Министерство образования и науки Республики Башкортостан
Государственное автономное профессиональное образовательное учреждение
Уфимский колледж статистики, информатики и вычислительной техники

УТВЕРЖДАЮ
Заместитель директора
по учебной работе
З.З. Курмашева
«__»_____2024 г.

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ УЧЕТА ЗАЯВОК НА ЗАМЕР ОКОН

Пояснительная записка к курсовому проекту
МДК 11.01 Технология разработки и защиты баз данных

Руководитель проекта
Р.Ф. Каримова
«___» _____ 2024 г.

Студент гр. 21П-1
В.В. Майструк
«___» _____ 2024 г.

2024

Министерство образования и науки Республики Башкортостан

Государственное автономное профессиональное образовательное учреждение

Уфимский колледж статистики, информатики и вычислительной техники

УТВЕРЖДАЮ
Заместитель директора
по учебной работе
З.З. Курмашева
«___»_____2024 г.

ЗАДАНИЕ

на курсовой проект студенту дневного отделения, группы 21П-1,
специальности 09.02.07 Информационные системы и программирование
Фамилия, имя, отчество: Майструк Виктория Витальевна
Тема курсового проекта: «Разработка приложения для учета заявок на замер окон».
Текст задания:
при выполнении курсового проекта должны быть решены следующие задачи:
а) спроектирована структура базы данных;
б) разработана структура программы;
в) реализованны функции менеджера: регистрация заявки на замер окон, изменение исполнителя. Также реализованы функции исполнителя: исполнение заявки, расчет замера.
В результате выполнения курсового проекта должны быть представлены:
а) пояснительная записка, состоящая из следующих разделов:
Введение
1 Постановка задачи
2 Экспериментальный раздел
Заключение

Приложения

Список сокращений

Список использованных источников

б) электронный носитель, содержащий разработанный программный продукт;

в) презентация курсового проекта в электронном виде.

Список рекомендуемых источников:

1 Култыгин, О. П. Култыгин, О. П. Администрирование баз данных. СУБД MS SQL Server [Текст] : учеб. пособ. / О. П. Култыгин. - М.: МФПА, 2012. - 232 с.

2 Фуфаев, Э.В. Базы данных [Текст]: учеб. пособ. для студ. учрежд. сред. проф. образования / Э.В. Фуфаев, Д.Э. Фуфаев. - 6-е изд., стер. - М.: Издательский центр «Академия», 2012.- 320 с.- (Среднее профессиональное образование)

3 Википедия [Электронный ресурс] // Свободная энциклопедия. – Режим доступа: http://ru.wikipedia.org/wiki/, свободный

Задание к выполнению получил «31» января 2024 г.

Студент Майструк Виктория Витальевна

Срок окончания «31» мая 2024 г.

Руководитель курсового проекта_____Р.Ф. Каримова

Задание рассмотрено на заседании цикловой комиссии информатики

«11» января 2024 г.

Председатель цикловой комиссии информатики _____О.В.Фатхулова

Министерство образования и науки Республики Башкортостан

Государственное бюджетное профессиональное образовательное учреждение

Уфимский колледж статистики, информатики и вычислительной техники

ЗАКЛЮЧЕНИЕ

на курсовой проект

Студент Майструк Виктория Витальевна

Группа 21П-1

Специальность 09.02.07 Информационные системы и программирование

Тема Проектирование базы данных и разработка приложения для учета заявок на замер окон

Объем курсового проекта:

количество листов пояснительной записки _____

количество листов графической части _____

Заключение о степени соответствия заданию на курсовое проектирование

_____

_____

_____

Характеристика качеств, проявленных студентом при работе над проектом: самостоятельность, дисциплинированность, умение планировать работу и пользоваться литературным материалом и т.д.

_____

_____

_____

_____

Положительные стороны курсового проекта

_____

_____

_____

_____

Недостатки курсового проекта

_____

_____

Характеристика общетехнической и специальной подготовки студента

_____

_____

_____

_____

_____

Заключение и предлагаемая оценка за курсовой проект

_____

_____

Руководитель курсового проекта Каримова Резида Флюновна_____

«____» _____2024 г.

Подпись_____

АННОТАЦИЯ

Пояснительная записка к курсовому проекту содержит постановку и программу решения задачи «Разработка приложения для учета заявок на замер окон».

Программа PlasticWindow.exe написана на языке C# в среде программирования VisualStudio 2022 с использованием системы управления базой данных MySQL Workbench предназначена для работы в операционной системе MSWindows10 и выше, отлажена на данных контрольного примера.

Содержание

ВВЕДЕНИЕ

Просто невозможно, представить себе дом без окон. Первые окошки упоминаются во 2-м тысячелетии до н.э. А потом окно прошло огромный путь развития, превратившись из простого отверстия к сложным системам.

История окон насчитывает больше 4 тысячелетий. Но первые конструкции были максимально примитивны. Их делали в виде отверстия для воздухообмена и естественного освещения.

Уже во времена Помпеи, обнаруживаются первые застекленные окна. Их установили в банях, а размеры составляли вполне приличные 90х120 см.

Из-за сложности производства и дороговизны материалов окошки сначала ставили в храмах, "административных зданиях", дворцах. А самыми востребованными по эстетичности стали витражи – изготовить было проще.

С 1935 года немец Вильгельм Франк запатентовал свое изобретение – поворотно-откидную фурнитуру. А в 1959-м А.Пилкингтон представил изготовление гладкого стекла.

Пришла эпоха пластика. В 1954-м году о себе заявила немецкая компания Dinamit Nobel, поставив в здание первый пластик. Эта компания была детищем того самого Альфреда Нобеля.

Важной вехой в истории пластиковых окон стала возможность окрашивания пластика во все цвета RAL – 179 оттенков, и дополнение дизайна ламинированием с любым оттенком и даже рисунком.

Актуальность данной работы обусловлена тем, что на данный момент пвх окна самый ходовой элемент при строительстве и ремонте жилых помещений имеющие очень обширные конфигурации и наполнение, что затрудняет расчёт замерщика. Существующие программы по интерфейсу интуитивно не достаточно понятны.

Цель курсового проекта – разработка приложения для упрощения работы замерщиков.

Задачами курсового проекта являются:

- описать предметную область;
- разработать структуру базы данных;
- разработать приложение;
- провести тестирование приложения.

1. Постановка задачи

1.1 Описание предметной области

Требуется разработать программу для замерщика пластиковых окон.

Данная информационная система предполагает наличие двух групп пользователей: менеджер и замерщик (исполнитель).

Менеджер имеет возможность:

- просматривать все заявки;
- регистрировать заявку;
- назначать исполнителей;
- находить заявку;
- отслеживать работу.

Замерщик имеет следующие возможности: просматривать конкретно свои заявки, исполнять заявки, редактировать заявки, оформлять отчет.

В базе данных должны храниться следующие справочники: справочники пользователей, справочники дополнительных параметров, справочники дополнительных аксессуаров, а также данные по замерам.

В пользователях хранится следующая информация:

- идентификатор пользователя;
- имя;
- номер телефона;
- почта;
- пароль;
- идентификатор роли.

В роли хранится следующая информация:

- идентификатор роли;
- наименование.

Для того чтобы оформить заявку, менеджеру необходимо заполнить поля формы «Оформление заявки», которая содержит следующие разделы:

– идентификатор заявки;
– имя клиента;
– номер телефона клиента;
– адрес;
– дата;
– имя замерщика.

После ввода данных для оформления заявки всё заносится в базу данных и передается замерщику, который непосредственно поедет на замер.

Сущность «Замеры» содержит следующую информацию:

– идентификатор замера;
– ширина окна;
– высота окна;
– толщина профиля;
– количество створок открывающихся;
– количество створок не открывающихся;
– количество створок поворотных;
– количество створок поворотно-откидных;
– камерность;
– идентификатор дополнительных параметров;
– идентификатор дополнительных аксессуаров;

Любой клиент может выбрать дополнительные параметры к своим окнам.

Для этого в «Дополнительные параметры» вносится следующая информация: идентификатор дополнительных параметров и наименование.

Любой клиент может выбрать дополнительные аксесуары к своим окнам. Для этого в «Дополнительные аксессуары» вносится следующая информация:

– идентификатор дополнительных аксесуаров;

– наименование.

Входными данными являются данные, которые нужны для оформления заявки на замер. Выходными данными является оформленный замер.

1.2 Описание входной информации

Входной информацией являются, справочники дополнительных параметров, дополнительных аксессуаров, зарегестрированная заявка на замер.

Описание входного документа представлено в таблице 1.2.1.

Таблица 1.2.1 – Описание входных документов

Наименование документа (шифр)	Дата поступления документа	Откуда поступает документ
Заявка на замер	При обращении клиента	От диспетчера

Выше представлена таблица вхордных данных.

1.3 Описание выходной информации

Выходной информацией является оформленный отчет:

оформленный отчет замерщика — это документ, содержащий подробную информацию о проведённых замерах, необходимых для выполнения определённых работ.

Описание выходных документов представлено в таблице 1.3.1.

Таблица 1.3.1 – Описание выходных документов

Наименование документа (шифр)	Периодичность выдачи документа	Кол-во экз.	Куда передаются	Поля сортировки	Поля группировки	Итоги
Оформленный отчет замерщика	По требованию заказчика	2	Передается клиенту, второй экземпляр замерщик оставляет себе	-	-	По согласованию оформляется заказ

Шаблоны выходных документов представлены в приложении А.

1.4. Концептуальное моделирование

Концептуальная модель — это некая наглядная диаграмма, нарисованная в принятых обозначениях и подробно показывающая связь между объектами и их характеристиками. Создается концептуальная модель для дальнейшего проектирования базы данных. На концептуальной модели в визуально удобном виде прописываются связи между объектами данных и их характеристиками.

В концептуальной модели есть принятые обозначения элементов. Сущность или объект обозначают прямоугольником, отношения обозначают ромбом, атрибуты объектов, обозначаются овалом. Если сущность связана с отношением, то их связь обозначается прямой линией со стрелкой.

