1. Подключиться к виртуальному столу:

Запустить приложение Tight VNC Viewer -> Ввести в поле Remote Host (**seat.demo:59XX**) -> Нажать кнопку «Connect»

1. Зайти на гит **http://gitea.demo:3000**, открыть репозиторий с данными проекта (логин: **userXX** пароль: **password**)
2. Создать свой репозиторий **repoXX** (ПОСТАВИТЬ ГАЛОЧКУ при создании Initialize Repository)
3. Сохранить ссылки на все репы (+ Паша и Олеся) в документ
4. Зайти на почту и скачать материалы

winget install -e --id Git.Git

cd (C:\Users\FEDPo\Desktop\new путьДоПапки)

git clone https://github.com/PavelIsProgrammer/demo.git

1. Создать свою папку **examXX**
2. Создать проект Приложение WPF(.NET Framework), подключить EntityFramework (управление пакетами NuGet)
3. Создать папки структуры проекта (Model, Resources, View)
4. Перенести код окон и страниц (создать страницы), стили в файле App.xaml
5. Сделать шаблон документа отладки, тест-кейса, диаграмм
6. Создать БД **dbXX** с одной таблицей Test с двумя полями: ID и Name. Заполнить две строчки данными. Связать с тестовым проектом

**[Ctrl]+[Shift]+[Alt]+[F]** – комбинация чтобы отобразить виртуальное окно удалённого доступа на весь экран

**Переименовать проект**

1. Переименовать общий главный файл проекта со всеми папками
2. Запустить программу. Переименовать в обозревателе решений Сам проект и Решение (первые 2 строчки): ПКМ Переименовать
3. Проект -> Свойства -> Приложение -> Имя сборки и Пространство имен ПЕРЕИМЕНОВАТЬ
4. -> Сведения о сборке -> Название и Продукт ПЕРЕИМЕНОВАТЬ -> ОК
5. СОХРАНИТЬ ВСЕ
6. Заходим в файл MainWindow.xaml.cs -> Находим namespace *Verfi* -> *Verfi* ПКМ Переименовать -> Enter -> СОХРАНИТЬ ВСЕ

**Инструкция к Блок-схемам**

СОХРАНЯЕМ И ИСХОДНИКИ И ПДФ terminator

**Инструкция к БД**

Подключиться к СУБД: Имя сервера: **mssql.demo** Логин: **userXX** Пароль: **password**

При подключении к серверу установить флажок «Доверять сертификату сервера»

* Скопировать чужой скрипт с измененным именем:

N'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\User22\_403\_Seqrity.mdf'

(Путь можно посмотреть: нажать на любую бд, Properties, Files, Path)

* Имя БД из бумажки: **dbXX**

Name nvarchar(150)

Date date

Birthday date 2002-11-05

Time time(0) 12:00:00

Phone char(19)

* Если Сохранение изменений не разрешено Saving changes is not permitted (Изменения требуют удаления и повторного создания таблиц):

Сервис->Параметры-Конструкторы->Запретить сохранение изменений, требующих повторного создания таблицы-> СНЯТЬ ГАЛОЧКУ

**Инструкция к Модели БД**

Папка Model -> Добавить класс -> Данные -> Модель ADO.NET EDM

Имя: Modelимя (ModelDrive)

-> Конструктор EF из базы данных

-> Создать соединение:

Источник данных: Microsoft SQL Server

Имя сервера: mssql.demo

Галочка Сертификат сервера

Проверка подлинности SQL Server

Имя пользователя и пароль из бумажки (Логин: userXX Пароль: password)

-> В окне мастер моделей ОБЯЗАТЕЛЬНО ВЫБИРАЕМ Да, включить конфиденциальные данные в строку подключения

Сохранить параметры соединения

Имя: DBимяEntities (DBDriveEntities)

-> таблицы dbo без sisdiagrams

Имя: NSимя (NSDrive)

**Инструкция к Коду**

F4 - свойства элемента

Меняем стили и логотип устанавливаем Source="путь к картинке" иконку ставим в свойствах.

Установить иконку приложения: Выбрать в файле XAML элемент Window. Перейти в панель «Свойства», вкладка «Общие», свойство «Icon». Из выпадающего списка выбираем «Resources.ico».

Пишем комментарии и try catch. Меняем имена переменных

**Инструкция к Отладчику**

* установка точки останова (клавиша F9);
* запуск приложения в режиме отладки в Visual Studio
* пошаговый режим работы отладчика (клавиша F10):

просмотр всплывающей подсказки для текущих значений локальных величин при наведении курсора мыши

**Инструкция к Excel**

Убрать пробелы: =СЖПРОБЕЛЫ(стобец1)

Добавить пробел перед текстом(\_П): =" "&A1

Разделить текст в 2 столбца(ID 101) : Данные -> текст по столбцам -> с разделителями -> пробел

ФИО сделать в один столбец(ФИ|О): =СЦЕПИТЬ(столбец1;столбец2)

Сделать без повторов и присвоить ID: Данные -> удалить дубликаты ->ctrl+H

Изображение выглядит как текст, снимок экрана, линия, Шрифт

Автоматически созданное описание

**Инструкция Тест-кейс**

Наименование: Описание:

Дата тестирования •27.05.2024

Приоритет тестирования (Низкий/Средний/ Высокий) •Средний

Название тестирования •Сортировка заявок по дате по убыванию

Шаги теста •1 Запустить приложение

2 Нажать ОК в сообщении об успешном соединении с базой данных

3 Нажать на выпадающий список «Сортировка по дате»

