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
для информационной системы «Страховое агентство»

Выполнил:
студент группы ИСТ-116
Брагин Д.А.
Принял:
доцент Вершинин В.В.

Владимир, 2018

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ

Бесчисленное множество новых технологий, вызванных бурным ростом информатизации общества, делает нашу жизнь невозможной без быстрого доступа к информации. В наше время очень легко получить информацию, одним из способов быстрого доступа к ней является веб-приложение.

Создание приложение на сегодняшний день, становится одной из наиболее актуальных и востребованных услуг.

Целью данного курсового проекта является разработка программной системы «Страховое агентство» для упрощения взаимодействия страховых агентов и страхователей.

2 ПОСТАНОВКА ЗАДАЧИ

В процессе разработки прототипа программной системы необходимо:

1. выполнить исследование и анализ предметной области;
2. разработать прототип ПС;
3. выполнить моделирование работы ПС;
4. разработать схему базы данных;
5. реализовать ПС с использованием выбранных средств и технологий.

Исходные данные:

1. методология проектирования и разработки RUP;
2. язык моделирования UML;
3. платформа разработки ASP.NET;

Минимальный набор функций: поддержка различных типов пользователей, заполнение, проверка, подтверждение, отказ заявок.

3 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Деятельность компании организована следующим образом: на сайт компанию обращаются клиенты с целью заключения договора о страховании. В зависимости от принимаемых на страхование объектов и страхуемых рисков, договор заключается по определенному виду страхования. При заключении договора фиксируется дата заключения, страховая сумма, вид страхования, тариф.

3.1 Словарь предметной области

Пользователь – лицо использующее программную систему.

Клиент – лицо, приобретающее страховку.

Агент – лицо заключающее договор страхования.

Администратор – лицо нормализирующее работу программной системы.

Оформить – заключение нового договора страхования.

Тариф – информация о предоставляемых услугах.

Договор – документ, подтверждающий оформления страховки.

Страховой случай – происшествие со страхователем.

Страховая сумма – размер страхового взноса и предельную величину выплат при наступлении страхового случая.

3.2 Сценарий взаимодействия пользователей с системой

Каждый пользователь имеет определенную роль, которая определяет его возможности. Можно выделить следующие роли и их возможности:

Пользователь:

- регистрация
- авторизация

Страхователь:

- просмотр тарифов
- оформление договора
- оформление страхового случая

Агент:

- заключение договора

- поддержка справочников юр/физ. лиц

- поиск договоров

Администратор:

- поддержка справочника тарифов

- поддержка пользователей

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

4 ПРОЕКТИРОВАНИЕ СИСТЕМЫ

4.1 Создание диаграмм UML

4.1.1 Диаграмма прецедентов

1. **Название:** «Просмотр тарифов»

Действующее лицо: «Клиент»

Основной поток: Страхователь открывает страницу (окно приложения), отображающую действующие тарифы.

Альтернативный поток: -

Постусловие: Клиент просматривает доступные тарифы

2. **Название:** «Оформление договора»

Действующее лицо: «Страхователь»

Основной поток: Пользователь нажимает на кнопку «Оформить договор» следом открывается окно для внесения данных. Заявитель заполняет все поля.

Альтернативный поток: Неправильно заполнены данные.

Постусловие: Заявка на оформление договора создана.

3. **Название:** «Обращение по страховому случаю»

Действующее лицо: «Страхователь»

Основной поток: Пользователь нажимает кнопку «Оформить страховой случай» и заполняет все поля.

Альтернативный поток: Неправильно заполнены данные.

Постусловие: Заявка на оформление страхового случая создана.

4. **Название:** «Заключение договора»

Действующее лицо: «Агент»

Основной поток: Агент нажимает на кнопку Одобрить, тем самым принимает договор, заполненный клиентом.

Альтернативный поток: Агент отклоняет заявление клиента.

Постусловие: Заявка на заключение договора одобрена.

5. **Название:** «Поиск договора»

Действующее лицо: «Агент»

Основной поток: Пользователь вводит параметры для поиска нажимает на кнопку «Поиск» следом отображаются данные соответствующие поиску. При необходимости пользователь может просмотреть сведения по интересующему его договору.

Альтернативный поток: Неправильно заполнены данные.

Постусловие: Заявка на заключение договора создана.

6. Название: «Поддержка справочника тарифов»

Действующее лицо: «Администратор»

Основной поток: Добавление новых, и изменение действующих тарифов.

Альтернативный поток: -

Постусловие: -

7. Название: «Поддержка пользователей»

Действующее лицо: «Администратор»

Основной поток: Изменение прав пользователя.

Альтернативный поток: -

Постусловие: -

Диаграмма прецедентов представлена на рисунке 1.

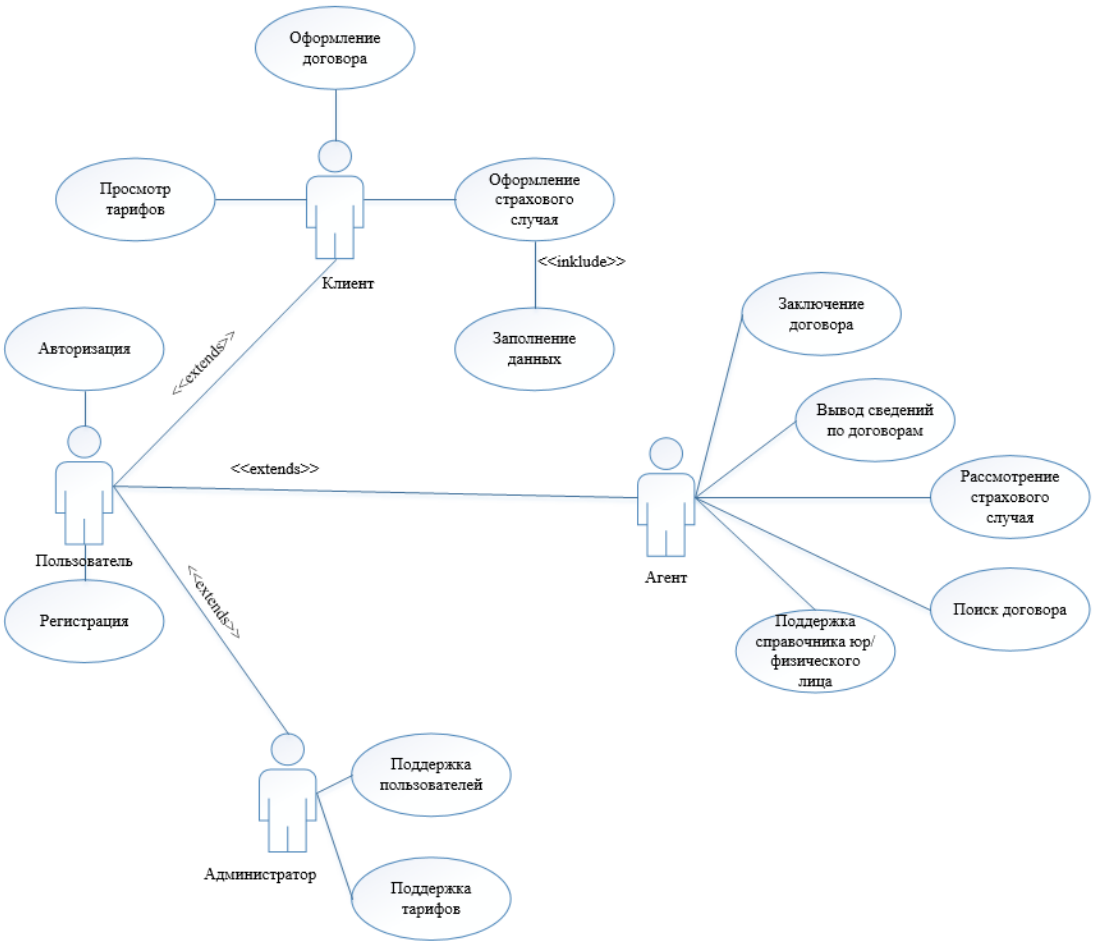

Рисунок 1. Диаграмма прецедентов

4.1.2 Диаграмма последовательностей

Диаграмма последовательностей для проверки заявки представлена на рисунке 2.

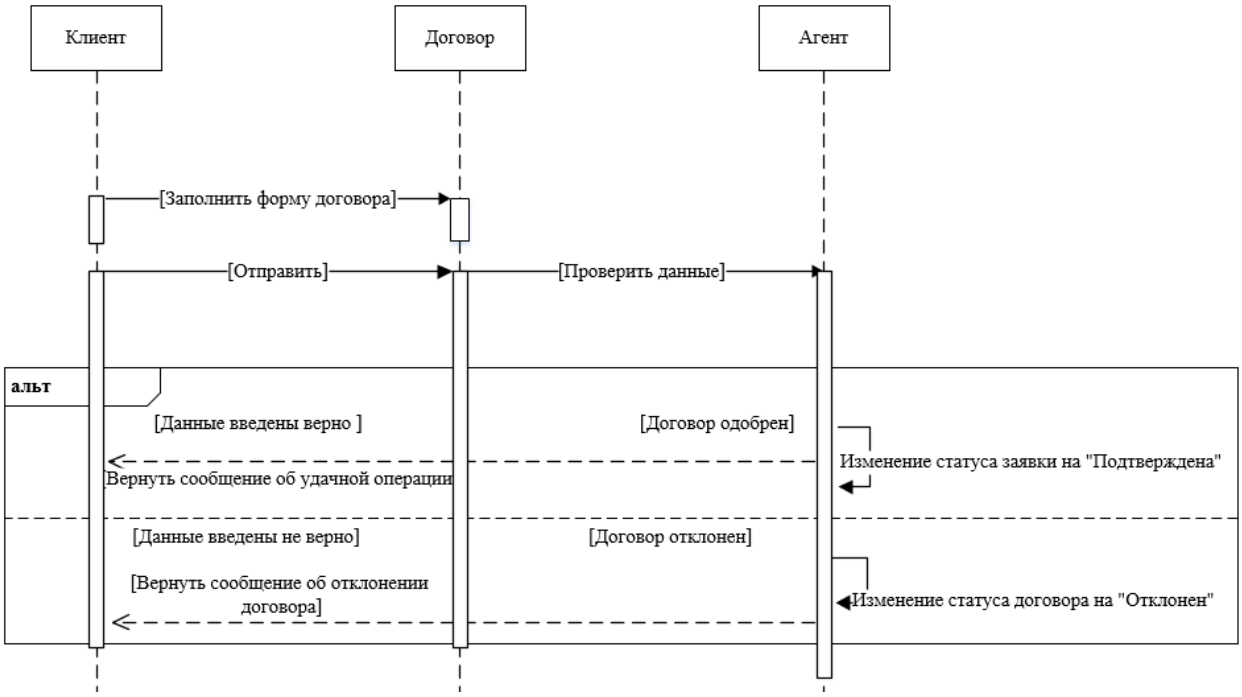

Рисунок 2. Диаграмма последовательностей для прецедента «Оформить договор»

4.1.3 Диаграмма состояний

Диаграмма состояний отображает жизненный цикл заявки на оформление договора.

Страхователь создает заявку на оформление договора и заполняет все необходимые поля. Администратор проверят верность введенных данных. Если информация заполнена неверно, то заявке присваивается статус «Отклонено, иначе присваивается статус «Одобрено». Заявка закрывается.

Диаграмма состояний объекта «Заявка» представлена на рисунке 2.

