Федеральное государственное бюджетное образовательное учреждение высшего образования

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

**Кафедра информационных систем и программной инженерии**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту по дисциплине   
"Технологии программирования"**

**на тему**

*Проектирование и разработка программной системы   
учёта знаний учащихся*

Выполнил: ст. гр. ИСТ-116

Сильянов С.О.

Принял: доц. кафедры ИСПИ

Вершинин В. В.

Владимир, 2018

**Аннотация**

Суть курсовой работы заключается в разработке программной системы учета знаний учащихся. Работа состоит из графической части и пояснительной записки. Графическая часть представляет собой 3 чертежа, выполненные на разных листах формата А4 в масштабе 1:8.

Пояснительная записка содержит:

* 20 страниц;
* 13 рисунков.

The essence of the course work is the development of a software system for accounting of students’ knowledge. The work consists of a graphic part and an explanatory note. Graphical part is 3 drawings, made on different sheets of A4 format on a scale of 1: 8.

The explanatory memorandum contains:

• 20 pages;

• 13 figures.

**Содержание**

[1. ВВЕДЕНИЕ 2](#_Toc501697091)

[2. Постановка задачи 3](#_Toc501697092)

[3. Описание предметной области 4](#_Toc501697093)

[3.1. Общие сведения о предмете автоматизации 4](#_Toc501697094)

[3.2. Словарь предметной области 4](#_Toc501697095)

[3.3. Сценарий взаимодействия пользователя с системой 5](#_Toc501697096)

[4. проектирование системы 7](#_Toc501697097)

[4.1 Диаграмма прецедентов 7](#_Toc501697098)

[4.2 Диаграмма классов 9](#_Toc501697099)

[4.3 Диаграмма состояний 10](#_Toc501697100)

[4.4 Диаграмма последовательности 11](#_Toc501697101)

[4.5 Структура базы данных 12](#_Toc501697102)

[5. реализация системы 14](#_Toc501697104)

[5.1 Принципы организации проекта 14](#_Toc501697105)

[5.2 Реализация модели 14](#_Toc501697106)

[5.3 Реализация поддержки различных типов пользователей 14](#_Toc501697107)

[5.4 Реализация поддержки различных справочников 15](#_Toc501697108)

[5.5 Реализация ведомости 15](#_Toc501697109)

[5.6 Пользовательский интерфейс 15](#_Toc501697110)

[6. заключение 19](#_Toc501697111)

[7. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 20](#_Toc501697112)

[ПРИЛОЖЕНИЕ А Исходный код программной системы 21](#_Toc501697113)

[ПРИЛОЖЕНИЕ Б Структура и Скрипты создания таблиц БД 35](#_Toc501697114)

1. ВВЕДЕНИЕ

В наше время уже сложно представить жизнь без информационных технологий. Информационные технологии проникли во все сферы деятельности. Поэтому в наши дни является очень актуальным создание программных продуктов, в особенности таких, как веб – приложение.

Целью данного курсового проекта является разработка программной системы учета знаний учащихся.

Предпосылки для создания информационной системы учета знаний учащихся диктуются высокими требованиями к учебному процессу, необходимостью быстрого и эффективного сбора данных об успеваемости студентов в любой отрезок учебного процесса, уменьшения нагрузки на деканаты и достижения большей эффективности в работе всей инфраструктуры института.

информационная система учет успеваемость

1. Постановка задачи

Необходимо разработать программное обеспечение, ведущее учет знаний учащихся.

Для решения поставленной задачи, необходимо последовательно выполнить этапы анализа, проектирования и реализации системы.

Анализ предметной области. Результатом данного этапа работы является описание процесса учета знаний учащихся и словарь предметной области.

Проектирование системы. Для проектирования системы необходимо использовать язык UML и включить следующие UML-диаграммы:

- диаграмма прецедентов;

- диаграмма классов;

- диаграмма состояний;

- диаграмма последовательностей.

Также этап проектирования включает схему базы данных в виде ER-диаграммы логической моделей данных.

Программная реализация.

Минимальный набор функций:

1. создание, заполнение и просмотр ведомости;
2. расчет рейтинга;
3. закрытие ведомости;
4. работа со справочниками.

Система должна быть реализована на языке C# в виде веб-приложения с использованием технологий ASP .NET MVC.

1. Описание предметной области
   1. Общие сведения о предмете автоматизации

Основной целью разработки программной системы учёта знаний учащихся было упрощение ведения учебной документации и системы контроля.

Система учёта знаний учащихся представляет собой некоторый вид электронной ведомости, которую заполняет преподаватель, а студент может просматривать свои баллы в течение семестра. Минимально необходимый набор функций: создание ведомости; заполнение ведомости; редактирование ведомости; просмотр ведомости; расчёт рейтинга; закрытие ведомости (в конце семестра); работа со справочниками.

* 1. Словарь предметной области

Преподаватель – должность в высшем учебном заведении. На преподавателей возлагается проведение лабораторных занятий, рейтинг- контролей, а также приём экзамена;

Студент – учащийся высшего учебного заведения. Студенты посещают лекции, участвуют в семинарах, после чего сдают лабораторные работы, рейтинг контроли, сдают экзамен;

Администратор – сотрудник высшего учебного заведения, занимающийся управлением основными сущностями высшего учебного заведения;

Экзамен – итоговая форма оценки знаний учащихся;

Балл – единица оценки знаний;

Ведомость – тип документа в котором проставляются оценки студентов за выполнение заданий в течение семестра;

Семестр - учебное полугодие в высших или специальных средних учебных заведениях;

Рейтинг - оперативный показатель успеваемости. Позволяет сравнить образовательные результаты студентов, полученные ими в течение семестра;

Лабораторная работа - особый вид индивидуальных работ, в ходе которых учащиеся используют теоретические знания на практике, применяют различный инструментарий и прибегают к помощи технических средств;

Роль – привилегии пользователя.

* 1. Сценарий взаимодействия пользователя с системой

В приложении используется разделение по ролям. В соответствии с должностями, были получены следующие роли:

* администратор;
* преподаватель;
* студент.

Пользователь с ролью Администратор может:

* создать, изменить, удалить и просмотреть информацию о предметах;
* создать, изменить, удалить и просмотреть информацию о группах;
* изменить и просмотреть информацию о пользователе;
* создать, изменить и просмотреть шаблон ведомости.

Пользователь с ролью Преподаватель может:

* создать, изменить и просмотреть шаблон ведомости;
* создать, изменить и просмотреть ведомость.

Пользователь с ролью Студент может:

* просмотреть шаблон ведомости;
* просмотреть ведомость.

Незарегистрированный пользователь может зарегистрироваться в системе.

Также, каждый зарегистрированный пользователь имеет возможность авторизоваться в системе и выйти из своего аккаунта.

1. проектирование системы
   1. Диаграмма прецедентов

Действующими лицами (в нотации UML – актерами) являются: Администратор, Преподаватель, Студент. Ниже приведен перечень и расширенное описание некоторых прецедентов в проектируемой системе.

Название: создать ведомость;

Актёры: администратор;

Предусловие: нет

Основной поток событий:

1. Актёр открывает форму «Создание ведомости»;
2. Актёр заполняет необходимые поля;
3. Актёр нажимает кнопку «Сохранить».

Альтернативный поток событий:

1. Актёр нажимает «Назад». Закрывается форма «Создание ведомости»; открывается главная форма;

Постусловие: ведомость успешно создана.

Название: заполнить ведомость;

Актёры: преподаватель;

Предусловие: нет

Основной поток событий:

1. Актёр открывает форму «Заполнение ведомости»;
2. Заполняет необходимые поля;
3. Нажимает кнопку «Сохранить».

Альтернативный поток событий:

1. Актёр нажимает «Назад». Закрывается форма «Заполнение ведомости»; открывается главная форма;

Постусловие: ведомость успешно заполнена.

