**Оглавление**

[Введение 2](#_Toc66062729)

[Анализ предметной области 3](#_Toc66062730)

[Проектирование схемы данных 7](#_Toc66062731)

[Реализация базы данных в среде SQL Server 8](#_Toc66062732)

[Создание таблиц с помощью SQL запроса 8](#_Toc66062733)

[Запросы к таблицам 10](#_Toc66062734)

[Разработка триггеров и хранимых процедур 14](#_Toc66062735)

[Триггер 14](#_Toc66062736)

[Хранимые процедуры 14](#_Toc66062737)

[Разработка клиентского приложения 16](#_Toc66062738)

[Заключение 26](#_Toc66062739)

[Cписок источников 27](#_Toc66062740)

# Введение

Целью данного курсового проекта является разработка базы данных для управляющей компании, а также проектирование пользовательского интерфейса для удобства пользования системой различными группами пользователей.

Применение баз данных является необходимым условием эффективной организации работы железнодорожного вокзала. Благодаря базам данных каждый работник вокзала, а также пассажиры, всегда имеют доступ к актуальной информации. Графический интерфейс упрощает доступ к информации и защищает базу данных от нежелательных изменений.

# Анализ предметной области

Группы пользователей разрабатываемой информационной системы (ИС)

|  |  |
| --- | --- |
| № пп | Наименование пользователя |
| 1 | Диспетчер |
| 2 | Администратор |
| 3 | Бухгалтер |

Функции групп пользователей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № пп | Выполняемая функция | Входные данные | Выходные данные | Функции, которые должны быть реализованы в ИС |
| ***Диспетчер*** | | | | |
| 1 | Прием(создание) заявок от людей | Время заявки, исполнитель, статус заявки(0 – принята, 1 – выполнена), вид работ, клиент | Созданная заявка | Форма заполнения новой заявки |
| 2 | Просмотр списка заявок по времени | Дата и время заявки | Список заявок | Вывод списка заявок |
| 3 | Изменение/удаление заявки человека(исполнитель, статус) | Номер рейса, станция отправления, станция прибытия | Список типов вагонов (купе, плацкарт и тд) и количество оставшихся мест | Форма для изменения информации |
| 4 | Ввод персональных данных клиентов | Имя, номер телефона, адрес работ | Список клиентов | Форма заполнения информации о клиенте |
| 5 | Изменение/удаление данных о клиенте | Имя, номер телефона, адрес | Список клиентов | Форма заполнения информации о клиенте |
| 6 | Вывод списка клиентов | Номер телефона клиента | Список клиентов | Форма для ввода телефонного номера |
| 7 | Создание времени для исполнителя после передачи заявки | Время исполнения | Вывод списка заявок для конкретного исполнителя | Форма для заполнения времени для исполнителя |
| 8 | Просмотр списка заявок для конкретного исполнителя | Паспорт исполнителя | Список заявок для исполнителя | Отображение списка заявок |
| 9 | Изменение личных данных | Ввод новых данных | Измененные данные |  |
| ***Администратор*** | | | | |
| 1 | Создание нового пользователя | Ввод паспортных данных, ФИО, номера телефона, должности пользователя, логина и пароля | Список пользователей | Форма для создания пользователя |
| 2 | Показать список пользователей |  | Список пользователей | Вывод списка пользователей |
| 3 | Изменение/удаление данных пользователя | Новые ФИО, паспортные данные, номер телефона, должность или логин | Обновленный список пользователей | Форма для ввода изменений |
| 4 | Добавление нового дома для планового осмотра и кап ремонта | Адрес дома, дата последней проверки и кап ремонта | Вывод списка домов для проведения осмотра и кап ремонта | Форма для добавления нового дома |
| 5 | Показать список домов для осмотра и кап ремонта |  | Вывод списка домов для проведения осмотра и кап ремонта | Вывод списка домов для проведения осмотра и кап ремонта |
| 6 | Изменение/удаление данных дома | Новая последняя дата осмотра или кап ремонта | Вывод списка домов для проведения осмотра и кап ремонта | Форма для изменения данных даты осмотра и кап ремонта |
| 7 | Создание нового исполнителя | Ввод паспортных данных, ФИО, номера телефона | Список исполнителей | Форма для создания исполнителя |
| 8 | Показать список исполнителей |  | Список исполнителей | Вывод списка исполнителе |
| 9 | Изменение/удаление данных исполнителя | Новые ФИО, паспортные данные, номер телефона | Обновленный список исполнителей | Форма для ввода изменений |
| 10 | Изменение личных данных | Ввод новых данных | Измененные данные |  |
| ***Бухгалтер*** | | | | |
| 1 | Прием запроса на выдачу справки | ФИО владельца квартиры, адрес проживания, статус(0 – принята, 1 – изготовлена, 2 – выдана) | Список заявок на выдачу справки | Форма для создания заявки |
| 2 | Просмотр списка заявок на выдачу справки |  | Список заявок на выдачу справки | Вывод списка заявок на выдачу справки |
| 3 | Изменение данных для выдачи справки | Новый адрес или статус | Список заявок на выдачу справки | Форма для изменения данных для справки |
| 4 | Расчет цены для заявки на работу | Цена исполнения работы | Вывод списка всех работ с ценником | Форма для ввода цены работы |
| 5 | Просмотр списка цен заявок на работу |  | Вывод списка всех работ с ценником | Вывод списка всех работ с ценником |
| 6 | Изменение цены работы | Новая цена работы | Вывод списка всех работ с ценником | Форма для ввода измененной цены на работы |
| 7 | Расчет цены на кап ремонт | Цена на кап ремонт | Вывод списка домов для кап ремонта с ценами |  |
| 8 | Просмотр цен на кап ремонт |  | Список цен на кап ремонт | Список цен на кап ремонт |
| 9 | Изменение стоимости кап ремонта | Новая стоимость кап ремонта | Список цен на кап ремонт | Форма для ввода новой стоимости |
| 10 | Изменение личных данных | Ввод новых данных | Измененные данные |  |

