1. **Код, БД, Тест-кейс, Отладка, Бэкап**
2. 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. 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 AngryKot

1. Создать свою папку **examXX**
2. Создать БД **dbXX** с Request, User, Role Связать с тестовым проектом
3. **БЭКАП:** ПКМ по БД, Задачи, Экспорт приложения уровня данных

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

**Инструкция к Модели БД**

ModelCondyDB, CondyEntities, CondyModel

**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 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:

1)Дата тестирования •27.05.2024

Приоритет тестирования (Низкий/Средний/ Высокий) •Высокий

Название тестирования •Авторизация заказчика с действительным логином и паролем

Шаги теста •1 Запустить приложение

2.Ввести в поле «Логин» логин

3.Ввести в поле «Пароль» пароль

4.Нажать кнопку «Войти»

Тестовые данные •Логин для входа: «Login»

Пароль для входа: «Password»

Ожидаемый результат •Пользователь успешно прошел авторизацию, появилось диалоговое окно

с приветствием

Фактический результат • Пользователь успешно прошел авторизацию, появилось диалоговое окно

с приветствием

Статус (зачет/незачет) •Зачет

2) Дата тестирования •27.05.2024

Приоритет тестирования (Низкий/Средний/ Высокий) •Высокий

Название тестирования •Авторизация заказчика с неверным логином

Шаги теста •1 Запустить приложение

2.Ввести в поле «Логин» неверный логин

3.Ввести в поле «Пароль» пароль

4.Нажать кнопку «Войти»

Тестовые данные •Логин для входа: «L»

Пароль для входа: «Password»

Ожидаемый результат •Пользователь не прошел авторизацию, появилось диалоговое окно

с оповещением об этом

Фактический результат • Пользователь не прошел авторизацию, появилось диалоговое окно

с оповещением об этом

Статус (зачет/незачет) •Зачет

3) Дата тестирования •27.05.2024

Приоритет тестирования (Низкий/Средний/ Высокий) •Высокий

Название тестирования •Авторизация заказчика с неверным паролем

Шаги теста •1 Запустить приложение

2.Ввести в поле «Логин» логин

3.Ввести в поле неверный «Пароль» пароль

4.Нажать кнопку «Войти»

Тестовые данные •Логин для входа: «L»

Пароль для входа: «Password»

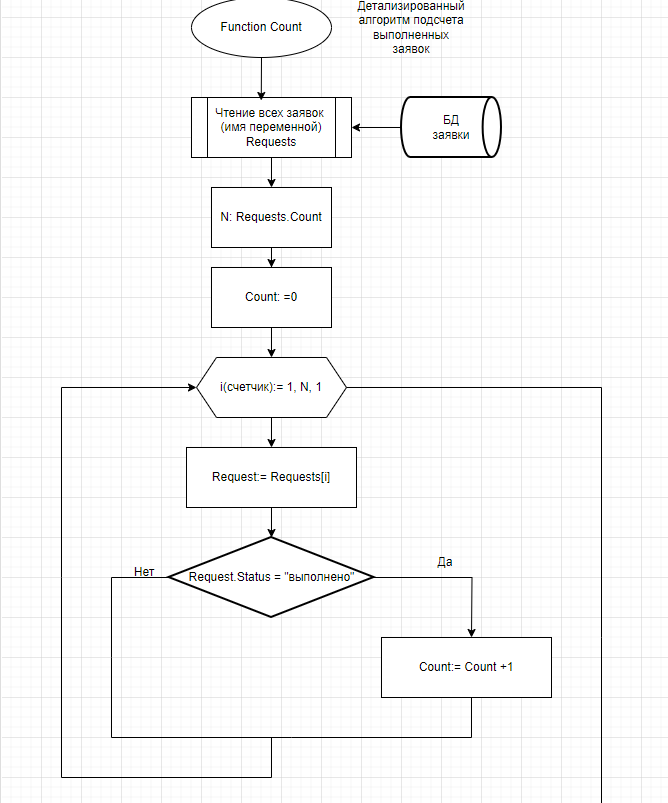
Ожидаемый результат •Пользователь не прошел авторизацию, появилось диалоговое окно

с оповещением об этом

Фактический результат • Пользователь не прошел авторизацию, появилось диалоговое окно