Название: редактировать ведомость;

Актёры: преподаватель;

Предусловие: нет

Основной поток событий:

1. Актёр открывает форму «Редактирование ведомости»;
2. Заполняет необходимые поля;
3. Нажимает кнопку «Сохранить».

Альтернативный поток событий:

1. Актёр нажимает «Назад». Закрывается форма «Заполнение ведомости»; открывается главная форма;

Постусловие: ведомость успешно отредактирована.

Диаграмма прецедентов представлена на рисунке 1.

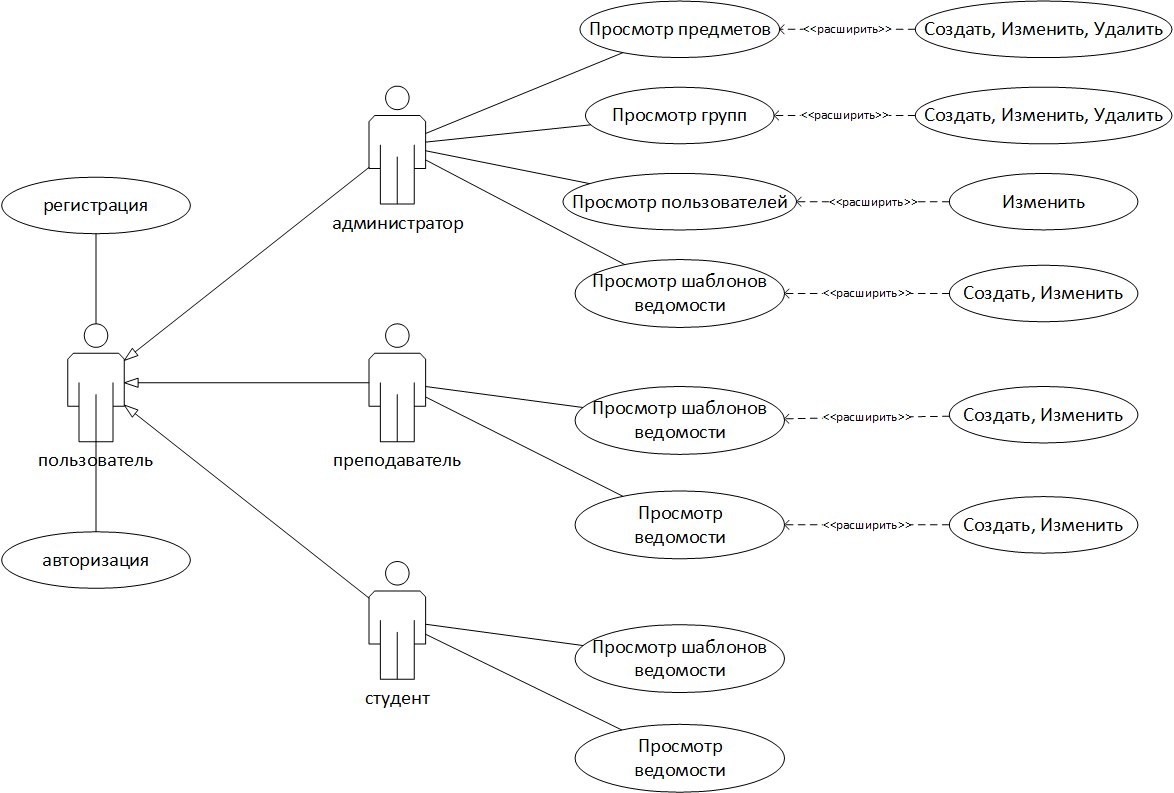


Рисунок 1 – Диаграмма прецедентов.

* 1. Диаграмма классов

Для проектирования структур и моделей данных программной системы, была разработана диаграмма классов. Диаграмма классов представлена на рисунке 2.

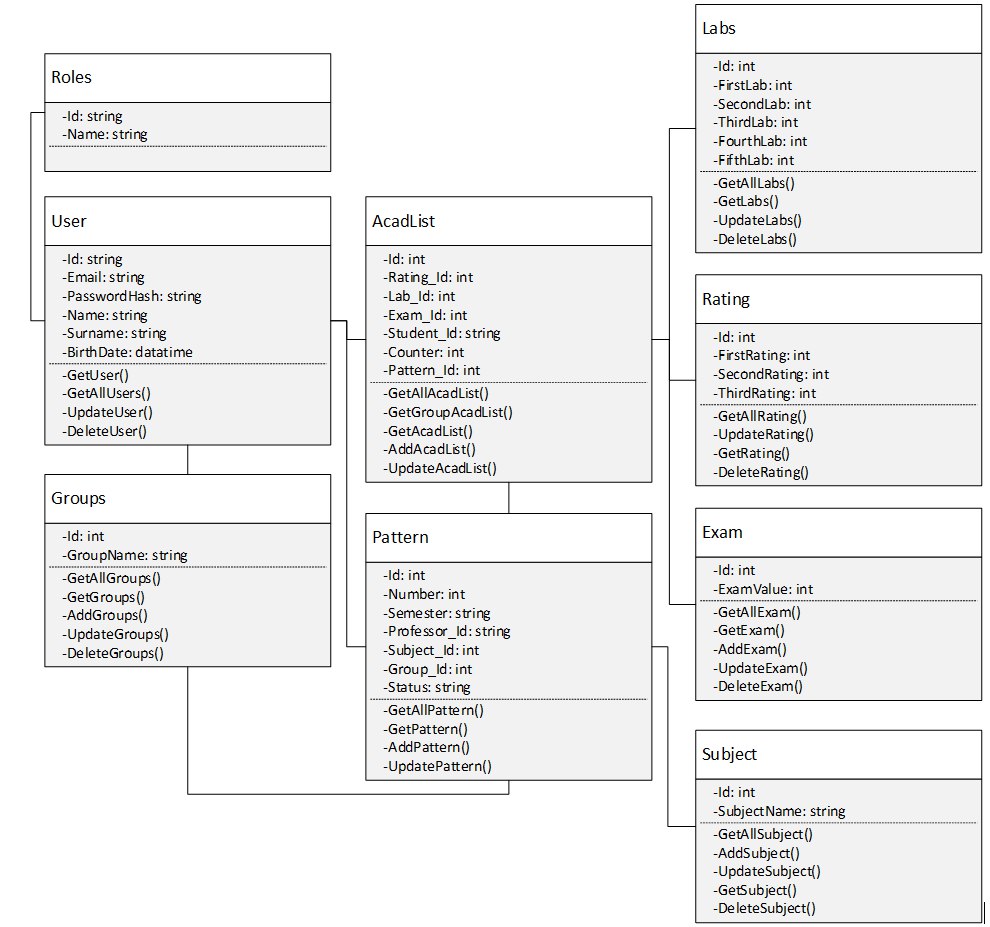


Рисунок 2 – Диаграмма классов.

В результате анализа были выделены следующие сущности.

* Пользователь (User)
* Роль (Roles)
* Группа (Groups)
* Ведомость (AcadList)
* Шаблон ведомости (Pattern)
* Предмет (Subject)
* Лабораторная работа (Labs)
* Рейтинг контроль (Rating)
* Экзамен (Exam)
  1. Диаграмма состояний

Ключевым объектом данной системы является шаблон ведомости, поэтому для него важно отслеживание состояния.

Состояния объекта «Шаблон ведомости»

1. Администратор создаёт шаблон ведомости и присваивает ему статус «черновик»
2. В начале семестра преподаватель или администратор переводят ведомость в статус «открыта»
3. В конце семестра преподаватель или администратор переводят ведомость в статус «закрыта»

Диаграмма состояний объекта «Шаблон ведомости» представлена на рисунке 3.



Рисунок 3 - Диаграмма состояний объекта «Шаблон ведомости».

* 1. Диаграмма последовательности

Диаграмма последовательности разработана для прецедента «Создать ведомость». Создание ведомости предполагает создание шаблона, а затем только создание ведомости.

Диаграмма последовательностей представлена на рисунке 4.

