**МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ**

**Государственное бюджетное профессиональное образовательное учреждение Московской области «Люберецкий техникум имени Героя Советского Союза, летчика-космонавта Ю. А. Гагарина»**

**ОТЧЕТ**

**ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИЧЕСКОЙ ПОДГОТОВКЕ**

Грибанова Александра Игоревича

(Фамилия, имя, отчество студента)

по профессиональному модулю

ПМ.01 «Разработка модулей программного обеспечения для компьютерных систем»

Специальность 09.02.07 «Информационные системы и программирование»

Курс 4 Группа № 195ИС

Период практической подготовки с «16» февраля 2023 г. по «14» апреля 2023 г.

Руководитель практической подготовки:

от техникума преподаватель Жирнова Ю.В./\_\_\_\_\_\_\_\_\_\_/

*(должность, ФИО)*

Руководитель практической подготовки

от организации зам. директора по финансам Старцева Е.В./\_\_\_\_\_\_\_\_\_\_\_/

*(должность, ФИО)*

Люберцы 2023

**Содержание**

[Введение 4](#_Toc131705669)

[Изучение предметной области 6](#_Toc131705670)

[**WPF 6**](#_Toc131705671)

[Изучение программных средств 6](#_Toc131705672)

[WPF 7](#_Toc131705673)

[Добавление данных 22](#_Toc131705674)

[Удаление данных 24](#_Toc131705675)

[Изменение данных 26](#_Toc131705676)

[Выводы 29](#_Toc131705677)

[Список литературы 30](#_Toc131705678)

# **Введение**

Целью производственной практической подготовки является: закрепление теоретических знаний, полученных при изучении базовых дисциплин; изучение организационной структуры предприятия и действующей на нем системы управления; ознакомление с содержанием основных работ и исследований, выполняемых на предприятии или в организации по месту прохождения практической подготовки.

Целью производственной практической подготовки является реализация и разработка приложения с помощью WPF и Microsoft SQL Server Management Studio (SSMS).

Для достижения данной цели требуется решить следующие задачи:

1. Знакомство с литературой
2. Реализация приложения на WPF

В ходе производственной практической подготовки планируется освоение следующего вида деятельности: «Разработка модулей программного обеспечения для компьютерных систем» (ПК):

ПК 1.1 Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.

ПК 1.2 Разрабатывать программные модули в соответствии с техническим заданием.

ПК 1.3 Выполнять отладку программных модулей с использованием специализированных программных средств

ПК 1.4 Выполнять тестирование программных модулей

ПК 1.5 Осуществлять рефакторинг и оптимизацию программного кода

ПК 1.6 Разрабатывать модули программного обеспечения для мобильных платформ.

ПК 2.1 Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент.

ПК 2.2 Выполнять интеграцию модулей в программное обеспечение.

ПК 2.3 Выполнять отладку программного модуля с использованием специализированных программных средств.

ПК 2.4 Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения.

ПК 2.5 Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования.

## Изучение предметной области

### WPF

**Разработка приложения на WPF** *(Windows Presentation Foundation (WPF)* — аналог WinForms, система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML) помогла мне научиться разрабатывать соединять простейшее приложение с базой данных SQL management Studio (SSMS) — утилита из Microsoft SQL Server 2005 и более поздних версий для конфигурирования, управления и администрирования всех компонентов Microsoft SQL Server. Утилита включает скриптовый редактор и графическую программу, которая работает с объектами и настройками сервера.

## Изучение программных средств

Во время практической подготовки были использованы следующие программные продукты:

* + Visual Studio Community 2019 (далее VS);
  + Microsoft SQL Server 2019;
  + Microsoft SQL Server Management Studio.

Для начала работы в Visual Studio 2019 необходимо выполнить следующий алгоритм действий:

* + открыть программу VS;
  + выбрать необходимые языки программирования, на котором вы хотите писать программы (я выбрал С#). После загрузятся необходимые инструменты для разработки;
  + после открываем VS, ищем вкладку “Создание проекта”, выбираем шаблон и создаем программу (рисунки 1 и 2).

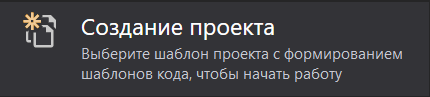


Рисунок. 1. Создание проекта в VS 2019

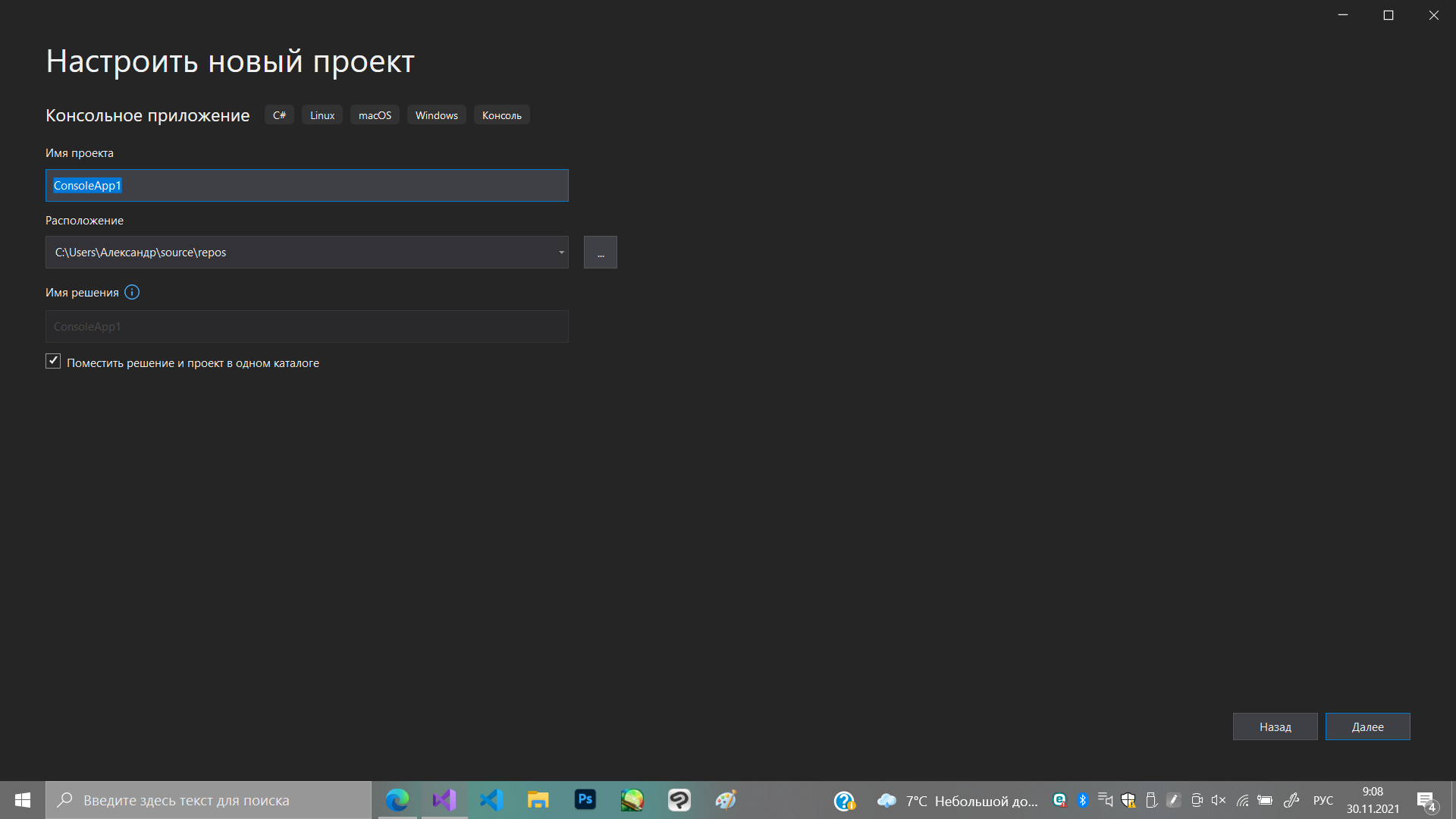


Рисунок. 2. Первоначальная настройка нового проекта

# **WPF**

**Windows Presentation Foundation** — аналог WinForms, система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая подсистема в составе .NET Framework, использующая язык XAML.

Для начала работы нам потребуется IDE Visual Studio 2019, так как она уже установлена на моем компьютере. Хоть эта версия и старая, но в ней хватает того количества функций, которые потребуются в будущем для разработки приложений (рисунок 3).

