Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Южно-Уральский государственный университет»

(национальный исследовательский университет)

Институт естественных и точных наук

Факультет математики, механики и компьютерных технологий

Кафедра прикладной математики и программирования

|  |  |
| --- | --- |
|  |  |

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

по дисциплине проектирование баз данных

ЮУрГУ–090304.2017.03.ПЗ КР

ЖКХ

|  |  |
| --- | --- |
|  | Руководитель работы,  \_\_\_\_\_\_\_\_\_\_/Т.Ю. Оленчикова  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2018 г. |
|  | Автор работы  Студент группы ЕТ-414  \_\_\_\_\_\_\_\_\_\_\_\_\_/ A.Н.Курочкин  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2018 г. |
|  |  |

Челябинск 2017

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc503126430)

[Цель работы 3](#_Toc503126431)

[1 ПОСТАНОВКА ЗАДАЧИ 4](#_Toc503126432)

[1.1 Инфологическая модель 4](#_Toc503126433)

[1.2 Пользователи системы 4](#_Toc503126434)

[1.3 Автоматизируемые функции 4](#_Toc503126435)

[2 КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ 5](#_Toc503126436)

[2.1 Построение ER-диаграммы 5](#_Toc503126437)

[2.2 Выявление ограничений и правил поддержания целостности данных 6](#_Toc503126438)

[Объект «Пользователь» 6](#_Toc503126439)

[Объект «Уровень доступа» 6](#_Toc503126440)

[Объект «Счет» 6](#_Toc503126441)

[Объект «Услуга» 6](#_Toc503126442)

[Объект «Тип расчета» 6](#_Toc503126443)

[Объект «Квартира» 7](#_Toc503126444)

[Объект «Право владения» 7](#_Toc503126445)

[Объект «Факт оплаты» 7](#_Toc503126446)

[3 ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ 8](#_Toc503126447)

[3.1 Перевод ER-модели в реляционную форму 8](#_Toc503126448)

[4 ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ 9](#_Toc503126449)

[4.1 Класс Person (Пользователь) 9](#_Toc503126450)

[4.2 Класс Flat (Квартира) 9](#_Toc503126451)

[4.3 Класс Role (Уровень доступа) 10](#_Toc503126452)

[4.4 Класс Possession (Право владения) 10](#_Toc503126453)

[4.5 Класс PayementFact (Факт оплаты) 10](#_Toc503126454)

[4.6 Класс PayementType (Тип расчета) 11](#_Toc503126455)

[4.7 Класс Service (Счет) 11](#_Toc503126456)

[4.8 Класс ServiceInfo (Услуга) 11](#_Toc503126457)

[4.9 Вспомогательный Класс Calculation (Расчет стоимости услуги для пользователя) 12](#_Toc503126458)

[4.10 Вспомогательный Класс ContexHelper (Вспомогательные функции) 13](#_Toc503126459)

[4.11 Вспомогательный Класс EntityContext 14](#_Toc503126460)

[5 ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ 16](#_Toc503126461)

[5.1 Подключение к источнику данных 16](#_Toc503126462)

[5.2 Рабочая область приложения 17](#_Toc503126463)

[6.1 Среда разработки 22](#_Toc503126464)

[6.2 Минимальные системные требования 22](#_Toc503126465)

[6.3 Накладываемые ограничения 22](#_Toc503126466)

[ЗАКЛЮЧЕНИЕ 23](#_Toc503126467)

[ЛИТЕРАТУРА 24](#_Toc503126468)

[ПРИЛОЖЕНИЕ 1. Код приложения 25](#_Toc503126469)

# ВВЕДЕНИЕ

## Цель работы

Необходимо разработать структуру реляционной базы данных для информационной системы «ЖКХ».

Система предназначена для задач, выполняемых в управляющих компаниях. Здесь обеспечивается хранение информации о услугах организации, квартирах, людях, проживающих в этих квартирах и т.д.

# 1 ПОСТАНОВКА ЗАДАЧИ

## Инфологическая модель

В сети ЖКХ для пользователей доступны действия: просмотр счетов за коммунальные услуги, для кассира: подтверждение оплаты счета, для администратора: управление счетами.  
 В данной базе данных хранится информация: о **счетах за ЖКХ** (Информация о типе услуги, кол-во предоставленной услуги (Например для электроэнергии – киловатт-часы), дата выставления счета (за какой месяц счет)), **об услугах** (Название, период действия, цена за единицу, тип расчета (По кв. метрам, по счетчику или по кол-ву прописанных людей), о Квартирах (Адрес), о **правах владения** (тут по этим подразумевается факт наличия части квартиры в собственности у пользователя), о **пользователях** (ФИО, паспорт, логин, пароль, уровень доступа (администратор, кассир или просто пользователь)), о **фактах оплаты услуг ЖКХ пользователем**.  
 Администратору предоставлен доступ ко всей информации базы данных. Администратор по своему усмотрению может менять, удалять или добавлять любой аспект базы данных, будь то пользователи, счета, услуги и т.д.

## 1.2 Пользователи системы

1. Администратор (А);
2. Кассир(К);
3. Пользователь (П).

## 1.3 Автоматизируемые функции

1. Просмотр счетов (П);
2. Подтверждение оплаты пользователем услуги ЖКХ (К);
3. Управление счетами (А);
4. Управление услугами (А);
5. Регистрация пользователей системы (А);
6. Управление правами владения, пользователями(А);
7. Управление правами доступа (А);
8. Управление типами расчетов (А);

# 2 КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ

Данный раздел посвящен проектированию и нормализации глобальной ER-модели. Строится ее графическое представление, специфицируются ограничения и правила поддержания целостности на уровне глобальной модели.

## 2.1 Построение ER-диаграммы

Выделено восемь сильных сущностей: «Пользователи», «Квартиры», «Счета», «Услуги», «Виды расчетов», «Права доступа», «Факт оплаты», «Права владения», пять связей типа «один ко многим» и две связи «один к нулю или одному» (см. рисунок 2.1).