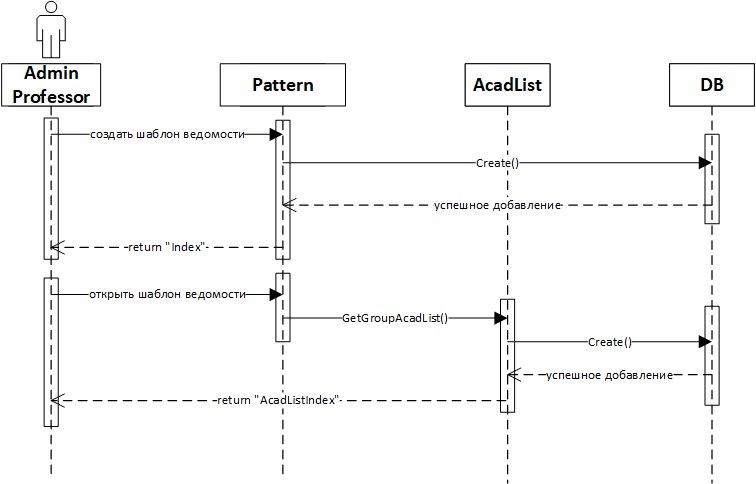


Рисунок 4 – Диаграмма последовательностей для прецедента «Создать ведомость».

* 1. Структура базы данных

Разработанная физическая структура базы данных является нормализованной к третьей нормальной форме. Схема физической модели базы данных приведена на рисунке 5. Скрипты создания таблиц базы данных приведены в приложении Б.

Рисунок 5 – Физическая ER-диаграмма схемы базы данных

1. реализация системы
   1. Принципы организации проекта

Разрабатываемая система построено на архитектуре MVC с использованием технологии Entity Framework, языка интегрированных запросов LINQ. Система представляет собой веб-приложение и, таким образом, было разделено на три части: контроллер, модель, представление.

Представления построены с использованием Razor – способа преобразования кода на странице в HTML. Исходный код веб-сервиса приведен в приложении В.

* 1. Реализация модели

Сначала была создана база данных, состоящая из 11 таблиц, 3 из них, являются системными таблицами, используемыми для работы с ролями и пользователями. На основе всех таблиц была создана модель с использованием технологии ADO.NET Entity Framework.

* 1. Реализация поддержки различных типов пользователей

Для реализации регистрации, аутентификации и авторизации пользователей были использованы стандартные методы шаблона Microsoft Identity.

Методы и представления имеют ограниченный доступ. Пользователям, в зависимости от роли, доступны различные функции. Это обеспечивается ограничением кнопок меню и кнопок представлений по ролям. Администратору присвоена роль “Admin”, преподавателю присвоена роль “Professor”, студент – “Student”. При регистрации система присваивает пользователю роль “Student”. Для незарегистрированного пользователя доступна только регистрация.

* 1. Реализация поддержки различных справочников

Администратор, преподаватель и студент имеют возможность работы с базой данных. Пользователи, в зависимости от роли, могут по-разному работать с каталогами предметов, групп, пользователей, а также с ведомостью.

* 1. Реализация ведомости

Администратор при выборе из меню кнопки «Ведомость» попадает на страницу создания шаблона ведомости. Когда шаблон создан можно перейти на страницу просмотра ведомости и создать её.

После создания ведомости в ней отображается список студентов и их баллы. Преподаватель может проставить баллы для каждого студента.

* 1. Пользовательский интерфейс

Пользовательский интерфейс представлен на Рисунках 6-14.

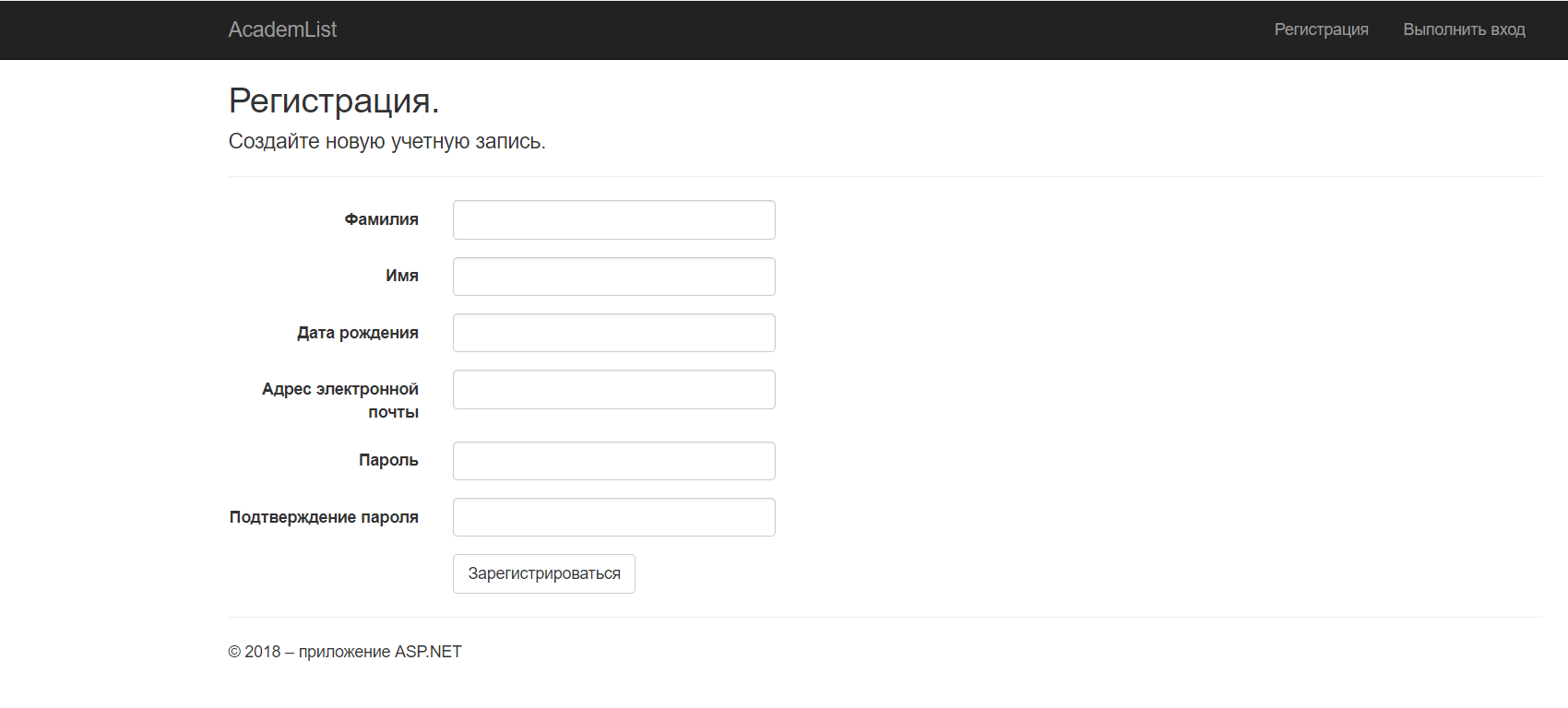


Рисунок 6 – Страница регистрации.

