Федеральное государственное образовательное бюджетное

учреждение высшего образования

**«Финансовый университет при Правительстве Российской Федерации»**

**(Финансовый университет)**

Колледж информатики и программирования

Специальность 09.02.07 «Информационные системы и программирование»

**ОТЧЕТ**

**ПО УЧЕБНОЙ ПРАКТИКЕ**

Профессиональный модуль ПМ.02 Осуществление интеграции программных модулей

*(наименование профессионального модуля)*

Выполнил:

обучающийся учебной группы № 519

Дерябин Г.А.

*(И.О. Фамилия)*

Проверил:

руководитель практики от колледжа:

Т.Г. Аксёнова

*(И.О. Фамилия)*

М.В. Морозова

*(И.О. Фамилия)*

О.Г. Пташкин

*(И.О. Фамилия)*

И.В. Сибирев

*(И.О. Фамилия)*

**Москва**

**20­­­­23**

**Перечень заданий/работ, выполненных в ходе учебной практики**

Специальность 09.02.07 «Информационные системы и программирование»

*(наименование специальности)*

Профессиональный модуль ПМ.02 Осуществление интеграции программных модулей

*(наименование профессионального модуля)*

|  |  |  |
| --- | --- | --- |
| **№ п/п** | **Темы учебной практики** | **Выполненные задания/работы** |
| 1 | Тема 1. Использование выбранной системы контроля версий | Организация работы с проектом с использованием системы контроля версий Git. Создание и клонирование репозитория Git. Фиксация и извлечение изменений в проекте, отправка коммитов, работа с ветвями |
| 2 | Тема 2. Использование методов для получения кода с заданной функциональностью и степенью качества | Реализация методов для получения кода с заданной функциональностью и степенью качества |
| 3 | Тема 3. Интеграция модулей в программное обеспечение | Осуществление интеграции программных модулей в программное обеспечение |
| 4 | Тема 4. Отладка программных модулей | Выполнение отладки программных модулей с использованием специализированных программных средств |
| 5 | Тема 5. Разработка тестовых наборов и тестовых сценариев для программного обеспечения | Осуществление разработки тестовых наборов и тестовых сценариев для программного обеспечения |
| 6 | Тема 6. Инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования | Проведение инспектирования компонент программного обеспечения на предмет соответствия стандартам кодирования |
| 7 | Тема 7. Разработка технического задания на веб-приложение в соответствии с требованиями заказчика | Разработка и оформление технического задания на веб-приложение в соответствии с требованиями заказчика |
| 8 | Тема 8. Разработка веб-приложения в соответствии с техническим заданием | Разработка веб-приложения в соответствии с техническим заданием |
| 9 | Тема 9. Разработка интерфейса пользователя веб-приложения в соответствии с техническим заданием | Разработка интерфейса пользователя веб-приложения в соответствии с техническим заданием |
| 10 | Тема 10. Осуществление технического сопровождения и восстановления веб-приложения в соответствии с техническим заданием | Осуществление технического сопровождения и восстановления веб-приложения в соответствии с техническим заданием |
| 11 | Тема 11. Тестирование разработанного веб-приложения | Осуществление тестирования разработанного веб-приложения |
| 12 | Тема 12. Размещение веб-приложения в сети в соответствии с техническим заданием | Размещение веб-приложения в сети в соответствии с техническим заданием |

СОДЕРЖАНИЕ

[**ОТЧЕТ О ВЫПОЛНЕНИИ ЗАДАНИЯ № 1** 5](#_Toc129716449)

[**Тема:** ТЦ, павильоны 5](#_Toc129716450)

[**ОТЧЕТ О ВЫПОЛНЕНИИ ЗАДАНИЯ № 2** 30](#_Toc129716451)

[**Тема:** Товары автосервиса 30](#_Toc129716452)

[**ОТЧЕТ О ВЫПОЛНЕНИИ ЗАДАНИЯ № 3** 38](#_Toc129716453)

[**Тема:** Госавтоинспекция 38](#_Toc129716454)

[**ОТЧЕТ О ВЫПОЛНЕНИИ ЗАДАНИЯ № 4** 55](#_Toc129716455)

[**Тема:** Импорт 07.02 от М.В. Морозовой 55](#_Toc129716456)

[**ОТЧЕТ О ВЫПОЛНЕНИИ ЗАДАНИЯ № 5** 60](#_Toc129716457)

[**Тема:** 14.02 задание от М.В. Морозовой 60](#_Toc129716458)

[**ОТЧЕТ О ВЫПОЛНЕНИИ ЗАДАНИЯ № 6** 68](#_Toc129716459)

[**Тема:** Тестирование от Аксёновой Т.Г. от 20.02.2023 68](#_Toc129716460)

[**ОТЧЕТ О ВЫПОЛНЕНИИ ЗАДАНИЯ № 7** 75](#_Toc129716468)

**Тема**: Задания по веб от Пташкина О.Г.………………………………….…………………..75

[**ОТЧЕТ О ВЫПОЛНЕНИИ ЗАДАНИЯ № 8** 78](#_Toc129716469)

[**Тема:** Конспект по Javascript 78](#_Toc129716470)

Федеральное государственное образовательное бюджетное

учреждение высшего образования

**«Финансовый университет при Правительстве Российской Федерации»**

**(Финансовый университет)**

Колледж информатики и программирования

# **ОТЧЕТ О ВЫПОЛНЕНИИ ЗАДАНИЯ № 1**

# **Тема:** ТЦ, павильоны

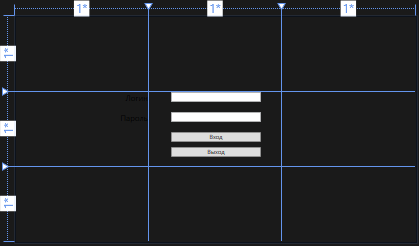
Студент: Дерябин Г.А.

Группа: 4ИСИП-519

Преподаватель: Аксенова Т.Г., Морозова М.В.

Дата: 14.03.2023

**Окно (Autoregistration.xaml)**

****

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="1\*"/>

<RowDefinition Height="1\*"/>

<RowDefinition Height="1\*"/>

</Grid.RowDefinitions>

<TextBox x:Name="Log" Height="20" Width="180" Grid.Row="1" Grid.Column="1" HorizontalAlignment="Center" VerticalAlignment="Top"/>

<TextBlock x:Name="TextLog" Text="Логин:" Grid.Row="1" Grid.Column="0" HorizontalAlignment="Right" VerticalAlignment="Top" FontSize="16" Height="21" Width="48" />

<PasswordBox x:Name="Pass" Height="20" Width="180" Grid.Row="1" Grid.Column="1" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0,40,0,0"/>

<TextBlock x:Name="TextPass" Text="Пароль:" HorizontalAlignment="Right" VerticalAlignment="Top" FontSize="16" Grid.Row="1" Grid.Column="0" Height="21" Width="58" Margin="0,40,0,0"/>

<Button x:Name="Reg" Content="Вход" Grid.Row="1" Grid.Column="1" Height="20" VerticalAlignment="Top" Margin="0,80,0,0" Width="180" Click="Reg\_Click" />

<Button x:Name="Back" Content="Выход" Grid.Row="1" Grid.Column="1" Height="20" VerticalAlignment="Top" Margin="0,110,0,0" Width="180" Click="Back\_Click"/>

<Grid x:Name="Grid" Visibility="Hidden" Grid.Row="1" Grid.Column="1" Background="White">

<TextBlock Text="Капча" HorizontalAlignment="Center" VerticalAlignment="Top" FontSize="16"/>

<TextBlock x:Name="Kapcha" Height="40" Width="200" VerticalAlignment="Top" Margin="25" FontSize="20" TextAlignment="Center"/>

<TextBox x:Name="UserText" Height="20" Width="200" VerticalAlignment="Bottom" Margin="0,0,0,50"/>

<Button x:Name="but1" Height="20" Width="200" VerticalAlignment="Bottom" Margin="10" Content="Проверить" Click="but1\_Click"/>

</Grid>

</Grid>

**Окно (Autoregistration.xaml.cs)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using WpfApp2;

using WpfApp2.Pages;

namespace sesion.Pages

{

/// <summary>

/// Логика взаимодействия для Autoregistration.xaml

/// </summary>

public partial class Autoregistration : Page

{

public Autoregistration()

{

InitializeComponent();

}

public int a = 0;

public string s = "";

private void Back\_Click(object sender, RoutedEventArgs e)

{

}

private void Reg\_Click(object sender, RoutedEventArgs e)

{

Auth(Log.Text, Pass.Password);

//if(a!=2)

//{

//}

//else

//{

// Grid.Visibility = Visibility;

// TextPass.Visibility = Visibility.Hidden;

// TextLog.Visibility = Visibility.Hidden;

// Random r = new Random();

// for (int i = 0; i < 5; i++)

// s += (char)(r.Next(1072, 1104));

// Kapcha.Text = s;

//}

}

private void but1\_Click(object sender, RoutedEventArgs e)

{

if(UserText.Text == s)

{

Grid.Visibility = Visibility.Hidden;

TextPass.Visibility = Visibility.Visible;

TextLog.Visibility = Visibility.Visible;

a = 0;

s = "";

Kapcha.Text = "";

}

else

{

s = "";

Random r = new Random();

for (int i = 0; i < 5; i++)

s += (char)(r.Next(1072, 1104));

Kapcha.Text = s;

}

}

public bool Auth(string login, string password)

{

if(string.IsNullOrEmpty(login)|| string.IsNullOrEmpty(password))

{

MessageBox.Show("gg");

return false;

}

using (var db = new Entities())

{

var asd = db.Sotrudniki.Where(i => i.Login == Log.Text && i.Password == Pass.Password).FirstOrDefault();

if (asd != null)

{

switch (asd.Role)

{

case "Менеджер С":

NavigationService.Navigate(new Pages.Menedjer\_C());

break;

case "Администратор":

NavigationService.Navigate(new Administrator());

break;

case "Менеджер А":

NavigationService.Navigate(new Menedjer\_A());

break;

}

}

else

{

MessageBox.Show("Проверьте введенные данные");

a++;

}

return true;

}

}

}

}

**Окно (Administrator.xaml)**

**Изображение выглядит как текст

Автоматически созданное описание**

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="2\*"/>

<RowDefinition Height="10\*"/>

<RowDefinition Height="1\*"/>

</Grid.RowDefinitions>

<DataGrid x:Name="grid\_Sotrudniki" AutoGenerateColumns="False" IsReadOnly="True" Grid.Row="1">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding FIO}" Header="ФИО" Width="auto"/>

<DataGridTextColumn Binding="{Binding Login}" Header="Логин" Width="auto"/>

<DataGridTextColumn Binding="{Binding Password}" Header="Пароль" Width="auto"/>

<DataGridTextColumn Binding="{Binding Role}" Header="Роль" Width="auto"/>

<DataGridTextColumn Binding="{Binding nomer\_telefona}" Header="номер телефона" Width="auto"/>

</DataGrid.Columns>

</DataGrid>

<TextBlock Text="ФИО:" Height="20" Width="150" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="20,0,0,0"/>

<TextBox SelectionChanged="Gorod\_SelectionChanged" x:Name="Name" Height="20" Width="150" HorizontalAlignment="Left" VerticalAlignment="Center" Margin="20 0 0 0"/>

<Button x:Name="Delete" Content="Удалить" HorizontalAlignment="Left" VerticalAlignment="Center" Height="20" Width="80" Grid.Row="2" Click="Delete\_Click"/>

<Button x:Name="Add" Content="Добавить" HorizontalAlignment="Left" VerticalAlignment="Center" Height="20" Width="80" Grid.Row="2" Margin="110,0,0,0" Click="Add\_Click"/>

<Button x:Name="Redact" Content="Редактировать" HorizontalAlignment="Left" VerticalAlignment="Center" Height="20" Width="80" Grid.Row="2" Margin="210,0,0,0" Click="Redact\_Click"/>

</Grid>

**Окно (Administrator.xaml.cs)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace WpfApp2.Pages

{

/// <summary>

/// Логика взаимодействия для Administrator.xaml

/// </summary>

public partial class Administrator : Page

{

public Administrator()

{

InitializeComponent();

grid\_Sotrudniki.ItemsSource = DataBaseConnection.entites.Sotrudniki.ToList();

}

private void Gorod\_SelectionChanged(object sender, RoutedEventArgs e)

{

if(Name.Text.Length> 0)

{

var abc = DataBaseConnection.entites.Sotrudniki.Where(i=>i.FIO.Contains(Name.Text)).ToList();

grid\_Sotrudniki.ItemsSource = abc;

}

}

private void Delete\_Click(object sender, RoutedEventArgs e)

{

var delete0 = grid\_Sotrudniki.SelectedItem as Sotrudniki;

if (delete0 != null)

{

try

{

DataBaseConnection.entites.Sotrudniki.Remove(delete0);

DataBaseConnection.entites.SaveChanges();

MessageBox.Show("успешно");

grid\_Sotrudniki.ItemsSource = DataBaseConnection.entites.Sotrudniki.ToList();

}

catch

{

MessageBox.Show("gg");

}

}

}

private void Add\_Click(object sender, RoutedEventArgs e)

{

NavigationService.Navigate(new Pages.Add\_Redac\_Sotrudniki(null));

}

private void Redact\_Click(object sender, RoutedEventArgs e)

{

NavigationService.Navigate(new Pages.Add\_Redac\_Sotrudniki(grid\_Sotrudniki.SelectedItem as Sotrudniki));

}

}

}

**Окно (Menedjer\_A.xaml)**

**Изображение выглядит как текст

Автоматически созданное описание**

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="2\*"/>

<RowDefinition Height="10\*"/>

<RowDefinition Height="1\*"/>

</Grid.RowDefinitions>

<DataGrid x:Name="Grid\_Arendator" IsReadOnly="True" AutoGenerateColumns="False" Grid.Row="1">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding nazvanie}" Header="Название" Width="auto"/>

<DataGridTextColumn Binding="{Binding nomer}" Header="Номер" Width="auto"/>

<DataGridTextColumn Binding="{Binding adres}" Header="адрес" Width="auto"/>

</DataGrid.Columns>

</DataGrid>

<TextBlock Text="Название:" Height="20" Width="150" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="20,0,0,0"/>

<TextBox SelectionChanged="Title\_SelectionChanged" x:Name="Title" Height="20" Width="150" HorizontalAlignment="Left" VerticalAlignment="Center" Margin="20 0 0 0"/>

<Button x:Name="Delete" Content="Удалить" HorizontalAlignment="Left" VerticalAlignment="Center" Height="20" Width="80" Grid.Row="2" Click="Delete\_Click"/>

<Button x:Name="Add" Content="Добавить" HorizontalAlignment="Left" VerticalAlignment="Center" Height="20" Width="80" Grid.Row="2" Margin="110,0,0,0" Click="Add\_Click"/>

<Button x:Name="Redact" Content="Редактировать" HorizontalAlignment="Left" VerticalAlignment="Center" Height="20" Width="80" Grid.Row="2" Margin="210,0,0,0" Click="Redact\_Click"/>

</Grid>

**Окно (Menedjer\_A.xaml.cs)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace WpfApp2.Pages

{

/// <summary>

/// Логика взаимодействия для Menedjer\_A.xaml

/// </summary>

public partial class Menedjer\_A : Page

{

public Menedjer\_A()

{

InitializeComponent();

Grid\_Arendator.ItemsSource = DataBaseConnection.entites.Arendator.ToList();

}

private void Title\_SelectionChanged(object sender, RoutedEventArgs e)

{

if(Title.Text.Length >0)

{

Grid\_Arendator.ItemsSource=DataBaseConnection.entites.Arendator.Where(i=>i.nazvanie.Contains(Title.Text)).ToList();

}

}

private void Delete\_Click(object sender, RoutedEventArgs e)

{

var delete = Grid\_Arendator.SelectedItem as Arendator;

if(delete != null)

{

try

{

DataBaseConnection.entites.Arendator.Remove(delete);

DataBaseConnection.entites.SaveChanges();

MessageBox.Show("Успешно");

Grid\_Arendator.ItemsSource = DataBaseConnection.entites.Arendator.ToList();

}

catch

{

MessageBox.Show("Ошибка");

}

}

}

private void Add\_Click(object sender, RoutedEventArgs e)

{

NavigationService.Navigate(new Pages.Add\_Redac\_Arendator(null));

}

private void Redact\_Click(object sender, RoutedEventArgs e)

{

NavigationService.Navigate(new Pages.Add\_Redac\_Arendator(Grid\_Arendator.SelectedItem as Arendator));

}

}

}

**Окно (Menedjer\_C.xaml)**

**Изображение выглядит как текст

Автоматически созданное описание**

<Grid Background="White">

<TabControl Margin="0,0,0,50" Name="Tab">

<TabItem Header="Торговые центры">

<DataGrid x:Name="grid\_TC" AutoGenerateColumns="False" IsReadOnly="True">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding Nazvanie}" Header="Название" Width="auto"/>

<DataGridTextColumn Binding="{Binding Status}" Header="Статус" Width="auto"/>

<DataGridTextColumn Binding="{Binding Kol\_Pavilionov}" Header="Количество павильонов" Width="auto"/>

<DataGridTextColumn Binding="{Binding Gorod}" Header="Город" Width="auto"/>

<DataGridTextColumn Binding="{Binding Stoimost}" Header="Стоимость" Width="auto"/>

<DataGridTextColumn Binding="{Binding Etajnost}" Header="Этажность" Width="auto"/>

<DataGridTextColumn Binding="{Binding Kof\_Dobav\_Stoimosti}" Header="Коф. добавочной стоимости" Width="auto"/>

</DataGrid.Columns>

</DataGrid>

</TabItem>

<TabItem Header="Павильоны">

<DataGrid x:Name="Grid\_pavilion" IsReadOnly="True" AutoGenerateColumns="False">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding TC1.Status}" Header="Статус ТЦ" Width="auto"/>

<DataGridTextColumn Binding="{Binding nazvanie\_Tc}" Header="Название ТЦ" Width="auto"/>

<DataGridTextColumn Binding="{Binding etaj}" Header="Этаж" Width="auto"/>

<DataGridTextColumn Binding="{Binding nomer\_paviliona}" Header="Номер павильона" Width="auto"/>

<DataGridTextColumn Binding="{Binding plochad}" Header="Площадь" Width="auto"/>

<DataGridTextColumn Binding="{Binding status}" Header="Статус" Width="auto"/>

<DataGridTextColumn Binding="{Binding kof\_dobav\_stoimosti}" Header="Коф.добавочной стоимости" Width="auto"/>

<DataGridTextColumn Binding="{Binding stoimost\_za\_kv}" Header="Стоимость за кв.м." Width="auto"/>

</DataGrid.Columns>

</DataGrid>

</TabItem>

</TabControl>

<Button x:Name="Add" Content="Добавить" HorizontalAlignment="Left" VerticalAlignment="Bottom" Height="20" Width="80" Margin="10,0,0,10" Click="Add\_Click"/>

<Button x:Name="Redac" Content="Реактировать" HorizontalAlignment="Left" VerticalAlignment="Bottom" Height="20" Width="80" Margin="110,0,0,10" Click="Redac\_Click"/>

<Button x:Name="Delete" Content="Удалить" HorizontalAlignment="Left" VerticalAlignment="Bottom" Height="20" Width="80" Margin="210,0,0,10" Click="Delete\_Click"/>

<Button x:Name="Requst" Content="Фильтры" HorizontalAlignment="Left" VerticalAlignment="Bottom" Height="20" Width="80" Margin="310,0,0,10" Click="Requst\_Click"/>

</Grid>

**Окно (Menedjer\_C.xaml.cs)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using WpfApp2;

namespace sesion.Pages

{

/// <summary>

/// Логика взаимодействия для Menedjer\_C.xaml

/// </summary>

public partial class Menedjer\_C : Page

{

public Menedjer\_C()

{

InitializeComponent();

grid\_TC.ItemsSource = DataBaseConnection.entites.TC1.ToList();

Grid\_pavilion.ItemsSource = DataBaseConnection.entites.pavilion.ToList();

}

private void Delete\_Click(object sender, RoutedEventArgs e)

{

switch(Tab.SelectedIndex)

{

case 0:

var delete0 = grid\_TC.SelectedItem as TC1;

if(delete0 != null)

{

DataBaseConnection.entites.TC1.Remove(delete0);

DataBaseConnection.entites.SaveChanges();

grid\_TC.ItemsSource = DataBaseConnection.entites.TC1.ToList();

}

break;

case 1:

var delete1 = Grid\_pavilion.SelectedItem as pavilion;

if(delete1 != null)