с оповещением об этом

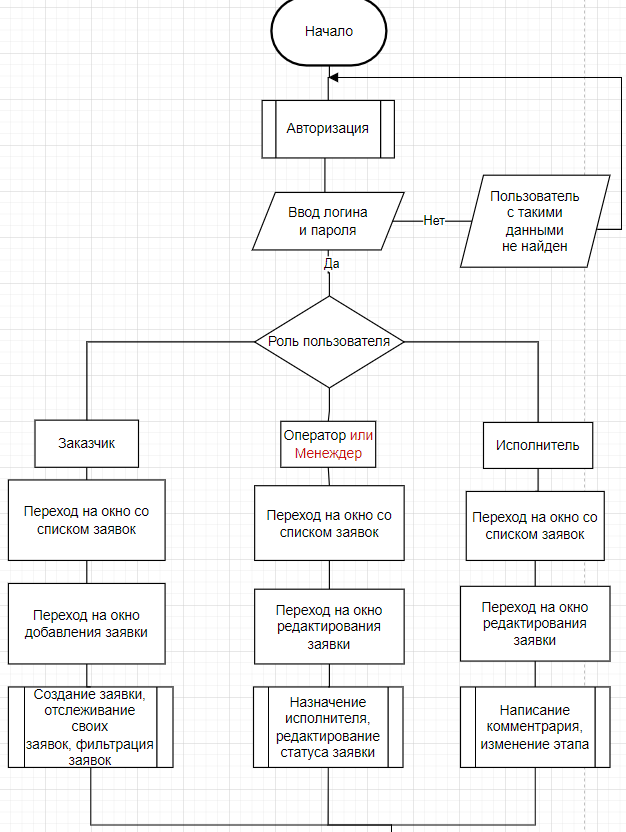
Статус (зачет/незачет) •Зачет

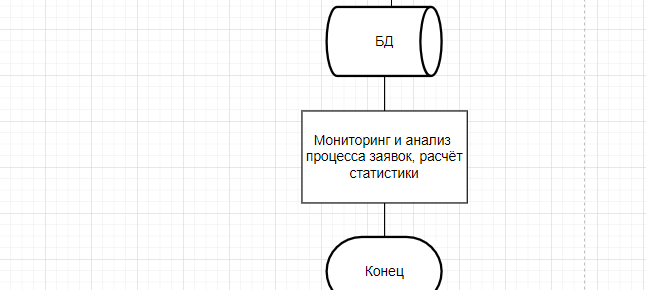


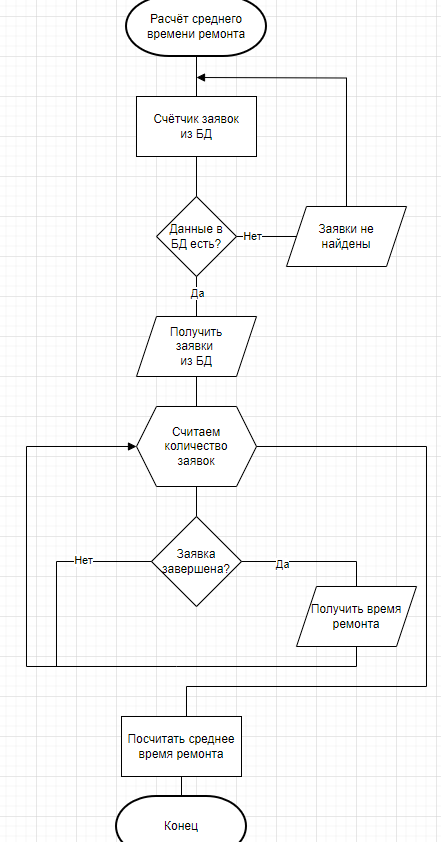
Изображение выглядит как диаграмма, линия, круг, График