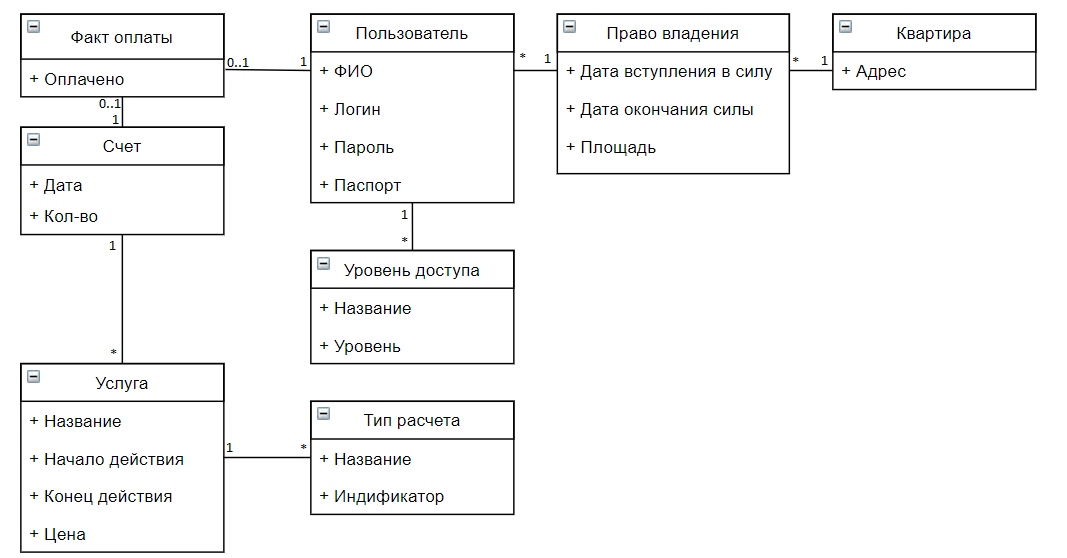


Рисунок 2.1 ­– ER-диаграмма

## 2.2 Выявление ограничений и правил поддержания целостности данных

Данный подраздел посвящен выявлению ограничений и правил поддержания целостности данных, которые будут размещаться в базе данных.

### Объект «Пользователь»

|  |  |
| --- | --- |
| Имя атрибута | Ограничения |
| Логин | Поле должно быть уникально. |
| Пароль | Поле должно быть длинной не более 25символов. |
| ФИО | Поле должно быть длинной не более 25символов. |
| Паспорт | Поле должно быть длинной 10символов. |

### 

### Объект «Уровень доступа»

|  |  |
| --- | --- |
| Имя атрибута | Ограничения |
| Уровень | Поле должно быть не больше 2 и не меньше 0. |
| Название | Поле должно быть уникально. |

### Объект «Счет»

|  |  |
| --- | --- |
| Имя атрибута | Ограничения |
| Дата | Формат мм.дд.гг |
| Кол-во | Поле не должно быть отрицательным. |

### Объект «Услуга»

|  |  |
| --- | --- |
| Имя атрибута | Ограничения |
| Название | Поле должно быть длинной не более 25 символов. |
| Начало действия | Формат мм.дд.гг |
| Конец действия | Формат мм.дд.гг, поле должно быть больше поля начало действия |
| Цена | Поле не должно быть отрицательным. |

### 

### Объект «Тип расчета»

|  |  |
| --- | --- |
| Имя атрибута | Ограничения |
| Название | Поле должно быть длинной не более 25 символов. |
| Индификатор | Поле должно быть не больше 2 и не меньше 0. |

### 

### Объект «Квартира»

|  |  |
| --- | --- |
| Имя атрибута | Ограничения |
| Адрес | Поле должно быть длинной не более 20 символов. |

### Объект «Право владения»

|  |  |
| --- | --- |
| Имя атрибута | Ограничения |
| Дата вступления в силу | Формат мм.дд.гг |
| Дата окончания силы | Формат мм.дд.гг, поле не должно быть меньше чем поле “Дата вступления в силу” |
| Площадь | Поле не должно быть отрицательным. |

### 

### Объект «Факт оплаты»

|  |  |
| --- | --- |
| Имя атрибута | Ограничения |
| Оплачено | Поле должно быть равно либо 0 (нет), либо 1 (да) |

# 3 ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

Данный раздел посвящен проектированию реляционной SQL-модели. Здесь выполняется перевод ER-модели в реляционную форму, специфицируются ограничения и правила поддержания целостности на реляционном уровне.

## 3.1 Перевод ER-модели в реляционную форму

Благодаря использованию Entity Framework и “Code-First” подхода Фреймворк сам создает реляционную модель в базе данных на основе классов (Классы приведены в следующем разделе).

# 

# 4 ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

В этом разделе будут подробно описаны используемые классы (на основе которых Entity Framework создает реляционную модель), функции и процедуры для работы с этими классами.