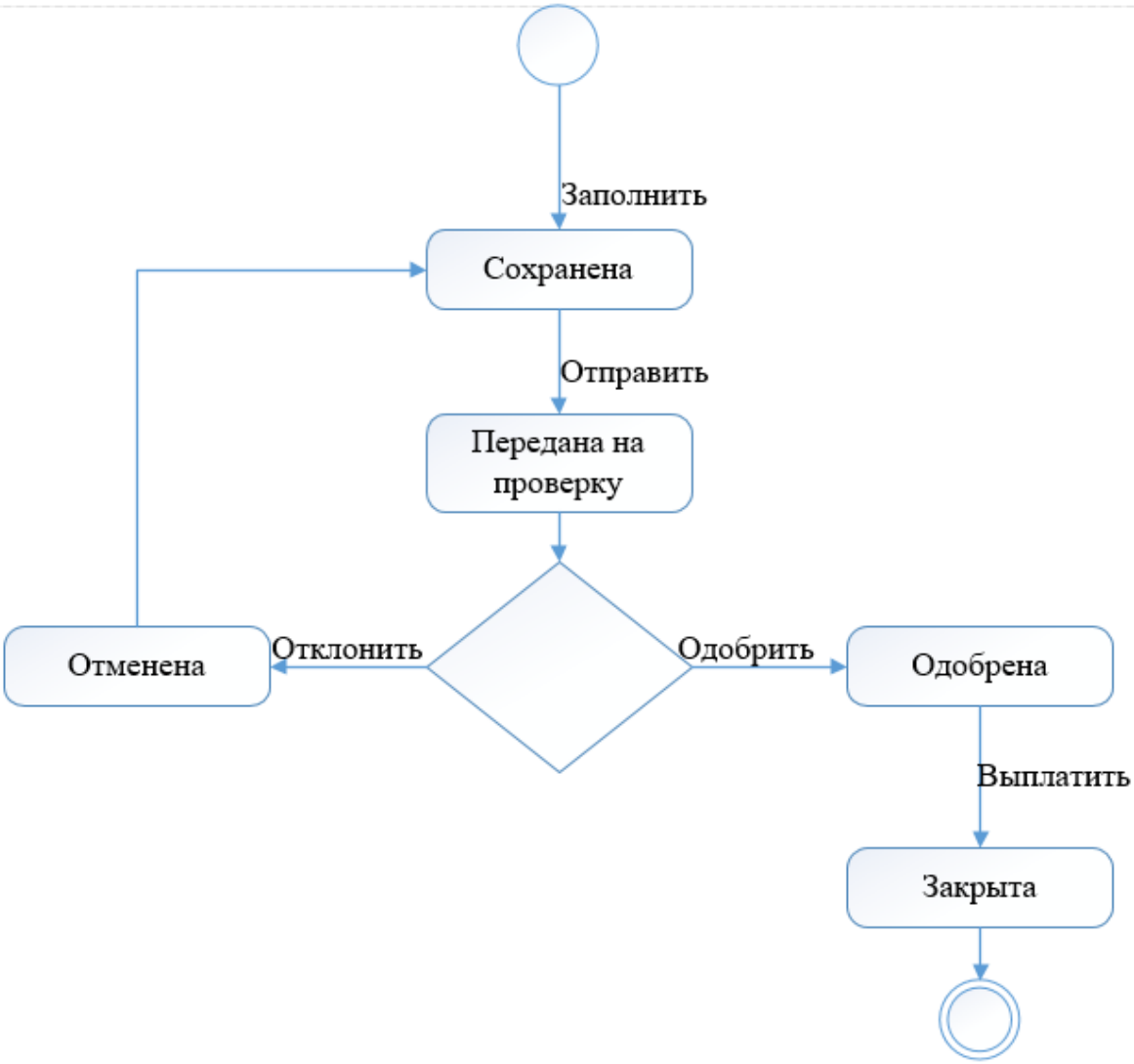

Рисунок 3. Диаграмма состояний заявки на оформление страхового случая.

4.1.4 Диаграмма классов

Диаграмма классов демонстрирует классы системы, их атрибуты, методы и взаимосвязи между ними.

Выделены 7 классов:

1. пользователь – родительский класс для классов «Пользователь», «Клиент» и «Агент», имеет переменные логин и пароль, имеет методы регистрация и авторизация (не авторизованный гость);
2. клиент – имеет переменные имя, фамилию и @mail, имеет методы оформление договора, посмотреть тарифы, оформление страхового случая (Авторизованный пользователь, клиент страхового агентства);
3. администратор – имеет переменные имя и фамилию, имеет методы поддержка тарифов, поддержка пользователей (администратор агентства);

4. агент – имеет переменные имя, фамилия и должность, имеет методы просмотр договоров, поддержка справочника юр/физ. лиц, рассмотрение страхового случая, заключение договора (сотрудник агентства);
5. тарифы – имеет переменные цена, объект договора и условия (услуги, предоставляемые агентством и их цены);
6. договор – имеет переменные номер договора, клиент, агент, тариф и дата (договор, заключаемый между клиентом и агентством);
7. страховой случай – имеет переменные клиент, номер договора, дата и описание (ситуация, при которой агентство выплачивает страховую сумму клиенту).

Диаграмма классов представлена на рисунке 4.

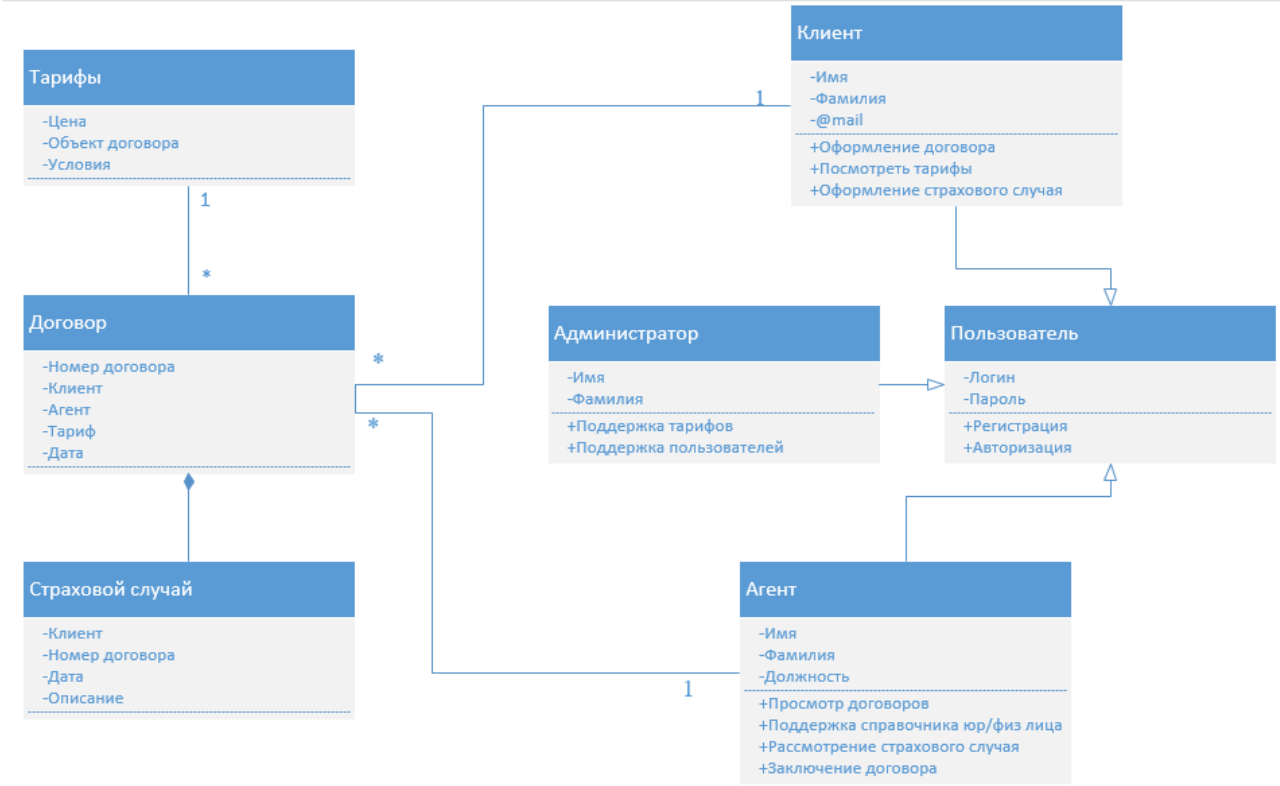

Рисунок 4. Диаграмма классов

4.1.5 Диаграмма развертывания

Диаграмма развертывания представлена на рисунке 5.

