

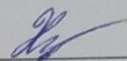
Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

Институт информационных технологий и анализа данных
наименование института

Допускаю к защите

Руководитель



подпись

В.А. Харахинов
И.О. Фамилия

Разработка клиента по технологии MVC
наименование темы

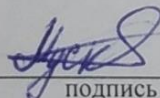
ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту по дисциплине

Управление данными

1.020.00.00 – ПЗ
обозначение документа

Выполнил студент

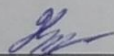
ИСТ6-17-1
шифр



подпись

И.А. Мускатин
И.О. Фамилия

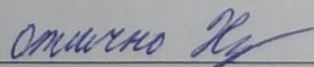
Нормоконтроль



подпись

В.А. Харахинов
И.О. Фамилия

Курсовой проект защищен с оценкой



Иркутск 2020 г.

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**ЗАДАНИЕ
НА КУРСОВОЕ ПРОЕКТИРОВАНИЕ**

По курсу Управление данными

Студенту Мускату И. А.

(фамилия, инициалы)

Тема проекта Разработка клиента по технологии MVC

Исходные данные

Имеются заявки на помощь сисадмина (номер заказа, дата/время оставления заявки, текст заявки, статус заявки и комментарий по заявке), имеются системные администраторы и пользователи (id, логин/пароль, ФИО, подразделение пользователя, уровень доступа, телефон). Каждая заявка может быть выполнена только одним сисадмином, каждый пользователь имеет возможность оставлять любое количество заявок.

Выходные документы:

- Выдать список завершенных заявок определенным сисадмином в определенный промежуток времени, с указанием основных атрибутов заявок (номер, время, текст, ФИО пользователя, ФИО администратора, выполнившего заявку, комментарий).

- Для заданного пользователя выдать полный список заявок, оставленных им, сортируя по дате.

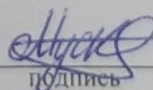
Рекомендуемая литература

1) Сосинская С.С. «Разработка клиента по технологии MVC»: Методические указания по выполнению курсового проекта» Электронный каталог кафедры вычислительной техники.

Графическая часть на _____ листах.

Дата выдачи задания « 07 » октября 2020 г.

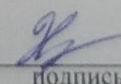
Задание получил


подпись

И.А. Мускатин
И.О. Фамилия

Дата представления проекта руководителю « 24 » декабря 2020 г.

Руководитель курсового проектирования


подпись

В.А. Харахинов
И.О. Фамилия

Содержание

Введение	4
Индивидуальный вариант	6
1 Структура базы данных	7
1.1 Метод «Объект-Связь»	7
1.2 DBDesigner	8
2 Структурная схема функционирования моделей, контроллеров и представлений	9
3 Модель данных	10
4 Контроллеры	13
4.1 Листинг кода контроллера adminsController	14
4.2 Листинг кода контроллера usersController	16
4.3 Листинг кода контроллера quest1Controller	19
5 Представления	20
5.1 Представление Index для admin	20
5.2 Представление Edit для admin	21
5.3 Представление Details для admin	23
5.4 Представление Delete для admin	24
5.5 Представление Create для admin	26
5.6 Представление Index для users	28
5.7 Представление Edit для users	29
5.8 Представление Details для users	31
5.9 Представление Delete для users	33
5.10 Представление Create для users	35
5.11 Представление Index для quest1	37
5.12 Представление query1 для quest1	38
6 Контроль	40
6.1 Модель admin с валидированными полями	40
6.2 Модель users с валидированными полями	41
6.3 Контроллер users с валидированными полями	41
7 Таблица тестов	43
8 Результаты тестирования	46
Заключение	57
Список использованных источников	58

Введение

Курсовой проект по дисциплине «Управление данными» посвящен изучению архитектурному принципу проектирования и разработки Web-приложения баз данных MVC и платформы ASP.NET MVC.

При выполнении индивидуального задания необходимо создать Web-приложение с помощью MVC, по описанной предметной области. Приложение должно выводить указанные в задании данные, производить их корректировку и удаление, а также выводить выходные документы.

Цель курсовой работы - получение навыков работы с MVC ASP.NET.

Задачи курсовой работы:

- изучить MVC,
- создать Web-приложение согласно предметной области, указанной в индивидуальном варианте.

ASP.NET MVC Framework — фреймворк для создания веб-приложений, который реализует шаблон Model-view-controller.

Платформа ASP.NET MVC базируется на взаимодействии трех компонентов: контроллера, модели и представления.

Модель

Модель предоставляет данные и методы работы с ними: запросы в базу данных, проверка на корректность. Модель не зависит от представления (не знает, как данные визуализировать) и контроллера (не имеет точек взаимодействия с пользователем), просто предоставляя доступ к данным и управлению ими.

Представление

Представление отвечает за получение необходимых данных из модели и отправляет их пользователю. Представление не обрабатывает введенные данные пользователя.

Контроллер

Контроллер обеспечивает «связи» между пользователем и системой. Он контролирует и направляет данные от пользователя к системе и наоборот, использует модель и представление для реализации необходимого действия.

Модель представляет слой, описывающий логику организации данных в приложении. Представление получает данные из контроллера и генерирует элементы пользовательского интерфейса для отображения информации.

Шаблон MVC позволяет создавать приложения, различные аспекты которых (логика ввода, бизнес-логика и логика интерфейса) разделены, но достаточно тесно взаимодействуют друг с другом. Эта схема указывает расположение каждого вида логики в приложении. Пользовательский интерфейс располагается в представлении. Логика ввода располагается в контроллере. Бизнес-логика находится в модели. Это разделение позволяет работать со сложными структурами при создании приложения. Например, разработчик может сконцентрироваться на создании представления отдельно от бизнес-логики.

Общая схема взаимодействия основных компонентов MVC представлена на рисунке 1.

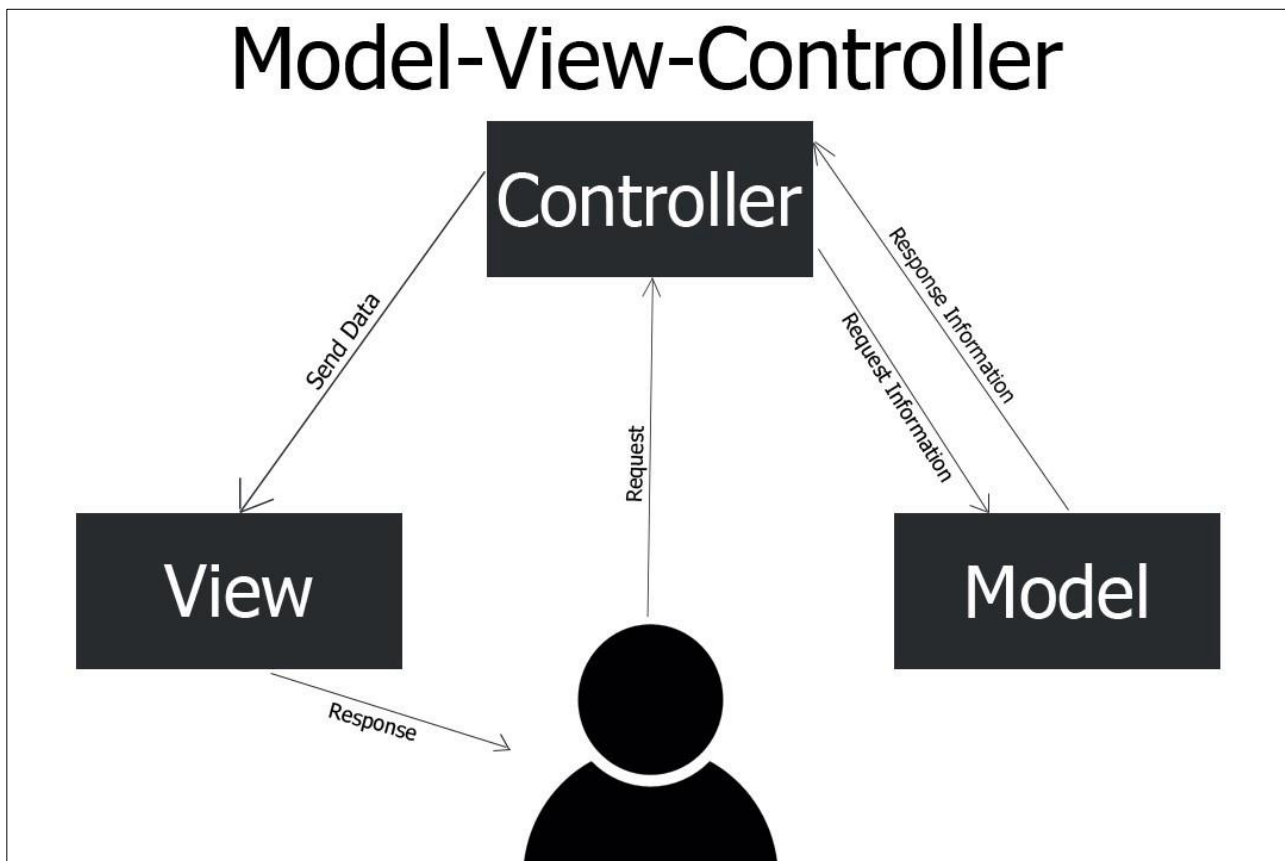


Рисунок 1 - Схема взаимодействия контроллера, модели и представления

Индивидуальный вариант

Имеются заявки на помощь сисадмина (номер заказа, дата/время оставления заявки, текст заявки, статус заявки и комментарий по заявке), имеются системные администраторы и пользователи (id, логин/пароль, ФИО, подразделение пользователя, уровень доступа, телефон). Каждая заявка может быть выполнена только одним сисадмином, каждый пользователь имеет возможность оставлять любое количество заявок.

Выходные документы:

- Выдать список завершенных заявок определенным сисадмином в определенный промежуток времени, с указанием основных атрибутов заявок (номер, время, текст, ФИО пользователя, ФИО администратора, выполнившего заявку, комментарий).
- Для заданного пользователя выдать полный список заявок, оставленных им, сортируя по дате.

1 Структура базы данных

1.1 Метод «Объект-Связь»

Перед созданием Web-приложения необходимо сформировать понятия о предметах, фактах и событиях, которые предполагает индивидуальное задание. Для того, чтобы привести эти понятия к модели данных, необходимо заменить их информационными представлениями. Наиболее удобным инструментом унифицированного представления данных является модель «объект-связь» (ER-model). Она определяет значения данных в контексте их взаимосвязи с другими данными.

Модель базы данных Web-приложения на основе метода «Объект-связь» отражена на рисунке 1.1 в виде ER-диаграммы.

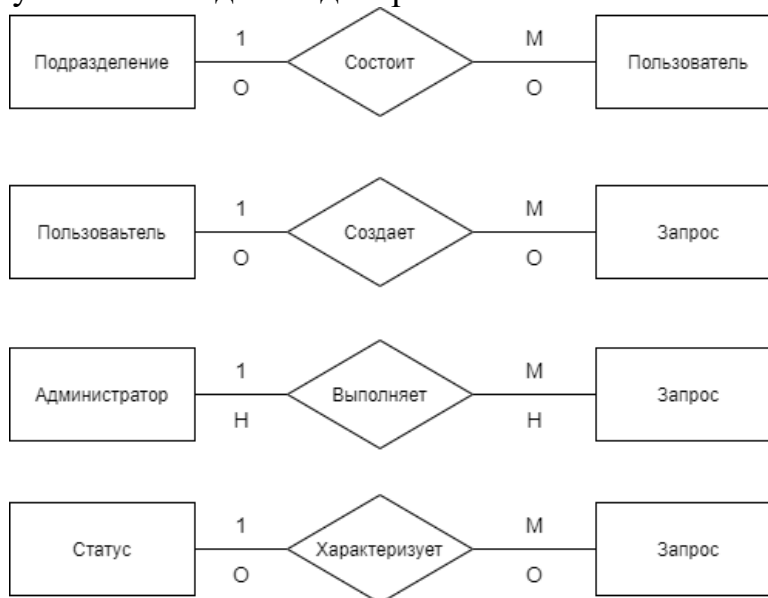


Рисунок 1.1 – ER-диаграмма

Атрибуты сущностей:

- **Пользователь** (Id, ФИО, Логин/Пароль, номер телефона, уровень доступа, подразделение);
- **Администратор** (Id, ФИО, Логин/Пароль, номер телефона);
- **Подразделение** (Id, наименование);
- **Статус** (Id, наименование);
- **Заявка** (Id заявки, Id пользователя, Id администратора, текст запроса, статус запроса, комментарий администратора, время оставления заявки).

