**Министерство образования Московской области**

**ГОУ ВО МО «Государственный гуманитарно-технологический университет» (ГГТУ) Ликино-Дулевский политехнический колледж- филиал ГГТУ**

Наименование колледжа/техникума

**О Т Ч Ё Т**

**ПО УЧЕБНОЙ ПРАКТИКЕ**

ПМ.01 Разработка модулей программного обеспечения для компьютерных систем

(код и наименование ПМ)

Обучающегося\_\_\_\_ Магомедова Мурада Мухтаровича

Ф.И.О.

Курс \_\_\_\_3\_\_\_\_\_ группа \_\_\_ИСП.22А\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Специальность 09.02.07 Информационные системы и программирование \_\_\_

Место практики Ликино-Дулевский политехнический колледж – филиал ГГТУ

Период практики 25.11.2024 г. – 07.12.2024 г., 07.04.2025 г. - 19.04.2025 г.

Руководители практики

от колледжа \_\_\_\_\_\_\_\_ Кузьмина Елена Евгеньевна \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гжегожевский Сергей Владимирович\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ветрова Карина Валерьевна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тишкин Валентин Васильевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ф.И.О.

г. Ликино-Дулево

2025 г.

Оглавление

[Введение 3](#_Toc184373809)

[1. Разработка технического задания 4](#_Toc184373810)

[2. Проектирование и создание базы данных 5](#_Toc184373811)

[3. Разработка интерфейса приложения 7](#_Toc184373812)

[4. Разработка программных модулей 11](#_Toc184373813)

[5. Разработка Api 26](#_Toc184373814)

[6. Библиотека классов 34](#_Toc184373815)

[Заключение 37](#_Toc184373816)

# Введение

Целью учебной практики является закрепление и углубление знаний, полученных обучающимися в процессе теоретического обучения по профессиональному модулю ПМ.01 «Разработка модулей программного обеспечения для компьютерных систем».

Задачи практики:

1. Получение первичных профессиональных умений и навыков;

2. Подготовка студентов к осознанному и глубокому изучению общепрофессиональных и специальных дисциплин;

3. Получение практических профессиональных умений и навыков по специальности 09.02.07 Информационные системы и программирование.

С целью овладения видом деятельности «Разработка модулей программного обеспечения для компьютерных систем» обучающийся должен обладать соответствующими профессиональными компетенциями:

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

ПК 1.2. Разрабатывать программные модули в соответствии с техническим заданием;

ПК 1.3. Выполнять отладку программных модулей с использованием специализированных программных средств;

ПК 1.4. Выполнять тестирование программных модулей;

1. Разработка технического задания

Предметная область представляет собой комплекс технологий и методов, предназначенных для сбора, анализа и интерпретации данных о состоянии природной среды. Основные параметры, которые отслеживаются, включают качество воздуха (уровни загрязняющих веществ, таких как углекислый газ, оксиды азота и мелкие частицы), качество воды (содержание химических веществ, биологических показателей и физико-химических характеристик), а также состояние почвы, уровень шума и другие экологические факторы.

1. Ключевые сущности выбранной предметной области:

Пользователи, отчеты, измерения, данные о загрязнении, местоположение, стандарты, датчики

Процессы: просмотр пользователей, просмотр отчетов, просмотр измерений, просмотр местоположения, просмотр датчиков, добавление пользователей, добавление отчетов, добавление измерений, добавление местоположения, добавление датчиков, поиск, удаление, сортировка, фильтрация.

2. Состав диаграммы сущностей и связей (ER-диаграмма), отражающую взаимосвязь основных понятий.

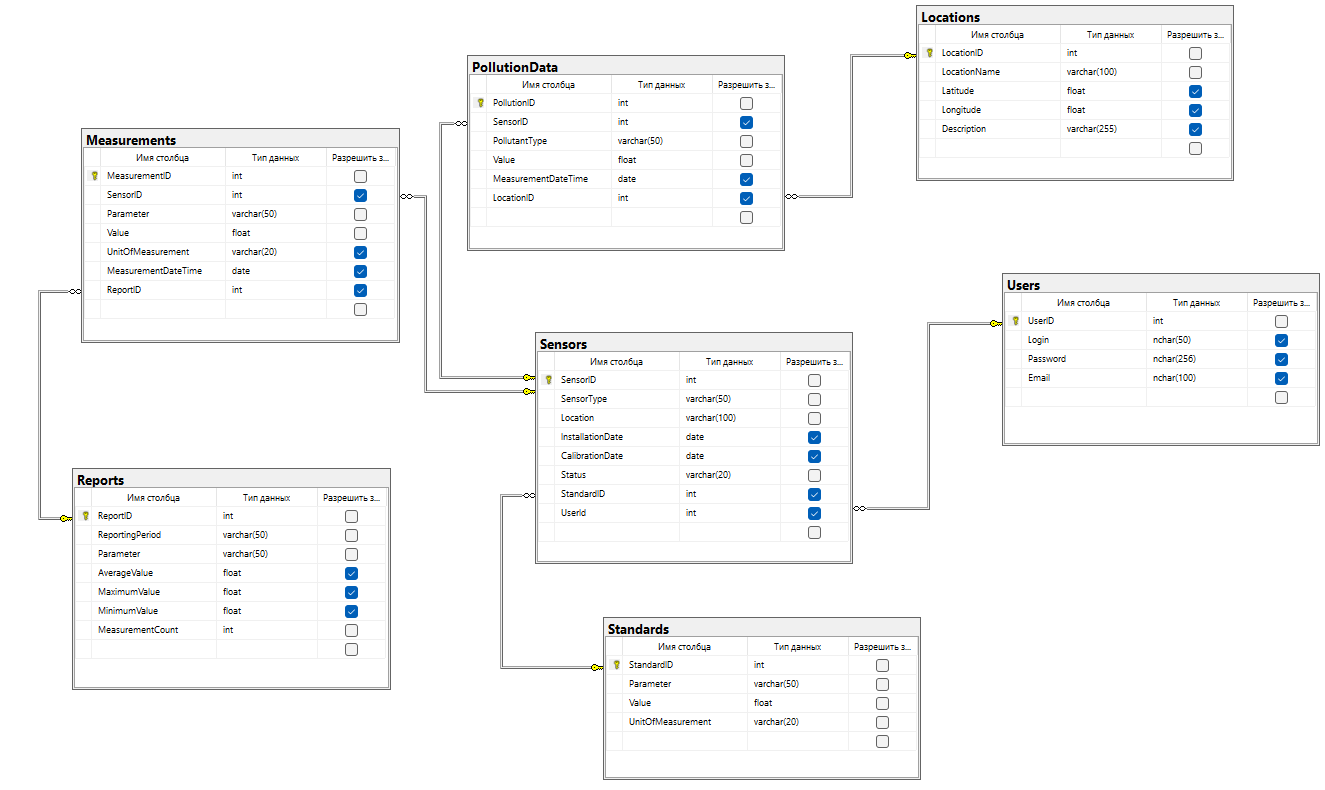


рис.1 «ER-диаграмма»

3. Описание функциональных требований.

1. Регистрация и авторизация
2. Просмотр, добавление, и удаление отчетов
3. Просмотр, добавление и удаление датчиков
4. Просмотр, добавление и удаление пользователей
5. Просмотр, добавление и удаление местоположений
6. Просмотр, добавление и удаление измерений

Описание нефункциональных требований.

1. Стабильная работа приложения на всех стадиях.
2. Работа приложения без задержек.

Анализ рисков и ограничений.

Возможны утечки данных пользователей из за халатности обращения к программе  
План тестирования.

Проверка всех функций на работоспособность и корректность

2. Проектирование и создание базы данных  
  
Цель:  
Разработка структуры базы данных для хранения данных выбранного программного продукта.

Задание:  
На основании описания предметной области, созданного ранее, разработайте базу данных, выполнив следующие шаги:  
  
СУБД: MS SQLServer 20  
- ER-диаграмма базы данных.

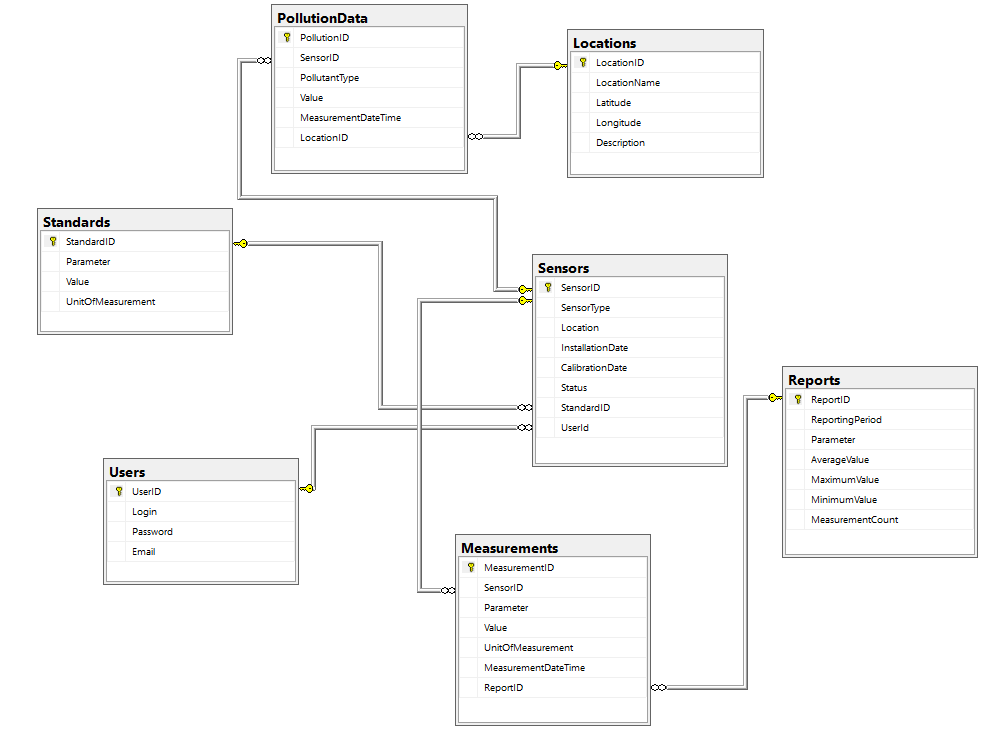


рис.2 «ER-диаграмма»

Подключение к проекту wpf

Nugget Для подключения БД требуются дополнительные пакеты Nugget

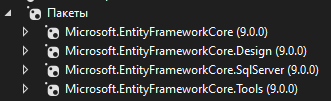


рис.3 «Пакеты Nugget»

После установки Nugget пакетов открываем консоль диспетчеров пакета и прописываем след.код EF:  
Scaffold-DbContext "Server=MM3A11;Database= EnvironmentalMonitoring;TrustServerCertificate=True;Trusted\_Connection=True"

Модель базы данных в виде структуры классов:

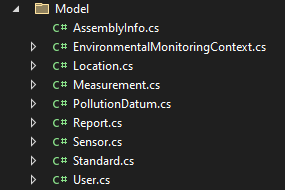


рис.4 « структура БД»

# 3. Разработка интерфейса приложения

Для разработки приложения wpf использовался стандартный шаблон:

Приложение WPF (Майкрософт)

MainWindow

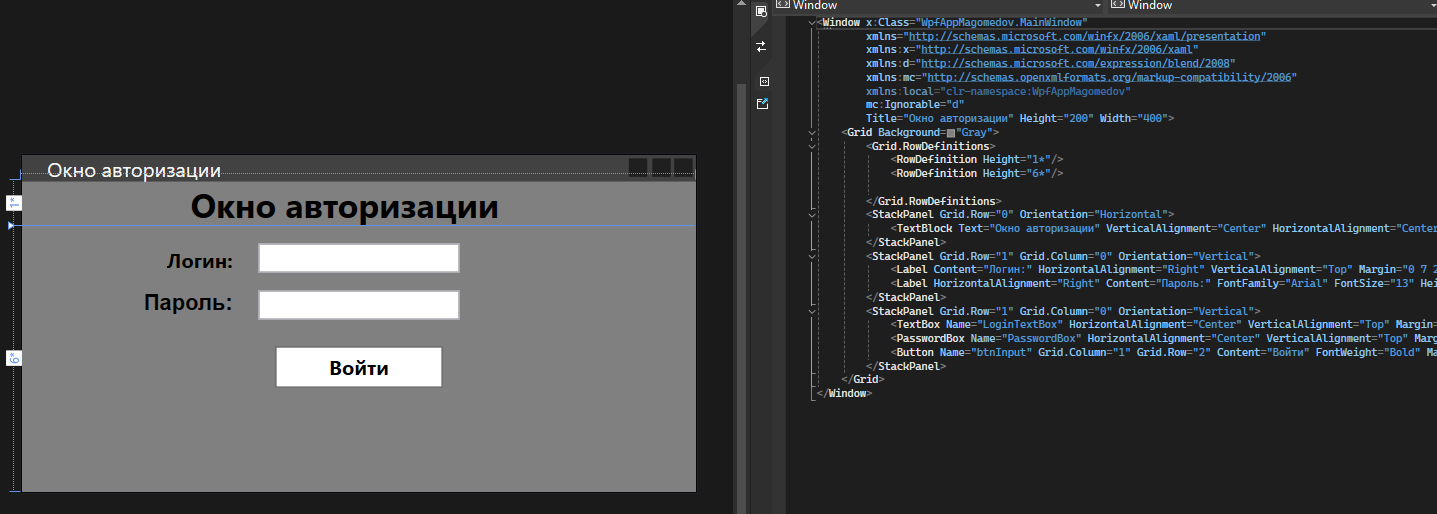


рис.5 «Окно авторизации»