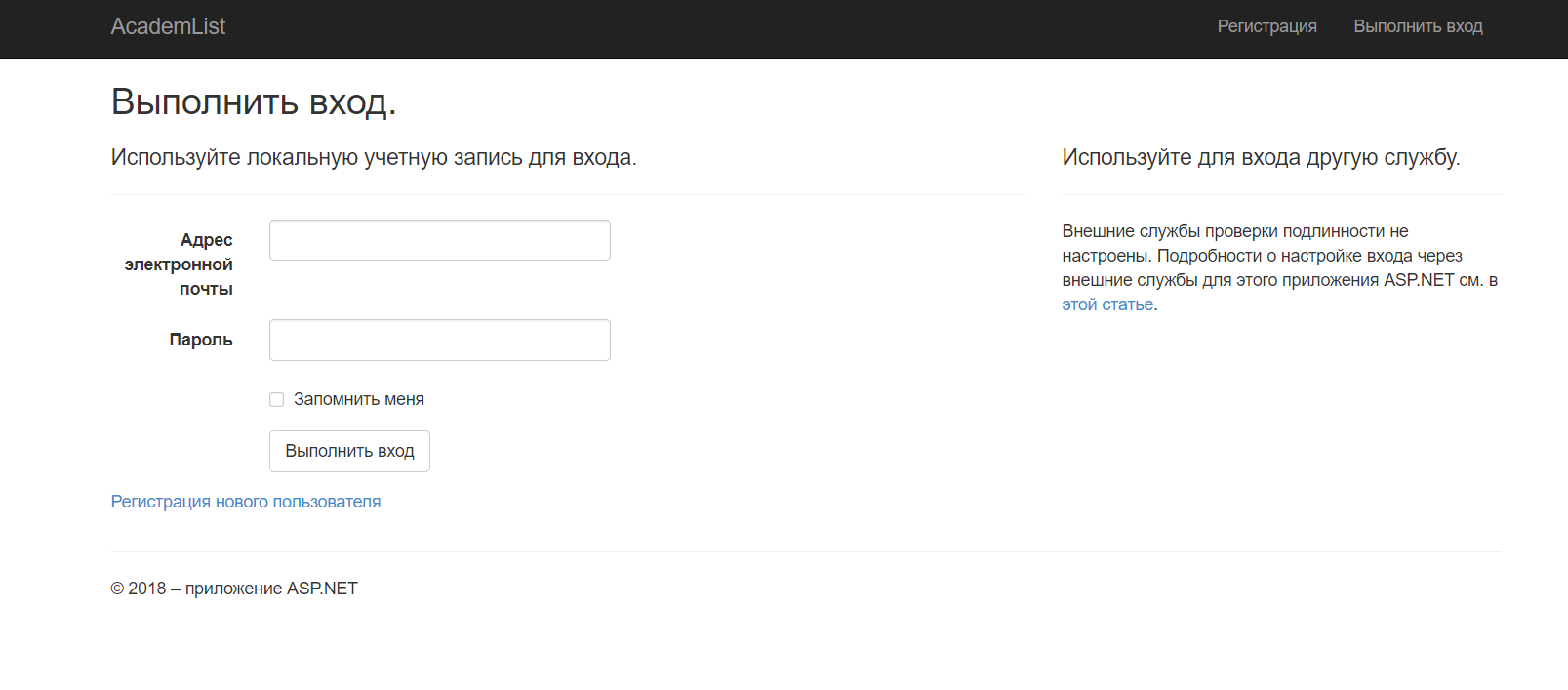


Рисунок 7 – Страница авторизации.

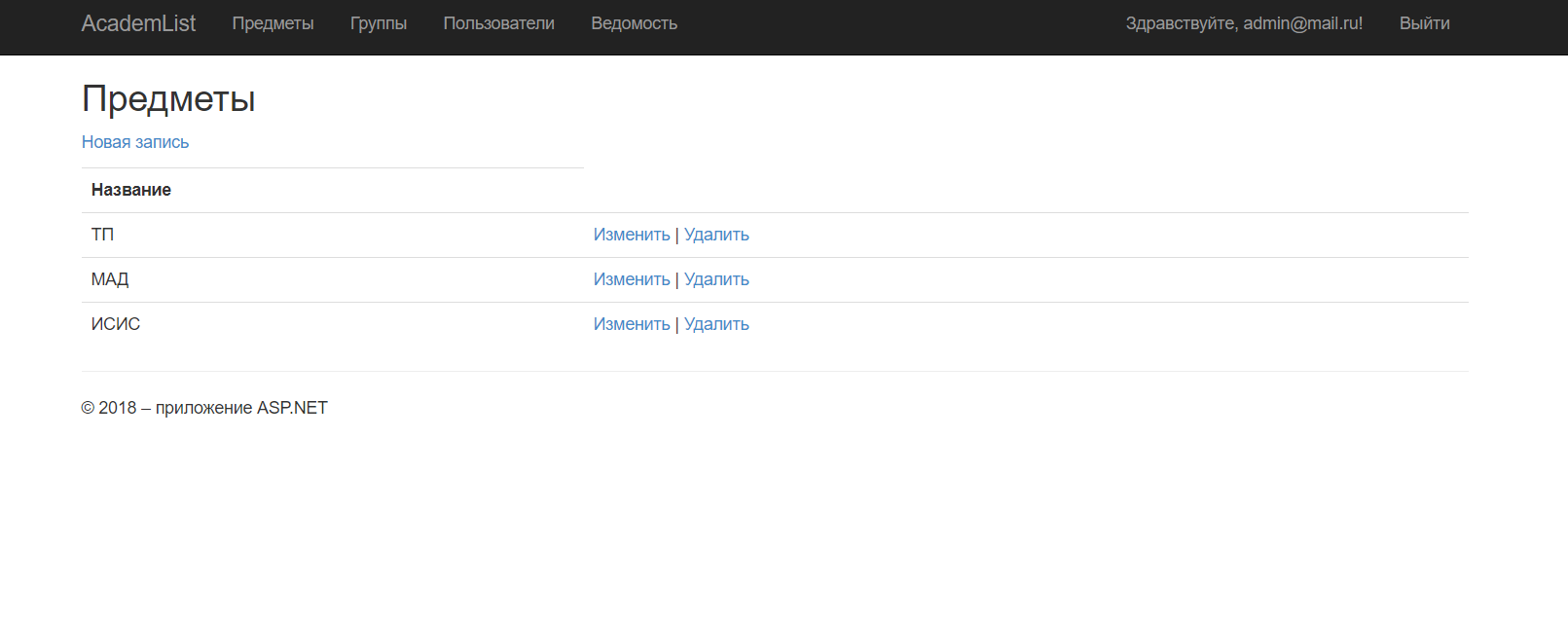


Рисунок 7 – Предметы.

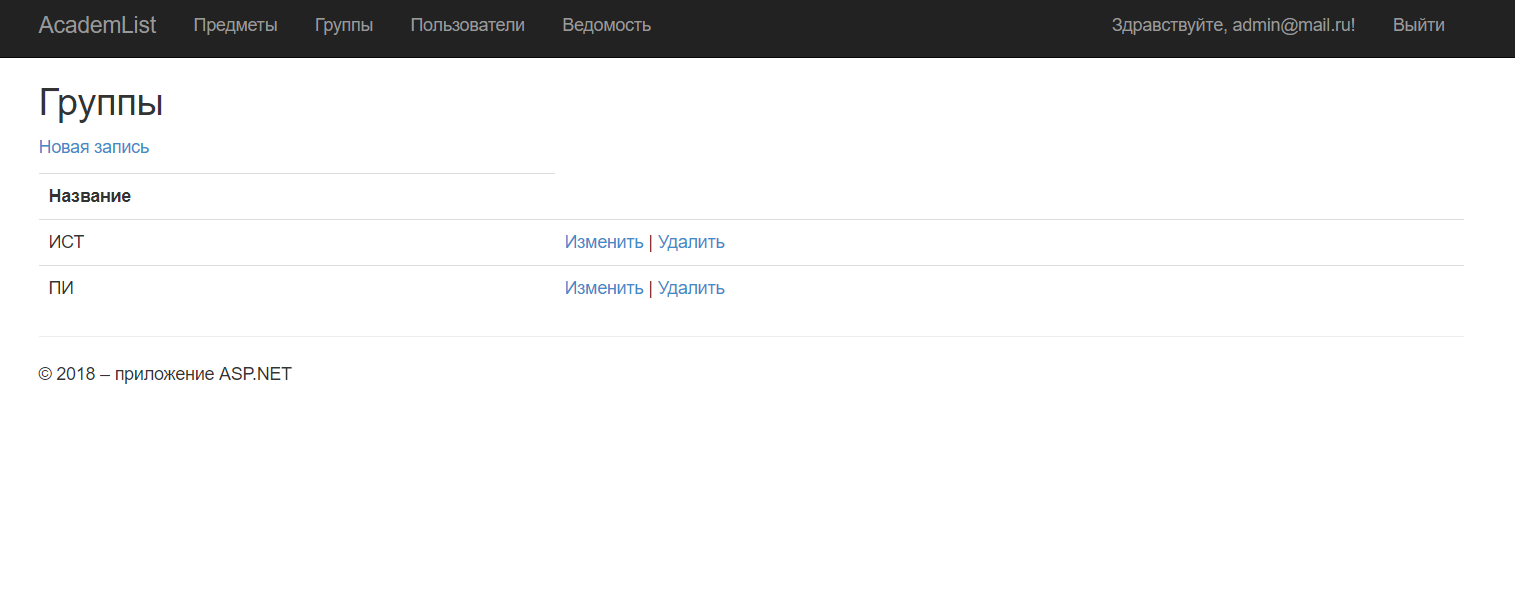


Рисунок 8 – Группы.