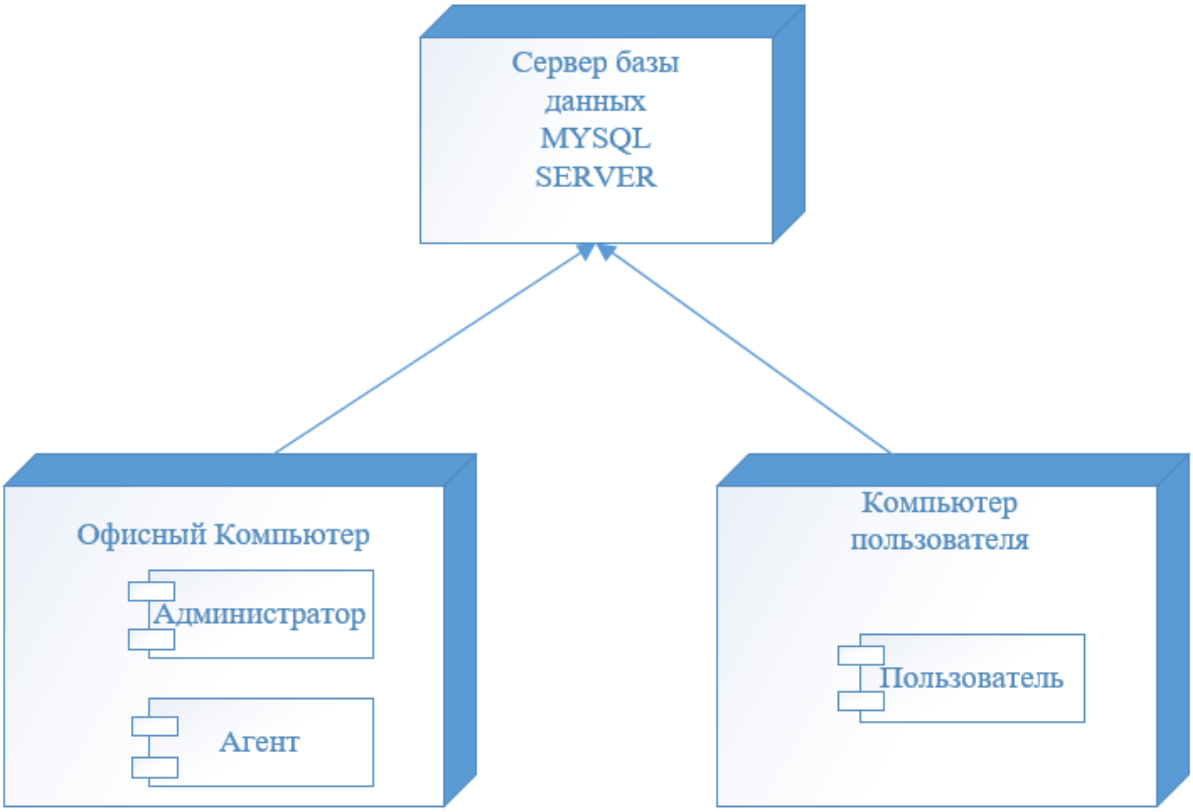

Рисунок 5. Диаграмма развертывания

4.1.6 Физическая модель базы данных

Физическая модель базы данных представлена на Рисунке. 6

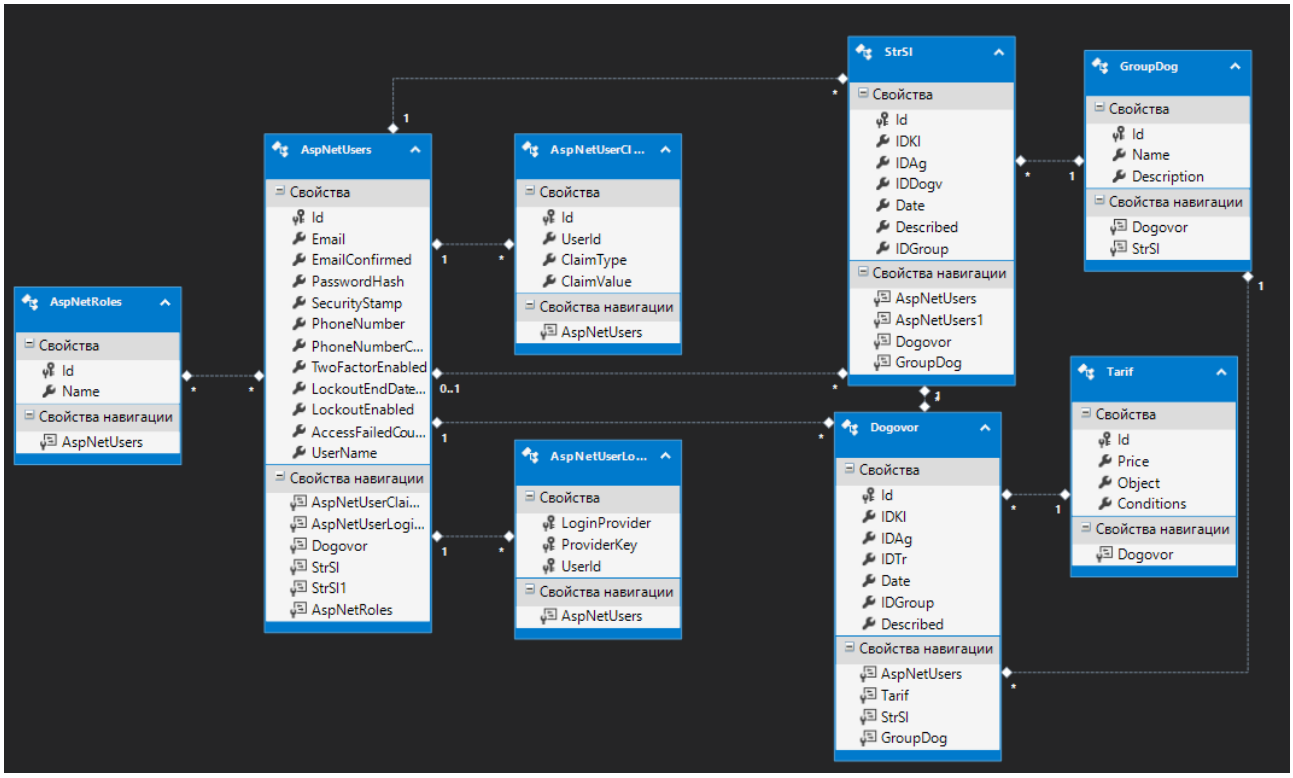

Рисунок 6. Физическая модель базы данных

5 РЕАЛИЗАЦИЯ СИСТЕМЫ

5.1 Общие принципы организации системы

Разрабатываемая система представляет собой веб-приложение, которое построено на базе архитектуры ASP.NET MVC (код программы представлен в приложении А). [26]

В приложении организована реализация трех компонентов: контроллера, модели и представления.

Представления построены с использованием HtmlHelper.

Код программы представлен в приложении А. []

5.2 Реализация модели

Для реализации данной системы был выбран подход Database-First и объектно-ориентированная технология доступа к данным Entity Framework. Сначала была создана база данных, состоящая из 8 таблиц. 4 из них, являются встроенными в проект таблицами, используемых для работы с ролями и пользователями. На основе 4 оставшихся таблиц были созданы модели для обработки данных их представлений.

5.3 Реализация механизма регистрации, аутентификации и авторизации

Для реализации регистрации, аутентификации и авторизации пользователей были использованы стандартные методы шаблона Microsoft Identity. К методам контроллера имеют доступ, только пользователи, соответствующей роли. Это обеспечивает аннотация вида [Authorize(Roles="роль")]. Клиенту присвоена роль "Polzovatel". Агенту присвоена роль "Agent". Администратору присвоена роль "Admin". Незарегистрированные пользователи не могут просматривать запрещенные для них просмотра, благодаря установки над методами [Authorize].

5.4 Реализация поддержки различных справочников

Пользователи системы имеют доступ к базе данных. Клиент может оформлять Договор и Страховой случай. Агент может редактировать статус заявки на оформление договора и информацию о клиентах, а также заключать договора.

5.5 Реализация заполнения, создание и одобрение заявок.

Клиент нажимает на кнопку «Оформить договор» или «Оформить страховой случай», заполняет все представленные поля, затем нажимает на кнопку создать. Система автоматически отправит данные в базу данных и присвоит первоначальный статус. Страхователь может отслеживать статус своих заявок в личном кабинете.

Агент может просмотреть конкретную заявку, отредактировать запись и присвоить необходимый статус, в зависимости от инструкций.

5.6 Пользовательский интерфейс.

Пользовательский интерфейс представлен на рисунках 6-7.

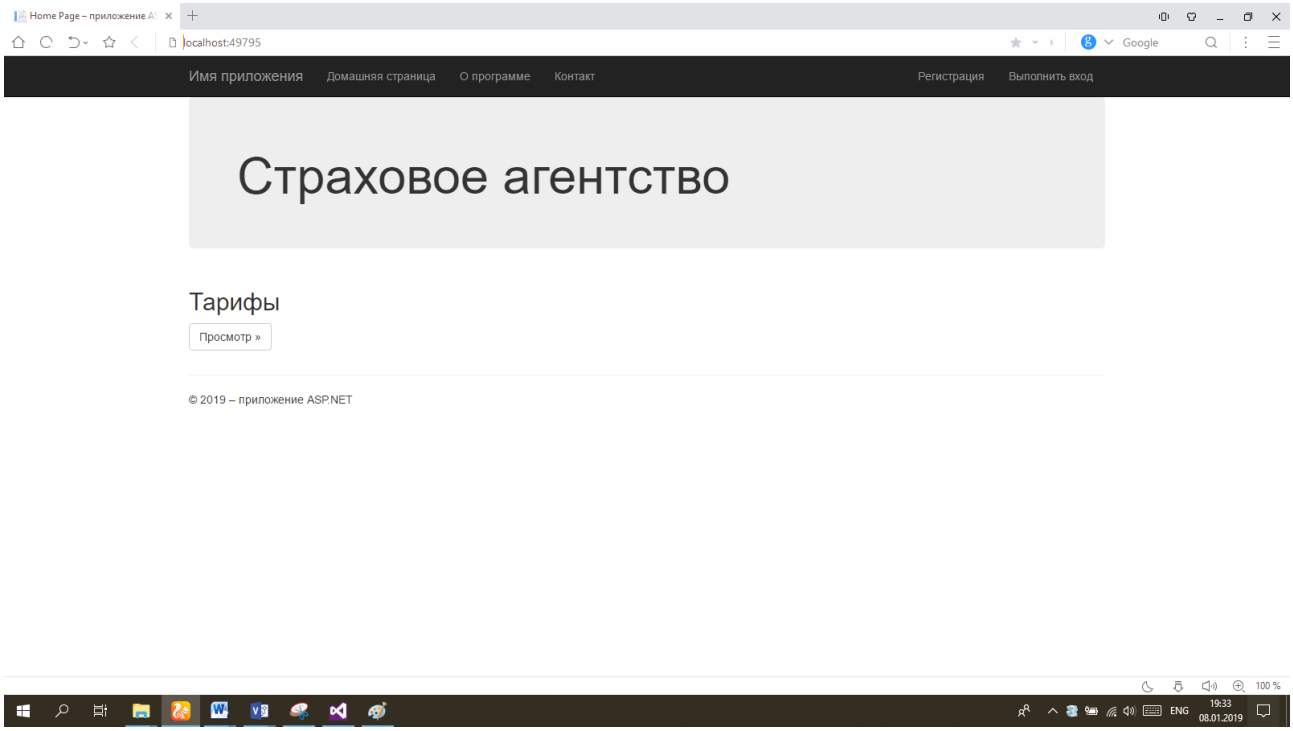

Рисунок 6. Форма пользователя.

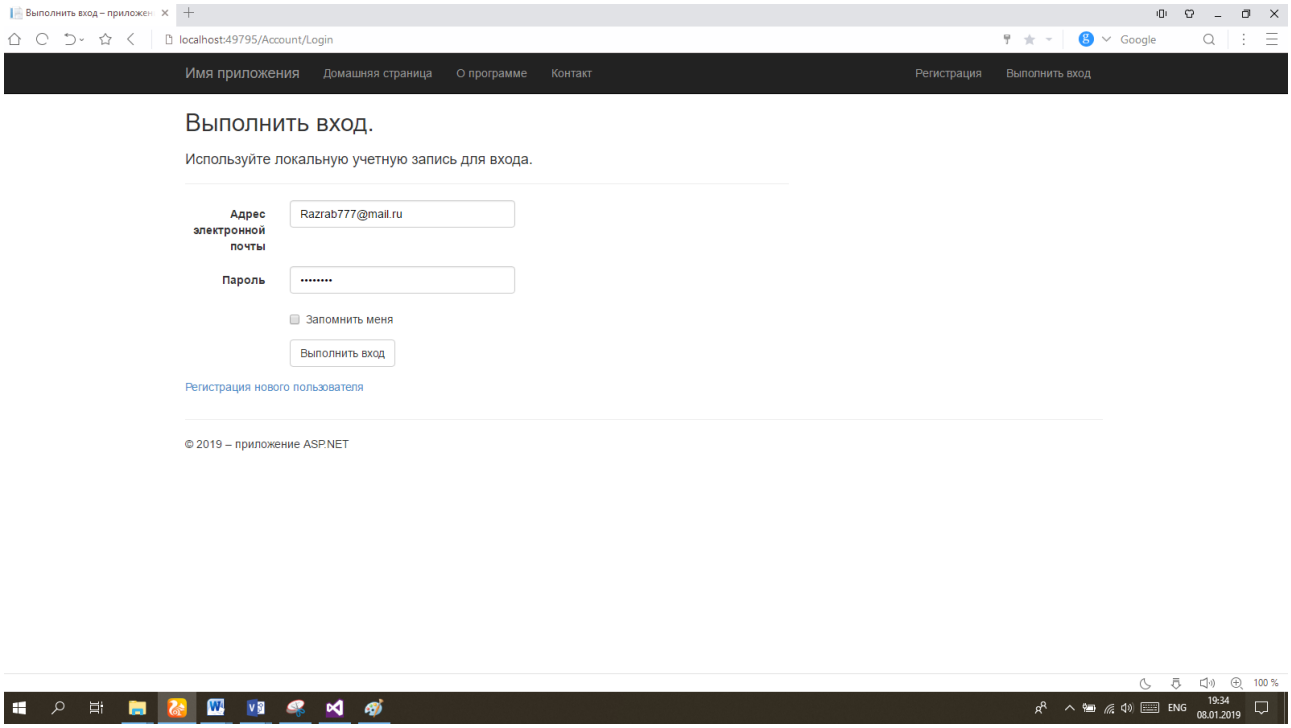

Рисунок 6. Страница авторизации.