# Проектирование схемы данных



Houses – список домов

HousesOverhaul – список домов для кап ремонта

HousesView – список домов для осмотра

Price – список цен

Overhaul – список цен на кап ремонт

References – список справок

Users – список пользователей

Clients – список клиентов

Bid – список заявок

BidPrice – список цен заявок

Executor – список исполнителей

ExecutorSchedule – сводная таблица исполнителей

LeadTime – время выполнения заказа

Role – список должностей

# Реализация базы данных в среде SQL Server

## Создание таблиц с помощью SQL запроса

CREATE TABLE [dbo].[Bid](

[Bid\_id] [int] IDENTITY(1,1) NOT NULL,

[BidTime] [datetime] NOT NULL,

[ExecutorSchedule\_id] [int] NOT NULL,

[BidStatus] [char](1) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

[TypeWork] [nvarchar](50) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

[Client\_id] [int] NOT NULL,

[BidPrice\_id] [int] NULL,

[User\_id] [int] NOT NULL,

CONSTRAINT [PK\_Bid] PRIMARY KEY CLUSTERED

(

[Bid\_id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

CREATE TABLE [dbo].[BidPrice](

[BidPrice\_id] [int] IDENTITY(1,1) NOT NULL,

[Price] [money] NOT NULL,

CONSTRAINT [PK\_BidPrice] PRIMARY KEY CLUSTERED

(

[BidPrice\_id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

CREATE TABLE [dbo].[Clients](

[Client\_id] [int] IDENTITY(1,1) NOT NULL,

[FIO] [nvarchar](50) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

[Phone] [nchar](11) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

[Adress] [nvarchar](50) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

CONSTRAINT [PK\_Clients] PRIMARY KEY CLUSTERED

(

[Client\_id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

CREATE TABLE [dbo].[Executor](

[Executor\_id] [int] IDENTITY(1,1) NOT NULL,

[FIO] [nvarchar](50) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

[Phone] [nchar](11) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

[Passport] [varchar](10) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

CONSTRAINT [PK\_Executor] PRIMARY KEY CLUSTERED

(

[Executor\_id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

CREATE TABLE [dbo].[ExecutorSchedule](

[ExecutorSchedule\_id] [int] IDENTITY(1,1) NOT NULL,

[Executor\_id] [int] NOT NULL,

[LeadTime\_id] [int] NULL,

CONSTRAINT [PK\_ExecutorSchedule] PRIMARY KEY CLUSTERED

(

[ExecutorSchedule\_id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

CREATE TABLE [dbo].[Houses](

[House\_id] [int] IDENTITY(1,1) NOT NULL,

[HouseView\_id] [int] NOT NULL,

[HouseOver\_id] [int] NOT NULL,

[Price\_id] [int] NULL,

[Adress] [nvarchar](20) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

CONSTRAINT [PK\_Houses] PRIMARY KEY CLUSTERED

(

[House\_id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

CREATE TABLE [dbo].[HousesOverhaul](

[HouseOver\_id] [int] IDENTITY(1,1) NOT NULL,

[TimeStamp] [datetime] NOT NULL,

[ListOfWork] [nvarchar](100) COLLATE Cyrillic\_General\_CI\_AS NULL,

CONSTRAINT [PK\_HousesOverhaul] PRIMARY KEY CLUSTERED

(

[HouseOver\_id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

CREATE TABLE [dbo].[HousesView](

[HouseView\_id] [int] IDENTITY(1,1) NOT NULL,

[TimeStamp] [datetime] NOT NULL,

[Defects] [nvarchar](100) COLLATE Cyrillic\_General\_CI\_AS NULL,

CONSTRAINT [PK\_HousesView] PRIMARY KEY CLUSTERED

(

[HouseView\_id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

CREATE TABLE [dbo].[LeadTime](

[LeadTime\_id] [int] IDENTITY(1,1) NOT NULL,

[Time] [nvarchar](16) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

CONSTRAINT [PK\_LeadTime] PRIMARY KEY CLUSTERED

(

[LeadTime\_id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

CREATE TABLE [dbo].[Overhaul](

[Overhaul\_id] [int] IDENTITY(1,1) NOT NULL,

[Price] [money] NOT NULL,

CONSTRAINT [PK\_Overhaul] PRIMARY KEY CLUSTERED

(

[Overhaul\_id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

CREATE TABLE [dbo].[Price](

[Price\_id] [int] IDENTITY(1,1) NOT NULL,

[BidPrice\_id] [int] NULL,

[User\_id] [int] NOT NULL,

[Overhaul\_id] [int] NOT NULL,

CONSTRAINT [PK\_Price] PRIMARY KEY CLUSTERED

(

[Price\_id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

CREATE TABLE [dbo].[References](

[Reference\_id] [int] IDENTITY(1,1) NOT NULL,

[Client\_id] [int] NOT NULL,

[Status] [char](1) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

[User\_id] [int] NOT NULL,

CONSTRAINT [PK\_References] PRIMARY KEY CLUSTERED

(

[Reference\_id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

CREATE TABLE [dbo].[Role](

[Role\_id] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](20) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

CONSTRAINT [PK\_Role] PRIMARY KEY CLUSTERED

(

[Role\_id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

CREATE TABLE [dbo].[Users](

[User\_id] [int] IDENTITY(1,1) NOT NULL,

[Role\_id] [int] NOT NULL,

[FIO] [nvarchar](60) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

[Passport] [varchar](10) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

[Phone] [nchar](11) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

[Login] [varchar](50) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

[Password] [varchar](64) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,

CONSTRAINT [PK\_Users] PRIMARY KEY CLUSTERED

(

[User\_id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

## Запросы к таблицам

Запросы к таблицам используются в приложении и вынесены в хранимые процедуры:

***Запрос на вывод времени каждого исполнителя***

select es.ExecutorSchedule\_id as NumExec

, FIO

, Phone

, Time

from ExecutorSchedule es

full join Executor e

on e.Executor\_id = es.Executor\_id

full join LeadTime lt

on lt.LeadTime\_id = es.LeadTime\_id

***Вывод номера последнего исполнителя***

select top 1 es.ExecutorSchedule\_id from ExecutorSchedule es order by es.ExecutorSchedule\_id desc

***Вывод айди для конкретизации параметров ввода***

select ExecutorSchedule\_id

from ExecutorSchedule es

full join Executor e

on es.Executor\_id = e.Executor\_id

full join LeadTime lt

on lt.LeadTime\_id = es.LeadTime\_id

where e.FIO = ''

and Phone = ''

and Time is null

***Конкретизация вывода таблицы справок***

select c.FIO as ClientFIO

, r.Status

, u.FIO as UserFIO

, rr.Name

from [References] r

inner join Clients c

on c.Client\_id=r.Client\_id

inner join Users u

on u.User\_id=r.User\_id

inner join Role rr

on rr.Role\_id=u.Role\_id

***Вывод таблицы заявок***

select Bid\_id as NumBid

, e.FIO

, BidTime

, BidStatus

, TypeWork

, Price

from Bid b

left join BidPrice bp

on b.BidPrice\_id=bp.BidPrice\_id

inner join ExecutorSchedule es

on es.ExecutorSchedule\_id=b.ExecutorSchedule\_id

inner join Executor e

on e.Executor\_id=es.Executor\_id

***Вывод таблицы справок с выборкой по клиентам и составителю***

select \* from [References]

where Status = ''

and Client\_id = (select Client\_id from Clients where FIO = '')

and User\_id = (select User\_id from Users where FIO = '')

***Вывод роли для конкретного пользователя***

SELECT Role.Name FROM Users, Role where Users.Role\_id = Role.Role\_id and Login = ''

Вывод списка домов для кап ремонта

select Adress, hv.TimeStamp as HView, ho.TimeStamp as Overhaul, Defects, ListOfWork

from HousesView hv

inner join Houses h

on h.HouseView\_id = hv.HouseView\_id

inner join HousesOverhaul ho

on ho.HouseOver\_id = h.HouseOver\_id

***Вывод списка домов для осмотра по заданному адресу***

select \* from HousesView where HouseView\_id = (select HouseView\_id from Houses where Adress = '')

***Вывод количества «пристроенных» заказов***

SELECT COUNT(es.ExecutorSchedule\_id) FROM Executor e, ExecutorSchedule es WHERE Phone = '' and es.Executor\_id=e.Executor\_id