Для каждой ER- диаграммы согласно правилу 4:

Пользователь (Id пользователя, Id подразделения, ...) – Подразделение (Id подразделения, наименование подразделения).

Пользователь (Id пользователя, Id подразделения ...) – Запрос (Id запроса, текст, Id статуса, комментарий, время).

Администратор (Id администратора, ...) – Запрос (Id запроса, Id администратора, Id пользователя, текст, статус, комментарий, время).

1.2 DBDesigner

На рисунке 1.2 отображена логическая модель базы данных, созданная с использованием CASE-средства DBDesigner.

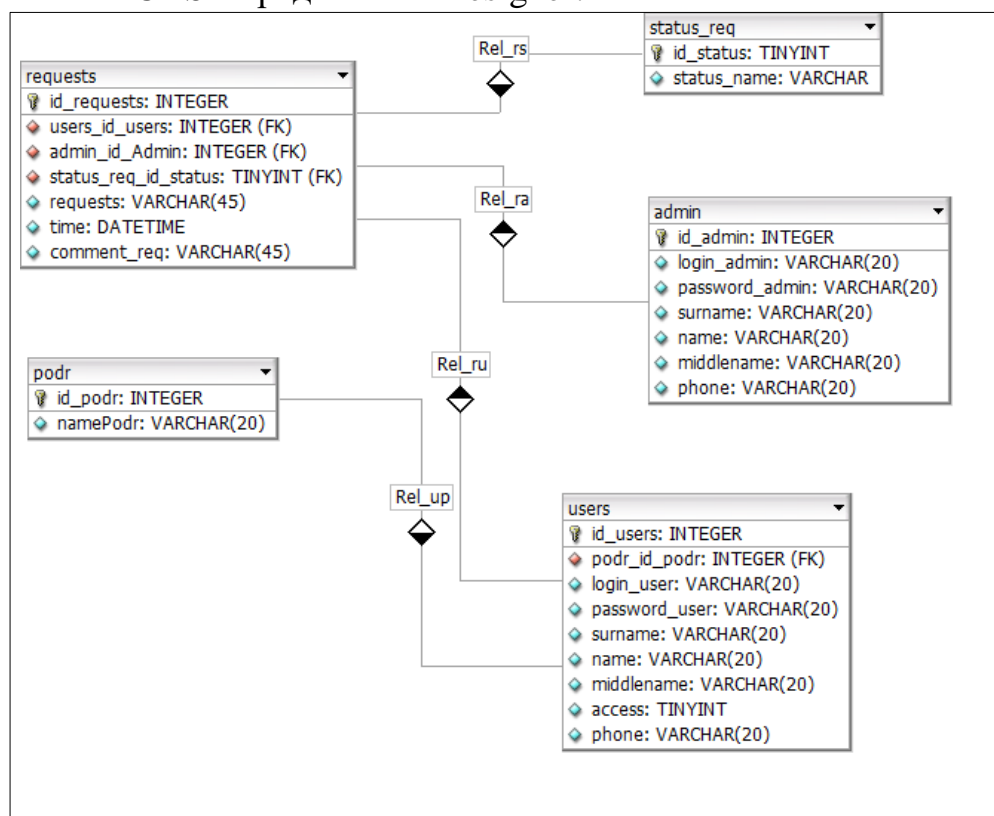


Рисунок 1.2 – Логическая модель БД «Запросы»

2 Структурная схема функционирования моделей, контроллеров и представлений

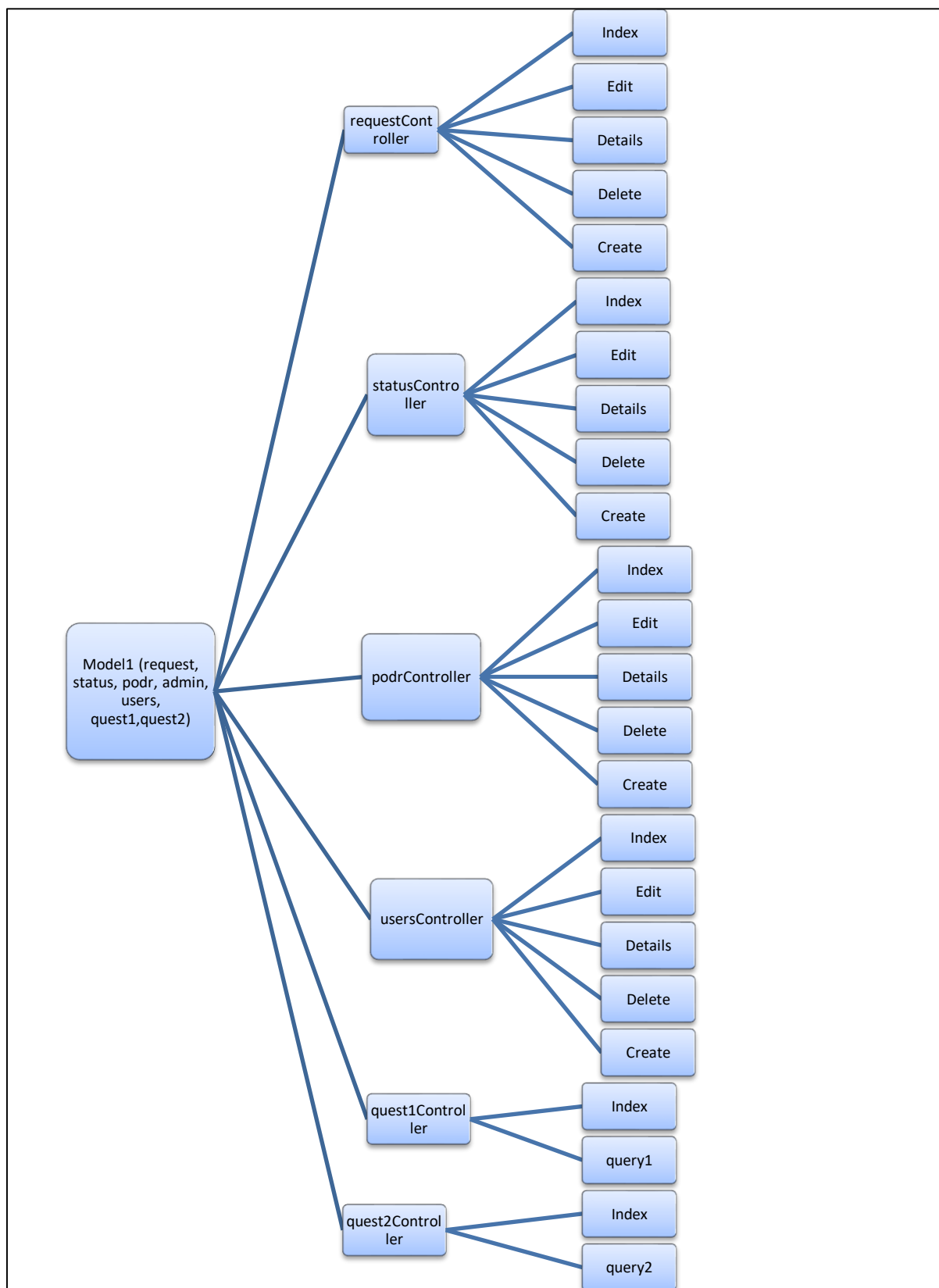


Рисунок 2.1 – Схема функционирования модели, контроллеров и представлений

3 Модель данных

Модель представляет слой, описывающий логику организации данных в приложении.

Для разработанного Web-приложения была использована существующая база данных, на основе которой создавалась модель данных проекта. Данный процесс описан далее.

Чтобы создать модель данных на основе базы данных необходимо в папке моделей добавить к проекту новый элемент в виде «ADO.NET EDM», далее выбрать создание модели на основе базы данных, выбрать необходимое подключение и указать таблицы требуемые для построения модели данных. Данный процесс отображен на рисунках 3.1-3.4.

После выполнения выше описанных шагов готовая модель отображается в обозревателе решений (рисунок 3.5). Также ее можно просмотреть отдельно в графическом виде конструкторе Entity Framework (рисунок 3.6).

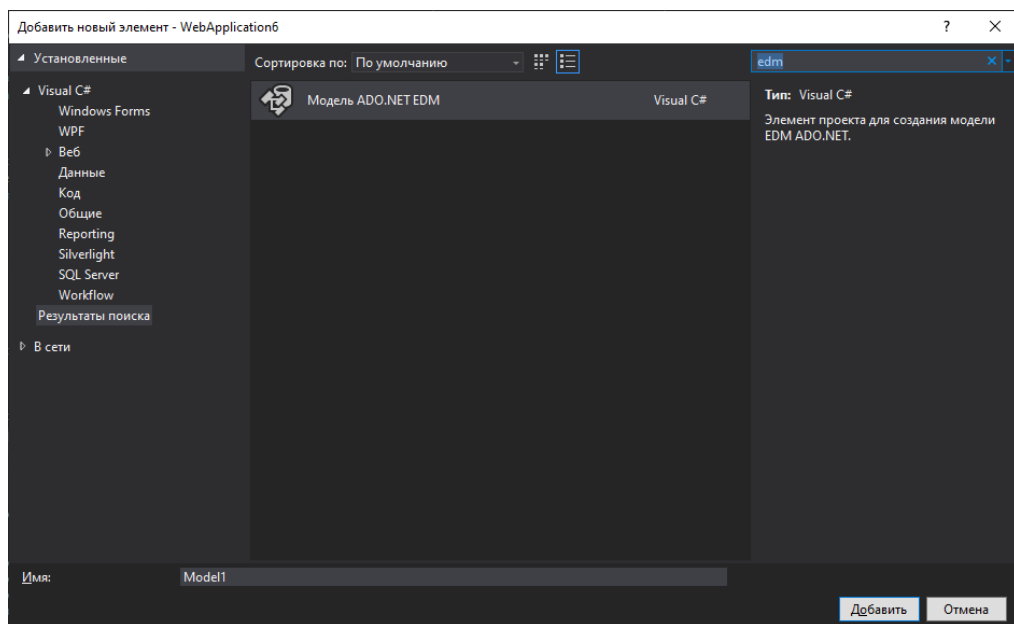


Рисунок 3.1 – Выбор элемента модели

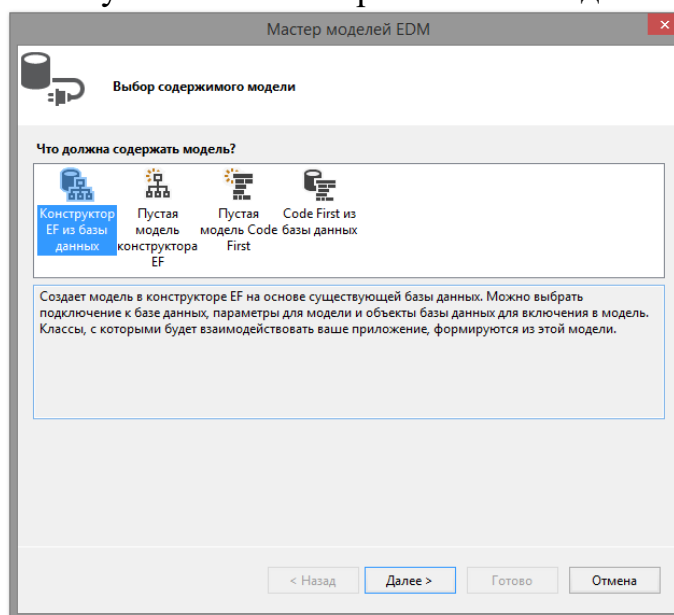


Рисунок 3.2 – Выбор содержимого модели (из существующей БД)

Мастер моделей EDM

Выбор подключения к данным

Какое подключение к данным будет использоваться приложением для подключения к базе данных?

labEntities (WebApplication6) Создать соединение...