Рисунок 9 – Пользователи.



Рисунок 10 – Шаблон ведомости.



Рисунок 11 – Ведомость.

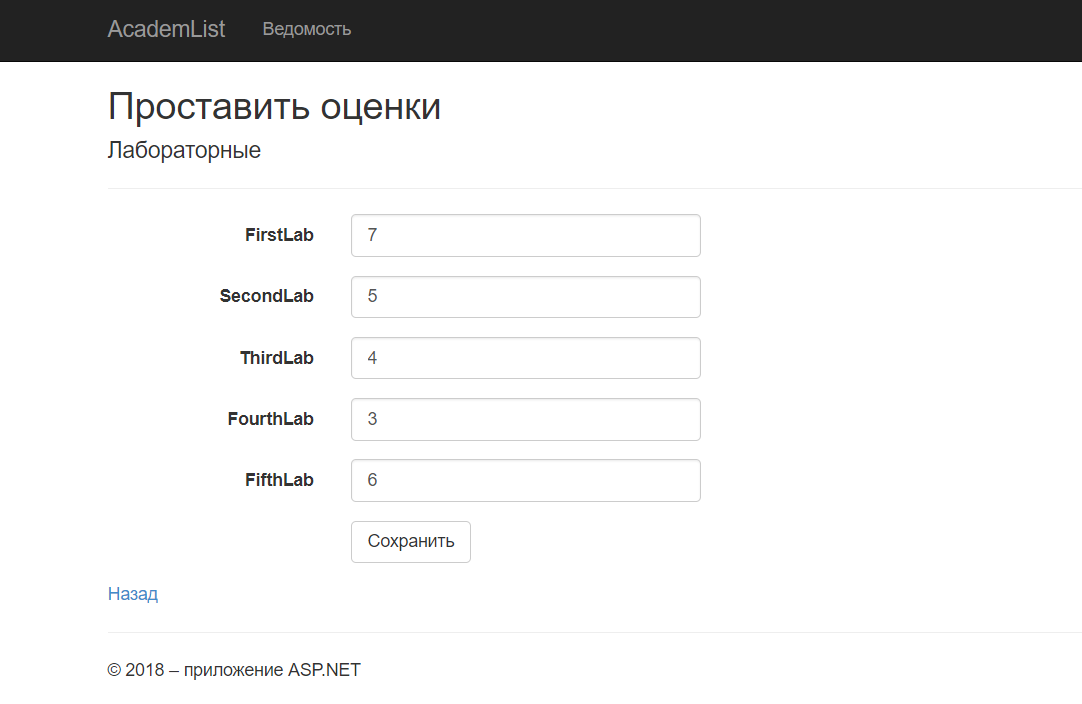


Рисунок 12 – Изменение оценок.

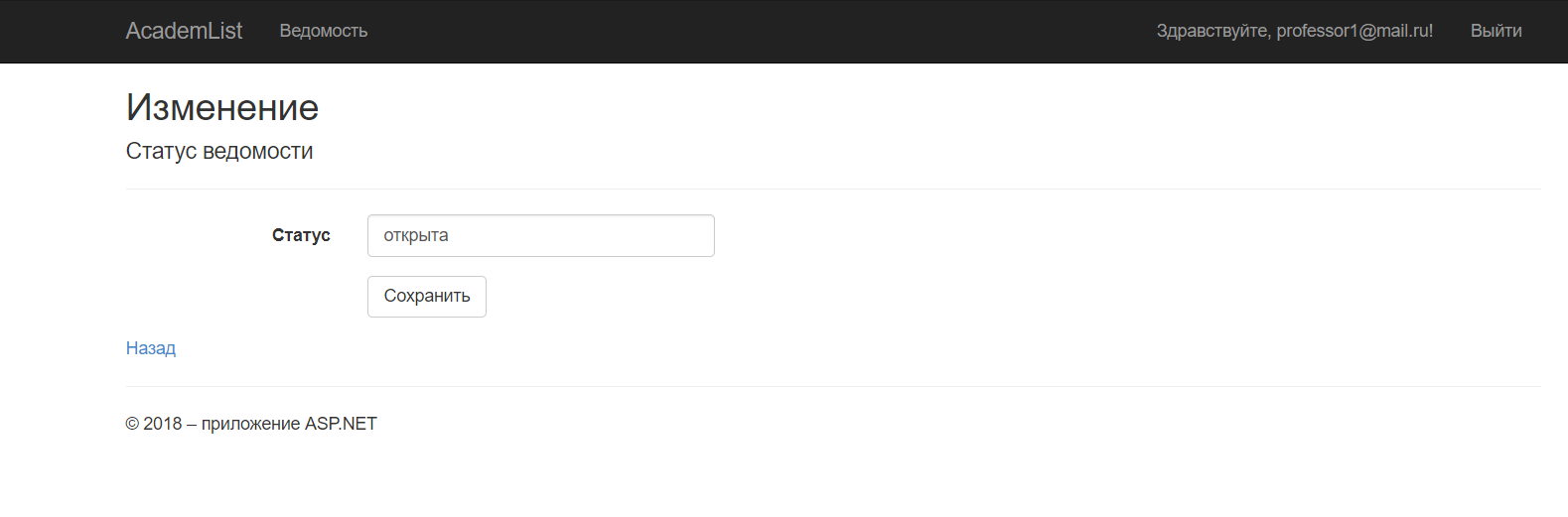


Рисунок 13 – Изменение статуса ведомости.

Исходный код программной системы представлен в приложении А.

* 1. заключение

В данном курсовом проекте спроектирована и реализована программная система учета знаний учащихся. Для выполнения поставленной задачи было выполнено:

- анализ предметной области и формирование требований к системе;

- разработка проекта программной системы;

- реализация программного обеспечения.

Разработка проекта программного продукта была выполнена на языке UML Программное обеспечение было реализовано на языке C# в виде веб-приложения, построенного по архитектуре MVC на базе программной платформы Microsoft ASP .NET.

Разработанная система позволяет вести учёт успеваемости студентов.

Данная программная система может быть использована в ВУЗах для учета знаний учащихся.

* 1. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Фримен А., Сандерсон С. ASP.NET MVC 4 Framework с примерами на C# для профессионалов. 4-е издание. Вильямс, 2014
2. Гольцман В. MySQL 5.0 Библиотека программиста. СПБ.:Питер, 2010
3. Фримен А., Джозеф C. Раттц-мл - LINQ. Язык интегрированных запросов в C# 2010 для профессионалов Вильямс, 2011
4. Рамбо Дж., Блаха М. UML 2.0. Объектно-ориентированное моделирование и разработка Питер, 2007
5. Коннолли Т., Бегг К. Базы данных. M.:Вильямс, 2003.