WindowGlobal

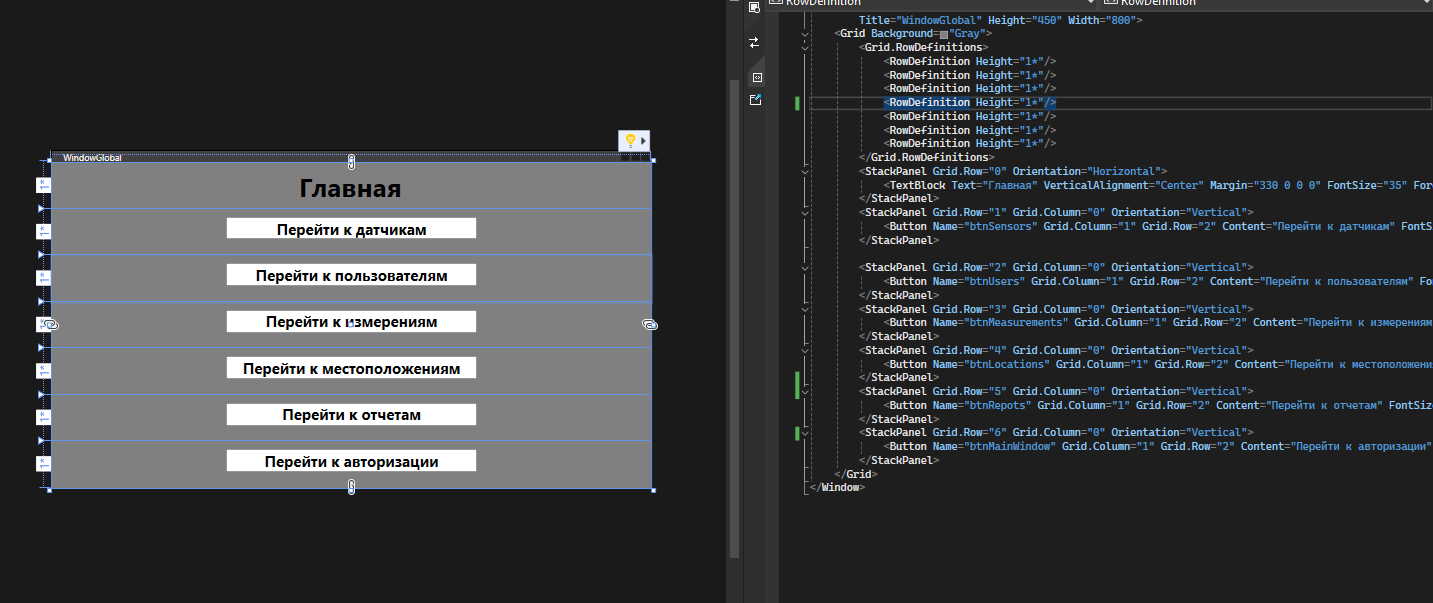


рис.6 «Главное меню»

WindowLocations

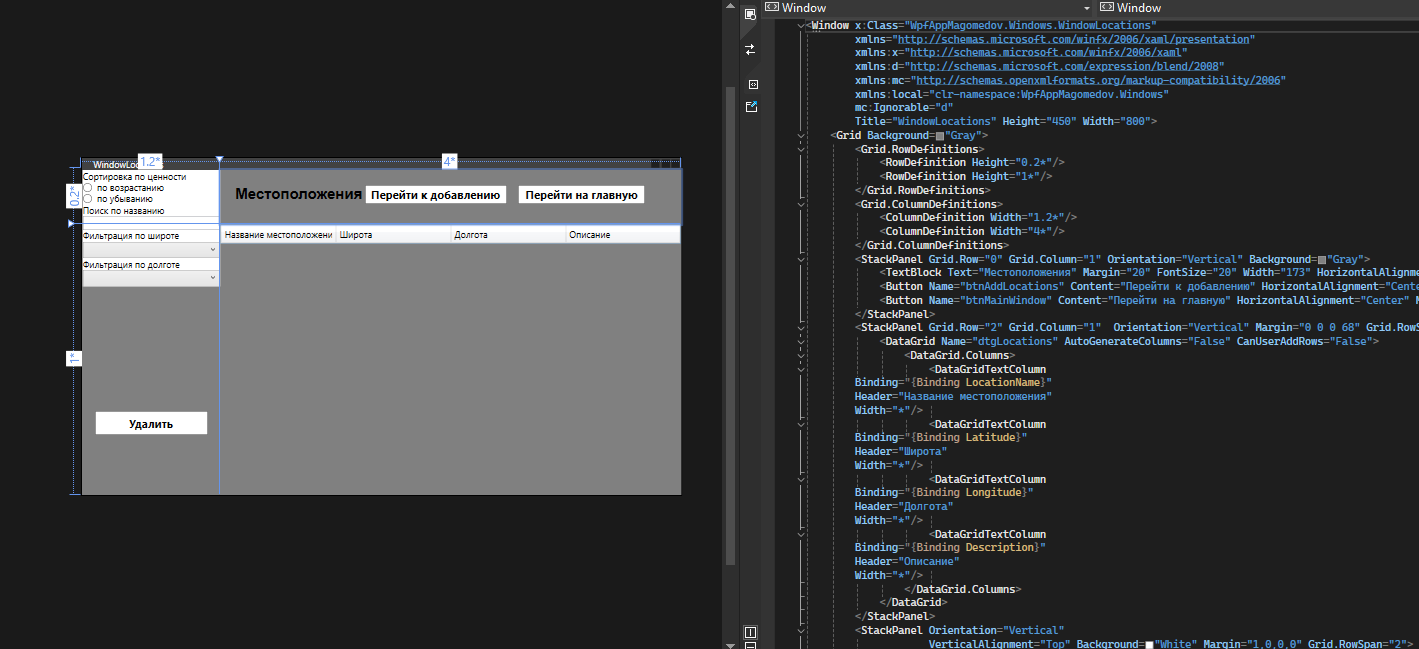


рис.6 «Окно местоположения»

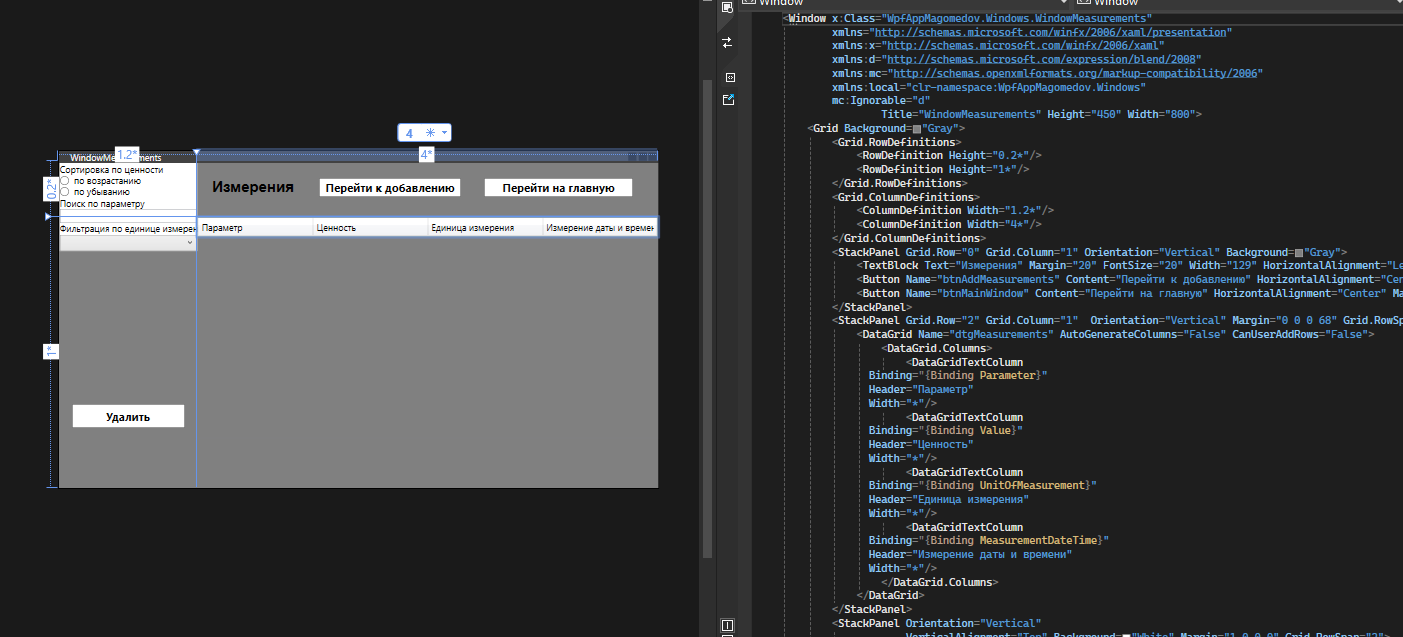
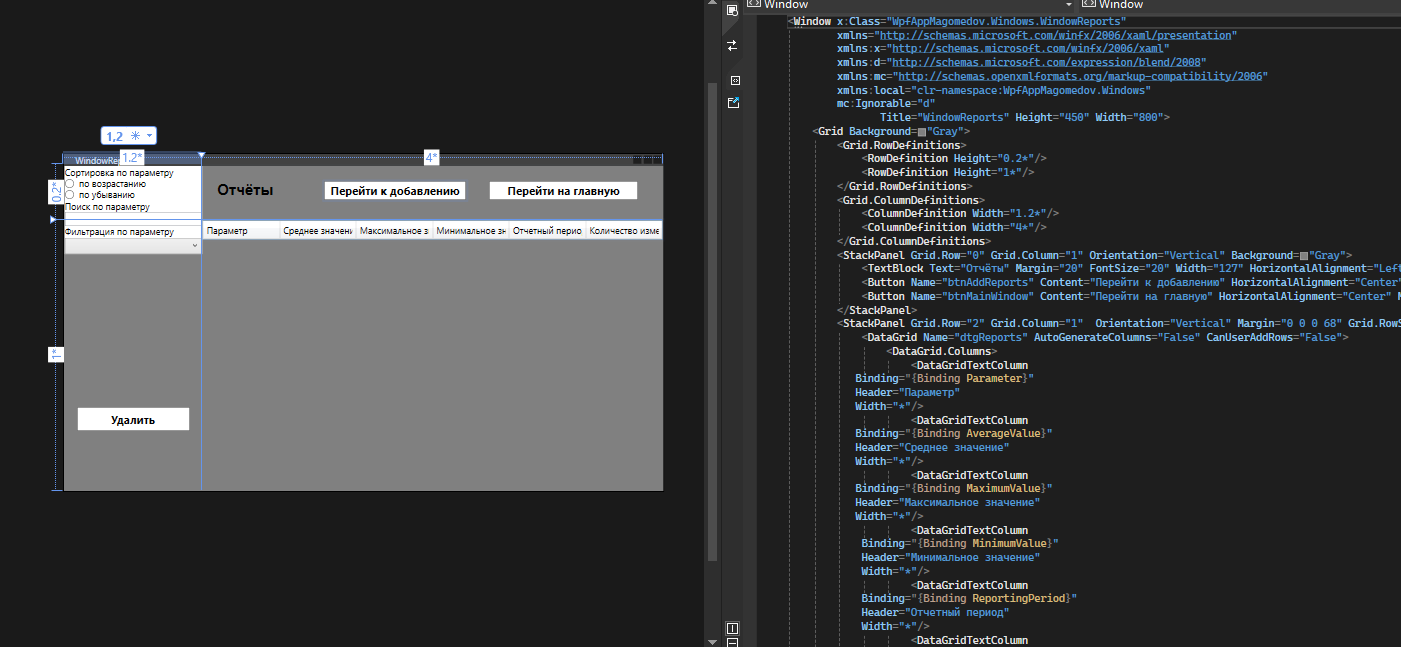
WindowMeasurements 