{

if (delete1.status == "Забронировано")

{

MessageBox.Show("Невозможно удалить. статус: Забронирован ");

}

else if (delete1.status == "Арендован")

{

MessageBox.Show("Невозможно удалить. статус: Арендован ");

}

else

{

DataBaseConnection.entites.pavilion.Remove(delete1);

DataBaseConnection.entites.SaveChanges();

Grid\_pavilion.ItemsSource = DataBaseConnection.entites.pavilion.ToList();

}

}

break;

}

}

private void Redac\_Click(object sender, RoutedEventArgs e)

{

switch (Tab.SelectedIndex)

{

case 0:

if(grid\_TC.SelectedItem != null)

{

NavigationService.Navigate(new Pages.Add\_Redac\_TC(grid\_TC.SelectedItem as TC1));

}

else

{

MessageBox.Show("Выберети элемент");

}

break;

case 1:

var redac = Grid\_pavilion.SelectedItem as pavilion;

if (redac != null)

{

if (redac.status == "Забронировано")

{

MessageBox.Show("Невозможно удалить. статус: Забронирован ");

}

else if (redac.status == "Арендован")

{

MessageBox.Show("Невозможно удалить. статус: Арендован ");

}

else

{

NavigationService.Navigate(new Pages.Add\_Redac\_Pavilion(Grid\_pavilion.SelectedItem as pavilion));

}

}

else

{

MessageBox.Show("Выберети элемент");

}

break;

}

}

private void Add\_Click(object sender, RoutedEventArgs e)

{

switch (Tab.SelectedIndex)

{

case 0:

NavigationService.Navigate(new Pages.Add\_Redac\_TC(null));

break;

case 1:

NavigationService.Navigate(new Pages.Add\_Redac\_Pavilion(null));

break;

}

}

private void Requst\_Click(object sender, RoutedEventArgs e)

{

switch (Tab.SelectedIndex)

{

case 0:

NavigationService.Navigate(new Pages.Requast\_Tc());

break;

case 1:

NavigationService.Navigate(new Pages.Requast\_Pavilion());

break;

}

}

}

}

**Окно (Add\_Redac\_Arendator.xaml)**

**Изображение выглядит как текст

Автоматически созданное описание**

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="25\*"/>

<RowDefinition Height="3\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="1\*"/>

</Grid.ColumnDefinitions>

<StackPanel VerticalAlignment="Center" Grid.Column="1" Grid.Row="0">

<TextBox Text="{Binding nazvanie}" x:Name="Txt\_0" Height="20" Width="180" Margin="10"/>

<TextBox Text="{Binding nomer}" x:Name="Txt\_1" Height="20" Width="180" Margin="10"/>

<TextBox Text="{Binding adres}" x:Name="Txt\_2" Height="20" Width="180" Margin="10"/>

</StackPanel>

<StackPanel VerticalAlignment="Center" Grid.Column="0" Grid.Row="0">

<TextBlock Text="Название компании:" HorizontalAlignment="Right" Height="20" Margin="10"/>

<TextBlock Text="Адрес:" HorizontalAlignment="Right" Height="20" Margin="10"/>

<TextBlock Text="Номер телефона:" HorizontalAlignment="Right" Height="20" Margin="10"/>

</StackPanel>

<Button x:Name="Save" Content="Сохранить" HorizontalAlignment="Left" Height="20" Width="80" Margin="10,0,0,0" Grid.Column="1" Grid.Row="1" Click="Save\_Click"/>

<Button x:Name="Back" Content="Назад" HorizontalAlignment="Right" Height="20" Width="80" Margin="0,0,10,0" Grid.Column="1" Grid.Row="1" Click="Back\_Click"/>

</Grid>

**Окно (Add\_Redac\_Arendator.xaml.cs)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace WpfApp2.Pages

{

/// <summary>

/// Логика взаимодействия для Add\_Redac\_Arendator.xaml

/// </summary>

public partial class Add\_Redac\_Arendator : Page

{

Arendator \_Arendator = new Arendator();

public Add\_Redac\_Arendator(Arendator selectedArendator)

{

InitializeComponent();

if (selectedArendator != null)

\_Arendator = selectedArendator;

DataContext = \_Arendator;

}

private void Save\_Click(object sender, RoutedEventArgs e)

{

StringBuilder errors = new StringBuilder();

if (string.IsNullOrWhiteSpace(Txt\_0.Text))

errors.AppendLine("Укажите !");

if (string.IsNullOrWhiteSpace(Txt\_1.Text))

errors.AppendLine("Укажите !");

if (string.IsNullOrWhiteSpace(Txt\_2.Text))

errors.AppendLine("Укажите !");

if (errors.Length > 0)

{

MessageBox.Show(errors.ToString());

}

else

{

if (\_Arendator.Arendator\_Id == 0)

DataBaseConnection.entites.Arendator.Add(\_Arendator);

try

{

DataBaseConnection.entites.SaveChanges();

MessageBox.Show("ок");

NavigationService.Navigate(new Pages.Menedjer\_A());

}

catch

{

MessageBox.Show("no ok");

}

}

}

private void Back\_Click(object sender, RoutedEventArgs e)

{

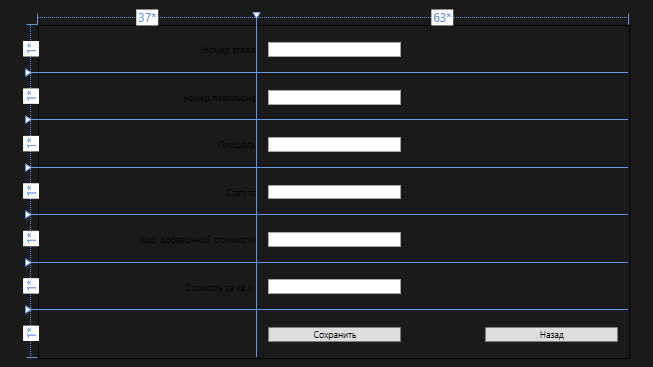
NavigationService.GoBack();

}

}

}

**Окно (Add\_Redac\_Pavilion.xaml)**

****

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="37\*"/>

<ColumnDefinition Width="63\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<TextBlock Text="Номер этажа:" Grid.Row="0" VerticalAlignment="Center" HorizontalAlignment="Right" Grid.Column="0"/>

<TextBox Text="{Binding etaj}" x:Name="number\_etaj" Height="20" Width="180" VerticalAlignment="Center" HorizontalAlignment="Left" Grid.Row="0" Grid.Column="1" Margin="15"/>

<TextBlock Text="номер павильона:" Grid.Row="1" VerticalAlignment="Center" HorizontalAlignment="Right" Grid.Column="0"/>

<TextBox Text="{Binding nomer\_paviliona}" x:Name="Number\_Pavilion" Height="20" Width="180" VerticalAlignment="Center" HorizontalAlignment="Left" Grid.Row="1" Grid.Column="1" Margin="15"/>

<TextBlock Text="Площадь:" Grid.Row="2" VerticalAlignment="Center" HorizontalAlignment="Right" Grid.Column="0"/>

<TextBox Text="{Binding plochad}" x:Name="plochad" Grid.Row="2" Grid.Column="1" Height="20" Width="180" VerticalContentAlignment="Center" HorizontalAlignment="Left" Margin="15"/>

<TextBlock Text="Статутс:" Grid.Row="3" VerticalAlignment="Center" HorizontalAlignment="Right" Grid.Column="0"/>

<TextBox Text="{Binding status}" x:Name="Status" Height="20" Width="180" VerticalAlignment="Center" HorizontalAlignment="Left" Grid.Row="3" Grid.Column="1" Margin="15"/>

<TextBlock Text="Коф. добавочной стоимости:" Grid.Row="4" VerticalAlignment="Center" HorizontalAlignment="Right" Grid.Column="0"/>

<TextBox Text="{Binding kof\_dobav\_stoimosti}" x:Name="kof\_stoimosti" Height="20" Width="180" VerticalAlignment="Center" HorizontalAlignment="Left" Grid.Row="4" Grid.Column="1" Margin="15"/>

<TextBlock Text="Стомость за кв.м.:" Grid.Row="5" VerticalAlignment="Center" HorizontalAlignment="Right" Grid.Column="0"/>

<TextBox Text="{Binding stoimost\_za\_kv}" x:Name="Stoimost\_za\_kv" Height="20" Width="180" VerticalAlignment="Center" HorizontalAlignment="Left" Grid.Row="5" Grid.Column="1" Margin="15"/>

<Button Content="Сохранить" x:Name="Save" Height="20" Width="180" VerticalAlignment="Center" HorizontalAlignment="Left" Grid.Row="8" Grid.Column="1" Margin="15" Click="Save\_Click"/>

<Button Content="Назад" x:Name="Back" Height="20" Width="180" VerticalAlignment="Center" HorizontalAlignment="Right" Grid.Row="8" Grid.Column="1" Margin="15" Click="Back\_Click"/>

</Grid>

**Окно (Add\_Redac\_Pavilion.xaml.cs)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using WpfApp2;

namespace sesion.Pages

{

/// <summary>

/// Логика взаимодействия для Add\_Redac\_Pavilion.xaml

/// </summary>

public partial class Add\_Redac\_Pavilion : Page

{

private pavilion \_currentPavilion = new pavilion();

public Add\_Redac\_Pavilion(pavilion selectedPavilion)

{

InitializeComponent();

if (selectedPavilion != null)

\_currentPavilion = selectedPavilion;

DataContext = \_currentPavilion;

}

private void Save\_Click(object sender, RoutedEventArgs e)

{

if(\_currentPavilion.ID == 0)

DataBaseConnection.entites.pavilion.Add(\_currentPavilion);

try

{

DataBaseConnection.entites.SaveChanges();

MessageBox.Show("ок");

NavigationService.Navigate(new Pages.Menedjer\_C());

}

catch

{

MessageBox.Show("no ok");

}

}

private void Back\_Click(object sender, RoutedEventArgs e)

{

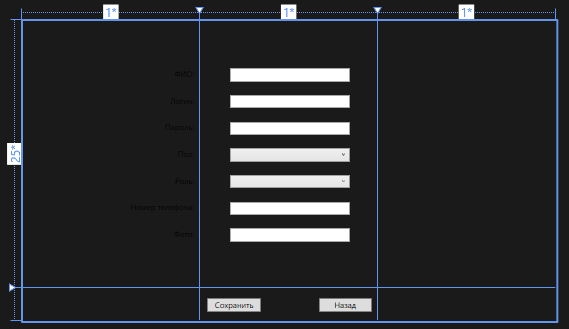
NavigationService.GoBack();

}

}

}

**Окно (Add\_Redac\_Sotrudniki.xaml)**

****

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="25\*"/>

<RowDefinition Height="3\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="1\*"/>

</Grid.ColumnDefinitions>

<StackPanel VerticalAlignment="Center" Grid.Column="1" Grid.Row="0">

<TextBox Text="{Binding FIO}" x:Name="Txt\_0" Height="20" Width="180" Margin="10"/>

<TextBox Text="{Binding Login}" x:Name="Txt\_1" Height="20" Width="180" Margin="10"/>

<TextBox Text="{Binding Password}" x:Name="Txt\_2" Height="20" Width="180" Margin="10"/>

<ComboBox Text="{Binding Pol}" x:Name="Txt\_3" Height="20" Width="180" Margin="10">

<ComboBoxItem>Мужской</ComboBoxItem>

<ComboBoxItem>Женский</ComboBoxItem>

</ComboBox>

<ComboBox Text="{Binding Role}" x:Name="Txt\_4" Height="20" Width="180" Margin="10">

<ComboBoxItem>Администратор</ComboBoxItem>

<ComboBoxItem>Менеджер А</ComboBoxItem>

<ComboBoxItem>Менеджер С</ComboBoxItem>

<ComboBoxItem>Удален</ComboBoxItem>

</ComboBox>

<TextBox Text="{Binding nomer\_telefona}" x:Name="Txt\_5" Height="20" Width="180" Margin="10"/>

<TextBox Text="{Binding Foto}" x:Name="Txt\_6" Height="20" Width="180" Margin="10"/>

</StackPanel>

<StackPanel VerticalAlignment="Center" Grid.Column="0" Grid.Row="0">

<TextBlock Text="ФИО:" HorizontalAlignment="Right" Height="20" Margin="10"/>

<TextBlock Text="Логин:" HorizontalAlignment="Right" Height="20" Margin="10"/>

<TextBlock Text="Пароль:" HorizontalAlignment="Right" Height="20" Margin="10"/>

<TextBlock Text="Пол:" HorizontalAlignment="Right" Height="20" Margin="10"/>

<TextBlock Text="Роль:" HorizontalAlignment="Right" Height="20" Margin="10"/>

<TextBlock Text="Номер телефона:" HorizontalAlignment="Right" Height="20" Margin="10"/>

<TextBlock Text="Фото:" HorizontalAlignment="Right" Height="20" Margin="10"/>

</StackPanel>

<Button x:Name="Save" Content="Сохранить" HorizontalAlignment="Left" Height="20" Width="80" Margin="10,0,0,0" Grid.Column="1" Grid.Row="1" Click="Save\_Click"/>

<Button x:Name="Back" Content="Назад" HorizontalAlignment="Right" Height="20" Width="80" Margin="0,0,10,0" Grid.Column="1" Grid.Row="1" Click="Back\_Click"/>

</Grid>

**Окно (Add\_Redac\_Sotrudniki.xaml.cs)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace WpfApp2.Pages

{

/// <summary>

/// Логика взаимодействия для Add\_Redac\_Sotrudniki.xaml

/// </summary>

public partial class Add\_Redac\_Sotrudniki : Page

{

Sotrudniki \_currentSotrudniki = new Sotrudniki();

public Add\_Redac\_Sotrudniki(Sotrudniki selectSotrudniki)

{

InitializeComponent();

if (selectSotrudniki != null)

\_currentSotrudniki = selectSotrudniki;

DataContext = \_currentSotrudniki;

}

private void Save\_Click(object sender, RoutedEventArgs e)

{

StringBuilder errors = new StringBuilder();

if (string.IsNullOrWhiteSpace(Txt\_0.Text))

errors.AppendLine("Укажите !");

if (string.IsNullOrWhiteSpace(Txt\_1.Text))

errors.AppendLine("Укажите !");

if (string.IsNullOrWhiteSpace(Txt\_2.Text))

errors.AppendLine("Укажите !");

if ((Txt\_3 == null))

errors.AppendLine("Выберите!");

if ((Txt\_4 == null))

errors.AppendLine("Выберите!");

if (string.IsNullOrWhiteSpace(Txt\_5.Text))

errors.AppendLine("Укажите !");

if (errors.Length > 0)

{

MessageBox.Show(errors.ToString());

}

else

{

if (\_currentSotrudniki.ID == 0)

DataBaseConnection.entites.Sotrudniki.Add(\_currentSotrudniki);

try

{

DataBaseConnection.entites.SaveChanges();

MessageBox.Show("ок");

NavigationService.Navigate(new Pages.Administrator());

}

catch

{

MessageBox.Show("no ok");

}

}

}

private void Back\_Click(object sender, RoutedEventArgs e)

{

NavigationService.GoBack();

}

}

}

**Окно (Requast\_Pavilion.xaml)**

**Изображение выглядит как текст

Автоматически созданное описание**

<Grid Background="White">

<Grid.RowDefinitions>

<RowDefinition Height="34\*"/>

<RowDefinition Height="191\*"/>

</Grid.RowDefinitions>

<DataGrid x:Name="Grid\_pavilion" IsReadOnly="True" AutoGenerateColumns="False" Grid.Row="1">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding Tc.Status}" Header="Статус ТЦ" Width="auto"/>

<DataGridTextColumn Binding="{Binding nazvanie\_Tc}" Header="Название ТЦ" Width="auto"/>

<DataGridTextColumn Binding="{Binding etaj}" Header="Этаж" Width="auto"/>

<DataGridTextColumn Binding="{Binding nomer\_paviliona}" Header="Номер павильона" Width="auto"/>

<DataGridTextColumn Binding="{Binding plochad}" Header="Площадь" Width="auto"/>

<DataGridTextColumn Binding="{Binding status}" Header="Статус" Width="auto"/>

<DataGridTextColumn Binding="{Binding kof\_dobav\_stoimosti}" Header="Коф.добавочной стоимости" Width="auto"/>

<DataGridTextColumn Binding="{Binding stoimost\_za\_kv}" Header="Стоимость за кв.м." Width="auto"/>

</DataGrid.Columns>

</DataGrid>

<TextBlock Text="Этаж:" Height="20" Width="150" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="20,0,0,0"/>

<TextBox SelectionChanged="Etaj\_SelectionChanged" x:Name="Etaj" Height="20" Width="150" HorizontalAlignment="Left" VerticalAlignment="Center" Margin="20 0 0 0"/>

<TextBlock Text="Сатус:" Height="20" Width="150" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="190,0,0,0"/>

<TextBox SelectionChanged="Status\_SelectionChanged" x:Name="Status" Height="20" Width="150" HorizontalAlignment="Left" VerticalAlignment="Center" Margin="190 0 0 0"/>

<TextBlock Text="Площадь от:" Height="20" Width="150" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="360,0,0,0"/>

<TextBox SelectionChanged="Ploshad\_ot\_SelectionChanged" x:Name="Ploshad\_ot" Height="20" Width="150" HorizontalAlignment="Left" VerticalAlignment="Center" Margin="360 0 0 0"/>

<TextBlock Text="Площадь до:" Height="20" Width="150" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="530,0,0,0"/>

<TextBox SelectionChanged="Ploshad\_do\_SelectionChanged" x:Name="Ploshad\_do" Height="20" Width="150" HorizontalAlignment="Left" VerticalAlignment="Center" Margin="530 0 0 0"/>

</Grid>

**Окно (Requast\_Pavilion.xaml.cs)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using WpfApp2;

namespace sesion.Pages

{

/// <summary>

/// Логика взаимодействия для Requast\_Pavilion.xaml

/// </summary>

public partial class Requast\_Pavilion : Page

{

public Requast\_Pavilion()

{

InitializeComponent();

Grid\_pavilion.ItemsSource = DataBaseConnection.entites.pavilion.ToList();

}

public void dc()

{

if(Status.Text.Length !=0)

{

var abc = DataBaseConnection.entites.pavilion.Where(i => i.status.Contains(Status.Text)).ToList();

Grid\_pavilion.ItemsSource = abc;

}

if(Etaj.Text.Length != 0)

{

var F = Convert.ToInt32(Etaj.Text);

var abc =DataBaseConnection.entites.pavilion.Where(i => i.etaj == F).ToList();

Grid\_pavilion.ItemsSource = abc;

}

if (Ploshad\_ot.Text.Length != 0 && Ploshad\_do.Text.Length != 0)

{

var G = Convert.ToDouble(Ploshad\_ot.Text);

var H = Convert.ToDouble(Ploshad\_do.Text);

var abc = (from a in DataBaseConnection.entites.pavilion

where a.plochad >=G

where a.plochad <= H

select a).ToList();

Grid\_pavilion.ItemsSource = abc;

}

if (Etaj.Text.Length != 0 && Status.Text.Length != 0)

{

var F = Convert.ToInt32(Etaj.Text);

var abc = (from a in DataBaseConnection.entites.pavilion

where a.status == Status.Text

where a.etaj == F

select a).ToList();

Grid\_pavilion.ItemsSource = abc;

}

if (Status.Text.Length != 0 && Ploshad\_ot.Text.Length != 0 && Ploshad\_do.Text.Length != 0)

{

var G = Convert.ToDouble(Ploshad\_ot.Text);

var H = Convert.ToDouble(Ploshad\_do.Text);

var abc = (from a in DataBaseConnection.entites.pavilion

where a.status == Status.Text

where a.plochad >= G

where a.plochad <= H

select a).ToList();

Grid\_pavilion.ItemsSource = abc;

}

if (Etaj.Text.Length != 0 && Ploshad\_ot.Text.Length != 0 && Ploshad\_do.Text.Length != 0)

{

var G = Convert.ToDouble(Ploshad\_ot.Text);

var H = Convert.ToDouble(Ploshad\_do.Text);

var F = Convert.ToInt32(Etaj.Text);

var abc = (from a in DataBaseConnection.entites.pavilion

where a.etaj ==F

where a.plochad >= G

where a.plochad <= H

select a).ToList();

Grid\_pavilion.ItemsSource = abc;

}

if (Status.Text.Length != 0 && Etaj.Text.Length != 0 && Ploshad\_ot.Text.Length != 0 && Ploshad\_do.Text.Length != 0)

