**Оглавление**

[Аннотация 4](#_Toc72446649)

[Введение 5](#_Toc72446650)

[1 ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММНОМ СРЕДСТВЕ 6](#_Toc72446651)

[1.1 Основное функциональное назначение программного средства 6](#_Toc72446652)

[1.2 Полное наименование программного средства 6](#_Toc72446653)

[1.3 Условное обозначение программного средства 6](#_Toc72446654)

[1.4 Разработчики программного средства 6](#_Toc72446655)

[2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ 7](#_Toc72446656)

[2.1 Основание для разработки 7](#_Toc72446657)

[2.2 Назначение разработки 7](#_Toc72446658)

[2.3 Требование к программному средству 7](#_Toc72446659)

[2.4 Требования к программной документации 8](#_Toc72446660)

[2.5 Требования к эргономике и технической эстетике 8](#_Toc72446661)

[2.6 Стадии и этапы разработки 9](#_Toc72446662)

[2.7 Порядок контроля и приемки 10](#_Toc72446663)

[3 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ПРОГРАММНОМУ ПРОДУКТУ 11](#_Toc72446664)

[3.1 Декомпозиция поставленной задачи 11](#_Toc72446665)

[3.2 Общая архитектура программного средства 12](#_Toc72446666)

[3.3 Разработка алгоритма решения задачи 15](#_Toc72446667)

[3.4 Реализация функционального назначения программного средства 16](#_Toc72446668)

[3.5 Структурная организация данных 16](#_Toc72446669)

[3.6 Разработка интерфейса ПС 22](#_Toc72446670)

[3.7 Описание структуры выходной информации 37](#_Toc72446671)

[4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 39](#_Toc72446672)

[4.1 Назначение программного средства 39](#_Toc72446673)

[4.2 Условия выполнения программного средства 39](#_Toc72446674)

[4.3 Эксплуатация программного средства 39](#_Toc72446675)

[4.4 Сообщения пользователю 51](#_Toc72446676)

[5 РАЗРАБОТКА ТЕСТОВОГО НАБОРА 52](#_Toc72446677)

[5.1 Определение верхней границы количества тестов 52](#_Toc72446678)

[5.2 Разработка тестовых вариантов 53](#_Toc72446679)

[5.3 Составление отчета о тестировании 55](#_Toc72446680)

[ЗАКЛЮЧЕНИЕ 61](#_Toc72446681)

[Оценка качества программного средства с помощью метрик 61](#_Toc72446682)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 64](#_Toc72446683)

[Приложение А Текст кода программного средства 66](#_Toc72446684)

[Приложение Б Спецификация 86](#_Toc72446685)

[Приложение B Диск с программным средством 88](#_Toc72446686)

**Аннотация**

На дипломную работу студента Изосимова Д.А. на тему: Фирма по установке и обслуживанию дверей. Дипломная работа представлена пояснительной запиской на 88 листов машинописного текста. Состоит из введения, 5 глав, заключения и списка литературных источников.

В главе 1 содержится общие сведения о программном средстве АИС Мир Дверьов. В главе 2 приведено техническое задание на проектирование БД согласно ГОСТ 19.201. Глава 3 содержит пояснительную записку к программному продукту согласно ГОСТ 19.404-79. В главе 4 содержится руководство пользователя согласно ГОСТ 19.504-79. Глава 5 содержит обоснование количества необходимых тестов и тестовые пакеты для оценки работоспособности программного продукта. В главе 6 приведены результаты ручного и автоматического тестирования программного продукта и выполнена оценка качества с помощью метрик.

В работе содержится общее описание программного средства, предназначенного для объекта проектирования – приложения для фирмы, изложены требования к программному средству, описаны его архитектура, реализация функций, интерфейс, приведено руководство пользователя, обоснованы тестовые пакеты и выполнена оценка качества программного продукта с помощью метрик.

**Введение**

В дипломной работе была поставлена задача разработки автоматизированной информационной системы для фирмы по установке и обслуживанию дверей, который позволяет автоматизировать деятельность фирмы по учету заказов.

Для достижения поставленной цели был составлен основной алгоритм работы программы, а также разработаны алгоритмы вспомогательных процедур.

Для реализации алгоритмов был выбран язык программирования C#. Выбор обусловлен тем, что возможностей этого языка достаточно для достижения поставленной цели моей дипломной работы.

При разработке интерфейса программы, я учитывал наиболее нужные и важные функциональные возможности, которые должна выполнять данная программа, а также тип пользователей, которые с ней будут работать.

**1 ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММНОМ СРЕДСТВЕ**

**1.1 Основное функциональное назначение программного средства**

Программа предназначена для автоматизации деятельности фирмы, который позволяет вести учёт заказов. Программа не требует владения современными информационными технологиями для эффективного использования данного программного средства, т.е. подходит для любого пользователя.

**1.2 Полное наименование программного средства**

Фирма по установке и обслуживанию дверей.

**1.3 Условное обозначение программного средства**

«DoorRepairApp 1.0»

**1.4 Разработчики программного средства**

Программное средство разработал студент Зеленодольского Механического колледжа группы 207 Изосимов Д.А.