Концептуальная модель базы данных представлена в рисунке 1.4.1

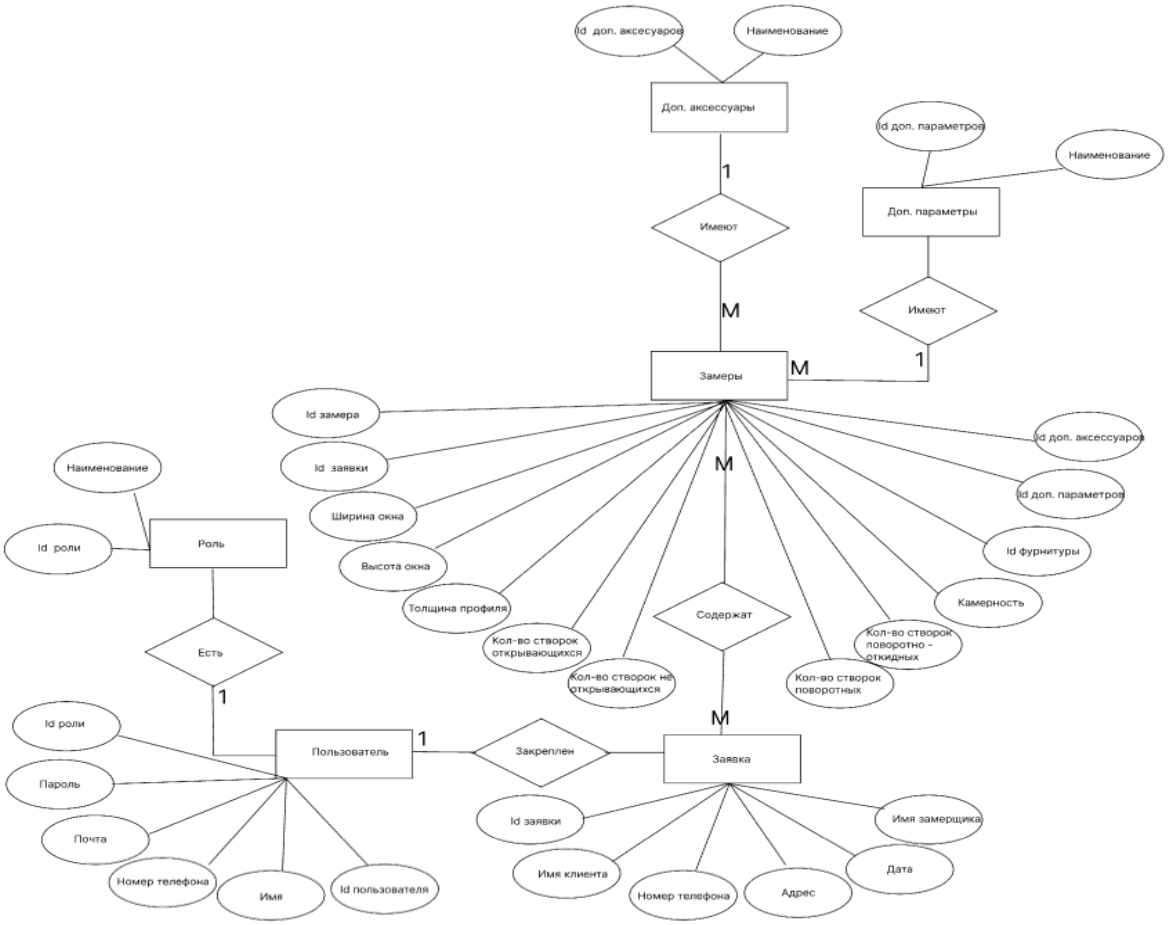

Рисунок 1.4.1 – Концептуальная модель БД

Выше представленна концептуальная модель базы данных.

1.5. Логическое моделирование

При логическом моделировании происходит окончательное определение структуры данных, определяются ограничения, накладываемые на эти данные, целью которых является обеспечить целостность данных. Наиболее распространенной моделью данных является реляционная модель. В этой модели данных каждая сущность представляется в виде таблицы.

Логическое моделирование заключается в переходе от концептуальной модели к взаимосвязанным таблицам. Этот переход состоит из следующих шагов:

1. Преобразование сущностей:

- каждая простая сущность становится таблицей;
- каждый атрибут становится столбцом таблицы;
- уникальный идентификатор сущности становится ключом таблицы.

2. Преобразование связи:

- сущности, связанные обязательной связью один к одному, можно объединить в одну таблицу;
- связи типа один к одному возможные и связи типа один ко многим реализуются путем переноса ключевых атрибутов таблиц, соответствующих сущностей, стоящих со стороны один в таблице соответствующих сущностей, стоящих со стороны многие;
- связи типа многие ко многим реализуются при помощи промежуточных таблиц, содержащих ключевые атрибуты связываемых таблиц в качестве внешних ключей.

Схема данных – это структура базы данных, описанная на формальном языке, поддерживаемом СУБД (системой управления базы данных).

В реляционных базах данных схема определяет таблицы, поля в каждой таблице и ограничения целостности, такие как первичный и внешний ключи.

Схема данных представлена на рисунке 1.5.1.

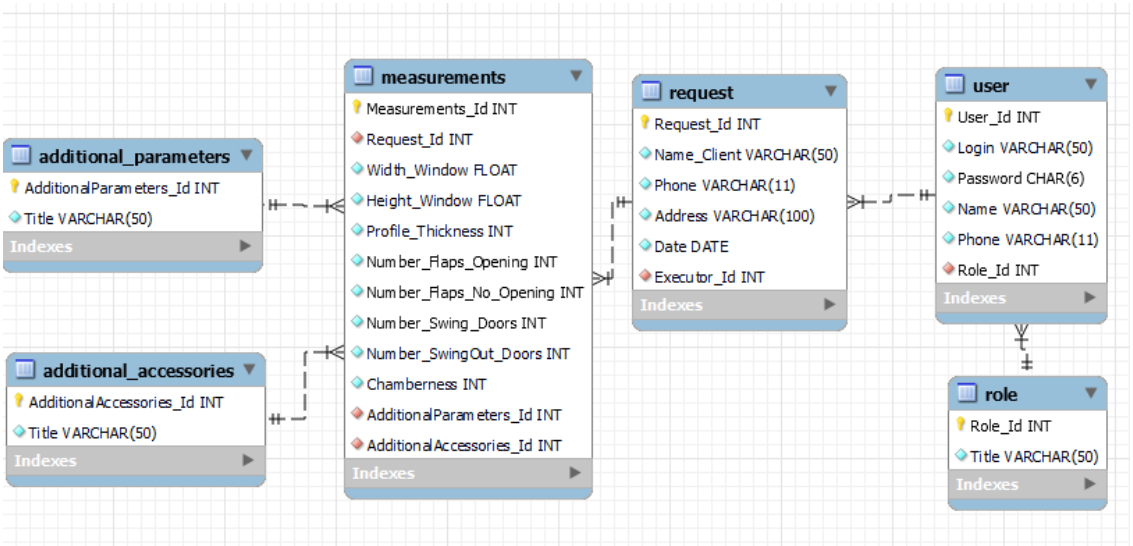

Рисунок 1.5.1 – Схема данных БД

1.6. Описание структуры базы данных

Описание структуры базы данных представлено в таблице 1.6.1.

Таблица 1.6.1 - Описание структуры базы данных

Имя поля	Описание поля	Тип данных	Размер поля	Тип ключа[1]
1	2	3	4	5
Additional_Accessories (дополнительные аксессуары)				
AdditionalAccessories_Id	ID доп. аксессуара	INT		PK
Title	Наименование доп. аксессуара	VARCHAR	50	
Additional_Parameters (дополнительные параметры)				
AdditionalParameters_Id	ID доп. параметра	INT		PK
Title	Наименование доп. парамтера	VARCHAR	50	

[1]PK-первичный ключ

FK-внешний ключ

Продолжение таблицы 1.6.1

1	2	3	4	5
Role (Роль пользователя)				
Role_Id	ID роли пользователя	INT		PK
Title	Наименование роли	VARCHAR	50	
User (Пользователи)				
User_Id	ID пользователя	INT		PK
Login	Логин	VARCHAR	50	
Password	Пароль	CHAR	6	
Name	Имя пользователя	VARCHAR		
Phone	Номер телефона	VARCHAR	11	
Role_Id	ID роли	INT		FK
Request (Заявка)				
Request_Id	ID заявки	INT	30	PK
Name_Client	Имя клиента	VARCHAR	50	
Phone	Номер телефона клиента	VARCHAR	11	
Address	Адрес клиента	VARCHAR	50	
Date	Дата удобная клиенту	DATE		
Executor_Id	Назначенный замерщик	INT		FK
Measurements (Замеры)				
Measurements_Id	ID замера	INT		PK
Request_Id	ID заявки	INT		FK
Width_Window	Ширина окна	FLOAT		

Продолжение таблицы 1.6.1

1	2	3	4	5
Height_Window	Высота окна	FLOAT		
Profile_Thickness	Толщина профиля	INT		
Number_Flaps_Opening	Количество створок открывающихся	INT		
Number_Flaps_No_Opening	Количество створок не открывающихся	INT	6	
Number_Swing_Doors	Количество створок поворотных	INT		
Number_SwingOut_Doors	Количество створок поворотно-откидных	INT		
Chamberness	Камерность	INT		
Furniture_Id	ID фурнитуры	INT		FK
AdditionalParameters_Id	ID дополнительных параметров	INT		FK
AdditionalAccessories_Id	ID дополнительных аксессуаров	INT		FK

1.7. Контрольный пример

Контрольный пример является ручным подсчётом задачи. По составленной программе обрабатываются исходные данные контрольного примера. Полученные результаты сравниваются с известными результатами контрольного примера. При несовпадении результатов производится поиск, исправление ошибок, и снова производится выполнение программы.

Входная информация контрольных примеров представлена ниже в приложение Б.

Выходные данные для контрольных примеров показаны в приложении В.

1.8 Общие требования к программному продукту

Пользователи должны иметь базовые навыки пользования персональным компьютером.

Минимальные требования к техническому обеспечению программного продукта следующие:

– ОС: Windows 7 Service Pack1/ Windows 7 64Bit Service Pack1/ Windows 8.1 64Bit / Windows 10 64Bit;

– процессор: 2.40 Ггц (четырехъядерный) / AMDPhenom 9850 (четырехъядерный) @ 2.5 Ггц;

– оперативная память: 512МБ (Win 7/Win8 и выше);

– видеокарта: NVIDIA 9800 GT с 512 МБ видеопамяти/ AMD HD 4870 с 1 Гб видеопамяти (DX 9, 10, 10.1);

– жесткий диск: 10 гигабайт свободного места;

– Microsoft DirectX версия 9.0c;

Функциональные возможности программного продукта:

– приложение должно формировать и отображать выходные данные пользователю;

– в приложении должен быть обеспечен просмотр таблиц (справочников) базы данных с возможностью добавления, редактирования, удаления данных.