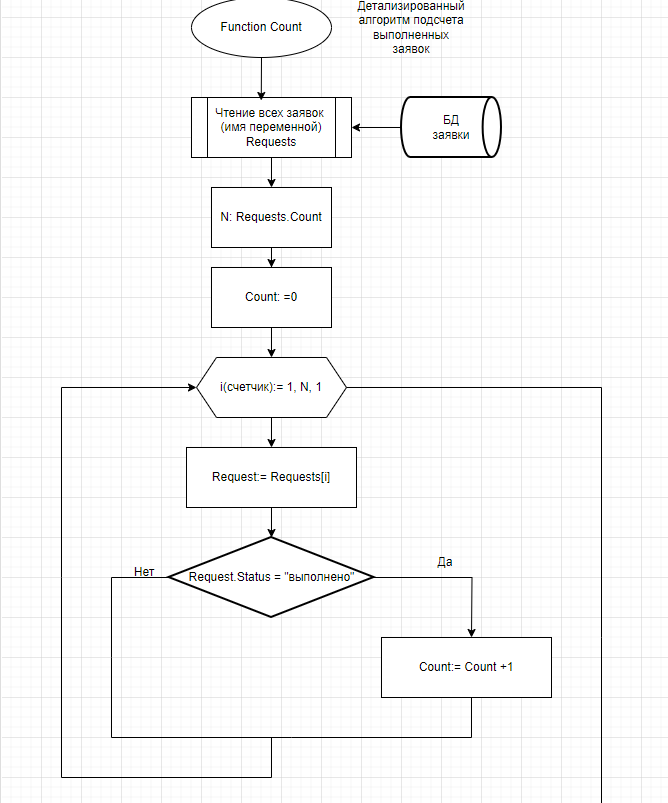
4 Выбрать из списка «По убыванию»

Тестовые данные •

Ожидаемый результат •Заявки успешно отсортированы по убыванию

Фактический результат •Заявки успешно отсортированы по убыванию

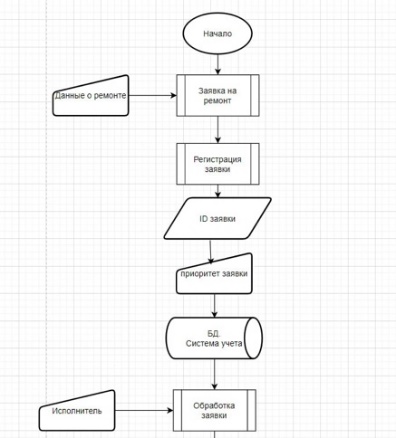
Статус (зачет/незачет) •Зачет



Изображение выглядит как диаграмма, линия, круг, График

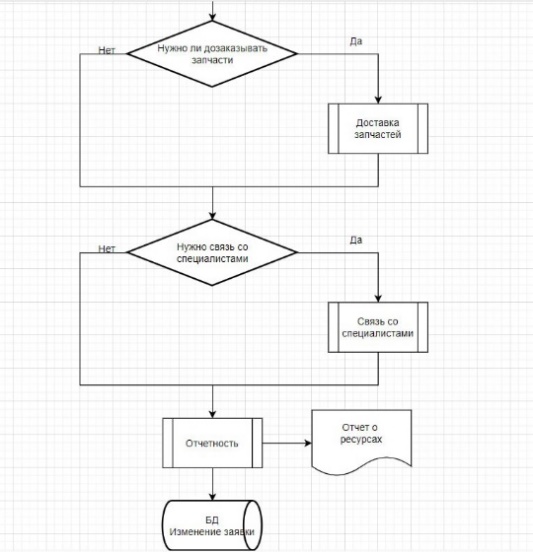
Автоматически созданное описание

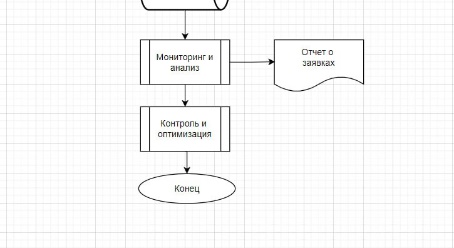
1. Блок-схема общая



Изображение выглядит как диаграмма, зарисовка, Технический чертеж, План

Автоматически созданное описание





1. Детальный алгоритм

Изображение выглядит как диаграмма, Технический чертеж, План, зарисовка

Автоматически созданное описание

Изображение выглядит как диаграмма, Технический чертеж, План, линия

Автоматически созданное описание

Изображение выглядит как текст, диаграмма, линия, Параллельный

Автоматически созданное описание

Изображение выглядит как диаграмма, линия, снимок экрана, круг

Автоматически созданное описание

1. Оформление работы с отладчиком:

* установка точки останова (клавиша F9);
* пошаговый режим работы отладчика (клавиша F10);
* просмотр всплывающей подсказки для текущих значений локальных величин при наведении курсора мыши;
* окно значений локальных переменных.

Установка точки прерывания

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, веб-страница

Автоматически созданное описание

Запуск приложения в режиме отладки в Visual Studio

Изображение выглядит как текст, программное обеспечение, веб-страница, Значок на компьютере

Автоматически созданное описание

Пошаговый режим работы отладчика: пошаговая отладка, во время данного процесса можно рассмотреть значения всех участвующих переменных и отследить выполнение команд на каждой строчке отдельно.

Просмотр всплывающей подсказки для текущих значений локальных величин при наведении курсора мыши

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, компьютер

Автоматически созданное описание

Пошаговая отладка

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, веб-страница

Автоматически созданное описание

1. Создание проекта

Выбрать тип приложения при создании проекта

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Подключить к проекту пакет Entity Framework из Интернета: источник пакета nuget.org. Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Структура проекта