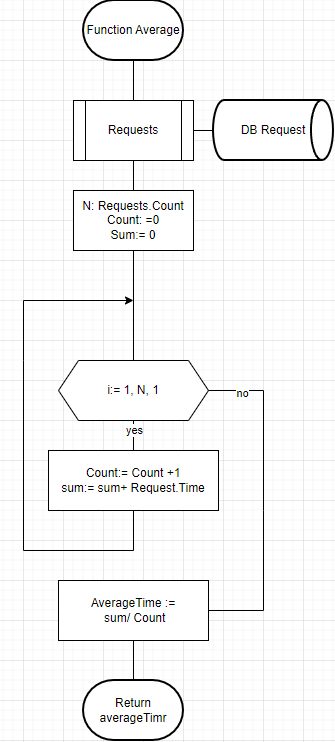
Автоматически созданное описание

1. Блок-схема общая









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

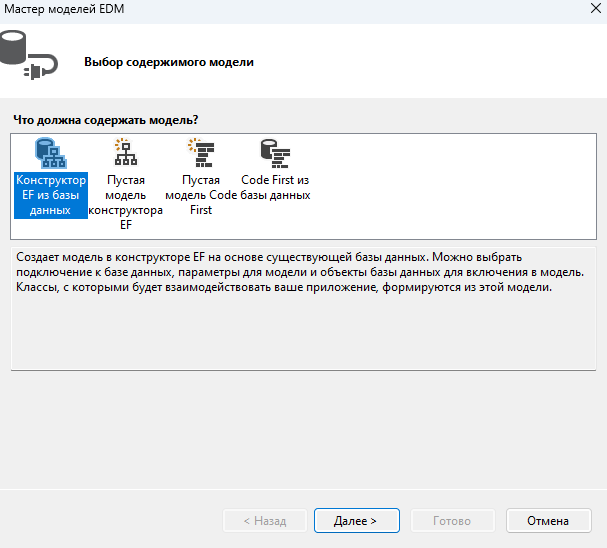
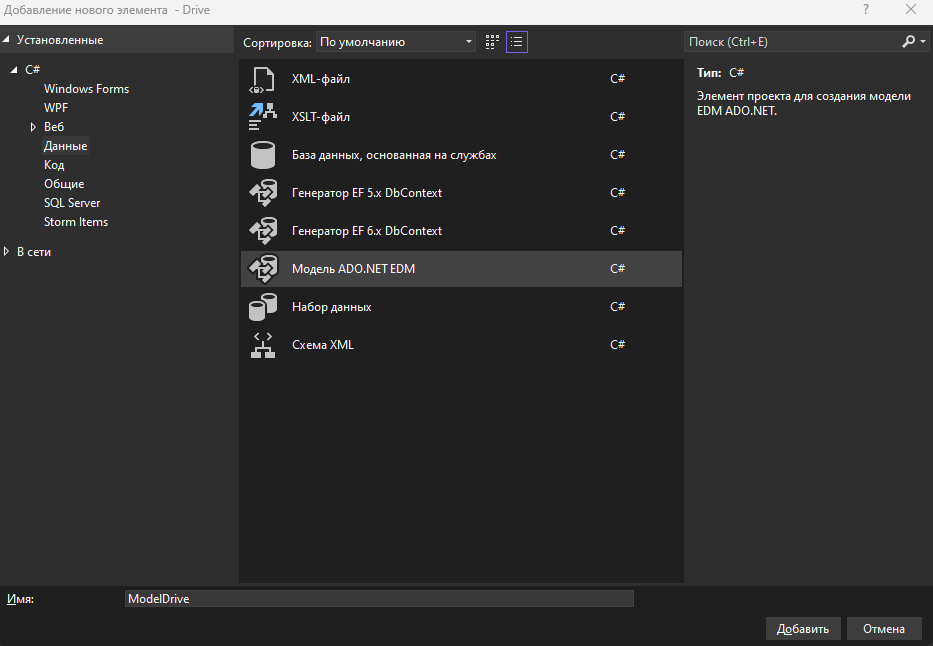
Установить иконку приложения. Выбрать в файле XAML элемент Window. Перейти в панель «Свойства», вкладка «Общие», свойство «Icon». Из выпадающего списка выбираем «Resources.ico».