### 4.1 Класс Person (Пользователь)

public class Person : Indf.Indf

{

[DisplayName("Логин")]

public string Login { get; set; }

[DisplayName("ФИО")]

public string Name { get; set; }

[DisplayName("Паспорт (серия и номер)")]

public string Passport { get; set; }

[DisplayName("Пароль")]

public string Password { get; set; }

[DisplayName("Должность")]

public Role Role { get; set;}

public List<Possession> Possessions { get; set; }

}

Листинг 4.1 – Класс «Person»

### 4.2 Класс Flat (Квартира)

public class Flat : Indf.Indf

{

[DisplayName("Адрес")]

public string Adress { get; set; }

public List<Possession> Possessions { get; set; }

public List<Services.Service> Services { get; set; }

}

Листинг 4.2 – Класс «Flat»

### 4.3 Класс Role (Уровень доступа)

public class Role : Indf.Indf

{

[DisplayName("Уровень доступа")]

public byte Level { get; set; }

[DisplayName("Название должности")]

public string Name { get; set; }

public List<Person> People { get; set; }

}

Листинг 4.3 – Класс «Role»

### 4.4 Класс Possession (Право владения)

public class Possession : Indf.Indf

{

[DisplayName("Вступление в силу")]

public DateTime Start { get; set; }

[DisplayName("Дата расторжения")]

public DateTime End { get; set; }

[DisplayName("Квартира")]

public Flat Flat { get; set; }

[DisplayName("Владелец")]

public Person Owner { get; set; }

[DisplayName("Площадь")]

public decimal Size { get; set; }

}

Листинг 4.4 – Класс «Possession»

### 4.5 Класс PayementFact (Факт оплаты)

public class PayementFact : Indf.Indf

{

[DisplayName("На кого выставлен счет")]

public Person Person { get; set; }

[DisplayName("Услуга")]

public Services.Service Service { get; set; }

[DisplayName("Оплачено")]

public bool Paid { get; set; }

}

Листинг 4.5 – Класс «PayementFact»

### 

### 4.6 Класс PayementType (Тип расчета)

public class PayementType : Indf.Indf

{

[DisplayName("Вид расчета")]

public string Name { get; set; }

[DisplayName("Индификатор вида расчета")]

public short Type { get; set; }

public List<ServiceInfo> ServicesInfos { get; set; }

}

Листинг 4.6 – Класс «PayementType»

### 4.7 Класс Service (Счет)

public class Service : Indf.Indf

{

[DisplayName("Информация о услуге")]

public ServiceInfo Info { get; set; }

[DisplayName("Квартира")]

public Flat Flat { get; set; }

[DisplayName("Дата начисления")]

public DateTime Date { get; set; }

[DisplayName("Кол-во")]

public decimal Count { get; set; }

}

Листинг 4.7 – Класс «Serivce»

### 4.8 Класс ServiceInfo (Услуга)

public class ServiceInfo : Indf.Indf

{

[DisplayName("Тип услуги")]

public string Name { get; set; }

[DisplayName("Начало действия")]

public DateTime StartTime { get; set; }

[DisplayName("Конец действия")]

public DateTime EndTime { get; set; }

[DisplayName("Цена за единицу")]

public decimal Price { get; set; }

[Required, DisplayName("Тип расчета")]

public PayementType paymentType { get; set; }

public List<Service> Services { get; set; }

}

Листинг 4.8 – Класс «SerivceInfo»

### 4.9 Вспомогательный Класс Calculation (Расчет стоимости услуги для пользователя)

public class Calculator

{

ContextHelper helper;

public Calculator(ContextHelper help)

{

helper = help;

}

public decimal Get(Models.Services.Service service, Models.Person person)

{

if (service.Info.paymentType.Type == 0)

{

decimal pl = helper.GetPersonFlatProportion(person, service.Flat);

return service.Info.Price \* service.Count \* pl;

}

else if (service.Info.paymentType.Type == 1)

{

decimal pl = helper.GetPersonFlatProportion(person, service.Flat);

return service.Info.Price \* service.Count \* helper.GetFlatPeopleCount(service.Flat) \* pl;

}

else if(service.Info.paymentType.Type == 2)

{

var res = helper.GetPossessionsByPerson(person).Where(x => x.Flat == service.Flat).Sum(x => x.Size);

return service.Info.Price \* res \* service.Count;

}

else return 0;

}

}

Листинг 4.9 – Класс «Calculation»

Пояснение: Для каждого жильца расчет стоимости услуги будет разный и зависит от кол-ва квадратных метров в собственности пользователя, на которую был выставлен счет за услугу. Система обрабатывает три вида расчетов, а именно:

1. По счетчику: (Стоимость=Цена услуги \* кол-во \* площадь в собственности / Общую площадь квартиры)
2. По количеству жильцов (Стоимость=Цена услуги \* кол-во \* кол-во жильцов \* площадь в собственности / Общую площадь квартиры)
3. По квадратным метрам: (Стоимость=Цена услуги \* кол-во \* площадь в собственности)

### 4.10 Вспомогательный Класс ContexHelper (Вспомогательные функции)

public class ContextHelper

{

private Interfaces.IContext context;

public ContextHelper(Interfaces.IContext cntx)

{

context = cntx;

}

public IEnumerable<Service> GetServicesByFlat(Flat flat)

{

return context.Services.Get.Where(x => x.Flat == flat);

}

public int GetFlatPeopleCount(Flat flat)

{

return context.Possession.Get.Where(x => x.Flat == flat).Select(x => x.Owner).Distinct().Count();

}

public decimal GetPersonFlatProportion(Person person, Flat flat)

{

var FlatPoss = context.Possession.Get.Where(x => x.Flat == flat);

return FlatPoss.Where(x => x.Owner == person).Sum(x => x.Size) / FlatPoss.Sum(x => x.Size);

}

public IEnumerable<Service> GetServicesByPerson(Person person)

{

var Poss = GetPossessionsByPerson(person);

var res = new List<Service>();

foreach (var curPoss in Poss)

{

foreach (var service in context.Services.Get.Where(x => x.Flat == curPoss.Flat)) res.Add(service);

}

return res;

}

public IEnumerable<Possession> GetPossessionsByPerson(Person person)

{

return context.Possession.Get.Where(x => x.Owner == person).ToList();

}

public IEnumerable<Possession> GetPossessionsByFlat(Flat flat)

{

return context.Possession.Get.Where(x => x.Flat == flat).ToList();

}

}

Листинг 4.10 – Класс «ContexHelper»

Пояснение: Класс выполняет вспомогательные функции, такие как:

1. Получить все счета для данной квартиры (GetServicesByFlat(Flat flat))
2. Получить все счета для данного пользователя (GetServicesByPerson(Person person))
3. Получить соотношение площади во владении пользователя для данной квартиры (GetPersonFlatProportion(Person person, Flat flat))
4. Получить совокупную площадь для данной квартиры (GetFlatPeopleCount(Flat flat))
5. Получить количество жильцов для данной квартиры (GetFlatPeopleCount(Flat flat))
6. Получить все права владения для данной квартиры (GetPossessionsByFlat(Flat flat))
7. Получить все права владения для данного пользователя (GetPossessionsByPerson(Person person))

### 4.11 Вспомогательный Класс EntityContext

public class EntityContext : DbContext, Interfaces.IContext

{

public Interfaces.IDBElem<Flat> Flats { get; private set; }

public Interfaces.IDBElem<Person> People { get; private set; }

public Interfaces.IDBElem<Possession> Possession { get; private set; }

public Interfaces.IDBElem<PayementType> PaymentTypes { get; private set; }

public Interfaces.IDBElem<Service> Services { get; private set; }

public Interfaces.IDBElem<ServiceInfo> ServicesInfos { get; private set; }

public Interfaces.IDBElem<Role> RoleInfos { get; private set; }

public IDBElem<PayementFact> PayementFacts { get; private set;}

//-------------------------------------------------------------------

public DbSet<Flat> FlatSet { get; set; }

public DbSet<Person> PeopleSet { get; set; }

public DbSet<Possession> PossessionsSet { get; set; }

public DbSet<PayementType> PayementTypesSet { get; set; }

public DbSet<Service> ServicesSet { get; set; }

public DbSet<ServiceInfo> ServicesInfosSet { get; set; }

public DbSet<Role> RolesSet { get; set; }

public DbSet<PayementFact> PayementFactsSet { get; set; }

//--------------------------------------------------------------------

public void Save()

{

SaveChanges();

}

public EntityContext(string dbName) : base(dbName)

{

//-----------------------------

RoleInfos = new Db.dbRoles(this,RolesSet);

PaymentTypes = new Db.dbPayementTypes(this,PayementTypesSet);

People = new Db.dbPeople(this, PeopleSet);

Flats = new Db.dbFlats(this, FlatSet);

Possession = new Db.dbPossessions(this, PossessionsSet);

ServicesInfos = new Db.dbServiceInfos(this, ServicesInfosSet);

Services = new Db.dbServices(this, ServicesSet);

PayementFacts = new Db.dbPayementFacts(this, PayementFactsSet);

}

}

Листинг 4.11 – Класс «EntityContext»

Пояснение: Класс-Адаптер между системой и Entity Framework.

# 5 ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

Данный раздел посвящен описанию приложения, реализующего описанные ранее автоматизированные функции.

## 5.1 Подключение к источнику данных

Работа с программой начинается с авторизации в системе.

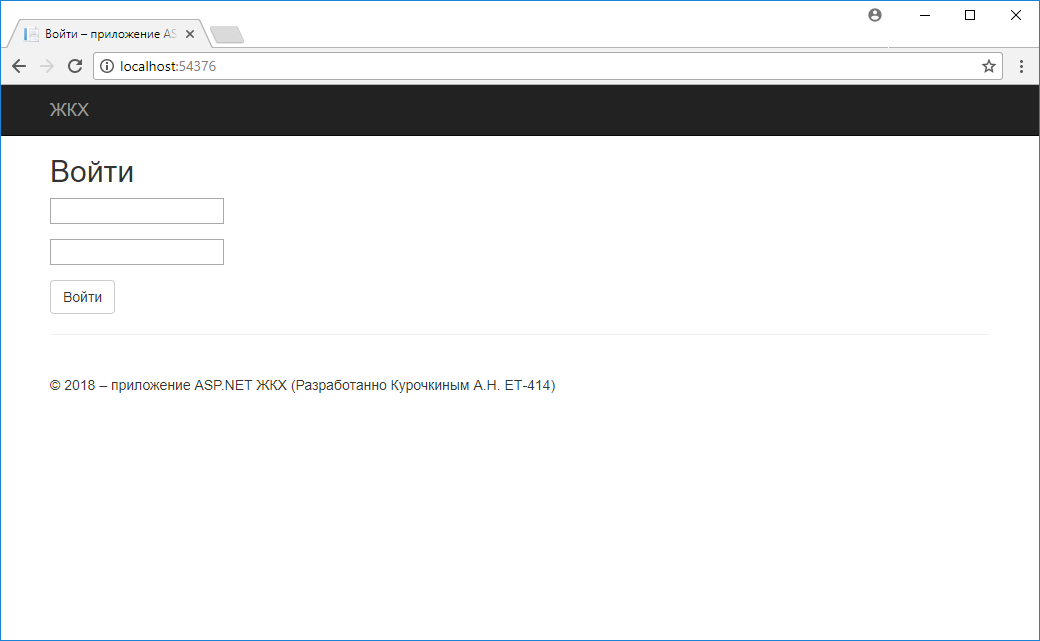


Рисунок 5.1 – Авторизация пользователя

## 5.2 Рабочая область приложения

Главная страница отличается для администратора, кассира или пользователе только наличием\отсутствием ссылки на вход в панель администратора или панель кассира