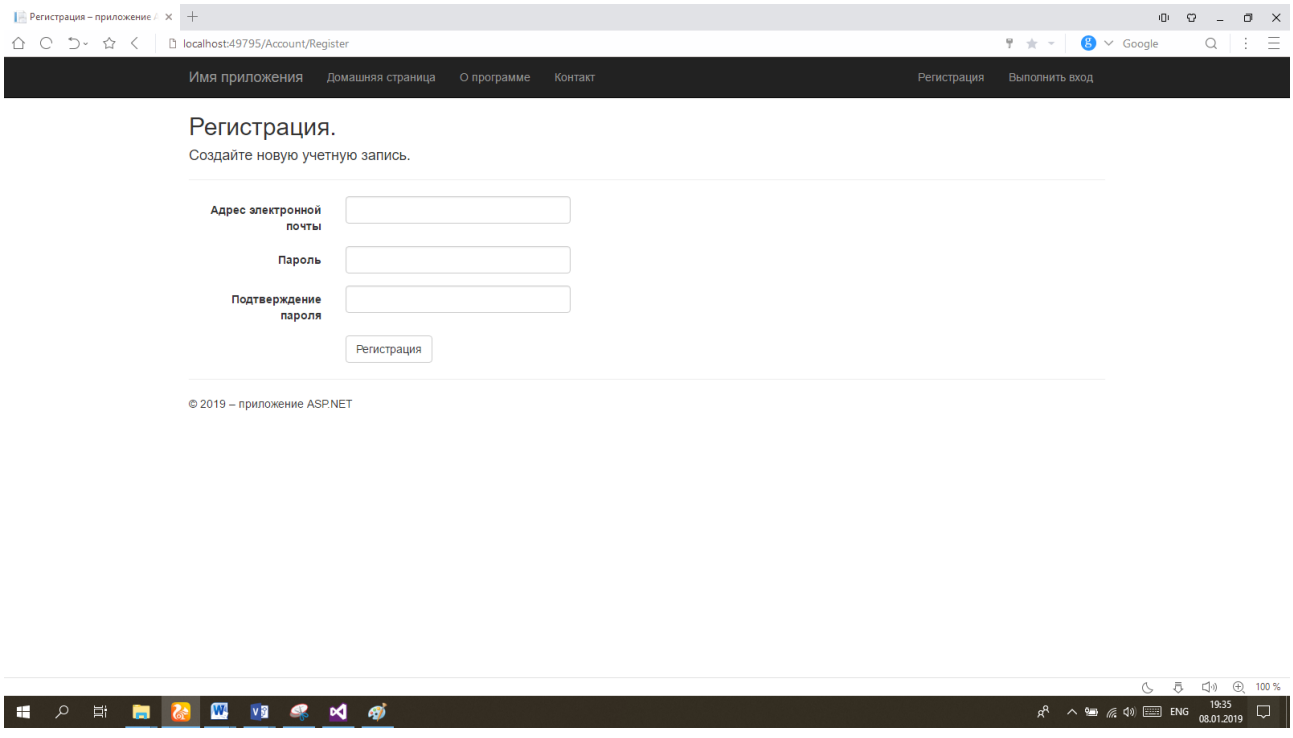

Рисунок 7. Страница регистрации.

Интерфейс администратора и агента представлен на рисунках 8-12.

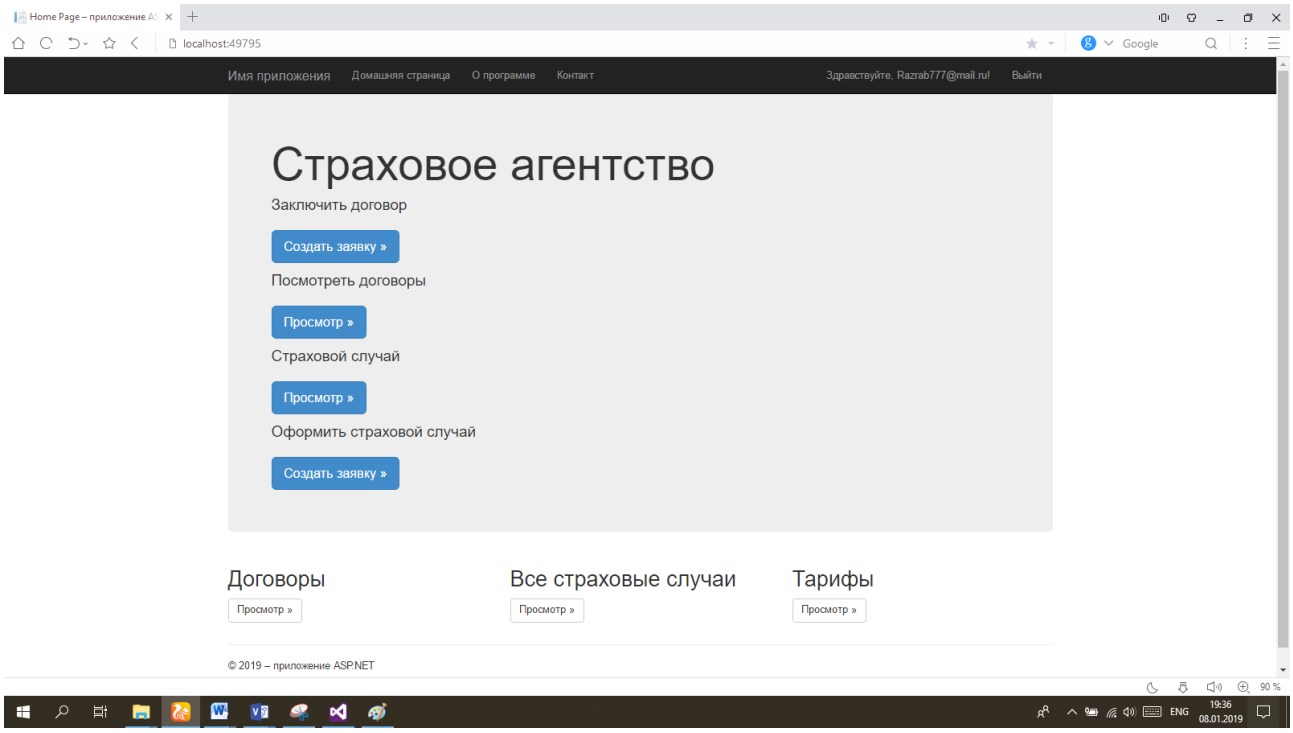

Рисунок 8. Панель управления администратора.

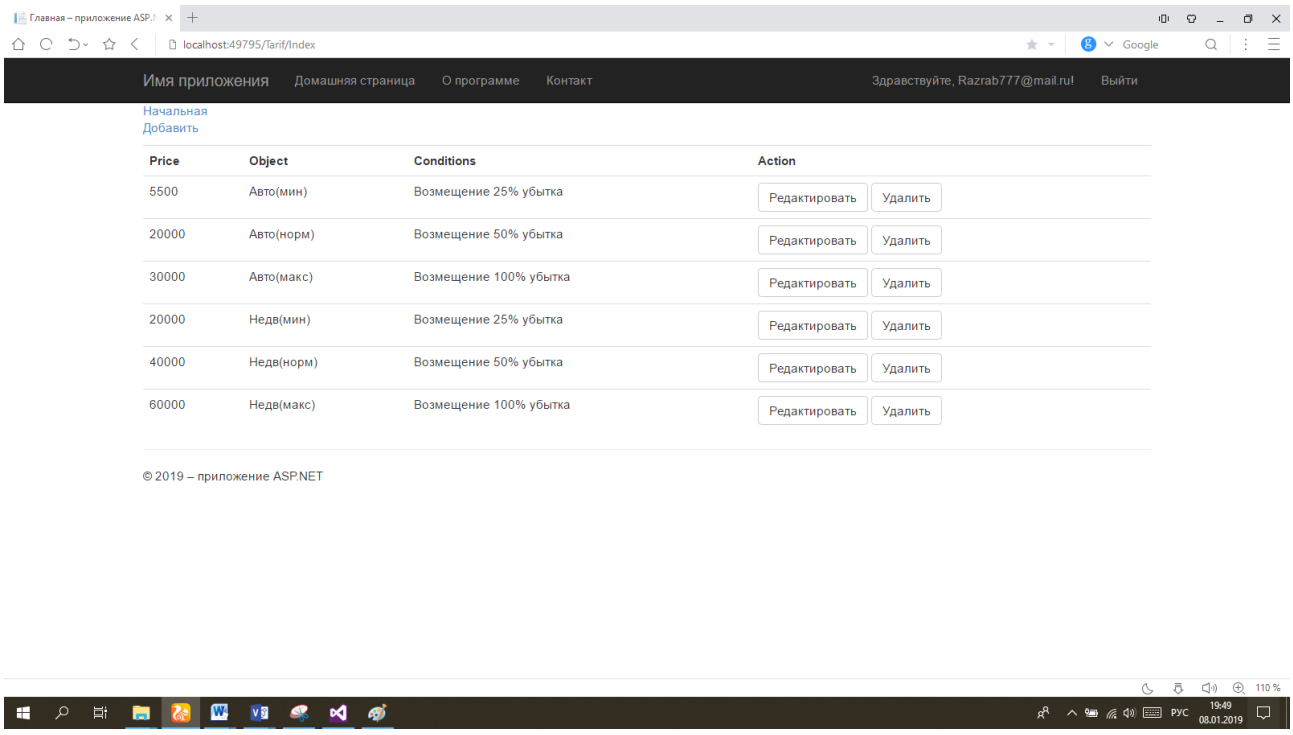

Рисунок 9. Окно работы с тарифами.

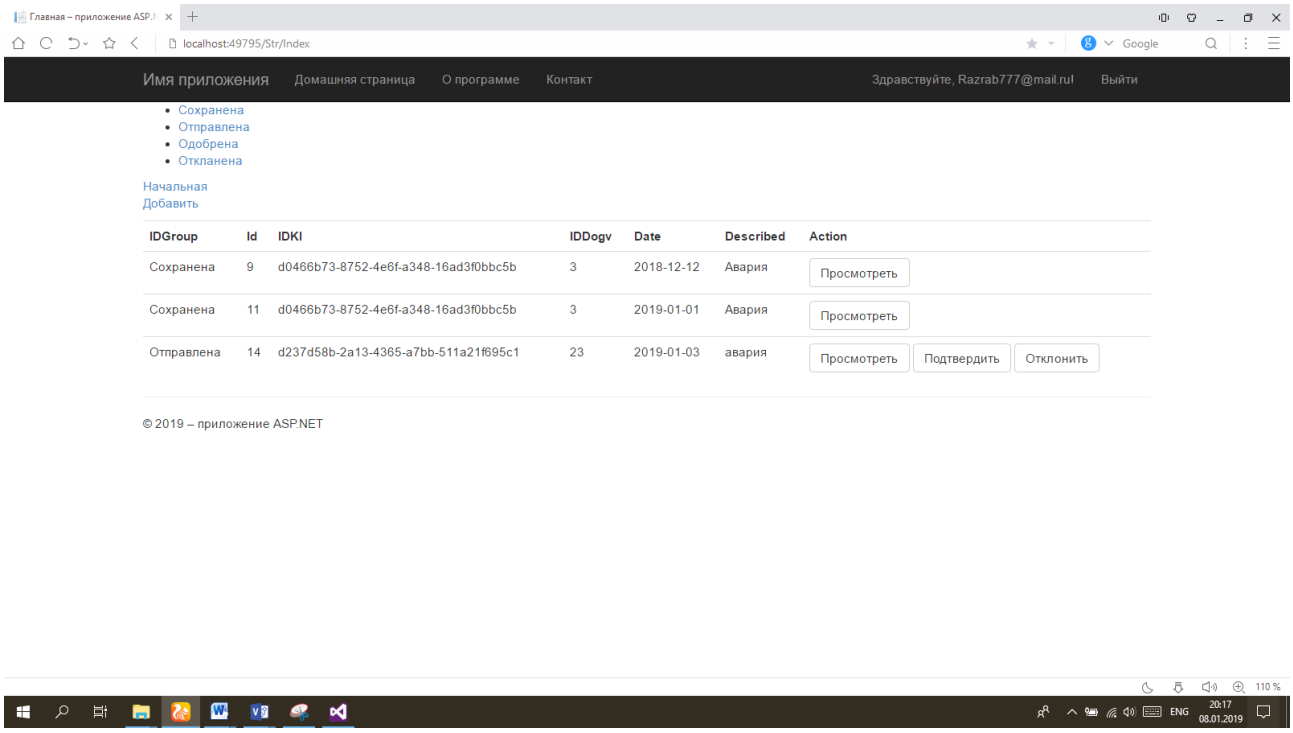

Рисунок 10. Окно работы со страховыми случаями.

