/windows/apps/jj657972.aspx>

[40] Apps for Windows. Retrieved from

<http://windows.microsoft.com/en-us/windows-8/apps#Cat=t1>

[41] Interaction with apps. Retrieved from

<http://msdn.microsoft.com/en-us/windows/apps/hh974576.aspx>

[42] Developing UI with XAML. Retrieved from

<http://msdn.microsoft.com/en-us/library/windows/apps/xaml/hh465340.aspx>

[43] Windows Store Apps Development. Retrieved from

<http://www.microsoftvirtualacademy.com/product-training/product-windows-store-apps>

[44] Windows 8 App development. Retrieved from

<https://www.dreamspark.com/student/windows-8-app-development.aspx>

[45] ASP.NET Web API. Retrieved from

<http://www.asp.net/web-api/overview>

[46] Building ASP.Net Web API Restful Service. Retrieved from

<http://www.asp.net/web-api/overview>

- [47] ASP.NET MVC 5 Full tutorial about new Authentication mechanism. Retrieved from <http://forums.asp.net/t/1944903.aspx?ASP+NET+MVC+5+Full+tutorial+about+new+Authentication+mechanism>
- [48] General ASP.NET development. Retrieved from <http://forums.asp.net/default.aspx/7?General+ASP+NET>
- [49] Introducing Entity Framework. Retrieved from <http://msdn.microsoft.com/en-us/data/aa937723>
- [50] Query Optimizations. Retrieved from <http://blog.farreachinc.com/2013/09/26/entity-framework-query-optimizations/>
- [51] Model Mappings. Retrieved from <http://blog.farreachinc.com/2013/10/10/entity-framework-part-2-model-mappings/>
- [52] Performance Considerations. Retrieved from <http://msdn.microsoft.com/en-us/library/cc853327%28v=vs.110%29.aspx>
- [53] Improving Query Performance using Graph-Based Query. Retrieved from <http://www.codeproject.com/Articles/247254/Improving-Entity-Framework-Query-Performance-Using>
- [54] European University Sports Association. Retrieved from http://www.eusa.eu/en/eusa/association/about_eusa?sso=done
- [55] European University Games. Retrieved from <http://eugames2014.eu/about-the-eugames/>
- [56] Croatian Academic Sports Federation. Retrieved from http://eusagames2016.com/?page_id=15
- [57] International University Sports Federation. Retrieved from <http://www.fisu.net/en/FISU-today-3417.html>
- [58] Ninject Extensions. Retrieved from <http://www.ninject.org/extensions.html>
- [59] Ninject Mini Tutorial part 1 (Stephano Ricciardi). Retrieved from <http://stefanoricciardi.com/2011/01/21/ninject-mini-tutorial-part-1/>
- [60] Ninject Mini Tutorial part 2 (Stephano Ricciardi). Retrieved from

<http://stefanoricciardi.com/2011/02/04/ninject-mini-tutorial-part-2/>

[61] Introduction to Microsoft DreamSpark. Retrieved from

<https://www.dreamspark.com/What-Is-Dreamspark.aspx>

[62] Visual Studio 2013. Retrieved from

<http://msdn.microsoft.com/en-us/library/dd831853.aspx>

[63] Building with Visual Studio. Retrieved from

<http://www.visualstudio.com/get-started/overview-of-get-started-tasks-vs>

[64] Server and Cloud Platform. Retrieved from

<http://www.microsoft.com/en-us/server-cloud/products/sql-server/default.aspx#fbid=j3bZqYVDzkt>