Требования к надежности:

– приложение должно обрабатывать ошибочные действия пользователя и сообщать ему об этом;

– приложение должно обеспечивать контроль входной и выходной информации.

Требования к информационной и программной совместимости: обеспечить работу приложения с таблицами СУБД MySQL Workbench.

2. Экспериментальный раздел

2.1 Описание программы

Программа имеет модульную структуру. При ее запуске выполняется проект на WPF Kurs.exe. Программа Kurs.exe написана на языке C# в среде программирования VisualStudio 2022 с использованием системы управления базой данных MySql Workbench 8.0 CE.

Схема взаимодействия модулей программы представлена на рисунке 2.1.1. Описание модулей и методов представлено в таблице 2.1.1.

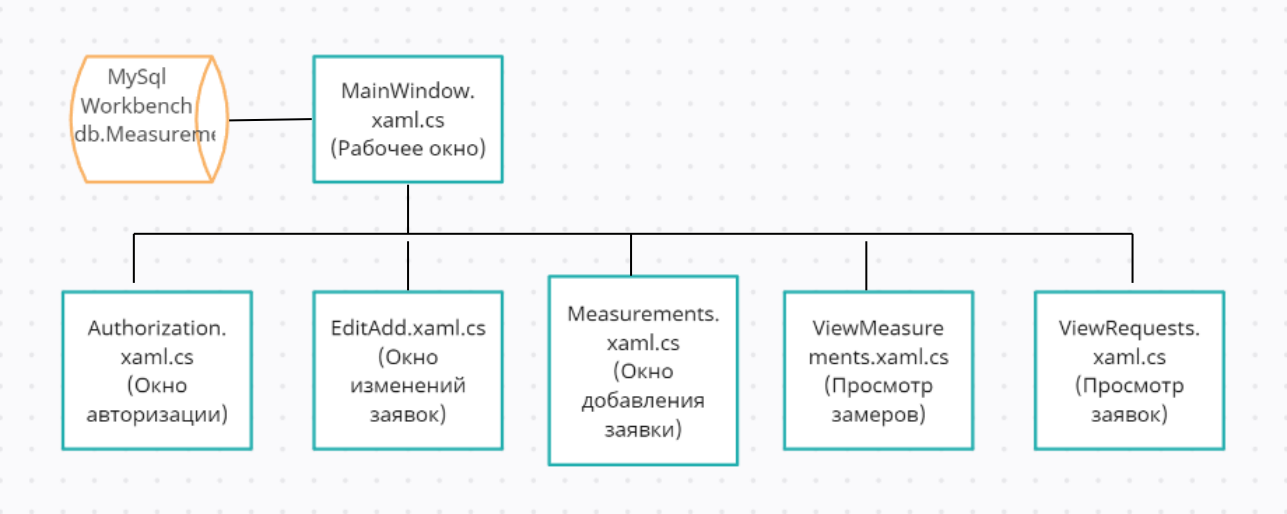

Рисунок 2.1.1 – Схема взаимодействия модулей

Таблица 2.1.1. - Описание модулей

Методы	Назначение
1	2
MainWindow.xaml.cs	
public MainWindow()	Переход на страницу входа

Продолжение таблицы 2.1.1

1	2
Authorization.xaml.cs	
Void Enter_Click ()	Переход на страницу заявок в случае удачной авторизации
EditAdd.xaml.cs	
void SetUp ()	Загрузка данных в combo box
void EditAdd ()	Загрузка методов SetUp(), LoadExecutors(), таблицы из базы данных с заявками
void LoadExecutors()	Загрузка исполнителей, для отображения их имен
void CreateRequest()	Создание самой заявки
void EditRequest()	Изменение заявки, присвоение
public EditAdd(Request request)	Загрузка методов SetUp(), LoadExecutors(), таблицы из базы данных с заявками, изменение заявки, ограничение редактирования у исполнителя
bool IsValid()	Проверка на валидацию заявки
void Apply_Click(object sender, RoutedEventArgs e)	Переход на окно просмотра заявок в случае если успешно добавили или изменили заявку

Продолжение таблицы 2.1.1

Measurements.xaml.cs	
void SetUp ()	Загрузка данных в combo box
public Measurements()	Инициализация, применение метода SetUp()
void CreateMeasurement()	Создание замера и его сохранение
void Apply_Click(object sender, RoutedEventArgs e)	Применение метода CreateMeasurement и переход на просмотр замеров
ViewMeasurements.xaml.cs	
public ViewMeasurements()	Инициализация, отображение роли, подгрузка зависимостей таблиц из базы данных
void ViewRequests_Click(object sender, RoutedEventArgs e)	Переход на отображение заявок
void View_Click(object sender, RoutedEventArgs e)	Переход на создание замера
void Measurements_Click(object sender, RoutedEventArgs e)	Переход на окно создания заявок в случае если роль пользователя 2, иначе просто кнопка не нажимается
ViewRequests.xaml,cs	
public ViewRequests()	Инициализация, подгрузка таблицы из базы данных, отображение роли пользователя, распределение пользоватлей по ролям, применение метода UpdateView()

Продолжение таблицы 2.1.1

1	2
public ViewRequests()	Инициализация, подгрузка таблицы из базы данных, отображение роли пользователя, распределение пользоватлей по ролям, применение метода UpdateView()
void ResetButton_Click(object sender, RoutedEventArgs e)	Очищение поля поиск и сортировки, применение метода UpdateView()
void UpdateView()	Поиск и сортировка
void Search_TextChanged(object sender, TextChangedEventArgs e)	Применение метода UpdateView()
void Sort_SelectionChanged(object sender, SelectionChangedEventArgs e)	Применение метода UpdateView()
void Edit_Click(object sender, RoutedEventArgs e)	Переход на окно управление заявкой
void Create_Click(object sender, RoutedEventArgs e)	Переход на окно управление заявкой
void Measurements_Click(object sender, RoutedEventArgs e)	Переход на окно создания замера
void MeasurementsView_Click(object sender, RoutedEventArgs e)	Переход на просмотр замеров

Код программы представлен в приложении Г.

2.2 Протокол тестирования программного продукта

В протоколе тестирования отражаются:

- тестирование на корректных данных;
- тестирование на некорректных данных;

– тестировании продукта на данных контрольного примера.

Тестирование на авторизацию при корректных данных, ожидаемое сообщение «Авторизация прошла успешно» (рисунок 2.2.1).

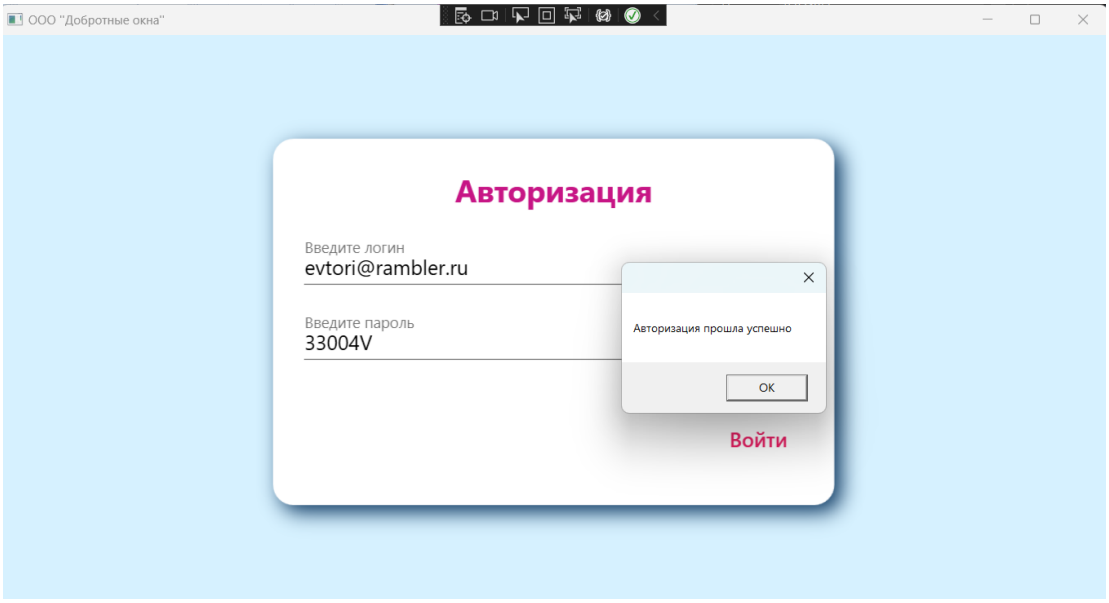

Рисунок 2.2.1 – Сообщение об успешной авторизации

Тестирование на изменение при корректных данных, ожидаемое сообщение «Вы изменили успешно» (рисунок 2.2.2).

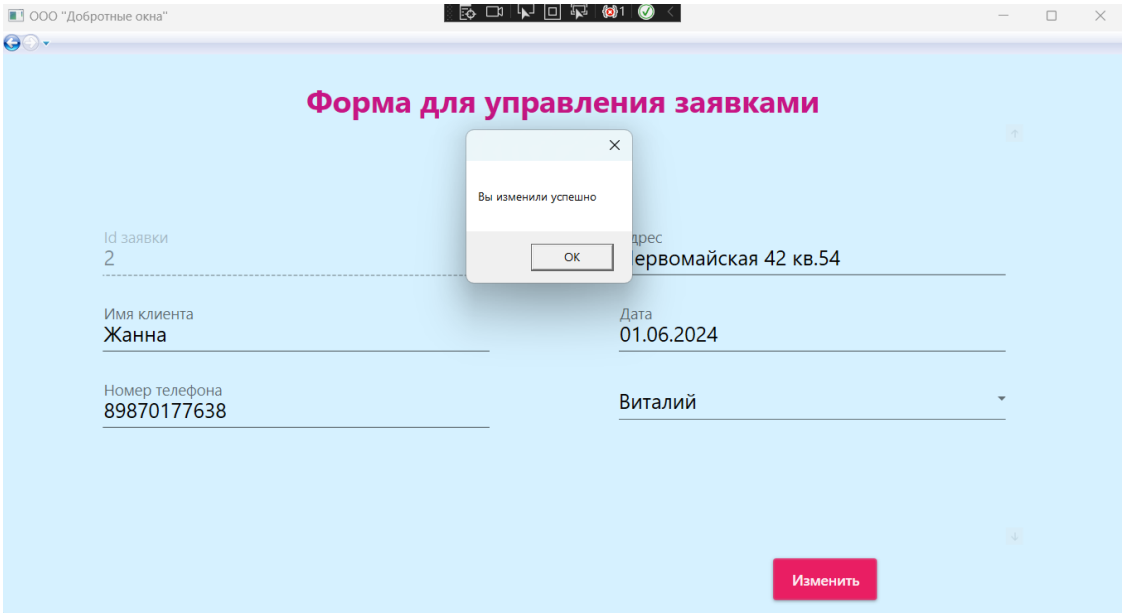

Рисунок 2.2.2 – Сообщение об успешном измении

Тестирование на изменение при не выбранном исполнителе, ожидаемое сообщение «Необходимо выбрать исполнителя!» (рисунок 2.2.3).

