Міністерство освіти і науки України

Дніпропетровський національний університет

ім. Олеся Гончара

Факультет фізики, електроніки і комп’ютерних систем

Кафедра електронних обчислювальних машин

**Курсова робота з дисципліни**

**«Інформаційна стійкість інформаційних систем»**

Виконав студент гр. КІ-14м-1

Мага О.Ю.

Перевірив доц.. каф. ЕОМ

Литвинов О.А.

Дніпропетровськ

2014 р.

**Содержание**

[Содержание 2](#_Toc405737323)

[Вступление 3](#_Toc405737324)

[1. Основная часть 4](#_Toc405737325)

[1.1. Требования 4](#_Toc405737326)

[1.2. Выбор архитектуры 5](#_Toc405737327)

[1.3. Выбор инструментов 5](#_Toc405737328)

[1.4. Организация работы 7](#_Toc405737329)

[2. Реализация 9](#_Toc405737330)

[3. Результаты работы 14](#_Toc405737332)

[Выводы 16](#_Toc405737333)

[Список использованных источников 17](#_Toc405737334)

**Вступление**

В настоящее время устойчивость информационных систем имеет большое значение в процес се жизни и развития любой организации, деятельность короторой так или иначе связана с финансовой деятельностью. Поэтому любой сбой такой ситстемы может стать причиной матеріального ущерба.

С целью предусмотреть, а также как можно скорее устранить такого рода сбои существуют «лучшие практики» по разработке информационных систем. Таких подходов существует довільно много. Самымыи звестнымисреди них являються SCRUM, CMMI и пр.

Данная курсовая работа выполнена с применением практики SCRUM, основных ее этапов и элементов.

**1.Основная часть**

## 1.1.Требования

Система FootballManager предназначена для администраторов и организаторов футбольных турниров. Администратор может создавать новый турнир, добавлять в него существующие клубы. Система может также использоваться владельцами клубов для добавление новых игроков в свой клуб, осуществлять трансферы.

Use Case 1. Регистрация.

Базовый сценарий

1.Пользователь входит на сайт.

2.Пользователь регистрируется

3.Система сохраняет его данные в базу.

4.Пользователь автоматически авторизуется.

Альтернативный сценарий

2.1.Введенные данных некорректны.

2.1.2.Система возвращает ошибку.

2.1.3.Предложение ввести новые данные.

Use Case 2. Создание нового клуба.

Pre-condition: пользователь должен авторизоваться чтобы стать админом.

Базовый сценарий

1.Администратор нажимает кнопку создать новый клуб.

2.Система предоставляет форму для создания.

3.Админ вводит данные, и сохраняет.

4.Система заносит данные в базу.

Use Case 3. Удаление клуба.

Базовый сценарий

1.Админ нажимает кнопку удалить турнир.

2.Система выдает окно, с подтверждением удаления.

3.Система удаляет турнир из базы. Возвращает главную страницу.

Альтернативный сценарий

1.Админ отменяет предложение удаления.

2.Система перенаправляет на главную страницу.

**Основные требования**

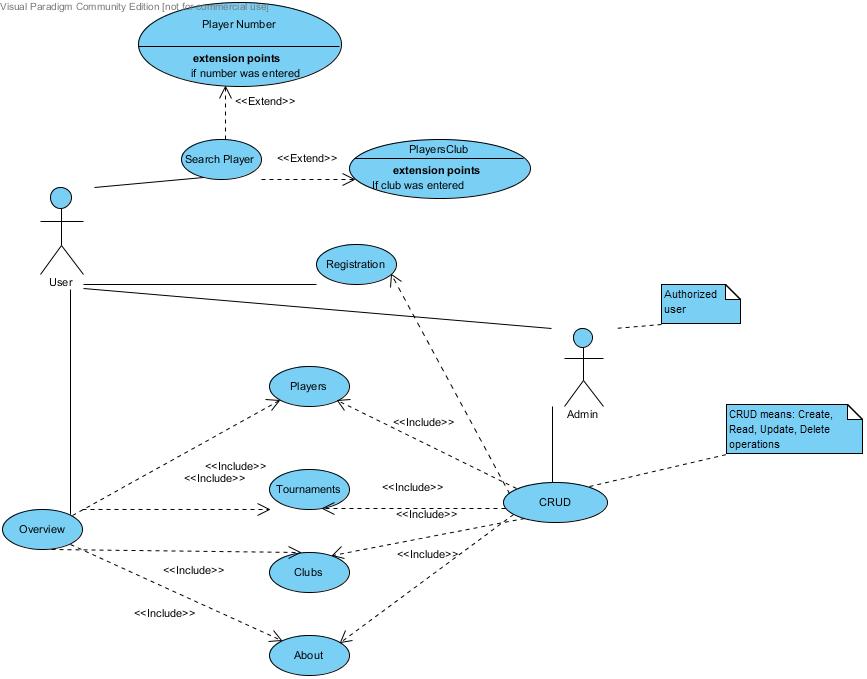
1. Поддерживаются CRUD – операции для всех типов сущностей, а именно турниры, клубы и игроки.
2. Система должна предоставлять удобный поиск игроков с фильтрами клубов и по номеру игрока.
3. Приложение должно иметь страницу о создателях и их контактные данные.
4. Система должна поддерживать авторизацию для пользователей.