рис.7 «Окно измерения»WindowReports

 рис.8 «Окно отчеты»

WindowSensors

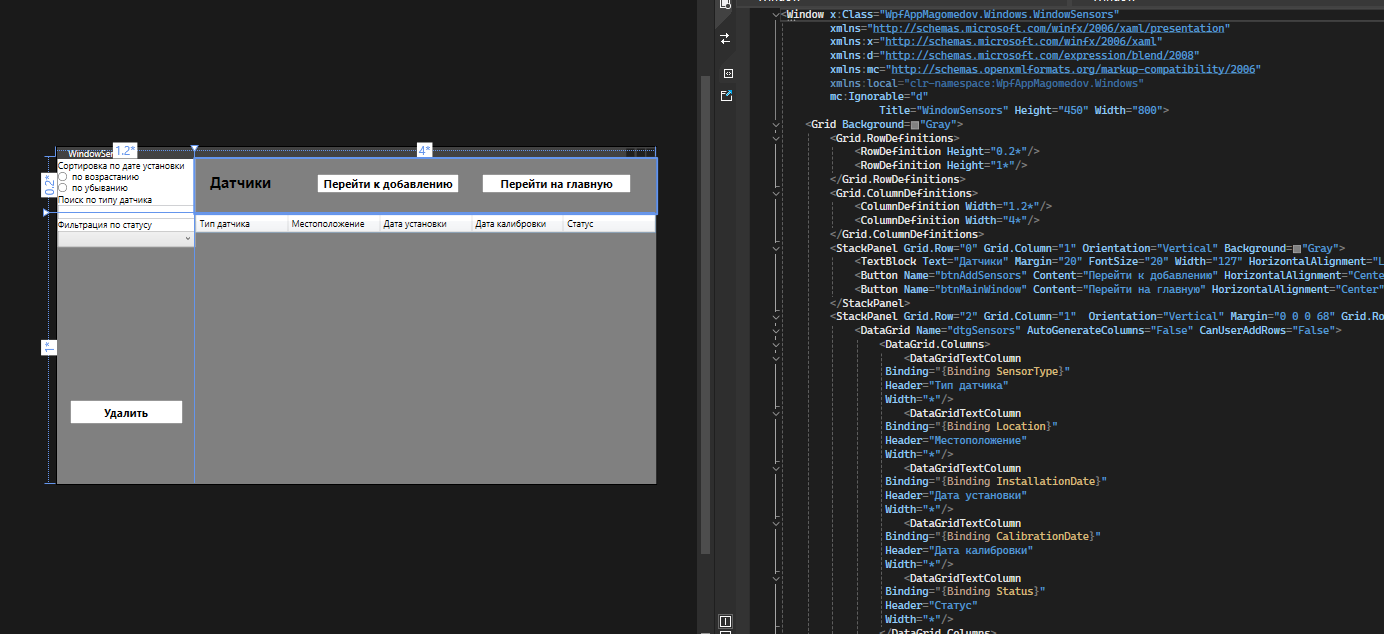


рис.9 «Окно датчики»

WindowUsers

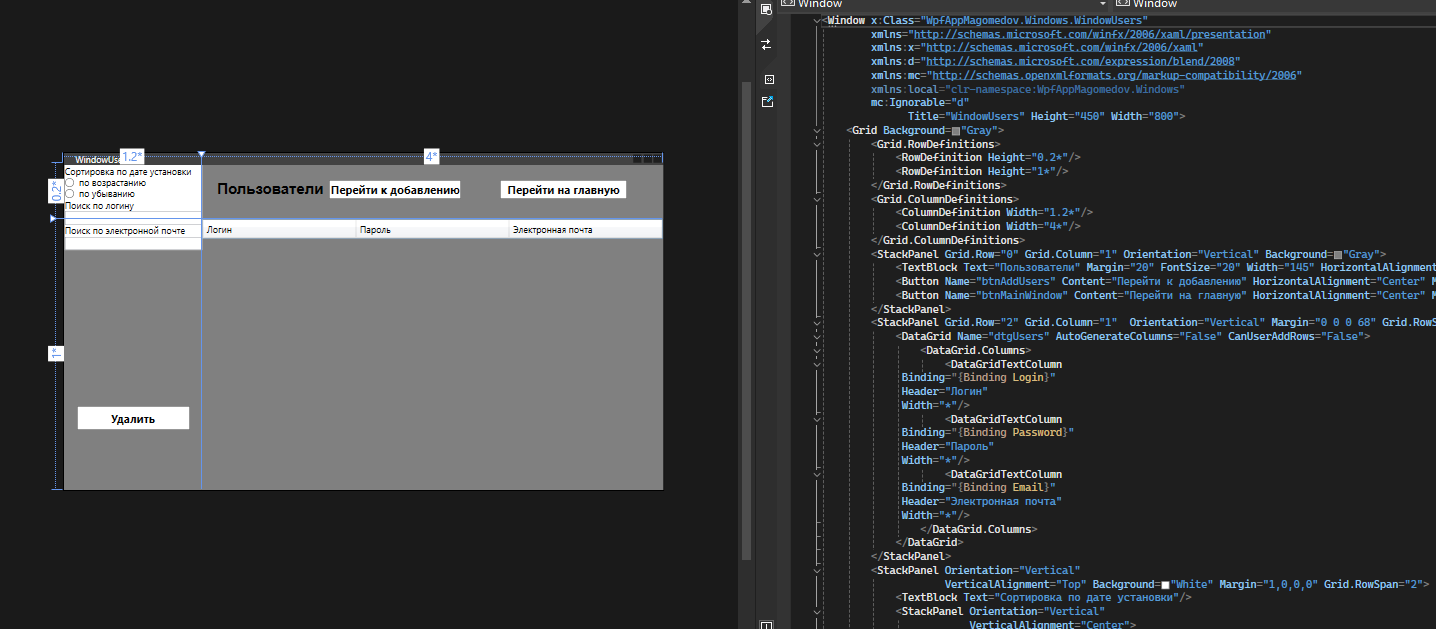


рис.10 «Окно пользователи»

AddLocations

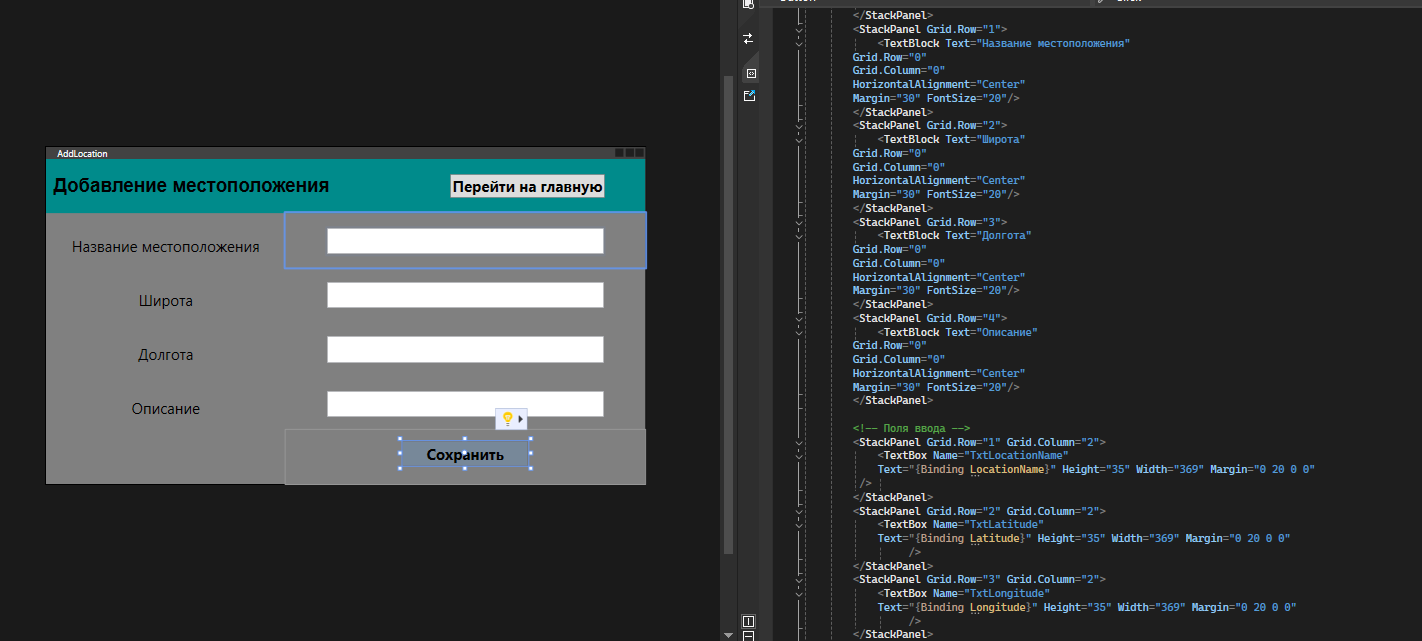


рис.11 «Окно добавление местоположения»

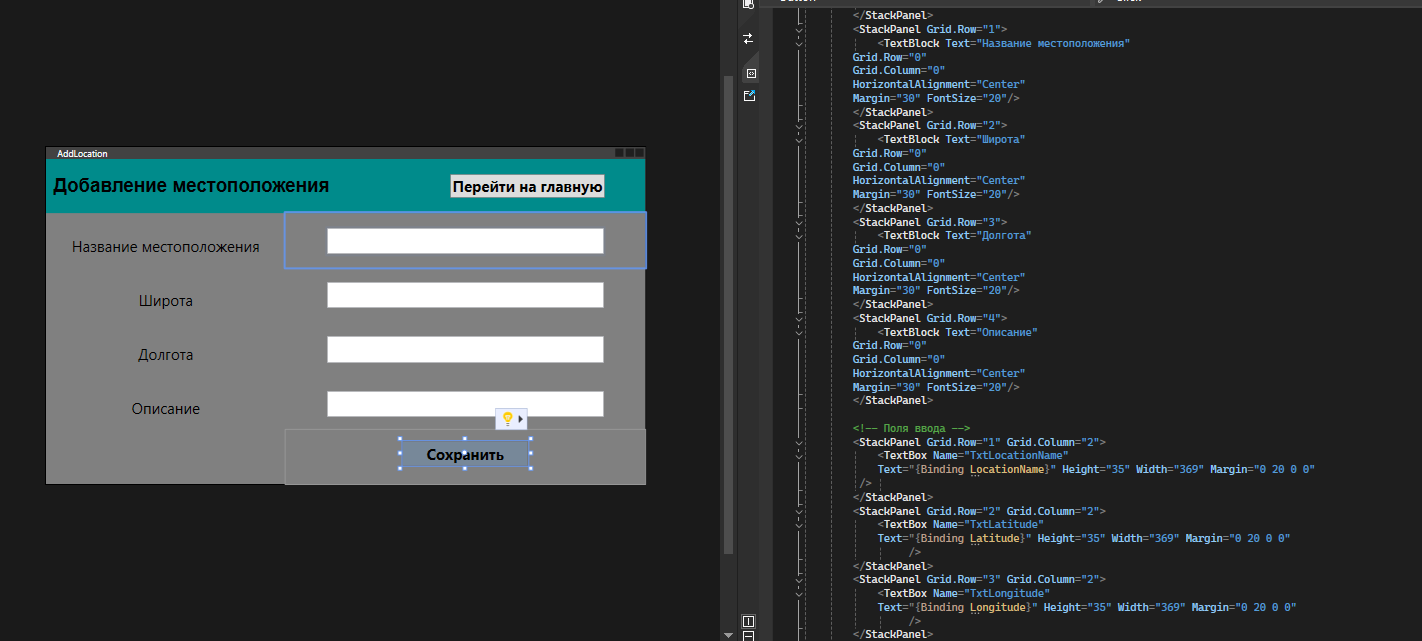
AddMeasurements

рис.12 «Окно добавление измерения»