Возможно, эта строка подключения содержит конфиденциальные данные (например, пароль), которые требуются для подключения к базе данных. Хранение конфиденциальных данных в строке подключения может представлять угрозу безопасности. Включить конфиденциальные данные в строку подключения?

☐ Нет, исключить конфиденциальные данные из строки подключения. Они будут заданы в коде приложения.

☐ Да, включить конфиденциальные данные в строку подключения.

Строка подключения:

```
metadata=res://*/Model1.csdl|res://*/Model1.ssdl|
res://*/Model1.msl;provider=System.Data.SqlClient;provider connection string="data source=
(localdb)\MSSQLLocalDB;initial catalog=lab;integrated
security=True;multipleactiveresultsets=True;application name=EntityFramework"
```

☒ Сохранить параметры соединения в Web.Config как:

labEntities

< Назад **Далее >** Готово Отмена

Рисунок 3.3 – Выбор подключения к данным.

Мастер моделей EDM

Выберите параметры и объекты базы данных

Какие объекты базы данных нужно включить в модель?

☒ Таблицы

☒ dbo

☒ admin

☒ podr

☒ requests

☒ status_req

☒ users

☐ Представления

☒ Хранимые процедуры и функции

☐ dbo

☐ Формировать имена объектов во множественном или единственном числе

☒ Включить столбцы внешних ключей в модель

☒ Импортировать выбранные хранимые процедуры и функции в модель сущностей

Пространство имен модели:

labModel1

< Назад **Далее >** Готово Отмена

Рисунок 3.4 – Выбор необходимых объектов

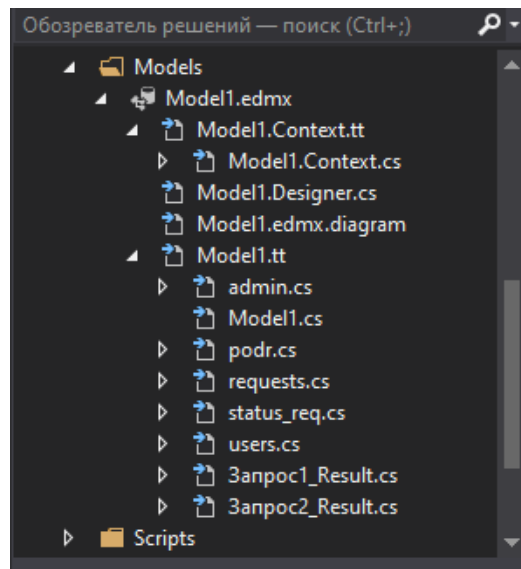


Рисунок 3.5 – Модель в окне «Обозреватель решений»

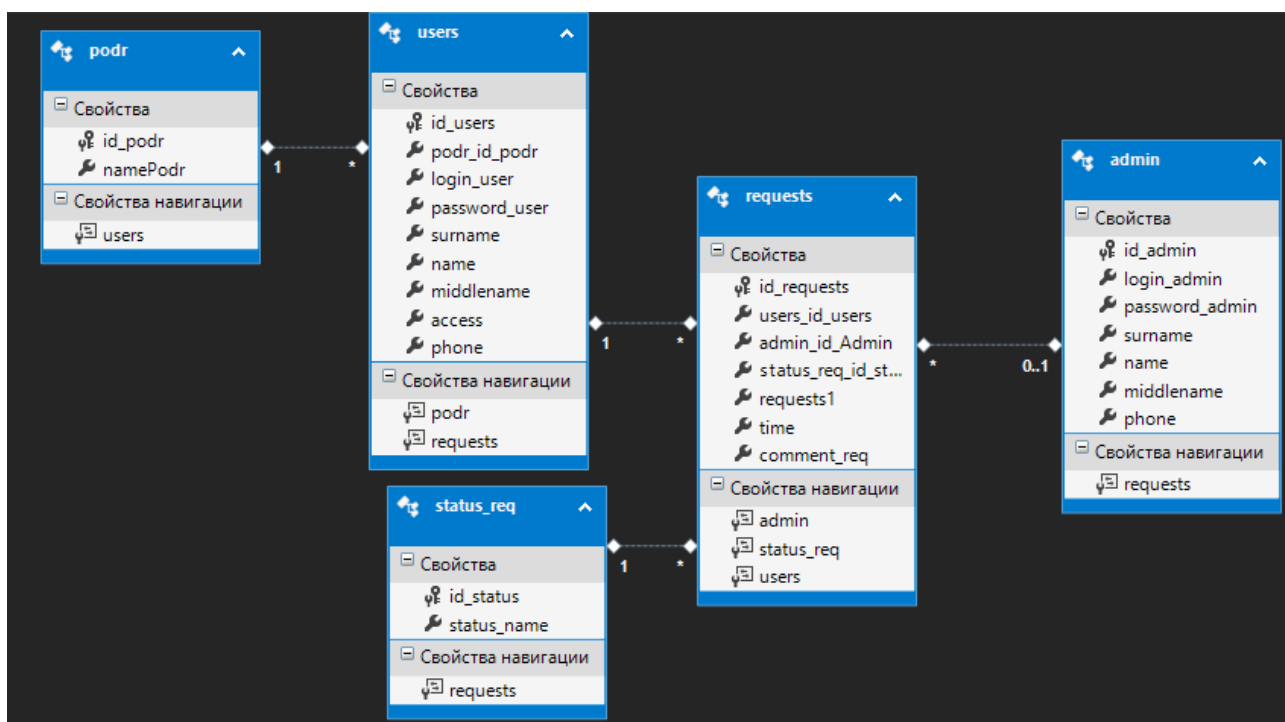


Рисунок 3.6 – Просмотр модели данных в конструкторе Entity Framework

4 Контроллеры

Контроллер в приложении необходим для того, чтобы принимать запросы, обрабатывать пользовательский ввод, взаимодействовать с моделью и представлением и возвращать пользователю результат обработки запроса.

Чтобы создать контроллер на основе существующей модели данных необходимо в папке контроллеров добавить к проекту новый контроллер, выбрать «Контроллер MVC 5 с представлениями, использующий Entity Framework», выбрать класс модели и класс контекста данных в опциях создания контроллера, назначить ему имя. Данный процесс отображен на рисунках 4.1-4.2.

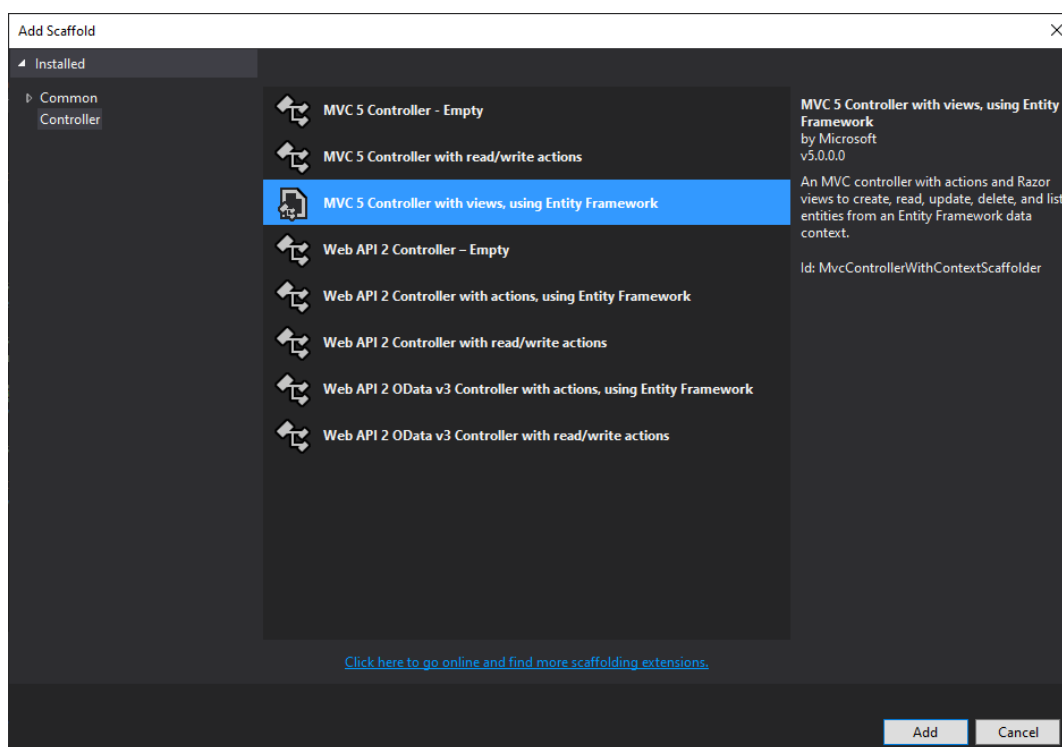


Рисунок 4.1 – Создание контроллера

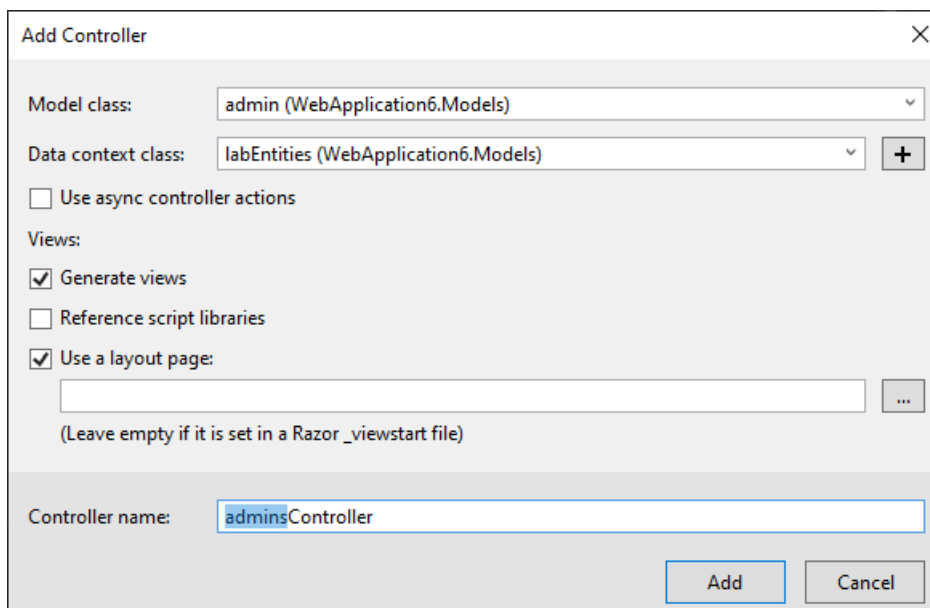


Рисунок 4.2 – Добавление контроллера

4.1 Листинг кода контроллера adminsController

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using WebApplication6.Models;

namespace WebApplication6.Controllers
{
    public class adminsController : Controller
    {
        private labEntities db = new labEntities();
        // Метод выводит на экран список администраторов в таблицу
        public ActionResult Index()
        {
            return View(db.admin.ToList());
        }
        // Метод открывает View подробности для конкретного администратора
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            admin admin = db.admin.Find(id);
            if (admin == null)
            {
                return HttpNotFound();
            }
            return View(admin);
        }
        // Метод открывающий View создания администратора
        public ActionResult Create()
        {
            return View();
        }
        // Метод обработки кнопки для создания администратора, включающий в себя валидацию –
        // совпадение логинов/телефонов пользователей в базе данных.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include = "id_admin,login_admin,password_admin,sur-
name,name,middlename,phone")] admin admin)
        {
            int logcheck = 0;
            int phonecheck = 0;
            logcheck = (from persons in db.admin where persons.id_admin != admin.id_admin
where persons.login_admin == admin.login_admin select persons).Count();
            logcheck += (from persons in db.users where persons.login_user == ad-
min.login_admin select persons).Count();
            phonecheck = (from persons in db.admin where persons.id_admin != admin.id_admin
where persons.phone == admin.phone select persons).Count();
            phonecheck += (from persons in db.users where persons.phone == admin.login_admin
select persons).Count();
            if (ModelState.IsValid && phonecheck == 0 && logcheck == 0)
            {
                db.admin.Add(admin);
                db.SaveChanges();
                return RedirectToAction("Index");
            }
        }
    }
}
```