Рис.1 Use-case диаграмма системы Footballmanager

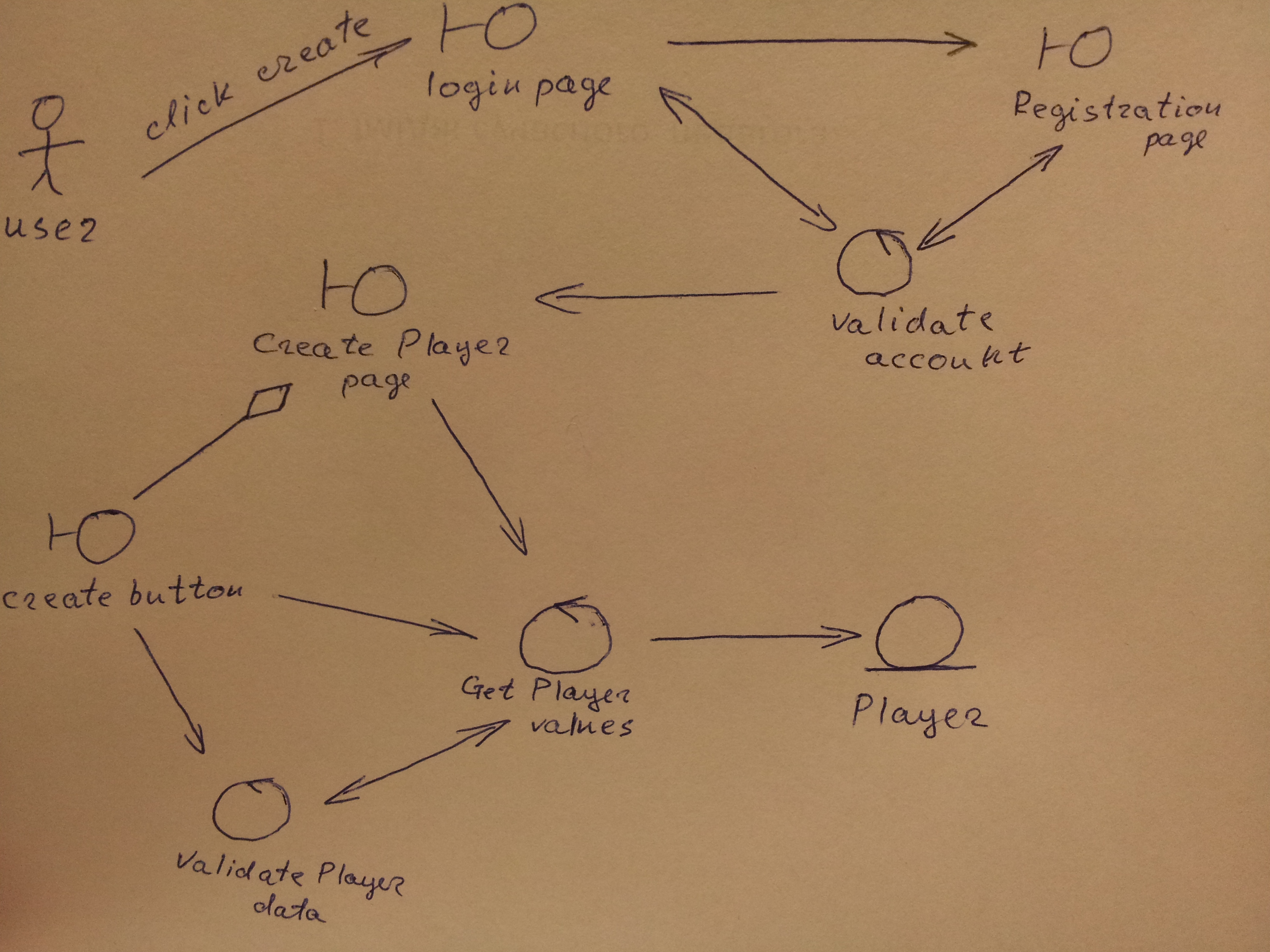


Рис. 2. Robustness diagram

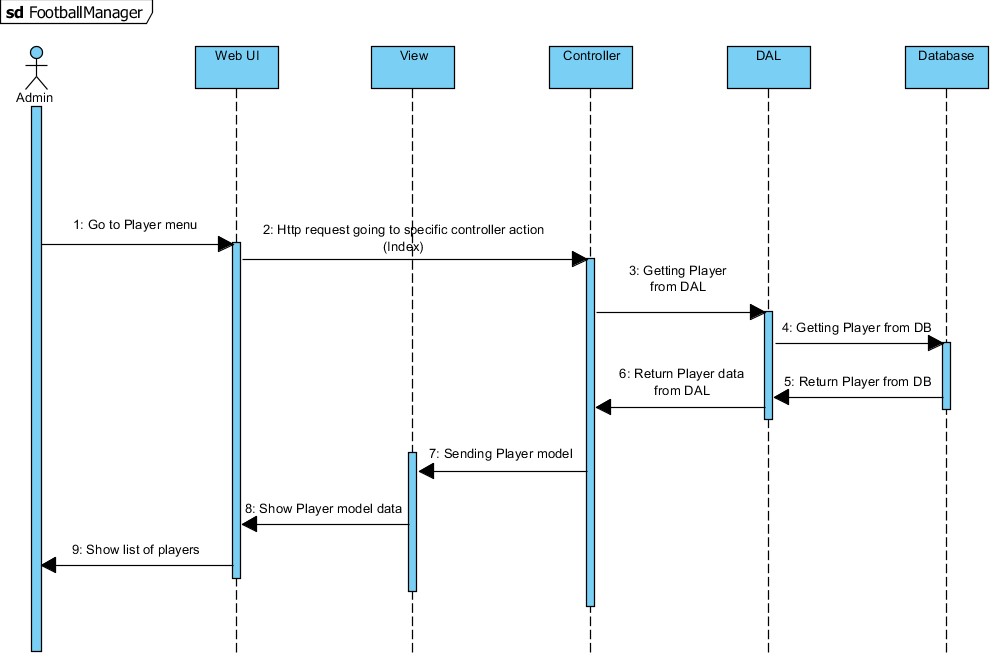


Рис. 3. Диаграмма последовательности

## 1.2.Выбор архитектуры

Для реализации поставленных требований, была выбрана двух уровневая архитектура с уровнями Web и DAL.