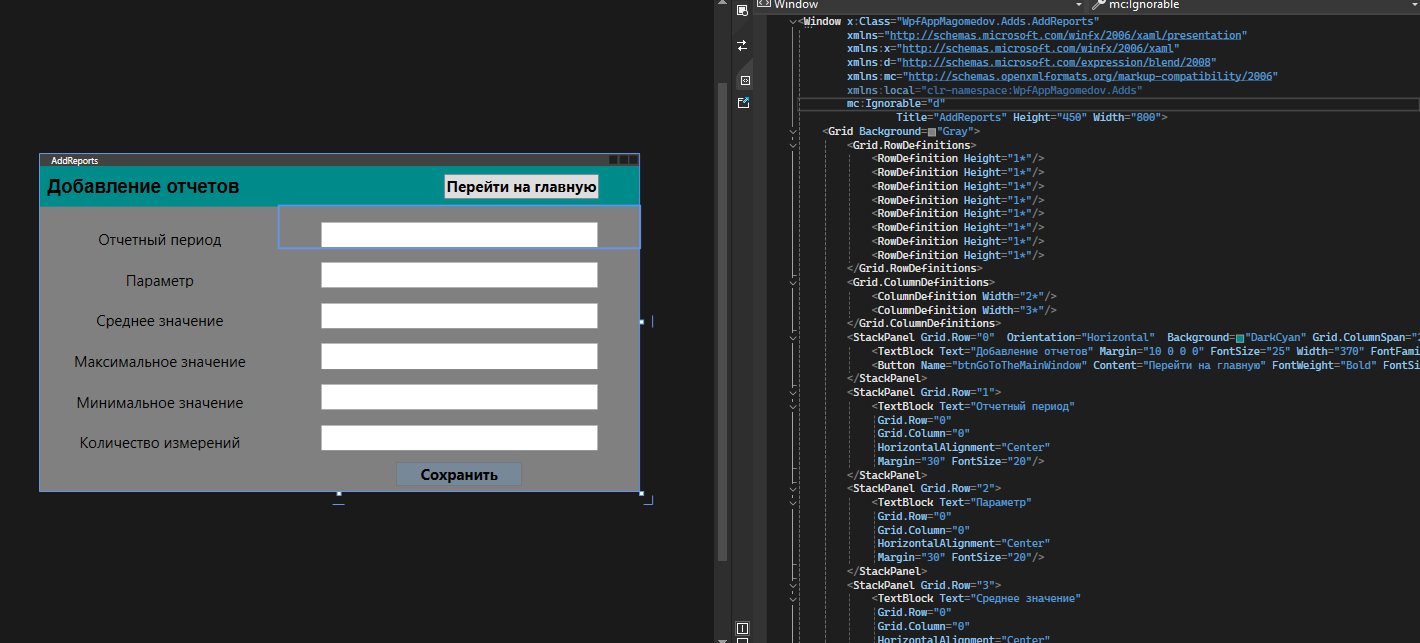
AddReports 

рис.13 «Окно добавление отчетов»

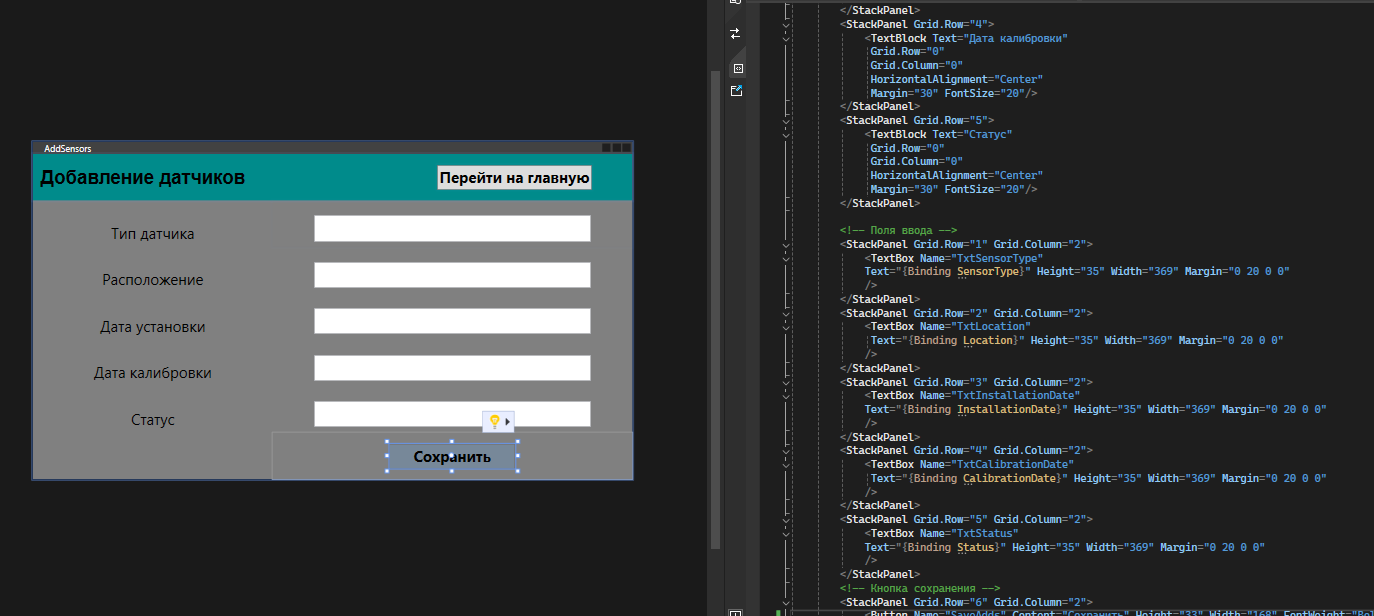
AddSensors 

рис.14 «Окно добавление датчиков»

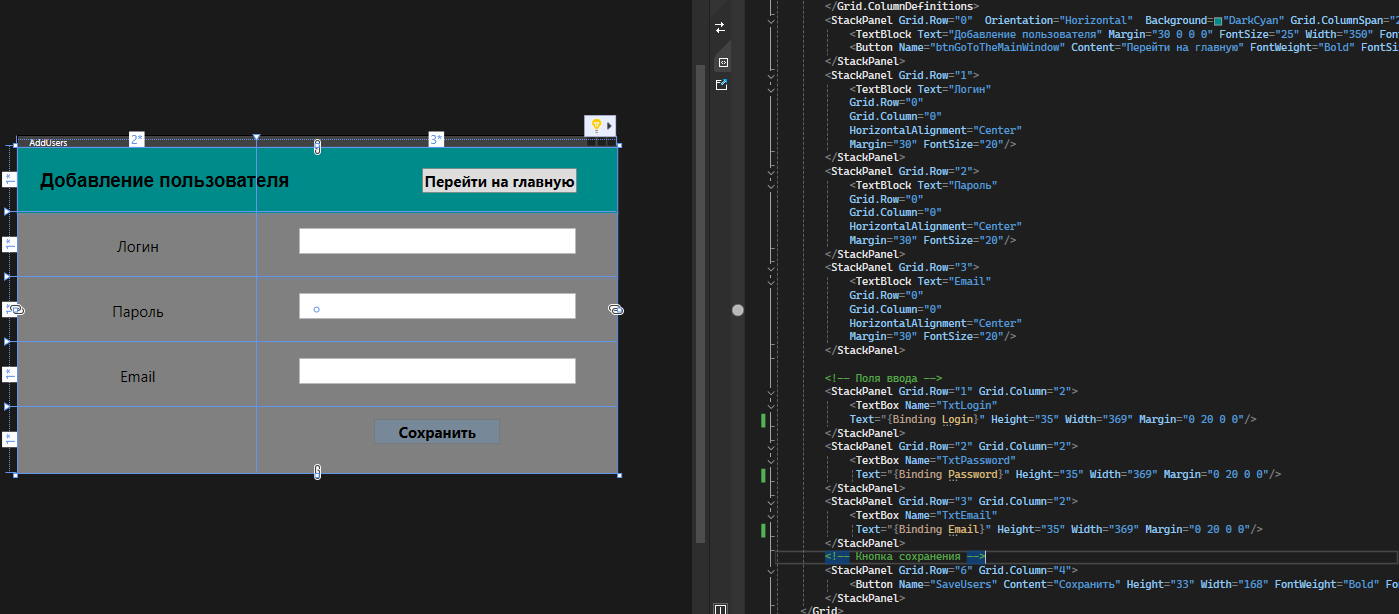
AddUsers 

рис.15 «Окно добавление пользователи»

# 4. Разработка программных модулей

Структура проекта

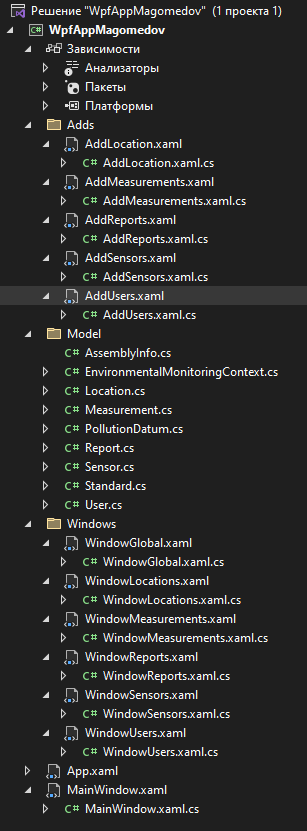


рис.16 «Структура программных модулей»

**Авторизация пользователя на странице MainWindow**

MainWindow.cs  
using System.Text;

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 WpfAppMagomedov.Windows;

namespace WpfAppMagomedov

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private bool AuthenticateUser(string login, string password)

{

using (EnvironmentalMonitoringContext db = new EnvironmentalMonitoringContext())

{

var user = db.Users.FirstOrDefault(e => e.Login == login && e.Password == password);

return user != null;

}

}

private bool AuthenticateUserLogin(string login)

{

using (EnvironmentalMonitoringContext db = new EnvironmentalMonitoringContext())

{

var user = db.Users.FirstOrDefault(e => e.Login == login);

return user != null;

}

}

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

{

string login = LoginTextBox.Text;

string password = PasswordBox.Password;

if (AuthenticateUser(login, password))

{

WindowGlobal windowGlobal = new WindowGlobal();

windowGlobal.Show();

this.Close();

}

else

{

if (AuthenticateUserLogin(login))

{

MessageBox.Show("Неверный пароль.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

MessageBox.Show("Пользователь не найден.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

}

}

}

**Главное меню на окне WindowGlobal**

WindowGlobal.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.Shapes;

namespace WpfAppMagomedov.Windows

{

/// <summary>

/// Логика взаимодействия для WindowGlobal.xaml

/// </summary>

public partial class WindowGlobal : Window

{

public WindowGlobal()

{

InitializeComponent();

}

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

{

WindowSensors windowSensors = new WindowSensors();

windowSensors.Show();

Close();

}

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

{

WindowUsers windowUsers = new WindowUsers();

windowUsers.Show();

this.Close();

}

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

{

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

this.Close();

}

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

{

WindowMeasurements windowMeasurements = new WindowMeasurements();

windowMeasurements.Show();

this.Close();

}

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

{

WindowLocations windowLocations = new WindowLocations();

windowLocations.Show();

this.Close();

}

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

{

WindowReports windowReports = new WindowReports();

windowReports.Show();

this.Close();

}

}

}

**Окно с местоположениями WindowLocations**

WindowLocations.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.Shapes;

using WpfAppMagomedov.Adds;

namespace WpfAppMagomedov.Windows

{

public partial class WindowLocations : Window

{

public EnvironmentalMonitoringContext db = new EnvironmentalMonitoringContext();

public WindowLocations()

{

InitializeComponent();

LoadData();

}

private void LoadData()

{

dtgLocations.ItemsSource = db.Locations.ToList();

cmbLongitude.ItemsSource = db.Locations.Select(l => l.Latitude).Distinct().ToList();

cmbDescription.ItemsSource = db.Locations.Select(l => l.Description).Distinct().ToList();

}

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

{

AddLocation addLocation = new AddLocation();

addLocation.Show();

Close();

}

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

{

WindowGlobal windowGlobal = new WindowGlobal();

windowGlobal.Show();

Close();

}

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

{

var selectedLocation = dtgLocations.SelectedItem as Location;

if (selectedLocation != null)

{

db.Locations.Remove(selectedLocation);

db.SaveChanges();

LoadData();

}

else

{

MessageBox.Show("Выберите местоположение для удаления.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

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

{

string searchText = txtSearch.Text.ToLower();

dtgLocations.ItemsSource = db.Locations

.Where(l => l.LocationName.ToLower().Contains(searchText))

.ToList();