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

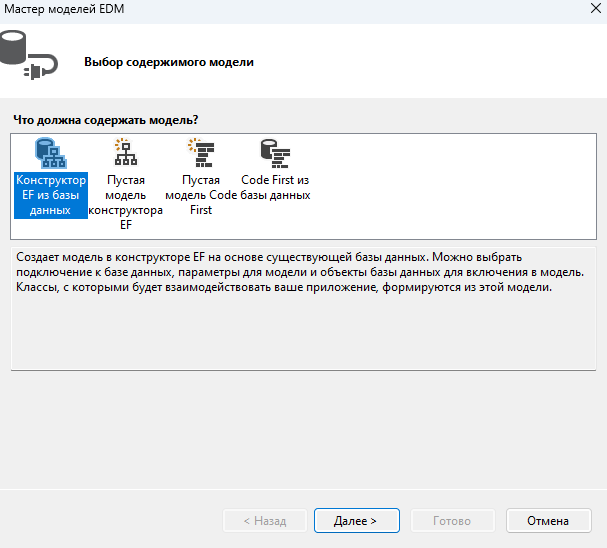
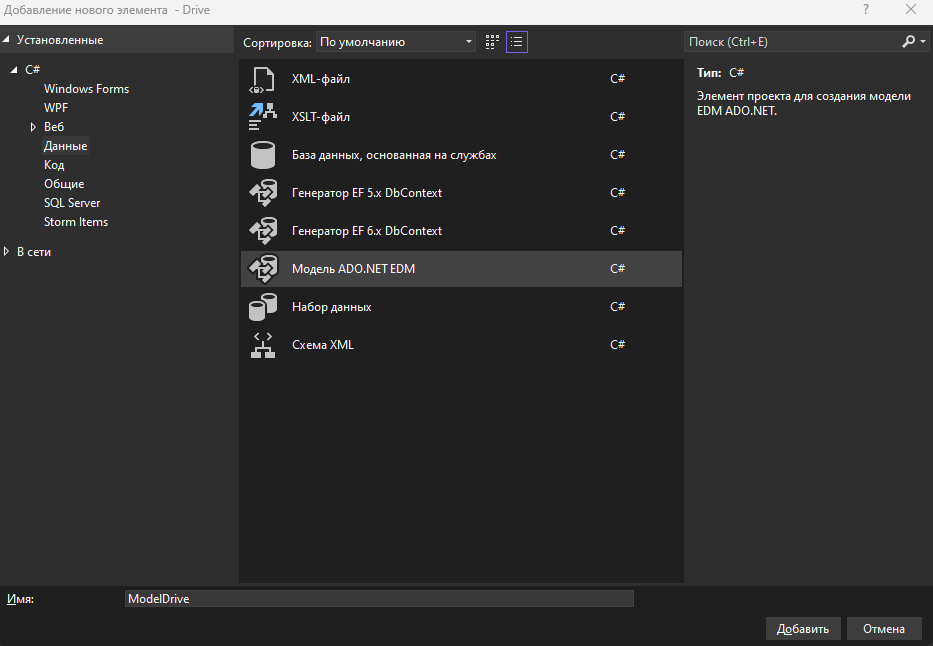
Установить иконку приложения. Выбрать в файле XAML элемент Window. Перейти в панель «Свойства», вкладка «Общие», свойство «Icon». Из выпадающего списка выбираем «Resources.ico».

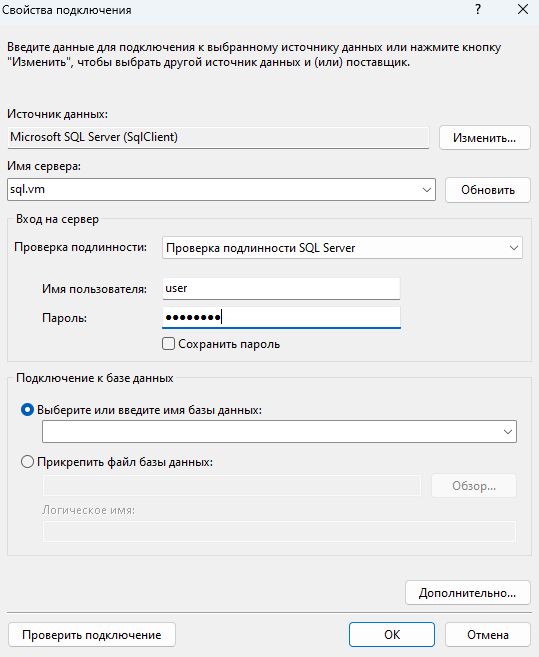
Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