ПРИЛОЖЕНИЕ А Исходный код программной системы

**Модели:**

AcadList

public partial class AcadList

{

public int Id { get; set; }

public int Rating\_Id { get; set; }

public int Lab\_Id { get; set; }

public int Exam\_Id { get; set; }

public int Counter { get; set; }

public string Student\_Id { get; set; }

public int Pattern\_Id { get; set; }

public virtual Exam Exam { get; set; }

public virtual Labs Labs { get; set; }

public virtual Rating Rating { get; set; }

public virtual AspNetUsers AspNetUsers { get; set; }

public virtual Pattern Pattern { get; set; }

}

Subject

public partial class Subject

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public Subject()

{

this.Pattern = new HashSet<Pattern>();

}

public int Id { get; set; }

public string SubjectName { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Pattern> Pattern { get; set; }

}

Rating

public partial class Rating

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public Rating()

{

this.AcadList = new HashSet<AcadList>();

}

public int Id { get; set; }

public int FirstRating { get; set; }

public int SecondRating { get; set; }

public int ThirdRating { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<AcadList> AcadList { get; set; }

}

Pattern

public partial class Pattern

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public Pattern()

{

this.AcadList = new HashSet<AcadList>();

}

public int Id { get; set; }

public int Number { get; set; }

public string Semester { get; set; }

public string Professor\_Id { get; set; }

public int Subject\_Id { get; set; }

public int Group\_Id { get; set; }

public string Status { get; set; }

public virtual AspNetUsers AspNetUsers { get; set; }

public virtual Groups Groups { get; set; }

public virtual Subject Subject { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<AcadList> AcadList { get; set; }

}

Labs

public partial class Labs

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public Labs()

{

this.AcadList = new HashSet<AcadList>();

}

public int Id { get; set; }

public int FirstLab { get; set; }

public int SecondLab { get; set; }

public int ThirdLab { get; set; }

public int FourthLab { get; set; }

public int FifthLab { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<AcadList> AcadList { get; set; }

}

Groups

public partial class Groups

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public Groups()

{

this.AspNetUsers = new HashSet<AspNetUsers>();

this.Pattern = new HashSet<Pattern>();

}

public int Id { get; set; }

public string GroupName { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<AspNetUsers> AspNetUsers { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Pattern> Pattern { get; set; }

}

Exam

public partial class Exam

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public Exam()

{

this.AcadList = new HashSet<AcadList>();

}

public int Id { get; set; }

public int ExamValue { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<AcadList> AcadList { get; set; }

}

AspNetUser

public partial class AspNetUsers

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public AspNetUsers()

{

this.AspNetRoles = new HashSet<AspNetRoles>();

this.Groups = new HashSet<Groups>();

this.AcadList = new HashSet<AcadList>();

this.Pattern = new HashSet<Pattern>();

}

public string Id { get; set; }

public string Email { get; set; }

public bool EmailConfirmed { get; set; }

public string PasswordHash { get; set; }

public string SecurityStamp { get; set; }

public string PhoneNumber { get; set; }

public bool PhoneNumberConfirmed { get; set; }

public bool TwoFactorEnabled { get; set; }

public Nullable<System.DateTime> LockoutEndDateUtc { get; set; }

public bool LockoutEnabled { get; set; }

public int AccessFailedCount { get; set; }

public string UserName { get; set; }

public string Name { get; set; }

public string Surname { get; set; }

public System.DateTime BirthDate { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<AspNetRoles> AspNetRoles { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Groups> Groups { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<AcadList> AcadList { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Pattern> Pattern { get; set; }

}

AspNetRoles

public partial class AspNetRoles

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public AspNetRoles()

{

this.AspNetUsers = new HashSet<AspNetUsers>();

}

public string Id { get; set; }

public string Name { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<AspNetUsers> AspNetUsers { get; set; }

}

**DAO:**

DAOSubject

public class DAOSubject

{

private Entities entities = new Entities();

public IEnumerable<Subject> GetAllSubject()

{

return entities.Subject.Select(n => n);

}

public bool AddSubject(Subject subject)

{

try

{

entities.Subject.Add(subject);

entities.SaveChanges();

}

catch

{

return false;

}

return true;

}

public void UpdateSubject(Subject subject)

{

var Entity = entities.Subject.FirstOrDefault(x => x.Id == subject.Id);

Entity.SubjectName = subject.SubjectName;

entities.SaveChanges();

}

public Subject GetSubject(int id)

{

return entities.Subject.Where(n => n.Id == id).First();

}

public void DeleteSubject(int id)

{

Subject sub = GetSubject(id);

entities.Subject.Remove(sub);

entities.SaveChanges();

}

}

DAORating

public class DAORating

{

private Entities entities = new Entities();

public IEnumerable<Rating> GetAllRating()

{

return entities.Rating.Select(n => n);

}

public bool UpdateRating(Rating rating)

{

try

{

var Entity = entities.Rating.FirstOrDefault(x => x.Id == rating.Id);

Entity.FirstRating = rating.FirstRating;

Entity.SecondRating = rating.SecondRating;

Entity.ThirdRating = rating.ThirdRating;

entities.SaveChanges();

}

catch

{

return false;

}

return true;

}

public Rating GetRating(int id)

{

return entities.Rating.Where(n => n.Id == id).First();

}

public bool DeleteRating(int id)

{

try

{

Rating rat = GetRating(id);

entities.Rating.Remove(rat);

entities.SaveChanges();

}

catch

{

return false;

}

return true;

}

}

DAOPattern

public class DAOPattern

{

private Entities entities = new Entities();

public IEnumerable<Pattern> GetAllPattern()

{

return entities.Pattern.Select(n => n);

}

public Pattern GetPattern(int id)

{

return entities.Pattern.Where(n => n.Id == id).First();

}

public IEnumerable<Pattern> GetUserPattern(string id)

{

return entities.Pattern.Where(n => n.Professor\_Id == id);

}

public bool AddPattern(Pattern pattern)

{

try

{

entities.Pattern.Add(pattern);

entities.SaveChanges();

}

catch

{

return false;

}

return true;

}

public bool UpdatePattern(Pattern pattern)

{

try

{

var Entity = entities.Pattern.FirstOrDefault(x => x.Id == pattern.Id);

Entity.Status = pattern.Status;

entities.SaveChanges();

}

catch

{

return false;

}

return true;

}

}

DAOLabs

public class DAOLabs

{

private Entities entities = new Entities();

public IEnumerable<Labs> GetAllLabs()

{

return entities.Labs.Select(n => n);

}

public bool UpdateLabs(Labs labs)

{

try

{

var Entity = entities.Labs.FirstOrDefault(x => x.Id == labs.Id);

Entity.FirstLab = labs.FirstLab;

Entity.SecondLab = labs.SecondLab;

Entity.ThirdLab = labs.ThirdLab;

Entity.FourthLab = labs.FourthLab;

Entity.FifthLab = labs.FifthLab;

entities.SaveChanges();

}

catch

{

return false;

}

return true;

}

public Labs GetLabs(int id)

{

return entities.Labs.Where(n => n.Id == id).First();

}

public bool DeleteLabs(int id)

{

try

{

Labs lb = GetLabs(id);

entities.Labs.Remove(lb);

entities.SaveChanges();

}

catch

{

return false;

}

return true;

}

}

DAOGroups

public class DAOGroups

{

private Entities entities = new Entities();

public IEnumerable<Groups> GetAllGroups()

{

return entities.Groups.Select(n => n);

}

public bool AddGroups(Groups groups)

{

try

{

entities.Groups.Add(groups);

entities.SaveChanges();

}

catch

{

return false;

}

return true;

}

public void UpdateGroups(Groups groups)

{

var Entity = entities.Groups.FirstOrDefault(x => x.Id == groups.Id);

Entity.GroupName = groups.GroupName;

entities.SaveChanges();

}

public Groups GetGroups(int id)

{

return entities.Groups.Where(n => n.Id == id).First();

}

public void DeleteGroups(int id)

{

Groups gr = GetGroups(id);

entities.Groups.Remove(gr);

entities.SaveChanges();