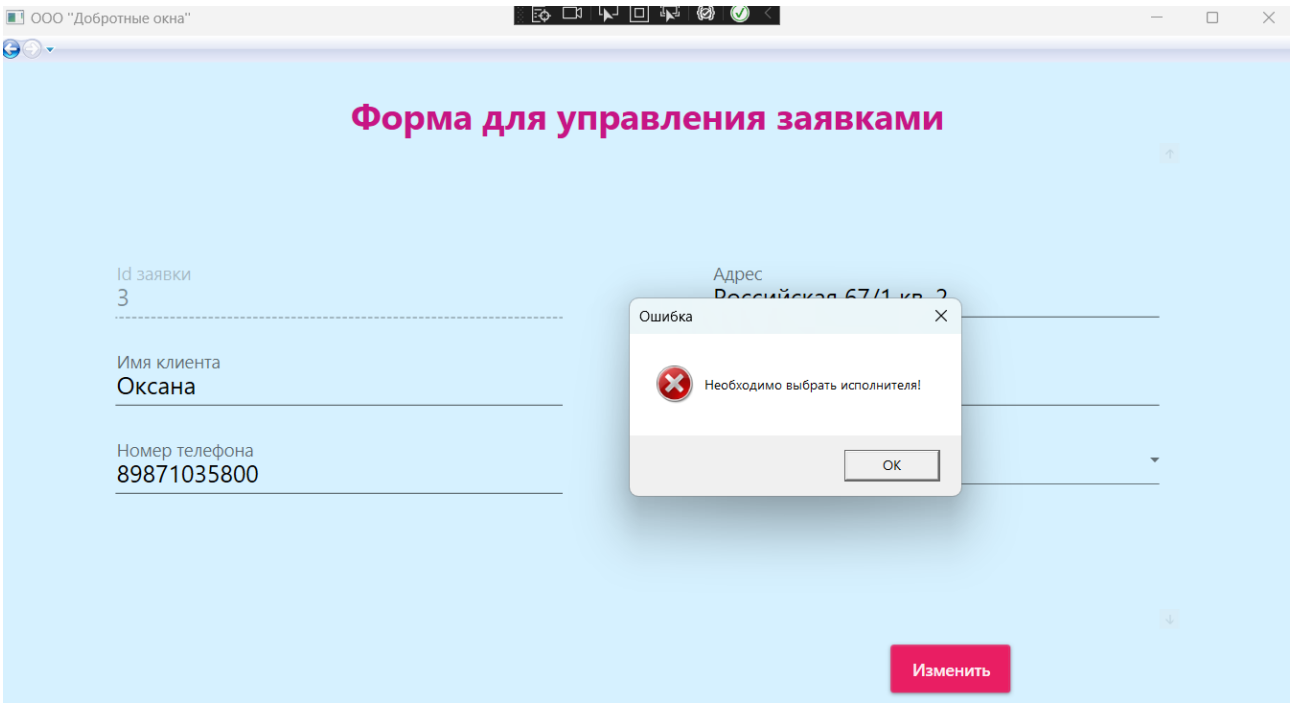

Рисунок 2.2.3 – Сообщение об ошибке

Тестирование на неккоректный ввод при авторизации, ожидаемое сообщение «Пользователь не найден» (рисунок 2.2.4).

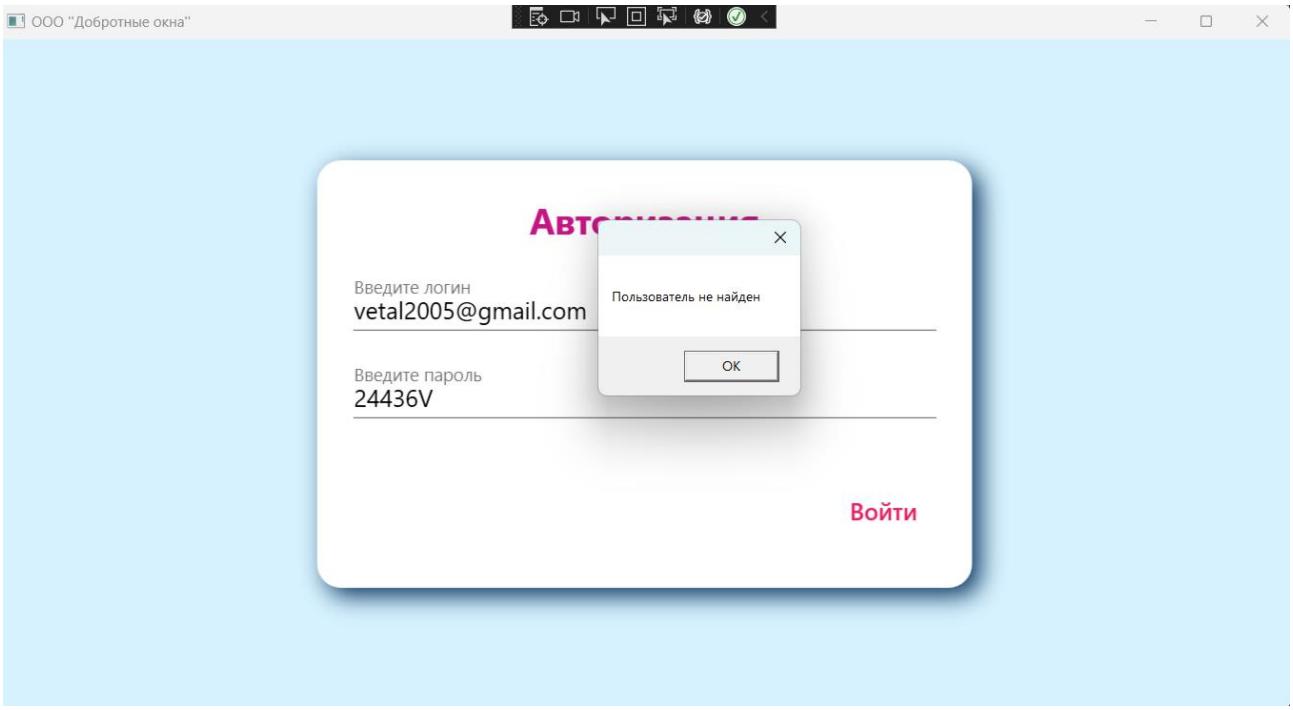

Рисунок 2.2.4 – Сообщение об ошибке

Тестирование на нажатие кнопки «Создать замер» менеджером, ожидаемый результат: кнопка станет неактивной (рисунок 2.2.5).

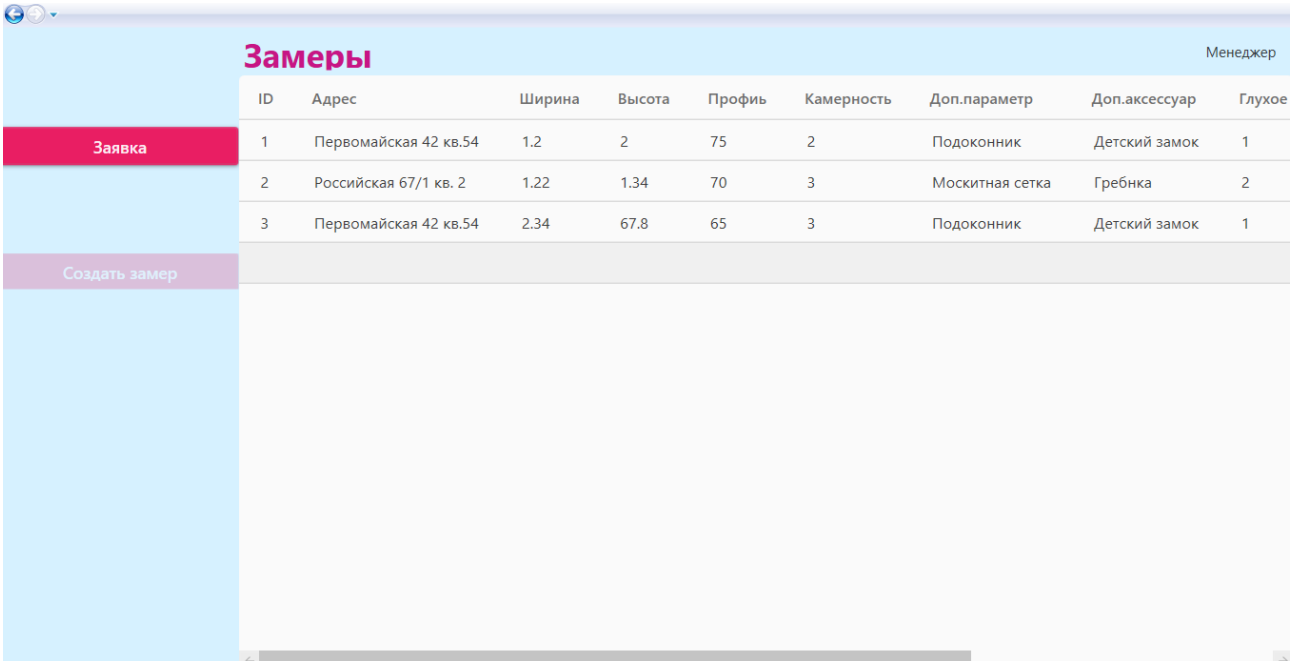

Рисунок 2.2.5 – Неактивная кнопка

При тестировании программы на данных контрольного примера, результаты полностью совпадают с ожидаемыми (рисунки 2.2.6 – 2.2.7)