```

        if (phonecheck != 0) { ModelState.AddModelError(String.Empty, "Пользователь с та-
ким телефоном уже существует"); }
        if (logcheck != 0) { ModelState.AddModelError(String.Empty, "Пользователь с та-
ким логином уже существует"); }
        return View(admin);
    }
    // Метод открывающий View редактирования для администратора
    public ActionResult Edit(int? id)
    {
        if (id == null)
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }
        admin admin = db.admin.Find(id);
        if (admin == null)
        {
            return HttpNotFound();
        }
        return View(admin);
    }
    // Метод обрабатывающий нажатие на кнопку редактирования записи с валидацией.
    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult Edit([Bind(Include = "id_admin,login_admin,password_admin,sur-
name,name,middlename,phone")] admin admin)
    {
        int logcheck = 0;
        int phonecheck = 0;
        logcheck = (from persons in db.admin where persons.id_admin != admin.id_admin
where persons.login_admin == admin.login_admin select persons).Count();
        logcheck += (from persons in db.users where persons.id_users != admin.id_admin
where persons.login_user == admin.login_admin select persons).Count();
        phonecheck = (from persons in db.admin where persons.id_admin != admin.id_admin
where persons.phone == admin.phone select persons).Count();
        phonecheck += (from persons in db.users where persons.id_users != admin.id_admin
where persons.phone == admin.login_admin select persons).Count();
        if (ModelState.IsValid && phonecheck == 0 && logcheck == 0)
        {
            db.Entry(admin).State = EntityState.Modified;
            db.SaveChanges();
            return RedirectToAction("Index");
        }
        if (phonecheck != 0) { ModelState.AddModelError(String.Empty, "Пользователь с
таким телефоном уже существует"); }
        if (logcheck != 0) { ModelState.AddModelError(String.Empty, "Пользователь с та-
ким логином уже существует"); }
        return View(admin);
    }
    // GET: Открытие View Удаления администратора
    public ActionResult Delete(int? id)
    {
        if (id == null)
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }
        admin admin = db.admin.Find(id);
        if (admin == null)
        {
            return HttpNotFound();
        }
        return View(admin);
    }
    // Обработка нажатия удаления администратора
    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]

```

```

        public ActionResult DeleteConfirmed(int id)
        {
            int usersreq = 0;
            usersreq = (from req in db.requests where req.admin_id_Admin == id select
req).Count();
            admin admin = db.admin.Find(id);
            if (usersreq == 0)
            {
                db.admin.Remove(admin);
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            if (usersreq != 0) { ModelState.AddModelError(String.Empty, $"Уданного админи-
стратора обнаружены закрепленные заявки в количестве {usersreq}. Для удаления администратора
удалите его записи"); }

            return View(admin);
        }
        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}

```

4.2 Листинг кода контроллера usersController

```

using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using System.Web.Mvc;
using WebApplication6.Models;

namespace WebApplication6.Controllers
{
    public class usersController : Controller
    {
        private labEntities db = new labEntities();
        // Открытие View пользователей с включенным набором данных из таблицы podr
        public ActionResult Index()
        {
            var users = db.users.Include(u => u.podr);
            return View(users.ToList());
        }
        // Открытие View детали пользователя
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            users users = db.users.Find(id);
            if (users == null)
            {
                return HttpNotFound();
            }
            return View(users);
        }
    }
}

```

```

// Открытие View создания пользователя
public ActionResult Create()
{
    ViewBag.podrr_id_podrr = new SelectList(db.podrr, "id_podrr", "namePodrr");
    return View();
}
// Обработчик кнопки создания пользователя
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create([Bind(Include = "id_users,podrr_id_podrr,login_user,password_user,surname,name,middlename,access,phone")] users users)
{
    int logcheck = 0;
    int phonecheck = 0;
    logcheck = (from persons in db.admin where persons.login_admin == users.login_user select persons).Count();
    logcheck += (from persons in db.users where persons.id_users != users.id_users where persons.login_user == users.login_user select persons).Count();
    phonecheck = (from persons in db.admin where persons.phone == users.phone select persons).Count();
    phonecheck += (from persons in db.users where persons.id_users != users.id_users where persons.phone == users.phone select persons).Count();
    if (ModelState.IsValid && phonecheck == 0 && logcheck == 0)
    {
        db.users.Add(users);
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    if (phonecheck != 0) { ModelState.AddModelError(String.Empty, "Пользователь с таким телефоном уже существует"); }
    if (logcheck != 0) { ModelState.AddModelError(String.Empty, "Пользователь с таким логином уже существует"); }
    ViewBag.podrr_id_podrr = new SelectList(db.podrr, "id_podrr", "namePodrr", users.podrr_id_podrr);
    return View(users);
}
// переход на View редактирования пользователя
public ActionResult Edit(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    users users = db.users.Find(id);
    if (users == null)
    {
        return HttpNotFound();
    }
    ViewBag.podrr_id_podrr = new SelectList(db.podrr, "id_podrr", "namePodrr", users.podrr_id_podrr);
    return View(users);
}
// обработчик нажатия на кнопку изменить пользователя с валидацией
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit([Bind(Include = "id_users,podrr_id_podrr,login_user,password_user,surname,name,middlename,access,phone")] users users)
{
    int logcheck = 0;
    int phonecheck = 0;
    logcheck = (from persons in db.admin where persons.id_admin != users.id_users where persons.login_admin == users.login_user select persons).Count();
    logcheck += (from persons in db.users where persons.id_users != users.id_users where persons.login_user == users.login_user select persons).Count();

```

```

        phonecheck = (from persons in db.admin where persons.id_admin != users.id_users
where persons.phone == users.phone select persons).Count();
        phonecheck += (from persons in db.users where persons.id_users != users.id_users
where persons.phone == users.phone select persons).Count();
        if (ModelState.IsValid && phonecheck == 0 && logcheck == 0)
        {
            db.Entry(users).State = EntityState.Modified;
            db.SaveChanges();
            return RedirectToAction("Index");
        }
        if (phonecheck != 0) { ModelState.AddModelError(String.Empty, "Пользователь с
таким телефоном уже существует"); }
        if (logcheck != 0) { ModelState.AddModelError(String.Empty, "Пользователь с та-
ким логином уже существует"); }
        ViewBag.podr_id_podr = new SelectList(db.podr, "id_podr", "namePodr", us-
ers.podr_id_podr);
        return View(users);
    }
    // переход на View удаление пользователя
    public ActionResult Delete(int? id)
    {
        if (id == null)
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }
        users users = db.users.Find(id);
        if (users == null)
        {
            return HttpNotFound();
        }
        return View(users);
    }
    // обработка кнопки удаления ползователя
    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]
    public ActionResult DeleteConfirmed(int id)
    {
        int usersreq = 0;
        usersreq = (from req in db.requests where req.users_id_users == id select
req).Count();
        users users = db.users.Find(id);
        if (usersreq == 0)
        {
            db.users.Remove(users);
            db.SaveChanges();
            return RedirectToAction("Index");
        }
        if (usersreq != 0) { ModelState.AddModelError(String.Empty, $"Уданного пользова-
теля обнаружены заявки в количестве {usersreq}. Для удаления пользователя удалите его за-
писи"); }
        return View(users);
    }
    protected override void Dispose(bool disposing)
    {
        if (disposing)
        {
            db.Dispose();
        }
        base.Dispose(disposing);
    }
}

```

4.3 Листинг кода контроллера quest1Controller

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using WebApplication6.Models;

namespace WebApplication6.Controllers
{
    public class quest1Controller : Controller
    {
        // Вывод окна запроса 1 с создание полей заполненных данными из admin и status
        private labEntities db = new labEntities();
        public ActionResult Index()
        {
            ViewBag.id_status = new SelectList(db.status_req, "id_status", "status_name");
            ViewBag.id_admin = new SelectList(db.admin, "id_admin", "surname");
            return View();
        }
        // Обработчик реакции нажатия на кнопку, выполнение запроса по введенным данным
        public ActionResult Query1(int id_admin, int id_status, DateTime time1, DateTime
time2)
        {
            var q1 = db.3anpoc1(id_admin, id_status, time1, time2);
            return View(q1);
        }
    }
}
```

5 Представления

Представление получает данные из контроллера и генерирует элементы пользовательского интерфейса для отображения информации.

При создании контроллера представления Index, Edit, Details, Delete, Create создаются автоматически. Их вид отображен на рисунках 5.1-5.12 для контроллеров admin, users и quest1.

5.1 Представление Index для admin

Запросы Администраторы Пользователи Подразделения Статус Запрос 1 Запрос 2						
Администраторы						
Создать						
Логин	Пароль	Фамилия	Имя	Отчество	Телефон	
admin3	123	Кирильчук	Николай	Олегович	+7 (544) 952-13-84	Изменить Подробнее Удалить
admin1	123	Петров	Василий	Иванович	+7 (234) 632-46-41	Изменить Подробнее Удалить
admin11	123	Сорокин	Василий	Олегович	+7 (345) 765-88-67	Изменить Подробнее Удалить
kkk222	123	Павлов	Игорь	Алексеевич	+7 (999) 777-77-77	Изменить Подробнее Удалить
admin12311	111	Мускатин	Игорь	Алексеевич	+7 (000) 111-11-11	Изменить Подробнее Удалить
© 2020 - мой курсовой проект						

Рисунок 5.1 – Представление Index для admin

```
@model IEnumerable<WebApplication6.Models.admin>
@{
    ViewBag.Title = "Администраторы";
}
<h2>Администраторы</h2>
<p>
    @Html.ActionLink("Создать", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.login_admin)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.password_admin)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.surname)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.name)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.middlename)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.phone)
        </th>
        <th></th>
    </tr>
    @foreach (var item in Model) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.login_admin)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.password_admin)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.surname)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.name)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.middlename)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.phone)
            </td>
            <td>
                @Html.ActionLink("Изменить", "Edit", new { id = item.id }) |
                @Html.ActionLink("Подробнее", "Details", new { id = item.id }) |
                @Html.ActionLink("Удалить", "Delete", new { id = item.id })
            </td>
        </tr>
    }
</table>
```



```

        <td>
            @Html.DisplayFor(modelItem => item.password_admin)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.surname)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.name)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.middlename)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.phone)
        </td>
        <td>
            @Html.ActionLink("Изменить", "Edit", new { id=item.id_admin }) |
            @Html.ActionLink("Подробнее", "Details", new { id=item.id_admin }) |
            @Html.ActionLink("Удалить", "Delete", new { id=item.id_admin })
        </td>
    </tr>
}
</table>

```

5.2 Представление Edit для admin

Рисунок 5.2 – Представление Edit для admin

```

@model WebApplication6.Models.admin
@{
    ViewBag.Title = "Изменение администратора";
}
<h2>Изменение администратора</h2>
@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()
    <div class="form-horizontal">
        <h4>Администратор</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.id_admin)
    </div>