Уровень Web будет представлять из себя веб-приложение с удобным пользовательским интерфейсом.Уровень DAL будет обслуживать операции доступа и возврата данных.

## 1.3.Выбор инструментов

Для решение поставленой задачи, были выбраны следующие инструменты и технологи:

1. Для разработки была выбрана среда Microsoft Vusial Studio 2012, и язык C# .NET, технология ASP .NET MVC 4.
2. Для хранения даннях выбран сервер MSSQL localdb.
3. Для доступа к данням будет использоваться ORM средство Entity Framework 6.0.
4. Для совместной работы над проектом используется система контроля версий Git.
5. Для таск-трекинга был использован ресурс <https://football-manager.visualstudio.com/>
6. Для модульного тестирования MSUnit Framework , library Moq.

При проектирования уровня DAL была предусмотрена возможность использования различных баз данных. Поэтому был примене паттерн репозитория для работы с сущностями [<http://msdn.microsoft.com/en-us/library/ff649690.aspx>].

public interface IRepository<T> {

    IEnumerable<T> FindAll();

    IEnumerable<T> FindBy(Expression<Func<T, bool>> predicate);

    T FindById(int id);

    void Add(T newEntity);

    void Remove(T entity);}

Предполагалось что база данных будет использваться и другими приложениями со своими таблицами, поэтому был использован паттерн

Unit of work [<http://design-pattern.ru/patterns/unit-of-work.html>].

public interface IUnitOfWork {

    IRepository<Tournament> Employees { get; }

    IRepository<Club> Orders { get; }

    IRepository<Player> Customers { get; }

    void Commit();

}

Для инверсии зависимостей был использован IoC – контейнер Ninject.

При создании приложения был принят поход Database first [<http://msdn.microsoft.com/ru-ru/data/jj206878>] При котором классы моделей генерируются из существвующей базы данных(рис. 4)

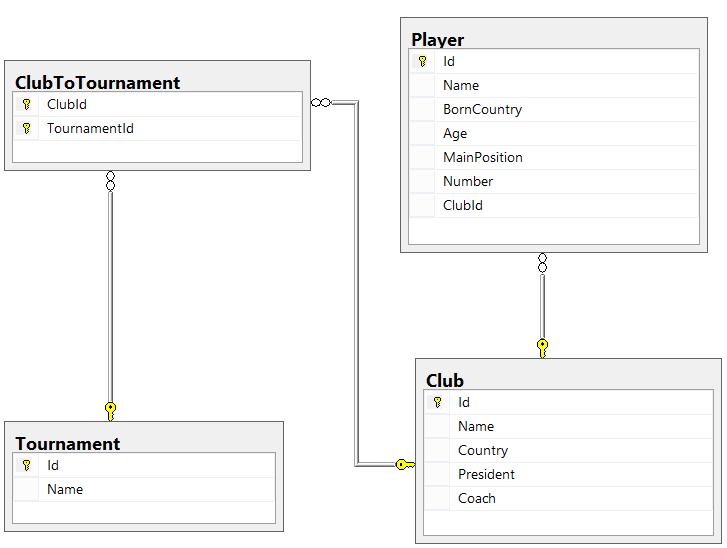


Рис. 4 Схема базы-данных.

Исходя из данных построена диаграмма компонентов (рис.5)

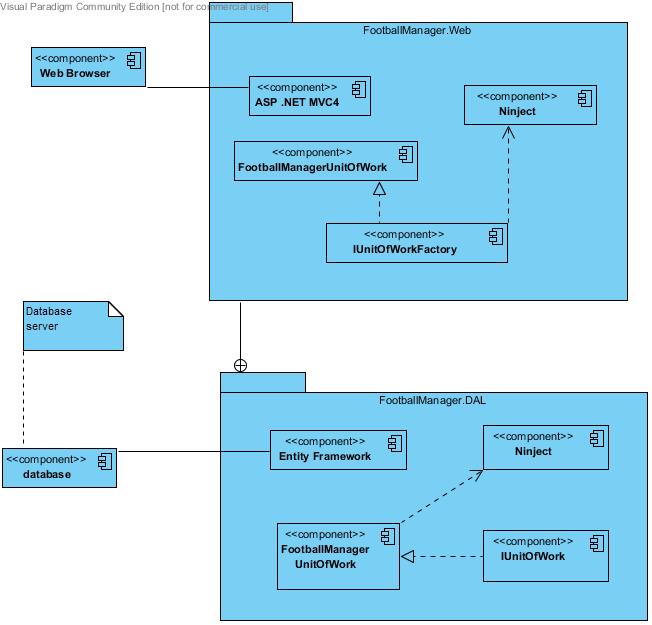


Рис.5 Диаграмма компонентов.