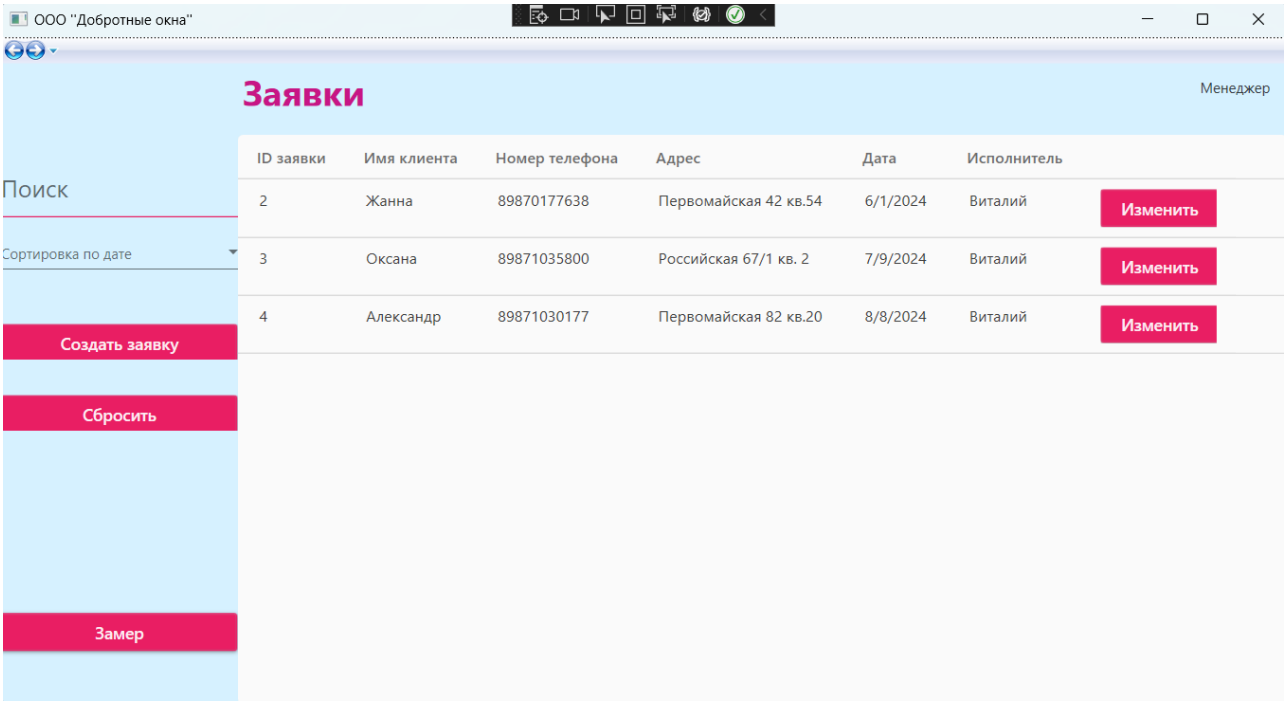

Рисунок 2.2.6 – Оформленные заявки

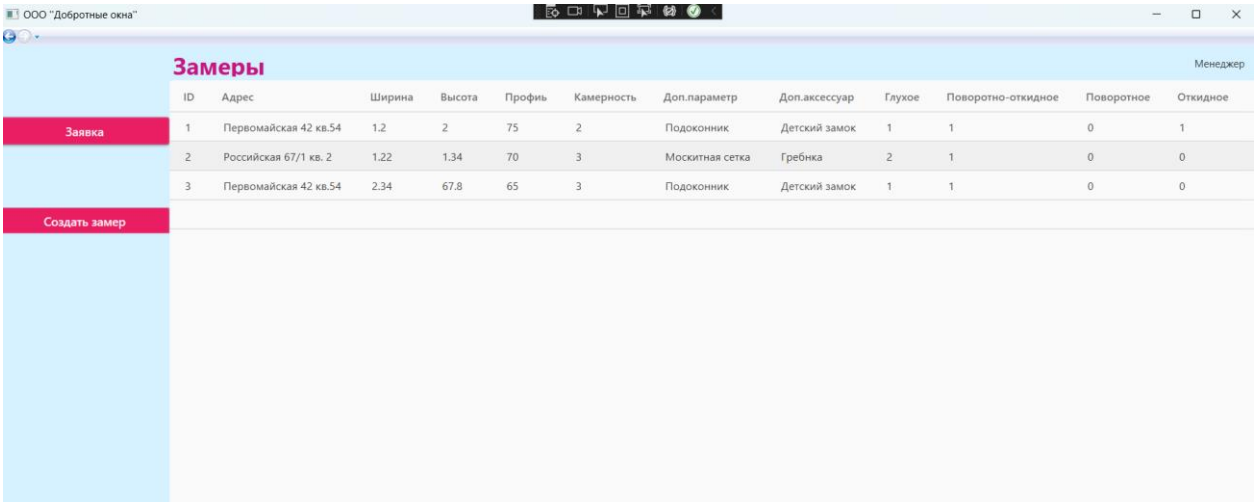

Рисунок 2.2.14 – Все оформленные замеры

Так замерщик может просмтаривать все свои оформленные замеры.

2.3 Руководство пользователя

Назначение системы

Программа «Учет заявок на замер окон» предназначена для упрощения работы замерщиков, и обьеденения 2-ух групп пользователей, замерщиков и менеджеров.

Основной целью данной информационной системы является формирование замеров по адресам клиентов, а также вывода списком все выполненные замеры.

Условия применения системы

Программное обеспечение разрабатывается для персональной вычислительной техники со следующими характеристиками:

- Microsoft Windows 7 / 8 / 10;
- процессор 1 ГГц;
- 128 МБ ОЗУ;
- 60 МБ свободного пространства на диске;
- разрешение экрана монитора не менее 1920 × 1080;

Программа «Учет заявок на замер окон» предназначена для пользователей,

имеющих как минимум первоначальные навыки работы с графической операционной системой, на которой будет запускаться данная программа.

Подготовка системы к работе

Для запуска программы необходимо запустить приложение kurs.exe из каталога, в котором установлен данный программный продукт. После этого открывается окно авторизации для входа в программу (рисунок 2.3.1).

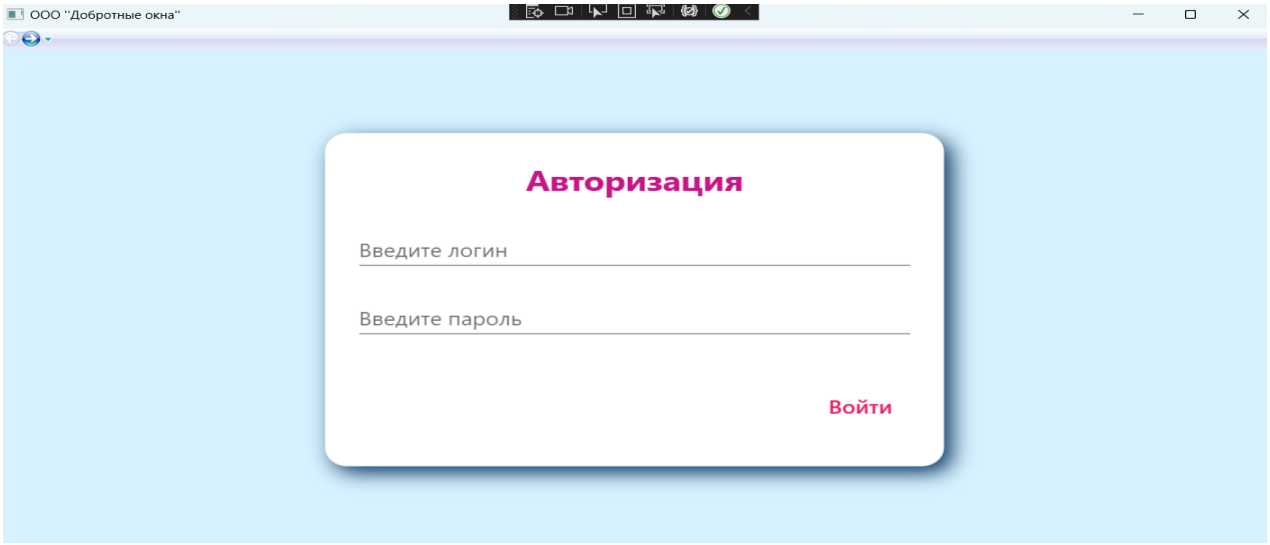

Рисунок 2.3.1 – Окно авторизации для входа в программу

Описание операций

После ввода правильного логина и пароля осуществляется вход в приложение (рисунок 2.3.2-2.3.3)