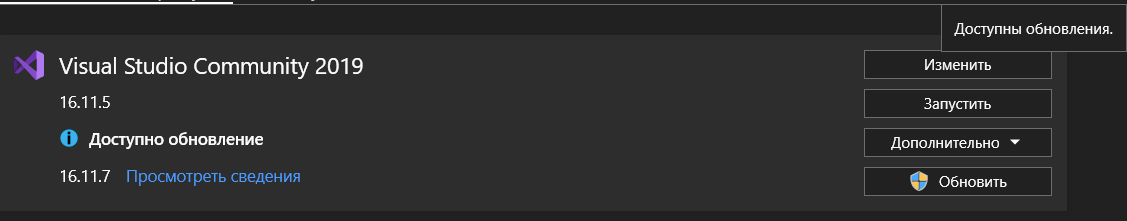


Рисунок 3. VS 2019

После установки SQL сервера, необходимо установить средство взаимодействия с базой данных SQL – SQL Server Management Studio (SSMS), скачивается данное приложение аналогично серверу, с официального сайта Microsoft (рисунок 4).

На рисунке 5 показан процесс установки SSMS.

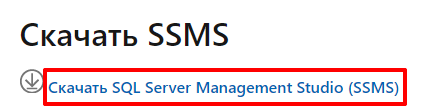


Рисунок 4. Скачивание SSMS с официального сайта Microsoft

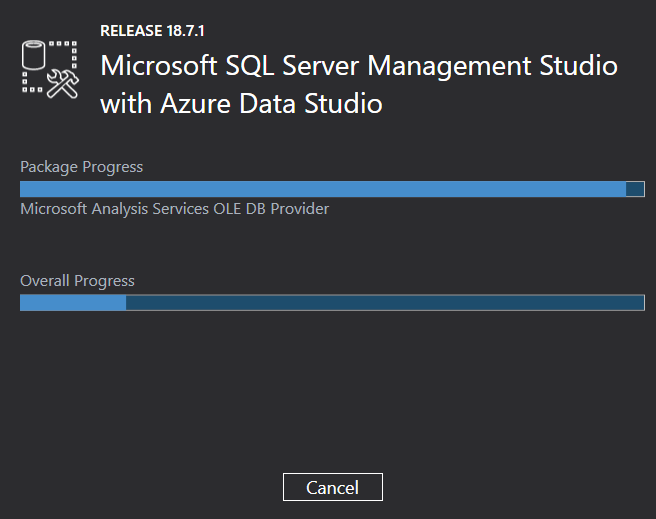


Рисунок 5. Процесс установки SSMS

При прохождении производственной практической подготовки было создано WPF-приложение предприятия, на котором происходило прохождение практической подготовки. WPF-приложение включает в себя:

* Меню с кнопками;
* Авторизацию;
* Вывод данных через элемент DataGrid;
* Инструменты для работы с базой данных предприятия.

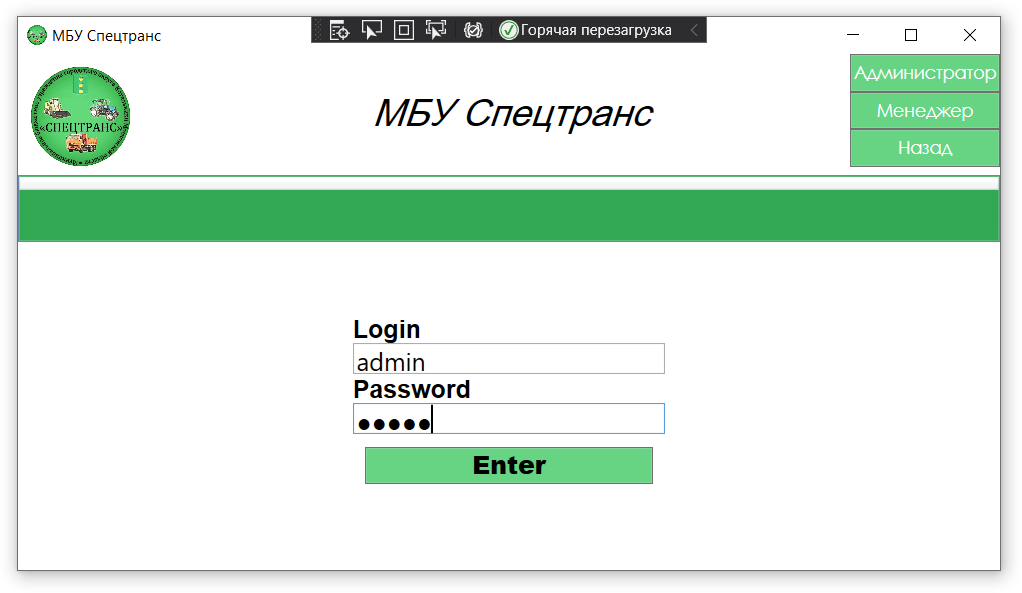


Рисунок 6. Окно авторизации

При разработке WPF-приложения, был создан новый проект в Visual Studio на основе WPF с платформой Net.Framework.

Даём название нашему проекту и переходим к окну с кодом. В окне MainWindow.xaml с помощью кода прописываем дизайн нашего приложения. Также задаем элемент “Frame”, внутри которого будут отображаться все используемые в приложении страницы.

Затем, в файле MainWindow.xaml.cs пишем функции для нашего приложения. Нам нужно, чтобы при начале работы с приложением появлялось окно авторизации. Прописываем данное свойство после InitializeComponent(); (рисунок 7).

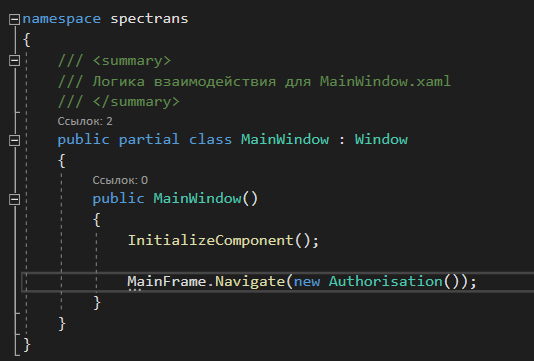


Рисунок 7. Отредактированный файл MainWindow.xaml.cs

После этого переходим в файл “App.xaml”, чтобы задать стиль всему приложению. Внутри тега “ApplicationResourses” прописываем тег “Style”, в котором указываем объект, к которому следует применить изменения, в нашем случае это элемент “Button” (Кнопка). Непосредственно, внутри тега “Style” задаем фоновый цвет всех кнопок приложения и шрифт текста, находящегося внутри них (рисунок 8)

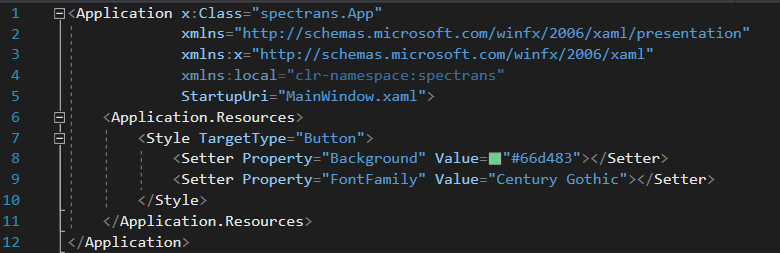


Рисунок 8. Отредактированный файл App.xaml

Далее, создадим дизайн программы с помощью кода, писать его нужно в файле MainWindow.xaml. Код представлен ниже:

<Window x:Class="spectrans.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:spectrans"

mc:Ignorable="d"

Title="МБУ Спецтранс" Height="450"

Width="800"

Icon="/res/logo.ico">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="150"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Image HorizontalAlignment="Left"

Height="80" Margin="10,10,0,0"

VerticalAlignment="Top"

Width="80"

Source="res/logo.png"/>

<StackPanel HorizontalAlignment="Right"

VerticalAlignment="Top">

<Button Content="Администратор"

Width="120"

Height="30"

FontSize="14"

Foreground="White"/>

<Button Content="Менеджер"

Width="120"

Height="30"

FontSize="14"

Foreground="White"/>

<Button Content="Назад"

Width="120"

Height="30"

FontSize="14"

Foreground="White"/>

</StackPanel>

<StackPanel HorizontalAlignment="Center"

VerticalAlignment="Top">

<TextBlock Text="МБУ Спецтранс"

Margin="100,30,0,0"

FontFamily="Arial"

FontSize="30"

FontStyle="Oblique"

Height="35"

Width="319"/>

</StackPanel>

<Frame Grid.Row="1"

x:Name="MainFrame"

NavigationUIVisibility="Hidden"/>

<Grid>

<DataGrid Margin="0,97,0,0"

Background="#32a852"/>

</Grid>

</Grid>

</Window>

После создания дизайна, переходим к получению данных от пользователя. Нам необходимо получить данные от пользователя и проверить их. Для этого пишем проверки вводимых данных

Далее, переходим к тестированию нашей программы. Вводим некорректные данные и видим, что наши поля для ввода «горят красным цветом». Это значит, что наши проверки работают и можно переходить к следующему этапу разработки.