## 1.4. Организация работы

Согласно Scrum[<http://ru.wikipedia.org/wiki/Scrum>] был составлен Product Backlog.(рис.6)

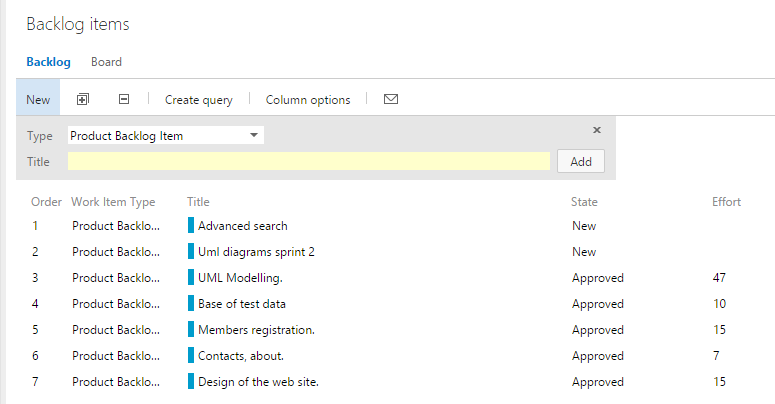


Рис. 6 Product Backlog

Работу планировалось осуществить за 2 спринта по 1 неделе.

На первый спринт были отобраны такие user story (рис. 7), на второй (рис. 8)

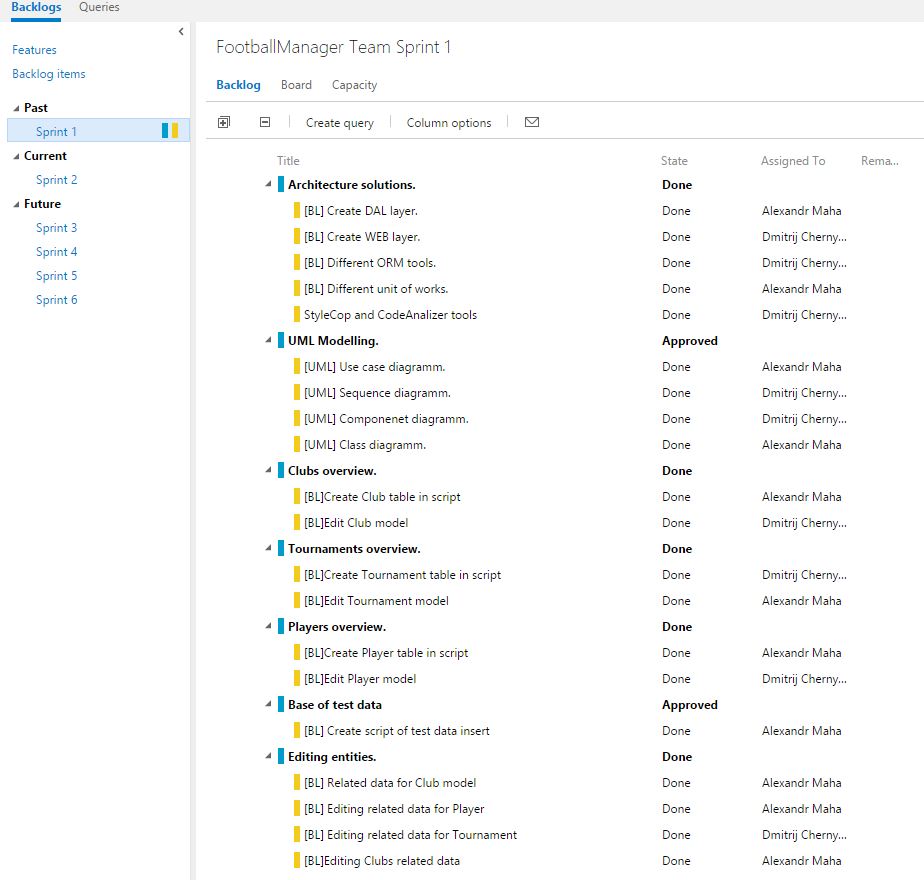


Рис.7 Sprint 1 backlog.

Burndown for sprint 1.