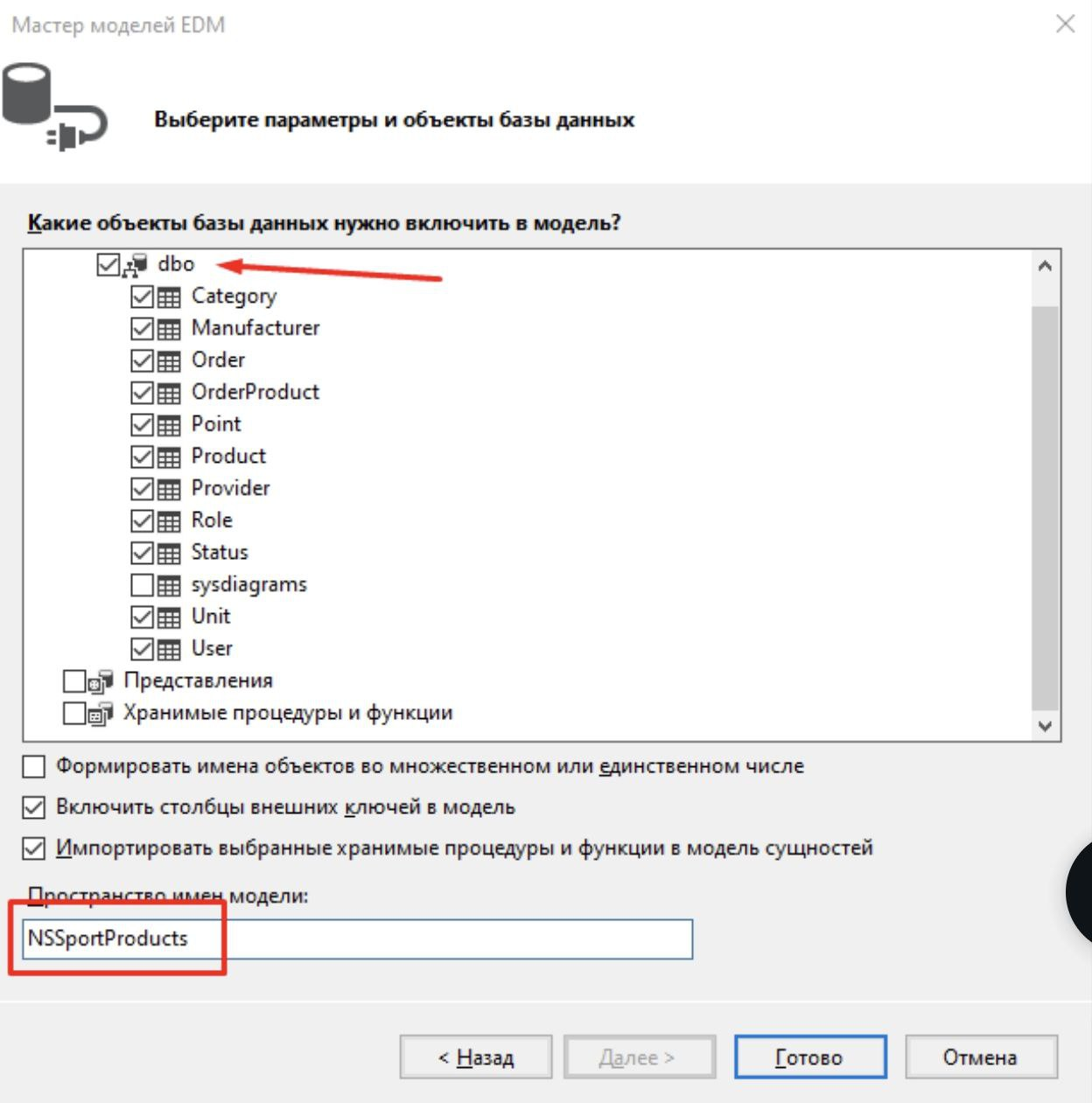
1. Создание модели БД

Model -> Добавить класс



Изображение выглядит как текст, снимок экрана, веб-страница, Веб-сайт

Автоматически созданное описание

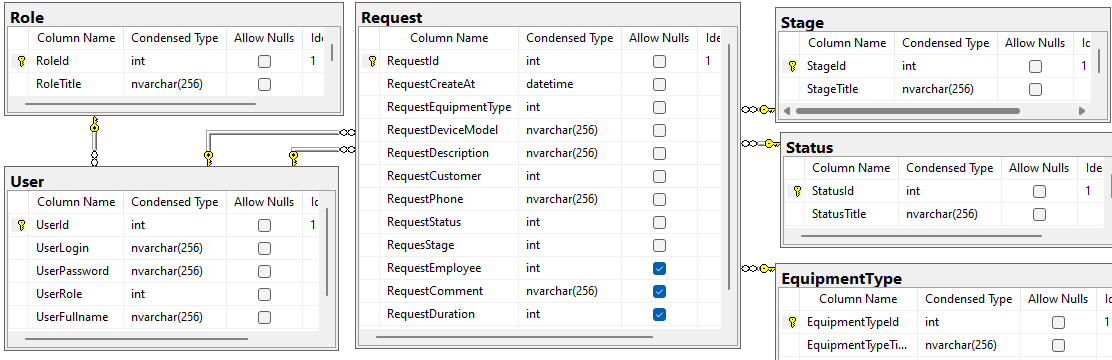


Изображение выглядит как снимок экрана, дизайн

Автоматически созданное описание

Изображение выглядит как снимок экрана

Автоматически созданное описание



Изображение выглядит как снимок экрана

Автоматически созданное описание

1. КОД:

* **App.xaml**

<Application x:Class="Drive.App"

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

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

xmlns:local="clr-namespace:Drive"

StartupUri="MainWindow.xaml">

<Application.Resources>

<Style TargetType="Button">

<Setter Property="FontFamily" Value="Comic Sans MS"/>

<Setter Property="FontSize" Value="20"/>

<Setter Property="Background" Value="#FF11C55A"/>

<Setter Property="Margin" Value="10"/>

</Style>

<Style TargetType="TextBlock">

<Setter Property="FontFamily" Value="Comic Sans MS"/>

<Setter Property="FontSize" Value="14"/>

<Setter Property="VerticalAlignment" Value="Center"/>

<Setter Property="HorizontalAlignment" Value="Center"/>

</Style>

</Application.Resources>

</Application>

* **App.xaml.cs**

namespace Drive

{

/// <summary>

/// Логика взаимодействия для App.xaml

/// </summary>

public partial class App : Application

{

public static Model.CondyEntities DB;

public static Model.User currentUser;

protected override void OnStartup(StartupEventArgs e)

{

base.OnStartup(e);

try

{

DB = new Model.CondyEntities();

}

catch (Exception)

{

MessageBox.Show("Ошибка! Не удалось подключиться к бд!");

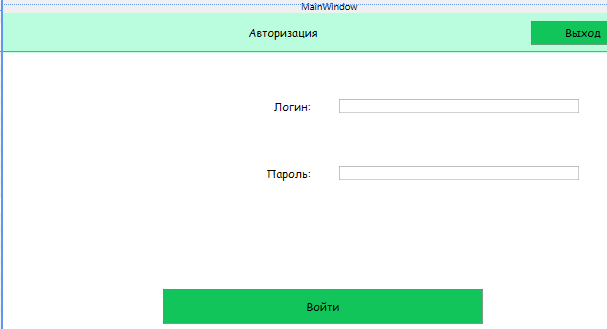
throw;

}

}

}

}



<Window x:Class="TestDemo.MainWindow"

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

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

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

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

xmlns:local="clr-namespace:TestDemo"

mc:Ignorable="d"

Title="MainWindow" Height="450" Width="800" MinHeight="450" MinWidth="800">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="50"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<!--Заголовок-->

<Grid Grid.Row="0" Background="#FFB9FDDE">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="50"/>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="150"/>

</Grid.ColumnDefinitions>

<Image Grid.Column="0" /> <!--Source="/Resources/Логотип.png"-->

<TextBlock Text="Авторизация" Grid.Column="1" HorizontalAlignment="Center"/>

<Button x:Name="btnExit" Content="Выход" Grid.Column="2" Click="btnExit\_Click" />