```

```

        <div class="form-group">
            @Html.LabelFor(model => model.login_admin, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.login_admin, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.login_admin, "", new { @class = "text-danger" })
            </div>
        </div>
        <div class="form-group">
            @Html.LabelFor(model => model.password_admin, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.password_admin, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.password_admin, "", new { @class = "text-danger" })
            </div>
        </div>
        <div class="form-group">
            @Html.LabelFor(model => model.surname, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.surname, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.surname, "", new { @class = "text-danger" })
            </div>
        </div>
        <div class="form-group">
            @Html.LabelFor(model => model.name, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.name, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.name, "", new { @class = "text-danger" })
            </div>
        </div>
        <div class="form-group">
            @Html.LabelFor(model => model.middlename, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.middlename, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.middlename, "", new { @class = "text-danger" })
            </div>
        </div>
        <div class="form-group">
            @Html.LabelFor(model => model.phone, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.phone, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.phone, "", new { @class = "text-danger" })
            </div>
        </div>
        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Изменить" class="btn btn-default" />
            </div>
        </div>
    </div>

```

```

    </div>
  }
  <div>
    @Html.ActionLink("Назад", "Index")
  </div>
  @section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
  }
}

```

5.3 Представление Details для admin

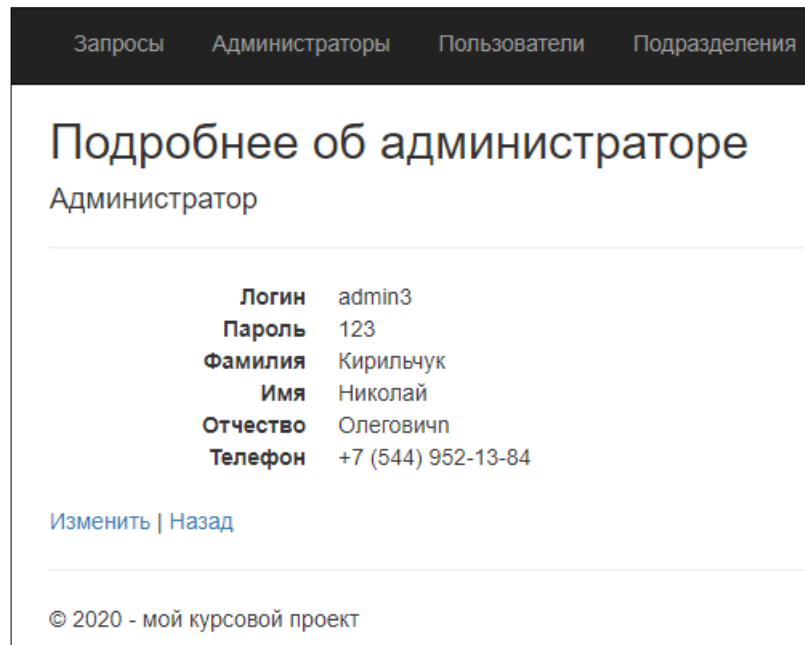


Рисунок 5.3 – Представление Details для admin

```

@model WebApplication6.Models.admin
@{
    ViewBag.Title = "Подробнее об администраторе";
}
<h2>Подробнее об администраторе</h2>
<div>
    <h4>Администратор</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.login_admin)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.login_admin)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.password_admin)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.password_admin)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.surname)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.surname)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.name)
        </dt>
    </dl>

```

```

        <dd>
            @Html.DisplayFor(model => model.name)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.middlename)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.middlename)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.phone)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.phone)
        </dd>
    </dl>
</div>
<p>
    @Html.ActionLink("Изменить", "Edit", new { id = Model.id_admin }) |
    @Html.ActionLink("Назад", "Index")
</p>

```

5.4 Представление Delete для admin

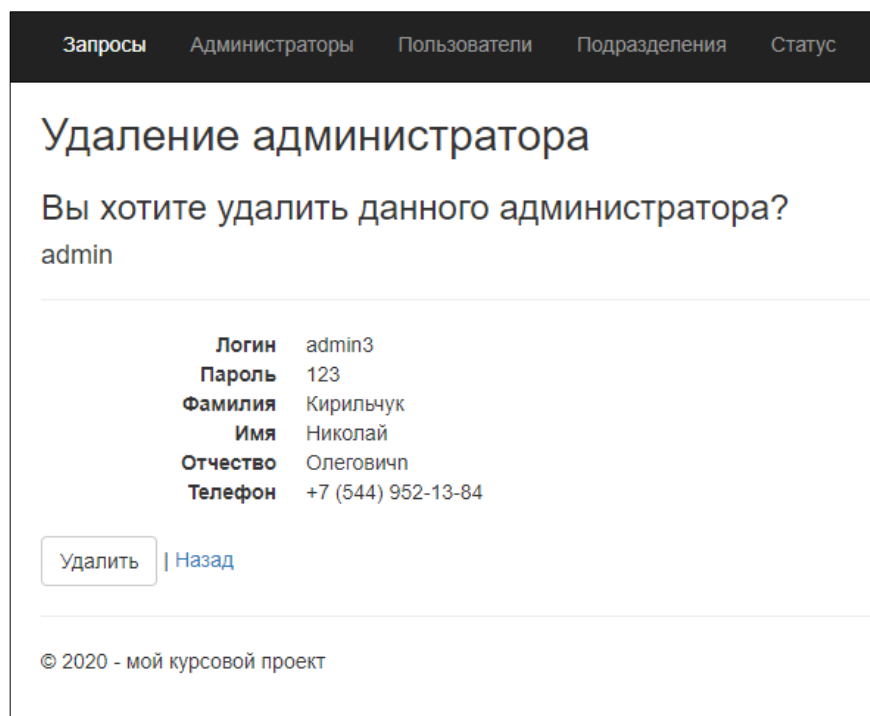


Рисунок 5.4 – Представление Delete для admin

```

@model WebApplication6.Models.admin
@{
    ViewBag.Title = "Удаление администратора";
}
<h2>Удаление администратора</h2>
<h3>Вы хотите удалить данного администратора?</h3>
<div>
    <h4>admin</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.login_admin)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.login_admin)

```

```

</dd>
<dt>
    @Html.DisplayNameFor(model => model.password_admin)
</dt>
<dd>
    @Html.DisplayFor(model => model.password_admin)
</dd>
<dt>
    @Html.DisplayNameFor(model => model.surname)
</dt>
<dd>
    @Html.DisplayFor(model => model.surname)
</dd>
<dt>
    @Html.DisplayNameFor(model => model.name)
</dt>
<dd>
    @Html.DisplayFor(model => model.name)
</dd>
<dt>
    @Html.DisplayNameFor(model => model.middlename)
</dt>
<dd>
    @Html.DisplayFor(model => model.middlename)
</dd>
<dt>
    @Html.DisplayNameFor(model => model.phone)
</dt>
<dd>
    @Html.DisplayFor(model => model.phone)
</dd>
</dl>
<dl class="dl-horizontal">

    @Html.ValidationSummary(true, "", new { @class = "text-danger" })
</dl>
@using (Html.BeginForm()) {
    @Html.AntiForgeryToken()

    <div class="form-actions no-color">
        <input type="submit" value="Удалить" class="btn btn-default" /> |
        @Html.ActionLink("Назад", "Index")
    </div>
}
</div>

```

5.5 Представление Create для admin

Запросы Администраторы Пользователи Подразделения Статус

Создание записи

Администратор

Логин

Пароль

Фамилия

Имя

Отчество

Телефон

[Назад](#)

Рисунок 5.5 – Представление Create для admin

```
@model WebApplication6.Models.admin
@{
    ViewBag.Title = "Создание администратора";
}
<h2>Создание записи</h2>
@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Администратор</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.login_admin, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.login_admin, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.login_admin, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.password_admin, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.password_admin, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.password_admin, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
```



```

        @Html.LabelFor(model => model.surname, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.surname, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.surname, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.name, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.name, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.name, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.middlename, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.middlename, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.middlename, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.phone, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.phone, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.phone, "Номер должен соответствовать маске +7 (____) ____-____", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Создать" class="btn btn-default" />
        </div>
    </div>
</div>
}

<div>
    @Html.ActionLink("Назад", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}

```

5.6 Представление Index для users

Запросы Администраторы Пользователи Подразделения Статус Запрос 1 Запрос 2						
Пользователи						
Создать						
Фамилия	Имя	Отчество	Телефон	Наименование подразделения	Логин	Пароль
Серов	Сергей	Олегович	+7 (489) 111-21-21	Подразделение 1	user12	123
Васечкин	Олег	Сергеевич	+7 (686) 797-80-01	Подразделение 1	vasya1	123
Мускатин	Игорь	Алексеевич	+7 (222) 888-99-12	Подразделение 1	kkk12	123
Пупкин	Леша	Олегович	+7 (654) 664-36-31	Подразделение 3	leha1	123
Саваельев	Игорь	Алексеевичasd	+7 (000) 111-11-16	Подразделение 4	44444fs	4444
Изменить Подробнее Удалить						
Изменить Подробнее Удалить						
Изменить Подробнее Удалить						
Изменить Подробнее Удалить						
Изменить Подробнее Удалить						
© 2020 - мой курсовой проект						

Рисунок 5.6 – Представление Index для users

```
@model IEnumerable<WebApplication6.Models.users>

@{
    ViewBag.Title = "Пользователи";
}

<h2>Пользователи</h2>

<p>
    @Html.ActionLink("Создать", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.surname)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.name)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.middlename)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.phone)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.poddr.namePodr)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.login_user)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.password_user)
        </th>
    </tr>

    @foreach (var item in Model) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.surname)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.name)
```

```

</td>
<td>
    @Html.DisplayFor(modelItem => item.middlename)
</td>
<td>
    @Html.DisplayFor(modelItem => item.phone)
</td>
<td>
    @Html.DisplayFor(modelItem => item.poddr.namePodr)
</td>
<td>
    @Html.DisplayFor(modelItem => item.login_user)
</td>
<td>
    @Html.DisplayFor(modelItem => item.password_user)
</td>
<td>
    @Html.ActionLink("Изменить", "Edit", new { id=item.id_users }) |
    @Html.ActionLink("Подробнее", "Details", new { id=item.id_users }) |
    @Html.ActionLink("Удалить", "Delete", new { id=item.id_users })
</td>
</tr>
}
</table>

```

5.7 Представление Edit для users

Рисунок 5.7 – Представление Edit для users

```

@model WebApplication6.Models.users

@{
    ViewBag.Title = "Изменение пользователя";
}

<h2>Изменение пользователя</h2>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

```

```

<div class="form-horizontal">
  <h4>Пользователь</h4>
  <hr />
  @Html.ValidationSummary(true, "", new { @class = "text-danger" })
  @Html.HiddenFor(model => model.id_users)

  <div class="form-group">
    @Html.LabelFor(model => model.podrid_podr, "Подразделение", htmlAttributes: new
{ @class = "control-label col-md-2" })
    <div class="col-md-10">
      @Html.DropDownList("podrid_podr", null, htmlAttributes: new { @class =
"form-control" })
      @Html.ValidationMessageFor(model => model.podrid_podr, "", new { @class =
"text-danger" })
    </div>
  </div>

  <div class="form-group">
    @Html.LabelFor(model => model.login_user, htmlAttributes: new { @class = "con-
trol-label col-md-2" })
    <div class="col-md-10">
      @Html.EditorFor(model => model.login_user, new { htmlAttributes = new {
@class = "form-control" } })
      @Html.ValidationMessageFor(model => model.login_user, "", new { @class =
"text-danger" })
    </div>
  </div>

  <div class="form-group">
    @Html.LabelFor(model => model.password_user, htmlAttributes: new { @class =
"control-label col-md-2" })
    <div class="col-md-10">
      @Html.EditorFor(model => model.password_user, new { htmlAttributes = new {
@class = "form-control" } })
      @Html.ValidationMessageFor(model => model.password_user, "", new { @class =
"text-danger" })
    </div>
  </div>

  <div class="form-group">
    @Html.LabelFor(model => model.surname, htmlAttributes: new { @class = "control-
label col-md-2" })
    <div class="col-md-10">
      @Html.EditorFor(model => model.surname, new { htmlAttributes = new { @class
= "form-control" } })
      @Html.ValidationMessageFor(model => model.surname, "", new { @class = "text-
danger" })
    </div>
  </div>

  <div class="form-group">
    @Html.LabelFor(model => model.name, htmlAttributes: new { @class = "control-la-
bel col-md-2" })
    <div class="col-md-10">
      @Html.EditorFor(model => model.name, new { htmlAttributes = new { @class =
"form-control" } })
      @Html.ValidationMessageFor(model => model.name, "", new { @class = "text-
danger" })
    </div>
  </div>

  <div class="form-group">
    @Html.LabelFor(model => model.middlename, htmlAttributes: new { @class = "con-
trol-label col-md-2" })