}

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

{

dtgLocations.ItemsSource = db.Locations.OrderBy(l => l.LocationName).ToList();

}

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

{

dtgLocations.ItemsSource = db.Locations.OrderByDescending(l => l.LocationName).ToList();

}

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

{

if (cmbLongitude.SelectedItem != null)

{

double selectedLatitude = (double)cmbLongitude.SelectedItem;

dtgLocations.ItemsSource = db.Locations

.Where(l => l.Latitude == selectedLatitude)

.ToList();

}

}

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

{

if (cmbDescription.SelectedItem != null)

{

string selectedDescription = cmbDescription.SelectedItem.ToString();

dtgLocations.ItemsSource = db.Locations

.Where(l => l.Description == selectedDescription)

.ToList();

}

}

}

}

**Окно с измерениями WindowMeasurements**

WindowMeasurements.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.Shapes;

using WpfAppMagomedov.Adds;

namespace WpfAppMagomedov.Windows

{

/// <summary>

/// Логика взаимодействия для WindowMeasurements.xaml

/// </summary>

public partial class WindowMeasurements : Window

{

public EnvironmentalMonitoringContext db = new EnvironmentalMonitoringContext();

public WindowMeasurements()

{

InitializeComponent();

LoadData();

}

private void LoadData()

{

dtgMeasurements.ItemsSource = db.Measurements.ToList();

cmbUnitOfMeasurement.ItemsSource = db.Measurements

.Select(m => m.UnitOfMeasurement)

.Distinct()

.ToList();

}

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

{

AddMeasurements addMeasurements = new AddMeasurements();

addMeasurements.Show();

Close();

}

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

{

WindowGlobal windowGlobal = new WindowGlobal();

windowGlobal.Show();

Close();

}

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

{

if (cmbUnitOfMeasurement.SelectedItem != null)

{

string selectedUnit = cmbUnitOfMeasurement.SelectedItem.ToString();

dtgMeasurements.ItemsSource = db.Measurements

.Where(m => m.UnitOfMeasurement == selectedUnit)

.ToList();

}

}

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

{

dtgMeasurements.ItemsSource = db.Measurements

.OrderByDescending(m => m.Value)

.ToList();

}

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

{

dtgMeasurements.ItemsSource = db.Measurements

.OrderBy(m => m.Value)

.ToList();

}

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

{

var selectedMeasurement = dtgMeasurements.SelectedItem as Measurement;

if (selectedMeasurement != null)

{

db.Measurements.Remove(selectedMeasurement);

db.SaveChanges();

LoadData();

}

else

{

MessageBox.Show("Выберите измерение для удаления.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

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

{

string searchText = txtSearch.Text.ToLower();

dtgMeasurements.ItemsSource = db.Measurements

.Where(m => m.Parameter.ToLower().Contains(searchText))

.ToList();

}

}

}

**Окно с отчетами WindowReports**

WindowReports.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.Shapes;

using WpfAppMagomedov.Adds;

namespace WpfAppMagomedov.Windows

{

/// <summary>

/// Логика взаимодействия для WindowReports.xaml

/// </summary>

public partial class WindowReports : Window

{

public EnvironmentalMonitoringContext db = new EnvironmentalMonitoringContext();

public WindowReports()

{

InitializeComponent();

LoadData();

}

private void LoadData()

{

// Загружаем данные в DataGrid

dtgReports.ItemsSource = db.Reports.ToList();

// Заполняем ComboBox параметрами для фильтрации

cmbParameter.ItemsSource = db.Reports

.Select(r => r.Parameter)

.Distinct()

.ToList();

}

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

{

AddReports addReports = new AddReports();

addReports.Show();

Close();

}

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

{

WindowGlobal windowGlobal = new WindowGlobal();

windowGlobal.Show();

Close();

}

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

{

AddReports addReports = new AddReports();

addReports.Show();

Close();

}

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

{

if (dtgReports.SelectedItem is Report selectedReport)

{

MessageBoxResult result = MessageBox.Show(

"Вы уверены, что хотите удалить выбранный отчет?",

"Удаление",

MessageBoxButton.YesNo,

MessageBoxImage.Warning);

if (result == MessageBoxResult.Yes)

{

db.Reports.Remove(selectedReport);

db.SaveChanges();

LoadData();

}

}

else

{

MessageBox.Show("Выберите отчет для удаления.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

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

{

dtgReports.ItemsSource = db.Reports

.OrderBy(r => r.Parameter)

.ToList();

}

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

{

dtgReports.ItemsSource = db.Reports

.OrderByDescending(r => r.Parameter)

.ToList();

}

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

{

string searchText = txtSearch.Text.ToLower();

dtgReports.ItemsSource = db.Reports

.Where(r => r.Parameter.ToLower().Contains(searchText))

.ToList();

}

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

{

if (cmbParameter.SelectedItem is string selectedParameter)

{

dtgReports.ItemsSource = db.Reports

.Where(r => r.Parameter == selectedParameter)

.ToList();

}

}

}

}

**Окно с датчиками AddSensors**

AddSensors.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.Shapes;

using WpfAppMagomedov.Adds;

namespace WpfAppMagomedov.Windows

{

/// <summary>

/// Логика взаимодействия для WindowSensors.xaml

/// </summary>

public partial class WindowSensors : Window

{

public EnvironmentalMonitoringContext db = new EnvironmentalMonitoringContext();

public WindowSensors()

{

InitializeComponent();

LoadData();

}

private void LoadData()

{

// Загружаем данные в DataGrid

dtgSensors.ItemsSource = db.Sensors.ToList();

// Заполняем ComboBox статусами для фильтрации

cmbStatus.ItemsSource = db.Sensors

.Select(s => s.Status)

.Distinct()

.ToList();

}

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

{

WindowGlobal windowGlobal = new WindowGlobal();

windowGlobal.Show();

Close();

}

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

{

AddSensors addSensors = new AddSensors();

addSensors.Show();

Close();

}

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

{

if (dtgSensors.SelectedItem is Sensor selectedSensor)

{

MessageBoxResult result = MessageBox.Show(

"Вы уверены, что хотите удалить выбранный датчик?",

"Удаление",

MessageBoxButton.YesNo,

MessageBoxImage.Warning);

if (result == MessageBoxResult.Yes)

{

db.Sensors.Remove(selectedSensor);

db.SaveChanges();

LoadData();

}

}

else

{

MessageBox.Show("Выберите датчик для удаления.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

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

{

string searchText = txtSearch.Text.ToLower();

dtgSensors.ItemsSource = db.Sensors

.Where(s => s.SensorType.ToLower().Contains(searchText))

.ToList();

}

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

{

dtgSensors.ItemsSource = db.Sensors

.OrderBy(s => s.InstallationDate)

.ToList();

}

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

{

dtgSensors.ItemsSource = db.Sensors

.OrderByDescending(s => s.InstallationDate)

.ToList();

}

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

{

if (cmbStatus.SelectedItem is string selectedStatus)

{

dtgSensors.ItemsSource = db.Sensors

.Where(s => s.Status == selectedStatus)

.ToList();

}

}

}

}

**Окно с пользователями AddUsers**

AddUsers.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.Shapes;

using WpfAppMagomedov.Adds;

namespace WpfAppMagomedov.Windows

{

public partial class WindowUsers : Window

{

public EnvironmentalMonitoringContext db = new EnvironmentalMonitoringContext();

public WindowUsers()

{

InitializeComponent();

dtgUsers.ItemsSource = db.Users.ToList(); // Заполняем DataGrid пользователями

}

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

{

AddUsers addUsers = new AddUsers();

addUsers.Show();

Close();

}

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

{

WindowGlobal windowGlobal = new WindowGlobal();

windowGlobal.Show();

this.Close();

}

// Обработчик для поиска по логину

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

{

string searchText = txtSearch.Text.ToLower(); // Приводим к нижнему регистру для нечувствительности к регистру

var filteredUsers = db.Users.Where(u => u.Login.ToLower().Contains(searchText)).ToList();

dtgUsers.ItemsSource = filteredUsers;

}

// Обработчик для поиска по электронной почте

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

{

string searchEmail = txtEmail.Text.ToLower();

var filteredUsers = db.Users.Where(u => u.Email.ToLower().Contains(searchEmail)).ToList();

dtgUsers.ItemsSource = filteredUsers;

}

// Сортировка по логину (по возрастанию)

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

{

var sortedUsers = db.Users.OrderBy(u => u.Login).ToList();

dtgUsers.ItemsSource = sortedUsers;

}

// Сортировка по логину (по убыванию)

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

{

var sortedUsers = db.Users.OrderByDescending(u => u.Login).ToList();

dtgUsers.ItemsSource = sortedUsers;

}

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

{

// Получаем выделенного пользователя

var selectedUser = dtgUsers.SelectedItem as User;

if (selectedUser != null)

{

db.Users.Remove(selectedUser); // Удаляем пользователя из базы

db.SaveChanges(); // Сохраняем изменения в базе данных

// Обновляем DataGrid

dtgUsers.ItemsSource = db.Users.ToList();

}

}

}

}

**Окно добавление местоположения AddLocations**

AddLocations.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.Shapes;

using WpfAppMagomedov.Windows;

namespace WpfAppMagomedov.Adds

{

/// <summary>

/// Логика взаимодействия для AddLocation.xaml

/// </summary>

public partial class AddLocation : Window

{

public Location location { get; set; }

public AddLocation()

{

InitializeComponent();

location = new Location();

this.DataContext = location; // Устанавливаем контекст данных

}

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

{

}

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

{

}

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

{

}

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

{

}

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

{

try

{

// Получаем значения из текстовых полей

string locationName = TxtLocationName.Text;

string latitudeText = TxtLatitude.Text;

string longitudeText = TxtLongitude.Text;

string description = TxtDescription.Text;

// Проверка ввода: убедитесь, что все обязательные поля заполнены корректно

if (string.IsNullOrWhiteSpace(locationName) ||

string.IsNullOrWhiteSpace(latitudeText) ||

string.IsNullOrWhiteSpace(longitudeText) ||

!double.TryParse(latitudeText, out double latitude) ||

!double.TryParse(longitudeText, out double longitude))

{

MessageBox.Show("Пожалуйста, заполните все поля корректно (Широта и Долгота должны быть числовыми).", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

// Создание нового местоположения

Location newLocation = new Location

{

LocationName = locationName,

Latitude = latitude,

Longitude = longitude,

Description = description

};

// Сохранение в базу данных

using (var db = new EnvironmentalMonitoringContext()) // Используйте правильный контекст

{

db.Locations.Add(newLocation);

db.SaveChanges();

}

// Уведомление об успешном сохранении

MessageBox.Show("Местоположение успешно сохранено!", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

// Открытие окна с новым списком местоположений (или возвращение на главную страницу)

WindowLocations windowLocations = new WindowLocations();

windowLocations.Show();

Close();

}

catch (Exception ex)

{

// Обработка ошибок

MessageBox.Show($"Ошибка при сохранении: {ex.Message}\n{ex.InnerException?.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

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

{

}

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

{

WindowGlobal windowGlobal = new WindowGlobal();

windowGlobal.Show();

Close();

}

private void SaveLocation\_Click\_2(object sender, RoutedEventArgs e)

{

}

}

}

**Окно добавление измерений AddMeasurement**

AddMeasurement.cs

using System;

using System.Windows;

using WpfAppMagomedov.Windows;

namespace WpfAppMagomedov.Adds

{

/// <summary>

/// Логика взаимодействия для AddMeasurements.xaml

/// </summary>

public partial class AddMeasurements : Window

{

public AddMeasurements()

{

InitializeComponent();

}

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

{

}

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

{

}

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

{

}

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

{

}

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

{

}

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

{

WindowGlobal windowGlobal = new WindowGlobal();

windowGlobal.Show();

Close();

}

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

{

try

{

// Получаем значения из полей ввода

string parameter = TxtParameter.Text;

string valueText = TxtValue.Text;

string unitOfMeasurement = TxtUnitOfMeasurement.Text;

string measurementDateTimeText = TxtMeasurementDateTime.Text;

// Проверка на пустые поля и корректность введенных данных

if (string.IsNullOrWhiteSpace(parameter) ||

string.IsNullOrWhiteSpace(valueText) ||

string.IsNullOrWhiteSpace(unitOfMeasurement) ||

!double.TryParse(valueText, out double value) ||

!DateTime.TryParse(measurementDateTimeText, out DateTime measurementDateTime))

{

MessageBox.Show("Пожалуйста, заполните все поля корректно.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

// Преобразуем DateTime в DateOnly

DateOnly measurementDate = DateOnly.FromDateTime(measurementDateTime);