{

var G = Convert.ToDouble(Ploshad\_ot.Text);

var H = Convert.ToDouble(Ploshad\_do.Text);

var F = Convert.ToInt32(Etaj.Text);

var abc = (from a in DataBaseConnection.entites.pavilion

where a.status == Status.Text

where a.etaj == F

where a.plochad >= G

where a.plochad <= H

select a).ToList();

Grid\_pavilion.ItemsSource = abc;

}

}

private void Etaj\_SelectionChanged(object sender, RoutedEventArgs e)

{

dc();

}

private void Status\_SelectionChanged(object sender, RoutedEventArgs e)

{

dc();

}

private void Ploshad\_ot\_SelectionChanged(object sender, RoutedEventArgs e)

{

dc();

}

private void Ploshad\_do\_SelectionChanged(object sender, RoutedEventArgs e)

{

dc();

}

}

}

**Окно (Requast\_Tc.xaml)**

**Изображение выглядит как текст

Автоматически созданное описание**

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="34\*"/>

<RowDefinition Height="191\*"/>

</Grid.RowDefinitions>

<DataGrid x:Name="Tc" AutoGenerateColumns="False" IsReadOnly="True" Grid.Row="1">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding Nazvanie}" Header="Название" Width="auto"/>

<DataGridTextColumn Binding="{Binding Status}" Header="Статус" Width="auto"/>

<DataGridTextColumn Binding="{Binding Kol\_Pavilionov}" Header="Количество павильонов" Width="auto"/>

<DataGridTextColumn Binding="{Binding Gorod}" Header="Город" Width="auto"/>

<DataGridTextColumn Binding="{Binding Stoimost}" Header="Стоимость" Width="auto"/>

<DataGridTextColumn Binding="{Binding Etajnost}" Header="Этажность" Width="auto"/>

<DataGridTextColumn Binding="{Binding Kof\_Dobav\_Stoimosti}" Header="Коф. добавочной стоимости" Width="auto"/>

</DataGrid.Columns>

</DataGrid>

<TextBlock Text="Город:" Height="20" Width="150" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="20,0,0,0"/>

<TextBox SelectionChanged="Gorod\_SelectionChanged" x:Name="Gorod" Height="20" Width="150" HorizontalAlignment="Left" VerticalAlignment="Center" Margin="20 0 0 0"/>

<TextBlock Text="Сатус:" Height="20" Width="150" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="190,0,0,0"/>

<TextBox SelectionChanged="Status\_SelectionChanged" x:Name="Status" Height="20" Width="150" HorizontalAlignment="Left" VerticalAlignment="Center" Margin="190 0 0 0"/>

</Grid>

**Окно (Requast\_Tc.xaml.cs)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using WpfApp2;

namespace sesion.Pages

{

/// <summary>

///

/// Логика взаимодействия для Requast\_Tc.xaml

/// </summary>

public partial class Requast\_Tc : Page

{

public Requast\_Tc()

{

InitializeComponent();

Tc.ItemsSource = DataBaseConnection.entites.TC1.ToList();

}

private async void cd()

{

await Task.Delay(100);

if(Gorod.Text.Length != 0)

{

var abc =DataBaseConnection.entites.TC1.Where(i=>i.Gorod.Contains(Gorod.Text) && i.Gorod != "удален").ToList();

Tc.ItemsSource = abc;

}

if(Status.Text.Length != 0)

{

var abc = DataBaseConnection.entites.TC1.Where(i => i.Status.Contains(Status.Text)).ToList();

Tc.ItemsSource = abc;

}

if (Gorod.Text.Length != 0 && Status.Text.Length != 0)

{

var abc = DataBaseConnection.entites.TC1.Where(i => i.Gorod.Contains(Gorod.Text) && i.Status.Contains(Status.Text)).ToList();

Tc.ItemsSource = abc;

}

}

private void Gorod\_SelectionChanged(object sender, RoutedEventArgs e)

{

cd();

}

private void Status\_SelectionChanged(object sender, RoutedEventArgs e)

{

cd();

}

}

}

Федеральное государственное образовательное бюджетное

учреждение высшего образования

**«Финансовый университет при Правительстве Российской Федерации»**

**(Финансовый университет)**

Колледж информатики и программирования

# **ОТЧЕТ О ВЫПОЛНЕНИИ ЗАДАНИЯ № 2**

# **Тема:** Товары автосервиса

Студент: Дерябин Г.А.

Группа: 4ИСИП-519

Преподаватель: Аксенова Т.Г., Морозова М.В.

Дата: 14.03.2023

**Окно (List0.xaml)**

**Изображение выглядит как текст

Автоматически созданное описание**

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="50"/>

<RowDefinition Height="10\*"/>

<RowDefinition Height="1\*"/>

</Grid.RowDefinitions>

<TextBlock Text="Название:" Grid.Row="0" Height="20" Width="auto" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="20 0 0 0"/>

<TextBox x:Name="txt\_Name\_P" Height="20" Width="150" HorizontalAlignment="Left" Margin="20 0 0 0 " SelectionChanged="txt\_Name\_P\_SelectionChanged"/>

<TextBlock Text="Описание:" Grid.Row="0" Height="20" Width="auto" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="190 0 0 0"/>

<TextBox x:Name="txt\_Description\_P" Height="20" Width="150" HorizontalAlignment="Left" Margin="190 0 0 0 " SelectionChanged="txt\_Description\_P\_SelectionChanged"/>

<TextBlock Text="Компания:" Grid.Row="0" Height="20" Width="auto" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="360 0 0 0"/>

<ComboBox x:Name="dsa" DisplayMemberPath="Name" Height="20" Width="150" Grid.Row="0" HorizontalAlignment="Left" Margin="360 0 0 0" SelectionChanged="dsa\_SelectionChanged"/>

<ComboBox x:Name="txt\_App\_Bot" Height="20" Width="150" HorizontalAlignment="Left" Margin="530 0 0 0" SelectionChanged="txt\_App\_Bot\_SelectionChanged">

<ComboBoxItem>

<TextBlock Text="По возрастанию"/>

</ComboBoxItem>

<ComboBoxItem>

<TextBlock Text="По убыванию"/>

</ComboBoxItem>

</ComboBox>

<Button x:Name="Delete" Grid.Row="2" Height="20" Width="100" Content="Delete" HorizontalAlignment="Left" Margin="20 0 0 0" Background="Transparent" Click="Delete\_Click"/>

<Button x:Name="Redact" Grid.Row="2" Height="20" Width="100" Content="Redact" HorizontalAlignment="Left" Margin="190 0 0 0" Background="Transparent" Click="Redact\_Click"/>

<Button x:Name="Add" Grid.Row="2" Height="20" Width="100" Content="Add" HorizontalAlignment="Left" Margin="360 0 0 0" Background="Transparent" Click="Add\_Click"/>

<TextBlock x:Name="Count\_gg" Grid.Row="2" Height="20" Width="100" HorizontalAlignment="Right"/>

<ListView Name="ListUser" Grid.Row="1" ScrollViewer.HorizontalScrollBarVisibility="Disabled" HorizontalContentAlignment="Center">

<ListView.ItemContainerStyle>

<Style TargetType="ListViewItem">

<Setter Property="Background" Value="White"></Setter>

<Style.Triggers>

<DataTrigger Binding="{Binding IsActive}" Value="False">

<Setter Property="Background" Value="Gray"/>

</DataTrigger>

</Style.Triggers>

</Style>

</ListView.ItemContainerStyle>

<ListView.ItemsPanel>

<ItemsPanelTemplate>

<WrapPanel Orientation="Horizontal" HorizontalAlignment="Center"></WrapPanel>

</ItemsPanelTemplate>

</ListView.ItemsPanel>

<ListView.ItemTemplate>

<DataTemplate>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="1\*"></RowDefinition>

<RowDefinition Height="10\*"></RowDefinition>

<RowDefinition Height="1\*"></RowDefinition>

</Grid.RowDefinitions>

<Image Width="250" Grid.Row="1" Stretch="UniformToFill" HorizontalAlignment="Center">

<Image.Source>

<Binding Path="MainImagePath">

<Binding.TargetNullValue>

<ImageSource>/Images/2RIST.png</ImageSource>

</Binding.TargetNullValue>

</Binding>

</Image.Source>

</Image>

<TextBlock TextAlignment="Center" Width="auto" Grid.Row="0" TextWrapping="Wrap" Text="{Binding Title}" FontSize="18" FontWeight="Bold"/>

<TextBlock VerticalAlignment="Center" TextAlignment="Left" Width="auto" Grid.Row="2" HorizontalAlignment="Left" FontSize="18" Text="{Binding Cost}"/>

<TextBlock Width="auto" Grid.Row="2" HorizontalAlignment="Right" FontSize="18" Text="{Binding IsActive}"/>

</Grid>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</Grid>

**Окно (List0.xaml.cs)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace session\_List

{

/// <summary>

/// Логика взаимодействия для List0.xaml

/// </summary>

public partial class List0 : Page

{

public int colvo;

public List0()

{

InitializeComponent();

dsa.ItemsSource = DataBaseConnection.entities.Manufacturer.ToList();

txt\_App\_Bot.SelectedIndex = 0;

colvo = DataBaseConnection.entities.Product.Count();

ca();

}

public void abc()

{

int display = ListUser.Items.Count;

Count\_gg.Text = display + " из " + colvo;

}

private void txt\_Name\_P\_SelectionChanged(object sender, RoutedEventArgs e)

{

ca();

}

private void txt\_Description\_P\_SelectionChanged(object sender, RoutedEventArgs e)

{

ca();

}

private void dsa\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

ca();

}

private void txt\_App\_Bot\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

ca();

}

private async void ca()

{

await Task.Delay(100);

var ItemsBD = DataBaseConnection.entities.Product.ToList();

if (txt\_Name\_P.Text.Length > 0)

{

ItemsBD = ItemsBD.Where(i => i.Title.ToLower().Contains(txt\_Name\_P.Text.ToLower())).ToList();

}

if (txt\_Description\_P.Text.Length > 0)

{

var requst=(from a in DataBaseConnection.entities.Product

where a.Description != null

select a).ToList();

ItemsBD = requst.Where(i => i.Description.Contains(txt\_Description\_P.Text)).ToList();

}

if (dsa.Text.Length > 0)

{

var requst = (from a in DataBaseConnection.entities.Manufacturer

where a.Name == dsa.Text

select a.ID).FirstOrDefault();

ItemsBD = ItemsBD.Where(i => i.ManufacturerID == requst).ToList();

}

if (txt\_App\_Bot.SelectedIndex == 0)

{

ItemsBD = ItemsBD.OrderBy(i => i.Cost).ToList();

}

if (txt\_App\_Bot.SelectedIndex == 1)

{

ItemsBD = ItemsBD.OrderByDescending(i => i.Cost).ToList();

}

ListUser.ItemsSource = ItemsBD.ToList();

abc();

}

private async void Delete\_Click(object sender, RoutedEventArgs e)

{

var delete0 = ListUser.SelectedItem as Product;

if (delete0 != null)

{

int id = delete0.ID;

var delete1 = (from a in DataBaseConnection.entities.ProductSale

where a.ProductID == id

select a).FirstOrDefault();

if(delete1 !=null)

{

DataBaseConnection.entities.ProductSale.Remove(delete1);

DataBaseConnection.entities.SaveChanges();

}

await Task.Delay(1000);

DataBaseConnection.entities.Product.Remove(delete0);

DataBaseConnection.entities.SaveChanges();

ca();

MessageBox.Show("Успешно");

}

}

private void Redact\_Click(object sender, RoutedEventArgs e)

{

NavigationService.Navigate(new Add\_Redact(ListUser.SelectedItem as Product));

}

private void Add\_Click(object sender, RoutedEventArgs e)

{

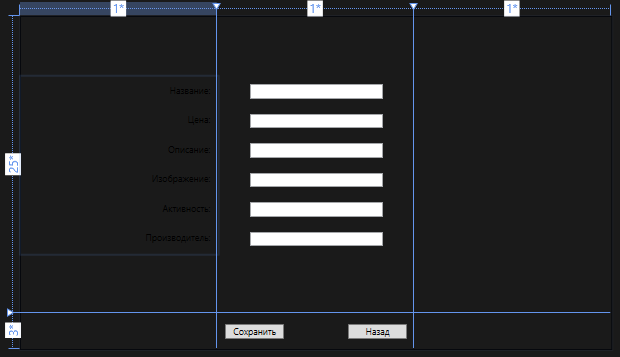
NavigationService.Navigate(new Add\_Redact(null));

}

}

}

**Окно (Add\_Redact.xaml)**

****

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="25\*"/>

<RowDefinition Height="3\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="1\*"/>

</Grid.ColumnDefinitions>

<StackPanel Grid.Column="1" Grid.Row="0" VerticalAlignment="Center">

<TextBox Text="{Binding Title}" x:Name="Txt\_0" Height="20" Width="180" Margin="10"/>

<TextBox Text="{Binding Cost}" x:Name="Txt\_1" Height="20" Width="180" Margin="10"/>

<TextBox Text="{Binding Description}" x:Name="Txt\_2" Height="20" Width="180" Margin="10"/>

<TextBox Text="{Binding MainImagePath}" x:Name="Txt\_3" Height="20" Width="180" Margin="10" SelectionChanged="Txt\_3\_SelectionChanged"/>

<TextBox Text="{Binding IsActive}" x:Name="Txt\_4" Height="20" Width="180" Margin="10"/>

<TextBox Text="{Binding ManufacturerID}" x:Name="Txt\_5" Height="20" Width="180" Margin="10"/>

</StackPanel>

<StackPanel Grid.Column="0" Grid.Row="0" VerticalAlignment="Center">

<TextBlock Text="Название:" HorizontalAlignment="Right" Height="20" Margin="10"/>

<TextBlock Text="Цена:" HorizontalAlignment="Right" Height="20" Margin="10"/>

<TextBlock Text="Описание:" HorizontalAlignment="Right" Height="20" Margin="10"/>

<TextBlock Text="Изображение:" HorizontalAlignment="Right" Height="20" Margin="10"/>

<TextBlock Text="Активность:" HorizontalAlignment="Right" Height="20" Margin="10"/>

<TextBlock Text="Производитель:" HorizontalAlignment="Right" Height="20" Margin="10"/>

</StackPanel>

<Image x:Name="Image0" Grid.Column="2" Grid.Row="0" Height="200" Width="200" Source="{Binding MainImagePath}"/>

<Button x:Name="Save" Content="Сохранить" HorizontalAlignment="Left" Height="20" Width="80" Margin="10,0,0,0" Grid.Column="1" Grid.Row="1" Click="Save\_Click"/>

<Button x:Name="Back" Content="Назад" HorizontalAlignment="Right" Height="20" Width="80" Margin="0,0,10,0" Grid.Column="1" Grid.Row="1" Click="Back\_Click"/>

</Grid>

**Окно (Add\_Redact.xaml.cs)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace session\_List

{

/// <summary>

/// Логика взаимодействия для Add\_Redact.xaml

/// </summary>

public partial class Add\_Redact : Page

{

Product \_product = new Product();

public Add\_Redact(Product SelectedProduct)

{

InitializeComponent();

if(SelectedProduct != null)

{

\_product = SelectedProduct;

}

DataContext= \_product;

}

private void Save\_Click(object sender, RoutedEventArgs e)

{

try

{

if (\_product.ID == 0)

DataBaseConnection.entities.Product.Add(\_product);

DataBaseConnection.entities.SaveChanges();

MessageBox.Show("ok");

NavigationService.Navigate(new List0());

}

catch { }

}

private void Back\_Click(object sender, RoutedEventArgs e)

{

NavigationService.Navigate(new List0());

}

private void Txt\_3\_SelectionChanged(object sender, RoutedEventArgs e)

{

try

{

string f = Txt\_3.Text;

Image0.Source = new ImageSourceConverter().ConvertFromString(f) as ImageSource;

}

catch { }

}

}

}

Федеральное государственное образовательное бюджетное

учреждение высшего образования

**«Финансовый университет при Правительстве Российской Федерации»**

**(Финансовый университет)**

Колледж информатики и программирования

# **ОТЧЕТ О ВЫПОЛНЕНИИ ЗАДАНИЯ № 3**

# **Тема:** Госавтоинспекция

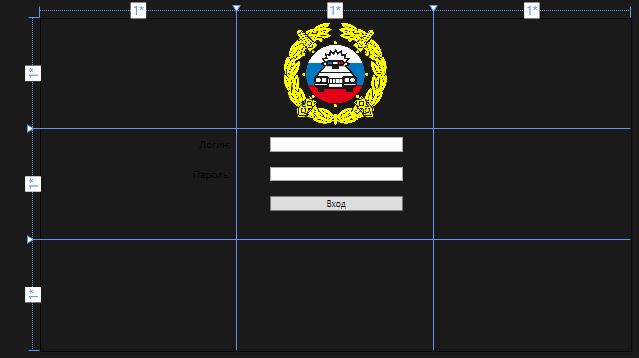
Студент: Дерябин Г.А.

Группа: 4ИСИП-519

Преподаватель: Аксенова Т.Г., Морозова М.В.

Дата: 14.03.2023

**Окно (wxod.xaml)**

****

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="1\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="1\*"/>

<RowDefinition Height="1\*"/>

<RowDefinition Height="1\*"/>

</Grid.RowDefinitions>

<StackPanel Grid.Row="1" Grid.Column="0">

<TextBlock Text="Логин:" HorizontalAlignment="Right" FontSize="14" Height="20" Margin="10"/>

<TextBlock Text="Пароль:" HorizontalAlignment="Right" FontSize="14" Height="20" Margin="10"/>

</StackPanel>

<StackPanel Grid.Row="1" Grid.Column="1">

<TextBox x:Name="Log" Height="20" Margin="10" Width="180"/>

<PasswordBox x:Name="Pass" Height="20" Margin="10" Width="180"/>

<Button x:Name="Reg" Content="Вход" Margin="10" Height="20" Width="180" Click="Reg\_Click"/>

</StackPanel>

<Grid x:Name="grid" Background="White" Grid.Row="1" Grid.ColumnSpan="3" Visibility="Hidden">

<StackPanel VerticalAlignment="Center" HorizontalAlignment="Center">

<TextBlock Text="Пинкод" FontSize="18" Margin="10" TextAlignment="Center"/>

<TextBox x:Name="pinUser" Height="25" Width="200"/>

<Button x:Name="pinClick" Height="25" Width="150" Content="Проверить" Background="Transparent" FontSize="14" Click="pinClick\_Click" Margin="10"/>

</StackPanel>

</Grid>

<Image Grid.Column="1" Grid.Row="0" Stretch="Uniform" Source="/Images/12.png" Margin="5"/>

</Grid>

**Окно (wxod.xaml.cs)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace session\_Gosauto.Pages

{

/// <summary>

/// Логика взаимодействия для wxod.xaml

/// </summary>

public partial class wxod : Page

{

public int pin;

public wxod()

{

InitializeComponent();

}

private void Reg\_Click(object sender, RoutedEventArgs e)

{

var user = DataBaseConnection.entities.users.Where(i => i.login == Log.Text && i.password == Pass.Password).FirstOrDefault();

if (user != null)

{

if (user.crush != 2)

{

Random rnd = new Random();

pin = rnd.Next(1000, 9999);

user.pincode = pin;

user.crush = 2;

DataBaseConnection.entities.SaveChanges();

MessageBox.Show("пинкод: " + pin, "Сообщение");

NavigationService.Navigate(new Pages.Page\_Drivers());

}

else

{

MessageBox.Show("Введите пинкод");

grid.Visibility = Visibility.Visible;

}

}

}

private void pinClick\_Click(object sender, RoutedEventArgs e)

{

int asd = Convert.ToInt32(pinUser.Text);

var a = DataBaseConnection.entities.users.Where(i => i.pincode ==asd).FirstOrDefault();

if (a != null)

{

NavigationService.Navigate(new Pages.Page\_Drivers());

}

else

MessageBox.Show("Неправильный пинкод");

}

}

}

**Окно (Page\_Drivers.xaml)**

**Изображение выглядит как текст

Автоматически созданное описание**

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="2\*"/>

<RowDefinition Height="10\*"/>

<RowDefinition Height="1\*"/>

</Grid.RowDefinitions>

<Grid x:Name="Voditeli" Grid.Row="2" Visibility="Hidden">

<Button x:Name="Delete" Grid.Row="2" Height="20" Width="100" Content="Delete" HorizontalAlignment="Left" Margin="20 0 0 0" Background="Transparent" Click="Delete\_Click"/>