```

```

        <div class="col-md-10">
            @Html.EditorFor(model => model.middlename, new { htmlAttributes = new {
@class = "form-control" } })
            @Html.ValidationMessageFor(model => model.middlename, "", new { @class =
"text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.phone, htmlAttributes: new { @class = "control-
label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.phone, new { htmlAttributes = new { @class =
"form-control" } })
            @Html.ValidationMessageFor(model => model.phone, "", new { @class = "text-
danger" })
        </div>
    </div>

    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Изменить" class="btn btn-default" />
        </div>
    </div>
</div>
}

<div>
    @Html.ActionLink("Назад", "Index")
</div>

```

5.8 Представление Details для users

Пользователь	
Логин	user12
Пароль	123
Фамилия	Серов
Имя	Сергей
Отчество	Олегович
Телефон	+7 (489) 111-21-21
Наименование подразделения	Подразделение 1

[Изменить](#) | [Назад](#)

Рисунок 5.8 – Представление Details для users

```

@model WebApplication6.Models.users

@{
    ViewBag.Title = "Удаление пользователя";
}

<h2>Удаление пользователя</h2>

<div>
    <h4>Пользователь</h4>
    <hr />

```

```

<dl class="dl-horizontal">
  <dt>
    @Html.DisplayNameFor(model => model.login_user)
  </dt>

  <dd>
    @Html.DisplayFor(model => model.login_user)
  </dd>

  <dt>
    @Html.DisplayNameFor(model => model.password_user)
  </dt>

  <dd>
    @Html.DisplayFor(model => model.password_user)
  </dd>

  <dt>
    @Html.DisplayNameFor(model => model.surname)
  </dt>

  <dd>
    @Html.DisplayFor(model => model.surname)
  </dd>

  <dt>
    @Html.DisplayNameFor(model => model.name)
  </dt>

  <dd>
    @Html.DisplayFor(model => model.name)
  </dd>

  <dt>
    @Html.DisplayNameFor(model => model.middlename)
  </dt>

  <dd>
    @Html.DisplayFor(model => model.middlename)
  </dd>

  <dt>
    @Html.DisplayNameFor(model => model.phone)
  </dt>

  <dd>
    @Html.DisplayFor(model => model.phone)
  </dd>

  <dt>
    @Html.DisplayNameFor(model => model.podr.namePodr)
  </dt>

  <dd>
    @Html.DisplayFor(model => model.podr.namePodr)
  </dd>

</dl>
</div>
<p>
  @Html.ActionLink("Изменить", "Edit", new { id = Model.id_users }) |
  @Html.ActionLink("Назад", "Index")
</p>

```


5.9 Представление Delete для users

Запросы Администраторы Пользователи Подразделения Статус Запрос 1 Запрос 2

Удаление пользователя

Вы хотите удалить данного пользователя?

Пользователь

Логин	user12
Пароль	123
Фамилия	Серов
Имя	Сергей
Отчество	Олегович
access	0
Телефон	+7 (489) 111-21-21
Наименование подразделения	Подразделение 1

| [Назад](#)

Рисунок 5.9 – Представление Delete для users

```
@model WebApplication6.Models.users

@{
    ViewBag.Title = "Удаление пользователя";
}

<h2>Удаление пользователя</h2>

<h3>Вы хотите удалить данного пользователя?</h3>
<div>
    <h4>Пользователь</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.login_user)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.login_user)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.password_user)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.password_user)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.surname)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.surname)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.name)
        </dt>
```

```

        <dd>
            @Html.DisplayFor(model => model.name)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.middlename)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.middlename)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.access)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.access)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.phone)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.phone)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.podr.namePodr)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.podr.namePodr)
        </dd>
    </dl>
    <dl class="dl-horizontal">

        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
    </dl>

    @using (Html.BeginForm()) {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Удалить" class="btn btn-default" /> |
            @Html.ActionLink("Назад", "Index")
        </div>
    }
</div>

```

5.10 Представление Create для users

Запросы Администраторы Пользователи Подразделения Статус Запрос 1 Запрос 2

Создание пользователя

Пользователь

Наименование подразделения

Логин

Пароль

Фамилия

Имя

Отчество

Телефон

[Назад](#)

Рисунок 5.10 – Представление Create для users

```
@model WebApplication6.Models.users

@{
    ViewBag.Title = "Создание пользователя";
}

<h2>Создание пользователя</h2>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Пользователь</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.podr_id_podr, "Наименование подразделения",
htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("podr_id_podr", null, htmlAttributes: new { @class =
"text-danger" })
                @Html.ValidationMessageFor(model => model.podr_id_podr, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.login_user, htmlAttributes: new { @class = "con-
trol-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.login_user, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.login_user, "", new { @class =
"text-danger" })
            </div>
        </div>
    </div>
```

```

        <div class="form-group">
            @Html.LabelFor(model => model.password_user, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.password_user, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.password_user, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.surname, htmlAttributes: new { @class = "control-
label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.surname, new { htmlAttributes = new { @class
= "form-control" } })
                @Html.ValidationMessageFor(model => model.surname, "", new { @class = "text-
danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.name, htmlAttributes: new { @class = "control-la-
bel col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.name, new { htmlAttributes = new { @class =
"form-control" } })
                @Html.ValidationMessageFor(model => model.name, "", new { @class = "text-
danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.middlename, htmlAttributes: new { @class = "con-
trol-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.middlename, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.middlename, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.phone, htmlAttributes: new { @class = "control-
label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.phone, new { htmlAttributes = new { @class =
"form-control" } })
                @Html.ValidationMessageFor(model => model.phone, "", new { @class = "text-
danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Создать" class="btn btn-default" />
            </div>
        </div>
    }
</div>
    @Html.ActionLink("Назад", "Index")
</div>

```

5.11 Представление Index для quest1

[Запросы](#) [Администраторы](#) [Пользователи](#) [Подразделения](#) [Статус](#) [Запрос 1](#) [Запрос 2](#)

Запрос 1

Список заявок выполненных администратором за определенный промежуток времени

Статус заявки

В работе

ФИО администратора

Кирильчук

Дата от

01.10.2020

Дата до

24.12.2020

Найти

© 2020 - мой курсовой проект

Рисунок 5.11 – Представление Index для quest1

```
@model WebApplication6.Models.Запрос1_Result

@{
    ViewBag.Title = "Запрос 1";
}
<h2>Запрос 1</h2>

@using (Html.BeginForm("query1", "quest1"))
{
    <div class="form-group">
        <label1>Список заявок выполненных администратором за определенный промежуток времени</label1>
    </div>
    @Html.ValidationSummary(true, "", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(model => model.Статус_заявки, "Статус заявки", htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.DropDownList("id_status", null, htmlAttributes: new { @class = "form-control" })
            @Html.ValidationMessageFor(model => model.Статус_заявки, "", new { @class = "text-danger" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.ФИО_администратора, "ФИО администратора", htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.DropDownList("id_admin", null, htmlAttributes: new { @class = "form-control" })
            @Html.ValidationMessageFor(model => model.ФИО_администратора, "", new { @class = "text-danger" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.Время_заявки, "Дата от", htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            <input id="time1" type="date" class="t" name="time1" />
        </div>
    </div>
</script>
```

```

        document.getElementById('time1').valueAsDate = new Date();
    </script>
    <div class="form-group">
        @Html.LabelFor(model => model.Время_заявки, "Дата до", htmlAttributes: new { @class
= "control-label col-md-2" })
        <div class="col-md-10">
            <input id="time2" type="date" class="t" name="time2" />
        </div>
    </div>
    <script>
        document.getElementById('time2').valueAsDate = new Date();
    </script>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Найти" class="btn btn-primary" />
        </div>
    </div>
}

```

5.12 Представление query1 для quest1

Запросы	Администраторы	Пользователи	Подразделения	Статус	Запрос 1	Запрос 2
Результат запроса 1						
№ запроса	Статус заявки	Заявка	ФИО пользователя	Подразделение	Время заявки	Комментарий администратора
1	В работе	запрос тест	Серов Сергей Олегович	Подразделение 1	26.11.2020 10:24:47	
2002	В работе	Запрос 123	Серов Сергей Олегович	Подразделение 1	08.12.2020 10:43:15	ррррр
5003	В работе	запрос тест	Пупкин Леша Олегович	Подразделение 3	22.12.2020 7:57:08	
7005	В работе	запрос тест	Серов Сергей Олегович	Подразделение 1	22.12.2020 9:56:42	
Вернуться к Запросу 1						

Рисунок 5.12 – Представление query1 для quest1

```

@model IEnumerable<WebApplication6.Models.Запрос1_Result>
@{
    ViewBag.Title = "Запрос 1";
}
<h2>Результат запроса 1</h2>
<table class="table">
    <tr>
        <th>
            <p class="h5">№ запроса</p>
        </th>
        <th>
            <p class="h5">Статус заявки</p>
        </th>
        <th>
            <p class="h5">Заявка</p>
        </th>
        <th>
            <p class="h5">ФИО пользователя</p>
        </th>
        <th>
            <p class="h5">Подразделение</p>
        </th>
        <th>
            <p class="h5">Время заявки</p>
        </th>
        <th>
            <p class="h5">Комментарий администратора</p>
        </th>
    </tr>

```

```

</tr>
@foreach (var item in Model)
{
    <tr>
        <td scope="col">
            @Html.DisplayFor(modelItem => item.C_)
        </td>
        <td scope="col">
            @Html.DisplayFor(modelItem => item.Статус_заявки)
        </td>
        <td scope="col">
            @Html.DisplayFor(modelItem => item.Заявка)
        </td>
        <td scope="col">
            @Html.DisplayFor(modelItem => item.ФИО_пользователя)
        </td>
        <td scope="col">
            @Html.DisplayFor(modelItem => item.Подразделение)
        </td>
        <td scope="col">
            @Html.DisplayFor(modelItem => item.Время_заявки)
        </td>
        <td scope="col">
            @Html.DisplayFor(modelItem => item.Комментарий_администратора)
        </td>
    </tr>
}
</table>

<div>
    @Html.ActionLink("Вернуться к Запросы 1", "Index")
</div>

```

6 Контроль

Контроль данных – процесс, используемый для определения, являются ли данные точными, полными и отвечающими заданным критериям. В созданном Web-приложении он осуществляется с помощью атрибута контроля [Required] – контроль наполненности поля и [RegularExpression] – контроль на регулярное выражение, предназначен в классе для контроля ввода телефона, при объявлении модели. Также для контроля вводимой информации используется язык LINQ (прим. `logcheck = (from persons in db.users where persons.login_user == admin.login_admin select persons).Count()`; часть запроса из контроллера `admin` – 4.1, отслеживающая совпадение полей в модели, которые не могут быть одинаковыми, а именно Логина и Телефона).

Отображение сообщения об ошибке происходит благодаря использованию хелпера контроля `Html.ValidationMessageFor`, при ошибках, обработанных с помощью LINQ текст ошибки передается в `Html.ValidationSummary` с помощью конструкции `ModelState.AddModelError`.

При добавлении нового пользователя/администратора контролю подвергаются следующие вводимые данные: ФИО пользователя/администратора, пароль, логин, номер телефона на незаполненные значения, номер телефона, также, проверяется на соответствие маске: +7 (____) ____-__-__.