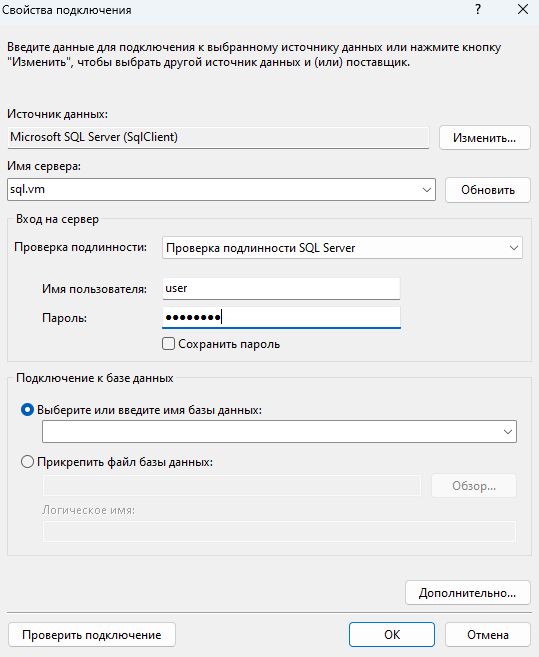
Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

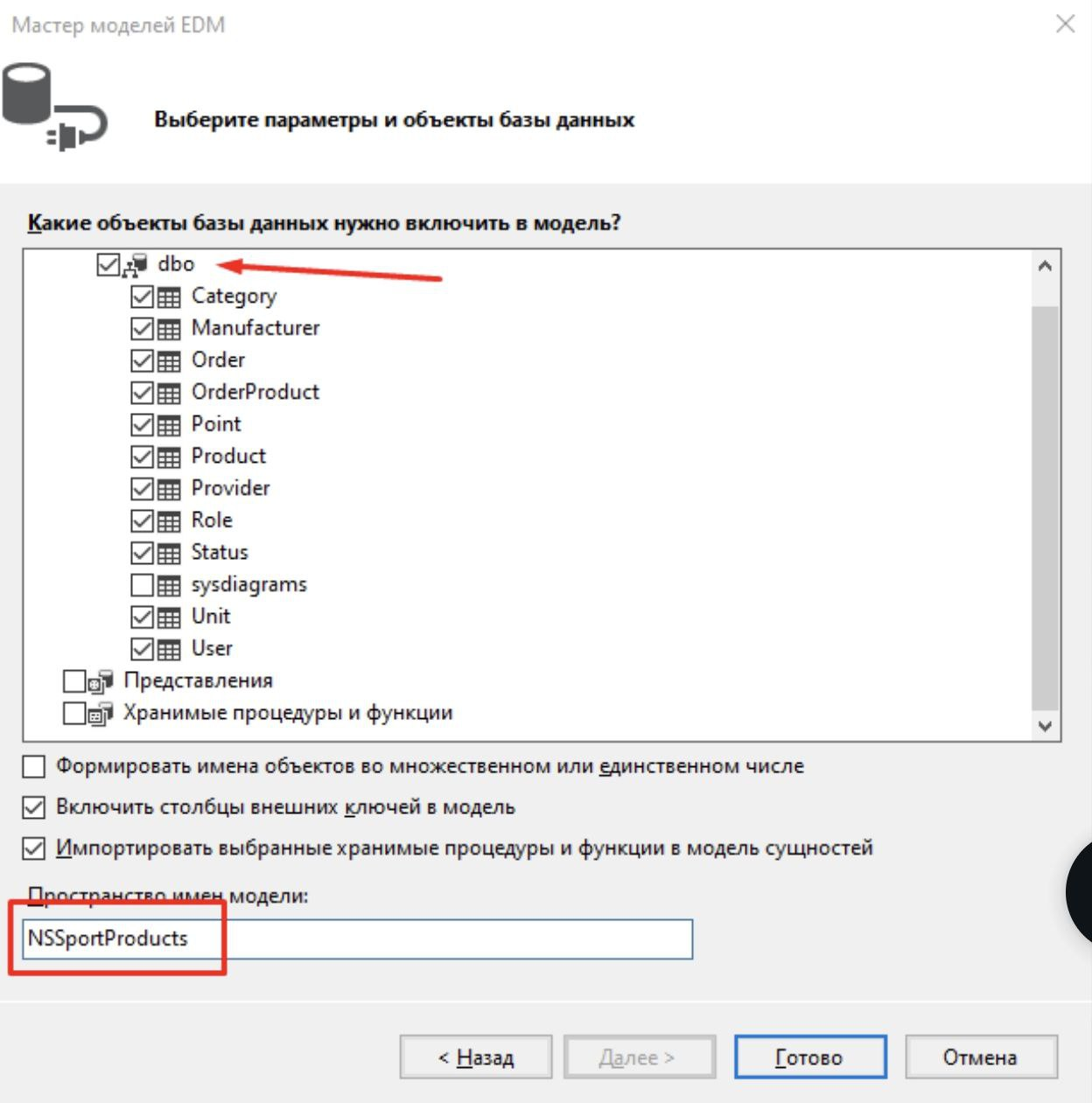
1. Создание модели БД

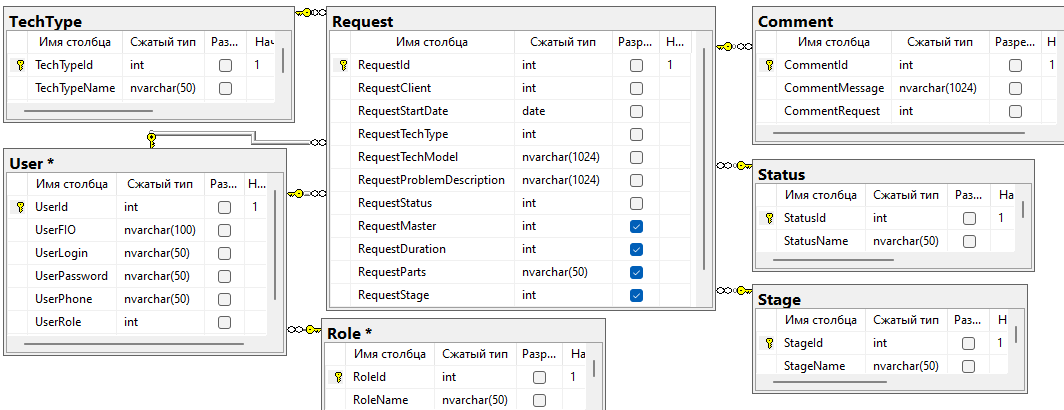
Model -> Добавить класс

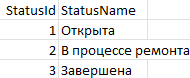
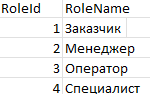
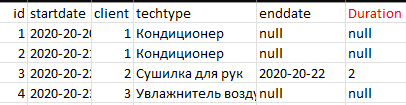


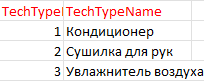
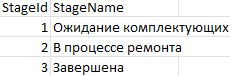
Изображение выглядит как текст, снимок экрана, веб-страница, Веб-сайт

Автоматически созданное описание









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="AliceBlue"/>

<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>

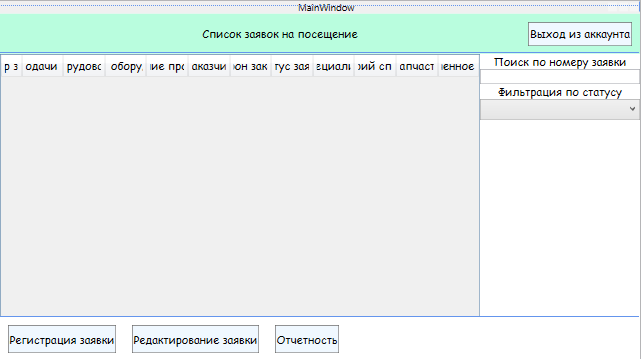
</Application>

* **Xaml.cs окон**

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

Заголовок шапка:

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



<!--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="Список заявок на посещение" Grid.Column="1" HorizontalAlignment="Center"/>

<Button x:Name="btn\_exit" Content="Выход из аккаунта" Grid.Column="2" Click="btn\_exit\_Click"/>

</Grid>

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

<DataGrid.Columns>

<DataGridTextColumn Header="Номер заявки" Binding="{Binding RequestId}"/>

<DataGridTextColumn Header="Заказчик" Binding="{Binding User.UserFIO}"/>

<DataGridTextColumn Header="Дата подачи заявки" Binding="{Binding RequestStartDate, StringFormat=dd:MM:yyyy}"/>

<DataGridTextColumn Header="Тип оборудования" Binding="{Binding TechType.TechTypeName}"/>

<DataGridTextColumn Header="Модель оборудования" Binding="{Binding RequestTechModel}"/>

<DataGridTextColumn Header="Описание проблемы" Binding="{Binding RequestProblemDescription}"/>