# Разработка триггеров и хранимых процедур

## Триггер

create trigger priceNotNull

on Overhaul

after insert, update

as

begin

set nocount on;

declare @price int

set @price = (select Price from inserted)

if @price < 0

begin

Rollback Tran

raiserror('Цена не может быть меньше 0', 16, 10)

end

end

На вставку отрицательной стоимости

## Хранимые процедуры

ALTER PROCEDURE [dbo].[DeleteHouses]

@HouseView\_id [int],

@HouseOver\_id [int]

WITH EXECUTE AS CALLER

AS

delete from HousesView where HouseView\_id = @HouseView\_id;

delete from HousesOverhaul where HouseOver\_id = @HouseOver\_id;

ALTER PROCEDURE [dbo].[insertBid]

@execFIO [nvarchar](50),

@clientFIO [nvarchar](50),

@typeWork [nvarchar](50),

@login [nvarchar](50)

WITH EXECUTE AS CALLER

AS

BEGIN TRANSACTION;

INSERT INTO ExecutorSchedule values ((select e.Executor\_id from Executor e where e.FIO = @execFIO), null);

INSERT INTO Bid VALUES (CURRENT\_TIMESTAMP

, (select top 1 es.ExecutorSchedule\_id from ExecutorSchedule es order by es.ExecutorSchedule\_id desc)

, 0

, @typeWork

, (select c.Client\_id from Clients c where c.FIO = @clientFIO)

, null

, (select User\_id from Users where Login = @login));

COMMIT;

ALTER PROCEDURE [dbo].[TimeUpdateExec]

@execID [int],

@leadTime [nvarchar](20)

WITH EXECUTE AS CALLER

AS

BEGIN TRANSACTION;

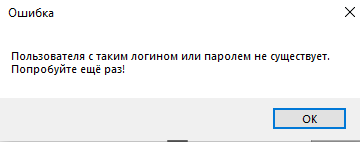
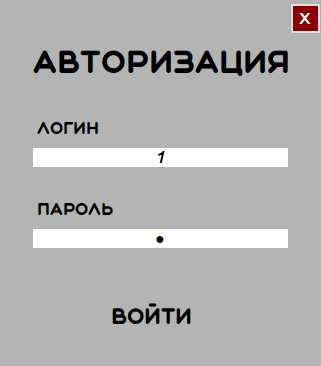
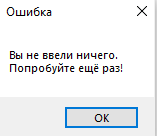
insert into LeadTime values (@leadTime);

update ExecutorSchedule set LeadTime\_id = (select top 1 LeadTime\_id from LeadTime order by LeadTime\_id desc) where ExecutorSchedule\_id = @execID;

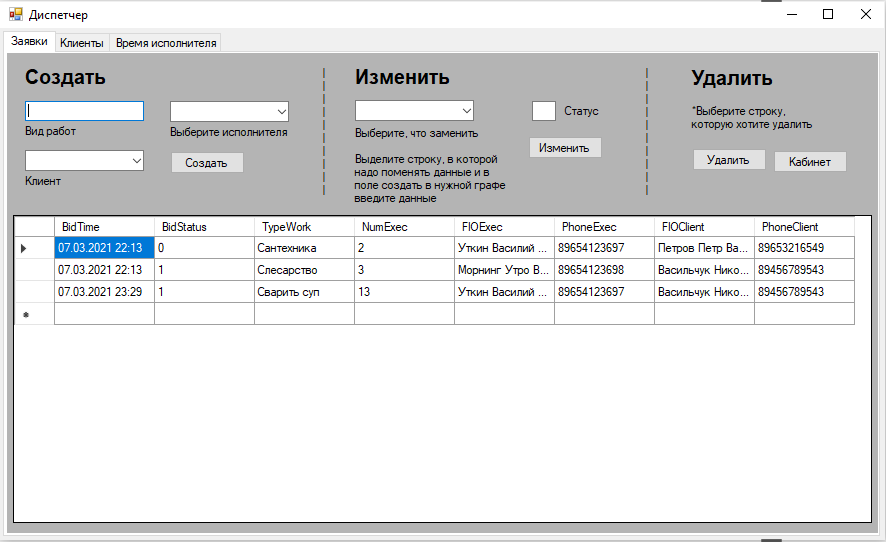
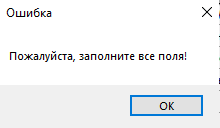
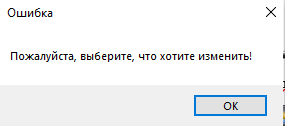
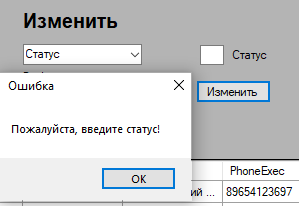
commit;