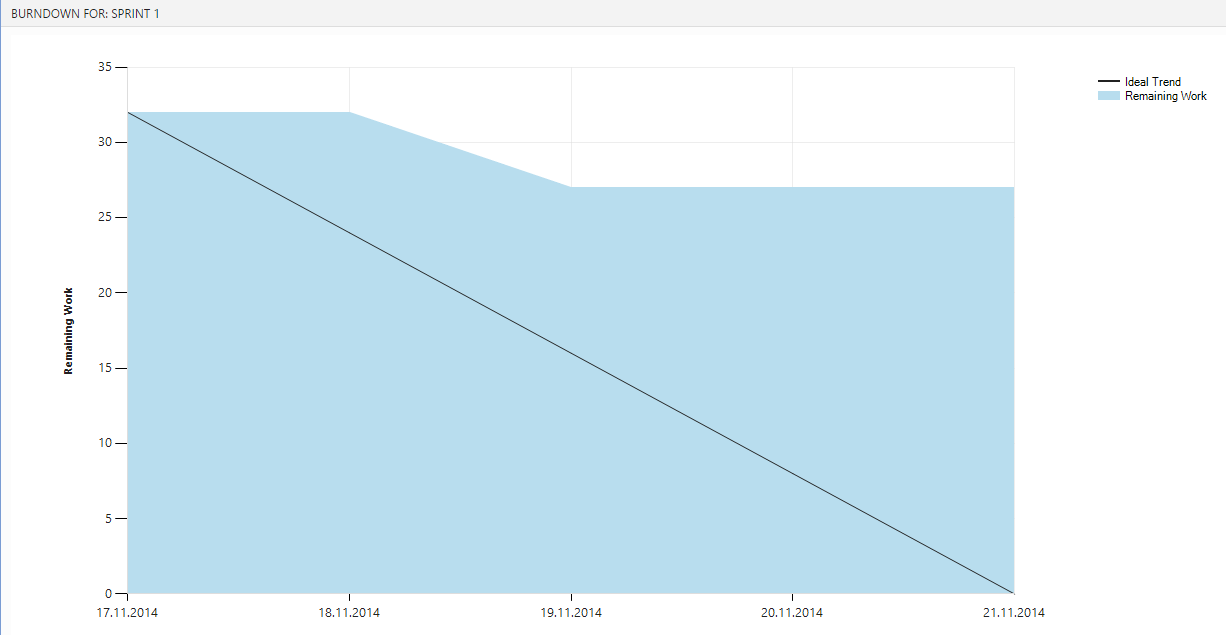


Рис. 8. Burndown chart for sprint 1

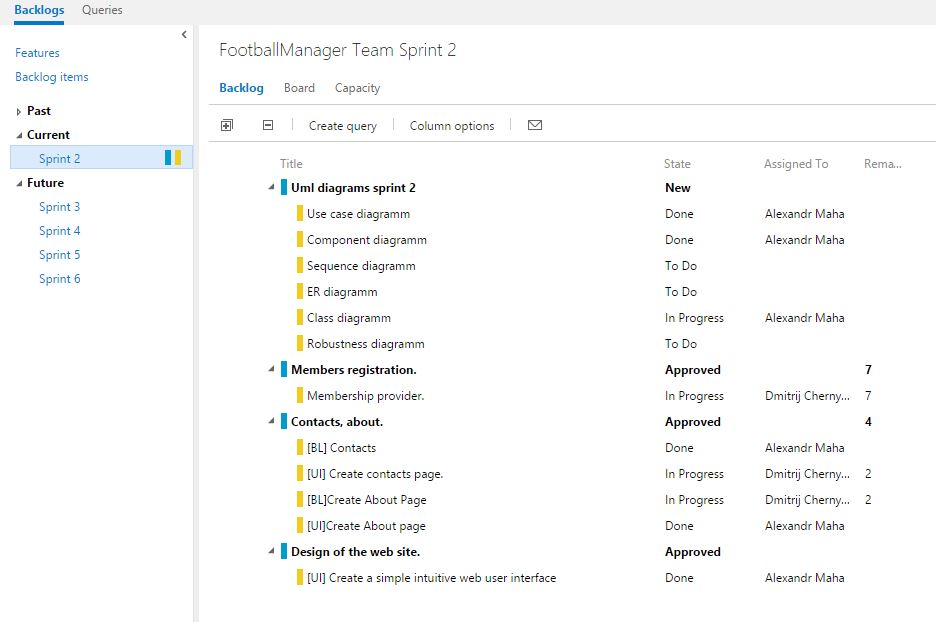


Рис.9 Sprint 2 backlog.

Burndown for sprint 2.

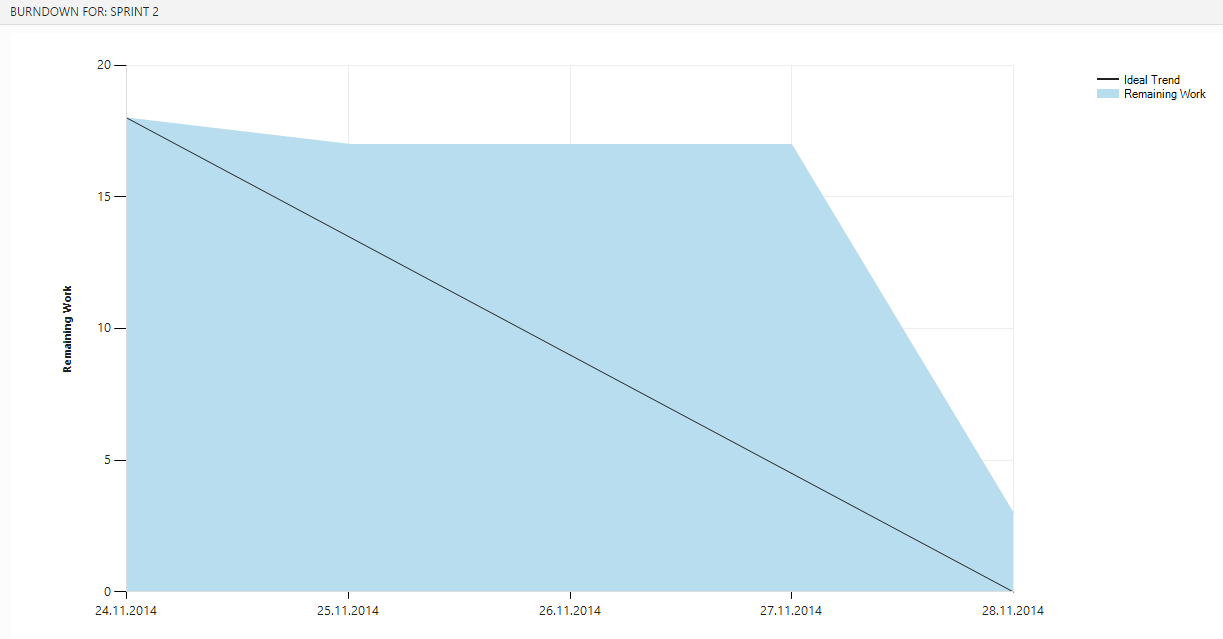


Рис. 10. Burndown chart for sprint 2

Таким образом, судя по графикам Burndown-а, можно видеть, что работа, запланированная на каждый спринт, выполнялось дольше, чем ожидалось, но все же была выполнена в сроки.

# 2. Реализация

3.Так как используется подход Database first. Необходимо написать скрипт создание и заполнения таблицы тестовыми данными.