6.1 Модель `admin` с валидированными полями

```
namespace WebApplication6.Models
{
    using System;
    using System.Collections.Generic;
    using System.ComponentModel.DataAnnotations;
    public partial class admin
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]
        public admin()
        {
            this.requests = new HashSet<requests>();
        }
        public int id_admin { get; set; }
        [Required(ErrorMessage = "Заполните поле")]
        [Display(Name = "Логин")]
        public string login_admin { get; set; }
        [Required(ErrorMessage = "Заполните поле")]
        [Display(Name = "Пароль")]
        public string password_admin { get; set; }
        [Required(ErrorMessage = "Заполните поле")]
        [Display(Name = "Фамилия")]
        public string surname { get; set; }
        [Required(ErrorMessage = "Заполните поле")]
        [Display(Name = "Имя")]
        public string name { get; set; }
        [Required(ErrorMessage = "Заполните поле")]
        [Display(Name = "Отчество")]
        public string middlename { get; set; }
        [Required(ErrorMessage = "Заполните поле")]
        [RegularExpression(@"^+\d{1,2}\s+?(\d{3,5}\)\s+?\d{1,3}-\d{2}-\d{2}$",
            ErrorMessage = "Номер должен соответствовать маске +7 (____) ____-__-__")]
        [Display(Name = "Телефон")]
        public string phone { get; set; }
    }
}
```



```

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]
public virtual ICollection<requests> requests { get; set; }
}
}

```

6.2 Модель users с валидированными полями

```

namespace WebApplication6.Models
{
    using System;
    using System.Collections.Generic;
    using System.ComponentModel.DataAnnotations;
    public partial class users
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]
        public users()
        {
            this.requests = new HashSet<requests>();
        }
        public int id_users { get; set; }
        public int podr_id_podr { get; set; }
        [Required(ErrorMessage = "Заполните поле")]
        [Display(Name = "Логин")]
        public string login_user { get; set; }
        [Required(ErrorMessage = "Заполните поле")]
        [Display(Name = "Пароль")]
        public string password_user { get; set; }
        [Required(ErrorMessage = "Заполните поле")]
        [Display(Name = "Фамилия")]
        public string surname { get; set; }
        [Required(ErrorMessage = "Заполните поле")]
        [Display(Name = "Имя")]
        public string name { get; set; }
        [Required(ErrorMessage = "Заполните поле")]
        [Display(Name = "Отчество")]
        public string middlename { get; set; }
        public byte access { get; set; }
        [Required(ErrorMessage = "Заполните поле")]
        [RegularExpression(@"^\+\d{1,2}s+?(\d{3,5}\)\s+?\d{1,3}-\d{2}-\d{2}$",
            ErrorMessage = "Номер должен соответствовать маске +7 (____) ____-__-__")]
        [Display(Name = "Телефон")]
        public string phone { get; set; }
        public virtual podr podr { get; set; }
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<requests> requests { get; set; }
    }
}

```

6.3 Контроллер users с валидированными полями

Проверка на совпадения уже зарегистрированных логинов и телефонов с помощью LINQ.

```

public ActionResult Create([Bind(Include = "id_users,podr_id_podr,login_user,password_user,surname,name,middlename,access,phone")] users users)
{
    int logcheck = 0;
    int phonecheck = 0;
    logcheck = (from persons in db.admin where persons.login_admin == users.login_user select persons).Count();
    logcheck += (from persons in db.users where persons.id_users != users.id_users where persons.login_user == users.login_user select persons).Count();
    phonecheck = (from persons in db.admin where persons.phone == users.phone select persons).Count();
}

```

```

        phonecheck += (from persons in db.users where persons.id_users != users.id_users
where persons.phone == users.phone select persons).Count();

        if (ModelState.IsValid && phonecheck == 0 && logcheck == 0)
        {
            db.users.Add(users);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        if (phonecheck != 0) { ModelState.AddModelError(String.Empty, "Пользователь с
таким телефоном уже существует"); }
        if (logcheck != 0) { ModelState.AddModelError(String.Empty, "Пользователь с та-
ким логином уже существует"); }

        ViewBag.podr_id_podr = new SelectList(db.podr, "id_podr", "namePodr", us-
ers.podr_id_podr);
        return View(users);
    }

```

7 Таблица тестов

Таблица тестов представлена в таблице 1.

Таблица 1 – Тесты

№	Таблица	Тест	Входные данные	Выходные данные
1	Пользователи	Добавление	Информация о пользователе (см. рисунок 8.1)	Таблица с добавленным пользователем (см. рисунок 8.2)
2		Удаление	Выбранный пользователь (см. рисунок 8.3)	Таблица без удаленного пользователя (см. рисунок 8.4)
3		Правка	Новая информация об пользователе (см. рисунок 8.5)	Таблица с отредактированным пользователем (см. рисунок 8.6)
4		Просмотр	Выбранный пользователь	Подробная информация о пользователе (см. рисунок 8.7)
5	Администраторы	Добавление	Информация о администраторе (см. рисунок 8.8)	Таблица с добавленным администратором (см. рисунок 8.9)
6		Удаление	Выбранный администратор (см. рисунок 8.10)	Таблица без удаленного администратора (см. рисунок 8.11)
7		Правка	Новая информация о администраторе (см. рисунок 8.12)	Таблица с отредактированным администратором (см. рисунок 8.13)
8		Просмотр	Выбранный администратор	Подробная информация о администраторе (см. рисунок 8.14)

Продолжение таблицы 1

№	Таблица	Тест	Входные данные	Выходные данные
9	Подразделение	Добавление	Информация о подразделении (см. рисунок 8.15)	Таблица с добавленным подразделением (см. рисунок 8.16)
10		Удаление	Выбранное подразделение (см. рисунок 8.17)	Таблица без удаленного подразделения (см. рисунок 8.18)
11	Заявки	Добавление	Информация о заявке (см. рисунок 8.19)	Таблица с добавленной заявкой (см. рисунок 8.20)
12		Правка	Новая информация о заявке (см. рисунок 8.21)	Таблица с отредактированной заявкой (см. рисунок 8.22)
13	Запрос 1	Показ	Введенный период дат, выбранный статус и администратор (см. рисунок 8.23)	Информация статусных заявках администратора за период времени (см. рисунок 8.24)
14	Запрос 2	Показ	Выбранный пользователь (см. рисунок 8.25)	Информация заявках пользователя (см. рисунок 8.26)
15	Пользователь	Проверка наполненности полей	Неверное введенная информация, не заполненные поля	Сообщения о некорректности информации (см. рисунок 8.27)
16		Проверка корректности введенного номера	Введение номера, не соответствующего маске	Сообщения о некорректности номера (см. рисунок 8.28)
17		Проверка совпадения с существующими номером или логином	Введение номера уже зарегистрированного в системе	Сообщения о совпадении номеров (см. рисунок 8.29)

Продолжение таблицы 1

№	Таблица	Тест	Входные данные	Выходные данные
18	Подразделение	Проверка удаление связанных данных	Запись подразделения, содержащая пользователей	Сообщение о ошибке (см. рисунок 8.30)

8 Результаты тестирования

В данной главе отображены результаты тестирования созданного Web-приложения по таблице 1, которые представлены на рисунках 8.1-8.30.

8.1 Результат теста 1

Создание пользователя

Пользователь

Наименование подразделения

Подразделение 3

Логин

Polzovatel123

Пароль

123123

Заполните поле

Фамилия

Павликов

Имя

Михаил

Отчество

Сергеевич

Телефон

+7 (456) 789-98-76

Создать

Рисунок 8.1 – Входные данные теста №1

Пользователи						
Создать						
Фамилия	Имя	Отчество	Телефон	Наименование подразделения	Логин	Пароль
Серов	Сергей	Олегович	+7 (489) 111-21-21	Подразделение 1	user12	123
Васечкин	Олег	Сергеевич	+7 (686) 797-80-01	Подразделение 1	vasya1	123
Мускатин	Игорь	Алексеевич	+7 (222) 888-99-12	Подразделение 1	kkk12	123
Пупкин	Леша	Олегович	+7 (654) 664-36-31	Подразделение 3	leha1	123
Павликов	Михаил	Сергеевич	+7 (456) 789-98-76	Подразделение 3	Polzovatel123	123123
Саваельев	Игорь	Алексеевича	+7 (000) 111-11-16	Подразделение 4	44444fs	4444
© 2020 - мой курсовой проект						

Рисунок 8.2 – Выходные данные теста №1

8.2 Результат теста 2

Удаление пользователя

Вы хотите удалить данного пользователя?

Пользователь

Логин	Polzovatel123
Пароль	123123
Фамилия	Павликов
Имя	Михаил
Отчество	Сергеевич
Телефон	+7 (456) 789-98-76
Наименование подразделения	Подразделение 3

| [Назад](#)

Рисунок 8.3 – Входные данные теста №2

Пользователи							
Создать							
Фамилия	Имя	Отчество	Телефон	Наименование подразделения	Логин	Пароль	
Серов	Сергей	Олегович	+7 (489) 111-21-21	Подразделение 1	user12	123	Изменить Подробнее Удалить
Васечкин	Олег	Сергеевич	+7 (686) 797-80-01	Подразделение 1	vasya1	123	Изменить Подробнее Удалить
Мускатин	Игорь	Алексеевич	+7 (222) 888-99-12	Подразделение 1	kkk12	123	Изменить Подробнее Удалить
Пупкин	Леша	Олегович	+7 (654) 664-36-31	Подразделение 3	leha1	123	Изменить Подробнее Удалить
Саваельев	Игорь	Алексеевича	+7 (000) 111-11-16	Подразделение 4	44444fs	4444	Изменить Подробнее Удалить

Рисунок 8.4 – Выходные данные теста №2

8.3 Результат теста 3

Изменение пользователя

Пользователь

Подразделение	Подразделение 4
Логин	44444fs
Пароль	4444
Фамилия	Саваельев
Имя	Игорь
Отчество	Алексеевича
Телефон	+7 (000) 111-11-16

Рисунок 8.5 – Входные данные теста №3

Пользователи							
Создать							
Фамилия	Имя	Отчество	Телефон	Наименование подразделения	Логин	Пароль	
Серов	Сергей	Олегович	+7 (489) 111-21-21	Подразделение 1	user12	123	Изменить Подробнее Удалить
Васечкин	Олег	Сергеевич	+7 (686) 797-80-01	Подразделение 1	vasya1	123	Изменить Подробнее Удалить
Мускатин	Игорь	Алексеевич	+7 (222) 888-99-12	Подразделение 1	kkk12	123	Изменить Подробнее Удалить
Пупкин	Леша	Олегович	+7 (654) 664-36-31	Подразделение 3	leha1	123	Изменить Подробнее Удалить
Савельев	Игорь	Алексеевича	+7 (000) 111-11-16	Подразделение 3	savell76	4444	Изменить Подробнее Удалить

Рисунок 8.6 – Выходные данные теста №3

8.4 Результат теста 4

Подробнее о пользователе

Пользователь

Логин

kkk12

Пароль

123

Фамилия

Мускатин

Имя

Игорь

Отчество

Алексеевич

Телефон

+7 (222) 888-99-12

Наименование подразделения

Подразделение 1

[Изменить](#) | [Назад](#)

Рисунок 8.7 – Выходные данные теста №4

8.5 Результат теста 5

Создание записи

Администратор

Логин

admin789

Пароль

987789

Фамилия

Сорокин

Имя

Алексей

Отчество

Иванович

Телефон

+7 (456) 789-98-78

Создать

[Назад](#)

Рисунок 8.8 – Входные данные теста №5

Администраторы						
Создать						
Логин	Пароль	Фамилия	Имя	Отчество	Телефон	
admin3	123	Кирильчук	Николай	Олегович	+7 (544) 952-13-84	Изменить Подробнее Удалить
admin1	123	Петров	Василий	Иванович	+7 (234) 632-46-41	Изменить Подробнее Удалить
admin11	123	Сорокин	Василий	Олегович	+7 (345) 765-88-67	Изменить Подробнее Удалить
kkk222	123	Павлов	Игорь	Алексеевич	+7 (999) 777-77-77	Изменить Подробнее Удалить
admin12311	111	Мускатин	Игорь	Алексеевич	+7 (000) 111-11-11	Изменить Подробнее Удалить
admin789	987789	Сорокин	Алексей	Иванович	+7 (456) 789-98-78	Изменить Подробнее Удалить

Рисунок 8.9 – Выходные данные теста №5

8.6 Результат теста 6

Удаление администратора

Вы хотите удалить данного администратора?