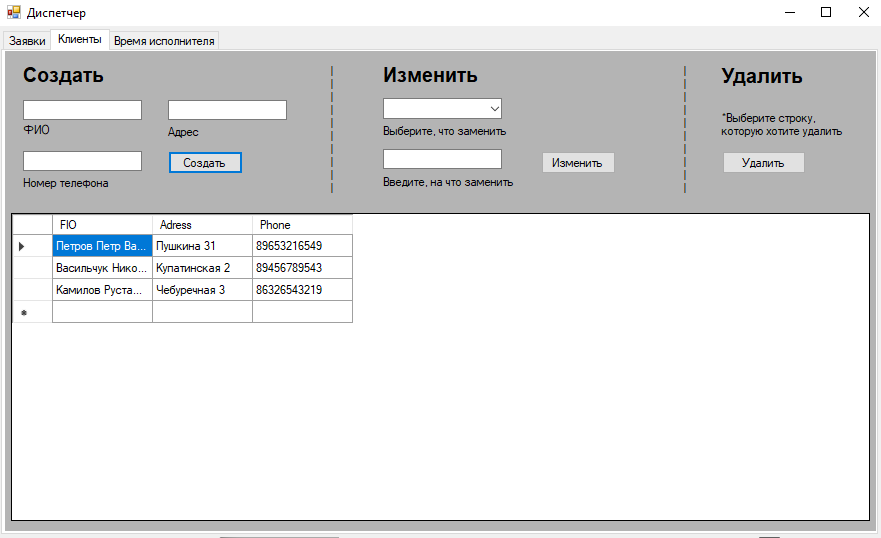
# Разработка клиентского приложения

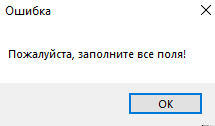
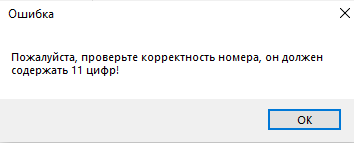
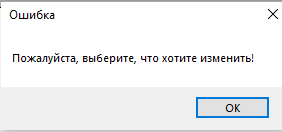
Форма входа:

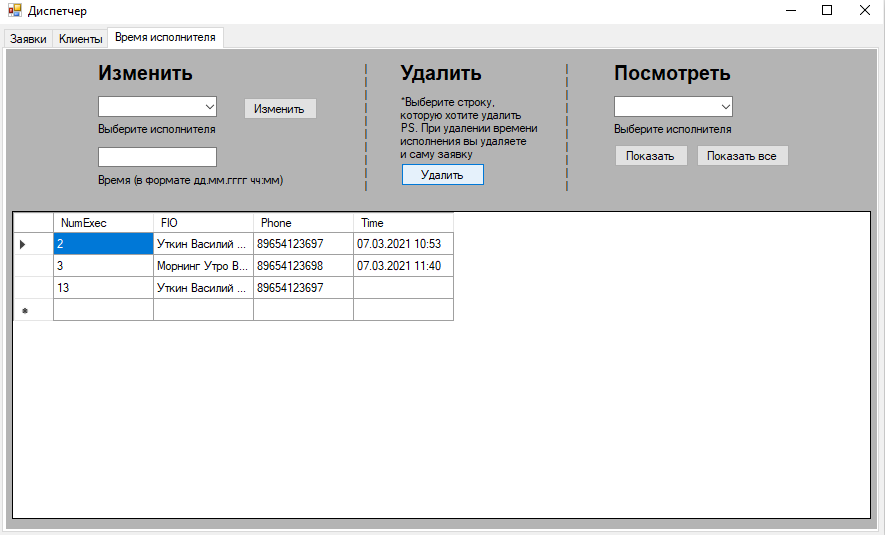
 

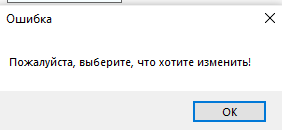
Форма диспетчера

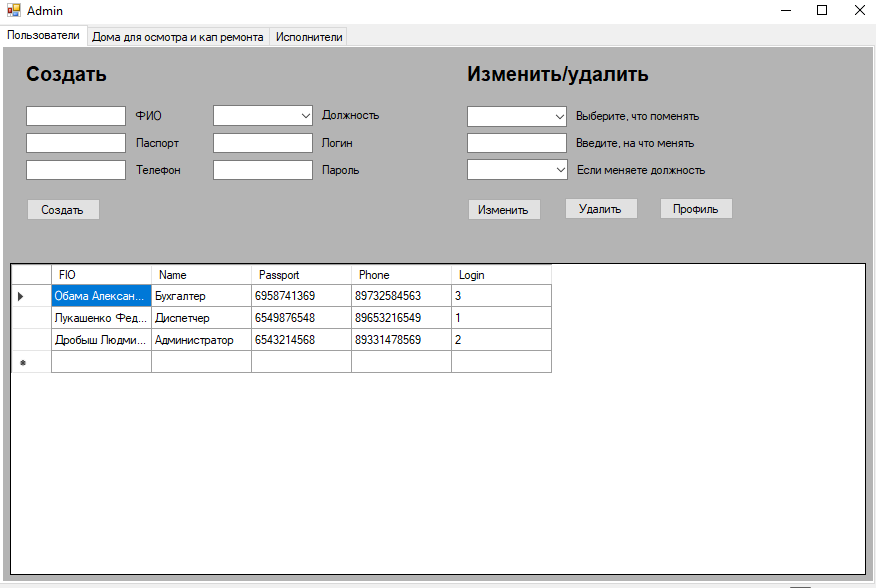


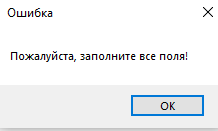
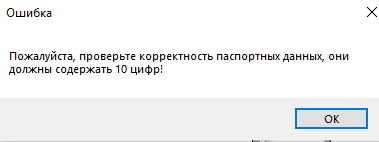
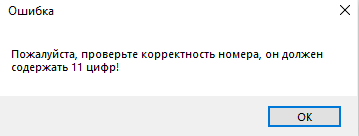
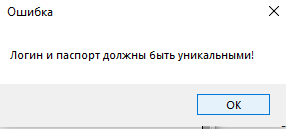
  

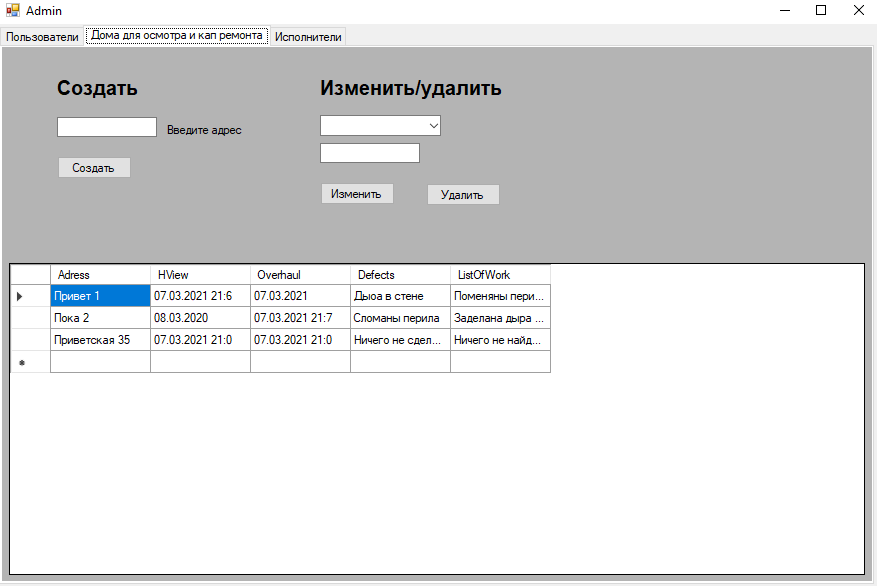
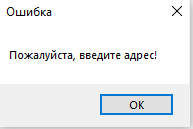