</Grid>

<Grid Grid.Row="1" x:Name="gridMain">

<Grid.RowDefinitions>

<RowDefinition Height="0.3\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="0.3\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<TextBlock Text="Логин:" Grid.Column="0" Grid.Row="1" VerticalAlignment="Center" HorizontalAlignment="Right" Margin="15" />

<TextBlock Text="Пароль:" Grid.Column="0" Grid.Row="2" VerticalAlignment="Center" HorizontalAlignment="Right" Margin="15"/>

<TextBox x:Name="tbLogin" Grid.Column="1" Grid.Row="1" VerticalAlignment="Center" HorizontalAlignment="Left" Margin="20,0,0,0" Width="300"/>

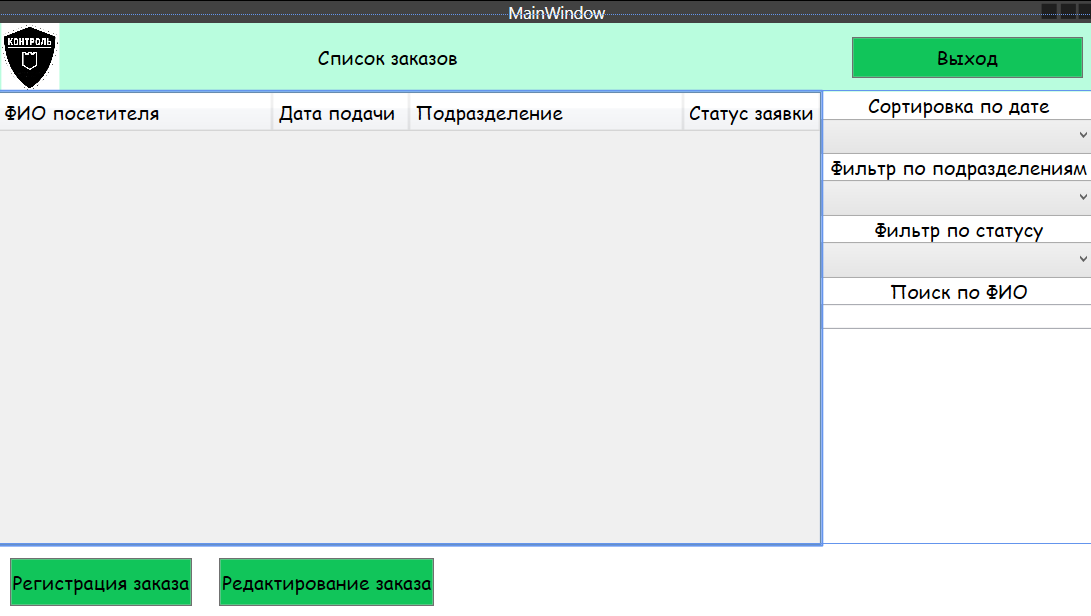
<PasswordBox Name="pbPassword" Grid.Row="2" Grid.Column="1" Margin="20,0,0,0" Width="300" VerticalAlignment="Center" HorizontalAlignment="Left"/>

<Button x:Name="btnSignIn" Grid.Row="4" Content="Войти" Grid.ColumnSpan="2" Margin="200, 20" Click="btnSignIn\_Click"/>

</Grid>

</Grid>

</Window>



Window x:Class="Drive.MainWindow"

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

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

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

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

xmlns:local="clr-namespace:Drive"

mc:Ignorable="d"

Title="MainWindow" Height="450" Width="800" Loaded="Window\_Loaded">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="50"/>

<RowDefinition Height="6\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<!--Заголовок-->

<Grid Grid.Row="0" Background="#FFB9FDDE">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="50"/>

<ColumnDefinition Width="95\*"/>

<ColumnDefinition Width="468\*"/>

<ColumnDefinition Width="188\*"/>

</Grid.ColumnDefinitions>

<Image Grid.Column="0" Source="/Resources/Логотип.png"/>

<TextBlock Text="Список заказов" Grid.Column="2" HorizontalAlignment="Left" Margin="89,0,0,0" />

<Button x:Name="btn\_exit" Content="Выход" Grid.Column="3" Click="btn\_exit\_Click" Margin="10,10,10,10"/>

</Grid>

<Grid Grid.Row="1">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="3\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<!--Таблица-->

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

<DataGrid.Columns>

<DataGridTextColumn Header="ФИО посетителя" Width="2\*" Binding="{Binding Visitor.VisitorFullName}"/>

<DataGridTextColumn Header="Дата подачи" Width="\*" Binding="{Binding RequestDateAdd, StringFormat=dd:MM:yyyy}"/>

<!--переменная со счетом end - start-->

<DataGridTextColumn Header="Количество дней(startEnd)" Width="\*" Binding="{Binding RequestQuantityDays}"/>

<DataGridTextColumn Header="Подразделение" Width="2\*" Binding="{Binding Subdivision.SubdivisionName}"/>

<DataGridTextColumn Header="Статус заявки" Width="\*" Binding="{Binding Status.StatusName}"/>

</DataGrid.Columns>

</DataGrid>

<!--Фильтры и поиск-->

<StackPanel Orientation="Vertical" Grid.Column="1">

<TextBlock Text="Сортировка по дате"/>

<ComboBox x:Name="cbSort" SelectionChanged="cbSort\_SelectionChanged">

<ComboBoxItem Content="Без сортировки"/>

<ComboBoxItem Content="По возрастанию"/>

<ComboBoxItem Content="По убыванию"/>

</ComboBox>

<TextBlock Text="Фильтр по подразделениям"/>

<ComboBox x:Name="cbDivision" SelectionChanged="cbDivision\_SelectionChanged">

</ComboBox>

<TextBlock Text="Фильтр по статусу"/>

<ComboBox x:Name="cbStatus" SelectionChanged="cbStatus\_SelectionChanged">

</ComboBox>

<TextBlock Text="Поиск по ФИО"/>

<TextBox x:Name="tbFullName" TextChanged="tbFullName\_TextChanged"/>

</StackPanel>

</Grid>

<!--Кнопки управления-->

<StackPanel Orientation="Horizontal" Grid.Row="2">

<Button x:Name="btnRegistration" Content="Регистрация заказа" Click="btnRegistration\_Click" />