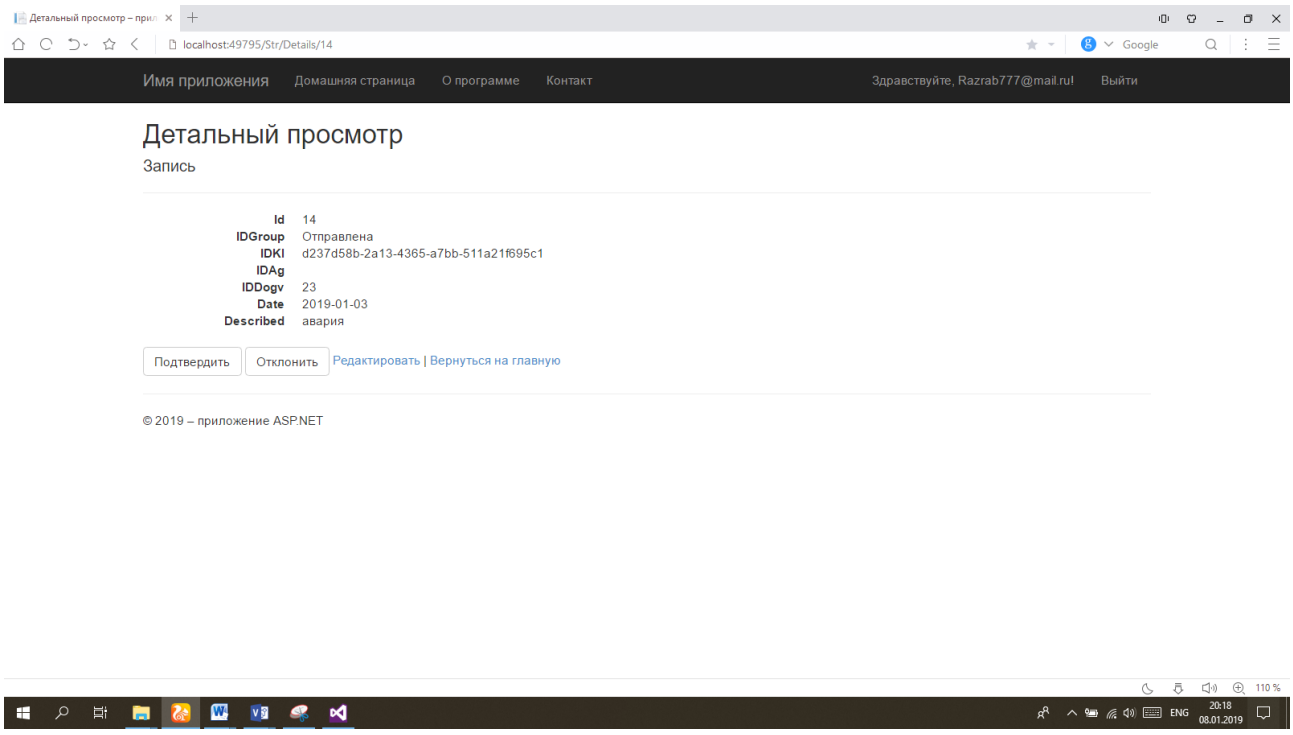

Рисунок 11. Окно смены статуса страхового случая.

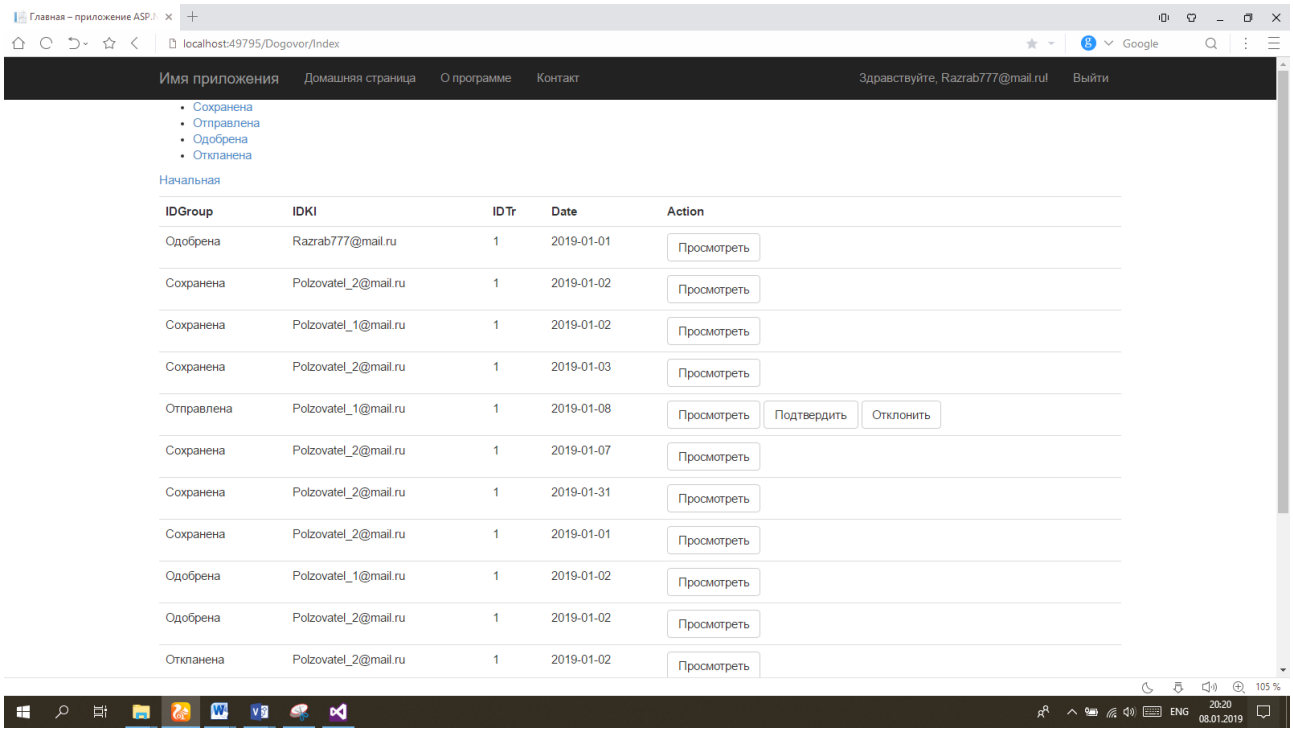

Рисунок 17. Общий список договоров.

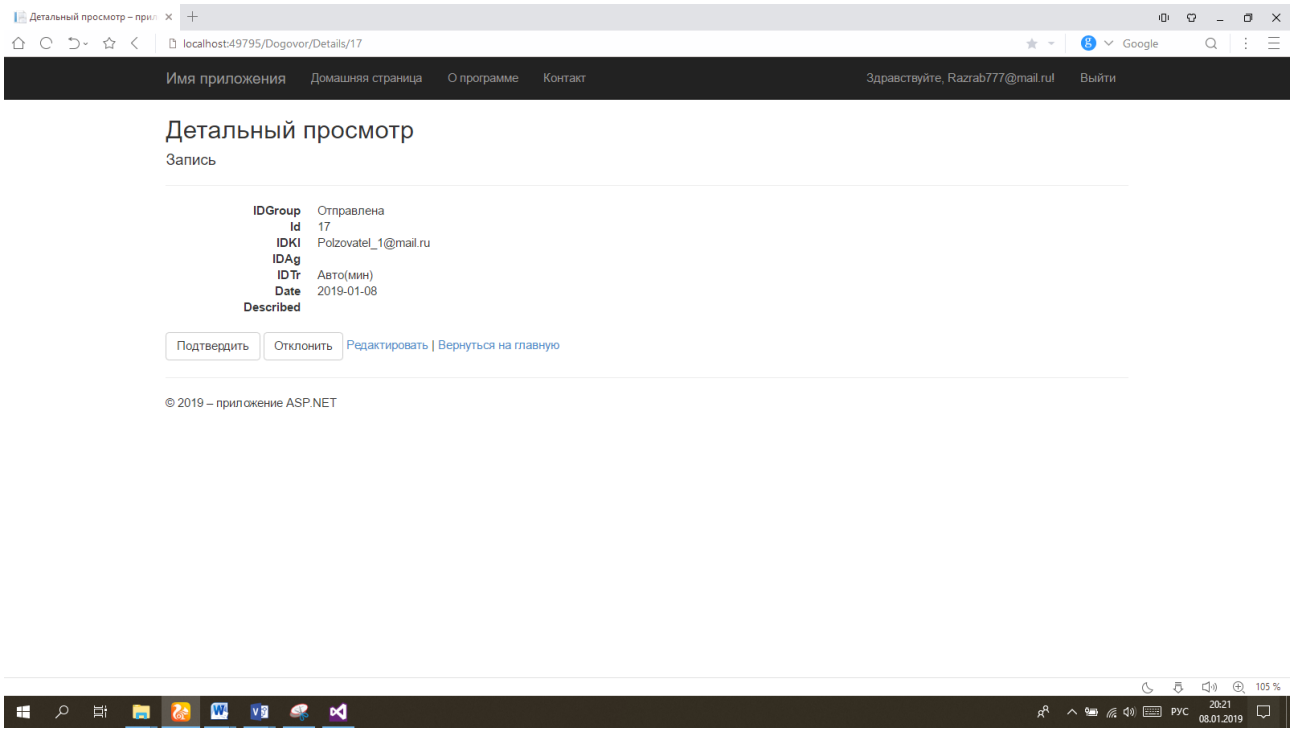

Рисунок 19. Редактирование статуса договора.

Интерфейс клиента представлен на рисунках 20-25.

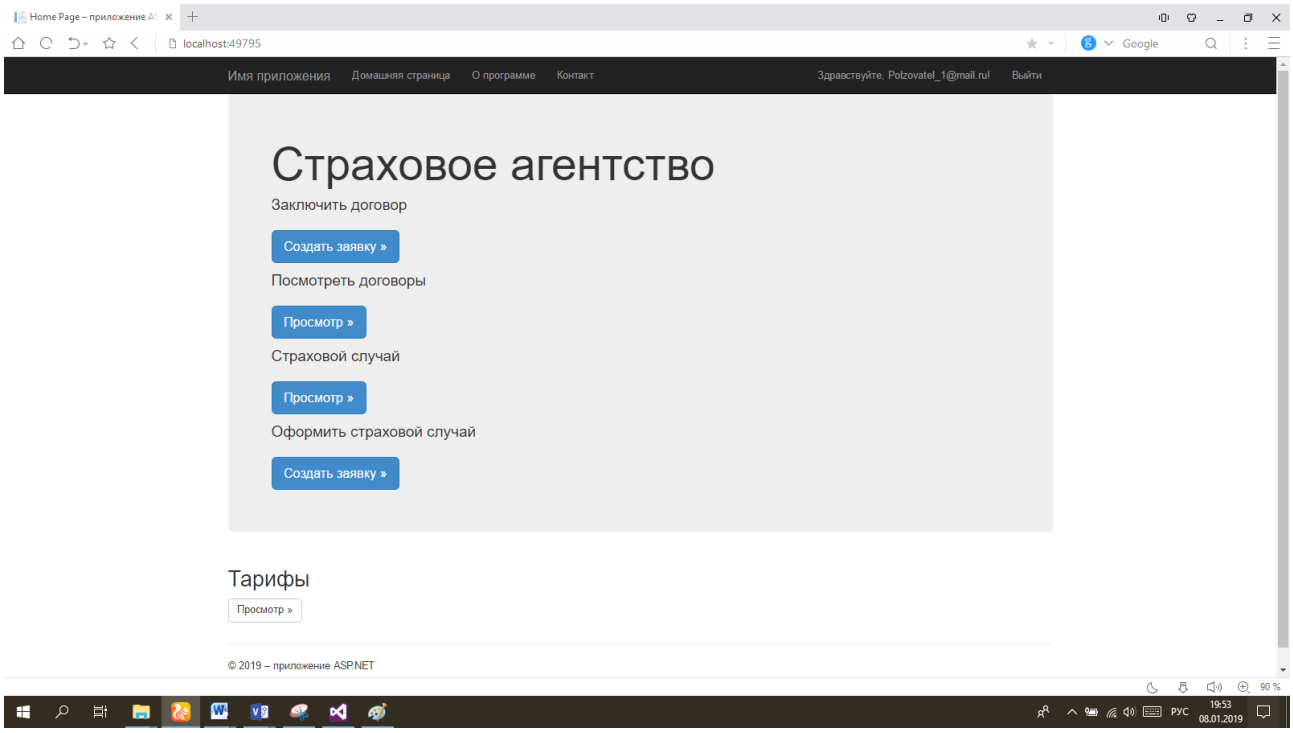

Рисунок 20. Форма клиента.