<Button x:Name="Redact" Grid.Row="2" Height="20" Width="100" Content="Redact" HorizontalAlignment="Left" Margin="190 0 0 0" Background="Transparent" Click="Redact\_Click"/>

<Button x:Name="Add" Grid.Row="2" Height="20" Width="100" Content="Add" HorizontalAlignment="Left" Margin="360 0 0 0" Background="Transparent" Click="Add\_Click"/>

</Grid>

<Grid x:Name="Licensia" Grid.Row="2" Visibility="Hidden">

<Button x:Name="Redact0" Grid.Row="2" Height="20" Width="100" Content="Redact" HorizontalAlignment="Left" Margin="20 0 0 0" Background="Transparent" Click="Redact0\_Click"/>

<Button x:Name="Add0" Grid.Row="2" Height="20" Width="100" Content="Add" HorizontalAlignment="Left" Margin="170 0 0 0" Background="Transparent" Click="Add0\_Click"/>

</Grid>

<Grid x:Name="Licensia\_redact" Grid.Row="0" Margin="0 0 130 0" Visibility="Hidden">

<ComboBox x:Name="Activiti" Height="20" Width="100" HorizontalAlignment="Left" Margin="30 0 0 0">

<TextBlock Text="Active"/>

<TextBlock Text="Paused"/>

<TextBlock Text="Expire"/>

<TextBlock Text="Withdrawn"/>

</ComboBox>

<TextBlock Text="Описание:" HorizontalAlignment="Left" VerticalAlignment="Center" Margin="150 0 0 0"/>

<TextBox x:Name="descript\_redact" HorizontalAlignment="Left" VerticalAlignment="Center" Height="20" Width="100" Margin="220 0 0 0"/>

<Button x:Name="Save\_activiti" Height="20" Width="70" Content="Save" VerticalAlignment="Center" HorizontalAlignment="Left" Background="Transparent" Click="Save\_activiti\_Click" Margin="350 0 0 0"/>

</Grid>

<Button x:Name="pdf\_create" Height="20" Width="100" Grid.Row="0" Content="Create PDF" HorizontalAlignment="Right" Margin="0 0 150 0" VerticalAlignment="Center" Click="pdf\_create\_Click" Background="Transparent"/>

<Button x:Name="exit\_1" Grid.Row="0" Height="20" Width="100" Content="Exit" HorizontalAlignment="Right" Margin="0 0 20 0" VerticalAlignment="Center" Click="exit\_1\_Click" Background="Transparent"/>

<TabControl x:Name="Tab" Grid.Row="1" SelectionChanged="Tab\_SelectionChanged">

<TabItem Header="Водители">

<ListView Name="ListDrivers" Grid.Row="1" ScrollViewer.HorizontalScrollBarVisibility="Disabled" HorizontalContentAlignment="Center">

<ListView.ItemsPanel>

<ItemsPanelTemplate>

<WrapPanel Orientation="Horizontal" HorizontalAlignment="Center"></WrapPanel>

</ItemsPanelTemplate>

</ListView.ItemsPanel>

<ListView.ItemTemplate>

<DataTemplate>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="1\*"></RowDefinition>

<RowDefinition Height="10\*"></RowDefinition>

<RowDefinition Height="1\*"></RowDefinition>

</Grid.RowDefinitions>

<Image Width="250" Grid.Row="1" Stretch="UniformToFill" HorizontalAlignment="Center" Source="{Binding Photo\_user}"/>

<StackPanel Grid.Row="0" Orientation="Horizontal" HorizontalAlignment="Center">

<TextBlock Text="{Binding MiddleName}" FontSize="15" Margin="5"/>

<TextBlock Text="{Binding Name}" FontSize="15" Margin="5"/>

<TextBlock Text="{Binding LastName}" FontSize="15" Margin="5"/>

</StackPanel>

<TextBlock Text="{Binding phone}" FontSize="15" Grid.Row="2" HorizontalAlignment="Center"/>

</Grid>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</TabItem>

<TabItem Header="Лицензия">

<DataGrid x:Name="GridLicences" AutoGenerateColumns="False" IsReadOnly="True">

<DataGrid.ItemContainerStyle>

<Style TargetType="DataGridRow">

<Style.Triggers>

<DataTrigger Binding="{Binding Status}" Value="Paused">

<Setter Property="Background" Value="Red"/>

</DataTrigger>

<DataTrigger Binding="{Binding Status}" Value="Withdrawn">

<Setter Property="Background" Value="Red"/>

</DataTrigger>

<DataTrigger Binding="{Binding Status}" Value="Active">

<Setter Property="Background" Value="Green"/>

</DataTrigger>

<DataTrigger Binding="{Binding Status}" Value="Expire">

<Setter Property="Background" Value="Gray"/>

</DataTrigger>

</Style.Triggers>

</Style>

</DataGrid.ItemContainerStyle>

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding id}" Header="id" Width="auto"/>

<DataGridTextColumn Binding="{Binding id\_driver}" Header="id водителя" Width="auto"/>

<DataGridTextColumn Binding="{Binding Licence\_date}" Header="дата" Width="auto"/>

<DataGridTextColumn Binding="{Binding expire\_date}" Header="дата истечения" Width="auto"/>

<DataGridTextColumn Binding="{Binding categories}" Header="категория" Width="auto"/>

<DataGridTextColumn Binding="{Binding licence\_series}" Header="серия" Width="auto"/>

<DataGridTextColumn Binding="{Binding licence\_number}" Header="номер" Width="auto"/>

<DataGridTextColumn Binding="{Binding Status}" Header="статус" Width="auto"/>

<DataGridTextColumn Binding="{Binding VIN}" Header="вин номер" Width="auto"/>

<DataGridTextColumn Binding="{Binding Manufacturer}" Header="производитель" Width="auto"/>

<DataGridTextColumn Binding="{Binding Model}" Header="модель" Width="auto"/>

<DataGridTextColumn Binding="{Binding Year}" Header="год" Width="auto"/>

<DataGridTextColumn Binding="{Binding Weight}" Header="масса" Width="auto"/>

<DataGridTextColumn Binding="{Binding Color}" Header="цвет" Width="auto"/>

<DataGridTextColumn Binding="{Binding Engine\_Type}" Header="тип двигателя" Width="auto"/>

<DataGridTextColumn Binding="{Binding type\_of\_drive}" Header="привод" Width="auto"/>

</DataGrid.Columns>

</DataGrid>

</TabItem>

<TabItem Header="История изменения статуса">

<DataGrid x:Name="GridHistori" AutoGenerateColumns="False" IsReadOnly="True">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding id}" Header="id" Width="auto"/>

<DataGridTextColumn Binding="{Binding id\_licences}" Header="id лицензии" Width="auto"/>

<DataGridTextColumn Binding="{Binding status\_licences}" Header="статус лицензии" Width="auto"/>

<DataGridTextColumn Binding="{Binding description}" Header="описание" Width="auto"/>

</DataGrid.Columns>

</DataGrid>

</TabItem>

</TabControl>

</Grid>

**Окно (Page\_Drivers.xaml.cs)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace session\_Gosauto.Pages

{

/// <summary>

/// Логика взаимодействия для Page\_Drivers.xaml

/// </summary>

public partial class Page\_Drivers : Page

{

public Page\_Drivers()

{

InitializeComponent();

ListDrivers.ItemsSource = DataBaseConnection.entities.drivers.ToList();

GridLicences.ItemsSource = DataBaseConnection.entities.licences.ToList();

GridHistori.ItemsSource=DataBaseConnection.entities.history\_licences.ToList();

}

private void Delete\_Click(object sender, RoutedEventArgs e)

{

var delete = ListDrivers.SelectedItem as drivers;

if (delete != null)

{

var asd = MessageBox.Show("вы веренны?", "предупреждение", MessageBoxButton.YesNo);

if (asd == MessageBoxResult.Yes)

{

try

{

DataBaseConnection.entities.drivers.Remove(delete);

DataBaseConnection.entities.SaveChanges();

ListDrivers.ItemsSource = DataBaseConnection.entities.drivers.ToList();

MessageBox.Show("Успешно", "сообщение");

}

catch

{

MessageBox.Show("невозможно удалить", "Ошибка");

}

}

}

else

{

MessageBox.Show("Выберите строку");

}

}

private void Redact\_Click(object sender, RoutedEventArgs e)

{

if (ListDrivers.SelectedItem != null)

NavigationService.Navigate(new Pages.Add\_Redact(ListDrivers.SelectedItem as drivers));

else

MessageBox.Show("Выберите строку редактирования", "Ошибка");

}

private void Add\_Click(object sender, RoutedEventArgs e)

{

NavigationService.Navigate(new Pages.Add\_Redact(null));

}

private void exit\_1\_Click(object sender, RoutedEventArgs e)

{

var a = DataBaseConnection.entities.users.FirstOrDefault();

a.crush = 1;

DataBaseConnection.entities.SaveChanges();

Environment.Exit(0);

}

private void Redact0\_Click(object sender, RoutedEventArgs e)

{

if(GridLicences.SelectedItem !=null)

{

Licensia\_redact.Visibility = Visibility.Visible;

}

else

{

MessageBox.Show("Выберите строку", "Сообщение");

}

}

private void Add0\_Click(object sender, RoutedEventArgs e)

{

NavigationService.Navigate(new Pages.Add\_licences());

}

private void Tab\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if(Tab.SelectedIndex == 0)

{

Licensia.Visibility = Visibility.Hidden;

Voditeli.Visibility = Visibility.Visible;

pdf\_create.Visibility= Visibility.Hidden;

}

if(Tab.SelectedIndex == 1)

{

pdf\_create.Visibility = Visibility.Visible;

Voditeli.Visibility = Visibility.Hidden;

Licensia.Visibility = Visibility.Visible;

}

if(Tab.SelectedIndex == 2)

{

Voditeli.Visibility = Visibility.Hidden;

Licensia.Visibility = Visibility.Hidden;

pdf\_create.Visibility = Visibility.Hidden;

}

}

private void Save\_activiti\_Click(object sender, RoutedEventArgs e)

{

var a = GridLicences.SelectedItem as licences;

if(a != null)

{

history\_licences abc = new history\_licences();

abc.id\_licences = a.id;

abc.status\_licences = a.Status;

abc.description = descript\_redact.Text;

DataBaseConnection.entities.history\_licences.Add(abc);

a.Status = Activiti.Text;

DataBaseConnection.entities.SaveChanges();

MessageBox.Show("Успешно", "Сообщение");

GridLicences.ItemsSource=DataBaseConnection.entities.licences.ToList();

GridHistori.ItemsSource = DataBaseConnection.entities.history\_licences.ToList();

Licensia\_redact.Visibility = Visibility.Hidden;

}

}

private void pdf\_create\_Click(object sender, RoutedEventArgs e)

{

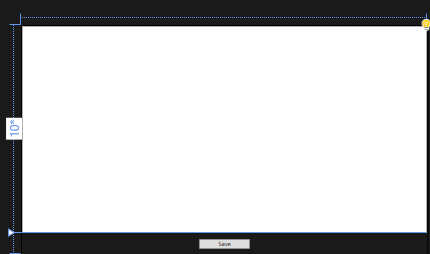
NavigationService.Navigate(new Pages.Create\_PDF(GridLicences.SelectedItem as licences));

}

}

}

**Окно (Create\_PDF.xaml)**

****

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="10\*"/>

<RowDefinition Height="1\*"/>

</Grid.RowDefinitions>

<ListView x:Name="List\_PDF" Grid.Row="0">

<ListView.ItemTemplate>

<DataTemplate>

<Grid Height="300" Width="700">

<Grid.RowDefinitions>

<RowDefinition Height="1.3\*"/>

<RowDefinition Height="2\*"/>

<RowDefinition Height="0.8\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="0.07\*"/>

<ColumnDefinition Width="0.33\*"/>

<ColumnDefinition Width="1\*"/>

</Grid.ColumnDefinitions>

<Image Grid.ColumnSpan="3" Stretch="Fill" Source="\Images\driver\_license.jpg" Grid.RowSpan="3"/>

<Image Grid.Column="1" Grid.Row="1" Source="{Binding drivers.Photo\_user}"/>

<StackPanel Orientation="Vertical" Grid.Column="2" Grid.Row="1" HorizontalAlignment="Left">

<StackPanel Orientation="Vertical">

<TextBlock Text="{Binding drivers.MiddleName}" FontSize="17"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<TextBlock Text="{Binding drivers.Name}" FontSize="17"/>

<TextBlock Text="{Binding drivers.LastName}" Margin="10 0 0 0" FontSize="17"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<TextBlock Text="{Binding Licence\_date, StringFormat=dd/mm/yyyy}" FontSize="17"/>

<TextBlock Text="{Binding expire\_date, StringFormat=dd/mm/yyyy}" Margin="20 0 0 0" FontSize="17"/>

</StackPanel>

<TextBlock Text="{Binding drivers.postcode}" FontSize="17"/>

<TextBlock Text="{Binding categories}" FontSize="17"/>

</StackPanel>

</Grid>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

<Button x:Name="SAVE" Grid.Row="1" Height="20" Width="100" HorizontalAlignment="Center" Content="Save" Click="SAVE\_Click"/>

</Grid>

**Окно (Create\_PDF.xaml.cs)**

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace session\_Gosauto.Pages

{

/// <summary>

/// Логика взаимодействия для Create\_PDF.xaml

/// </summary>

public partial class Create\_PDF : Page

{

licences \_licences = new licences();

public Create\_PDF(licences Selectlicences)

{

InitializeComponent();

if (Selectlicences != null)

{

\_licences = Selectlicences;

}

DataContext = \_licences;

string f;

f = \_licences.licence\_series;

f += \_licences.licence\_number;

var idS = \_licences.id;

var requst = (from a in DataBaseConnection.entities.licences

where a.id == idS

select a).ToList();

if (requst == null)

MessageBox.Show("fd");

List\_PDF.ItemsSource = requst;

}

private void SAVE\_Click(object sender, RoutedEventArgs e)

{

SaveFileDialog saveFileDialog1 = new SaveFileDialog();

saveFileDialog1.Filter = "(\*.pdf)|\*.pdf";

if (saveFileDialog1.ShowDialog() == true)

{

using (StreamWriter sw = new StreamWriter(saveFileDialog1.OpenFile(), System.Text.Encoding.Default))

{

sw.Write(List\_PDF.Items);

sw.Close();

}

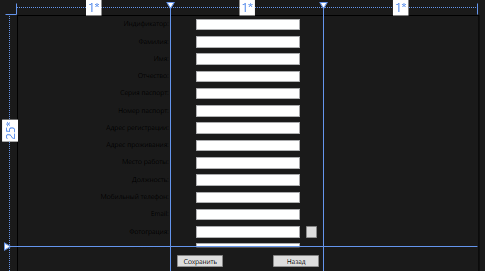
}

}

}

}

**Окно (Add\_Redact.xaml)**

****

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="25\*"/>

<RowDefinition Height="3\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="1\*"/>

</Grid.ColumnDefinitions>

<StackPanel Grid.Column="1" Grid.Row="0">

<TextBox Text="{Binding id}" x:Name="ID\_user" Height="20" Width="180" Margin="5"/>

<TextBox Text="{Binding MiddleName}" x:Name="MiddleName\_user" Height="20" Width="180" Margin="5"/>

<TextBox Text="{Binding Name}" x:Name="Name\_user" Height="20" Width="180" Margin="5"/>

<TextBox Text="{Binding LastName}" x:Name="LastName\_user" Height="20" Width="180" Margin="5"/>

<TextBox Text="{Binding passport\_serial}" x:Name="Pasport\_serial\_user" Height="20" Width="180" Margin="5"/>

<TextBox Text="{Binding passport\_number}" x:Name="Pasport\_number\_user" Height="20" Width="180" Margin="5"/>

<TextBox Text="{Binding address}" x:Name="Address\_reg\_user" Height="20" Width="180" Margin="5"/>

<TextBox Text="{Binding address\_life}" x:Name="Addres\_life\_user" Height="20" Width="180" Margin="5"/>

<TextBox Text="{Binding company}" x:Name="company\_user" Height="20" Width="180" Margin="5"/>

<TextBox Text="{Binding jobname}" x:Name="jobname\_user" Height="20" Width="180" Margin="5"/>

<TextBox Text="{Binding phone}" x:Name="phone\_user" Height="20" Width="180" Margin="5"/>

<TextBox Text="{Binding email}" x:Name="Email" Height="20" Width="180" Margin="5"/>

<StackPanel Orientation="Horizontal" Margin="30 5 0 5" HorizontalAlignment="Center">

<TextBox Text="{Binding Photo\_user}" x:Name="photo\_user" Height="20" Width="180"/>

<Button x:Name="Btn\_photo" Height="20" Width="20" Margin="10 0 0 0" Click="Btn\_photo\_Clik"/>

</StackPanel>

<TextBox Text="{Binding description}" x:Name="descriptiom\_user" Height="20" Width="180" Margin="5"/>

</StackPanel>

<StackPanel Grid.Column="0" Grid.Row="0">

<TextBlock Text="Индификатор:" HorizontalAlignment="Right" Height="20" Margin="5"/>

<TextBlock Text="Фамилия:" HorizontalAlignment="Right" Height="20" Margin="5"/>

<TextBlock Text="Имя:" HorizontalAlignment="Right" Height="20" Margin="5"/>

<TextBlock Text="Отчество:" HorizontalAlignment="Right" Height="20" Margin="5"/>

<TextBlock Text="Серия паспорт:" HorizontalAlignment="Right" Height="20" Margin="5"/>

<TextBlock Text="Номер паспорт:" HorizontalAlignment="Right" Height="20" Margin="5"/>

<TextBlock Text="Адрес регистрации:" HorizontalAlignment="Right" Height="20" Margin="5"/>

<TextBlock Text="Адрес проживания:" HorizontalAlignment="Right" Height="20" Margin="5"/>

<TextBlock Text="Место работы:" HorizontalAlignment="Right" Height="20" Margin="5"/>

<TextBlock Text="Должность:" HorizontalAlignment="Right" Height="20" Margin="5"/>

<TextBlock Text="Мобильный телефон:" HorizontalAlignment="Right" Height="20" Margin="5"/>

<TextBlock Text="Email:" HorizontalAlignment="Right" Height="20" Margin="5"/>

<TextBlock Text="Фотограция:" HorizontalAlignment="Right" Height="20" Margin="5"/>

<TextBlock Text="Замечание:" HorizontalAlignment="Right" Height="20" Margin="5"/>

</StackPanel>

<Button x:Name="Save" Content="Сохранить" HorizontalAlignment="Left" Height="20" Width="80" Margin="10,0,0,0" Grid.Column="1" Grid.Row="1" Click="Save\_Click"/>

<Button x:Name="Back" Content="Назад" HorizontalAlignment="Right" Height="20" Width="80" Margin="0,0,10,0" Grid.Column="1" Grid.Row="1" Click="Back\_Click"/>

</Grid>

**Окно (Add\_Redact.xaml.cs)**

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Net;

using System.Security.Policy;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using static System.Net.Mime.MediaTypeNames;

namespace session\_Gosauto.Pages

{

/// <summary>

/// Логика взаимодействия для Add\_Redact.xaml

/// </summary>

public partial class Add\_Redact : Page

{

drivers \_drivers = new drivers();

public Add\_Redact(drivers SelectDrivers)

{

InitializeComponent();

if(SelectDrivers != null)

{

\_drivers = SelectDrivers;

}

DataContext = \_drivers;

}

private void Btn\_photo\_Clik(object sender, RoutedEventArgs e)

{

OpenFileDialog openFileDialog = new OpenFileDialog();

if(openFileDialog.ShowDialog() == true)

{

photo\_user.Text =Convert.ToString(new FileInfo(openFileDialog.FileName));

}

}

private void Save\_Click(object sender, RoutedEventArgs e)

{

StringBuilder errors = new StringBuilder();

if (string.IsNullOrWhiteSpace(ID\_user.Text))

errors.AppendLine("Укажите индефикатор!");

if (string.IsNullOrWhiteSpace(MiddleName\_user.Text))

errors.AppendLine("Укажите фамилию!");

if (string.IsNullOrWhiteSpace(Name\_user.Text))

errors.AppendLine("Укажите имя!");

if (string.IsNullOrWhiteSpace(LastName\_user.Text))

errors.AppendLine("Укажите отчество!");

if (string.IsNullOrWhiteSpace(Pasport\_serial\_user.Text))

errors.AppendLine("Укажите серию паспорта!");

if (string.IsNullOrWhiteSpace(Pasport\_number\_user.Text))

errors.AppendLine("Укажите номер паспорта!");

if (string.IsNullOrWhiteSpace(Address\_reg\_user.Text))

errors.AppendLine("Укажите адрес регистрации!");

if (string.IsNullOrWhiteSpace(Addres\_life\_user.Text))

errors.AppendLine("Укажите адрес проживания!");

if (string.IsNullOrWhiteSpace(phone\_user.Text))