<Button x:Name="btnEdit" Content="Редактирование заказа" Click="btnEdit\_Click" />

</StackPanel>

</Grid>

</Window>

**Код страницы Списка заявок**

namespace Drive

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

//связь с бд

try

{

App.DB = new Model.DBDriveEntities();

MessageBox.Show("Подключение к бд успешно");

}

catch

{

MessageBox.Show("Связь не установлена");

App.Current.Shutdown();

}

ShowOrders();

cbSort.SelectedIndex = 0; //изначально выбрано

List<Model.Subdivision> divisions = App.DB.Subdivision.ToList(); //для фильтра по подразделениям

Model.Subdivision division = new Model.Subdivision(); // новое подразделение для всех подразделений сразу (как без фильтра)

division.SubdivisionID = 0;

division.SubdivisionName = "Все подразделения";

cbDivision.DisplayMemberPath = "SubdivisionName"; //видимые значения

cbDivision.SelectedValuePath = "SubdivisionID"; //получаемые значения

cbDivision.ItemsSource = divisions;

divisions.Insert(0, division);

cbDivision.SelectedIndex = 0;

List<Model.Status> statuses = App.DB.Status.ToList();

Model.Status status = new Model.Status(); // новый статус

status.StatusID = 0;

status.StatusName = "Все статусы";

cbStatus.DisplayMemberPath = "StatusName"; //видимые значения

cbStatus.SelectedValuePath = "StatusID"; //получаемые значения

cbStatus.ItemsSource = statuses;

statuses.Insert(0, status);

cbStatus.SelectedIndex = 0;

}

private void btn\_exit\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

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

{

ShowOrders();

}

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

{

ShowOrders();

}

public void ShowOrders()

{

//получить список всех заявок из БД

List<Model.Request> requests = App.DB.Request.ToList(); //подключение к ТАБЛИЦЕ бд

//обработка заявок

//сортировка

if (cbSort.SelectedIndex == 1)

{

requests = requests.OrderBy(r => r.RequestDateAdd /\*по какому полю сортировать заказы\*/).ToList();

}

else if (cbSort.SelectedIndex == 2)

{

requests = requests.OrderByDescending(r => r.RequestDateAdd).ToList();

}

//фильтрация по подразделениям

if (cbDivision.SelectedIndex > 0)

{

requests = requests.Where(r => r.RequestSubdivision == (int)cbDivision.SelectedValue).ToList();

}

//фильтрация по статусу

if (cbStatus.SelectedIndex > 0)

{

requests = requests.Where(r => r.RequestStatus == (int)cbStatus.SelectedValue).ToList();

}

//поиск по фио

if (tbFullName.Text.Length != 0)

{

requests = requests.Where(r => r.Visitor.VisitorFullName.ToLower().Contains(tbFullName.Text.ToLower())).ToList();

}

//отобразить список отсортированных заявок в окне

dgRequest.ItemsSource = requests;

}

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

{

ShowOrders();

}

private void tbFullName\_TextChanged(object sender, TextChangedEventArgs e)

{

ShowOrders();

}

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

{

View.AddOrder viewAddOrder = new View.AddOrder();

this.Hide();

viewAddOrder.ShowDialog();

this.ShowDialog();

ShowOrders(); //обновление списка грида

}

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

{

if(dgRequest.SelectedItem != null)

{

View.EditOrder viewEditOrder = new View.EditOrder((Model.Request)dgRequest.SelectedItem);

this.Hide();

viewEditOrder.ShowDialog();

this.ShowDialog();

}

else

{

MessageBox.Show("Выберите запись");

}

}

private void Window\_IsVisibleChanged(object sender, DependencyPropertyChangedEventArgs e)

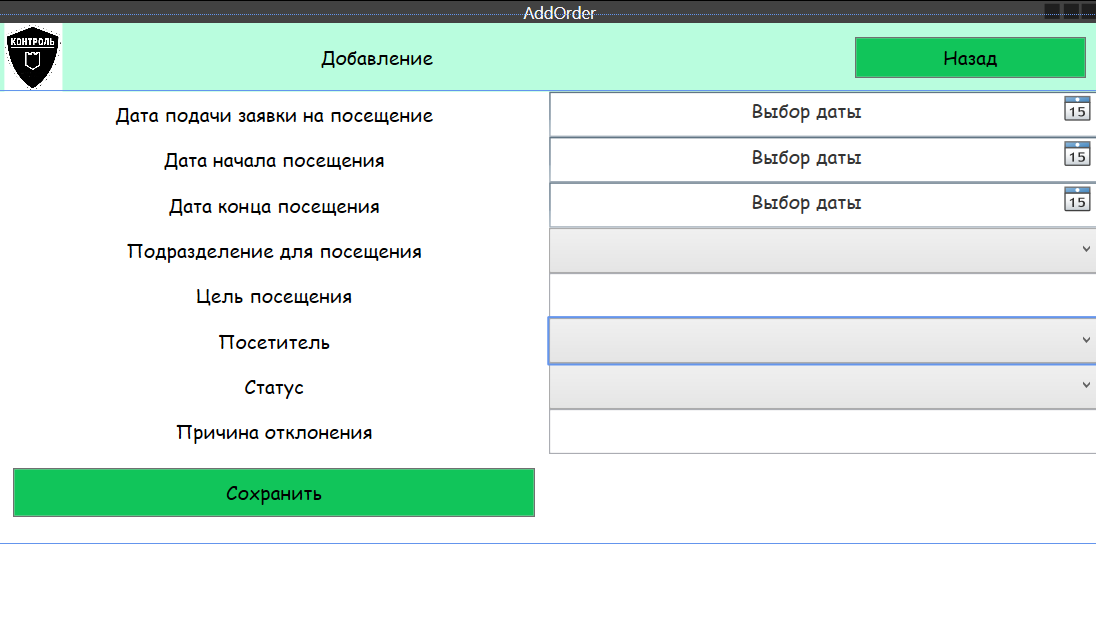
{

ShowOrders(); //обновление списка грида

}

}

}



<Window x:Class="Drive.View.AddOrder"