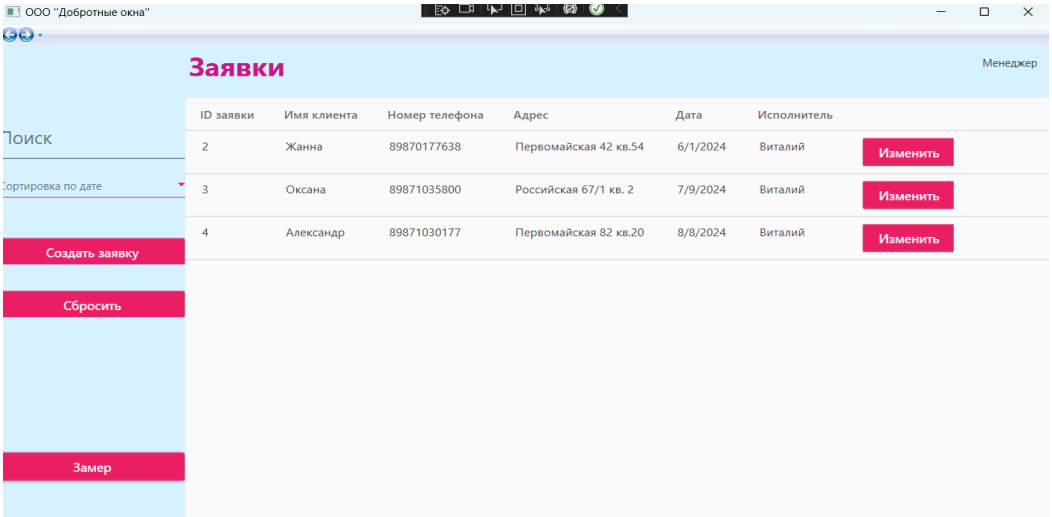

Рисунок 2.3.2 – Окно заявок, если зашел администратор

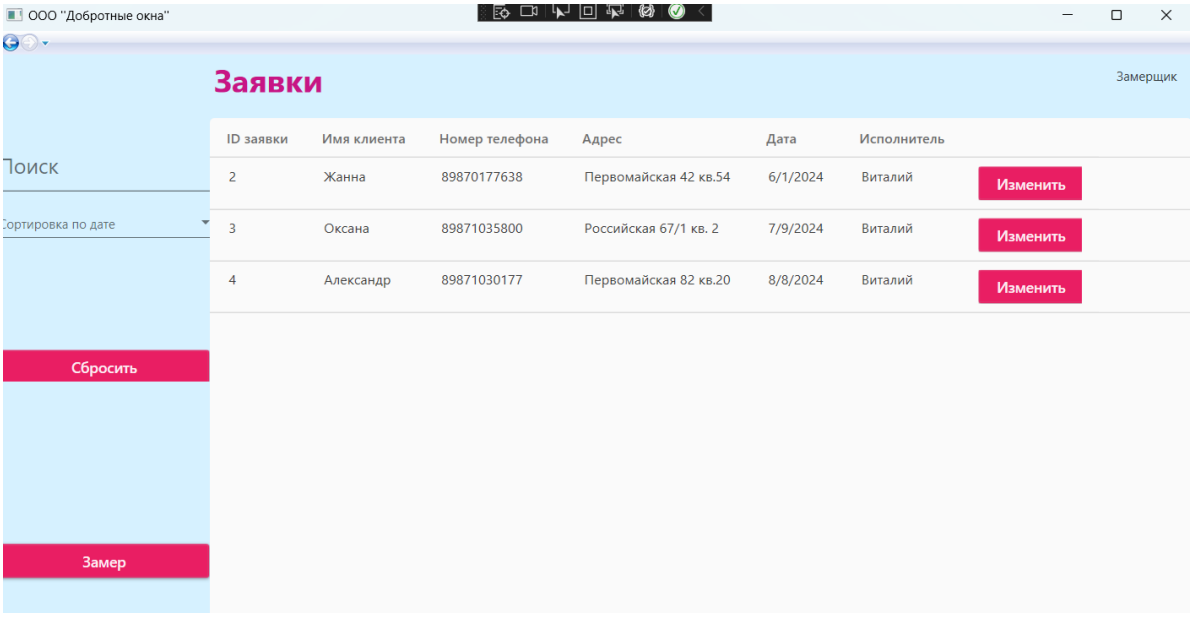

Рисунок 2.3.3 – Окно заявок, если зашел пользователь

При входе в приложение отображается список заявок, в меню слева администратор может отсортировать по дате и искать по адресу заявку и также, когда ему больше не нужно будет пользоваться сортировкой он может сбросить, нажав на одну кнопку ниже «Сбросить». Также у администратора есть возможность создать заявку, нажимая на кнопку выше на «Создать заявку», ему открывается форма управления заявкой рисунок (2.3.4). Тоже самое будет если он рядом с заявкой справа нажмет на кнопку «Изменить» (рисунок 2.3.5).

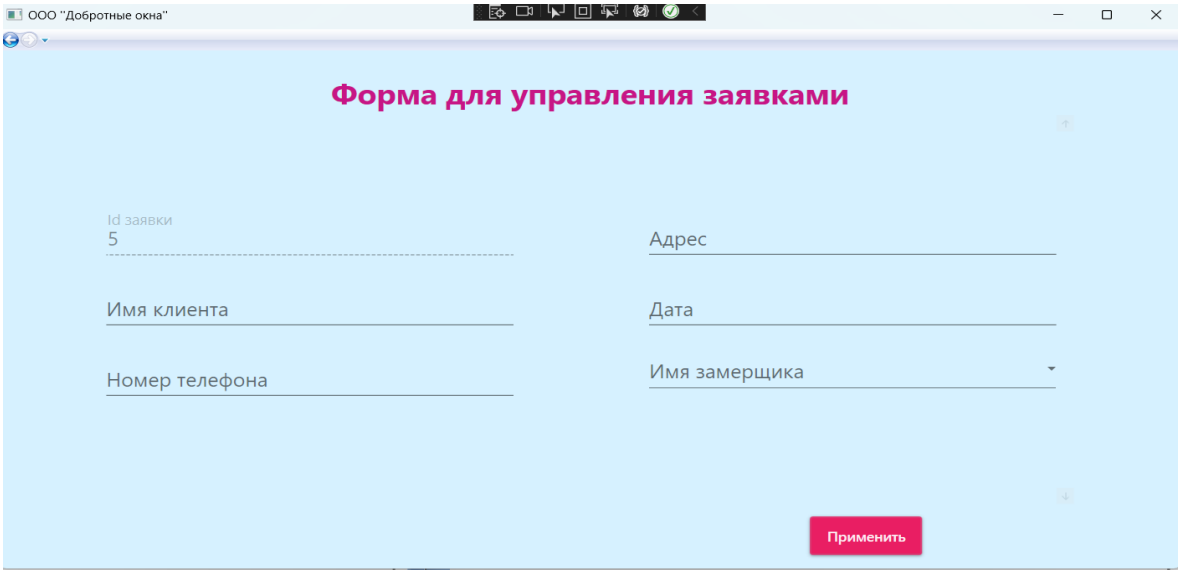

Рисунок 2.3.4 – Окно управления заявкой

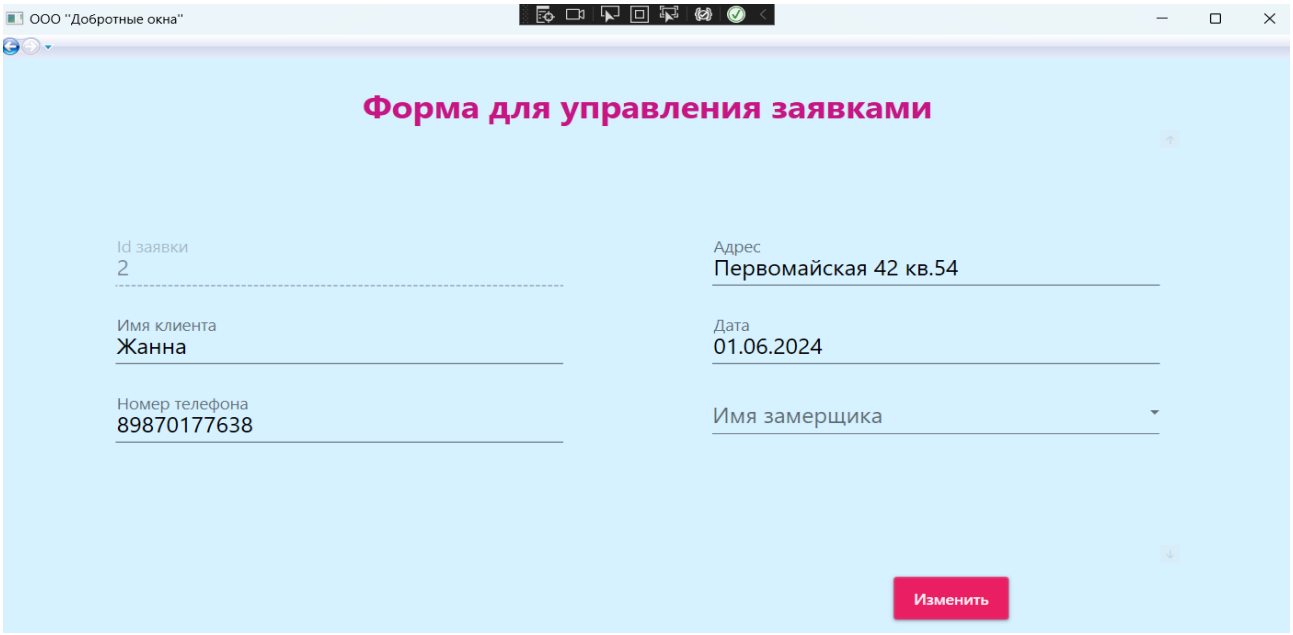

Рисунок 2.3.5 – Окно для изменения заявки

Далее можно вернуться назад и из окна с заявками перейти на окно с замерами, нажав на кнопку «Замеры», перед нами открывается список замеров (рисунок2.3.6).

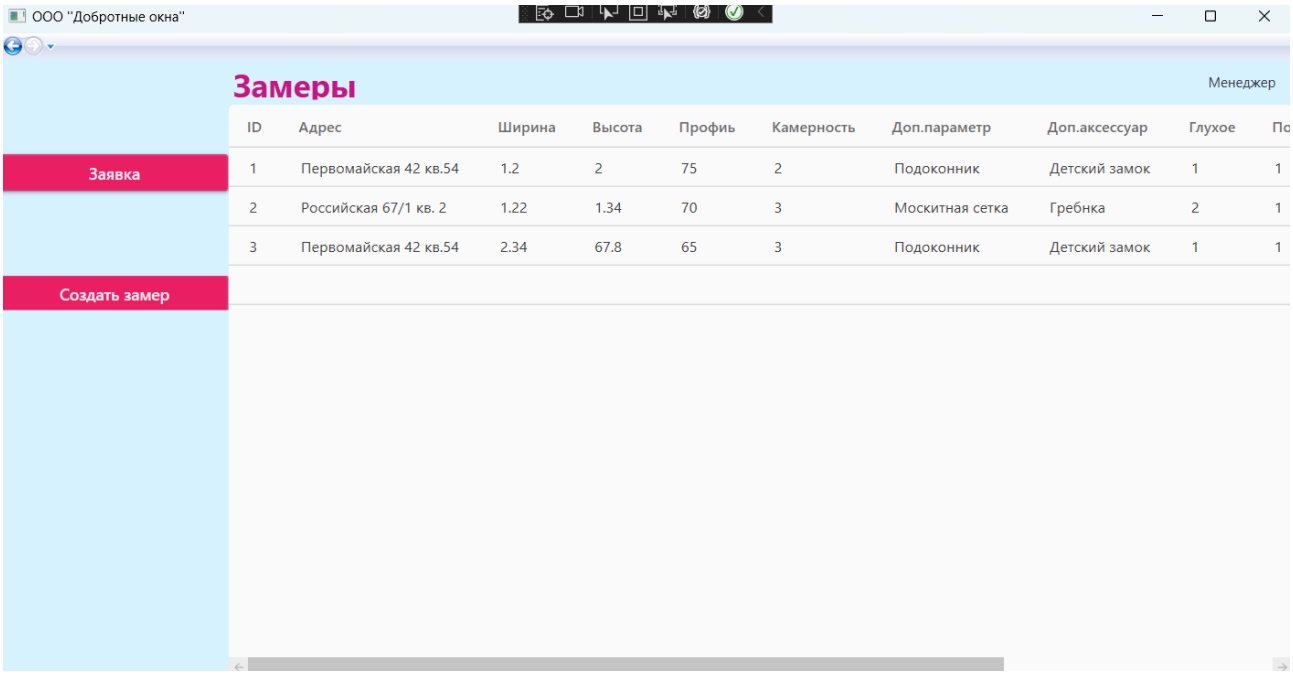

Рисунок 2.3.6 – Окно с спсиком замеров

Если захотим вернуться обратно к заявкам это всегда можно сделоать через меню.