}

}

DAOExam

public class DAOExam

{

private Entities entities = new Entities();

public IEnumerable<Exam> GetAllExam()

{

return entities.Exam.Select(n => n);

}

public bool AddExam(Exam exam)

{

try

{

entities.Exam.Add(exam);

entities.SaveChanges();

}

catch

{

return false;

}

return true;

}

public void UpdateExam(Exam exam)

{

var Entity = entities.Exam.FirstOrDefault(x => x.Id == exam.Id);

Entity.ExamValue = exam.ExamValue;

entities.SaveChanges();

}

public Exam GetExam(int id)

{

return entities.Exam.Where(n => n.Id == id).First();

}

public void DeleteExam(int id)

{

Exam ex = GetExam(id);

entities.Exam.Remove(ex);

entities.SaveChanges();

}

}

DAOAspNetUsers

public class DAOAspNetUser

{

private Entities entities = new Entities();

public IEnumerable<AspNetUsers> GetAllUsers()

{

return entities.AspNetUsers.Select(n => n);

}

public bool UpdateUser(AspNetUsers user)

{

try

{

var Entity = entities.AspNetUsers.FirstOrDefault(x => x.Id == user.Id);

Entity.Surname = user.Surname;

Entity.Email = user.Email;

entities.SaveChanges();

}

catch

{

return false;

}

return true;

}

public AspNetUsers GetUser(string id)

{

return entities.AspNetUsers.Where(n => n.Id == id).First();

}

public bool DeleteUser(string id)

{

try

{

AspNetUsers anu = GetUser(id);

entities.AspNetUsers.Remove(anu);

entities.SaveChanges();

}

catch

{

return false;

}

return true;

}

}

DAOAcadList

public class DAOAcadList

{

private Entities entities = new Entities();

DAOPattern daopat = new DAOPattern();

public IEnumerable<AcadList> GetAllAcadList()

{

return entities.AcadList.Select(n => n);

}

public IEnumerable<AcadList> GetGroupAcadList(int id)

{

return entities.AcadList.Select(n => n)

.Where(n => n.Pattern\_Id == id);

}

public AcadList GetAcadList(int id)

{

return entities.AcadList.Where(n => n.Id == id).First();

}

public bool AddAcadList(AcadList aclist)

{

try

{

entities.AcadList.Add(aclist);

entities.SaveChanges();

}

catch

{

return false;

}

return true;

}

public bool UpdateAcadList(AcadList aclist)

{

try

{

var Entity = entities.AcadList.FirstOrDefault(x => x.Id == aclist.Id);

Entity.Counter = aclist.Counter;

entities.SaveChanges();

}

catch

{

return false;

}

return true;

}

}

**Контроллеры:**

AspNetUserControllers

public class DAOAspNetUserController : Controller

{

DAOAspNetUser daouser = new DAOAspNetUser();

public ActionResult Index()

{

return View(daouser.GetAllUsers());

}

public ActionResult Details(int id)

{

return View();

}

public ActionResult Edit(string id)

{

return View(daouser.GetUser(id));

}

[HttpPost]

public ActionResult Edit(string id, AspNetUsers anu)

{

if ((id != null) && (anu != null) && (ModelState.IsValid))

{

daouser.UpdateUser(anu);

return RedirectToAction("Index");

}

else

{

return View("Edit");

}

}

public ActionResult Delete(string id)

{

return View("Delete", daouser.GetUser(id));

}

[HttpPost]

public ActionResult Delete(string id, FormCollection collection)

{

if (id != null && ModelState.IsValid)

{

daouser.DeleteUser(id);

return RedirectToAction("Index");

}

else

return View();

}

}

SubjectControllers

public class DAOSubjectController : Controller

{

Entities entities = new Entities();

DAOSubject daosub = new DAOSubject();

public ActionResult Index(int? page)

{

return View(daosub.GetAllSubject());

}

public ActionResult Details(int id)

{

return View(daosub.GetSubject(id));

}

public ActionResult Create()

{

return View("Create");

}

[HttpPost]

public ActionResult Create([Bind(Exclude = "Id")] Subject subject)

{

if (daosub.AddSubject(subject))

return RedirectToAction("Index");

else

{

return View("Create");

}

}

public ActionResult Edit(int id)

{

return View(daosub.GetSubject(id));

}

[HttpPost]

public ActionResult Edit(int id, Subject subject)

{

if ((id > 0) && (subject != null) && (ModelState.IsValid))

{

daosub.UpdateSubject(subject);

return RedirectToAction("Index");

}

else

{

return View("Edit");

}

}

public ActionResult Delete(int id)

{

return View("Delete", daosub.GetSubject(id));

}

[HttpPost]

public ActionResult Delete(int id, FormCollection collection)

{

if (id > 0 && ModelState.IsValid)

{

daosub.DeleteSubject(id);

return RedirectToAction("Index");

}

else return View();

}

}

RatingController

public class DAORatingController : Controller

{

DAORating daorat = new DAORating();

public ActionResult Edit(int id)

{

return View(daorat.GetRating(id));

}

[HttpPost]

public ActionResult Edit(Rating r)

{

int id = Convert.ToInt32(Request["id"]);

if ((id > 0) && (r != null) && (ModelState.IsValid))

{

daorat.UpdateRating(r);

return RedirectToAction("ProffIndex", "DAOPattern");

}

else

{

return View("Edit");

}

}

}

PatternController

public class DAOPatternController : Controller

{

DAOPattern daopattern = new DAOPattern();

DAOAspNetUser daouser = new DAOAspNetUser();

DAOSubject daosub = new DAOSubject();

DAOGroups daogr = new DAOGroups();

public ActionResult Index()

{

return View(daopattern.GetAllPattern());

}

public ActionResult ProffIndex()

{

string id = User.Identity.GetUserId();

return View(daopattern.GetUserPattern(id));

}

public ActionResult Edit(int id)

{

return View(daopattern.GetPattern(id));

}

[HttpPost]

public ActionResult Edit(int id, Pattern pat)

{

if ((id > 0) && (pat != null) && (ModelState.IsValid))

{

daopattern.UpdatePattern(pat);

return RedirectToAction("ProffIndex");

}

else

{

return View("Edit");

}

}

public ActionResult Create()

{

var user = daouser.GetAllUsers();

ViewData["Professor\_Id"] = new SelectList(user, "Id", "Surname");

var sub = daosub.GetAllSubject();

ViewData["Subject\_Id"] = new SelectList(sub, "Id", "SubjectName");

var group = daogr.GetAllGroups();

ViewData["Group\_Id"] = new SelectList(group, "Id", "GroupName");

return View("Create");

}

[HttpPost]

public ActionResult Create([Bind(Exclude = "Id")] Pattern pat)

{

if (daopattern.AddPattern(pat))

return RedirectToAction("Index");

else

{

return View("Create");

}

}

}

LabsController

public class DAOLabsController : Controller

{

DAOLabs daolabs = new DAOLabs();

public ActionResult Edit(int id)

{

return View(daolabs.GetLabs(id));

}

[HttpPost]

public ActionResult Edit( Labs l)

{

int id = Convert.ToInt32(Request["id"]);

if ((id > 0) && (l != null) && (ModelState.IsValid))

{

daolabs.UpdateLabs(l);

return RedirectToAction("ProffIndex", "DAOPattern");

}

else

{

return View("Edit");

}

}

}

GroupsController

public class DAOGroupsController : Controller

{

Entities entities = new Entities();

DAOGroups daogroup = new DAOGroups();

public ActionResult Index(int? page)

{

return View(daogroup.GetAllGroups());

}

public ActionResult Details(int id)

{

return View(daogroup.GetGroups(id));

}

public ActionResult Create()

{

return View("Create");

}

[HttpPost]

public ActionResult Create([Bind(Exclude = "Id")] Groups groups)

{

if (daogroup.AddGroups(groups))

return RedirectToAction("Index");

else

{

return View("Create");

}

}

public ActionResult Edit(int id)