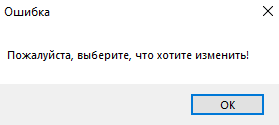
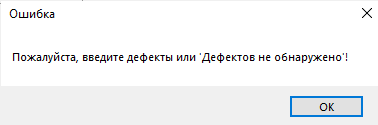
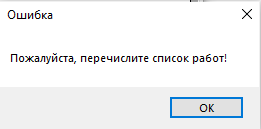


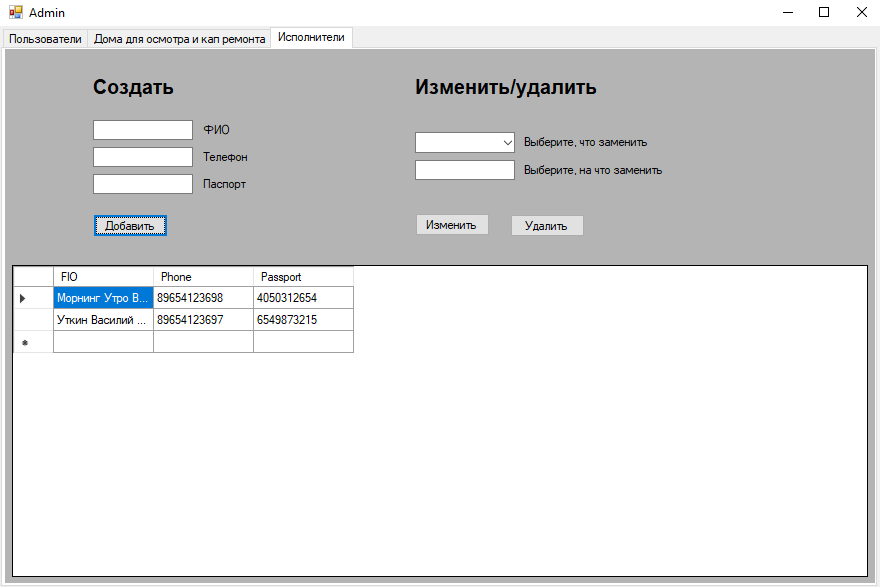
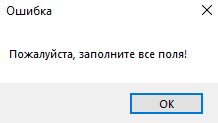
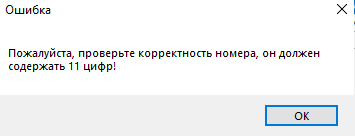
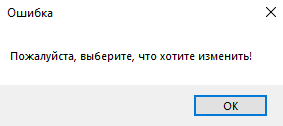
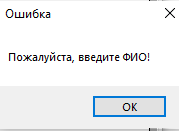
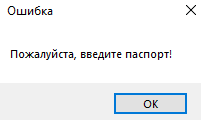