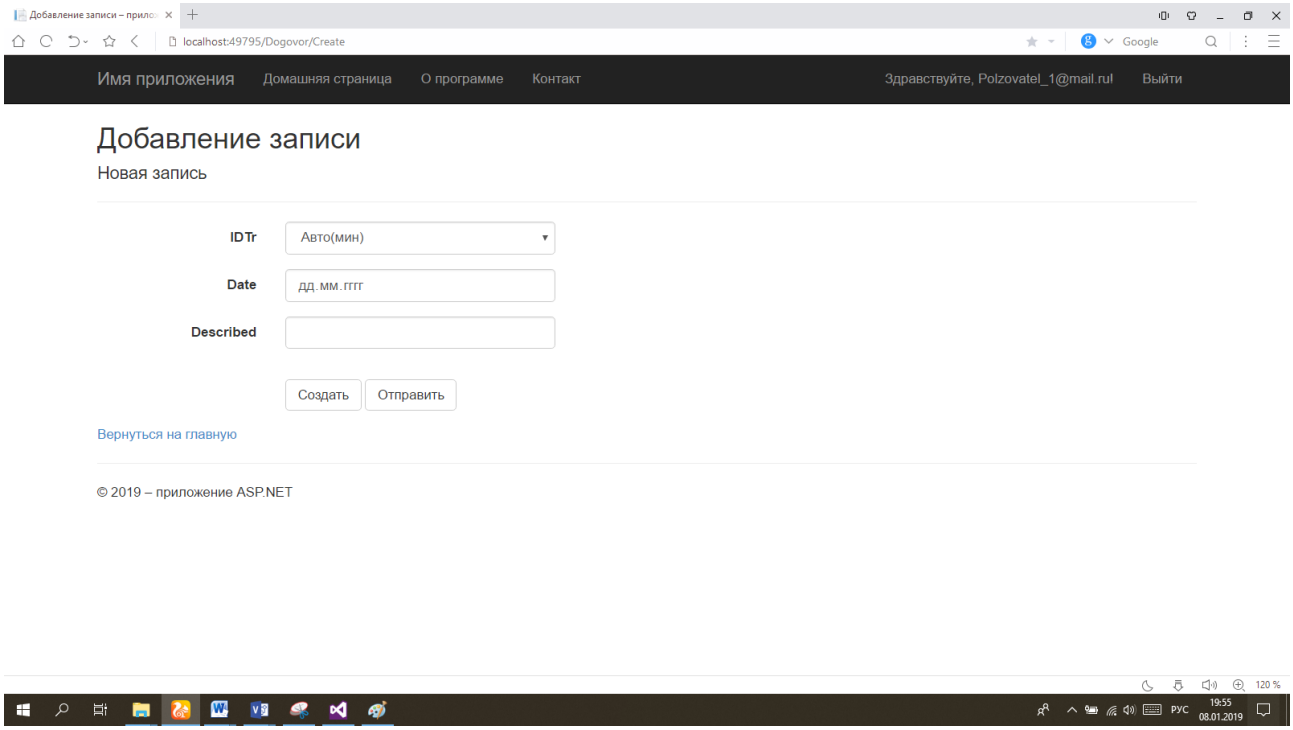

Рисунок 20. Окно оформления договора страхователем.

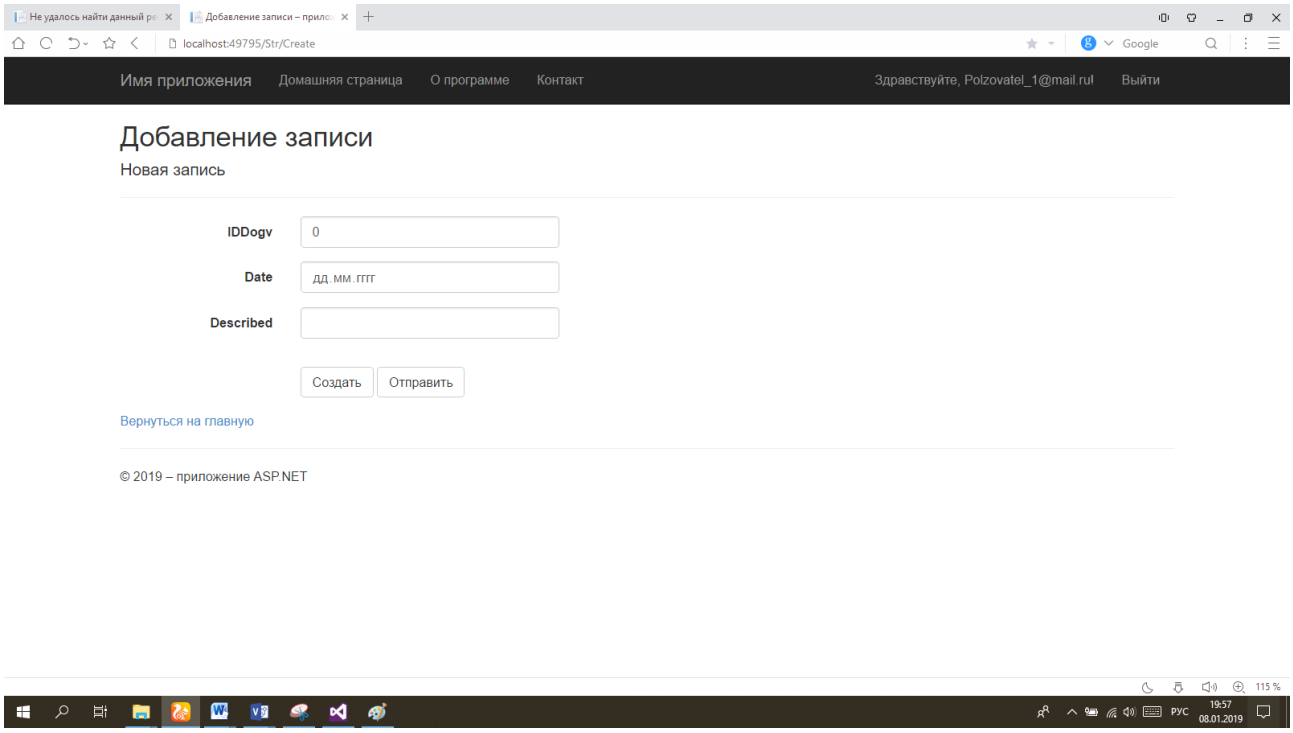

Рисунок 21. Окно оформления страхового случая страхователем.

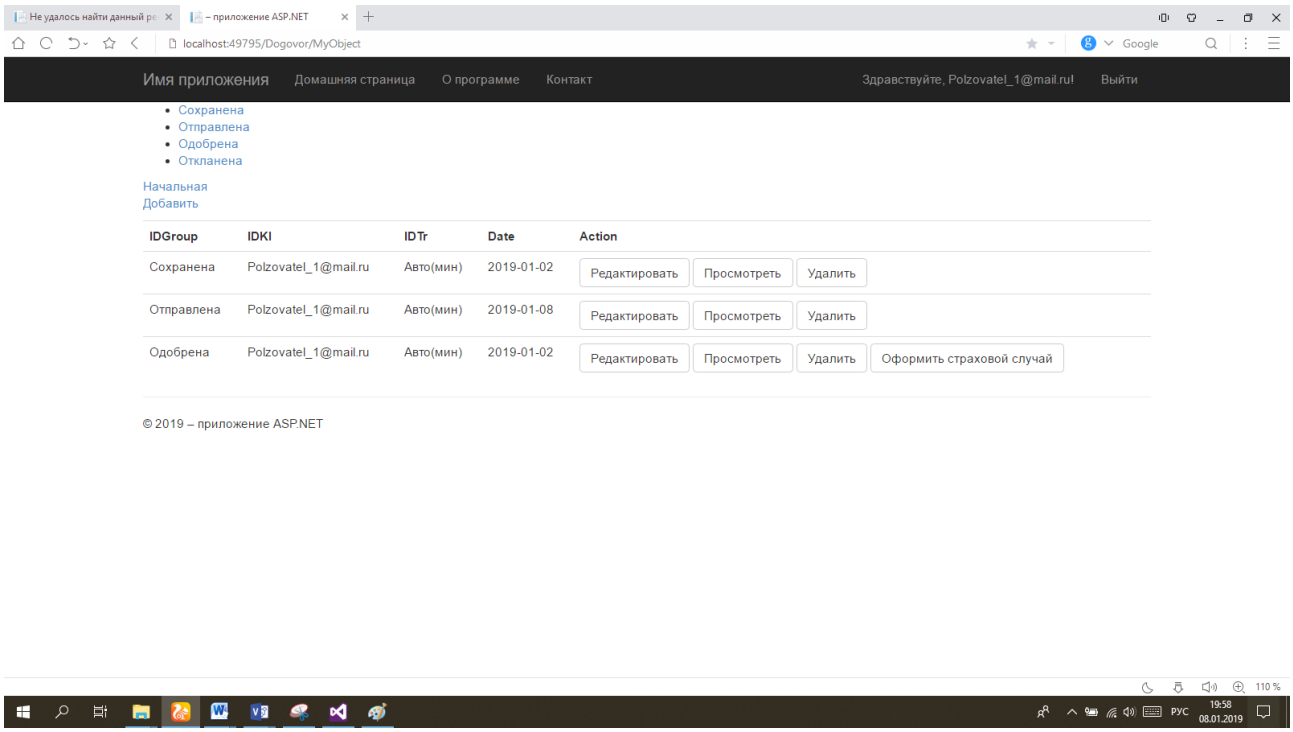

Рисунок 24. Список договоров конкретного страхователя.

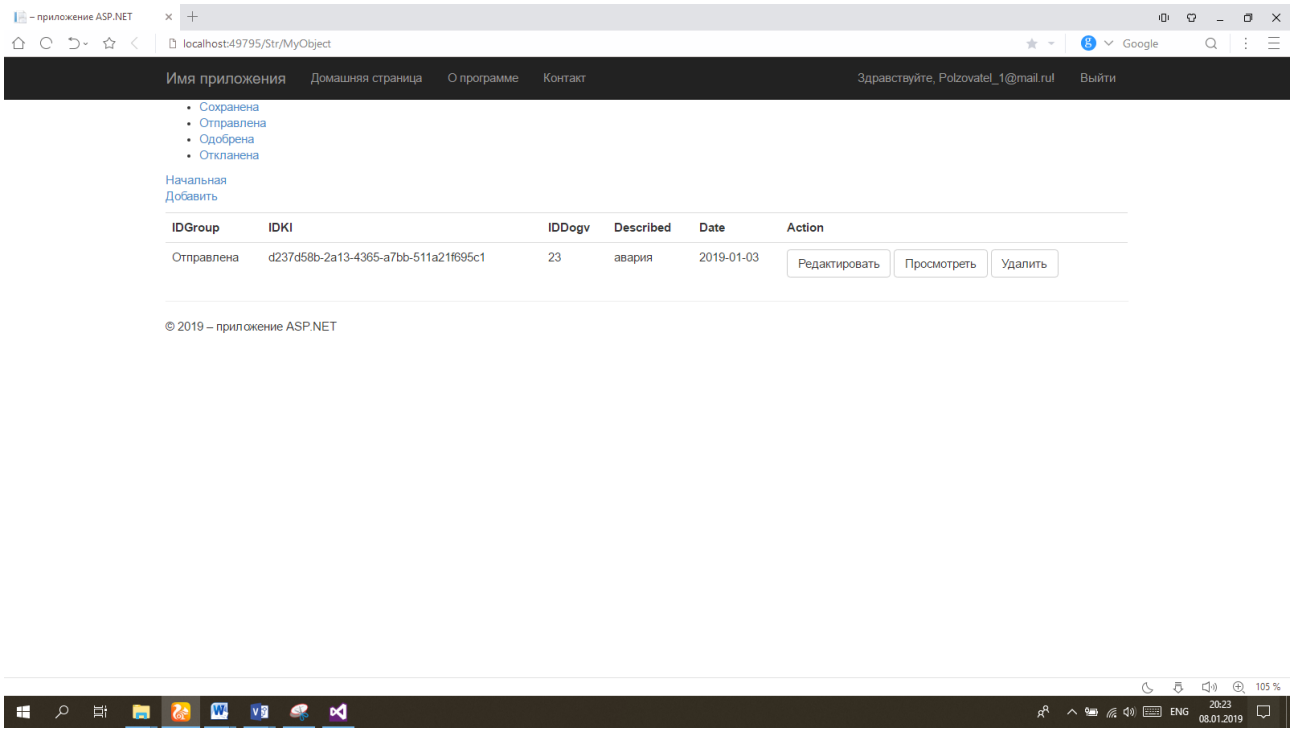

Рисунок 25. Список страховых случаев конкретного страхователя.