Следующим этапом разработки является создание и подключение Базы данных к нашей программе, для этого потребуется скачать и установить Microsoft SQL Server Management Studio 18. Скачать эту программу можно на [официальном сайте Microsoft](https://docs.microsoft.com/ru-ru/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15#download-ssms). Показано выше.

Следуя инструкциям по установке, производим установки программы и запускаем её. Перед нами открывается окно программы. Вводим требуемые данные и переходим к созданию нашей базы данных (рисунок 9)

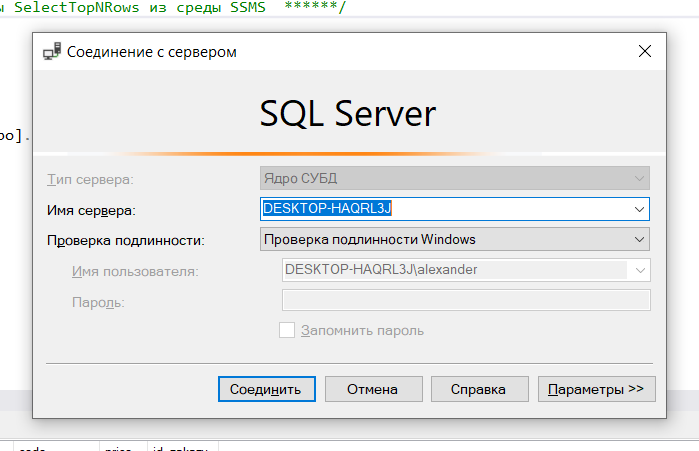


Рисунок 9. Начало работы SSMS

Затем, в обозревателе объектов выбираем «Базы данных» и создаём базу данных

После создания базы данных создаём таблицу «bd» и переходим к подключению базы к нашему приложению. Для этого нужно нажать правой кнопкой мыши по файлу нашего проекта и нажать кнопку «Добавить» - «Создать элемент» (рисунок 10)

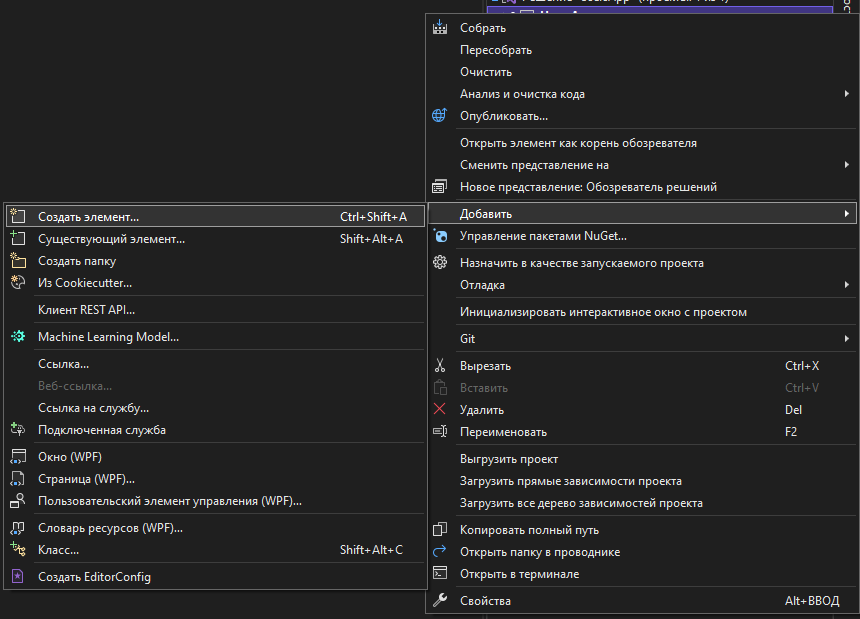


Рисунок 10. Создаём элемент

В окне «Добавление нового элемента» в поле «Поиск» выбираем вкладку «Данные» и выбираем Модель ADO.NET.EDM (рисунок 11)

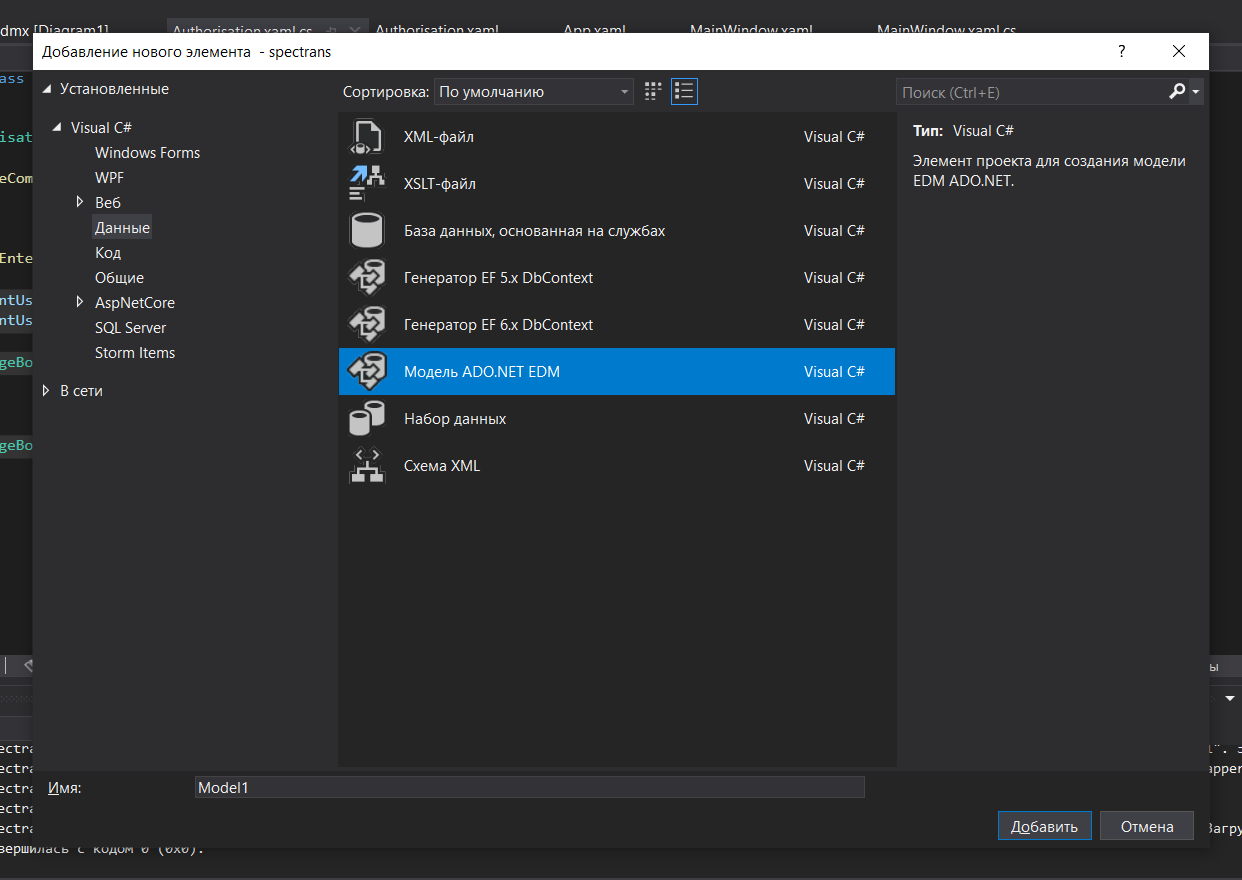


Рисунок 11. Добавляем Модель ADO.NET.EDM

Далее, следуя инструкция, вводим имя нашего сервера, выбираем нашу БД, выбираем таблицу «Users» и затем, происходит подключение

После этого, создаём класс-модель для работы с нашей таблицей

Затем, создаём класс для работы с базой данных

Следующим шагом мы будем добавлять данные в БД. Для этого прописываем следующий код:

**private** **void** **SaveBtn\_Click**(**object** sender, RoutedEventArgs e)

{

**try**

{

uslugi usl = **new** uslugi();

**int** a = Int32.Parse(PriceTbx.Text);

**int** b = Int32.Parse(IdUslugaTbx.Text);

usl.ID =b;

usl.price = a;

usl.code = CodeUsluga.Text;

//данные из комбаря

**var** CurrentUsluga = CmbTitle.SelectedItem **as** uslugi;

usl.title = CurrentUsluga.title;

AppData.db.uslugi.Add(usl);

AppData.db.SaveChanges();

MessageBox.Show("Данные внесены успешно");

NavigationService.GoBack();

}

**catch** (Exception ex)

{

MessageBox.Show("Не удалось добавить данные", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

Затем, добавим проверку введённых пользователем данных и проверим их на соответствие с данными из нашей Базы данных. Код представлен ниже:

**var** CurrentUser = Models.AppData.db.Users.FirstOrDefault(u => u.Login == TbxLogin.Text && u.Password == PbxPassword.Password);

**if** (CurrentUser != **null**)

{

MessageBox.Show("Вы успешно вошли!");

}

**else**

{

MessageBox.Show("Пользователя нет в БД");

}

После написания кода, проверяем работу нашей программы (рисунок 12)

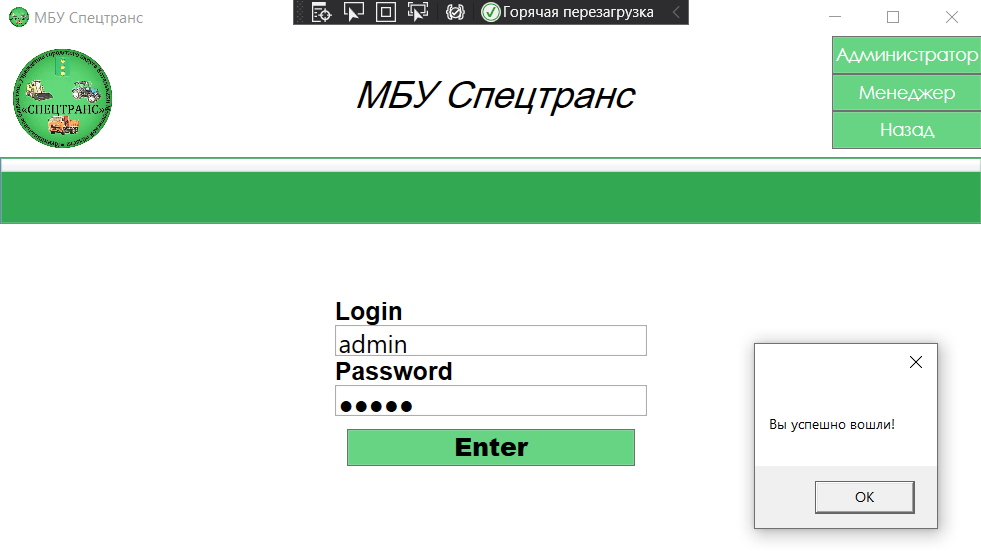


Рисунок 12. Проверка входа пользователя

На данном рисунке показан вариант успешного входа в приложения в аккаунт администратора. В следующем примере рассмотрим случай, когда указанного логина нет в базе, но пользователь пытается через него войти. В таком случае появляется окно со следующим сообщением (рисунок 13).

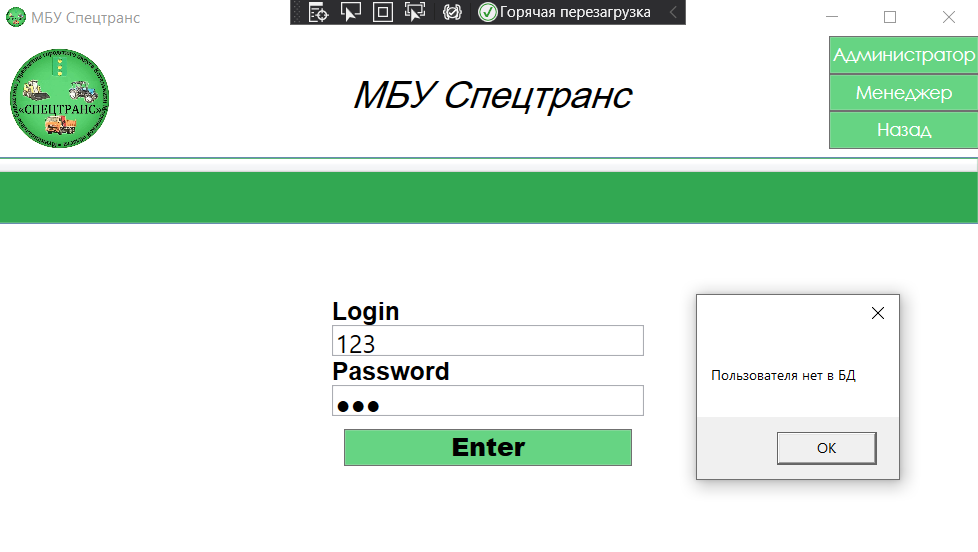


Рисунок 13. Попытка входа с несуществующим логином и паролем

После создания окна авторизации, можно перейти к созданию «Кабинета пользователя». Для этого нам необходимо настроить переход между страницами. Код представлен ниже:

**public** **partial** **class** **MainWindow** : Window

{

**public** **MainWindow**()

{

InitializeComponent();

MainFrame.Navigate(**new** DataPage());

}

**private** **void** **AdminBtn\_Click**(**object** sender, RoutedEventArgs e)

{

MainFrame.Navigate(**new** Authorisation());

}

**private** **void** **UserBtn\_Click**(**object** sender, RoutedEventArgs e)

{

MainFrame.Navigate(**new** Authorisation());

}

**private** **void** **BackBtn\_Click**(**object** sender, RoutedEventArgs e)

{

MainFrame.GoBack();

}

}

Далее, создадим само окно «Кабинет пользователя» (рисунок 13).

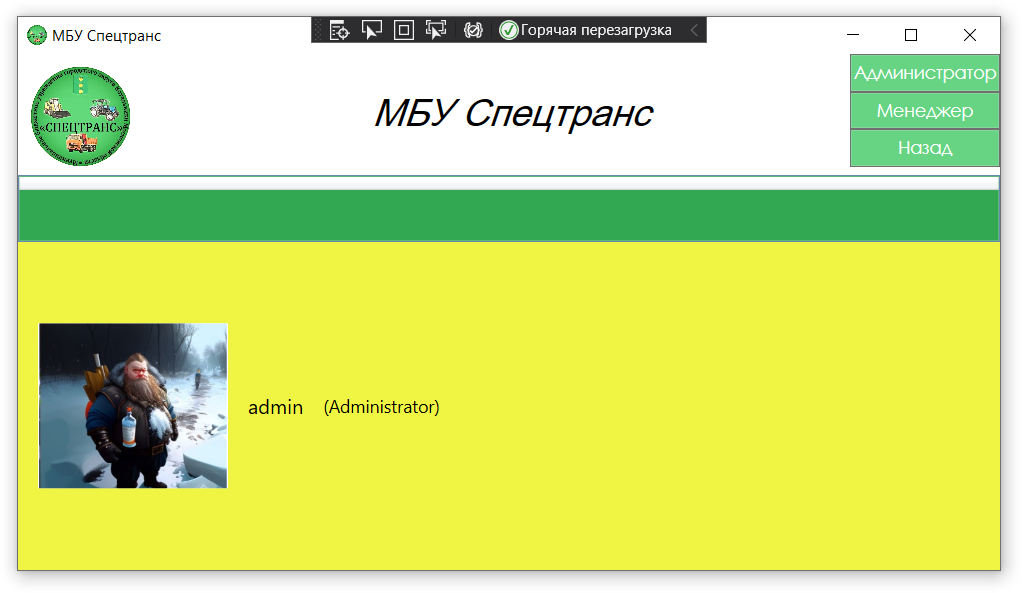


Рисунок 13. Кабинет пользователя

Прописываем код вывода всех доступных услуг и проверяем работу программы. Данные отобразятся в DataGrid в виде таблицы и будут наглядно видны пользователю программы (рисунок 14):

<Page x:Class=”spec.Pages.DataPage”

xmlns=”http://schemas.microsoft.com/winfx/2006/xaml/presentation”

xmlns:x=”http://schemas.microsoft.com/winfx/2006/xaml”

xmlns:mc=”http://schemas.openxmlformats.org/markup-compatibility/2006”

xmlns:d=”http://schemas.microsoft.com/expression/blend/2008”

xmlns:local=”clr-namespace:spec.Pages”

mc:Ignorable=”d”

d:DesignHeight=”450” d:DesignWidth=”800”

Title=”DataPage” Loaded=”Page\_Loaded”>

<Grid>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition Height=”70”/>

</Grid.RowDefinitions>

<DataGrid Grid.Row=”0”

x:Name=”DgridUslugi”

AutoGenerateColumns=”False”>

<DataGrid.Columns>

<DataGridTextColumn Header=”id” Binding=”{Binding ID}” Width=”\*”></DataGridTextColumn>

<DataGridTextColumn Header=”Наименование” Binding=”{Binding title}” Width=”\*”></DataGridTextColumn>

<DataGridTextColumn Header=”Код заказа” Binding=”{Binding code}” Width=”\*”></DataGridTextColumn>

<DataGridTextColumn Header=”Стоимость” Binding=”{Binding price}” Width=”\*”></DataGridTextColumn>

<DataGridTemplateColumn Width=”auto”>

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button Content=”Редактировать” Name=”BtnEdit” Click=”BtnEdit\_Click”></Button>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

<StackPanel

Grid.Row=”2”