Форма администратора

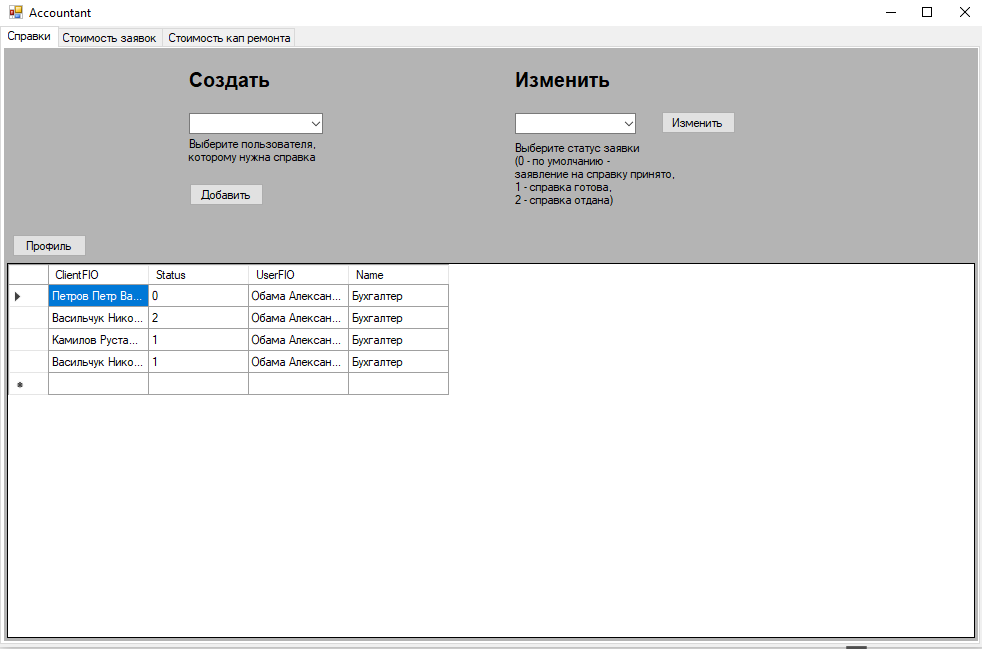
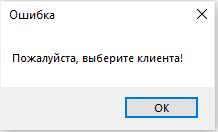
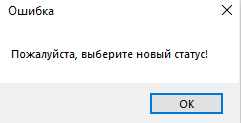
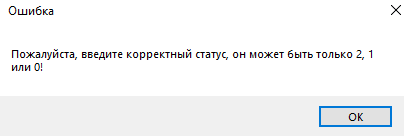