Листинг 1. Скрипт создания таблицы:

CREATE TABLE dbo.Club (

Id int identity(1, 1) NOT NULL

CONSTRAINT PK\_Club\_Id PRIMARY KEY CLUSTERED,

Name varchar(60) NOT NULL

CONSTRAINT Club\_Name UNIQUE(Name),

Country varchar(30) NOT NULL,

President varchar(60) NOT NULL,

Coach varchar(60) NOT NULL,

);

GO

Листинг 2. Скрипт заполнения тестовыми данными:

SET IDENTITY\_INSERT dbo.Club ON;

INSERT INTO dbo.Club (

[Id], [Name],[Country],[President],[Coach]

) VALUES

(1, 'Read Madrid','Spain','Florentino Peres','Carlo Ancelotti'),

(2, 'Barcelona','Spain','Joseph Bartomeu','Louis Enrique'),

(3,'Chelsea','England','Roman Abramovich','Jose Mourinho'),

(4,'Bayern Munchen','Germany','Karl Hophner', 'Josep Guardiola'),

(5,'Milan AC','Italy','Silvio Berluskoni','Filippo Indzaggi')

GO

SET IDENTITY\_INSERT dbo.Club OFF;

Далее были сгенерированы классы моделей.

Листинг 3. Код класса Club.

[Table("Club")]

public partial class Club {

public Club() {

this.Players = new HashSet<Player>();

this.Tournaments = new HashSet<Tournament>();

}

public int Id { get; set; }

[Required]

[StringLength(60)]

public string Name { get; set; }

[Required]

[StringLength(30)]

public string Country { get; set; }

[Required]

[StringLength(60)]

public string President { get; set; }

[Required]

[StringLength(60)]

public string Coach { get; set; }

public virtual ICollection<Player> Players { get; set; }

public virtual ICollection<Tournament> Tournaments { get; set; }

}

За главную страницу сайта отвечает HomeController и его Index View.

Было решено отображать на ней список всех игроков, и левое меню с клубами.

Листинг 4. Код класса HomeController

public class HomeController : Controller

{

private readonly IFootballManagerUnitOfWork unitOfWork;

public HomeController(IUnitOfWorkFactory unitOfWorkFactory)

{

this.unitOfWork = unitOfWorkFactory.CreateFootballManagerUnitOfWork();

}

public ActionResult Index()

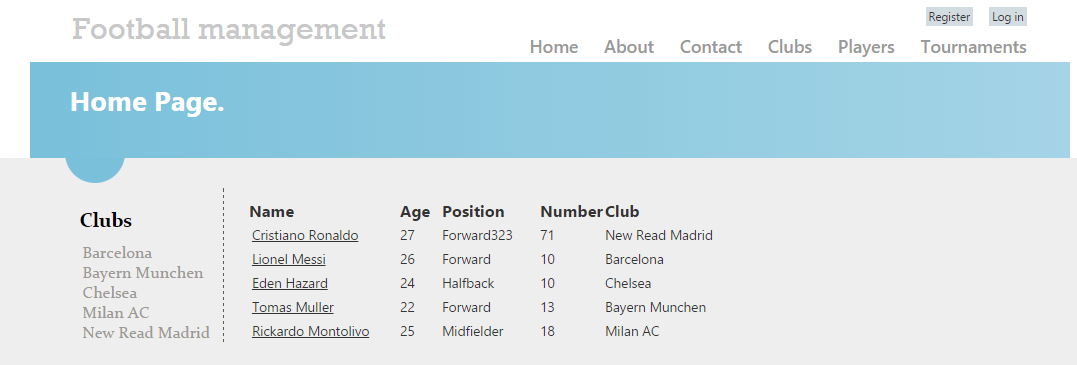
{

var players = unitOfWork.Players.FindAll().Include(p=>p.Club).ToList();

return View(players);

}

Главная страница сайта:



Пример редактирования существующего клуба.

Пользователь нажимает кнопку редактировать, отображается Edit View.

Листинг 1. Код Clubs Edit view.

@model SoftServe.FootballManager.DAL.Models.Club

@{

ViewBag.Title = "Edit";

}

<h2>Edit</h2>

@using (Html.BeginForm()) {

@Html.ValidationSummary(true)

<fieldset>

<legend>Club</legend>

@Html.HiddenFor(model => model.Id)

@Html.Partial("\_CreateOrEdit", Model)

<p>

<input type="submit" value="Save" />

</p>

</fieldset>

}

<div>

@Html.ActionLink("Back to List", "Index")

</div>

И пользвателю приходит \_CreateOrEdit View, с полями для ввода новых значений. Дальше вызывается метод [HttpPost]Edit в контроллере TournamentsController, после чего обновляются данные.

Листинг 2. Метод Edit контроллера Clubs.

// POST: /Clubs/Edit/5

[HttpPost]

public ActionResult Edit(int id, string[] selectedTournaments)

{

Club club = unitOfWork.Clubs.FindWhere(x => x.Id == id).Single();

if (TryUpdateModel(club, "",

new string[] { "Name", "Clubs" }))

{

try

{

UpdateClubTournaments(selectedTournaments, club);

unitOfWork.Clubs.Update(club);

unitOfWork.Commit();

return RedirectToAction("Index");

}

catch (DataException)

{

ModelState.AddModelError("", "Unable to save changes. Try again, and if the problem persists see your system administrator.");

}

}

PopulateAssignedTournaments(club);

return View(club);

}

И происходит переадресация на страницу Index, где пользователь может видеть клуб с новыми данными которыми он ввел. Контроллер обращается к unitOfwork тот в свою очередь к репозиторию Tournaments, где вызывается метод Update

Листинг 3. Метод Update класса EfRepository.

public void Update(T updatedEntity)

{

this.objectSet.Attach(updatedEntity);

this.objectSet.Context.ObjectStateManager.ChangeObjectState(updatedEntity, EntityState.Modified);

}

Модульный тест на метод Update.