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

<DataGridTextColumn Header="Мастер" Binding="{Binding User1.UserFIO}"/>

<DataGridTextColumn Header="Затраченное время" Binding="{Binding RequestDuration}"/>

<DataGridTextColumn Header="Запчасти" Binding="{Binding RequestParts}"/>

<DataGridTextColumn Header="Этап выполнения" Binding="{Binding Stage.StageName}"/>

</DataGrid.Columns>

</DataGrid>

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

<TextBlock Text="Поиск по номеру заявки"/>

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

<TextBlock Text="Фильтрация по статусу"/>

<ComboBox x:Name="cb\_statusFilter" SelectionChanged="cb\_statusFilter\_SelectionChanged" />

</StackPanel>

**Код страницы Списка заявок**

namespace TestDemo.View

{

/// <summary>

/// Логика взаимодействия для ListRequests.xaml

/// </summary>

public partial class ListRequests : Window

{

public ListRequests()

{

InitializeComponent();

var statuses = App.DB.Status.ToList();

cbStatusFilter.SelectedValuePath = "StatusId";

cbStatusFilter.DisplayMemberPath = "StatusName";

cbStatusFilter.ItemsSource = statuses;

cbStatusFilter.SelectedIndex = 0;

if(App.currentUser.UserRole == 1)

{

btn\_editRequest.Visibility = Visibility.Hidden;

btn\_report.Visibility = Visibility.Hidden;

}

if(App.currentUser.UserRole == 4)

{

btn\_createRequest.Visibility = Visibility.Hidden;

btn\_report.Visibility = Visibility.Hidden;

}

ShowRequests();

}

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

{

ShowRequests();

}

private void ShowRequests()

{

var requests = App.DB.Request.ToList();

if (tbSearchById.Text.Length > 0)

{

requests = requests.FindAll(req => req.RequestId.ToString().Contains(tbSearchById.Text));

}

requests = requests.FindAll(req => req.RequestStatus == (int)cbStatusFilter.SelectedValue);

dgRequest.ItemsSource = requests;

}

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

{

string statistic = $"Количество выполненых заявок: {App.DB.Request.Count(request => request.RequestStatus == 4)}";

var completedRequests = App.DB.Request.Where(request => request.RequestDuration != null).ToList();

var averageTime = "";

if (completedRequests.Count != 0)

{

averageTime = completedRequests.Average(request => request.RequestDuration).ToString();

}

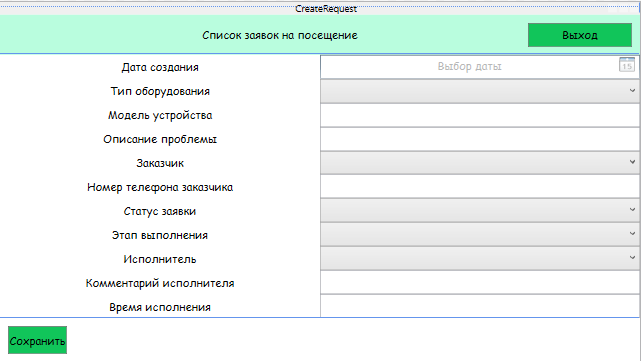
statistic += $"\nСреднее время выполнения заявки: {averageTime} часов";

MessageBox.Show(statistic);

}

}

}

****

<Window x:Class="OOO\_Condy.View.CreateRequest"

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:OOO\_Condy.View"

mc:Ignorable="d"

Title="CreateRequest" 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="\*" />

<ColumnDefinition Width="150"/>

</Grid.ColumnDefinitions>

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

<TextBlock Text="Список заявок на посещение" Grid.Column="1" HorizontalAlignment="Center"/>