HorizontalAlignment=”Center”

VerticalAlignment=»Center»

Orientation=»Horizontal»>

<!–тут отображается таблица из бд, все кратко и по делу🡪

<Button Content=”Добавить”

Name=”BtnAdd”

Click=”BtnAdd\_Click”

Width=”150” Height=”30”/>

<!–добавление данных и переход на соответсвующую страницу🡪

<Button Content=”Удалить”

Name=”BtnDelete”

Click=»BtnDelete\_Click»

Width=»150»/>

<!–удаление данных прямо из таблицы🡪

</StackPanel>

</Grid>

</Page>

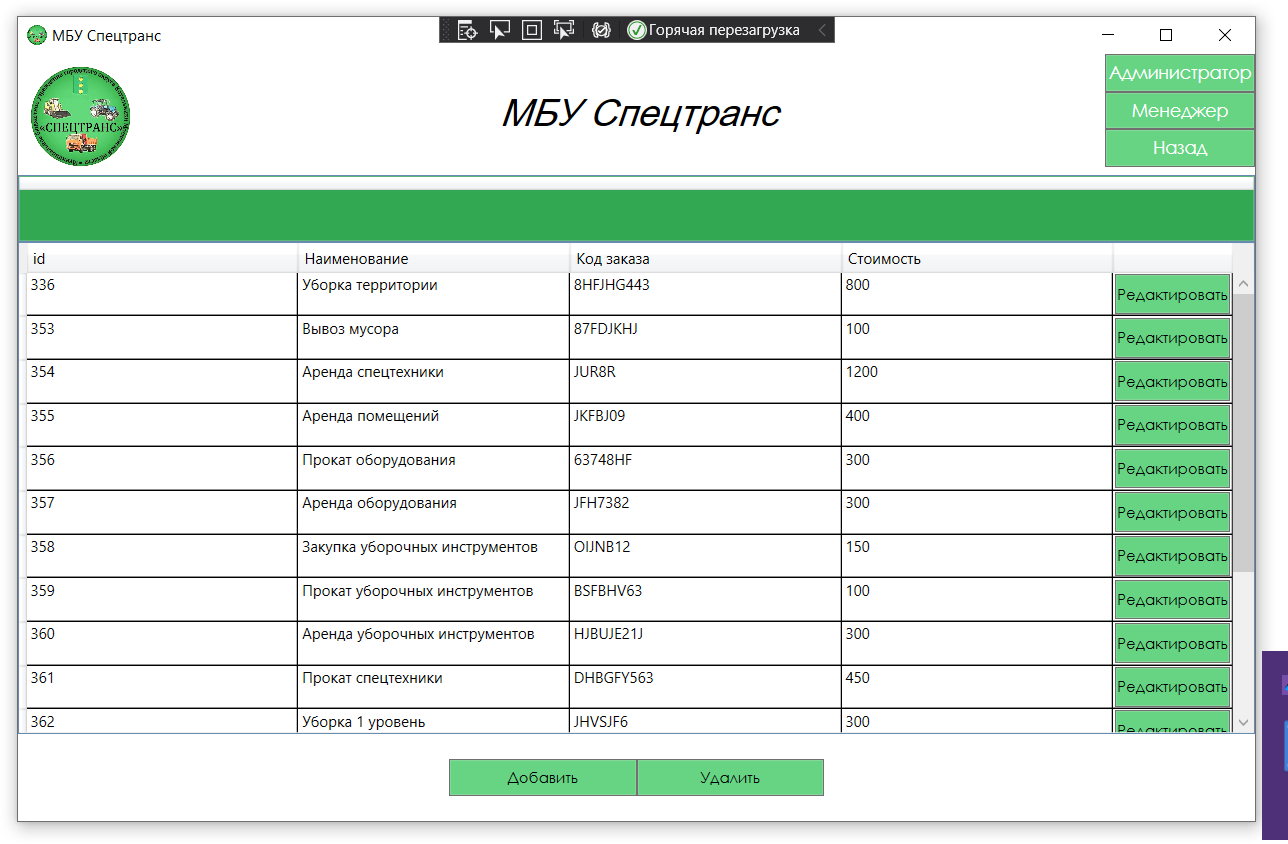


Рисунок 14. Вывод всех доступных услуг при помощи DataGrid

Для работы с данными также существует отдельная страница, в которой содержатся необходимые для заполнения формы. Записи оттуда добавляются в основную таблицу, где доступны для чтения и редактирования. На странице помимо текстовых контейнеров есть также выпадающий список, именуемый ComboBox и кнопка для сохранения данных. Страница для редактирования и добавления данных представлена на рисунке 15.

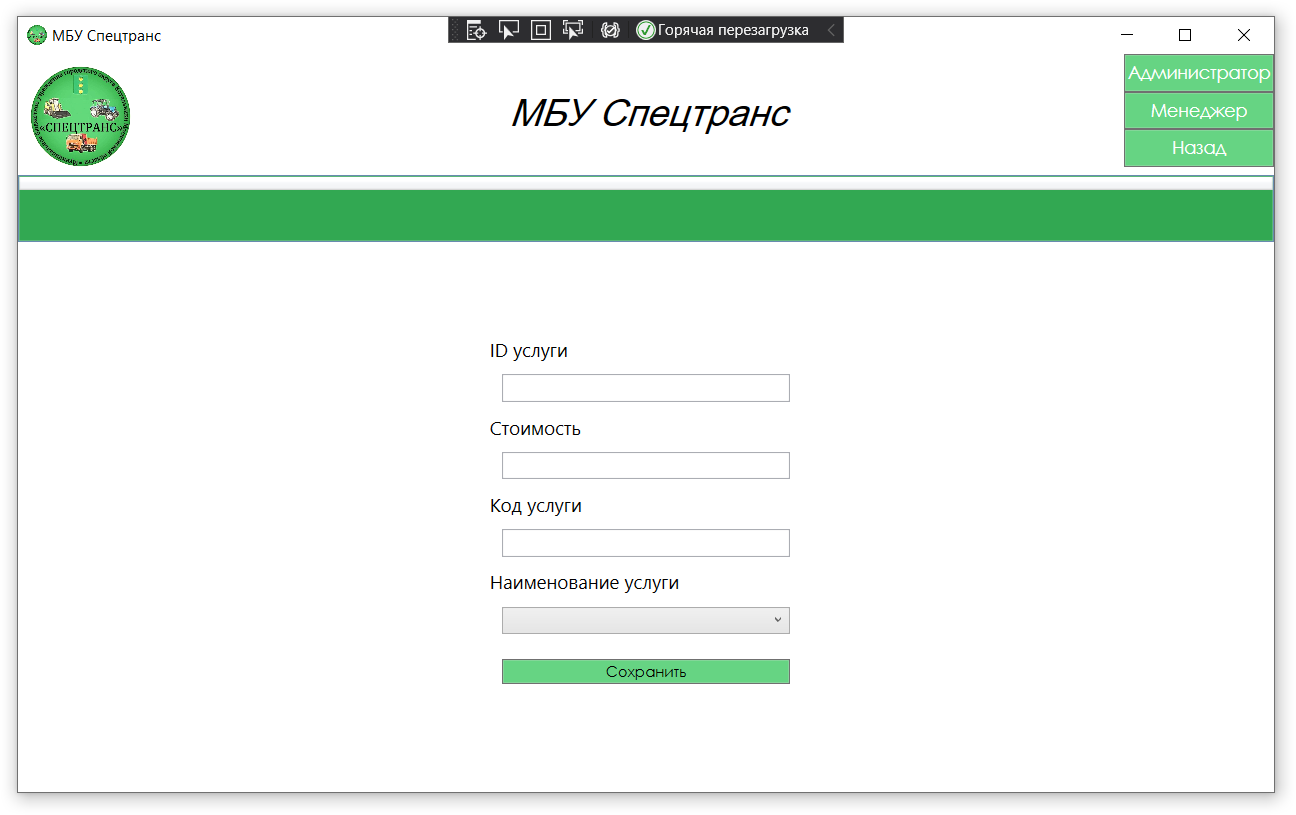


Рисунок 15. Страница для редактирования и добавления данных

Первым делом пользователь вводит вручную ID услуги, стоимость в рублях и ее код, состоящий из определенного набора цифр и букв. В выпадающем списке находятся доступные услуги. Код страницы представлен ниже:

<Page x:Class=”spec.Pages.EditDataPage”

xmlns=”http://schemas.microsoft.com/winfx/2006/xaml/presentation”

xmlns:x=”http://schemas.microsoft.com/winfx/2006/xaml”

xmlns:mc=”http://schemas.openxmlformats.org/markup-compatibility/2006”

xmlns:d=”http://schemas.microsoft.com/expression/blend/2008”

xmlns:local=”clr-namespace:spec.Pages”

mc:Ignorable=”d”

d:DesignHeight=”450” d:DesignWidth=”800”