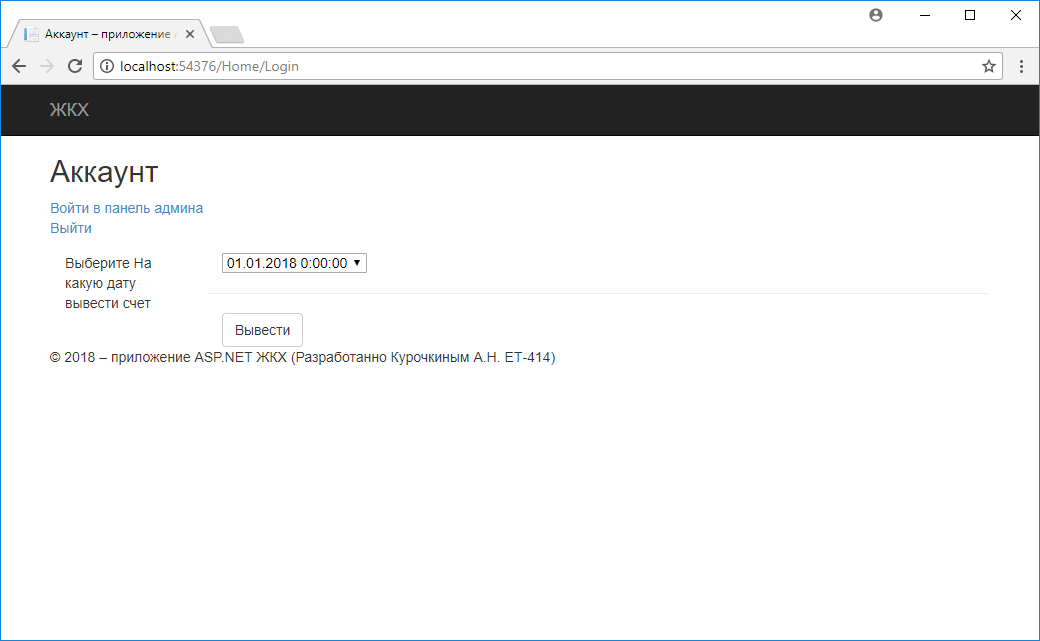


Рисунок 5.3 – Главнная страница для пользователя (Администратора)

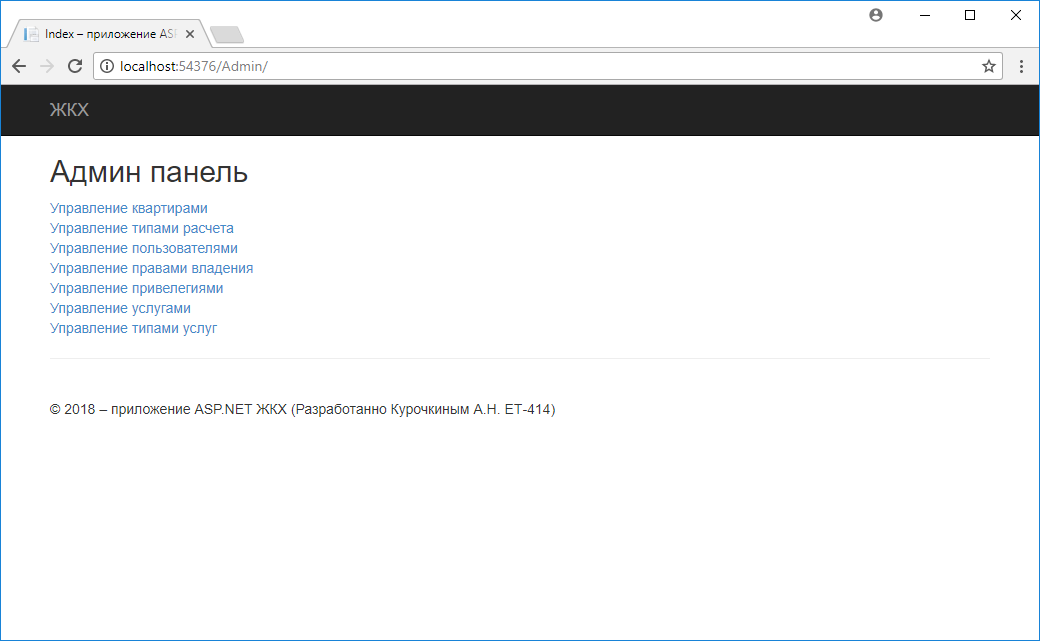


Рисунок 5.4 ­– Панель администратора

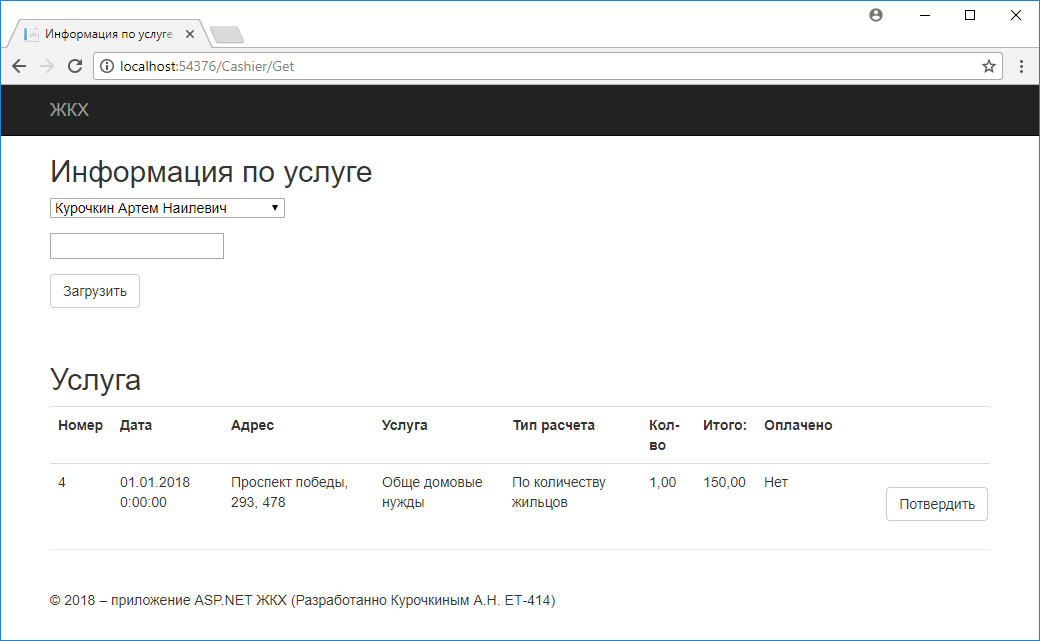


Рисунок 5.5 ­– Панель кассира

Кассир может загрузить счет услуги для пользователя выбрав его в верхнем выпадающем списке (рис. 5.5) и введя индикационный номер счета услуги в область ввода ниже, после чего загрузиться запрашиваемая информация. Кассир может подтвердить факт оплаты нажав на кнопку подтвердить.