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсового проекта были разработаны структура и алгоритм работы WPF-приложения «Учет заявок за замер окон».

При этом были изучены особенности реализации компонентов WPF для построения клиентских приложений с визуально привлекательными возможностями взаимодействия с пользователем.

Результатом работы стало создание WPF-приложения для сокращения времени работы замерщиков для заполнения замера по адресу.

WPF-приложение написано на языке C# в среде разработки Visual Studio 2022 с использованием языка разметки XAML и системы управления базой данных MySql Workbench 8.0. CE.

Были проведены опытная эксплуатация и отладочное тестирование WPF приложения. По результатам отладочного тестирования были устранены некоторые недостатки, в частности были обнаружены и исправлены неточности в реализации алгоритма: усовершенствован контроль на входные данные и отформатирован вывод документов. После этого было написано руководство пользователя.

С помощью приложения на основании данных контрольного примера были получены результаты, которые полностью совпадают с выходной информацией контрольного примера.

Приложение А

Шаблон выходных документов

ID	Адрес	Ширина	Высота	Профиль	Камерность	Доп. параметр	Доп. аксессуар	Глухое	Поворотно-откидное	Поворотное	Откидное
1	{Адрес}	{Ширина}	{Высота}	{Профиль}	{Камерность}	{Доп. параметр}	{Доп. аксессуар}	{Глухое}	{Поворотно - откидное}	{Поворотное}	{Откидное}
2	{Адрес}	{Ширина}	{Высота}	{Профиль}	{Камерность}	{Доп. параметр}	{Доп. аксесуар}	{Глухое}	{Поворотно - откидное}	{Поворотное}	{Откидное}
3	{Адрес}	{Ширина}	{Высота}	{Профиль}	{Камерность}	{Доп. параметр}	{Доп. аксессуар}	{Глухое}	{Поворотно - откидное}	{Поворотное}	{Откидное}

Приложение Б

Входные данные контрольного примера

Таблица Б. – Справочник заявок

ID заявки	Имя клиента	Номер телефона	Адрес	Дата	Исполнитель
1	2	3	4	5	6
1	Жанна	89870177638	Первомайская 45 кв.54	7/1/2024	Виталий
2	Оксана	89871035800	Российская 67/1 кв.2	7/9/2024	Виталий
3	Александр	89871030177	Первомайская 80 кв.20	8/8/2024	Радмилла

Приложение В

Выходные данные контрольного примера

Таблица В. – Замеров

ID	Адрес	Ширина	Высота	Профиль	Камерность	Доп. параметр	Доп. аксессуар	Глухое	Поворотно-откидное	Поворотное	Откидное
1	Первомайская 45 кв.54	1,2	2	75	2	Подоконник	Детский замок	1	1	0	1
2	Российская 67/1 кв.2	1,22	1,34	70	3	Москитная сетка	Гребенка	2	1	0	0
3	Первомайская 45 кв.54	2,34	67,8	65	3	Подоконник	Детский замок	1	1	0	0

Приложение Г

Пример описания структуры базы данных

Таблица Г.1- Role (Роль пользователей)

Содержание поля	Имя поля	Тип, длина	Примечания
Идентификатор роли	D_IDATE	INT	Первичный ключ
Наименование	D_NAME	VARCHAR(50)	Обязательное поле

Таблица Г.2- User (Пользователи)

Содержание поля	Имя поля	Тип, длина	Примечания
Идентификатор пользователя	User_Id	INT	Первичный ключ
Логин пользователя	Login	VARCHAR(50)	Обязательное поле
Пароль	Password	CHAR(6)	Обязательное поле
Имя	Name	VARCHAR(50)	Обязательное поле
Номер телефона	Phone	VARCHAR(11)	
Идентификатор роли	Role_Id	INT	Внешний ключ(к Role)

Таблица Г.3- Additional_Accessories (Дополнительные аксессуары)

Содержание поля	Имя поля	Тип, длина	Примечания
Идентификатор доп. аксесуара	Additional_Accessories_Id	INT	Первичный ключ
Оклад	Title	VARCHAR(50)	Обязательное поле

Таблица Г.4- Measurements (Замеры)

Содержание поля	Имя поля	Тип, длина	Примечания
Идентификатор замера	Measurements_Id	INT	Первичный ключ
Идентификатор заявки	Request_Id	INT	Обязательное поле
Ширина окна	Width_Window	FLOAT	Обязательное поле
Высота окна	Height_Window	FLOAT	Обязательное поле
Толщина профиля	Profile_Thickness	INT	Обязательное поле
Глухая створка	Number_Flaps_No_Opening	INT	Обязательное поле
Поворотно-откидная створка	Number_Flaps_Opening	INT	Обязательное поле
Поворотная створка	Number_Swing_Doors	INT	Обязательное поле
Откидная створка	Number_SwingOut_Doors	INT	Обязательное поле
Камерность	Chamberness	INT	Обязательное поле
Идентификатор доп.параметра	AdditionalParameters_Id	INT	Внешний ключ
Идентификатор доп.аксесуара	AdditionalAccessories_Id	INT	Внешний ключ

Таблица Г.5- Request (Заявки)

Содержание поля	Имя поля	Тип, длина	Примечания
Идентификатор заявки	Request_Id	INT	Первичный ключ
Имя клиента	Name_Client	VARCHAR(50)	Обязательное поле
Телефон	Phone	VARCHAR(11)	Обязательное поле
Адрес	Address	VARCHAR(100)	Обязательное поле
Дата	Date	DATE	Обязательное поле
Идентификатор исполнителя	Executor_Id	INT	Внешний ключ

Таблица Г.6- Additional_Parameters (Дополнительные параметры)

Содержание поля	Имя поля	Тип, длина	Примечания
Идентификатор доп. параметра	AdditionalParameters_Id	INT	Первичный ключ
Наименование доп. параметра	Title	VARCHAR(50)	Обязательное поле

Код программы

Форма MainWindow.xaml.cs (Рабочее окно)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
namespace kurs
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public static Frame Frame;
        public MainWindow()
        {
            InitializeComponent();
            Frame = MainFrame;
            Frame.Content = new Authorization();
        }
    }

}
```

Форма Authorization.xaml.cs (Окно авторизации)

```
using kurs.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
namespace kurs
{
    /// <summary>
    /// Логика взаимодействия для Authorization.xaml
    /// </summary>
    public partial class Authorization : Page
    {
        public Authorization()
        {
            InitializeComponent();
        }
        private void Enter_Click(object sender, RoutedEventArgs e)
        {
            MeasurementsContext context = new MeasurementsContext();
            User? user = context.Users.FirstOrDefault(u => u.Login == Login.Text);
            if (user == null)
            {
                MessageBox.Show("Пользователь не найден");
            }
```

```
            else if (user.Password != Pass.Text)
            {
                MessageBox.Show("Пароль не подходит");
            }
            else
            {
                MessageBox.Show("Авторизация прошла успешно");
                User.ActiveUser = user;
                MainWindow.Frame.Content = new ViewRequests();
            }
        }
    }
}
```

Форма EditAdd.xaml.cs (Окно изменения заявки)

```
using kurs.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
namespace kurs
{
    /// <summary>
    /// Логика взаимодействия для EditAdd.xaml
    /// </summary>
    public partial class EditAdd : Page
```

```
    {
        public void SetUp() // загрузка данных в combo boxs
        {
            MeasurementsContext repairContext = new MeasurementsContext();
            Executor.ItemsSource = repairContext.Users.Where(user => user.RoleId == 2).Select(user
=> user.UserId).ToList();
        }
        public EditAdd()
        {
            InitializeComponent();
            SetUp();
            LoadExecutors();
            MeasurementsContext repairContext = new MeasurementsContext();
            if (repairContext.Requests.Count() == 0)
            {
                RequestId.Text = "1";
            }
            else
            {
                RequestId.Text = (repairContext.Requests.Max(request => request.RequestId) +
1).ToString();
            }
        }
        private void LoadExecutors()
        {
            MeasurementsContext context = new MeasurementsContext();
            List<User> users = context.Users.ToList();
            Executor.ItemsSource = users;
            Executor.DisplayMemberPath = "Name";
            Executor.SelectedValuePath = "Id";
        }
        private void CreateRequest()
        {
            Apply.Content = "Применить";
            MeasurementsContext repairContext = new MeasurementsContext();
```

```
            //User selectedExecutor = repairContext.Users.FirstOrDefault(u => u.UserId == ); //
Получаем выбранного пользователя
            Request request = new Request(
                Convert.ToInt32(RequestId.Text),
                NameClient.Text,
                NumberPhone.Text,
                Address.Text,
                DateOnly.Parse(Data.Text),
                ((User)Executor.SelectedItem).UserId
            );
            repairContext.Requests.Add(request);
            repairContext.SaveChanges();
        }
        private void EditRequest()
        {
            var repairContext = new MeasurementsContext();
            var current = repairContext.Requests.First(r => r.RequestId ==
Convert.ToInt32(RequestId.Text));
            current.NameClient = NameClient.Text;
            current.Address = Address.Text;
            current.Phone = NumberPhone.Text;
            current.Date = DateOnly.Parse(Data.Text);
            current.ExecutorId = ((User)Executor.SelectedItem).UserId;
            repairContext.SaveChanges();
        }
        public EditAdd(Request request) // редактирование
        {
            InitializeComponent();
            SetUp();
            LoadExecutors();
            Apply.Content = "Изменить";
            Executor.SelectedItem = request.ExecutorId ?? -1;
            RequestId.Text = request.RequestId.ToString();
            NameClient.Text = request.NameClient.ToString();
            NumberPhone.Text = request.Phone.ToString();
```