{

return View(daogroup.GetGroups(id));

}

[HttpPost]

public ActionResult Edit(int id, Groups groups)

{

if ((id > 0) && (groups != null) && (ModelState.IsValid))

{

daogroup.UpdateGroups(groups);

return RedirectToAction("Index");

}

else

{

return View("Edit");

}

}

public ActionResult Delete(int id)

{

return View("Delete", daogroup.GetGroups(id));

}

[HttpPost]

public ActionResult Delete(int id, FormCollection collection)

{

if (id > 0 && ModelState.IsValid)

{

daogroup.DeleteGroups(id);

return RedirectToAction("Index");

}

else return View();

}

}

ExamController

public class DAOExamController : Controller

{

DAOExam daoex = new DAOExam();

public ActionResult Edit(int id)

{

return View(daoex.GetExam(id));

}

[HttpPost]

public ActionResult Edit(Exam ex)

{

int id = Convert.ToInt32(Request["id"]);

if ((id > 0) && (ex != null) && (ModelState.IsValid))

{

daoex.UpdateExam(ex);

return RedirectToAction("ProffIndex", "DAOPattern");

}

else

{

return View("Edit");

}

}

}

AcadListController

public class DAOAcadListController : Controller

{

Entities entities = new Entities();

DAOAcadList daoacadlist = new DAOAcadList();

DAOPattern daopat = new DAOPattern();

DAOAspNetUser daouser = new DAOAspNetUser();

public ActionResult Index()

{

return View(daoacadlist.GetAllAcadList());

}

[HttpGet]

public ActionResult AcadListIndex(int id)

{

return View(daoacadlist.GetGroupAcadList(id));

}

[HttpPost]

public ActionResult AcadListIndex()

{

int id = Convert.ToInt32(Request["id"]);

return View(daoacadlist.GetGroupAcadList(id));

}

public ActionResult Create()

{

var user = daouser.GetAllUsers();

ViewData["Student\_Id"] = new SelectList(user, "Id", "Surname");

var pat = daopat.GetAllPattern();

ViewData["Pattern\_Id"] = new SelectList(pat, "Id", "Number");

return View("Create");

}

[HttpPost]

public ActionResult Create([Bind(Exclude = "Id")] AcadList alist)

{

if (daoacadlist.AddAcadList(alist))

return RedirectToAction("Index", "DAOPattern");

else

{

return View("Create");

}

}

public ActionResult Edit(int id)

{

return View(daoacadlist.GetAcadList(id));

}

[HttpPost]

public ActionResult Edit(int id, AcadList al)

{

if ((id > 0) && (al != null) && (ModelState.IsValid))

{

daoacadlist.UpdateAcadList(al);

return RedirectToAction("ProffIndex", "DAOPattern");

}

else

{

return View("Edit");

}

}

}

ПРИЛОЖЕНИЕ Б Структура и Скрипты создания таблиц БД

CREATE TABLE [dbo].[Subject] (

[Id] INT IDENTITY (1, 1) NOT NULL,

[SubjectName] NVARCHAR (50) NOT NULL,

PRIMARY KEY CLUSTERED ([Id] ASC)

);

CREATE TABLE [dbo].[Rating] (

[Id] INT IDENTITY (1, 1) NOT NULL,

[FirstRating] INT DEFAULT ((0)) NOT NULL,

[SecondRating] INT DEFAULT ((0)) NOT NULL,

[ThirdRating] INT DEFAULT ((0)) NOT NULL,

PRIMARY KEY CLUSTERED ([Id] ASC)

);

CREATE TABLE [dbo].[Pattern] (

[Id] INT IDENTITY (1, 1) NOT NULL,

[Number] INT NOT NULL,

[Semester] NVARCHAR (50) NOT NULL,

[Professor\_Id] NVARCHAR (128) NOT NULL,

[Subject\_Id] INT NOT NULL,

[Group\_Id] INT NOT NULL,

[Status] NVARCHAR (50) NOT NULL,

PRIMARY KEY CLUSTERED ([Id] ASC),

CONSTRAINT [FK\_Pattern\_AspNetUsers] FOREIGN KEY ([Professor\_Id]) REFERENCES [dbo].[AspNetUsers] ([Id]),

CONSTRAINT [FK\_Pattern\_Groups] FOREIGN KEY ([Group\_Id]) REFERENCES [dbo].[Groups] ([Id]),

CONSTRAINT [FK\_Pattern\_Subject] FOREIGN KEY ([Subject\_Id]) REFERENCES [dbo].[Subject] ([Id])

);

CREATE TABLE [dbo].[Labs] (

[Id] INT IDENTITY (1, 1) NOT NULL,

[FirstLab] INT DEFAULT ((0)) NOT NULL,

[SecondLab] INT DEFAULT ((0)) NOT NULL,

[ThirdLab] INT DEFAULT ((0)) NOT NULL,

[FourthLab] INT DEFAULT ((0)) NOT NULL,

[FifthLab] INT DEFAULT ((0)) NOT NULL,

PRIMARY KEY CLUSTERED ([Id] ASC)

);

CREATE TABLE [dbo].[Groups] (

[Id] INT IDENTITY (1, 1) NOT NULL,

[GroupName] NVARCHAR (50) NOT NULL,

PRIMARY KEY CLUSTERED ([Id] ASC)

);

CREATE TABLE [dbo].[Exam] (

[Id] INT IDENTITY (1, 1) NOT NULL,

[ExamValue] INT DEFAULT ((0)) NOT NULL,

PRIMARY KEY CLUSTERED ([Id] ASC)

);

CREATE TABLE [dbo].[AspNetUsers] (

[Id] NVARCHAR (128) NOT NULL,

[Email] NVARCHAR (256) NULL,

[EmailConfirmed] BIT NOT NULL,

[PasswordHash] NVARCHAR (MAX) NULL,

[SecurityStamp] NVARCHAR (MAX) NULL,

[PhoneNumber] NVARCHAR (MAX) NULL,

[PhoneNumberConfirmed] BIT NOT NULL,

[TwoFactorEnabled] BIT NOT NULL,

[LockoutEndDateUtc] DATETIME NULL,

[LockoutEnabled] BIT NOT NULL,

[AccessFailedCount] INT NOT NULL,

[UserName] NVARCHAR (256) NOT NULL,

[Name] NVARCHAR (MAX) NULL,

[Surname] NVARCHAR (MAX) NULL,

[BirthDate] DATETIME DEFAULT ('1900-01-01T00:00:00.000') NOT NULL,

CONSTRAINT [PK\_dbo.AspNetUsers] PRIMARY KEY CLUSTERED ([Id] ASC)

);

CREATE TABLE [dbo].[AcadList] (

[Id] INT IDENTITY (1, 1) NOT NULL,

[Pattern\_Id] INT NOT NULL,

[Student\_Id] NVARCHAR (128) NOT NULL,

[Rating\_Id] INT NOT NULL,

[Lab\_Id] INT NOT NULL,

[Exam\_Id] INT NOT NULL,

[Counter] INT DEFAULT ((0)) NOT NULL,

PRIMARY KEY CLUSTERED ([Id] ASC),

CONSTRAINT [FK\_AcadList\_Rating] FOREIGN KEY ([Rating\_Id]) REFERENCES [dbo].[Rating] ([Id]),

CONSTRAINT [FK\_AcadList\_Labs] FOREIGN KEY ([Lab\_Id]) REFERENCES [dbo].[Labs] ([Id]),

CONSTRAINT [FK\_AcadList\_Exam] FOREIGN KEY ([Exam\_Id]) REFERENCES [dbo].[Exam] ([Id]),

CONSTRAINT [FK\_AcadList\_AspNetUsers] FOREIGN KEY ([Student\_Id]) REFERENCES [dbo].[AspNetUsers] ([Id]),

CONSTRAINT [FK\_AcadList\_Pattern] FOREIGN KEY ([Pattern\_Id]) REFERENCES [dbo].[Pattern] ([Id])

);