{

errors.AppendLine("Укажите телефон!");

}

long number;

if (long.TryParse(phone\_user.Text.TrimStart('+', '7', '(', '\*', '\*', '\*', ')', '\*', '\*', '\*', '-'), out number))

{

errors.AppendLine("Номер неправильного формата!");

}

if (string.IsNullOrWhiteSpace(Email.Text))

errors.AppendLine("Укажите email!");

if (string.IsNullOrWhiteSpace(photo\_user.Text))

errors.AppendLine("Укажите фото!");

if (errors.Length > 0)

{

MessageBox.Show(errors.ToString());

}

else

{

try

{

var a = photo\_user.Text;

\_drivers.Photo\_user = a;

if (\_drivers.id == 0)

{

DataBaseConnection.entities.drivers.Add(\_drivers);

}

DataBaseConnection.entities.SaveChanges();

MessageBox.Show("Успешно", "Сообщение");

NavigationService.Navigate(new Pages.Page\_Drivers());

}

catch

{

MessageBox.Show("Проверьте правильность введенных данных", "Ошибка");

}

}

}

private void Back\_Click(object sender, RoutedEventArgs e)

{

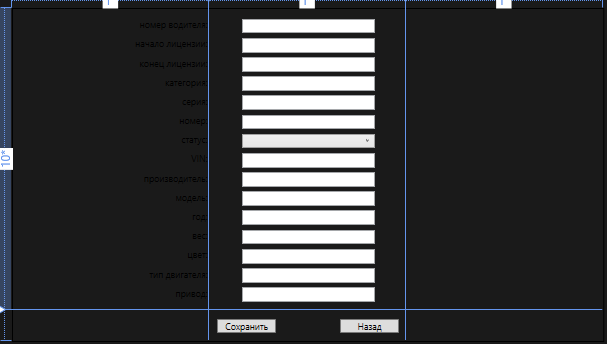
NavigationService.Navigate(new Pages.Page\_Drivers());

}

}

}

**Окно (Add\_licences.xaml)**

****

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="1\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="10\*"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Button x:Name="Save" Content="Сохранить" HorizontalAlignment="Left" Height="20" Width="80" Margin="10,0,0,0" Grid.Column="1" Grid.Row="1" Click="Save\_Click"/>

<Button x:Name="Back" Content="Назад" HorizontalAlignment="Right" Height="20" Width="80" Margin="0,0,10,0" Grid.Column="1" Grid.Row="1" Click="Back\_Click"/>

<StackPanel Grid.Row="0" Grid.Column="0" HorizontalAlignment="Right" Margin="0 10 0 0">

<TextBlock Text="номер водителя:" HorizontalAlignment="Right" Height="20" Margin="3"/>

<TextBlock Text="начало лицензии:" HorizontalAlignment="Right" Height="20" Margin="3"/>

<TextBlock Text="конец лицензии:" HorizontalAlignment="Right" Height="20" Margin="3"/>

<TextBlock Text="категория:" HorizontalAlignment="Right" Height="20" Margin="3"/>

<TextBlock Text="серия:" HorizontalAlignment="Right" Height="20" Margin="3"/>

<TextBlock Text="номер:" HorizontalAlignment="Right" Height="20" Margin="3"/>

<TextBlock Text="статус:" HorizontalAlignment="Right" Height="20" Margin="3"/>

<TextBlock Text="VIN:" HorizontalAlignment="Right" Height="20" Margin="3"/>

<TextBlock Text="производитель:" HorizontalAlignment="Right" Height="20" Margin="3"/>

<TextBlock Text="модель:" HorizontalAlignment="Right" Height="20" Margin="3"/>

<TextBlock Text="год:" HorizontalAlignment="Right" Height="20" Margin="3"/>

<TextBlock Text="вес:" HorizontalAlignment="Right" Height="20" Margin="3"/>

<TextBlock Text="цвет:" HorizontalAlignment="Right" Height="20" Margin="3"/>

<TextBlock Text="тип двигателя:" HorizontalAlignment="Right" Height="20" Margin="3"/>

<TextBlock Text="привод:" HorizontalAlignment="Right" Height="20" Margin="3"/>

</StackPanel>

<StackPanel Grid.Row="0" Grid.Column="1" HorizontalAlignment="Center" Margin="0 10 0 0">

<TextBox Text="{Binding id\_driver}" x:Name="Txt\_0" Height="20" Width="180" Margin="3"/>

<TextBox Text="{Binding Licence\_date}" x:Name="Txt\_1" Height="20" Width="180" Margin="3"/>

<TextBox Text="{Binding expire\_date}" x:Name="Txt\_2" Height="20" Width="180" Margin="3"/>

<TextBox Text="{Binding categories}" x:Name="Txt\_3" Height="20" Width="180" Margin="3"/>

<TextBox Text="{Binding licence\_series}" x:Name="Txt\_4" Height="20" Width="180" Margin="3"/>

<TextBox Text="{Binding licence\_number}" x:Name="Txt\_5" Height="20" Width="180" Margin="3"/>

<ComboBox Margin="3" x:Name="Activiti" Height="20" Text="{Binding Status}">

<TextBlock Text="Active"/>

<TextBlock Text="Paused"/>

<TextBlock Text="Expire"/>

<TextBlock Text="Withdrawn"/>

</ComboBox >

<TextBox Text="{Binding VIN}" x:Name="Txt\_7" Height="20" Width="180" Margin="3"/>

<TextBox Text="{Binding Manufacturer}" x:Name="Txt\_8" Height="20" Width="180" Margin="3"/>

<TextBox Text="{Binding Model}" x:Name="Txt\_9" Height="20" Width="180" Margin="3"/>

<TextBox Text="{Binding Year}" x:Name="Txt\_10" Height="20" Width="180" Margin="3"/>

<TextBox Text="{Binding Weight}" x:Name="Txt\_11" Height="20" Width="180" Margin="3"/>

<TextBox Text="{Binding Color}" x:Name="Txt\_12" Height="20" Width="180" Margin="3"/>

<TextBox Text="{Binding Engine\_Type}" x:Name="Txt\_13" Height="20" Width="180" Margin="3"/>

<TextBox Text="{Binding type\_of\_drive}" x:Name="Txt\_14" Height="20" Width="180" Margin="3"/>

</StackPanel>

</Grid>

**Окно (Add\_licences.xaml)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace session\_Gosauto.Pages

{

/// <summary>

/// Логика взаимодействия для Add\_licences.xaml

/// </summary>

public partial class Add\_licences : Page

{

public licences \_licences = new licences();

public Add\_licences()

{

InitializeComponent();

DataContext= \_licences;

}

private void Save\_Click(object sender, RoutedEventArgs e)

{

int a = Convert.ToInt32(Txt\_0.Text);

var abc = DataBaseConnection.entities.licences.Where(i=>i.id\_driver == a).FirstOrDefault();

if(abc !=null)

{

if(abc.Status == "Expire")

{

ca();

}

else

{

MessageBox.Show("уже есть");

}

}

else

{

ca();

}

}

public void ca()

{

DataBaseConnection.entities.licences.Add(\_licences);

DataBaseConnection.entities.SaveChanges();

MessageBox.Show("успешно");

NavigationService.Navigate(new Pages.Page\_Drivers());

}

private void Back\_Click(object sender, RoutedEventArgs e)

{

NavigationService.GoBack();

}

}

}

Федеральное государственное образовательное бюджетное

учреждение высшего образования

**«Финансовый университет при Правительстве Российской Федерации»**

**(Финансовый университет)**

Колледж информатики и программирования

# **ОТЧЕТ О ВЫПОЛНЕНИИ ЗАДАНИЯ № 4**

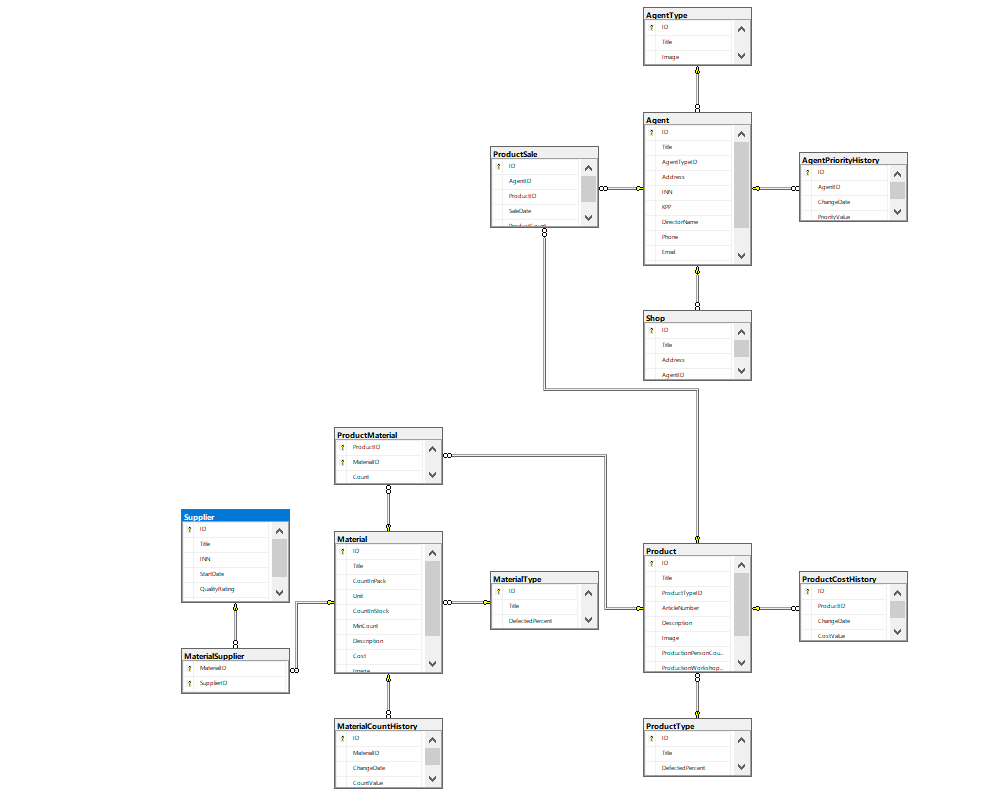
# **Тема:** Импорт 07.02 от М.В. Морозовой

Студент: Дерябин Г.А.

Группа: 4ИСИП-519

Преподаватель: Морозова М.В.

Дата: 14.03.2023

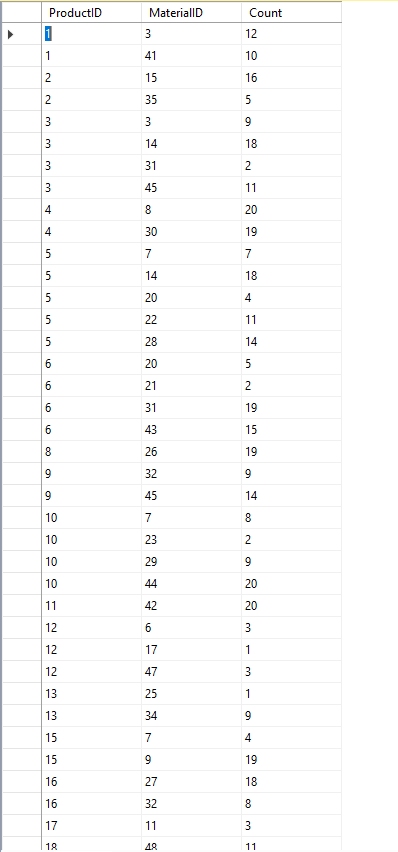
****

**Таблица (product)**

**Изображение выглядит как стол

Автоматически созданное описание**

**Таблица (ProductMaterial)**

****

**Таблица (ProductType)**

**Изображение выглядит как стол

Автоматически созданное описание**

**Таблица (Material)**

**Изображение выглядит как стол

Автоматически созданное описание**

**Таблица (MaterialType)**

**Изображение выглядит как стол

Автоматически созданное описание**

Федеральное государственное образовательное бюджетное

учреждение высшего образования

**«Финансовый университет при Правительстве Российской Федерации»**

**(Финансовый университет)**

Колледж информатики и программирования

# **ОТЧЕТ О ВЫПОЛНЕНИИ ЗАДАНИЯ № 5**

# **Тема:** 14.02 задание от М.В. Морозовой

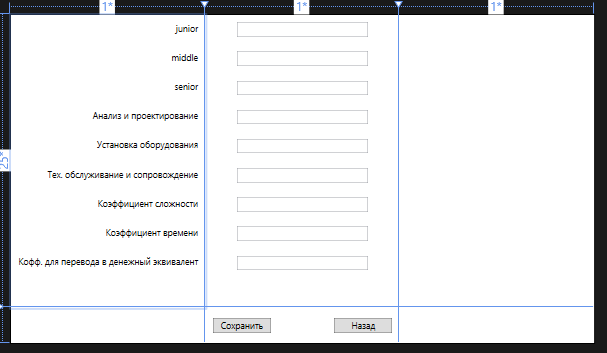
Студент: Дерябин Г.А.

Группа: 4ИСИП-519

Преподаватель: Морозова М.В.

Дата: 14.03.2023

**Окно (Redact.xaml)**

****

<Grid Background="White">

<Grid.RowDefinitions>

<RowDefinition Height="25\*"/>

<RowDefinition Height="3\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="1\*"/>

</Grid.ColumnDefinitions>

<StackPanel Grid.Column="1" Grid.Row="0">

<TextBox Text="{Binding Junior}" x:Name="Txt\_0" Height="20" Width="180" Margin="10"/>

<TextBox Text="{Binding Middle}" x:Name="Txt\_1" Height="20" Width="180" Margin="10"/>

<TextBox Text="{Binding Senior}" x:Name="Txt\_2" Height="20" Width="180" Margin="10"/>

<TextBox Text="{Binding Kof\_Analiz\_proektirovanie}" x:Name="Txt\_3" Height="20" Width="180" Margin="10"/>

<TextBox Text="{Binding Kof\_ystanovka\_obor}" x:Name="Txt\_4" Height="20" Width="180" Margin="10"/>

<TextBox Text="{Binding Kof\_Texnicheskoe\_obslujivanie}" x:Name="Txt\_5" Height="20" Width="180" Margin="10"/>

<TextBox Text="{Binding Kof\_slojnosti}" x:Name="Txt\_6" Height="20" Width="180" Margin="10"/>

<TextBox Text="{Binding kof\_vremeni}" x:Name="Txt\_7" Height="20" Width="180" Margin="10"/>

<TextBox Text="{Binding kof\_perevoda}" x:Name="Txt\_8" Height="20" Width="180" Margin="10"/>

</StackPanel>

<StackPanel Grid.Column="0" Grid.Row="0">

<TextBlock Text="junior" HorizontalAlignment="Right" Height="20" Margin="10"/>

<TextBlock Text="middle" HorizontalAlignment="Right" Height="20" Margin="10"/>

<TextBlock Text="senior" HorizontalAlignment="Right" Height="20" Margin="10"/>

<TextBlock Text="Анализ и проектирование" HorizontalAlignment="Right" Height="20" Margin="10"/>

<TextBlock Text="Установка оборудования" HorizontalAlignment="Right" Height="20" Margin="10"/>

<TextBlock Text="Тех. обслуживание и сопровождение" HorizontalAlignment="Right" Height="20" Margin="10"/>

<TextBlock Text="Коэффициент сложности" HorizontalAlignment="Right" Height="20" Margin="10"/>

<TextBlock Text="Коэффициент времени" HorizontalAlignment="Right" Height="20" Margin="10"/>

<TextBlock Text="Кофф. для перевода в денежный эквивалент" HorizontalAlignment="Right" Height="20" Margin="10"/>

</StackPanel>

<Button x:Name="Save" Content="Сохранить" HorizontalAlignment="Left" Height="20" Width="80" Margin="10,0,0,0" Grid.Column="1" Grid.Row="1" Click="Save\_Click"/>

<Button x:Name="Back" Content="Назад" HorizontalAlignment="Right" Height="20" Width="80" Margin="0,0,10,0" Grid.Column="1" Grid.Row="1" Click="Back\_Click"/>

</Grid>

**Окно (Redact.xaml.cs)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace ggg.pages

{

/// <summary>

/// Логика взаимодействия для Redact.xaml

/// </summary>

public partial class Redact : Page

{

public users \_users = new users();

public Redact(users selectusers)

{

InitializeComponent();

if (selectusers != null)

{

\_users = selectusers;

}

DataContext = \_users;

}

private void Save\_Click(object sender, RoutedEventArgs e)

{

DataBaseConnection.entities.SaveChanges();

MessageBox.Show("Успешно");

NavigationService.GoBack();

}

private void Back\_Click(object sender, RoutedEventArgs e)

{

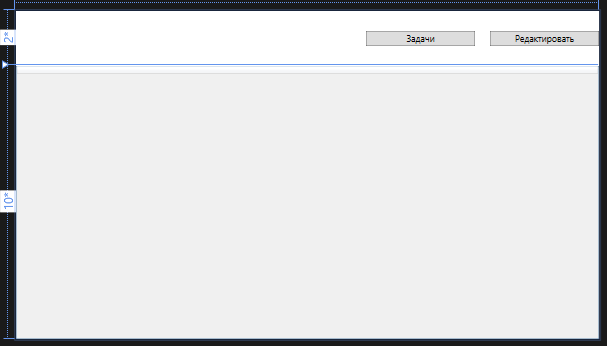
NavigationService.GoBack();

}

}

}

**Окно (meneger.xaml)**

****

<Grid Background="White">

<Grid.RowDefinitions>

<RowDefinition Height="2\*"/>

<RowDefinition Height="10\*"/>

</Grid.RowDefinitions>

<DataGrid x:Name="Grid\_meneger" Grid.Row="1" AutoGenerateColumns="True" IsReadOnly="True">

</DataGrid>

<Button x:Name="task" Content="Задачи" Grid.Row="0" Margin="0 0 170 0" Height="20" Width="150" HorizontalAlignment="Right" Click="task\_Click"/>

<Button x:Name="redact" Content="Редактировать" Grid.Row="0" Height="20" Width="150" HorizontalAlignment="Right" Click="redact\_Click"/>

</Grid>

**Окно (meneger.xaml.cs)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace ggg.pages

{

/// <summary>

/// Логика взаимодействия для meneger.xaml

/// </summary>

public partial class meneger : Page

{

public meneger()

{

InitializeComponent();

var men =Convert.ToString(App.Current.Resources["menedje"]);

var abc = (from a in DataBaseConnection.entities.users

where a.Login\_menedjer == men

select a).ToList();

Grid\_meneger.ItemsSource = abc;

}

private void redact\_Click(object sender, RoutedEventArgs e)

{

NavigationService.Navigate(new pages.Redact(Grid\_meneger.SelectedItem as users));

}

private void task\_Click(object sender, RoutedEventArgs e)

{

NavigationService.Navigate(new pages.listtask());

}

}

}

**Окно (listtask.xaml)**

**Изображение выглядит как текст

Автоматически созданное описание**

<Grid Background="White">

<Grid.RowDefinitions>

<RowDefinition Height="1\*"/>

<RowDefinition Height="10\*"/>

</Grid.RowDefinitions>

<DataGrid Grid.Row="1" x:Name="Grid\_task" AutoGenerateColumns="False" IsReadOnly="True">

<DataGrid.Columns>

<DataGridTextColumn Header="Заголовок задачи" Binding="{Binding task\_for\_executers.Title}"/>

<DataGridTextColumn Header="Статус задачи" Binding="{Binding task\_for\_executers.Status}"/>

<DataGridTextColumn Header="ФИО исполнителя" Binding="{Binding users1.FIO\_ispolnitel}"/>

<DataGridTextColumn Header="ФИО менеджер" Binding="{Binding users1.FIO\_menedjer}"/>

</DataGrid.Columns>

</DataGrid>

<StackPanel Orientation="Horizontal">

<Button x:Name="add" Content="Add" Height="20" Width="100" Margin="5" Background="Transparent" Click="add\_Click"/>

<Button x:Name="edit" Content="Edit" Height="20" Width="100" Margin="5" Background="Transparent" Click="edit\_Click"/>

<Button x:Name="delete" Content="Delete" Height="20" Width="100" Margin="5" Background="Transparent" Click="delete\_Click"/>

</StackPanel>

</Grid>

**Окно (listtask.xaml.cs)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace ggg.pages

{

/// <summary>

/// Логика взаимодействия для listtask.xaml

/// </summary>

public partial class listtask : Page

{

public listtask()

{

InitializeComponent();

gg1();

}

public void gg1()

{

var men = Convert.ToInt32(App.Current.Resources["ispolnitel"]);

var abc = (from a in DataBaseConnection.entities.asd

where a.users == men

select a).ToList();

Grid\_task.ItemsSource = abc;

}

private void add\_Click(object sender, RoutedEventArgs e)

{

}

private void edit\_Click(object sender, RoutedEventArgs e)

{

}

private void delete\_Click(object sender, RoutedEventArgs e)

{

var ar = Grid\_task.SelectedItem as asd;

DataBaseConnection.entities.asd.Remove(ar);

DataBaseConnection.entities.SaveChanges();

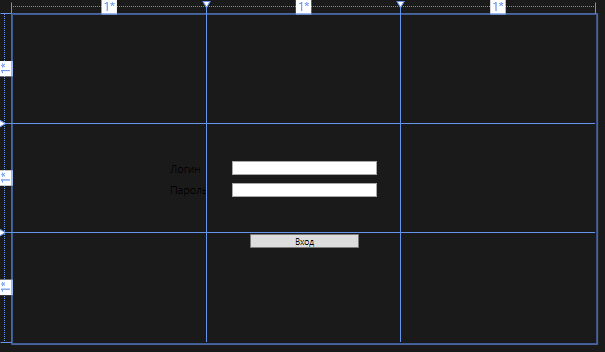
gg1();

}

}

}

**Окно (Auto\_page.xaml)**

****

**Окно (Auto\_page.xaml)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace ggg.pages

{

/// <summary>

/// Логика взаимодействия для Auto\_page.xaml

/// </summary>

public partial class Auto\_page : Page

{

public Auto\_page()

{

InitializeComponent();

}

private void Vxod\_Click(object sender, RoutedEventArgs e)

{

var gg = DataBaseConnection.entities.users.Where(i => i.Login\_menedjer == Login.Text && i.Password == Password.Password).FirstOrDefault();

if (gg == null)

{

MessageBox.Show("Пользователь не найден");

}

else

{

var b0= Convert.ToString(gg.Login\_menedjer);

var b1 = Convert.ToString(gg.id);

App.Current.Resources["menedje"] = b0;

App.Current.Resources["ispolnitel"] = b1;

NavigationService.Navigate(new pages.meneger());

}

}

}

}

Федеральное государственное образовательное бюджетное

учреждение высшего образования

**«Финансовый университет при Правительстве Российской Федерации»**

**(Финансовый университет)**

Колледж информатики и программирования

# **ОТЧЕТ О ВЫПОЛНЕНИИ ЗАДАНИЯ № 6**

# **Тема:** Тестирование от Аксёновой Т.Г. от 20.02.2023

Студент: Дерябин Г.А.