// Создаем новый объект Measurement

Measurement newMeasurement = new Measurement

{

Parameter = parameter,

Value = value,

UnitOfMeasurement = unitOfMeasurement,

MeasurementDateTime = measurementDate // Используем DateOnly

};

// Сохранение в базу данных

using (var db = new EnvironmentalMonitoringContext())

{

db.Measurements.Add(newMeasurement);

db.SaveChanges();

}

// Уведомление об успешном сохранении

MessageBox.Show("Измерение успешно сохранено!", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

// После сохранения можно закрыть окно или выполнить другие действия

WindowMeasurements windowMeasurements = new WindowMeasurements();

windowMeasurements.Show();

this.Close();

}

catch (Exception ex)

{

// Обработка ошибок

MessageBox.Show($"Ошибка при сохранении: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

}

**Окно добавление измерений AddReports**

AddReports.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.Shapes;

using WpfAppMagomedov.Windows;

namespace WpfAppMagomedov.Adds

{

/// <summary>

/// Логика взаимодействия для AddReports.xaml

/// </summary>

public partial class AddReports : Window

{

public Report report { get; set; }

public AddReports()

{

InitializeComponent();

report = new Report();

this.DataContext = report; // Устанавливаем контекст данных

}

private void TxtReportingPeriod\_SelectionChanged(object sender, RoutedEventArgs e) { }

private void TxtParameter\_SelectionChanged(object sender, RoutedEventArgs e) { }

private void TxtAverageValue\_SelectionChanged(object sender, RoutedEventArgs e) { }

private void TxtMaximumValue\_SelectionChanged(object sender, RoutedEventArgs e) { }

private void TxtMinimumValue\_SelectionChanged(object sender, RoutedEventArgs e) { }

private void TxtMeasurementCount\_SelectionChanged(object sender, RoutedEventArgs e) { }

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

{

// Извлекаем данные из полей формы

var reportingPeriod = TxtReportingPeriod.Text?.Trim();

var parameter = TxtParameter.Text?.Trim();

var averageValueText = TxtAverageValue.Text?.Trim();

var maximumValueText = TxtMaximumValue.Text?.Trim();

var minimumValueText = TxtMinimumValue.Text?.Trim();

var measurementCountText = TxtMeasurementCount.Text?.Trim();

// Если одно из полей пустое, ничего не делаем

if (string.IsNullOrWhiteSpace(reportingPeriod) ||

string.IsNullOrWhiteSpace(parameter) ||

string.IsNullOrWhiteSpace(averageValueText) ||

string.IsNullOrWhiteSpace(maximumValueText) ||

string.IsNullOrWhiteSpace(minimumValueText) ||

string.IsNullOrWhiteSpace(measurementCountText))

{

MessageBox.Show("Пожалуйста, заполните все поля.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

// Преобразуем данные из текстовых полей в типы данных

if (!double.TryParse(averageValueText, out double averageValue) ||

!double.TryParse(maximumValueText, out double maximumValue) ||

!double.TryParse(minimumValueText, out double minimumValue) ||

!int.TryParse(measurementCountText, out int measurementCount))

{

MessageBox.Show("Пожалуйста, введите корректные значения для числовых полей.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

try

{

// Создаем новый объект Report с полученными значениями

var newReport = new Report

{

ReportingPeriod = reportingPeriod,

Parameter = parameter,

AverageValue = averageValue,

MaximumValue = maximumValue,

MinimumValue = minimumValue,

MeasurementCount = measurementCount

};

// Сохраняем отчет в базе данных

using (var context = new EnvironmentalMonitoringContext())

{

context.Reports.Add(newReport);

context.SaveChanges();

}

MessageBox.Show("Отчет успешно сохранен.", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

// Закрываем окно после сохранения

WindowReports windowReports = new WindowReports();

windowReports.Show();

this.Close();

}

catch (Exception ex)

{

MessageBox.Show($"Произошла ошибка при сохранении отчета: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

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

{

WindowGlobal windowGlobal = new WindowGlobal();

windowGlobal.Show();

Close();

}

}

}

**Окно добавление измерений AddSensors**

AddSensors.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.Shapes;

using WpfAppMagomedov.Windows;

namespace WpfAppMagomedov.Adds

{

public partial class AddSensors : Window

{

public AddSensors()

{

InitializeComponent();

}

// Обработчик кнопки "Сохранить"

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

{

// Преобразуем строки в DateOnly (предположим, что они в формате "yyyy-MM-dd")

DateOnly installationDate;

DateOnly calibrationDate;

bool isInstallationDateValid = DateOnly.TryParse(TxtInstallationDate.Text, out installationDate);

bool isCalibrationDateValid = DateOnly.TryParse(TxtCalibrationDate.Text, out calibrationDate);

// Проверяем, что все поля заполнены и даты валидны

if (string.IsNullOrWhiteSpace(TxtSensorType.Text) ||

string.IsNullOrWhiteSpace(TxtLocation.Text) ||

string.IsNullOrWhiteSpace(TxtStatus.Text) ||

!isInstallationDateValid ||

!isCalibrationDateValid)

{

MessageBox.Show("Пожалуйста, заполните все поля корректно!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

// Создаём объект Sensor и заполняем его данными из полей

var newSensor = new Sensor

{

SensorType = TxtSensorType.Text.Trim(),

Location = TxtLocation.Text.Trim(),

InstallationDate = installationDate, // Используем DateOnly

CalibrationDate = calibrationDate, // Используем DateOnly

Status = TxtStatus.Text.Trim()

};

try

{

// Сохранение данных в базу данных (обновите строку подключения и используйте свой контекст)

using (var context = new EnvironmentalMonitoringContext())

{

context.Sensors.Add(newSensor); // Добавляем новый датчик

context.SaveChanges(); // Сохраняем изменения в базу данных

}

// Уведомляем пользователя об успешном сохранении

MessageBox.Show("Датчик успешно добавлен!", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

// Закрытие окна после успешного сохранения

WindowSensors windowSensors = new WindowSensors();

windowSensors.Show();

this.Close();

}

catch (Exception ex)

{

// Логируем ошибку, если произошла ошибка при сохранении

MessageBox.Show($"Произошла ошибка при добавлении датчика: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

// Кнопка для перехода на главную страницу

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

{

WindowGlobal windowGlobal = new WindowGlobal();

windowGlobal.Show();

Close();

}

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

{

}

}

}

**Окно добавление измерений AddUsers**

AddUsers.cs

using System;

using System.Linq;

using System.Text.RegularExpressions;

using System.Windows;

using System.Security.Cryptography;

using System.Text;

using WpfAppMagomedov.Windows;

using Microsoft.EntityFrameworkCore;

namespace WpfAppMagomedov.Adds

{

/// <summary>

/// Логика взаимодействия для AddUsers.xaml

/// </summary>

public partial class AddUsers : Window

{

public AddUsers()

{

InitializeComponent();

}

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

{

}

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

{

}

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

{

}

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

{

string login = TxtLogin.Text.Trim();

string password = TxtPassword.Text.Trim();

string email = TxtEmail.Text.Trim();

// Проверка на пустые поля

if (string.IsNullOrWhiteSpace(login) || string.IsNullOrWhiteSpace(password) || string.IsNullOrWhiteSpace(email))

{

MessageBox.Show("Все поля должны быть заполнены!");

return;

}

var newUser = new User

{

Login = login,

Password = password,

Email = email

};

try

{

using (var context = new EnvironmentalMonitoringContext())

{

context.Users.Add(newUser);

context.SaveChanges();

}

MessageBox.Show("Пользователь успешно добавлен!");

}

catch (Exception ex)

{

// Выводим полное описание ошибки

MessageBox.Show($"Произошла ошибка при сохранении: {ex.Message}\n{ex.StackTrace}");

}

}

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

{

WindowGlobal windowGlobal = new WindowGlobal();

windowGlobal.Show();

Close();

}

}

}

# 5. Разработка Api

API — это набор правил и и протоколов, которые позволяют различным системам обмениваться информацией между собой. API — это общий термин, который применяется в контексте совершенно разных систем.

Swagger — это инструмент, который помогает разработчикам создавать, документировать и проверять API.

Для работы с API использовался шаблон Веб-API ASP.NET Core (Майкрософт)

Структура

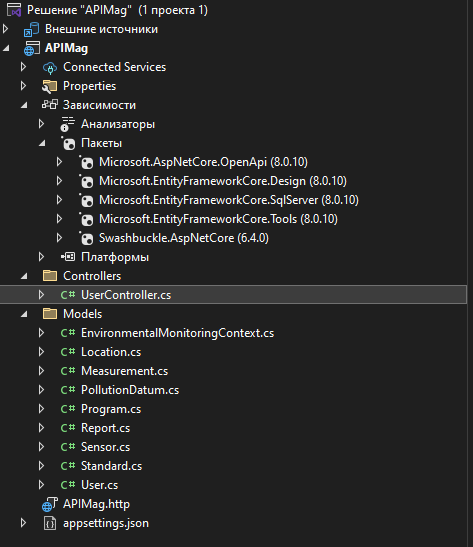


рис.13 «структура API»

Код AddController:

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using APIMag.Models;

namespace APIMag.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class UserController : ControllerBase

{

private readonly EnvironmentalMonitoringContext \_context;

public UserController(EnvironmentalMonitoringContext context)

{

\_context = context;

}

// GET: api/user

[HttpGet]

public async Task<ActionResult<IEnumerable<User>>> GetAll()

{

return await \_context.Users.ToListAsync();

}

// GET: api/user/{id}

[HttpGet("{Id}")]

public async Task<ActionResult<User>> GetById(int Id)

{

var users = await \_context.Users.FindAsync(Id);

if (users is null)

return NotFound();

return users;

}

// POST: api/users

[HttpPost]

public async Task<IActionResult> Create(User user)

{

try

{

\_context.Users.Add(user);

await \_context.SaveChangesAsync();

return CreatedAtAction(nameof(GetById), new { Id = user.UserId }, user);

}

catch (DbUpdateException)

{

// Обработка исключения

return BadRequest("Ошибка при создании Пользователя.");

}

}

// PUT: api/users/{id}

[HttpPut("{Id}")]

public async Task<IActionResult> PutUser(int Id, User user)

{

if (Id != user.UserId)

{

return BadRequest();

}

\_context.Entry(user).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!UserExists(Id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

private bool UserExists(int Id)

{

return \_context.Users.Any(e => e.UserId == Id);

}

// DELETE: api/user/{id}

[HttpDelete("{Id}")]

public async Task<IActionResult> DeleteUser(int Id)

{

var user = await \_context.Users.FindAsync(Id);

if (user == null)

{

return NotFound();

}

\_context.Users.Remove(user);

await \_context.SaveChangesAsync();

return NoContent();