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

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

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

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

xmlns:local="clr-namespace:Drive.View"

mc:Ignorable="d"

Title="AddOrder" Height="450" Width="800" Loaded="Window\_Loaded">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="50"/>

<RowDefinition Height="6\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<!--Заголовок-->

<Grid Grid.Row="0" Background="#FFB9FDDE">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="50"/>

<ColumnDefinition Width="95\*"/>

<ColumnDefinition Width="468\*"/>

<ColumnDefinition Width="188\*"/>

</Grid.ColumnDefinitions>

<Image Grid.Column="0" Source="/Resources/Логотип.png"/>

<TextBlock Text="Добавление" Grid.Column="2" HorizontalAlignment="Left" Margin="89,0,0,0" />

<Button x:Name="btn\_exit" Content="Назад" Grid.Column="3" Click="btn\_exit\_Click" Margin="10,10,10,10"/>

</Grid>

<UniformGrid Grid.Row="1" Columns="2" Rows="10">

<TextBlock Text="Дата подачи заявки на посещение"/>

<DatePicker x:Name="dpDateAdd"/>

<TextBlock Text="Дата начала посещения"/>

<DatePicker x:Name="dpDateStart"/>

<TextBlock Text="Дата конца посещения"/>

<DatePicker x:Name="dpDateEnd"/>

<TextBlock Text="Подразделение для посещения"/>

<ComboBox x:Name="cbDivision">

</ComboBox>

<TextBlock Text="Цель посещения"/>

<TextBox x:Name="tbPurpose"/>

<TextBlock Text="Посетитель"/>

<ComboBox Name="cbVisitor">

</ComboBox>

<TextBlock Text="Статус"/>

<ComboBox Name="cbStatus">

</ComboBox>

<TextBlock Text="Причина отклонения"/>

<TextBox x:Name="tbReason"/>

<Button x:Name="btnSave" Content="Сохранить" Margin="10,10,10,-13" Click="btnSave\_Click"/>

</UniformGrid>

</Grid>

</Window>

**Код страницы Добавления заявок**

namespace Drive.View

{

/// <summary>

/// Логика взаимодействия для AddOrder.xaml

/// </summary>

public partial class AddOrder : Window

{

public AddOrder()

{

InitializeComponent();

dpDateAdd.DisplayDateStart = DateTime.Now.Date;

dpDateEnd.DisplayDateStart = DateTime.Now.Date;

dpDateStart.DisplayDateStart = DateTime.Now.Date;

List<Model.Subdivision> divisions = App.DB.Subdivision.ToList();

cbDivision.DisplayMemberPath = "SubdivisionName"; //видимые значения

cbDivision.SelectedValuePath = "SubdivisionID"; //получаемые значения

cbDivision.ItemsSource = divisions;

cbDivision.SelectedIndex = 0;

List<Model.Status> statuses = App.DB.Status.ToList();

cbStatus.DisplayMemberPath = "StatusName"; //видимые значения

cbStatus.SelectedValuePath = "StatusID"; //получаемые значения

cbStatus.ItemsSource = statuses;

cbStatus.SelectedIndex = 0;

List<Model.Visitor> visitors = App.DB.Visitor.ToList();

cbVisitor.DisplayMemberPath = "VisitorFullName"; //видимые значения

cbVisitor.SelectedValuePath = "VisitorID"; //получаемые значения

cbVisitor.ItemsSource = visitors;

cbVisitor.SelectedIndex = 0;

}

private void btn\_exit\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

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

{

if (dpDateAdd.SelectedDate!=null && dpDateStart.SelectedDate!=null && dpDateEnd.SelectedDate != null)

{

Model.Request request = new Model.Request();

request.RequestDateAdd = (DateTime)dpDateAdd.SelectedDate;

request.RequestDateStart = (DateTime)dpDateStart.SelectedDate;

request.RequestDateEnd = (DateTime)dpDateEnd.SelectedDate;

request.RequestSubdivision = (int)cbDivision.SelectedValue;

request.RequestPurpose = tbPurpose.Text;

request.RequestVisitor = (int)cbVisitor.SelectedValue;

request.RequestStatus = (int)cbStatus.SelectedValue;

App.DB.Request.Add(request);

try

{

App.DB.SaveChanges();

MessageBox.Show("Сохранено");

}

catch (Exception)

{

MessageBox.Show("Ошибка");

throw;

}

}

else

{

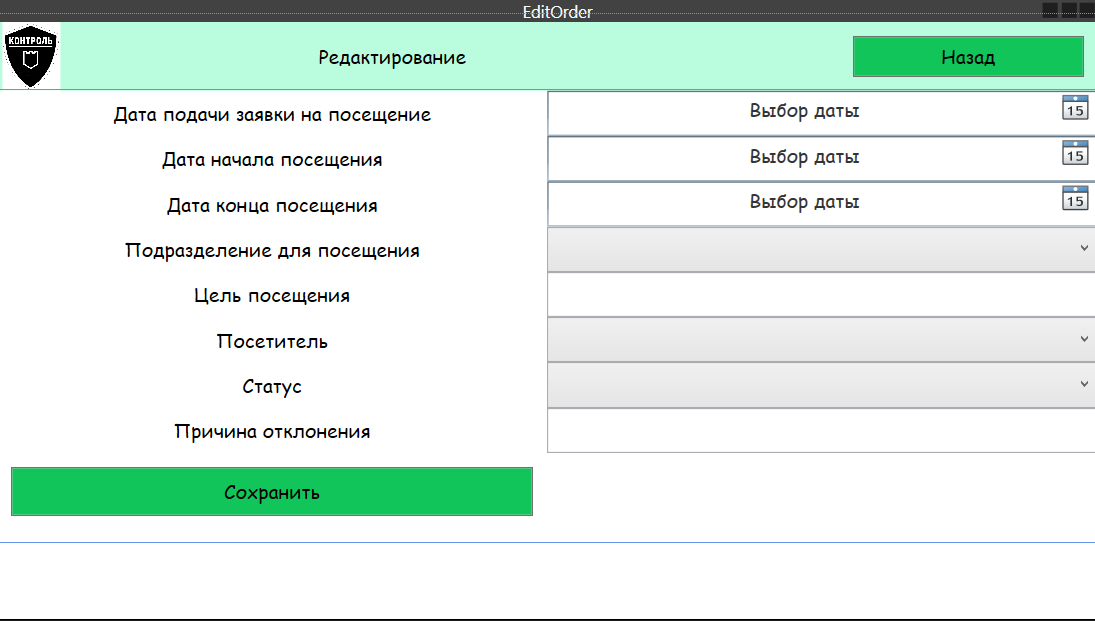
MessageBox.Show("Поля не заполнены");