Группа: 4ИСИП-519

Преподаватель: Аксенова Т.Г.

Дата: 14.03.2023

**UnitTestManA.cs**

using CHM2019.Pages;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

namespace ChMUnitTestProject

{

[TestClass]

public class UnitTestManA

{

[TestMethod]

public void AuthTestManASuccess()

{

var page = new AuthPage();

Assert.IsTrue(page.Auth("Vladlena@gmai.com", "07i7Lb"));

Assert.IsTrue(page.Auth("Galina@gmai.com", "8KC7wJ"));

Assert.IsTrue(page.Auth("Anisa@gmai.com", "Wh4OYm"));

}

}

}

**UnitTestManC.cs**

using CHM2019.Pages;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

namespace ChMUnitTestProject

{

[TestClass]

public class UnitTestManC

{

[TestMethod]

public void AuthTestManCSuccess()

{

var page = new AuthPage();

Assert.IsTrue(page.Auth("Adam@gmai.com", "7SP9CV"));

Assert.IsTrue(page.Auth("Vseslava@gmai.com", "fhDSBf"));

Assert.IsTrue(page.Auth("Feafan@gmai.com", "Kc1PeS"));

}

}

}

**UnitTestAdmin.cs**

using CHM2019.Pages;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

namespace ChMUnitTestProject

{

[TestClass]

public class UnitTestAdmin

{

[TestMethod]

public void AuthTestAdminSuccess()

{

var page = new AuthPage();

Assert.IsTrue(page.Auth("Elizor@gmai.com", "yntiRS"));

Assert.IsTrue(page.Auth("Yli@gmai.com", "GwffgE"));

Assert.IsTrue(page.Auth("Vasilisa@gmai.com", "d7iKKV"));

Assert.IsTrue(page.Auth("Miron@gmai.com", "x58OAN"));

}

}

}

**UnitTest1.cs**

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

namespace ChMUnitTestProject

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void TestMethod1()

{

int res = 2 + 2;

Assert.AreEqual(res, 4);

Assert.AreNotEqual(res,5);

Assert.IsFalse(res>5);

Assert.IsTrue(res<5);

}

}

}

## **Аннотация теста**

|  |  |
| --- | --- |
| **Название проекта** | Торговые центры |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Дерябин Глеб Александрович |
| **Дата(ы) теста** | 07.03.2023 |

## **Расшифровка тестовых информационных полей:**

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| **Название проекта** | Название тестируемого проекта |
| **Рабочая версия** | Версия проекта/программного обеспечения (первый тест считается 1.0). |
| **Имя тестирующего** | Имя того, кто проводил тесты |
| **Дата(ы) теста** | Дата(ы) проведения тестов – это один или несколько дней. Если тесты проводились в более протяженный период времени, нужно отметить отдельную дату для каждого теста. |
| **Тестовый пример #** | Уникальный ID для каждого тестового примера. Следуйте некоторым конвенциям, чтобы указать типы тестов. Например,‘TC\_UI\_1′ означает‘user interface test case #1′ ( ТС\_ПИ\_1: тестовый случай пользовательского интерфейса#1) |
| **Приоритет тестирования**  *(Низкий/Средний/Высокий)* | Насколько важен каждый тест. Приоритет тестирования для бизнес-правил и функциональных тестовых случаев может быть средним или высоким, в то время как незначительные случаи пользовательского интерфейса могут иметь низкий приоритет. |
| **Заголовок/название теста** | Название тестового случая. Например, Подтвердите страницу авторизации с действительным именем пользователя и паролем. |
| **Краткое изложение теста** | Описание того, что должен достичь тест. |
| **Этапы теста** | Перечислите все этапы теста подробно. Запишите этапы теста в том порядке, в котором они должны быть реализованы. Предоставьте как можно больше подробностей и разъяснений. Пронумерованный список – хорошая идея. |
| **Тестовые данные** | Перечислите/опишите все тестовые данные, используемые для данного тестового случая. Так, фактические используемые входные данные можно отслеживать по результатам тестирования. Например, Имя пользователя и пароль для подтверждения входа. |
| **Ожидаемый результат** | Каким должен быть вывод системы после выполнения теста? Подробно опишите ожидаемый результат, включая все сообщения/ошибки, которые должны отображаться на экране. |
| **Фактический результат** | Каким должен быть фактический результат после выполнения теста? Опишите любое релевантное поведение системы после выполнения теста. |
| **Предварительное условие** | Любые предварительные условия, которые должны быть выполнены до выполнения теста. Перечислите все предварительные условия для выполнения этого тестового случая. |
| **Постусловие** | Каким должно быть состояние системы после выполнения теста? |
| **Статус**  *(Зачет/Незачет)* | Если фактический результат не соответствует ожидаемому результату, отметьте тест как неудачный. В ином случае обновление пройдено. |
| **Примечания/комментарии** | Используйте эту область для любых дополнительных заметок/комментариев/вопросов. Эта область предназначена для поддержки вышеуказанных полей (например, если есть некоторые особые условия, которые не могут быть описаны в любом из вышеуказанных полей, или если есть вопросы, связанные с ожидаемыми или фактическими результатами). |

## Тестовый пример #1:

|  |  |
| --- | --- |
| **Тестовый пример #** | ТЦ\_1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка окна авторизации |
| **Краткое изложение теста** | Проверка роли администратор |
| **Этапы теста** | Ввод логина и пароля |
| **Тестовые данные** | Логин([Elizor@gmai.com](mailto:Elizor@gmai.com))  Пароль(yntiRS) |
| **Ожидаемый результат** | Переход на окно администратора |
| **Фактический результат** | Переход на окно администратора |
| **Статус** | Зачет |
| **Предварительное условие** | Открыть программу |
| **Постусловие** | Нет |
| **Примечания/комментарии** | Нет |

## Тестовый пример #2:

|  |  |
| --- | --- |
| **Тестовый пример #** | ТЦ\_2 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка окна авторизации |
| **Краткое изложение теста** | Проверка роли Менеджер С |
| **Этапы теста** | Ввод логина и пароля |
| **Тестовые данные** | Логин(Adam@gmai.com)  Пароль(7SP9CV) |
| **Ожидаемый результат** | Переход на окно Менеджер С |
| **Фактический результат** | Переход на окно Менеджер С |
| **Статус** | Зачет |
| **Предварительное условие** | Открыть программу |
| **Постусловие** | Нет |
| **Примечания/комментарии** | Нет |

## Тестовый пример #3:

|  |  |
| --- | --- |
| Тестовый пример **#** | ТЦ\_3 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка окна авторизации |
| **Краткое изложение теста** | Проверка роли Менеджер А |
| **Этапы теста** | Ввод логина и пароля |
| **Тестовые данные** | Логин(Vladlena@gmai.com)  Пароль(07i7Lb) |
| **Ожидаемый результат** | Переход на окно Менеджер А |
| **Фактический результат** | Переход на окно Менеджер А |
| **Статус** | Зачет |
| **Предварительное условие** | Открыть программу |
| **Постусловие** | Нет |
| **Примечания/комментарии** | Нет |

## Тестовый пример #4:

|  |  |
| --- | --- |
| Тестовый пример **#** | ТЦ\_4 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка окна авторизации |
| **Краткое изложение теста** | Проверка роли Удален |
| **Этапы теста** | Ввод логина и пароля |
| **Тестовые данные** | Логин(Kar@gmai.com)  Пароль(6QF1WB) |
| **Ожидаемый результат** | Вывод сообщения с информацией, что пользователь удален |
| **Фактический результат** | Сообщения, что пользователь удален |
| **Статус** | Зачет |
| **Предварительное условие** | Открыть программу |
| **Постусловие** | Нет |
| **Примечания/комментарии** | Нет |

## Test case #5:

|  |  |
| --- | --- |
| **Test Case #** | ТЦ\_5 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка окна авторизации |
| **Краткое изложение теста** | Проверка несуществующего пользователя |
| **Этапы теста** | Ввод логин и пароль |
| **Тестовые данные** | Логин(111@gmai.com)  Пароль(55F1WB) |
| **Ожидаемый результат** | Вывод сообщения с информацией, что пользователь не найден |
| **Фактический результат** | Сообщения, что пользователь не найден |
| **Статус** | Зачет |
| **Предварительное условие** | Открыть программу |
| **Постусловие** | Нет |
| **Примечания/комментарии** | Нет |

Федеральное государственное образовательное бюджетное

учреждение высшего образования

**«Финансовый университет при Правительстве Российской Федерации»**

**(Финансовый университет)**

Колледж информатики и программирования

# **ОТЧЕТ О ВЫПОЛНЕНИИ ЗАДАНИЯ № 7**

**Тема:** Задания по веб от Пташкина О.Г.

Студент: Дерябин Г.А.

Группа: 4ИСИП-519

Преподаватель: Пташкин О.Г.

Дата: 14.03.2023

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Отдел продаж</title>

<link rel="stylesheet" href="aaa.css">

</head>

<body>

    <div class="Q">

        <a class="a1" href="#">Отдел продаж</a>

        <a class="a1" href="#">Каталог объектов</a>

        <a class="a1" href="#">Блог</a>

        <a class="a1" href="#">Команда</a>

        <a class="a1" href="#">FAQ</a>

        <a class="a1" href="#">Контакты</a>

        <a class="a1" href="#">+7(495)659-99-88</a>

        <a class="a1" href="#">Заказать звонок</a>

        </div>

<h1> Искусство обращения </h1>

</h1>с недвижимостью</h1>

<p>Недвижимость-то,чем мы живем и то, что заставляет двигаться вперед. Отвечаем за результат с 2015

<p>

<img src="mk.png" alt="">

<div class="two\_\_header">

<h3>АН “Отдел продаж” - 15 лет

на рынке Сочи</h3>

<p>Более 15 лет наши специалисты в области недвижимости и права помогают покупать квартиры максимально безопасно и дешево</p>

</div>

<div class="two\_\_section">

<div class="two\_\_block">

<div>

<span class="text\_\_blue">16</span>

домов

</div>

<p class="text\_\_gray">Полностью заселены всеми довольными клиентами</p>

</div>

<div class="two\_\_block">

<div>

<span class="text\_\_blue">3000</span>

клиентов

</div>

<p class="text\_\_gray">Уже обратились к нам и поселились в своих квартирах</p>

</div>

<div class="two\_\_block">

<div>

<span class="text\_\_blue">2000</span>

человек

</div>

<p class="text\_\_gray">Переедут в Сочи благодаря нашим услугам и так далее</p>

</div>

<div class="two\_\_block">

<div>

<span class="text\_\_blue">350+</span>

сделок

</div>

<p class="text\_\_gray">Каждый день по всему Сочи</p>

<div>

<div>

<p>Получите выгодное

предложение!

<p>Ещё год назад цены на недвижимость были ниже на 34%.

Лучшее время для покупки квартиры - "прошлой зимой"

</p>

</div>

<div>

<input+7 (\_\_\_) \_\_\_ - \_\_ - \_\_>

<button>Оставить заявку</button>

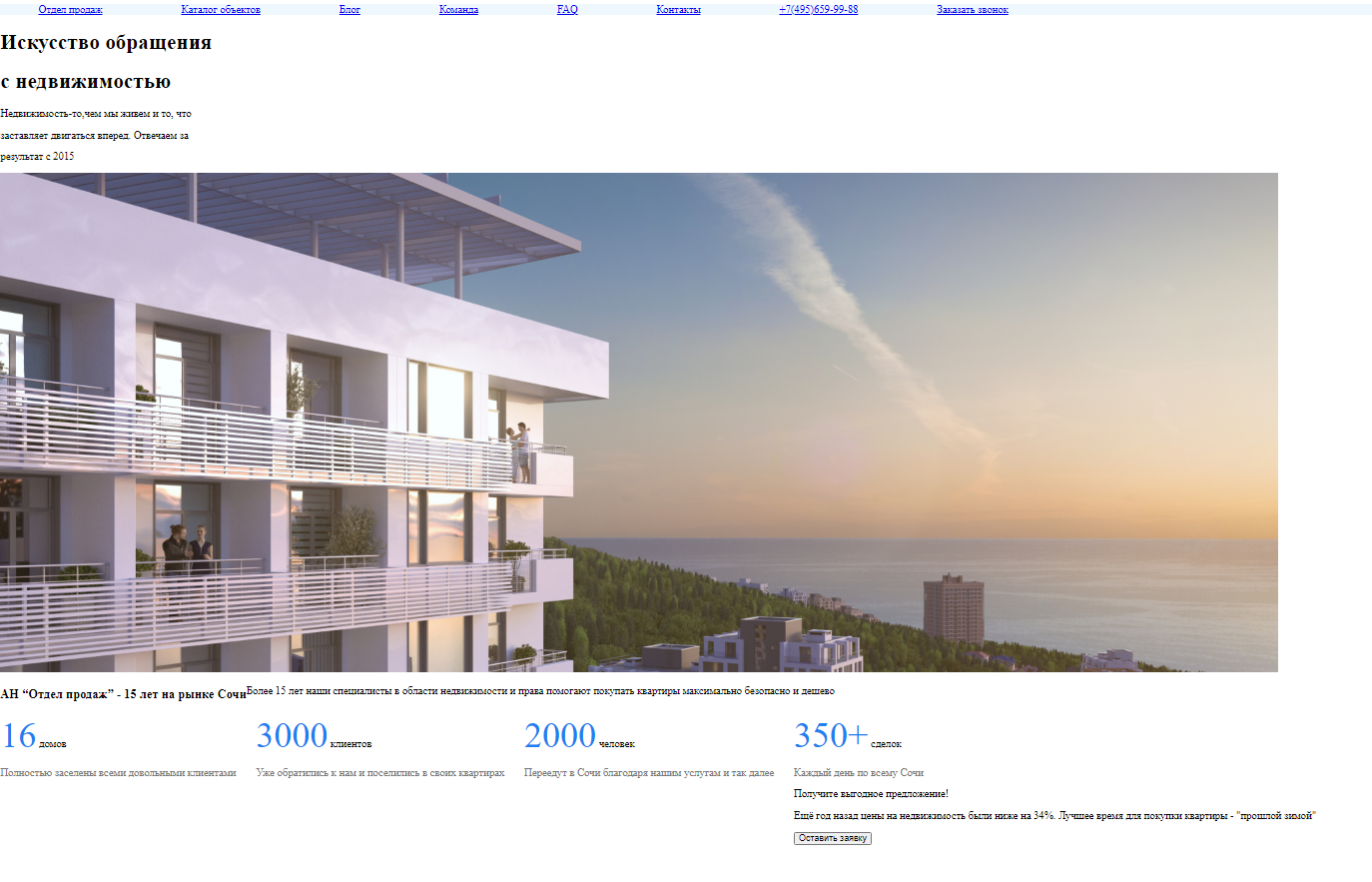
</div>

</div>

</div>

</body>

</html>



Федеральное государственное образовательное бюджетное

учреждение высшего образования

**«Финансовый университет при Правительстве Российской Федерации»**

**(Финансовый университет)**

Колледж информатики и программирования

# **ОТЧЕТ О ВЫПОЛНЕНИИ ЗАДАНИЯ № 8**

# **Тема:** Конспект по Javascript

Студент: Дерябин Г.А.

Группа: 4ИСИП-519

Преподаватель: Сибирев И.В.

Дата: 14.03.2023

Javascript

Javascript – динамический скриптовый язык программирования высокого уровня. Он отличается мультипарадигменностью. Речь идет о поддержке функционального, императивного, событийно-ориентированного стилей. Чаще всего язык используется для создания интерактивных веб-страниц и приложений. Востребованность, сферы применения. Неизменно высокий интерес к Javascript подтверждают специальные международные рейтинги. JavaScript — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией спецификации ECMAScript (стандарт ECMA-262).

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

На JavaScript оказали влияние многие языки, при разработке была цель сделать язык похожим на Java. Языком JavaScript не владеет какая-либо компания или организация, что отличает его от ряда языков программирования, используемых в веб-разработке.

Название «JavaScript» является зарегистрированным товарным знаком корпорации Oracle в США.

В 1992 году компания Nombas (впоследствии приобретённая Openwave) начала разработку встраиваемого скриптового языка Cmm (Си-минус-минус), который, по замыслу разработчиков, должен был стать достаточно мощным, чтобы заменить макросы, сохраняя при этом схожесть с Си, чтобы разработчикам не составляло труда изучить его. Главным отличием от Си была работа с памятью. В новом языке всё управление памятью осуществлялось автоматически: не было необходимости создавать буфера, объявлять переменные, осуществлять преобразование типов. В остальном языки сильно походили друг на друга: в частности, Cmm поддерживал стандартные функции и операторы Си. Cmm был переименован в ScriptEase, поскольку исходное название звучало слишком негативно, а упоминание в нём Си «отпугивало» людей. На основе этого языка был создан проприетарный продукт CEnvi. В конце ноября 1995 года Nombas разработала версию CEnvi, внедряемую в веб-страницы. Страницы, которые можно было изменять с помощью скриптового языка, получили название Espresso Pages — они демонстрировали использование скриптового языка для создания игры, проверки пользовательского ввода в формы и создания анимации. Espresso Pages позиционировались как демоверсия, призванная помочь представить, что случится, если в браузер будет внедрён язык Cmm. Работали они только в 16-битовом Netscape Navigator под управлением Windows.

JavaScript, наверное, самый известный мультипарадигменный язык, в котором очень много неочевидных особенностей. Но тем не менее любим ли мы его или ругаем, факт остается фактом — это основной язык, на котором работает современный web.  
  
В ушедшем году, вышел стандарт ECMAScript 2015 (неформально ES6), который сильно изменил, то к чему мы привыкли. Появилась масса новых возможностей, которые по сути представляют собой современное надмножество языка, пытающегося решить существующие проблемы. Class, let, const, стрелочные функции… разработчик, который ранее не видел код, написанный на ES6, не сразу догадается, что перед ним, по сути, старый добрый JS.  
  