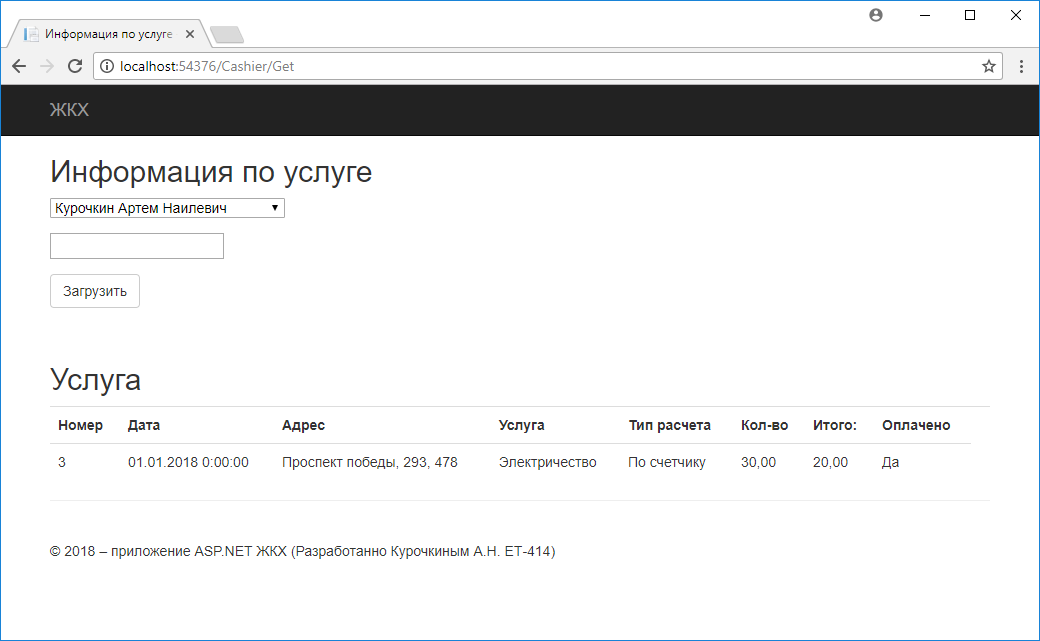


Рисунок 5.6 ­– Оплаченный счет в панели кассира

Пользователь может выбрать (рис 5.3) на какую дату вывести счета. После нажатия кнопки “Вывести” выведется все счета пользователя на эту дату (рис 5.7)