```
            Address.Text = request.Address?.ToString();
            Data.Text = request.Date.ToString();
            MeasurementsContext repairContext = new MeasurementsContext();
            User selectedExecutor = repairContext.Users.FirstOrDefault(user => user.UserId ==
request.ExecutorId);

            if (selectedExecutor != null)
            {
                Executor.SelectedItem = selectedExecutor;
            }
            if (User.ActiveUser.RoleId == 2) // ограничение редактирования у исполнителя
            {
                RequestId.IsEnabled = NameClient.IsEnabled = NumberPhone.IsEnabled =
Executor.IsEnabled = Address.IsEnabled = Data.IsEnabled = false;
            }
        }
        private void Delete_Click(object sender, RoutedEventArgs e)
        {
            //DeleteRequest();
            MainWindow.Frame.Content = new ViewRequests();
        }
        private bool IsValid()
        {
            if (!DateOnly.TryParse(Data.Text, out _))
            {
                MessageBox.Show("Некорректный формат даты!",
                    "Ошибка", MessageBoxButton.OK,
                    MessageBoxImage.Error);
                return false;
            }
            if (NameClient.Text.Length > 50)
            {
                MessageBox.Show("Имя клиента не может быть длинее 50 символов!",
                    "Ошибка", MessageBoxButton.OK,
                    MessageBoxImage.Error);
```

```
            return false;
        }
        if (NumberPhone.Text.Length > 11)
        {
            MessageBox.Show("Номер телефона не может быть длинее 11 символов!",
                "Ошибка", MessageBoxButton.OK,
                MessageBoxImage.Error);
            return false;
        }
        if (Address.Text.Length > 50)
        {
            MessageBox.Show("Адрес не может быть длинее 50 символов!",
                "Ошибка", MessageBoxButton.OK,
                MessageBoxImage.Error);
            return false;
        }
        if (Executor.SelectedIndex == -1)
        {
            MessageBox.Show("Необходимо выбрать исполнителя!",
                "Ошибка", MessageBoxButton.OK,
                MessageBoxImage.Error);
            return false;
        }
        return true;
    }
    private void Apply_Click(object sender, RoutedEventArgs e)
    {
        if (!IsValid()) return;
        //DeleteRequest();
        if (((Button)sender).Content == "Применить")
        {
            CreateRequest();
            MessageBox.Show("Вы изменили успешно");
        }
        else
```

```
            {
                EditRequest();
            }
            MessageBox.Show("Вы изменили успешно");
            MainWindow.Frame.Navigate( new ViewRequests());
        }
    }
}
```

Форма Measurements.xaml.cs (Окно изменения замера)

```
using kurs.Models;
using System;
using System.CodeDom.Compiler;
using System.Collections.Generic;
using System.Linq;
using System.Net;
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.Markup;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
namespace kurs
{
    /// <summary>
    /// Логика взаимодействия для Measurements.xaml
    /// </summary>
    public partial class Measurements : Page
    {
        private void SetUp() // Загрузка данных в ComboBox
```

```
        {
            MeasurementsContext repairContext = new MeasurementsContext();
            repairContext.AdditionalAccessories.ToList();
            repairContext.AdditionalParameters.ToList();
            RequestId.ItemsSource = repairContext.Requests.ToList();
            RequestId.DisplayMemberPath = "Address";
            RequestId.SelectedValuePath = "Id";
            AdditionalParametersId.ItemsSource = repairContext.AdditionalParameters.ToList();
            AdditionalParametersId.DisplayMemberPath = "Title";
            AdditionalParametersId.SelectedValuePath = "Id";

            AdditionalAccessoriesId.ItemsSource = repairContext.AdditionalAccessories.ToList();
            AdditionalAccessoriesId.DisplayMemberPath = "Title";
            AdditionalAccessoriesId.SelectedValuePath = "Id";
        }


        public Measurements()
        {
            InitializeComponent();
            SetUp(); // Загрузка данных в ComboBox
        }
        private void CreateMeasurement()
        {
            using (MeasurementsContext repairContext = new MeasurementsContext())
            {
                // Получаем значения из элементов управления
                int requestId = RequestId.SelectedItem != null ?
Convert.ToInt32(RequestId.SelectedValue) : 0;
                float widthWindow = float.Parse(WidthWindow.Text);
                float heightWindow = float.Parse(HeightWindow.Text);
                int profileThickness = Convert.ToInt32(ProfileThickness.Text);
                int numberFlapsOpening = Convert.ToInt32(NumberFlapsOpening.Text);
                int numberFlapsNoOpening = Convert.ToInt32(NumberFlapsNoOpening.Text);
                int numberSwingDoors = Convert.ToInt32(NumberSwingDoors.Text);
```

```
            int numberSwingOutDoors = Convert.ToInt32(NumberSwingOutDoors.Text);
            int chamberness = Convert.ToInt32(Chamberness.Text);
            // Проверяем, выбраны ли дополнительные параметры и аксессуары
            int? additionalParametersId = AdditionalParametersId.SelectedItem != null?
Convert.ToInt32(AdditionalParametersId.SelectedValue) : (int?)null;
            int? additionalAccessoriesId = AdditionalAccessoriesId.SelectedItem != null?
Convert.ToInt32(AdditionalAccessoriesId.SelectedValue): (int?)null;

            // Создаем новый замер
            Measurement measurement = new Measurement
            {
               RequestId = requestId,
               WidthWindow = widthWindow,
               HeightWindow = heightWindow,
               ProfileThickness = profileThickness,
               NumberFlapsOpening = numberFlapsOpening,
               NumberFlapsNoOpening = numberFlapsNoOpening,
               NumberSwingDoors = numberSwingDoors,
               NumberSwingOutDoors = numberSwingOutDoors,
               Chamberness = chamberness,
               AdditionalParametersId = additionalParametersId,
               AdditionalAccessoriesId = additionalAccessoriesId,
            };

            repairContext.Measurements.Add(measurement);
            repairContext.SaveChanges();
        }
     }
    private void Apply_Click(object sender, RoutedEventArgs e)
    {
       CreateMeasurement();
       MainWindow.Frame.Navigate(new ViewMeasurements());
        MessageBox.Show("Вы успешно добавили замер!");
    }
  }
```

```
}
```

Форма ViewMeasurements.xaml.cs (Окно просмотра замера)

```
using kurs.Models;
using Microsoft.EntityFrameworkCore;
using System;
using System.CodeDom.Compiler;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
namespace kurs
{
    /// <summary>
    /// Логика взаимодействия для ViewMeasurements.xaml
    /// </summary>
    public partial class ViewMeasurements : Page
    {
        private List<Measurement> measurements;
        private List<Measurement> viewMeasurement = new List<Measurement>();
        private List<Request> requests;
        private List<Request> viewRequests = new List<Request>();
        public ViewMeasurements()
        {
            InitializeComponent();
            MeasurementsContext measurementsContext = new MeasurementsContext();
```

```
            Rolelabel.Content = measurementsContext.Roles.FirstOrDefault(r => r.RoleId ==
User.ActiveUser.RoleId).Title;
            MeasurementsDataGrid.ItemsSource =
measurementsContext.AdditionalAccessories.ToList();
            MeasurementsDataGrid.ItemsSource =
measurementsContext.AdditionalParameters.ToList();
            MeasurementsDataGrid.ItemsSource = measurementsContext.Requests.ToList();
            MeasurementsDataGrid.ItemsSource = measurementsContext.Measurements.ToList();
            //Rolelabel.Content = measurementsContext.Roles.FirstOrDefault(r => r.RoleId ==
User.ActiveUser.RoleId).Title;
            UpdateView();
        }
        private void ViewRequests_Click(object sender, RoutedEventArgs e)
        {
            MainWindow.Frame.Content = new ViewRequests();
        }
        private void UpdateView()
        {
            viewMeasurement.Clear();
        }
        private void View_Click(object sender, RoutedEventArgs e)
        {
            MainWindow.Frame.Content = new Measurements();
        }
        private void Measurements_Click(object sender, RoutedEventArgs e)
        {
            if (User.ActiveUser.RoleId == 2)
            {
                MainWindow.Frame.Content = new Measurements();
            }
            else
            {
                Measurements.IsEnabled = false;
            }
        }
```

```
    }
}
```

Форма ViewRequests.xaml.cs (Просмотр заявок)

```
using Microsoft.EntityFrameworkCore;
using kurs.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
namespace kurs
{
    /// <summary>
    /// Логика взаимодействия для ViewRequests.xaml
    /// </summary>
    public partial class ViewRequests : Page
    {
        private List<Request> requests;
        private List<Request> viewRequests = new List<Request>();
        public ViewRequests()
        {
            InitializeComponent();
            MeasurementsContext measurementsContext = new MeasurementsContext();
            RequestsDataGrid.ItemsSource = measurementsContext.Requests.ToList();
            Rolelabel.Content = measurementsContext.Roles.FirstOrDefault(r => r.RoleId ==
User.ActiveUser.RoleId).Title;
```

```
            if (User.ActiveUser.RoleId == 2) // исполнитель
            {
                Create.Visibility = Visibility.Collapsed;
                requests = measurementsContext.Requests.Where(r => r.Executor.UserId ==
User.ActiveUser.UserId).Include(r => r.Executor).ToList();
            }
            else // администратор
            {
                requests = measurementsContext.Requests.Include(r => r.Executor).ToList();
            }
            UpdateView();
        }
        private void UpdateView()
        {
            viewRequests.Clear();
            foreach (Request request in requests)
            {
                if ((!request.Address.ToLower().Contains(Search.Text.ToLower()))) // поиск
                {
                    continue;
                }
                viewRequests.Add(request);
            }
            if (Sort.SelectedIndex == 1) // сортировка
            {
                viewRequests = viewRequests.OrderByDescending(request => request.Date).ToList();
            }
            else
            {
                viewRequests = viewRequests.OrderBy(request => request.Date).ToList();
            }
            RequestsDataGrid.ItemsSource = viewRequests;
        }
        private void ResetButton_Click(object sender, RoutedEventArgs e)
        {
```

```
            Sort.SelectedIndex = -1;
            Search.Text = String.Empty;
            UpdateView();
        }
        private void Search_TextChanged(object sender, TextChangedEventArgs e)
        {
            UpdateView();
        }
        private void Sort_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            UpdateView();
        }
        private void Edit_Click(object sender, RoutedEventArgs e)
        {
            MainWindow.Frame.Content = new EditAdd((sender as FrameworkElement).DataContext
as Request);
        }
        private void Create_Click(object sender, RoutedEventArgs e)
        {
            MainWindow.Frame.Content = new EditAdd();
        }
        private void Measurements_Click(object sender, RoutedEventArgs e)
        {
            MainWindow.Frame.Content = new Measurements();
        }
        private void MeasurementsView_Click(object sender, RoutedEventArgs e)
        {
            MainWindow.Frame.Content = new ViewMeasurements();
        }
    }
}
```