Есть масса прекрасных статей, посвященных современному стандарту. В этом же посте я хочу показать, что нам может предложить современный JS, когда необходимо решить насущную задачу. Например, поздравить всех c Новым Годом.

JavaScript в элементе script

Самый простой способ внедрения JavaScript в HTML-документ – использование тега <script>. Теги <script> часто помещают в элемент <head>, и ранее этот способ считался чуть ли не обязательным. Однако в наши дни теги <script> используются как в элементе <head>, так и в теле веб-страниц.

Таким образом, на одной веб-странице могут располагаться сразу несколько сценариев. В какой последовательности браузер будет выполнять эти сценарии? Как правило, выполнение сценариев браузерами происходит по мере их загрузки. Браузер читает HTML-документ сверху вниз и, когда он встречает тег <script>, рассматривает текст программы как сценарий и выполняет его. Остальной контент страницы не загружается и не отображается, пока не будет выполнен весь код в элементе <script>.

<!DOCTYPE html>  
<html>  
<head>  
<meta charset="utf-8">  
</head>  
<body>  
<p>Это обычный HTML документ</p>  
  
<script language="JavaScript">  
    alert("Привет, мир!");  
</script>  
  
<p>Выходим обратно в HTML</p>  
</body>  
</html>

Внешний JavaScript

Если JavaScript-кода много – его выносят в отдельный файл, который, как правило, имеет расширение .js.

Чтобы включить в HTML-документ JavaScript-кoд из внешнего файла, нужно использовать атрибут src (source) тега <script>. Его значением должен быть URL-aдpec файла, в котором содержится JS-код:

<script src="/scripts/script.js"></script>

В этом примере указан абсолютный путь к файлу с именем script.js, содержащему скрипт (из корня сайта). Сам файл должен содержать только JavaScript-кoд, который иначе располагался бы между тегами <script> и </script>.

Комментарии в javascript

В JavaScript есть два типа комментариев. Однострочные комментарии в JavaScript записываются с помощью двух слешей //.// This is a comment. Интерпретатор будет игнорировать все символы после // до конца строки. Многострочные комментарии заключаются в теги /\* и \*/. Если вы знакомы с CSS, вы уже знаете, как работают такие комментарии./ \* This is a comment\*/

Где писать код?

JavaScript код пишется прямо на HTML странице внутри тега script. Этот тег можно размещать в любом месте страницы.

JavaScript — вывод информации

Первый способ: функция writeln

Второй способ: функция alert

Третий способ: функция getElementById

Заключение

Здравствуйте, в прошлой статье мы рассматривали синтаксис языка JavaScript. Мы узнали что такое циклы, переменные, массивы, логические операторы, функции и так далее.

Все эти операции проходят внутри страницы и никак не отображаются. Сегодня же мы научимся выводить необходимые данные на экран. Выводить будем обычную числовую переменную с помощью ActionScript, самыми разными способами.

Для того чтобы использовать все способы, которые будут указаны ниже, нам необходимо обращаться к объекту document. Это отдельный элемент языка JavaScript, который предназначен для работы с веб-сайтами.

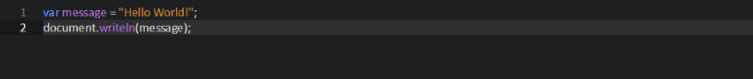
Первый способ: функция writeln

Для вывода переменной таким образом нам необходимо её создать. Напишем такую строку: var message = «Hello World!».

Теперь поговорим о функции writeln. Её лучше всего использовать только при загрузке исходного варианта страницы, так внутренний контент будет изменен.

С помощью таких функций как writeln можно создать страницу, полностью написанную на JavaScript. Кроме этого, данная функция может быть полезна в том случае, когда необходимо уменьшить количество запросов с сервера.

Для того чтобы воспользоваться данной функцией, записываем команду document.writeln, в скобках записываем название нашей переменной, которое затем будет показано на экране.



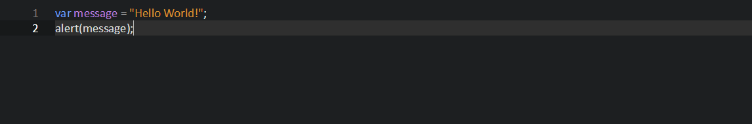
Второй способ: функция alert

Она не входит в пространство имен document, как writeln, поэтому его вписывать не нужно. Alert может отлично подойти для отладки кода веб-страницы. Его можно использовать и для взаимодействия с пользователем, но делать это не рекомендуется.

Alert имеет стандартный синтаксис любой JavaScript функции. Сначала записываем команду alert, в скобках записываем значение или переменной.

После этого в верхней части страницы всплывет окно с сообщением для пользователя, которое содержит кнопку «OK».

Данная функция может иметь множество применений, но не стоит злоупотреблять ей при работе с веб-сайтом, так как всплывающие окна могут вывести пользователя из себя. Кроме того, некоторые плагины блокируют функции типа alert.

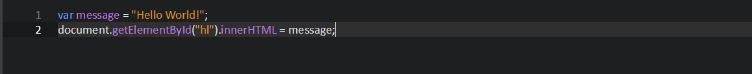


Третий способ: функция getElementById

Это самый сложный и наиболее популярный способ вывода информации на экран. С его помощью вы сможете изменять текстовый контент вашей HTML страницы.

Как мы знаем — все HTML теги могут иметь идентификаторы. Именно к ним и обращается наша функция. GetElementById находит элемент по его ID, изменяет содержимое тега по желанию, при этом остальной контент остается без изменений.

Данная функция входит в пространство имен document, что требует указывать его перед самой функцией.



Как вы могли заметить, кроме getElementById присутствует непонятная команда innerHTML, после него идет знак равенства и текстовая строка. Так вот, эта команда, вместе со знаком равенства, находит определенный тег и «приравнивает» его содержимое строке в кавычках.

Данный метод лучше всего подходит для вставки текста через JavaScript, поскольку он имеет целый ряд преимуществ:

Отсутствие всплывающих окон при работе функции.

Функция работает только с HTML-элементами.

Текст в кавычках может быть обернут в теги. Таким образом можно написать страницу на JavaScript с нуля.

Заключение

В этой статье мы поговорили с вами о том, как выводить информацию на экран. Это можно сделать с помощью сообщений, а также меняя внутренний контент веб-страницы. Мы лишь немного затронули пространство имен document, которое имеет целый ряд функция для работы с веб-страницей. В следующих уроках мы ознакомимся с ними детально.

Массивы

Объекты позволяют хранить данные со строковыми ключами. Это замечательно.

Но довольно часто мы понимаем, что нам необходима упорядоченная коллекция данных, в которой присутствуют 1-й, 2-й, 3-й элементы и т.д. Например, она понадобится нам для хранения списка чего-либо: пользователей, товаров, элементов HTML и т.д.

В этом случае использовать объект неудобно, так как он не предоставляет методов управления порядком элементов. Мы не можем вставить новое свойство «между» существующими. Объекты просто не предназначены для этих целей.

Для хранения упорядоченных коллекций существует особая структура данных, которая называется массив, Array.

Объявление

Существует два варианта синтаксиса для создания пустого массива:

let arr = new Array();

let arr = [];

Практически всегда используется второй вариант синтаксиса. В скобках мы можем указать начальные значения элементов:

let fruits = ["Яблоко", "Апельсин", "Слива"];

Элементы массива нумеруются, начиная с нуля.

Мы можем получить элемент, указав его номер в квадратных скобках:

let fruits = ["Яблоко", "Апельсин", "Слива"];

alert( fruits[0] ); // Яблоко

alert( fruits[1] ); // Апельсин

alert( fruits[2] ); // Слива

Мы можем заменить элемент:

fruits[2] = 'Груша'; // теперь ["Яблоко", "Апельсин", "Груша"]

…Или добавить новый к существующему массиву:

fruits[3] = 'Лимон'; // теперь ["Яблоко", "Апельсин", "Груша", "Лимон"]

Общее число элементов массива содержится в его свойстве length:

let fruits = ["Яблоко", "Апельсин", "Слива"];

alert( fruits.length ); // 3

Вывести массив целиком можно при помощи alert.

let fruits = ["Яблоко", "Апельсин", "Слива"];

alert( fruits ); // Яблоко, Апельсин, Слива

В массиве могут храниться элементы любого типа.

Например:

// разные типы значений

let arr = [ 'Яблоко', { name: 'Джон' }, true, function() { alert('привет'); } ];

// получить элемент с индексом 1 (объект) и затем показать его свойство

alert( arr[1].name ); // Джон

// получить элемент с индексом 3 (функция) и выполнить её

arr[3](); // привет

Висячая запятая

Список элементов массива, как и список свойств объекта, может оканчиваться запятой:

let fruits = [

"Яблоко",

"Апельсин",

"Слива",

];

«Висячая запятая» упрощает процесс добавления/удаления элементов, так как все строки становятся идентичными.

Получение последних элементов при помощи «at»

Новая возможность

Эта возможность была добавлена в язык недавно. В старых браузерах может понадобиться полифил.

Допустим, нам нужен последний элемент массива.

Некоторые языки программирования позволяют использовать отрицательные индексы для той же цели, как-то так: fruits[-1].

Однако, в JavaScript такая запись не сработает. Её результатом будет undefined, поскольку индекс в квадратных скобках понимается буквально.

Мы можем явно вычислить индекс последнего элемента, а затем получить к нему доступ вот так: fruits[fruits.length - 1].

let fruits = ["Apple", "Orange", "Plum"];

alert( fruits[fruits.length-1] ); // Plum

Немного громоздко, не так ли? Нам нужно дважды написать имя переменной.

К счастью, есть более короткий синтаксис: fruits.at (-1):

let fruits = ["Apple", "Orange", "Plum"];

// то же самое, что и fruits[fruits.length-1]

alert( fruits.at(-1) ); // Plum

Другими словами, arr.at(i):

это ровно то же самое, что и arr[i], если i >= 0.

для отрицательных значений i, он отступает от конца массива.

Методы pop/push, shift/unshift

Очередь – один из самых распространённых вариантов применения массива. В области компьютерных наук так называется упорядоченная коллекция элементов, поддерживающая два вида операций:

push добавляет элемент в конец.

shift удаляет элемент в начале, сдвигая очередь, так что второй элемент становится первым.

Массивы поддерживают обе операции.

На практике необходимость в этом возникает очень часто. Например, очередь сообщений, которые надо показать на экране.

Существует и другой вариант применения для массивов – структура данных, называемая стек.

Она поддерживает два вида операций:

push добавляет элемент в конец.

pop удаляет последний элемент.

Таким образом, новые элементы всегда добавляются или удаляются из «конца».

Примером стека обычно служит колода карт: новые карты кладутся наверх и берутся тоже сверху:

Массивы в JavaScript могут работать и как очередь, и как стек. Мы можем добавлять/удалять элементы как в начало, так и в конец массива.

В компьютерных науках структура данных, делающая это возможным, называется двусторонняя очередь.

Методы, работающие с концом массива:

pop

Удаляет последний элемент из массива и возвращает его:

let fruits = ["Яблоко", "Апельсин", "Груша"];

alert( fruits.pop() ); // удаляем "Груша" и выводим его

alert( fruits ); // Яблоко, Апельсин

И fruits.pop() и fruits.at(-1) возвращают последний элемент массива, но fruits.pop() также изменяет массив, удаляя его.

push

Добавляет элемент в конец массива:

let fruits = ["Яблоко", "Апельсин"];

fruits.push("Груша");

alert( fruits ); // Яблоко, Апельсин, Груша

Вызов fruits.push(...) равнозначен fruits[fruits.length] = ....

Методы, работающие с началом массива:

shift

Удаляет из массива первый элемент и возвращает его:

let fruits = ["Яблоко", "Апельсин", "Груша"];

alert( fruits.shift() ); // удаляем Яблоко и выводим его

alert( fruits ); // Апельсин, Груша

unshift

Добавляет элемент в начало массива:

let fruits = ["Апельсин", "Груша"];

fruits.unshift('Яблоко');

alert( fruits ); // Яблоко, Апельсин, Груша

Методы push и unshift могут добавлять сразу несколько элементов:

let fruits = ["Яблоко"];

fruits.push("Апельсин", "Груша");

fruits.unshift("Ананас", "Лимон");

// ["Ананас", "Лимон", "Яблоко", "Апельсин", "Груша"]

alert( fruits );

Внутреннее устройство массива

Массив – это особый подвид объектов. Квадратные скобки, используемые для того, чтобы получить доступ к свойству arr[0] – это по сути обычный синтаксис доступа по ключу, как obj[key], где в роли obj у нас arr, а в качестве ключа – числовой индекс.

Массивы расширяют объекты, так как предусматривают специальные методы для работы с упорядоченными коллекциями данных, а также свойство length. Но в основе всё равно лежит объект.

Следует помнить, что в JavaScript существует 8 основных типов данных. Массив является объектом и, следовательно, ведёт себя как объект.

Например, копируется по ссылке:

let fruits = ["Банан"]

let arr = fruits; // копируется по ссылке (две переменные ссылаются на один и тот же массив)

alert( arr === fruits ); // true

arr.push("Груша"); // массив меняется по ссылке

alert( fruits ); // Банан, Груша - теперь два элемента

…Но то, что действительно делает массивы особенными – это их внутреннее представление. Движок JavaScript старается хранить элементы массива в непрерывной области памяти, один за другим, так, как это показано на иллюстрациях к этой главе. Существуют и другие способы оптимизации, благодаря которым массивы работают очень быстро.

Но все они утратят эффективность, если мы перестанем работать с массивом как с «упорядоченной коллекцией данных» и начнём использовать его как обычный объект.

Например, технически мы можем сделать следующее:

let fruits = []; // создаём массив

fruits[99999] = 5; // создаём свойство с индексом, намного превышающим длину массива

fruits.age = 25; // создаём свойство с произвольным именем

Это возможно, потому что в основе массива лежит объект. Мы можем присвоить ему любые свойства.

Но движок поймёт, что мы работаем с массивом, как с обычным объектом. Способы оптимизации, используемые для массивов, в этом случае не подходят, поэтому они будут отключены и никакой выгоды не принесут.

Варианты неправильного применения массива:

Добавление нечислового свойства, например: arr.test = 5.

Создание «дыр», например: добавление arr[0], затем arr[1000] (между ними ничего нет).

Заполнение массива в обратном порядке, например: arr[1000], arr[999] и т.д.

Массив следует считать особой структурой, позволяющей работать с упорядоченными данными. Для этого массивы предоставляют специальные методы. Массивы тщательно настроены в движках JavaScript для работы с однотипными упорядоченными данными, поэтому, пожалуйста, используйте их именно в таких случаях. Если вам нужны произвольные ключи, вполне возможно, лучше подойдёт обычный объект {}.

Эффективность

Методы push/pop выполняются быстро, а методы shift/unshift – медленно.

Почему работать с концом массива быстрее, чем с его началом? Давайте посмотрим, что происходит во время выполнения:

fruits.shift(); // удаляем первый элемент с начала

Просто взять и удалить элемент с номером 0 недостаточно. Нужно также заново пронумеровать остальные элементы.

Операция shift должна выполнить 3 действия:

Удалить элемент с индексом 0.

Сдвинуть все элементы влево, заново пронумеровать их, заменив 1 на 0, 2 на 1 и т.д.

Обновить свойство length .

Чем больше элементов содержит массив, тем больше времени потребуется для того, чтобы их переместить, больше операций с памятью.

То же самое происходит с unshift: чтобы добавить элемент в начало массива, нам нужно сначала сдвинуть существующие элементы вправо, увеличивая их индексы.

А что же с push/pop? Им не нужно ничего перемещать. Чтобы удалить элемент в конце массива, метод pop очищает индекс и уменьшает значение length.

Действия при операции pop:

fruits.pop(); // удаляем один элемент с конца

Метод pop не требует перемещения, потому что остальные элементы остаются с теми же индексами. Именно поэтому он выполняется очень быстро.

Аналогично работает метод push.

Перебор элементов

Одним из самых старых способов перебора элементов массива является цикл for по цифровым индексам:

let arr = ["Яблоко", "Апельсин", "Груша"];

for (let i = 0; i < arr.length; i++) {

alert( arr[i] );

}

Но для массивов возможен и другой вариант цикла, for..of:

let fruits = ["Яблоко", "Апельсин", "Слива"];

// проходит по значениям

for (let fruit of fruits) {

alert( fruit );

}

Цикл for..of не предоставляет доступа к номеру текущего элемента, только к его значению, но в большинстве случаев этого достаточно. А также это короче.

Технически, так как массив является объектом, можно использовать и вариант for..in:

let arr = ["Яблоко", "Апельсин", "Груша"];

for (let key in arr) {

alert( arr[key] ); // Яблоко, Апельсин, Груша

}

Но на самом деле это – плохая идея. Существуют скрытые недостатки этого способа:

Цикл for..in выполняет перебор всех свойств объекта, а не только цифровых.

В браузере и других программных средах также существуют так называемые «псевдомассивы» – объекты, которые выглядят, как массив. То есть, у них есть свойство length и индексы, но они также могут иметь дополнительные нечисловые свойства и методы, которые нам обычно не нужны. Тем не менее, цикл for..in выведет и их. Поэтому, если нам приходится иметь дело с объектами, похожими на массив, такие «лишние» свойства могут стать проблемой.

Цикл for..in оптимизирован под произвольные объекты, не массивы, и поэтому в 10-100 раз медленнее. Увеличение скорости выполнения может иметь значение только при возникновении узких мест. Но мы всё же должны представлять разницу.

В общем, не следует использовать цикл for..in для массивов.

Немного о «length»

Свойство length автоматически обновляется при изменении массива. Если быть точными, это не количество элементов массива, а наибольший цифровой индекс плюс один.

Например, единственный элемент, имеющий большой индекс, даёт большую длину:

let fruits = [];

fruits[123] = "Яблоко";

alert( fruits.length ); // 124

Обратите внимание, что обычно мы не используем массивы таким образом.

Ещё один интересный факт о свойстве length – его можно перезаписать.

Если мы вручную увеличим его, ничего интересного не произойдёт. Зато, если мы уменьшим его, массив станет короче. Этот процесс необратим, как мы можем понять из примера:

let arr = [1, 2, 3, 4, 5];

arr.length = 2; // укорачиваем до двух элементов

alert( arr ); // [1, 2]

arr.length = 5; // возвращаем length как было

alert( arr[3] ); // undefined: значения не восстановились

Таким образом, самый простой способ очистить массив – это arr.length = 0;.

new Array()

Существует ещё один вариант синтаксиса для создания массива:

let arr = new Array("Яблоко", "Груша", "и тд");

Он редко применяется, так как квадратные скобки [] короче. Кроме того, у него есть хитрая особенность.

Если new Array вызывается с одним аргументом, который представляет собой число, он создаёт массив без элементов, но с заданной длиной.

Давайте посмотрим, как можно оказать себе медвежью услугу:

let arr = new Array(2); // создастся ли массив [2]?

alert( arr[0] ); // undefined! нет элементов.

alert( arr.length ); // length 2

Как мы видим, в коде, представленном выше, в new Array(number) все элементы равны undefined.

Чтобы избежать появления таких неожиданных ситуаций, мы обычно используем квадратные скобки, если, конечно, не знаем точно, что по какой-то причине нужен именно Array.

Многомерные массивы

Массивы могут содержать элементы, которые тоже являются массивами. Это можно использовать для создания многомерных массивов, например, для хранения матриц:

let matrix = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

];

alert( matrix[1][1] ); // 5, центральный элемент

toString

Массивы по-своему реализуют метод toString, который возвращает список элементов, разделённых запятыми.

Например:

let arr = [1, 2, 3];

alert( arr ); // 1,2,3

alert( String(arr) === '1,2,3' ); // true

Давайте теперь попробуем следующее:

alert( [] + 1 ); // "1"

alert( [1] + 1 ); // "11"

alert( [1,2] + 1 ); // "1,21"

Массивы не имеют ни Symbol.toPrimitive, ни функционирующего valueOf, они реализуют только преобразование toString, таким образом, здесь [] становится пустой строкой, [1] становится "1", а [1,2] становится "1,2".

Когда бинарный оператор плюс "+" добавляет что-либо к строке, он тоже преобразует это в строку, таким образом:

alert( "" + 1 ); // "1"

alert( "1" + 1 ); // "11"

alert( "1,2" + 1 ); // "1,21"

Не сравнивайте массивы при помощи ==

В JavaScript, в отличие от некоторых других языков программирования, массивы не следует сравнивать при помощи оператора ==.