Title=”EditDataPage”>

<Grid Background=”White”>

<StackPanel VerticalAlignment=”Center”

HorizontalAlignment=”Center”

Width=”250”>

<TextBlock Text=”ID услуги”

FontSize=”15”

/>

<TextBox Name=”IdTbx”

FontSize=”15”

Margin=”10”/>

<TextBlock Text=”Стоимость”

FontSize=”15”/>

<TextBox Name=”Price”

FontSize=”15”

Margin=”10”/>

<TextBlock Text=”Код услуги”

FontSize=”15”

/>

<TextBox Name=”CodeUsluga”

FontSize=”15”

Margin=”10”/>

<TextBlock Text=”Наименование услуги”

FontSize=”15”

/>

<ComboBox x:Name=”CmbTitle”

DisplayMemberPath=”ID”

Margin=”10”/>

<Button x:Name=”SaveBtn”

Height=”20”

Margin=”10”

Content=”Сохранить”

Click=”SaveBtn\_Click”/>

</StackPanel>

</Grid>

</Page>

## *Добавление данных*

После записи нужных данных в форму, пользователь нажимает кнопку сохранить, чтобы данные записались в базу. В этом случае появляется окно о том, что данные внесены успешно (рисунок 16). На рисунке 17 видно, как введенные пользователем данные отображаются в таблице.

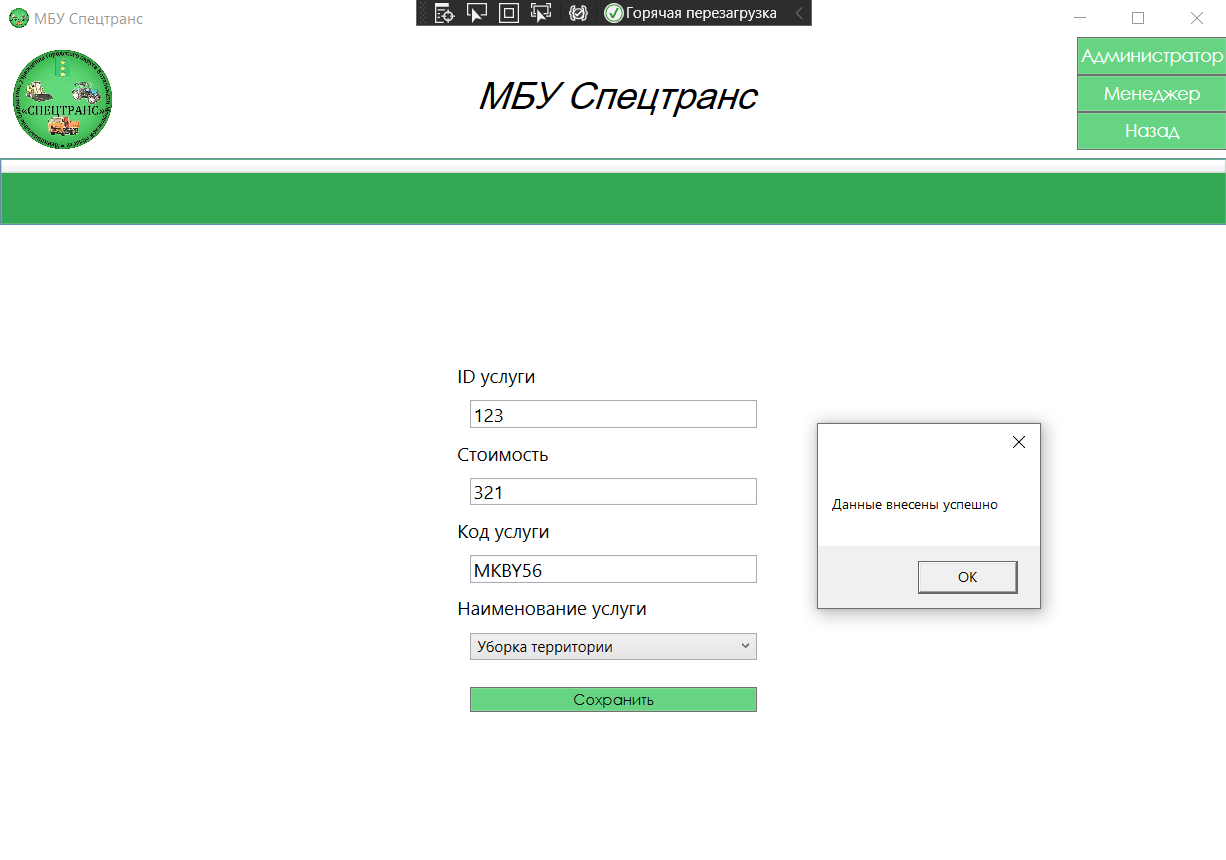


Рисунок 16. Сообщение об успешном вводе данных

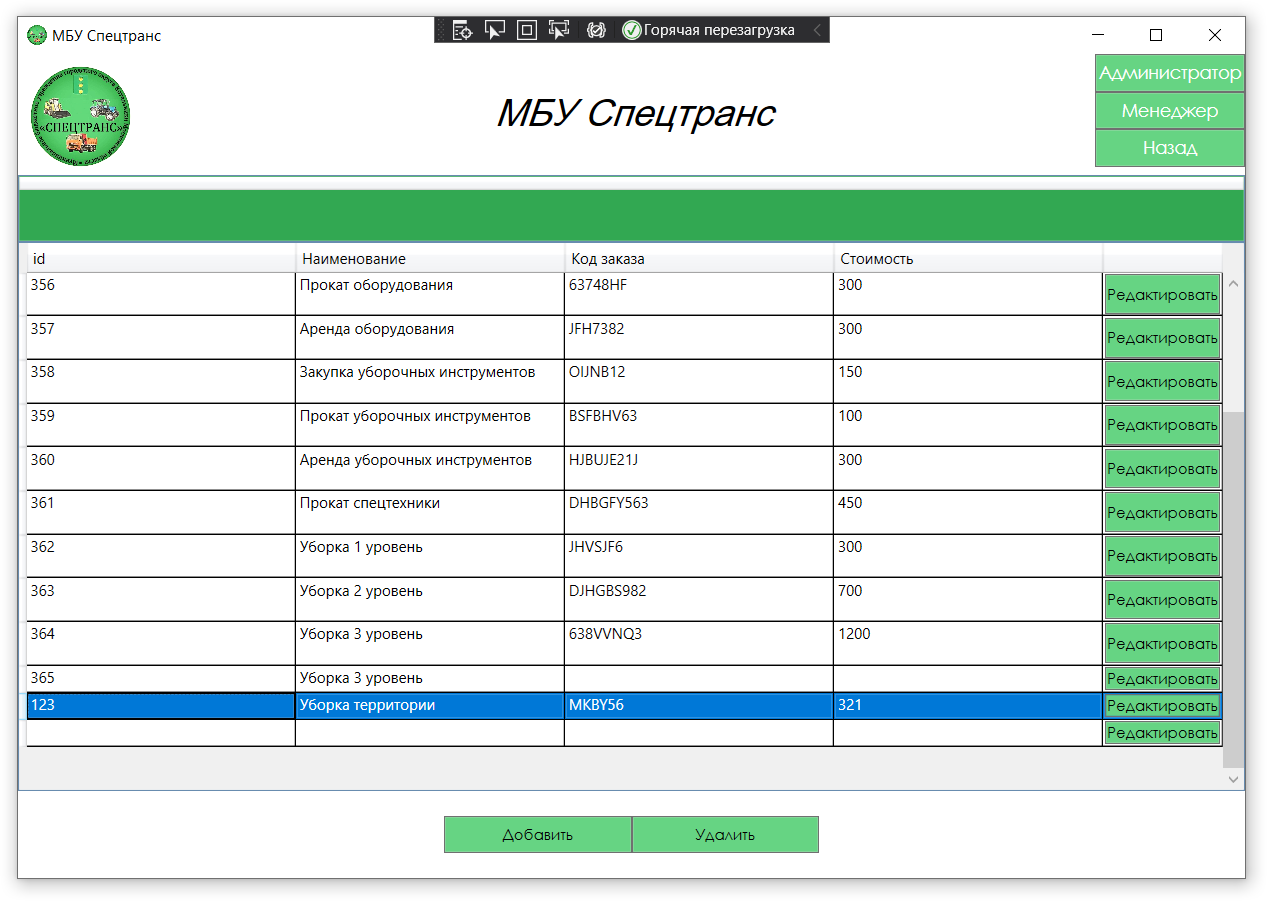


Рисунок 17. Введенные пользователем данные в таблице

Код добавления данных в таблицу:

**private** **void** **SaveBtn\_Click**(**object** sender, RoutedEventArgs e)

{

**try**

{

uslugi usl = **new** uslugi();

**int** a = Int32.Parse(Price.Text);

**int** b = Int32.Parse(IdTbx.Text);

usl.ID =b;

usl.price = a;

usl.code = CodeUsluga.Text;

//данные из комбаря

**var** CurrentUsluga = CmbTitle.SelectedItem **as** uslugi;

usl.title = CurrentUsluga.title;

AppData.db.uslugi.Add(usl);

AppData.db.SaveChanges();

MessageBox.Show("Данные внесены успешно");

NavigationService.GoBack();

}

**catch** (Exception ex)

{

MessageBox.Show("Не удалось добавить данные", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

## *Удаление данных*

Также необходимым действием при работе с базами данных является удаление данных. Это может быть полезно, если данные внесены по ошибке или больше не являются актуальными. Для этих целей на странице с таблицей есть кнопка «Удалить». Перед удалением появляется окно с вопросом пользователю, действительно ли он желает удалить данные (рисунок 18). В случае отрицательного ответа окно просто исчезает, а в случае положительного строчка удаляется и появляется еще одно сообщение об успешном удалении данных (рисунок 19).