Листинг 4. Код Unit теста.

[TestMethod]

public void Update\_NewDataForExistingTournament\_TournamentDataChanged()

{

var repositoryMock = new Mock<IRepository<Tournament>>();

Tournament oldTournament = new Tournament { Id = 1, Name = "Tournament 1" };

Tournament expectedTournament = new Tournament { Id = 2, Name = "Tournament" };

List<Tournament> tournaments = new List<Tournament>();

repositoryMock.Setup(r => r.Update(It.IsAny<Tournament>()))

.Callback((Tournament tournament) =>

{

oldTournament.Id = tournament.Id;

oldTournament.Name = tournament.Name;

});

IRepository<Tournament> tournamentRepository = repositoryMock.Object;

tournamentRepository.Update(expectedTournament);

Assert.AreEqual(oldTournament.Id, expectedTournament.Id);

}

# 3.Результаты работы

После авторизации, пользователь становится админом, заходит на страницу клубов, по адресу <http://localhost:50457/Clubs>. (рис. 7)

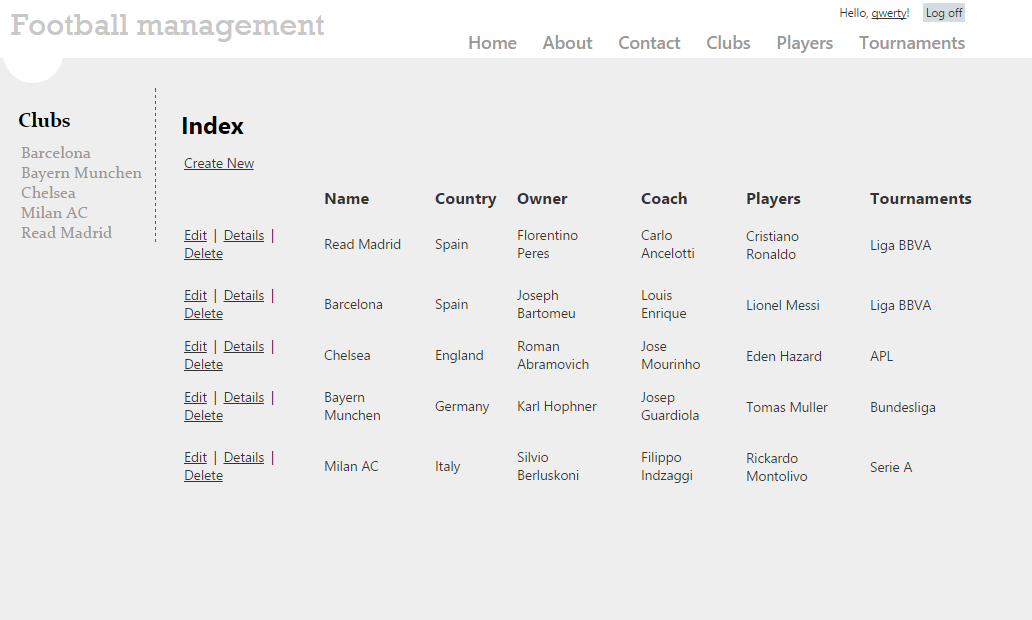


Рис.7 Главная страница клубов.

После нажатия кнопки Edit.(рис.8)

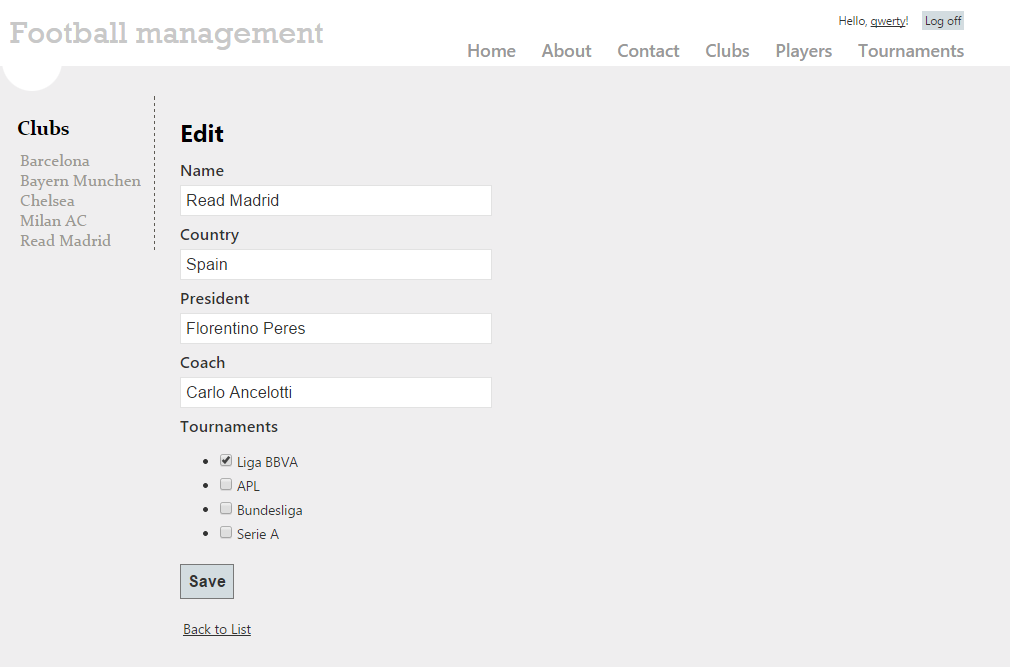


Рис.8 Страница редактирования клуба.