Администратор

Логин

admin12311

Пароль

111

Фамилия

Мускатин

Имя

Игорь

Отчество

Алексеевич

Телефон

+7 (000) 111-11-11

Удалить

Назад

Рисунок 8.10 – Входные данные теста №6

Администраторы						
Создать						
Логин	Пароль	Фамилия	Имя	Отчество	Телефон	
admin3	123	Кирильчук	Николай	Олегович	+7 (544) 952-13-84	Изменить Подробнее Удалить
admin1	123	Петров	Василий	Иванович	+7 (234) 632-46-41	Изменить Подробнее Удалить
admin11	123	Сорокин	Василий	Олегович	+7 (345) 765-88-67	Изменить Подробнее Удалить
kkk222	123	Павлов	Игорь	Алексеевич	+7 (999) 777-77-77	Изменить Подробнее Удалить
admin789	987789	Сорокин	Алексей	Иванович	+7 (456) 789-98-78	Изменить Подробнее Удалить

Рисунок 8.11 – Выходные данные теста №6

8.7 Результат теста 7

Изменение администратора

Администратор

Логин	<input type="text" value="admin8585"/>
Пароль	<input type="text" value="147896523"/>
Фамилия	<input type="text" value="Сорокин"/>
Имя	<input type="text" value="Василий"/>
Отчество	<input type="text" value="Олегович"/>
Телефон	<input type="text" value="+7 (345) 765-88-67"/>
<input type="button" value="Изменить"/>	

Рисунок 8.12 – Входные данные теста №7

Администраторы						
Создать						
Логин	Пароль	Фамилия	Имя	Отчество	Телефон	
admin3	123	Кирильчук	Николай	Олегович	+7 (544) 952-13-84	Изменить Подробнее Удалить
admin1	123	Петров	Василий	Иванович	+7 (234) 632-46-41	Изменить Подробнее Удалить
admin8585	147896523	Сорокин	Василий	Олегович	+7 (345) 765-88-67	Изменить Подробнее Удалить
kkk222	123	Павлов	Игорь	Алексеевич	+7 (999) 777-77-77	Изменить Подробнее Удалить
admin789	987789	Сорокин	Алексей	Иванович	+7 (456) 789-98-78	Изменить Подробнее Удалить

Рисунок 8.13 – Выходные данные теста №7

8.8 Результат теста 8

Подробнее об администраторе

Администратор

Логин	admin1
Пароль	123
Фамилия	Петров
Имя	Василий
Отчество	Иванович
Телефон	+7 (234) 632-46-41

[Изменить](#) | [Назад](#)

Рисунок 8.14 – Выходные данные теста №8

8.9 Результат теста 9

Создание подразделения

Подразделение

Наименование подразделения	<input type="text" value="Подразделение 5"/>
<input type="button" value="Создать"/>	

Рисунок 8.15 – Входные данные теста №9

Подразделения	
Создать подразделение	
Наименование подразделения	
Подразделение 1	Изменить Подробнее Удалить
Подразделение 2	Изменить Подробнее Удалить
Подразделение 3	Изменить Подробнее Удалить
Подразделение 4	Изменить Подробнее Удалить
Подразделение 5	Изменить Подробнее Удалить

Рисунок 8.16 – Выходные данные теста №9

8.10 Результат теста 10

Удаление подразделения

Вы хотите удалить данное подразделение?

Подразделение

Наименование подразделения	Подразделение 5
<input type="button" value="Удалить"/> Назад	

Рисунок 8.17 – Входные данные теста №10

Подразделения	
Создать подразделение	
Наименование подразделения	
Подразделение 1	Изменить Подробнее Удалить
Подразделение 2	Изменить Подробнее Удалить
Подразделение 3	Изменить Подробнее Удалить
Подразделение 4	Изменить Подробнее Удалить

Рисунок 8.18 – Выходные данные теста №10

8.11 Результат теста 11

Создать запрос

Запрос

Фамилия пользователя

Пупкин

Фамилия администратора

Павлов

Статус

Не завершен

Запрос

Запрос про запрос

Комментарий

Текст

Создать

Рисунок 8.19 – Входные данные теста №11

Запросы							
Создать запрос							
№ запроса	Запрос	Время запроса	Комментарий	Фамилия администратора	Статус	Фамилия пользователя	
9003	Запрос про запрос	24.12.2020 20:40:44	Текст	Павлов	Не завершен	Пупкин	Изменить Подробнее Удалить
8004	Запрос для проверки	22.12.2020 11:48:26	Тут комментарий	Кирильчук	Не завершен	Савельев	Изменить Подробнее Удалить
8003	запрос тест 111	22.12.2020 11:17:47		Павлов	В работе	Пупкин	Изменить Подробнее Удалить

Рисунок 8.20 – Выходные данные теста №11

8.12 Результат теста 12

Изменение запроса

Запросы

Пользователь

Пупкин

Администратор

Павлов

Статус

Завершен

Запрос

Вместо <запрос тест 111> стал ЗАПРО

Комментарий

Тут комментарий

Изменить

Рисунок 8.21 – Выходные данные теста №12

Запросы							
Создать запрос							
№ запроса	Запрос	Время запроса	Комментарий	Фамилия администратора	Статус	Фамилия пользователя	
9003	Запрос про запрос	24.12.2020 20:40:44	Текст	Павлов	Не завершен	Пупкин	Изменить Подробнее Удалить
8004	Запрос для проверки	22.12.2020 11:48:26	Тут комментарий	Кирильчук	Не завершен	Савельев	Изменить Подробнее Удалить
8003	Вместо <запрос тест 111> стал ЗАПРОС	22.12.2020 11:17:47	Тут комментарий	Павлов	Завершен	Пупкин	Изменить Подробнее Удалить

Рисунок 8.22 – Выходные данные теста №12

8.13 Результат теста 13

Запрос 1

Список заявок выполненных администратором за определенный промежуток времени

Статус заявки

Завершен

ФИО администратора

Кирильчук

Дата от

01.10.2020

Дата до

24.12.2020

Найти

© 2020 - мой курсовой проект

Рисунок 8.23 – Входные данные теста №13

Результат запроса 1						
№ запроса	Статус заявки	Заявка	ФИО пользователя	Подразделение	Время заявки	Комментарий администратора
2	Завершен	zaproz 2	Серов Сергей Олегович	Подразделение 1	26.11.2020 10:24:51	
2005	Завершен	Кто спер клавиатуру?	Пупкин Леша Олегович	Подразделение 3	10.12.2020 11:22:07	Холла

[Вернуться к Запросу 1](#)

Рисунок 8.24 – Выходные данные теста №13

8.14 Результат теста 14

Запрос 2

Список заявок определенного пользователя

Серов

Найти

© 2020 - мой курсовой проект

Рисунок 8.25 – Входные данные теста №14

Результат запроса 2			
№ запроса	Статус заявки	Заявка	Время заявки
1	В работе	запрос тест	26.11.2020 10:24:47
2	Завершен	zaproz 2	26.11.2020 10:24:51
2002	В работе	Запрос 123	08.12.2020 10:43:15
2004	В работе	фыафываg	08.12.2020 13:08:08
3002	Не завершен	От винта	17.12.2020 10:13:58
7003	Не завершен	запрос тест для запроса 1	22.12.2020 9:56:21
7004	Не завершен	запрос тест для запроса 1,1	22.12.2020 9:56:32
7005	В работе	запрос тест	22.12.2020 9:56:42

[Вернуться к Запросу 2](#)

Рисунок 8.26 – Выходные данные теста №14

8.15 Результат теста 15

Создание пользователя

Пользователь

Наименование подразделения

Подразделение 3

Логин

Polzovatel123

Пароль

Заполните поле

Фамилия

Павликов

Имя

Михаил

Отчество

Сергеевич

Телефон

+7 (456) 789-98-76

Создать

[Назад](#)

Рисунок 8.27 – Проверка наполненности полей

8.16 Результат теста 16

Создание пользователя

Пользователь

Наименование подразделения: Подразделение 3

Логин: Polzovatel123

Пароль: 123123

Фамилия: Павликов

Имя: Михаил

Отчество: Сергеевич

Телефон: +7 (456) 789-98-7

Номер должен соответствовать маске +7 (____) ____-__-__

Создать

Рисунок 8.28 – Данные введённые в поле телефона не соответствуют его маске

8.17 Результат теста 17

Создание пользователя

Пользователь

- Пользователь с таким телефоном уже существует
- Пользователь с таким логином уже существует

Наименование подразделения: Подразделение 1

Логин: kkk12

Пароль: 1111

Фамилия: Тест

Имя: Тест

Отчество: Тест

Телефон: +7 (489) 111-21-21

Создать

Рисунок 8.29 – Проверка совпадения логинов/телефонов с уже зарегистрированными

8.18 Результат теста 18

Удаление подразделения

Вы хотите удалить данное подразделение?

Подразделение

Наименование подразделения	
Подразделение 3	

- К данному подразделению относятся пользователи в количестве 2. Для удаления в подразделении не должно быть пользователей

| [Назад](#)

Рисунок 8.30 – Попытка удаления подразделения, в котором числятся сотрудники

Заключение

В ходе выполнения курсового проекта на тему «Разработка клиента по технологии MVC» по дисциплине «Управление данными» были получены навыки проектирования и разработки Web-приложения баз данных с использованием архитектурного принципа MVC и платформа ASP.NET MVC.

Для того, чтобы реализовать Web-приложение согласно указанному в варианте заданию, необходимо было освоить принцип разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных составляющих, называемых компонентами. Были изучены такие компоненты MVC как: модель, контроллер, представление.

В результате выполнения курсового проекта было спроектировано и разработано Web-приложение по индивидуальному варианту, выполняющее такие задачи, как: добавление, изменение, удаление, просмотр таблиц пользователей, администраторов, подразделений, статусов и заявок, выполнение запросов по выводу всех заявок выбранного статуса, привязанных к определенному администратору в выбранный промежуток времени и по выводу всех оставленных выбранным пользователем заявок.

Список использованных источников

1. Стасышин В. М., Стасышина Т. Л. Базы данных: Технологии доступа 2-е изд., испр. и доп. Учебное пособие для академического бакалавриата, 2019. 164 с. ISBN: 978-5-534-09888-4 Серия: Профессиональное образование. URL: <https://biblio-online.ru/viewer/bazy-dannyhtehnologii-dostupa-415342#page/1>.
2. Голицына О. Л. Базы данных: учебное пособие для вузов по направлению 230700 "Прикладная информатика" / О. Л. Голицына, Н. В. Максимов, И. И. Попов, 2012. 399 с.
3. Сосинская С. С. Использование языка С# в различных информационных технологиях: учебник для студентов вузов, обучающихся по направлению "Информационные системы и технологии" / Сосинская, 2014. 367 с.
4. Шубина М.А. Управление данными: учебное пособие для студентов направлений подготовки 09.03.02 и 09.04.02 «Информационные системы и технологии» / Шубина М.А., 2016. 132 с. ISBN: 978-5-9239-0832-9. URL: https://e.lanbook.com/book/74029#book_name.
5. Сосинская С.С. «Разработка клиента по технологии MVC»: Методические указания по выполнению курсового проекта» Электронный каталог кафедры вычислительной техники (дата обращения: 15.12.2020).
6. Основы LINQ // METANIT.COM. URL: <https://metanit.com/sharp/tutorial/15.1.php> (дата обращения: 15.12.2020).
7. Model First // METANIT.COM. URL: <https://metanit.com/sharp/entityframework/2.5.php> (дата обращения: 15.12.2020).
8. Руководство по ASP.NET MVC 5 // METANIT.COM. URL: <https://metanit.com/sharp/mvc5/> (дата обращения: 15.12.2020).