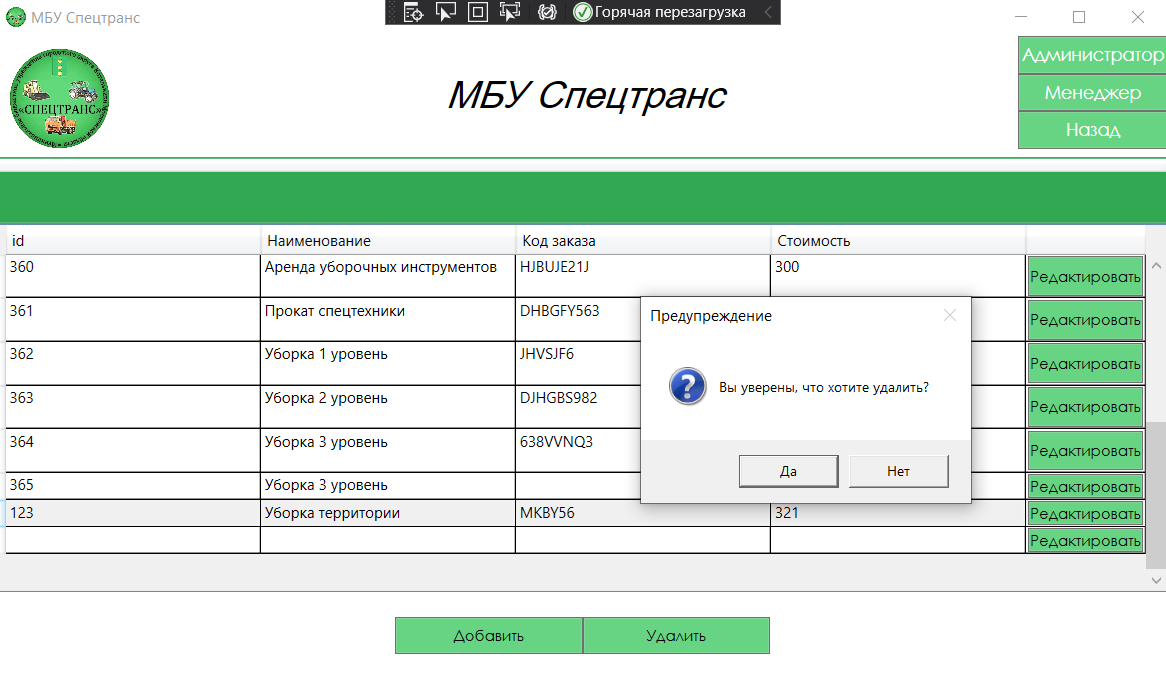


Рисунок 18. Окно с вопросом, хочет ли пользователь удалить данные

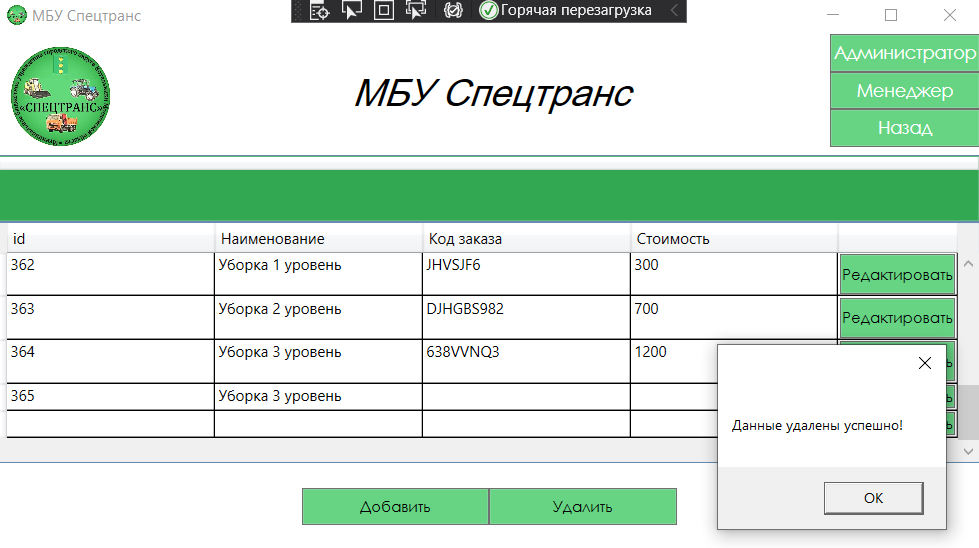


Рисунок 19. Сообщение об успешном удалении данных

Код удаления данных:

**private** **void** **BtnDelete\_Click**(**object** sender, RoutedEventArgs e)

{

**if** (MessageBox.Show(“Вы уверены, что хотите удалить?”, “Предупреждение”, MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

**var** CurrentUsl = DGridUslugi.SelectedItem **as** uslugi;

AppData.db.uslugi.Remove(CurrentUsl);

AppData.db.SaveChanges();

DgridUslugi.ItemsSource = AppData.db.uslugi.ToList();

MessageBox.Show(«Данные удалены успешно!»);

}

}

## *Изменение данных*

Также существует функция изменения данных. Она отчасти похожа с функцией добавления, только от пользователя требуется просто поменять уже имеющиеся в базе данные. Если пользователь захочет отредактировать данные, ему нужно нажать на кнопку «Редактировать», находящуюся на каждой строчке (рисунок 20). После этого откроется окно добавления с уже вписанными данными. Там пользователь изменяет то, что ему нужно, в данном случае, будет изменена цена услуги.

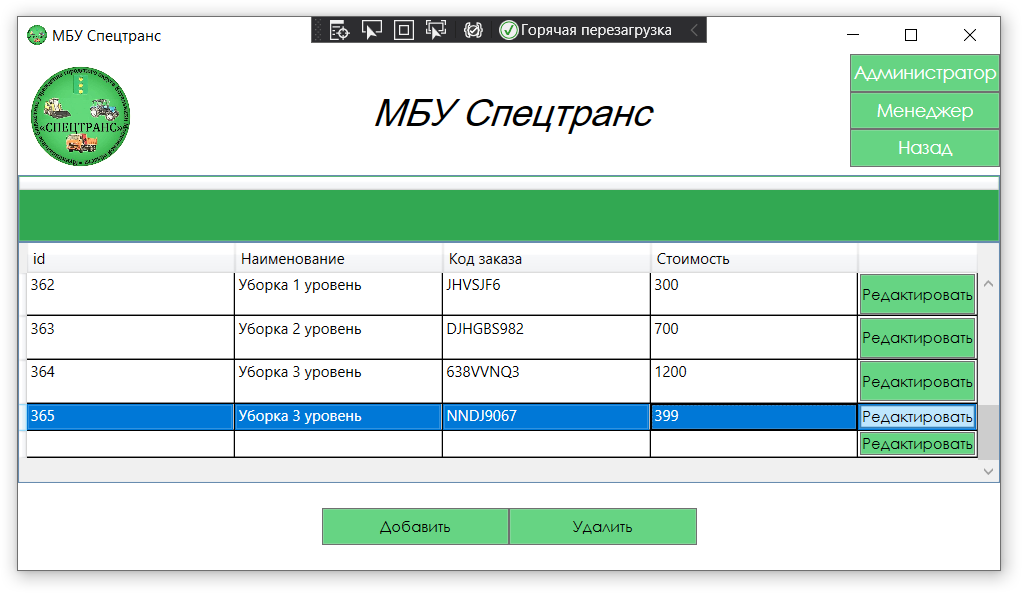


Рисунок 20. Кнопка редактирования в таблице

После добавления данных остается только нажать на кнопку сохранения и появится сообщение об успешном добавлении данных (рисунок 21). В таблице отобразится измененная цена услуги (рисунок 22).