После введения новых значений(рис.9)

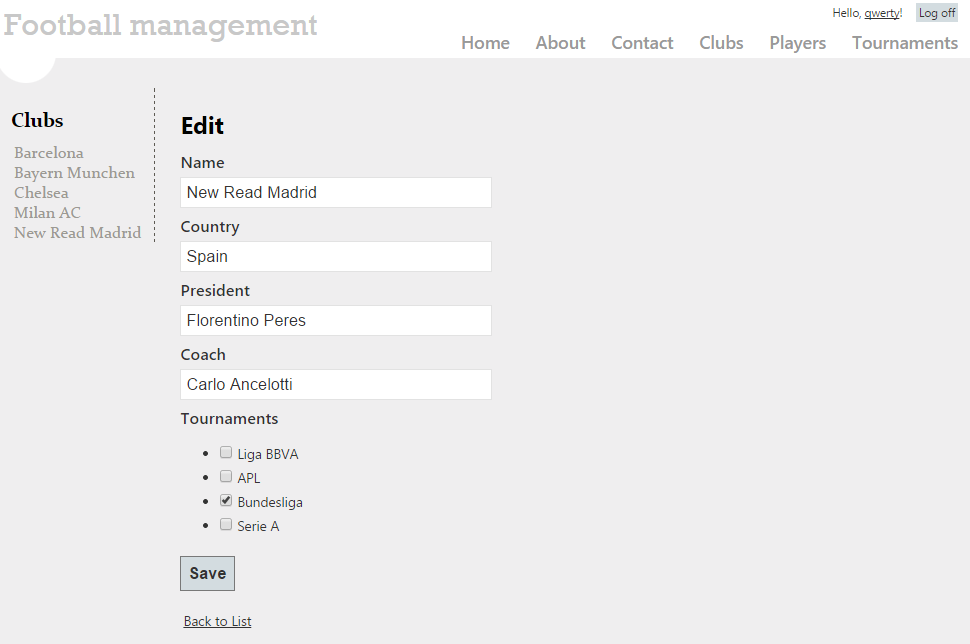


Рис.9 Новые значения для клуба.

После перенаправления на главную страницу клуба, видим изменившуюся информацию по клубу (рис.10)

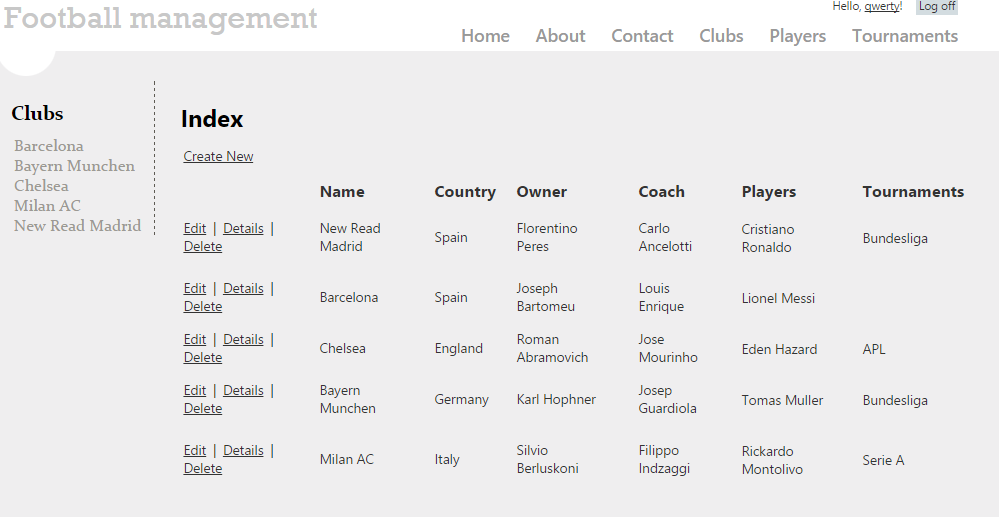


Рис.10 Клуб после редактирования.

# Выводы

В ходе выполнения данной курсовой работы создано полноценное веб-приложение по технологии командной разработки SCRUM. Данная техника позволила участвовать одновременно нескольким людям над выполнением проекта. Была проделана работа по планированию, созданы требования, которые потом разбились на конкретные задачи (с ограничениями по времени) и были назначены конкретным людям. В процессе выполнения был закреплен навык работы в команде.

В практической части курсовой работы представлен, на примерах, процесс разработки и построения приложения. В пункте реализация показан процесс взаимодействия пользователя с системой: поиск с использованием фильтров, создание нового игрока и его редактирование.

# Список использованных источников

# 1. <http://msdn.microsoft.com/en-us/library/ff649690.aspx> - The Repository Pattern

# 2. <http://design-pattern.ru/patterns/unit-of-work.html> - Unit of Work (Единица работы)

3. <http://msdn.microsoft.com/ru-ru/data/jj206878> - Data access and storage.

# 4. Adam Freeman - Pro ASP.NET MVC 5 (Expert's Voice in ASP.Net) Paperback – December 20, 2013

# 5. [Brian Driscoll](http://www.amazon.com/Brian-Driscoll/e/B00J1SF8ZY/ref=dp_byline_cont_book_1), [Nitin Gupta](http://www.amazon.com/s/ref=dp_byline_sr_book_2?ie=UTF8&field-author=Nitin+Gupta&search-alias=books&text=Nitin+Gupta&sort=relevancerank) - Entity Framework 6 Recipes Paperback – October 29, 2013