У этого оператора нет специального подхода к массивам, он работает с ними, как и с любыми другими объектами.

Давайте ещё раз напомним правила:

Два объекта равны друг другу == только в том случае, если они ссылаются на один и тот же объект.

Если один из аргументов == является объектом, а другой – примитивом, то объект преобразуется в примитив, как описано в главе Преобразование объектов в примитивы.

…За исключением null и undefined, которые равны == друг другу и ничему больше.

Оператор строгого равенства === ещё проще, так как он не преобразует типы.

Итак, если мы всё же сравниваем массивы с помощью ==, то они никогда не будут одинаковыми, если только мы не сравним две переменные, которые ссылаются на один и тот же массив

Например:

alert( [] == [] ); // false

alert( [0] == [0] ); // false

Технически эти массивы являются разными объектами. Так что они не равны. Оператор == не выполняет поэлементное сравнение.

Сравнение с примитивами также может дать, казалось бы, странные результаты:

alert( 0 == [] ); // true

alert('0' == [] ); // false

Здесь, в обоих случаях, мы сравниваем примитив с объектом массива. Таким образом, массив [] преобразуется в примитив с целью сравнения и становится пустой строкой ''.

Затем продолжается процесс сравнения с примитивами, как описано в главе Преобразование типов:

// после того, как [] был преобразован в ''

alert( 0 == '' ); // true, так как '' преобразуется в число 0

alert('0' == '' ); // false, нет преобразования типов, разные строки

Так как же сравнить массивы?

Это просто: не используйте оператор ==. Вместо этого сравните их поэлементам в цикле или используя методы итерации, описанные в следующей главе.

Итого

Массив – это особый тип объекта, предназначенный для работы с упорядоченным набором элементов.

Объявление:

// квадратные скобки (обычно)

let arr = [item1, item2...];

// new Array (очень редко)

let arr = new Array(item1, item2...);

Вызов new Array(number) создаёт массив с заданной длиной, но без элементов.

Свойство length отражает длину массива или, если точнее, его последний цифровой индекс плюс один. Длина корректируется автоматически методами массива.

Если мы уменьшаем length вручную, массив укорачивается.

Получение элементов:

Мы можем получить элемент по его индексу, например arr[0].

Также мы можем использовать метод at(i) для получения элементов с отрицательным индексом, для отрицательных значений i, он отступает от конца массива. В остальном он работает так же, как arr[i], если i >= 0.

Мы можем использовать массив как двустороннюю очередь, используя следующие операции:

push(...items)добавляет items в конец массива.

pop() удаляет элемент в конце массива и возвращает его.

shift() удаляет элемент в начале массива и возвращает его.

unshift(...items) добавляет items в начало массива.

Чтобы пройтись по элементам массива:

for (let i=0; i<arr.length; i++) – работает быстрее всего, совместим со старыми браузерами.

for (let item of arr) – современный синтаксис только для значений элементов (к индексам нет доступа).

for (let i in arr) – никогда не используйте для массивов!

Мы вернёмся к массивам и изучим другие методы добавления, удаления, выделения элементов и сортировки массивов в главе: Методы массивов.

Циклы while и for

При написании скриптов зачастую встаёт задача сделать однотипное действие много раз.

Например, вывести товары из списка один за другим. Или просто перебрать все числа от 1 до 10 и для каждого выполнить одинаковый код.

Для многократного повторения одного участка кода предусмотрены циклы.

Циклы for…of и for…in

Небольшое объявление для продвинутых читателей.

В этой статье рассматриваются только базовые циклы: while, do..while и for(..;..;..).

Если вы пришли к этой статье в поисках других типов циклов, вот указатели:

См. for…in для перебора свойств объекта.

См. for…of и Перебираемые объекты для перебора массивов и перебираемых объектов.

В противном случае, продолжайте читать.

Цикл «while»

Цикл while имеет следующий синтаксис:

while (condition) {

// код

// также называемый "телом цикла"

}

Код из тела цикла выполняется, пока условие condition истинно.

Например, цикл ниже выводит i, пока i < 3:

let i = 0;

while (i < 3) { // выводит 0, затем 1, затем 2

alert( i );

i++;

}

Одно выполнение тела цикла по-научному называется итерация. Цикл в примере выше совершает три итерации.

Если бы строка i++ отсутствовала в примере выше, то цикл бы повторялся (в теории) вечно. На практике, конечно, браузер не позволит такому случиться, он предоставит пользователю возможность остановить «подвисший» скрипт, а JavaScript на стороне сервера придётся «убить» процесс.

Любое выражение или переменная может быть условием цикла, а не только сравнение: условие while вычисляется и преобразуется в логическое значение.

Например, while (i) – более краткий вариант while (i != 0):

let i = 3;

while (i) { // когда i будет равно 0, условие станет ложным, и цикл остановится

alert( i );

i--;

}

Фигурные скобки не требуются для тела цикла из одной строки

Если тело цикла состоит лишь из одной инструкции, мы можем опустить фигурные скобки {…}:

let i = 3;

while (i) alert(i--);

Цикл «do…while»

Проверку условия можно разместить под телом цикла, используя специальный синтаксис do..while:

do {

// тело цикла

} while (condition);

Цикл сначала выполнит тело, а затем проверит условие condition, и пока его значение равно true, он будет выполняться снова и снова.

Например:

let i = 0;

do {

alert( i );

i++;

} while (i < 3);

Такая форма синтаксиса оправдана, если вы хотите, чтобы тело цикла выполнилось хотя бы один раз, даже если условие окажется ложным. На практике чаще используется форма с предусловием: while(…) {…}.

Цикл «for»

Более сложный, но при этом самый распространённый цикл — цикл for.

Выглядит он так:

for (начало; условие; шаг) {

// ... тело цикла ...

}

Давайте разберёмся, что означает каждая часть, на примере. Цикл ниже выполняет alert(i) для i от 0 до (но не включая) 3:

for (let i = 0; i < 3; i++) { // выведет 0, затем 1, затем 2

alert(i);

}

Рассмотрим конструкцию for подробней:

| часть |  |  |
| --- | --- | --- |
| начало | let i = 0 | Выполняется один раз при входе в цикл |
| условие | i < 3 | Проверяется перед каждой итерацией цикла. Если оно вычислится в false, цикл остановится. |
| тело | alert(i) | Выполняется снова и снова, пока условие вычисляется в true. |
| шаг | i++ | Выполняется после тела цикла на каждой итерации перед проверкой условия. |

В целом, алгоритм работы цикла выглядит следующим образом:

Выполнить начало

→ (Если условие == true → Выполнить тело, Выполнить шаг)

→ (Если условие == true → Выполнить тело, Выполнить шаг)

→ (Если условие == true → Выполнить тело, Выполнить шаг)

→ ...

То есть, начало выполняется один раз, а затем каждая итерация заключается в проверке условия, после которой выполняется тело и шаг.

Если тема циклов для вас нова, может быть полезным вернуться к примеру выше и воспроизвести его работу на листе бумаги, шаг за шагом.

Вот в точности то, что происходит в нашем случае:

// for (let i = 0; i < 3; i++) alert(i)

// Выполнить начало

let i = 0;

// Если условие == true → Выполнить тело, Выполнить шаг

if (i < 3) { alert(i); i++ }

// Если условие == true → Выполнить тело, Выполнить шаг

if (i < 3) { alert(i); i++ }

// Если условие == true → Выполнить тело, Выполнить шаг

if (i < 3) { alert(i); i++ }

// ...конец, потому что теперь i == 3

Встроенное объявление переменной

В примере переменная счётчика i была объявлена прямо в цикле. Это так называемое «встроенное» объявление переменной. Такие переменные существуют только внутри цикла.

for (let i = 0; i < 3; i++) {

alert(i); // 0, 1, 2

}

alert(i); // ошибка, нет такой переменной

Вместо объявления новой переменной мы можем использовать уже существующую:

let i = 0;

for (i = 0; i < 3; i++) { // используем существующую переменную

alert(i); // 0, 1, 2

}

alert(i); // 3, переменная доступна, т.к. была объявлена снаружи цикла

Пропуск частей «for»

Любая часть for может быть пропущена.

Для примера, мы можем пропустить начало если нам ничего не нужно делать перед стартом цикла.

Вот так:

let i = 0; // мы уже имеем объявленную i с присвоенным значением

for (; i < 3; i++) { // нет необходимости в "начале"

alert( i ); // 0, 1, 2

}

Можно убрать и шаг:

let i = 0;

for (; i < 3;) {

alert( i++ );

}

Это сделает цикл аналогичным while (i < 3).

А можно и вообще убрать всё, получив бесконечный цикл:

for (;;) {

// будет выполняться вечно

}

При этом сами точки с запятой ; обязательно должны присутствовать, иначе будет ошибка синтаксиса.

Прерывание цикла: «break»

Обычно цикл завершается при вычислении условия в false.

Но мы можем выйти из цикла в любой момент с помощью специальной директивы break.

Например, следующий код подсчитывает сумму вводимых чисел до тех пор, пока посетитель их вводит, а затем – выдаёт:

let sum = 0;

while (true) {

let value = +prompt("Введите число", '');

if (!value) break; // (\*)

sum += value;

}

alert( 'Сумма: ' + sum );

Директива break в строке (\*) полностью прекращает выполнение цикла и передаёт управление на строку за его телом, то есть на alert.

Вообще, сочетание «бесконечный цикл + break» – отличная штука для тех ситуаций, когда условие, по которому нужно прерваться, находится не в начале или конце цикла, а посередине или даже в нескольких местах его тела.

Переход к следующей итерации: continue

Директива continue – «облегчённая версия» break. При её выполнении цикл не прерывается, а переходит к следующей итерации (если условие все ещё равно true).

Её используют, если понятно, что на текущем повторе цикла делать больше нечего.

Например, цикл ниже использует continue, чтобы выводить только нечётные значения:

for (let i = 0; i < 10; i++) {

// если true, пропустить оставшуюся часть тела цикла

if (i % 2 == 0) continue;

alert(i); // 1, затем 3, 5, 7, 9

}

Для чётных значений i, директива continue прекращает выполнение тела цикла и передаёт управление на следующую итерацию for (со следующим числом). Таким образом alert вызывается только для нечётных значений.

Директива continue позволяет избегать вложенности

Цикл, который обрабатывает только нечётные значения, мог бы выглядеть так:

for (let i = 0; i < 10; i++) {

if (i % 2) {

alert( i );

}

}

С технической точки зрения он полностью идентичен. Действительно, вместо continue можно просто завернуть действия в блок if.

Однако мы получили дополнительный уровень вложенности фигурных скобок. Если код внутри if более длинный, то это ухудшает читаемость, в отличие от варианта с continue.

Нельзя использовать break/continue справа от оператора „?“

Обратите внимание, что эти синтаксические конструкции не являются выражениями и не могут быть использованы с тернарным оператором ?. В частности, использование таких директив, как break/continue, вызовет ошибку.

Например, если мы возьмём этот код:

if (i > 5) {

alert(i);

} else {

continue;

}

…и перепишем его, используя вопросительный знак:

(i > 5) ? alert(i) : continue; // continue здесь приведёт к ошибке

…то будет синтаксическая ошибка.

Это ещё один повод не использовать оператор вопросительного знака ? вместо if.

Метки для break/continue

Бывает, нужно выйти одновременно из нескольких уровней цикла сразу.

Например, в коде ниже мы проходимся циклами по i и j, запрашивая с помощью prompt координаты (i, j) с (0,0) до (2,2):

for (let i = 0; i < 3; i++) {

for (let j = 0; j < 3; j++) {

let input = prompt(`Значение на координатах (${i},${j})`, '');

// Что если мы захотим перейти к Готово (ниже) прямо отсюда?

}

}

alert('Готово!');

Нам нужен способ остановить выполнение, если пользователь отменит ввод.

Обычный break после input лишь прервёт внутренний цикл, но этого недостаточно. Достичь желаемого поведения можно с помощью меток.

Метка имеет вид идентификатора с двоеточием перед циклом:

labelName: for (...) {

...

}

Вызов break <labelName> в цикле ниже ищет ближайший внешний цикл с такой меткой и переходит в его конец.

outer: for (let i = 0; i < 3; i++) {

for (let j = 0; j < 3; j++) {

let input = prompt(`Значение на координатах (${i},${j})`, '');

// если пустая строка или Отмена, то выйти из обоих циклов

if (!input) break outer; // (\*)

// сделать что-нибудь со значениями...

}

}

alert('Готово!');

В примере выше это означает, что вызовом break outer будет разорван внешний цикл до метки с именем outer.

Таким образом управление перейдёт со строки, помеченной (\*), к alert('Готово!').

Можно размещать метку на отдельной строке:

outer:

for (let i = 0; i < 3; i++) { ... }

Директива continue также может быть использована с меткой. В этом случае управление перейдёт на следующую итерацию цикла с меткой.

Метки не позволяют «прыгнуть» куда угодно

Метки не дают возможности передавать управление в произвольное место кода.

Например, нет возможности сделать следующее:

break label; // не прыгает к метке ниже

label: for (...)

Директива break должна находиться внутри блока кода. Технически, подойдет любой маркированный блок кода, например:

label: {

// ...

break label; // работает

// ...

}

…Хотя в 99.9% случаев break используется внутри циклов, как мы видели в примерах выше.

К слову, continue возможно только внутри цикла.

Итого

Мы рассмотрели 3 вида циклов:

while – Проверяет условие перед каждой итерацией.

do..while – Проверяет условие после каждой итерации.

for (;;) – Проверяет условие перед каждой итерацией, есть возможность задать дополнительные настройки.

Чтобы организовать бесконечный цикл, используют конструкцию while (true). При этом он, как и любой другой цикл, может быть прерван директивой break.

Если на данной итерации цикла делать больше ничего не надо, но полностью прекращать цикл не следует – используют директиву continue.

Обе этих директивы поддерживают метки, которые ставятся перед циклом. Метки – единственный способ для break/continue выйти за пределы текущего цикла, повлиять на выполнение внешнего.

Заметим, что метки не позволяют прыгнуть в произвольное место кода, в JavaScript нет такой возможности.

Первое правило Junior Front-End разработчика, и разработчика вообще - это выполнение задач от руководства или заказчиков.  
Использование конкретных инструментов - jQuery, JavaScript, Angular, Brainf\*ck - задается требованиями конкретного руководителя или заказчика, либо остается на усмотрение разработчика (если пишем с нуля), который в любом случае смотрит на задачу, и никогда не определяется числом голосов в интернет-холиварах, где всегда одни за легкое и простое, другие за глубокое, универсальное, но тяжелое, и все друг друга ненавидят!  
  
Поэтому, чтобы понять куда двигаться дальше, необходимо изучить эти самые требования руководства и заказчиков, никто в здравом уме не будет против того, что вы лично поинтересуетесь, чем они занимаются и как к ним устроиться, а не ограничитесь текстом вакансии или "треда" на фриланс-бирже.

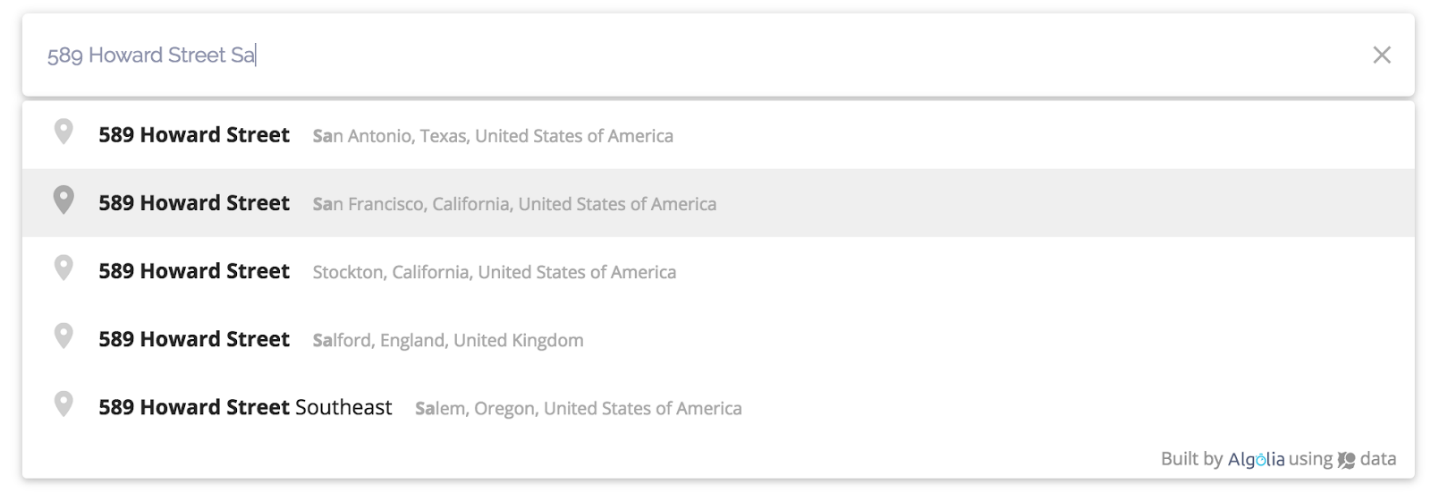
Доклад: java библиотеки

Сегодня мы рады выпустить Algolia Places, быстрый и простой способ превратить любой HTML-ввод в автозаполнение адреса.

Одной из наших основных миссий в Algolia является **улучшение пользовательского опыта, связанного с поиском**. Адресные формы занимали важное место в нашем списке, потому что их трудно создать и они часто представляют собой важный шаг в потоках пользователей, таких как регистрация и проверка. Наша цель с Algolia Places - предоставить что-то, что **легко интегрировать разработчикам** и **приятно использовать посетителям**. Он также легко настраивается и может быть обогащен дополнительными источниками данных.

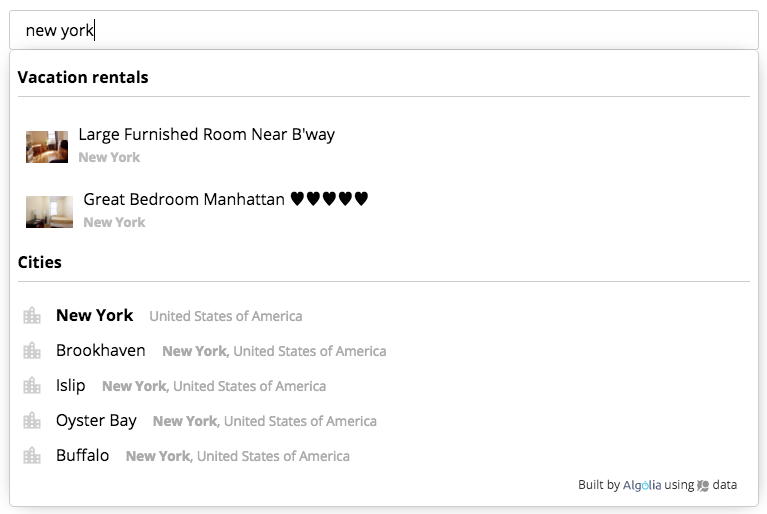
Algolia Places упакована как простая, легкая библиотека JavaScript и доступна на npm и jsDelivr. Вы можете найти исчерпывающую документацию и живые примеры на веб-сайте Algolia Places.

В Algolia мы стараемся создавать инструменты с разумным дизайном по умолчанию. Библиотека JavaScript Algolia Places отображает современное и чистое раскрывающееся меню с момента ее установки. Несколько строк CSS - это все, что вам нужно, чтобы настроить его в соответствии с точным внешним видом вашего сайта. Все возможности api поиска Algolia по отношению к опечаткам и подсветке уже встроены.



Algolia Places совместима с библиотекой автозаполнения.js. Вы можете легко создавать интерфейсы, которые объединяют результаты адресного поиска с другими результатами автозаполнения.js источников данных, включая индексы Algolia. Вот пример. Раздел «Города» содержит результаты из раздела «Места», а раздел «Аренда для отдыха» содержит результаты из обычного индекса Algolia.

|  |
| --- |
| <input type="search" id="address-input" placeholder="Start typing an address..." /> |
|  | <script src="https://cdn.jsdelivr.net/places.js/1/places.min.js"></script> |
|  | <script> |
|  | var placesAutocomplete = places({ |
|  | container: document.querySelector('#address-input') |
|  | }); |
|  | </script> |



Algolia Places также совместима с мгновенным поиском.js, библиотекой для создания полностраничного поиска по мере использования. Дополнительные сведения