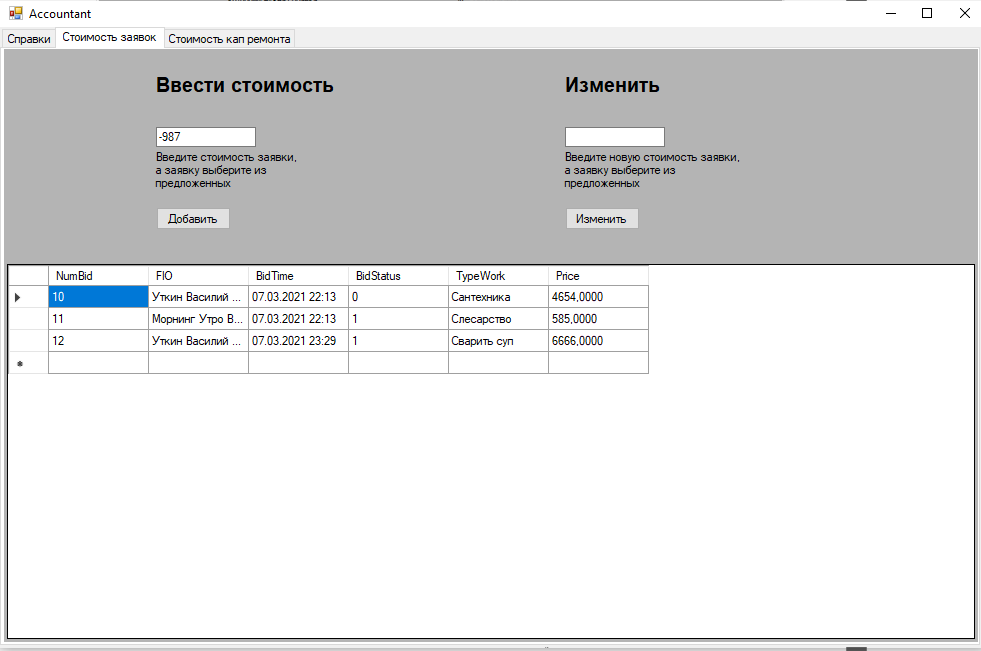
  

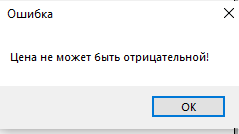
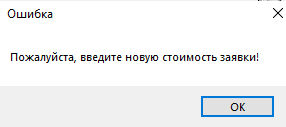
    

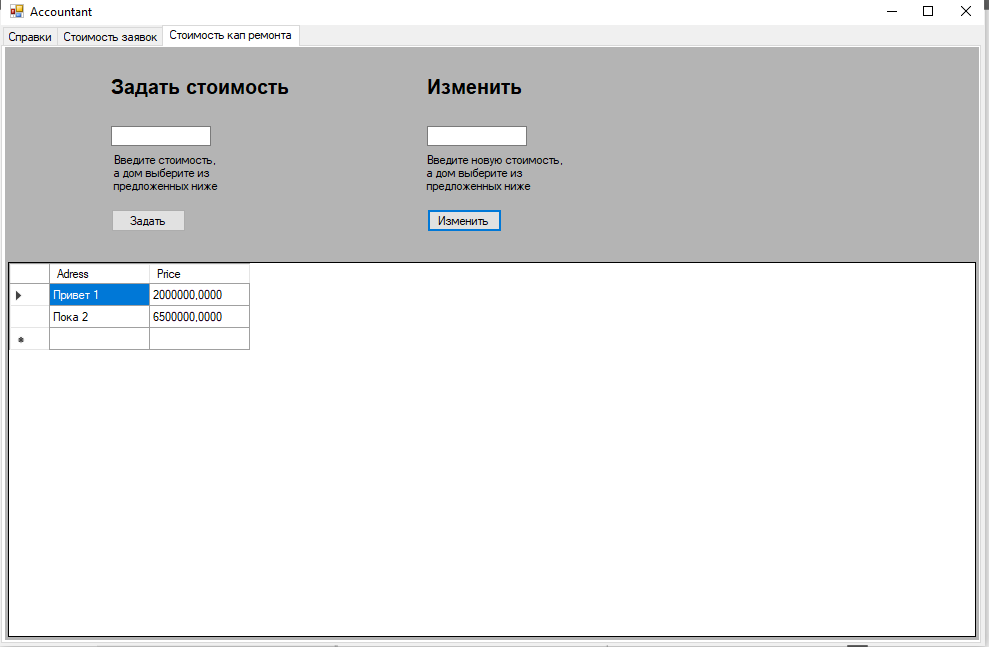
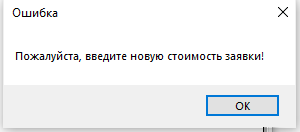
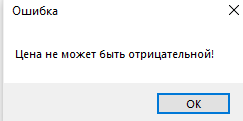
     

Бухгалтер

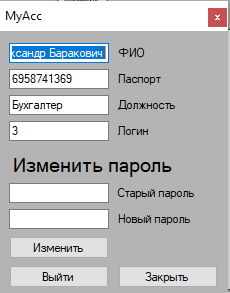
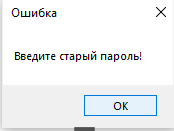
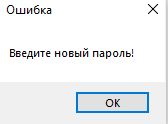
   



Личный кабинет

# Заключение

В результате курсового проектирования была разработана база данных для управляющей компанией. Также был разработан визуальный интерфейс для управления ей.

# Cписок источников

1. Джеффри Рихтер. CLR via C#. Программирование на платформе   
Microsoft.NET Framework 4.5 на языке C#. — 4-е изд. — СПб.: Питер, 2013. — 896 с.

2. <https://msdn.microsoft.com/ru-ru/library/ms188655(v=sql.120).aspx> (дата обращения 19.12.2018)

3. <https://docs.microsoft.com/ru-ru/sql/t-sql/statements/create-view-transact-sql> (дата обращения 12.12.2018)

4. <http://www.sql.ru/> (дата обращения 09.12.2018)

5. <http://english.spbstu.ru/education/programs/degree-programs/bachelor-degree/> (дата обращения 25.12.2018)