}

}

}

Проверка работы

1. GET all

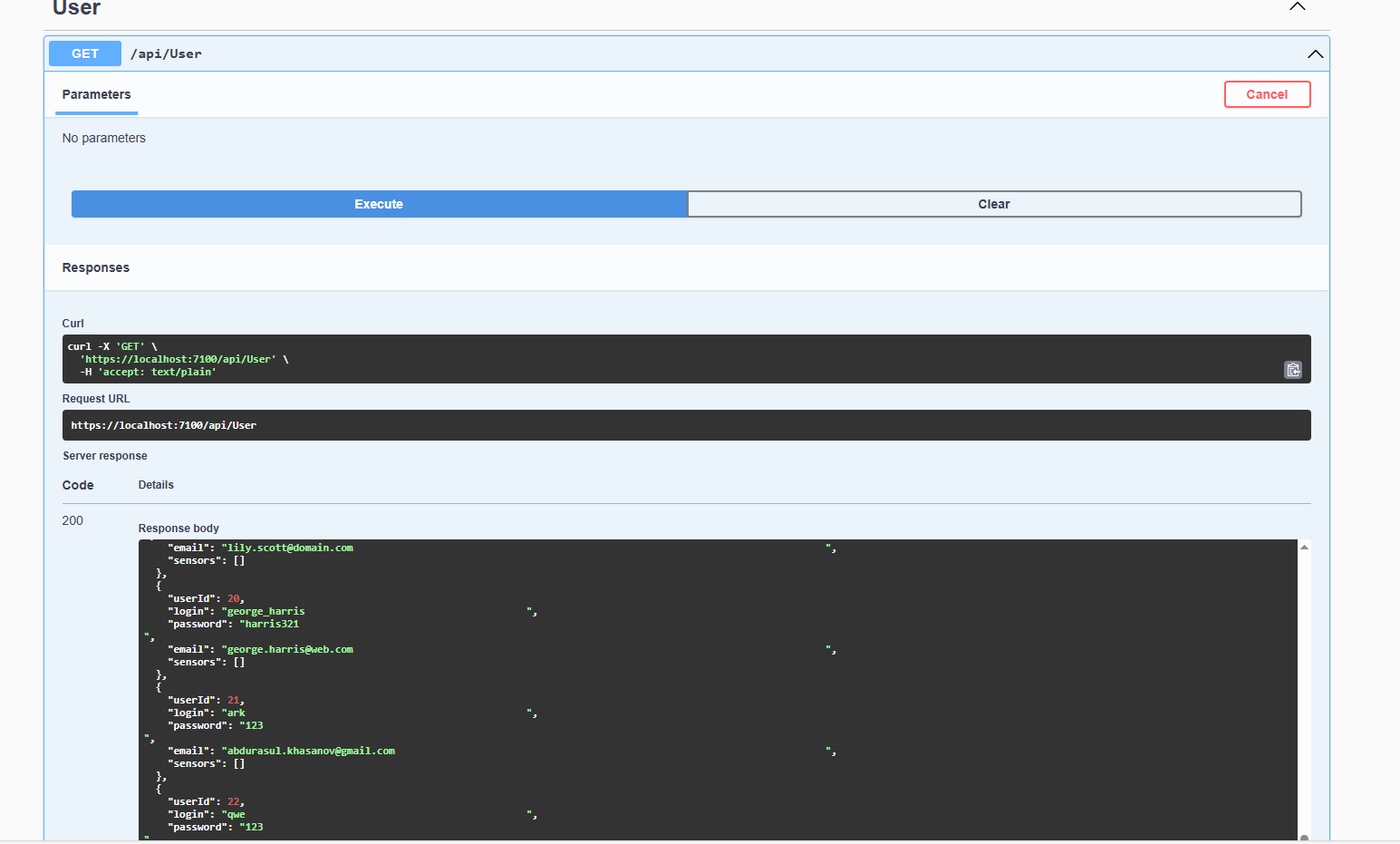


рис.14 «Get all»

2. POST

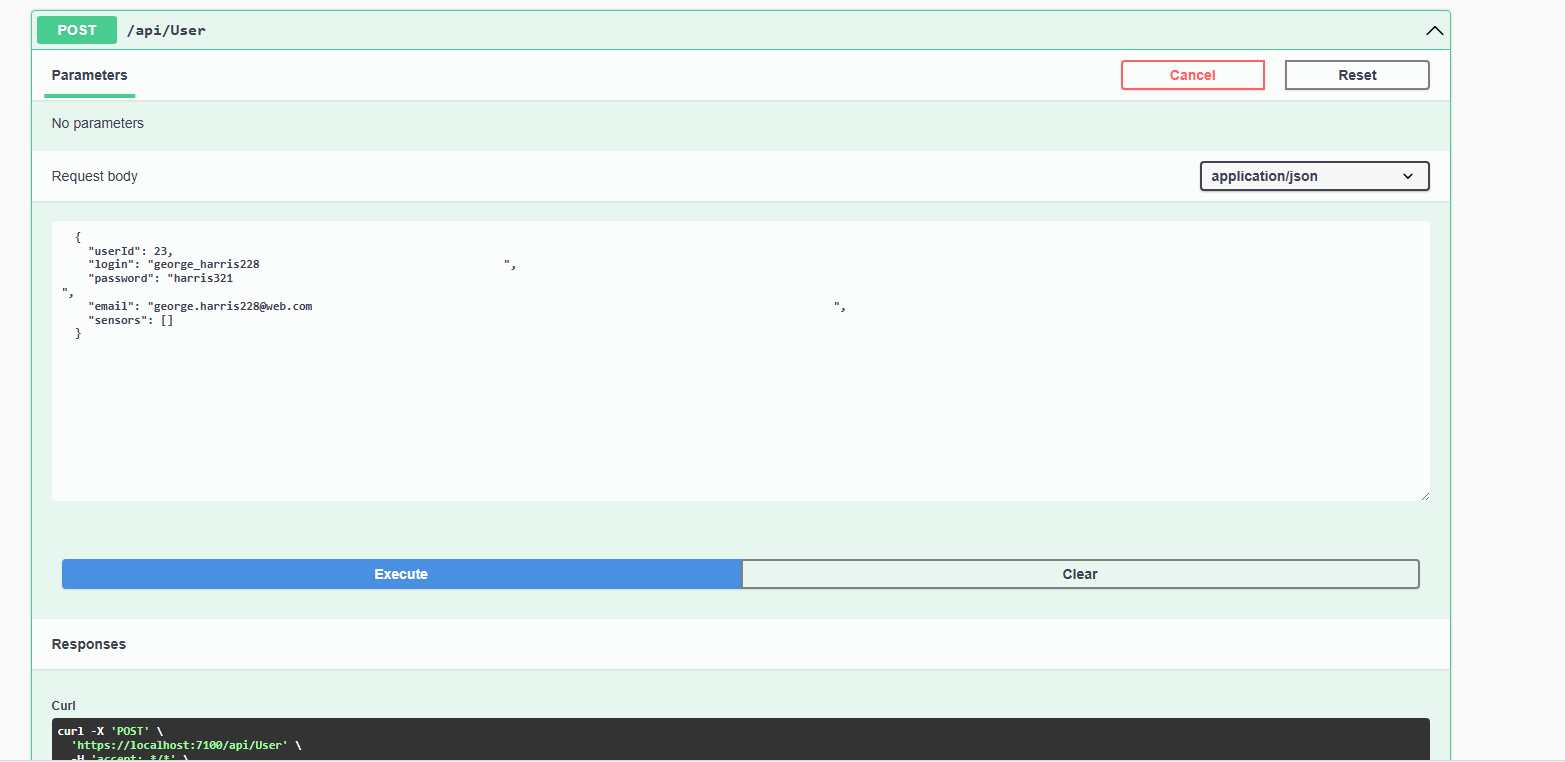


рис.15 «Post»

3.GET id

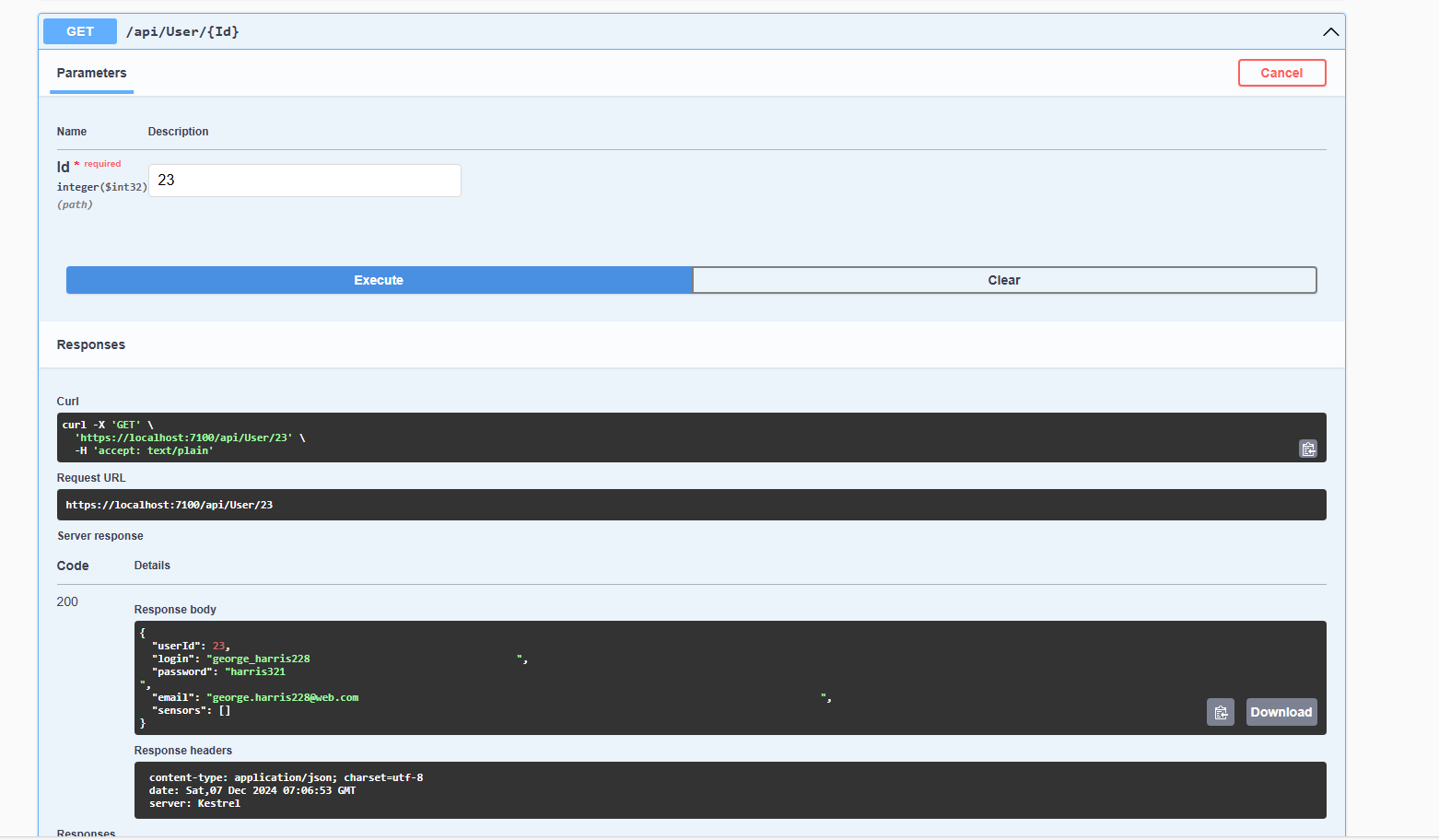


рис.16 «Get id»

4. PUT

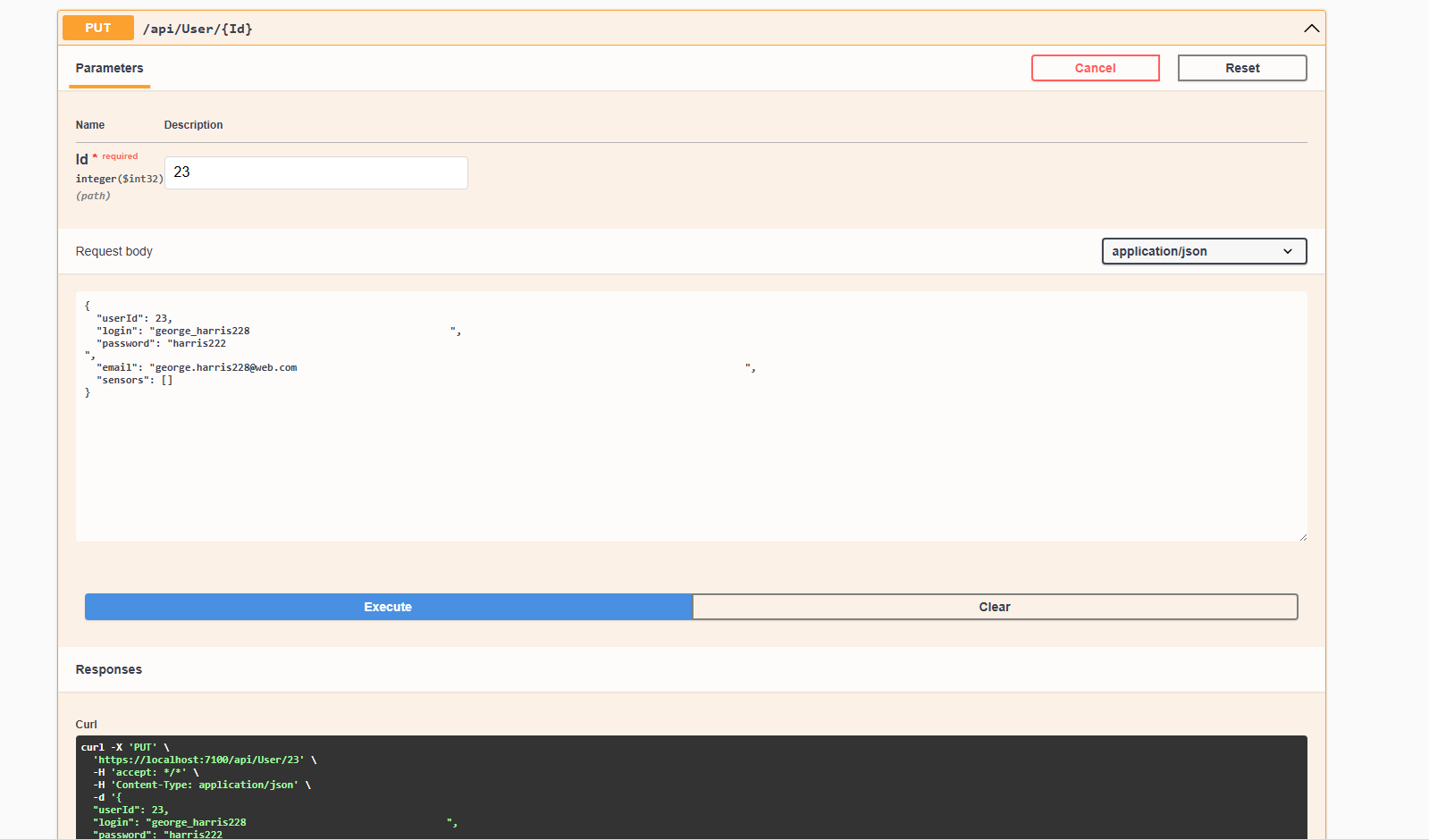


рис.17 «Put»