}

}

}

}



<Window x:Class="Drive.View.EditOrder"

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

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

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

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

xmlns:local="clr-namespace:Drive.View"

mc:Ignorable="d"

Title="EditOrder" Height="450" Width="800">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="50"/>

<RowDefinition Height="6\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<!--Заголовок-->

<Grid Grid.Row="0" Background="#FFB9FDDE">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="50"/>

<ColumnDefinition Width="95\*"/>

<ColumnDefinition Width="468\*"/>

<ColumnDefinition Width="188\*"/>

</Grid.ColumnDefinitions>

<Image Grid.Column="0" Source="/Resources/Логотип.png"/>

<TextBlock Text="Редактирование" Grid.Column="2" HorizontalAlignment="Left" Margin="89,0,0,0" />

<Button x:Name="btn\_exit" Content="Назад" Grid.Column="3" Click="btn\_exit\_Click" Margin="10,10,10,10"/>

</Grid>

<UniformGrid Grid.Row="1" Columns="2" Rows="10">

<TextBlock Text="Дата подачи заявки на посещение"/>

<DatePicker x:Name="dpDateAdd"/>

<TextBlock Text="Дата начала посещения"/>

<DatePicker x:Name="dpDateStart"/>

<TextBlock Text="Дата конца посещения"/>

<DatePicker x:Name="dpDateEnd"/>

<TextBlock Text="Подразделение для посещения"/>

<ComboBox x:Name="cbDivision">

</ComboBox>

<TextBlock Text="Цель посещения"/>

<TextBox x:Name="tbPurpose"/>

<TextBlock Text="Посетитель"/>

<ComboBox Name="cbVisitor">

</ComboBox>

<TextBlock Text="Статус"/>

<ComboBox Name="cbStatus">

</ComboBox>

<TextBlock Text="Причина отклонения"/>

<TextBox x:Name="tbReason"/>

<Button x:Name="btnSave" Content="Сохранить" Margin="10,10,10,-13" Click="btnSave\_Click"/>

</UniformGrid>

</Grid>

</Window>

**Код страницы Редактирования**

namespace Drive.View

{

/// <summary>

/// Логика взаимодействия для EditOrder.xaml

/// </summary>

public partial class EditOrder : Window

{

private Model.Request workRequest; //создаем объект выбранной заявки с прошлого окна

public EditOrder(Model.Request editRequest)

{

InitializeComponent();

workRequest = editRequest;

dpDateAdd.DisplayDateStart = DateTime.Now.Date;

dpDateEnd.DisplayDateStart = DateTime.Now.Date;

dpDateStart.DisplayDateStart = DateTime.Now.Date;

List<Model.Subdivision> divisions = App.DB.Subdivision.ToList();

cbDivision.DisplayMemberPath = "SubdivisionName"; //видимые значения

cbDivision.SelectedValuePath = "SubdivisionID"; //получаемые значения

cbDivision.ItemsSource = divisions;

cbDivision.SelectedIndex = 0;

List<Model.Status> statuses = App.DB.Status.ToList();

cbStatus.DisplayMemberPath = "StatusName"; //видимые значения

cbStatus.SelectedValuePath = "StatusID"; //получаемые значения

cbStatus.ItemsSource = statuses;

cbStatus.SelectedIndex = 0;

List<Model.Visitor> visitors = App.DB.Visitor.ToList();

cbVisitor.DisplayMemberPath = "VisitorFullName"; //видимые значения

cbVisitor.SelectedValuePath = "VisitorID"; //получаемые значения

cbVisitor.ItemsSource = visitors;

cbVisitor.SelectedIndex = 0;

dpDateAdd.SelectedDate = workRequest.RequestDateAdd;

dpDateStart.SelectedDate = workRequest.RequestDateStart;

dpDateEnd.SelectedDate = workRequest.RequestDateEnd;

cbDivision.SelectedValue = workRequest.RequestSubdivision;

tbPurpose.Text = workRequest.RequestPurpose;

cbVisitor.SelectedValue = workRequest.RequestVisitor;

cbStatus.SelectedValue = workRequest.RequestStatus;

}

private void btn\_exit\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

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

{

workRequest.RequestDateAdd = (DateTime)dpDateAdd.SelectedDate;

workRequest.RequestDateStart = (DateTime)dpDateStart.SelectedDate;

workRequest.RequestDateEnd = (DateTime)dpDateEnd.SelectedDate;

workRequest.RequestSubdivision = (int)cbDivision.SelectedValue;

workRequest.RequestPurpose = tbPurpose.Text;

workRequest.RequestVisitor = (int)cbVisitor.SelectedValue;

workRequest.RequestStatus = (int)cbStatus.SelectedValue;

try

{

App.DB.SaveChanges();

MessageBox.Show("Сохранено");

}

catch (Exception)

{

MessageBox.Show("Ошибка");

throw;

}

}

}

}