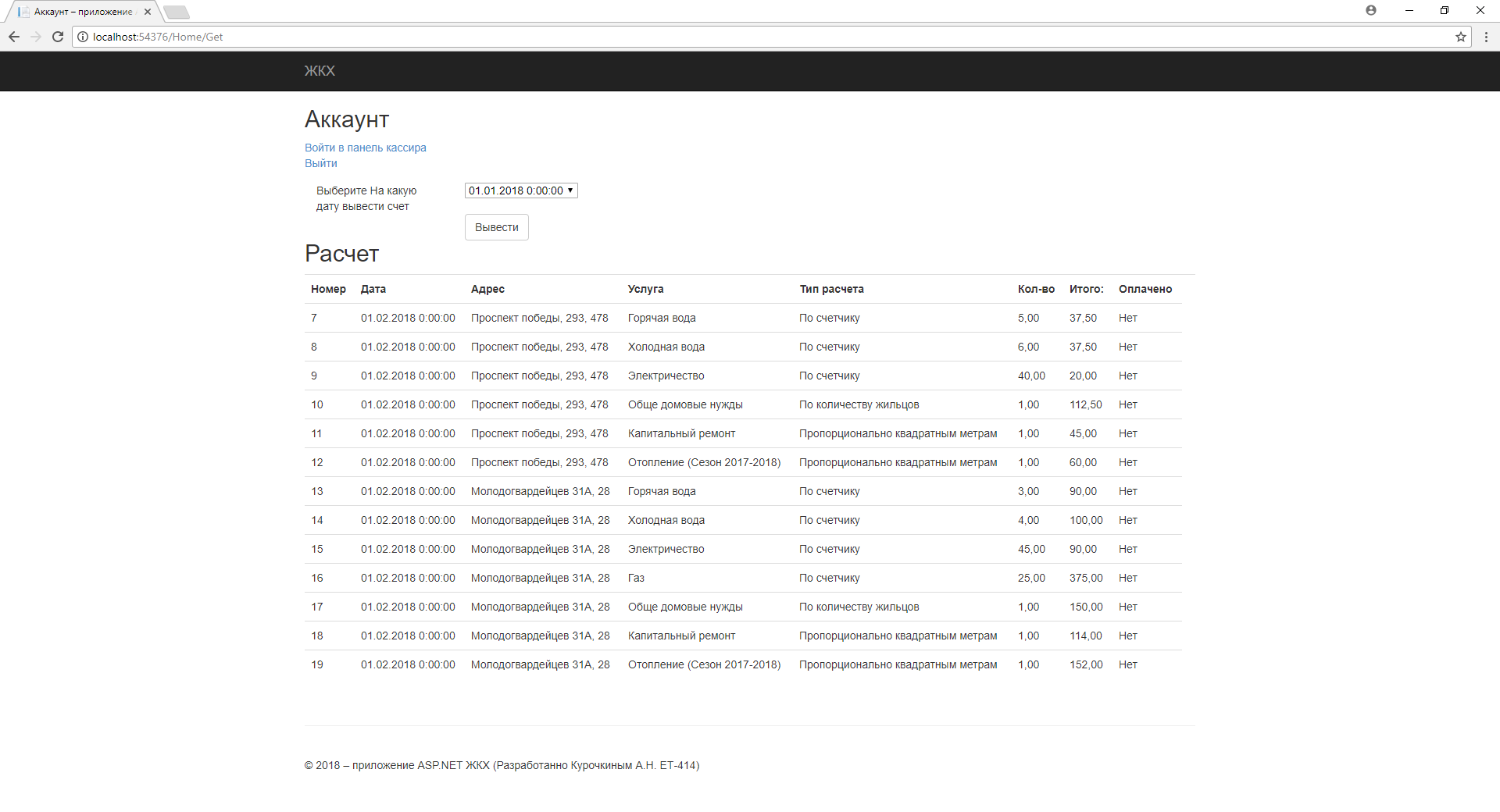


Рисунок 5.7 ­– Счета Пользователя

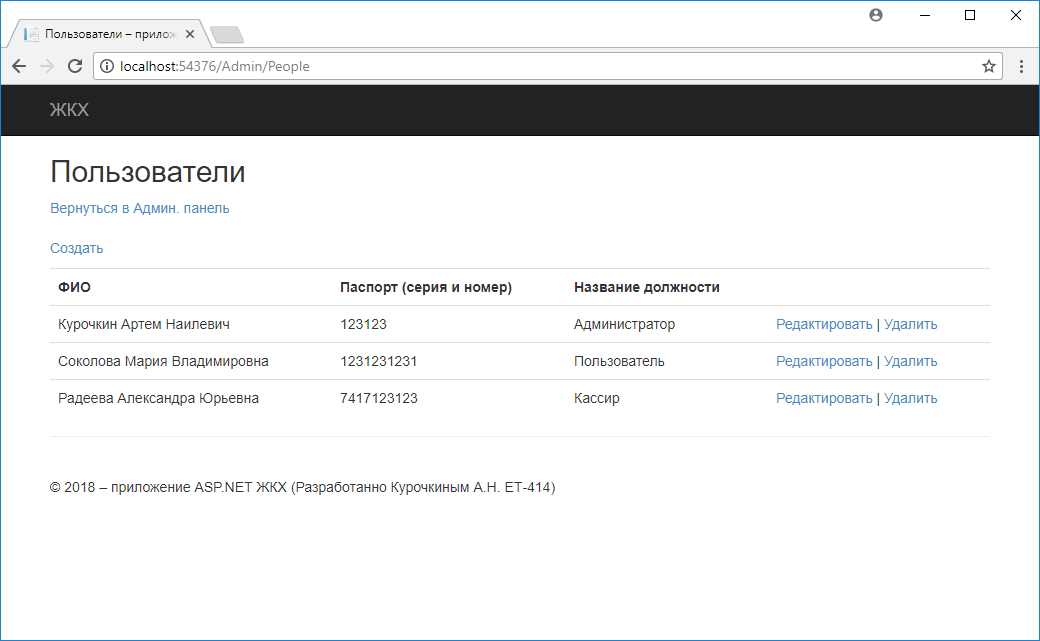


Рисунок 5.8 ­– Вкладка пользователи в админ панели

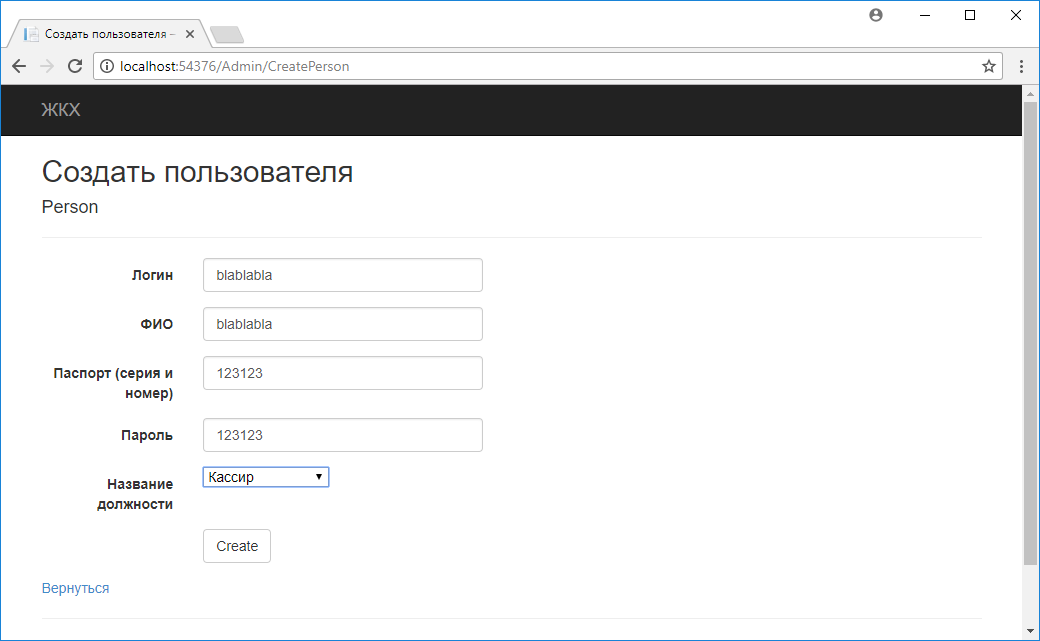


Рисунок 5.9 ­– Создание пользователя в админ панели

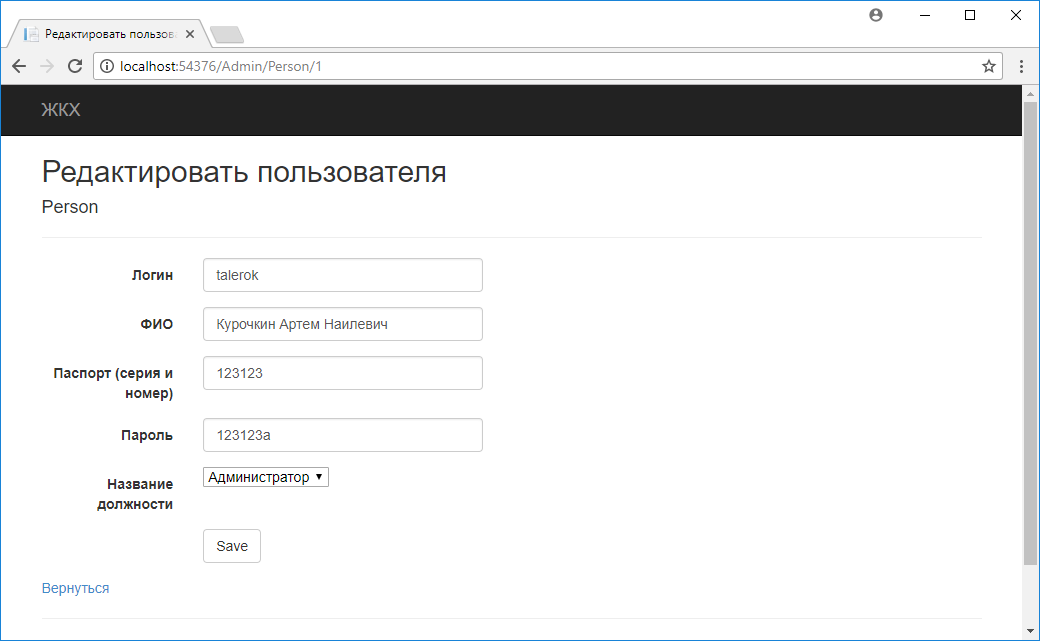


Рисунок 5.10 ­– Редактирование пользователя в админ панели

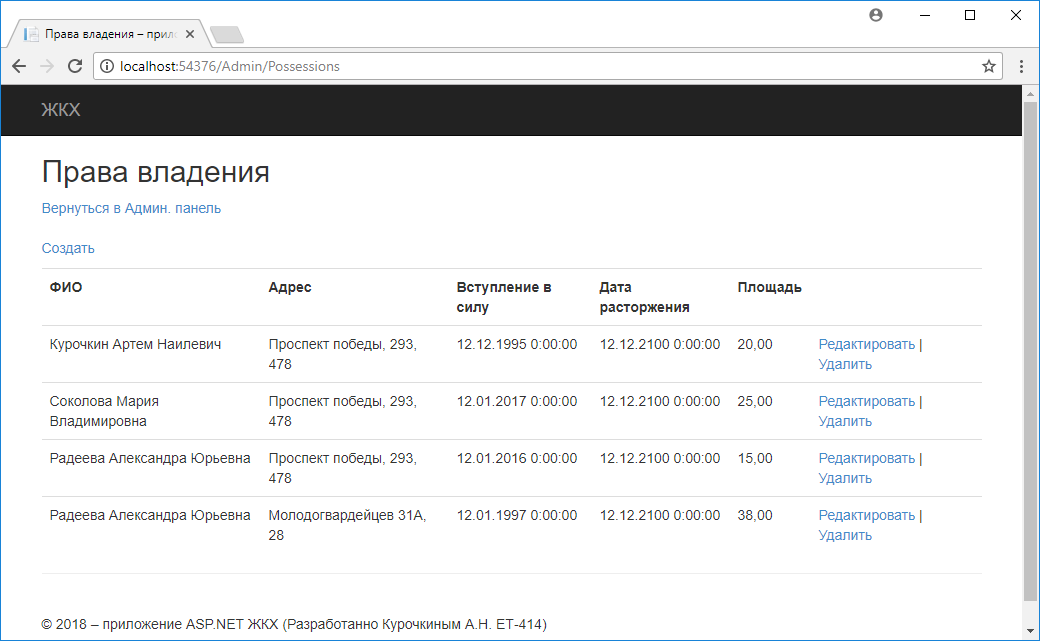


Рисунок 5.11 ­– Вкладка права владения в админ панели

6 ОПИСАНИЕ ПРОГРАММЫ

## 6.1 Среда разработки

Приложение написано в Visual Studio 2017 на C#. Для создания базы данных использован Microsoft SQL Server MS 2014, Entity Framework.

## 6.2 Минимальные системные требования

* Наличие не менее 256 мегабайт оперативной памяти;
* Наличие не менее 50 мегабайт свободного места на жестком диске;
* Наличие операционной системы не ниже Windows7;
* Наличие установленного .NET Framework не ниже версии 3.5;
* Наличие подключения к SQL серверу.

## 6.3 Накладываемые ограничения

За исключением системных требований никаких других ограничений на программу не накладываются.

# ЗАКЛЮЧЕНИЕ

В результате проделанной работы выполнены поставленные задачи, а именно:

разработана структура реляционной базы данных для информационной системы «ЖКХ», а также спроектировано веб приложение для работы с этой системой.

# ЛИТЕРАТУРА

1. Нейгел, К. C# 2008 и платформа .NET 3.5 для профессионалов / К. Нейгел [и др.] – М.: ООО “И.Д. Вильямс”, 2009. – 1392 с.
2. Евсеева, О.Н. Работа с базами данных на языке C#: учебное пособие / О.Н. Евсеева, А.Б. Шамшев. ­– Ульяновск, 2009. – 171 с.

# ПРИЛОЖЕНИЕ 1. Код приложения

Исходный код проекта приведен в электронном виде в папке «Грузоперевозки».