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

</Grid>

<UniformGrid Grid.Row="1" Columns="2" Rows="11">

<TextBlock Text="Дата создания"/>

<DatePicker x:Name="dpDateAdd" IsEnabled="False"/>

<TextBlock Text="Тип оборудования"/>

<ComboBox x:Name="cbEquipmentType"/>

<TextBlock Text="Модель устройства"/>

<TextBox x:Name="tbDeviceModel" />

<TextBlock Text="Описание проблемы"/>

<TextBox x:Name="tbDescription" />

<TextBlock Text="Заказчик"/>

<ComboBox x:Name="cbCustomer"/>

<TextBlock Text="Номер телефона заказчика"/>

<TextBox x:Name="tbCustomerPhone"/>

<TextBlock Text="Статус заявки"/>

<ComboBox Name="cbStatus" />

<TextBlock Text="Этап выполнения"/>

<ComboBox Name="cbStage"/>

<TextBlock Text="Исполнитель"/>

<ComboBox x:Name="cbEmployee"/>

<TextBlock Text="Комментарий исполнителя"/>

<TextBox x:Name="tbComment" />

<TextBlock Text="Время исполнения"/>

<TextBox x:Name="tbDuration" />

</UniformGrid>

<!--Кнопки управления-->

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

<Button x:Name="btnSave" Content="Сохранить" Click="btnSave\_Click"/>

</StackPanel>

</Grid>

</Window>

**Код страницы Добавления и Изменения заявок**

namespace OOO\_Condy.View

{

/// <summary>

/// Логика взаимодействия для CreateRequest.xaml

/// </summary>

public partial class CreateRequest : Window

{

private bool isEdit = false;

private Model.Request request;

public CreateRequest(Model.Request request)

{

InitializeComponent();

isEdit = request != null;

this.request = request;

var statuses = App.DB.Status.ToList();

cbStatus.SelectedValuePath = "StatusId";

cbStatus.DisplayMemberPath = "StatusTitle";

cbStatus.ItemsSource = statuses;

cbStatus.SelectedIndex = 0;

var stages = App.DB.Stage.ToList();

cbStage.SelectedValuePath = "StageId";

cbStage.DisplayMemberPath = "StageTitle";

cbStage.ItemsSource = stages;

cbStage.SelectedIndex = 0;

var equipmentTypes = App.DB.EquipmentType.ToList();

cbEquipmentType.SelectedValuePath = "EquipmentTypeId";

cbEquipmentType.DisplayMemberPath = "EquipmentTypeTitle";

cbEquipmentType.ItemsSource = equipmentTypes;

cbEquipmentType.SelectedIndex = 0;

var users = App.DB.User.ToList();

cbCustomer.SelectedValuePath = "UserId";

cbCustomer.DisplayMemberPath = "UserFullname";

cbEmployee.SelectedValuePath = "UserId";

cbEmployee.DisplayMemberPath = "UserFullname";

cbCustomer.ItemsSource = users.FindAll(user => user.UserRole == 1);

cbCustomer.SelectedIndex = 0;

cbEmployee.ItemsSource = users.FindAll(user => user.UserRole == 3);

cbEmployee.SelectedIndex = 0;

if (isEdit)

{

dpDateAdd.SelectedDate = request.RequestCreateAt;

cbEquipmentType.SelectedValue = request.RequestEquipmentType;

tbDeviceModel.Text = request.RequestDeviceModel;

tbDescription.Text = request.RequestDescription;

tbCustomerPhone.Text = request.RequestPhone;

cbCustomer.SelectedValue = request.RequestCustomer;

cbEmployee.SelectedValue = request.RequestEmployee;

cbStatus.SelectedValue = request.RequestStatus;

cbStage.SelectedValue = request.RequesStage;

tbComment.Text = request.RequestComment;

tbDuration.Text = request.RequestDuration.ToString();

if (App.currentUser.UserRole == 2)

{

cbStage.IsEnabled = false;

cbEquipmentType.IsEnabled = false;

cbCustomer.IsEnabled = false;

tbDeviceModel.IsEnabled = false;

tbDescription.IsEnabled = false;

tbCustomerPhone.IsEnabled = false;

tbComment.IsEnabled = false;

tbDuration.IsEnabled = false;

}

else

{

cbStatus.IsEnabled = false;

cbEmployee.IsEnabled = false;

cbEquipmentType.IsEnabled = false;

cbCustomer.IsEnabled = false;

tbDeviceModel.IsEnabled = false;

tbDescription.IsEnabled = false;

tbCustomerPhone.IsEnabled = false;

}

}

else

{

cbStatus.IsEnabled = false;

cbStage.IsEnabled = false;

cbEmployee.IsEnabled = false;

tbComment.IsEnabled = false;

tbDuration.IsEnabled = false;

}

}

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

{

Close();

}

}

}