6 ЗАКЛЮЧЕНИЕ

В ходе работы была разработана программная система по информационной области «Страховое агентство». Была использована система контроля версий. Для удобства работы с системой были созданы различные методы для взаимодействия разных пользователей с системой.

Разработанная база данных является актуальной на сегодняшний день и имеет большую практическую значимость.

7 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Липаев В.В. Проектирование программных систем. М.: Высш. шк, 1990.
2. Буч Г. Объективно-ориентированное проектирование / Пер. с анг. Канкорд, 1996.
3. Майерс Г. Надежность программного обеспечения. М.:Мир, 1980
4. Калянов Г.Н. CASE структурный системный анализ (автоматизация и применение.) – М.: «ЛОРИ», 1996.
5. Шмуллер Дж. Освой самостоятельно UML. 2-е изд. М.: изд. дом «Вильямс», 2002
6. Калберстон Р., Браун К., Кобб Г. Быстрое тестирование М.: изд. дом «Вильямс», 2002
7. Кармайкл Э., Хейвуд Д. Быстрая и качественная разработка программного обеспечения. М.: изд. дом «Вильямс», 2003
8. Руководство по ASP.NET MVC 5 [Электронный ресурс]: Metantit.com. – Режим доступа: http://metanit.com/sharp/mvc5/

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММНОЙ СИСТЕМЫ

Модель:

Класс Договор */Agent/Models/Dogovor

```
namespace Agent.Models
{
    using System;
    using System.Collections.Generic;

    public partial class Dogovor
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
        public Dogovor()
        {
            this.StrSl = new HashSet<StrSl>();
        }

        public int Id { get; set; }
        public string IDKl { get; set; }
        public string IDAg { get; set; }
        public int IDTr { get; set; }
        public string Date { get; set; }
        public int IDGroup { get; set; }
        public string Described { get; set; }

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

Класс Статус */Agent/Models/GroupDog

```
namespace Agent.Models
{
    using System;
    using System.Collections.Generic;

    public partial class GroupDog
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
        public GroupDog()
        {
            this.Dogovor = new HashSet<Dogovor>();
            this.StrSl = new HashSet<StrSl>();
        }

        public int Id { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }

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

```
}
```

Класс Страховой случай */Agent/Models/StrSl

```
namespace Agent.Models
{
    using System;
    using System.Collections.Generic;

    public partial class StrSl
    {
        public int Id { get; set; }
        public string IDKl { get; set; }
        public string IDAg { get; set; }
        public int IDDogv { get; set; }
        public string Date { get; set; }
        public string Described { get; set; }
        public int IDGroup { get; set; }

        public virtual AspNetUsers AspNetUsers { get; set; }
        public virtual AspNetUsers AspNetUsers1 { get; set; }
        public virtual Dogovor Dogovor { get; set; }
        public virtual GroupDog GroupDog { get; set; }
    }
}
```

Класс Тариф */Agent/Models/Tarif

```
namespace Agent.Models
{
    using System;
    using System.Collections.Generic;

    public partial class Tarif
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
        public Tarif()
        {
            this.Dogovor = new HashSet<Dogovor>();
        }

        public int Id { get; set; }
        public double Price { get; set; }
        public string Object { get; set; }
        public string Conditions { get; set; }

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

Класс Лог */Model/Logger

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using log4net;
using log4net.Config;

namespace Agent
{
    public static class Logger
    {
        private static ILog log = LogManager.GetLogger("ALL");
        public static ILog Log
        {
            get { return log; }
```

```
        }

        public static void InitLogger()
        {
            XmlConfigurator.Configure();
        }
    }
}
```

Работа с базой данных:

Класс */DAO/DogovorDAO

```
using Agent.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Agent.DAO
{
    public class DogovorDAO
    {
        private Entities _entities = new Entities();

        public List<Dogovor> GetMyObject(string id)
        {
            List<Dogovor> myobject = new List<Dogovor>();
            try
            {
                {
                    myobject = _entities.Dogovor
                        .Include("GroupDog")
                        .Include("AspNetUsers")
                        .Where(c => c.IDKl == id).ToList();
                }
            }
            catch (Exception ex)
            {
                Logger.Log.Error("Ошибка: ", ex);
            }
            return myobject;
        }

        public IEnumerable<Dogovor> GetAllDogovor()
        {
            return (from c in _entities.Dogovor.Include("GroupDog") select c);
        }

        public GroupDog GetGroupDog(int? id)
        {
            if (id != null) //возращает запись по её Id
                return (from c in _entities.GroupDog
                        where c.Id == id
                        select c).FirstOrDefault();
            else // возращает первую запись в таблице
                return (from c in _entities.GroupDog
                        select c).FirstOrDefault();
        }
        public Dogovor getDogovor(int id)
        {
            return (from c in _entities.Dogovor.Include("GroupDog")
                    where c.Id == id
                    select c).FirstOrDefault();
        }
```

```
        //
        public void CreateDogovor(Dogovor model)
        {
            try
            {
                using (var ctx = new Entities())
                {
                    string query = "INSERT INTO Dogovor (IDKl, IDAg, IDTr, Date, IDGroup,
Described) VALUES(@P0, @P1, @P2, @P3, @P4, @P5)";
                List<object> parameterList = new List<object>{
                    model.IDKl,
                    model.IDAg,
                    model.IDTr,
                    model.Date,
                    model.IDGroup,
                    model.Described
                };
                    object[] parameters = parameterList.ToArray();
                    int result = ctx.Database.ExecuteSqlCommand(query, parameters);
                    Logger.Log.Info("Данные успешно добавлены в Dogovor");

                }
            }
            catch (Exception ex)
            {
                Logger.Log.Error("Ошибка: ", ex);
            }
        }



        public bool updateDogovor( Dogovor Dogov)
        {
            Dogovor originalRecords = getDogovor(Dogov.Id);
            try
            {

                originalRecords.IDTr = Dogov.IDTr;
                originalRecords.Date = Dogov.Date;
                originalRecords.IDGroup = Dogov.IDGroup;
                originalRecords.Described = Dogov.Described;

                _entities.SaveChanges();
            }
            catch (Exception ex)
            {
                Logger.Log.Error("Ошибка: ", ex);
                return false;
            }
            return true;
        }
        public bool deleteDogovor(int Id)
        {
            Dogovor originalRecords = getDogovor(Id);
            try
            {
                _entities.Dogovor.Remove(originalRecords);
                _entities.SaveChanges();
            }
            catch (Exception ex)
            {
                Logger.Log.Error("Ошибка: ", ex);
                return false;
            }
            return true;
```

```
        }

        public bool UpdateGroup(Dogovor dogovor)
        {
            try
            {
                var Entity = _entities.Dogovor.FirstOrDefault(x => x.Id == dogovor.Id);
                Entity.IDGroup = dogovor.IDGroup;
                Entity.IDAg = dogovor.IDAg;
                _entities.SaveChanges();
            }
            catch (Exception ex)
            {
                Logger.Log.Error("Ошибка: ", ex);
                return false;
            }
            return true;
        }
    }
}
```

Класс */DAO/StrSlDAO

```
using Agent.Models;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Web;

namespace Agent.DAO
{
    public class StrSlDAO : DAO
    {
        private Entities _entities = new Entities();

        public List<StrSl> GetMyObject(string id)
        {
            List<StrSl> myobject = new List<StrSl>();
            try
            {
                {
                    myobject = _entities.StrSl
                        .Include("GroupDog")
                        .Include("AspNetUsers")
                        .Where(c => c.IDKl == id).ToList();
                }
            }
            catch (Exception ex)
            {
                Logger.Log.Error("Ошибка: ", ex);
            }
            return myobject;
        }

        public IEnumerable<StrSl> GetAllStrSl()
        {
            return (from c in _entities.StrSl.Include("GroupDog") select c);
        }
        public GroupDog GetGroupDog(int? id)
        {
            if (id != null) //возращает запись по её Id
                return (from c in _entities.GroupDog
                        where c.Id == id
                        select c).FirstOrDefault();
            else // возращает первую запись в таблице
```

```
                return (from c in _entities.GroupDog
                        select c).FirstOrDefault();
        }
        public StrSl getStrSl(int id)
        {
            return (from c in _entities.StrSl.Include("GroupDog")
                    where c.Id == id
                    select c).FirstOrDefault();
        }

        public void CreateStrSl(StrSl model)
        {
            try
            {
                using (var ctx = new Entities())
                {
                    string query = "INSERT INTO StrSl (IDKl, IDAg, IDDogv, Date, Described,
IDGroup) VALUES(@P0, @P1, @P2, @P3, @P4, @P5)";
                    List<object> parameterList = new List<object>{
                    model.IDKl,
                    model.IDAg,
                    model.IDDogv,
                    model.Date,
                    model.Described,
                    model.IDGroup
                };
                    object[] parameters = parameterList.ToArray();
                    int result = ctx.Database.ExecuteSqlCommand(query, parameters);
                             Logger.Log.Info("Данные успешно добавлены в StrSl");

                }
            }
            catch (Exception ex)
            {
                Logger.Log.Error("Ошибка: ", ex);
            }
        }
        public bool updateStrSl( StrSl Str)
        {
            StrSl originalRecords = getStrSl(Str.Id);

            try
            {
                originalRecords.IDKl = Str.IDKl;
                originalRecords.IDDogv = Str.IDDogv;
                originalRecords.Date = Str.Date;
                originalRecords.Described = Str.Described;
                originalRecords.IDGroup = Str.IDGroup;


                _entities.SaveChanges();
            }
            catch (Exception ex)
            {
                Logger.Log.Error("Ошибка: ", ex);
                return false;
            }
            return true;
        }
        public bool deleteStrSl(int Id)
        {
            StrSl originalStrSl = getStrSl(Id);
            try
            {
                _entities.StrSl.Remove(originalStrSl);
                _entities.SaveChanges();
```

```
            }
            catch (Exception ex)
            {
                Logger.Log.Error("Ошибка: ", ex);
                return false;
            }
            return true;
        }
    }
}
```

Класс */DAO/TarifDAO

```
using Agent.Models;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Web;

namespace Agent.DAO
{
    public class TarifDAO : DAO
    {
        public List<Tarif> GetAllTarif()
        {
            Connect();
            List<Tarif> tarifList = new List<Tarif>();
            try
            {
                SqlCommand command = new SqlCommand("SELECT * FROM Tarif", Connection);
                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    Tarif tarif = new Tarif();
                    tarif.Id = Convert.ToInt32(reader["Id"]);
                    tarif.Price = Convert.ToInt32(reader["Price"]);
                    tarif.Object = Convert.ToString(reader["Object"]);
                    tarif.Conditions = Convert.ToString(reader["Conditions"]);

                    tarifList.Add(tarif);
                }
                reader.Close();
            }
            catch (Exception ex)
            {
                Logger.Log.Error("Ошибка: ", ex);
            }
            finally { Disconnect(); }
            return tarifList;
        }

        public bool AddTarif(Tarif tarif)
        {
            bool result = true;
            Connect();
            try
            {
                SqlCommand cmd = new SqlCommand(
                    "INSERT INTO  Tarif (Price, Object, Conditions) " +
                    "VALUES (@Price, @Object, @Conditions)", Connection
                    );
                cmd.Parameters.AddWithValue("@Price", tarif.Price);
                cmd.Parameters.AddWithValue("@Object", tarif.Object);
                cmd.Parameters.AddWithValue("@Conditions", tarif.Conditions);

                cmd.ExecuteNonQuery();
```

```
            }
            catch (Exception ex)
            {
                Logger.Log.Error("Ошибка: ", ex);
                result = false;
            }
            finally { Disconnect(); }
            return result;
        }
        public void EditTarif(Tarif tarif)
        {
            try
            {
                Connect();
                string str = "UPDATE Tarif SET Price = '" + tarif.Price
                    + "', Object = '" + tarif.Object
                    + "', Conditions = '" + tarif.Conditions
                    + "'WHERE Id = " + tarif.Id;
                SqlCommand com = new SqlCommand(str, Connection);
                com.ExecuteNonQuery();
            }
            catch (Exception ex)
            {
                Logger.Log.Error("Ошибка: ", ex);
            }
            finally
            {
                Disconnect();
            }
        }
        public void DeleteTarif(int id)
        {
            try
            {
                Connect();
                string str = "DELETE FROM Tarif WHERE Id=" + id;
                SqlCommand com = new SqlCommand(str, Connection);
                com.ExecuteNonQuery();
            }
            catch (Exception ex)
            {
                Logger.Log.Error("Ошибка: ", ex);
            }
            finally
            {
                Disconnect();
            }
        }

    }
}
```