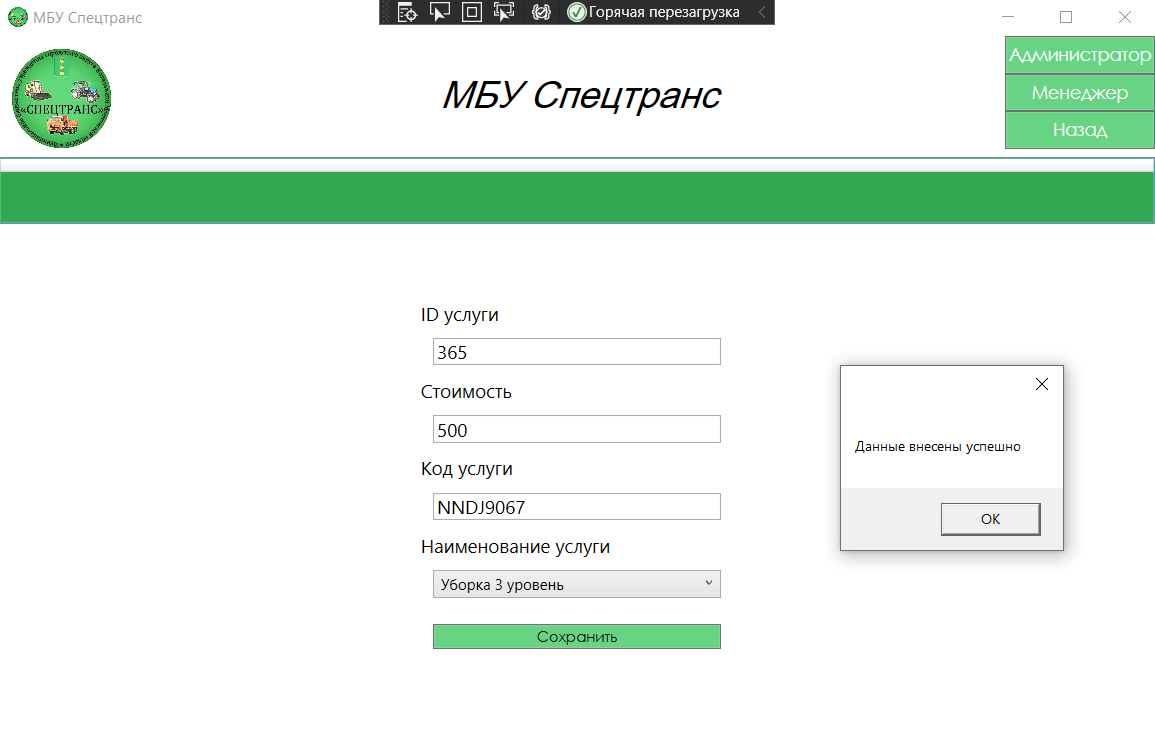


Рисунок 21. Сообщение об успешном изменении данных

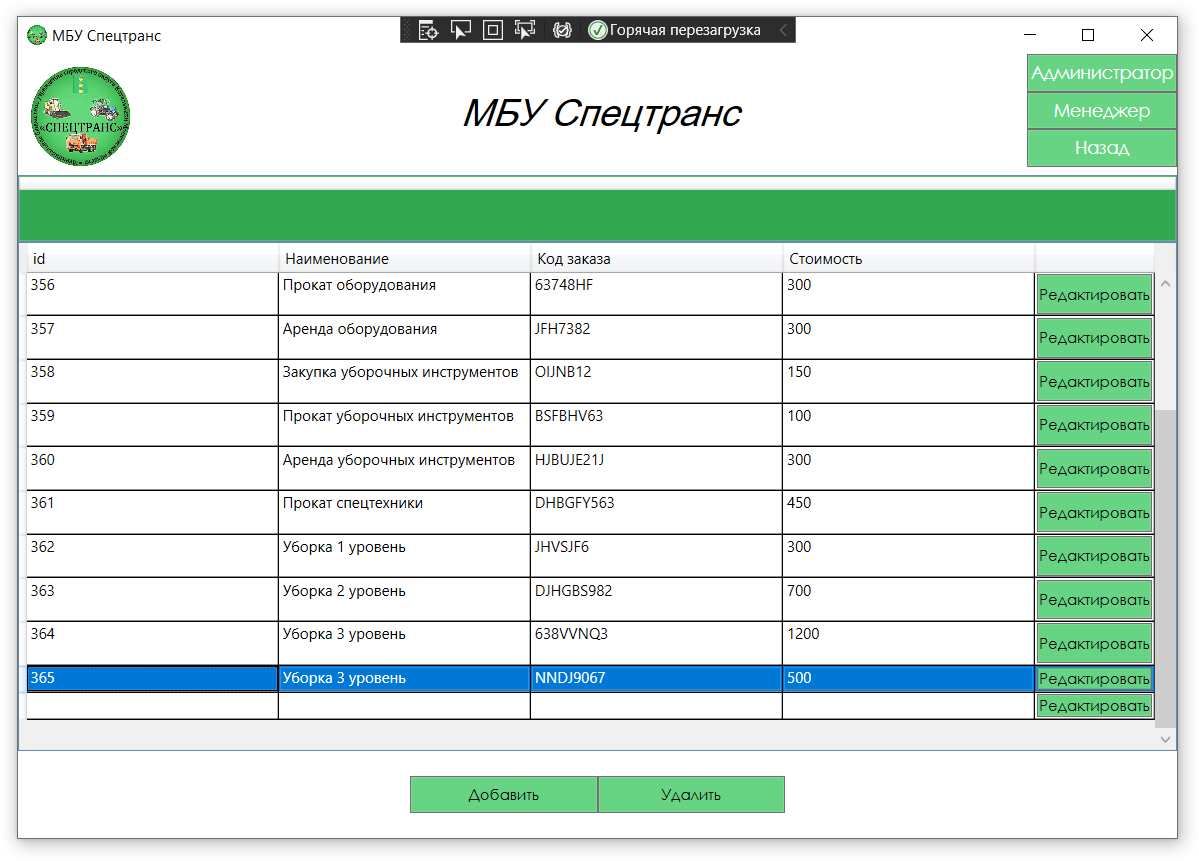


Рисунок 22 – измененные данные в таблице

Вот мы и закончили создание нашего приложения WPF. Во время разработки этого приложения, мы научились работать с базой данных, создавать переход между окнами, добавлять новых пользователей в базу данных и т.д. Разработка данного приложения нужна была для закрепления навыков создания подобных приложений и работы с базой данных.

# **Выводы**

По окончании производственной практической подготовки были решены следующие задачи:

1. Знакомство с литературой
2. Реализация приложения на платформе WPF

В ходе этой практической подготовки были сформированы следующие профессиональные компетенции:

ПК 1.1 Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.

ПК 1.2 Разрабатывать программные модули в соответствии с техническим заданием.

ПК 1.3 Выполнять отладку программных модулей с использованием специализированных программных средств

ПК 1.4 Выполнять тестирование программных модулей

ПК 1.5 Осуществлять рефакторинг и оптимизацию программного кода

ПК 1.6 Разрабатывать модули программного обеспечения для мобильных платформ.

ПК 2.1. Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент.

ПК 2.2. Выполнять интеграцию модулей в программное обеспечение.

ПК 2.3. Выполнять отладку программного модуля с использованием специализированных программных средств.

ПК 2.4. Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения.

ПК 2.5. Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования.

# **Список литературы**

1. Карпова Т.С. Базы данных: модели, разработка, реализация. – СПб.: Питер, 2015.
2. Стружкин Н. П. Базы данных. Проектирование. Учебник / Н.П. Стружкин, В.В. Годин. – М.: Юрайт, 2016.
3. Дейт, К. Дж. SQL и реляционная теория. Как грамотно писать код на SQL / К.Дж. Дейт. - М.: Символ-плюс, 2016.
4. Хабибулин И.Ш. Программирование на языке высокого уровня. C/C++. – СПб.: БХВ– Петербург, 2017.
5. Огнева М.В., Кудрина Е.В. O38 Структуры данных и алгоритмы: программирование на языке C#: Учеб, пособие часть 1. - Саратов: ООО Издательский Центр «Наука», 2017.
6. Ликнесс Дж. Приложения для Windows 8 на C# и XAML; Питер - Москва, 2013.
7. Прайс, Джейсон Oracle Database 11g: SQL. Операторы SQL и программы PL/SQL / Джейсон Прайс. - М.: ЛОРИ, 2018.
8. Джепикс Филипп, Троелсен Эндрю. Язык программирования C# 7 и платформы .NET и .NET Core / Джепикс Филипп, Троелсен Эндрю – М.; Вильямс, 2018.
9. [Адам Натан](https://monster-book.com/adam-natan). WPF 4 Подробное руководство. – СПб.: Питер, 2018.