5.DELETE

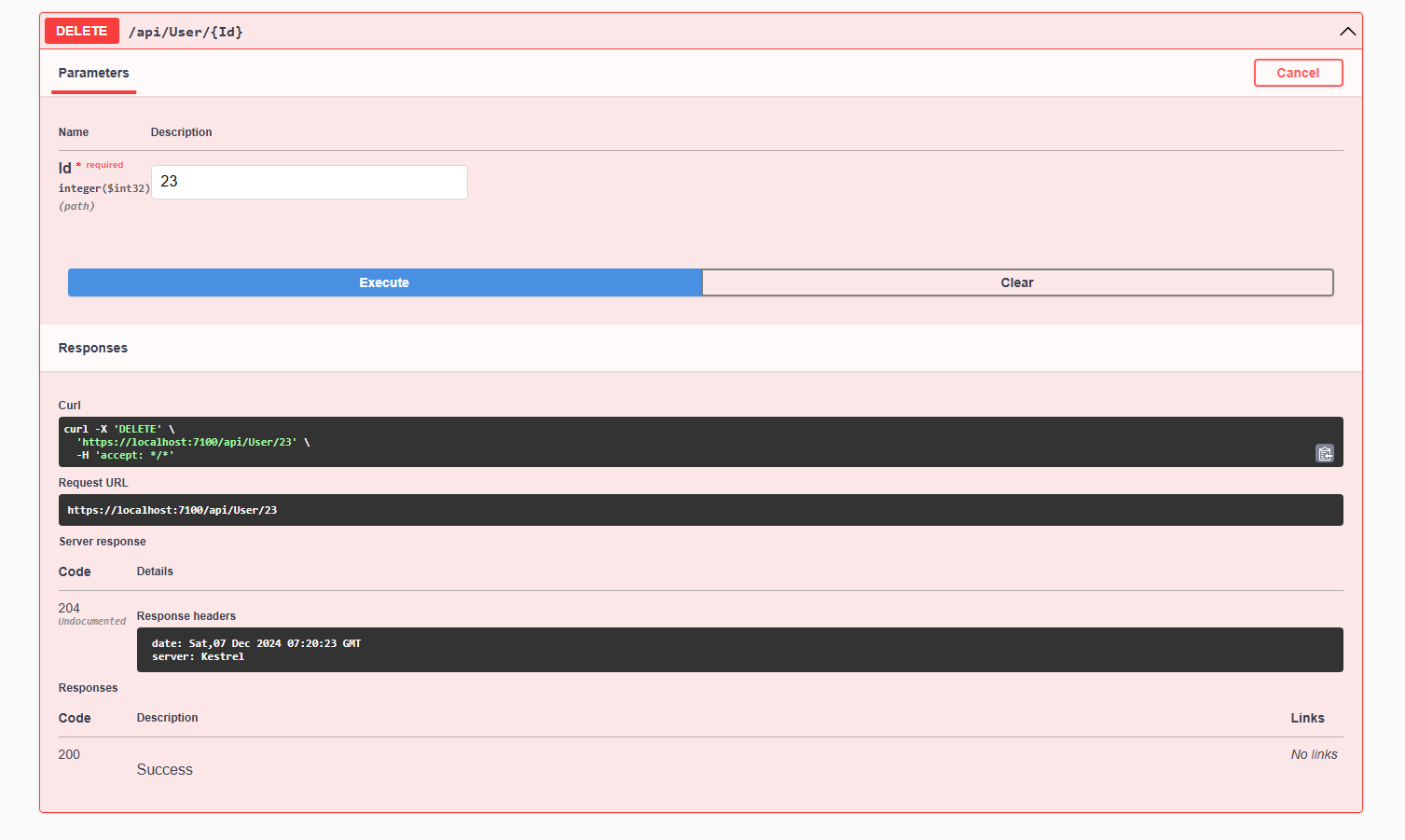


рис.18 «Delete»

# 6. Библиотека классов

Для разработки библиотеки классов использовался шаблон Библиотека классов (Майкрософт)

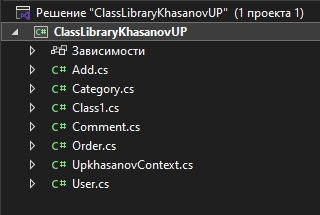


рис.19 «Структура библиотеки классов»

Класс в библиотеки используется в качестве логики и использован для тестов работы авторизации.

Sensor  
using System;

using System.Collections.Generic;

namespace ClassLibrary1;

public partial class Sensor

{

public int SensorId { get; set; }

public string SensorType { get; set; } = null!;

public string Location { get; set; } = null!;

public DateOnly? InstallationDate { get; set; }

public DateOnly? CalibrationDate { get; set; }

public string Status { get; set; } = null!;

public int? StandardId { get; set; }

public int? UserId { get; set; }

public virtual ICollection<Measurement> Measurements { get; set; } = new List<Measurement>();

public virtual ICollection<PollutionDatum> PollutionData { get; set; } = new List<PollutionDatum>();

public virtual Standard? Standard { get; set; }

public virtual User? User { get; set; }

}

Для тестов использовался шаблон проекта MSTest

Структура приложения MSTest

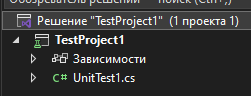


рис.20 «Структура MSTest»

Проект модульного тестирования

using System;

using System.Collections.Generic;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System.Linq; // Для использования ElementAt

namespace TestProject1

{

[TestClass]

public sealed class Test1

{

// Тест на создание объекта Sensor и проверку свойств

[TestMethod]

public void TestCreateSensor()

{

// Arrange

var sensor = new Sensor

{

SensorId = 1,

SensorType = "Temperature",

Location = "Room 101",

InstallationDate = new DateOnly(2023, 12, 9),

CalibrationDate = new DateOnly(2024, 1, 1),

Status = "Active"

};

// Assert

Assert.IsNotNull(sensor);

Assert.AreEqual(1, sensor.SensorId);

Assert.AreEqual("Temperature", sensor.SensorType);

Assert.AreEqual("Room 101", sensor.Location);

Assert.AreEqual(new DateOnly(2023, 12, 9), sensor.InstallationDate);

Assert.AreEqual(new DateOnly(2024, 1, 1), sensor.CalibrationDate);

Assert.AreEqual("Active", sensor.Status);

}

// Тест на добавление Measurement в коллекцию Measurements

[TestMethod]

public void TestAddMeasurementToSensor()

{

// Arrange

var sensor = new Sensor

{

SensorId = 1,

SensorType = "Temperature",

Location = "Room 101",

Status = "Active"

};

var measurement = new Measurement

{

MeasurementId = 1,

Value = 25.5,

Timestamp = new DateOnly(2024, 12, 9)

};

// Act

sensor.Measurements.Add(measurement);

// Assert

Assert.AreEqual(1, sensor.Measurements.Count);

Assert.AreEqual(25.5, sensor.Measurements.ElementAt(0).Value); // Используем ElementAt

}

// Тест на правильность ассоциации объекта User с Sensor

[TestMethod]

public void TestAssociateUserWithSensor()

{

// Arrange

var user = new User

{

UserId = 1,

Username = "testuser"

};

var sensor = new Sensor

{

SensorId = 1,

SensorType = "Humidity",

Location = "Room 102",

Status = "Inactive",

User = user

};

// Assert

Assert.IsNotNull(sensor.User);

Assert.AreEqual(1, sensor.User.UserId);

Assert.AreEqual("testuser", sensor.User.Username);

}

// Тест на установку CalibrationDate

[TestMethod]

public void TestSetCalibrationDate()

{

// Arrange

var sensor = new Sensor

{

SensorId = 1,

SensorType = "Temperature",

Location = "Room 101",

Status = "Active"

};

var calibrationDate = new DateOnly(2024, 1, 10);

sensor.CalibrationDate = calibrationDate;

// Assert

Assert.AreEqual(calibrationDate, sensor.CalibrationDate);

}

// Тест на добавление PollutionDatum в коллекцию PollutionData

[TestMethod]

public void TestAddPollutionDataToSensor()

{

// Arrange

var sensor = new Sensor

{

SensorId = 1,

SensorType = "Air Quality",

Location = "Outside",

Status = "Active"

};

var pollutionDatum = new PollutionDatum

{

PollutionId = 1,

PollutionType = "PM2.5",

Value = 35.6,

MeasurementDate = new DateOnly(2024, 12, 9)

};// Act

sensor.PollutionData.Add(pollutionDatum);

// Assert

Assert.AreEqual(1, sensor.PollutionData.Count);

Assert.AreEqual(35.6, sensor.PollutionData.ElementAt(0).Value); // Используем ElementAt

}

}

// Модели для использования в тестах (замените на реальные)

public class Sensor

{

public int SensorId { get; set; }

public string SensorType { get; set; } = null!;

public string Location { get; set; } = null!;

public DateOnly? InstallationDate { get; set; }

public DateOnly? CalibrationDate { get; set; }

public string Status { get; set; } = null!;

public int? StandardId { get; set; }

public int? UserId { get; set; }

public virtual ICollection<Measurement> Measurements { get; set; } = new List<Measurement>();

public virtual ICollection<PollutionDatum> PollutionData { get; set; } = new List<PollutionDatum>();

public virtual Standard? Standard { get; set; }

public virtual User? User { get; set; }

}

public class Measurement

{

public int MeasurementId { get; set; }

public double Value { get; set; }

public DateOnly Timestamp { get; set; }

}

public class PollutionDatum

{

public int PollutionId { get; set; }

public string PollutionType { get; set; } = null!;

public double Value { get; set; }

public DateOnly MeasurementDate { get; set; }

}

public class User

{

public int UserId { get; set; }

public string Username { get; set; } = null!;

}

public class Standard

{

public int StandardId { get; set; }

public string StandardName { get; set; } = null!;

}

}

Тест проверяет логин и пароль существующего клиента или арендодателя сравнивая их с данными в базе данных

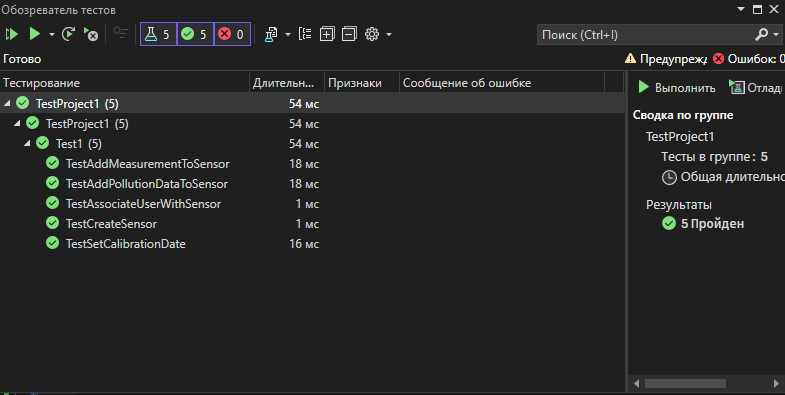


рис.21 «Отчет об удачном завершении теста»

# Заключение

Приложение выполнено в соответствии с техническим заданием, работает корректно и имеет дополнительное приложение API и библиотеку классов.

В дальнейшем я бы переработал поиск по параметру пользователем, добавил бы больший функционал, например фильтрацию погоды. Улучшил бы интерфейс убрав баги стилей и сделал бы его более понятным.