Класс */DAO/GroupDogDAO

```
using Agent.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Agent.DAO
{
    public class GroupDogDAO
    {
        private Entities _entities = new Entities();
        public IEnumerable<GroupDog> GetAllGroups()
```

```
        {
            return (from c in _entities.GroupDog
                    select c);
        }
    }
}
```

Контроллеры:

Класс */Controllers/DogovorController

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Agent.DAO;
using Agent.Models;
using Microsoft.AspNet.Identity;
using log4net.Repository.Hierarchy;
using log4net;
using log4net.Config;

namespace Agent.Controllers
{
    public class DogovorController : Controller
    {

        GroupDogDAO groupDAO = new GroupDogDAO();
        DogovorDAO dogovorDAO = new DogovorDAO();
        TarifDAO tarifDAO = new TarifDAO();

        [Authorize]
        public ActionResult Confirm(int id)
        {
            Dogovor dogovor = dogovorDAO.getDogovor(id);
            string userId = User.Identity.GetUserId();
            dogovor.IDAg = userId;
            dogovor.IDGroup = 3;
            dogovorDAO.UpdateGroup(dogovor);
            return RedirectToAction("Index");
        }
        public ActionResult Reject(int id)
        {
            Dogovor dogovor = dogovorDAO.getDogovor(id);
            string userId = User.Identity.GetUserId();
            dogovor.IDAg = userId;
            dogovor.IDGroup = 4;
            dogovorDAO.UpdateGroup(dogovor);
            return RedirectToAction("Index");
        }

        public ActionResult MyObject(int? id)
        {
            ViewData["Groups"] = groupDAO.GetAllGroups();
            var dogovor = id == null ? dogovorDAO.GetAllDogovor() :
dogovorDAO.GetAllDogovor().Where(x => x.GroupDog.Id == id);
            string userid = User.Identity.GetUserId();
            var myobject = dogovorDAO.GetMyObject(userid);
            if (!Request.IsAjaxRequest())
                return View(myobject);
            else

                return PartialView("DogovorList", dogovor);
        }
```

```
        public ActionResult Index(int? id)
        {
            ViewData["Groups"] = groupDAO.GetAllGroups();
            var dogovor = id == null ? dogovorDAO.GetAllDogovor() :
dogovorDAO.GetAllDogovor().Where(x => x.GroupDog.Id == id);
            if (!Request.IsAjaxRequest())
                return View(dogovor);
            else
                return PartialView("DogovorList", dogovor);

        }


        public ActionResult Details(int id)
        {
            return View(dogovorDAO.getDogovor(id));
        }

        protected bool ViewDataSelectList(int GroupId)
        {
            var groups = groupDAO.GetAllGroups();
            ViewData["GroupId"] = new SelectList(groups, "Id", "Name", GroupId);
            return groups.Count() > 0;
        }

        // Create
        [HttpGet]
        public ActionResult Create(string id)
        {
            var tarif = new SelectList(tarifDAO.GetAllTarif(), "Id", "Object");
            ViewData["IDTr"] = tarif;
            Dogovor dogovor = new Dogovor();
            return View(dogovor);
        }
        [HttpPost]
        public ActionResult Create(Dogovor model)
        {
            var tarif = new SelectList(tarifDAO.GetAllTarif(), "Id", "Object");
            ViewData["IDTr"] = tarif;
            string userId = User.Identity.GetUserId();
            model.IDKl = userId;
            try
            {
                dogovorDAO.CreateDogovor(model);
                return RedirectToAction("MyObject");
            }
            catch (Exception ex)
            {
               Logger.Log.Error("Ошибка: ", ex);
            }
            return RedirectToAction("Create");
        }


        public ActionResult Edit(int id)
        {
            Dogovor Dogov = dogovorDAO.getDogovor(id);
            if (!ViewDataSelectList(Dogov.GroupDog.Id))
                return RedirectToAction("MyObject");
            return View(dogovorDAO.getDogovor(id));
        }

        [HttpPost]
        public ActionResult Edit( Dogovor Dogov)
        {
            ViewDataSelectList(-1);
```

```
            try
            {
                if (ModelState.IsValid && dogovorDAO.updateDogovor( Dogov))
                    return RedirectToAction("MyObject");
                else
                    return View(Dogov);
            }
            catch (Exception ex)
            {
                Logger.Log.Error("Ошибка: ", ex);
                return View(Dogov);
            }
        }

        public ActionResult Delete(int id)
        {
            return View(dogovorDAO.getDogovor(id));
        }

        [HttpPost]
        public ActionResult Delete(int id, Dogovor Dogov)
        {
            try
            {
                if (dogovorDAO.deleteDogovor(id))
                    return RedirectToAction("MyObject");
                else
                    return View(dogovorDAO.getDogovor(id));
            }
            catch (Exception ex)
            {
                Logger.Log.Error("Ошибка: ", ex);
                return View(dogovorDAO.getDogovor(id));
            }
        }
    }
}
```

Класс */Controllers/StrController

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Agent.DAO;
using Agent.Models;
using Microsoft.AspNet.Identity;

namespace Agent.Controllers
{
    public class StrController : Controller
    {
        GroupDogDAO groupDAO = new GroupDogDAO();
        StrSlDAO strslDAO = new StrSlDAO();
        DogovorDAO dogovorDAO = new DogovorDAO();

        public ActionResult Index(int? id)
        {
            ViewData["Groups"] = groupDAO.GetAllGroups();
            var strsl = id == null ? strslDAO.GetAllStrSl() : strslDAO.GetAllStrSl().Where(x
=> x.GroupDog.Id == id);
            if (!Request.IsAjaxRequest())
                return View(strsl);
            else
                return PartialView("StrSlList", strsl);
```

```
        }

        public ActionResult MyObject(int? id)
        {
            ViewData["Groups"] = groupDAO.GetAllGroups();
            var strsl = id == null ? strslDAO.GetAllStrSl() : strslDAO.GetAllStrSl().Where(x
=> x.GroupDog.Id == id);
            string userid = User.Identity.GetUserId();
            var myobject = strslDAO.GetMyObject(userid);
            if (!Request.IsAjaxRequest())
                return View(myobject);
            else

                return PartialView("StrSlList", strsl);
        }

        public ActionResult Details(int id)
        {
            return View(strslDAO.getStrSl(id));
        }
        protected bool ViewDataSelectList(int GroupId)
        {
            var groups = groupDAO.GetAllGroups();
            ViewData["GroupId"] = new SelectList(groups, "Id", "Name", GroupId);
            return groups.Count() > 0;
        }


        [HttpGet]
        public ActionResult Create(string id)
        {
            string userId = User.Identity.GetUserId();

            StrSl strsl = new StrSl();
            using (Entities ent = new Entities())
            {
                strsl.IDKl = userId;
            }
            return View(strsl);
        }
        [HttpPost]
        public ActionResult Create(StrSl model)
        {
            try
            {
                strslDAO.CreateStrSl(model);
                return RedirectToAction("MyObject");
            }
            catch (Exception ex)
            {
                Logger.Log.Error("Ошибка: ", ex);
            }
            return RedirectToAction("Create");
        }

        public ActionResult Edit(int id)
        {
            StrSl Str = strslDAO.getStrSl(id);
            if (!ViewDataSelectList(Str.GroupDog.Id))
                return RedirectToAction("MyObject");
            return View(strslDAO.getStrSl(id));
        }

        [HttpPost]
        public ActionResult Edit( StrSl Str)
```

```
        {
            ViewDataSelectList(-1);
            try
            {
                if (ModelState.IsValid && strslDAO.updateStrSl( Str))
                    return RedirectToAction("MyObject");
                else
                    return View(Str);
            }
            catch (Exception ex)
            {
                Logger.Log.Error("Ошибка: ", ex);
                return View(Str);
            }
        }

        public ActionResult Delete(int id)
        {
            return View(strslDAO.getStrSl(id));
        }

        [HttpPost]
        public ActionResult Delete(int id, StrSl Str)
        {
            try
            {
                if (strslDAO.deleteStrSl(id))
                    return RedirectToAction("MyObject");
                else
                    return View(strslDAO.getStrSl(id));
            }
            catch (Exception ex)
            {
                Logger.Log.Error("Ошибка: ", ex);
                return View(strslDAO.getStrSl(id));
            }
        }
        [Authorize]
        public ActionResult Confirm(int id)
        {
            Dogovor dogovor = dogovorDAO.getDogovor(id);
            dogovor.IDGroup = 3;
            dogovorDAO.UpdateGroup(dogovor);
            return RedirectToAction("Index");
        }
        public ActionResult Reject(int id)
        {
            Dogovor dogovor = dogovorDAO.getDogovor(id);
            dogovor.IDGroup = 4;
            dogovorDAO.UpdateGroup(dogovor);
            return RedirectToAction("Index");
        }
    }
}
```

Класс */Controllers/TarifController

```
using Agent.DAO;
using Agent.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Agent.Controllers
```

```
{
    public class TarifController : Controller
    {
        TarifDAO tarifDAO = new TarifDAO();
        List<Tarif> tarif;

        [HttpGet]

        public ActionResult Index()
        {
            return View(tarifDAO.GetAllTarif());
        }

        [HttpGet]   // Атрибут, используемый для ограничения метода таким образом, чтобы
этот метод обрабатывал только HTTP-заппросы GET
        public ActionResult Details(int id)
        {
            tarif = tarifDAO.GetAllTarif();
            int trueid = 0;
            for (int i = 0; i < tarif.Count; i++) if (tarif[i].Id == id) trueid = i;
            return View(tarif[trueid]);
        }

        [HttpGet]
        public ActionResult Create()
        {
            return View();
        }


        [HttpPost] // Атрибут, используемый для ограничения метода таким образом, чтобы этот
метод обрабатывал только HTTP-запросы Post
        public ActionResult Create([Bind(Exclude = "Id")]Tarif tarif)
        {
            try
            {
                if (tarifDAO.AddTarif(tarif))
                {
                    return RedirectToAction("Index");
                }
                else
                {

                    return View("Create");
                }
            }
            catch (Exception ex)
            {
                Logger.Log.Error("Ошибка: ", ex);
                return View("Create");
            }
        }

        [HttpGet]
        public ActionResult Edit(int id)
        {
            tarif = tarifDAO.GetAllTarif();
            int trueid = 0;
            for (int i = 0; i < tarif.Count; i++) if (tarif[i].Id == id) trueid = i;
            return View(tarif[trueid]);
        }


        [HttpPost]
        public ActionResult Edit(Tarif tarif)
        {
```

```
            if (ModelState.IsValid)
            {
                tarifDAO.EditTarif(tarif);
                return View("Index");
            }
            else
            {
                return View("Ошибка");
            }
        }


        [HttpGet]
        public ActionResult Delete(int id)
        {
            try
            {
                tarif = tarifDAO.GetAllTarif();
                int trueid = 0;
                for (int i = 0; i < tarif.Count; i++) if (tarif[i].Id == id) trueid = i;
                return View(tarif[trueid]);
            }
            catch (Exception ex)
            {
                Logger.Log.Error("Ошибка: ", ex);
                return View("Ошибка");
            }
        }

        [HttpPost]
        public ActionResult Delete(string id)
        {
            try
            {
                int rec_id = Convert.ToInt32(id);
                tarifDAO.DeleteTarif(rec_id);
                return View("Index");
            }
            catch (Exception ex)
            {
                Logger.Log.Error("Ошибка: ", ex);
                return View("Ошибка");
            }
        }

    }
}
```