**2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

**2.1 Основание для разработки**

Работа ведётся на основании задания на дипломную работу по специальности 09.02.03 — «Программирование в компьютерных системах».

**2.2 Назначение разработки**

Программа «DoorRepairApp 1.0» работает под управление операционной системы Windows 7, Windows 8, Windows 10. Программа предназначена для автоматизации деятельности фирмы по установке и обслуживанию дверей. Цель создания программы – учёт заказов. Программа позволяет:

* создавать и редактировать базу услуг и товаров;
* создавать базу клиентов;
* оформление заказов;
* редактировать Услуга-лист.

**2.3 Требование к программному средству**

«DoorRepairApp 1.0» устанавливается на персональный компьютер, имеющий представленные минимальные системные требования:

Оперативная память: 4 Гб;

Процессор: Intel Core i / Phenom II X4 / AMD Athlon64 X2 4600 + 2.4 ГГц;

Видео карта: nVidia GeForce 8600 / ATI Radeon HD 2400;

Звуковая карта: Совместимая с DirectX;

Программный продукт «DoorRepairApp 1.0» является кроссплатформенным продуктом, что позволяет не ограничиваться в работе с определенной версией ОС.

«DoorRepairApp 1.0» написана на объектно-ориентированном языке программировании C# (си-шарп) в интегрированной среде разработки Visual Studio 2019. В качестве СУБД используется MS SQL. Программа не может подлежать сторонней оптимизации и редактированию программного кода.

**2.4 Требования к программной документации**

Состав программной документации должен включать в себя:

1. техническое задание;

2. сроки выполнения работ;

3. программу и методику испытаний;

4. эксплуатационные инструкции пользователю;

**2.5 Требования к эргономике и технической эстетике**

Система обеспечивает удобный для пользователей системный интерфейс, отвечающий следующим требованиям:

- единый стиль оформления для пользовательских интерфейсов;

- удобная, интуитивно понятная навигация в интерфейсе пользователя;

- взаимодействие пользователя с системой осуществляется на двух языках по выбору (русский и английский), для наибольшего удобства

Пользовательские интерфейсы системы спроектированы и разработаны с применением единых принципов графического представления информации и организации доступа к функциональным возможностям и сервисам. Разработан графический дизайн пользовательских интерфейсов, цветовые, шрифтовые и композиционные решения для отображения текстов, изображений, таблиц, гиперссылок, управляющих и навигационных элементов (меню, кнопок, форм и т.п.).

Экранные формы спроектированы с учетом требований унификации:

- все экранные формы пользовательского интерфейса выполнены в едином графическом дизайне, с одинаковым расположением основных элементов управления и навигации;

- для обозначения сходных операций использованы сходные графические значки, кнопки и другие управляющие (навигационные) элементы. Термины, используемые для обозначения типовых операций (добавление информационной сущности, редактирование поля данных), а также последовательности действий пользователя при их выполнении унифицированы;

- внешнее поведение сходных элементов интерфейса (реакция на наведение указателя «мыши», переключение фокуса, нажатие кнопки) реализованы одинаково для однотипных элементов.

**2.6 Стадии и этапы разработки**

Разработка должна быть проведена в три стадии:

1. разработка технического задания;

2. рабочее проектирование;

3. внедрение.

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

1. разработка программы;

2. разработка программной документации;

3. испытания программы.

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

1. постановка задачи;

2. определение и уточнение требований к техническим средствам;

3. определение требований к программе;

4. определение стадий, этапов и сроков разработки программы и документации на нее;

5. выбор языков программирования;

6. согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с пунктом Предварительный состав программной документации настоящего технического задания.

На этапе испытаний программы должны быть выполнены перечисленные ниже виды работ:

1. разработка, согласование и утверждение порядка и методики испытаний;

2. проведение испытаний;

3. корректировка программы и программной документации по результатам испытаний.

**2.7 Порядок контроля и приемки**

Сдача-приёмка работ производится поэтапно, в соответствии с рабочей программой и календарным планом. Сдача-приемка осуществляется комиссией, в состав которой входят представители ГАПОУ и работодатель. По результатам приемки подписывается акт приемочной комиссии.

В указанном подразделе, согласно таблице «Сроки выполнения работ» технического задания, будет описаны перечень участвующих организаций, место и сроки проведения работ, согласно п. 2.8 ГОСТ 34.602-89.

Порядок согласования и утверждения приемочной документации должен регламентироваться организационно-распорядительной документацией организации, принимающей участие в создании программного продукта. Согласно разделу «Приемка результатов разработки» ГОСТ 15.001-88 для согласования и утверждения приемочной документации создается приемочная комиссия (приказом).

Статус приемочной комиссии определяется Заказчиком до проведения испытаний.

**3 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ПРОГРАММНОМУ ПРОДУКТУ**

**3.1 Декомпозиция поставленной задачи**

Для достижения цели работы была поставлена следующая задача:

Создать модуль “CodeDatabase” отвечающий за сохранение БД. Результат работы модуля - сохранение БД.

Поставленные задачи наглядно отображаются на диаграмме деятельности.

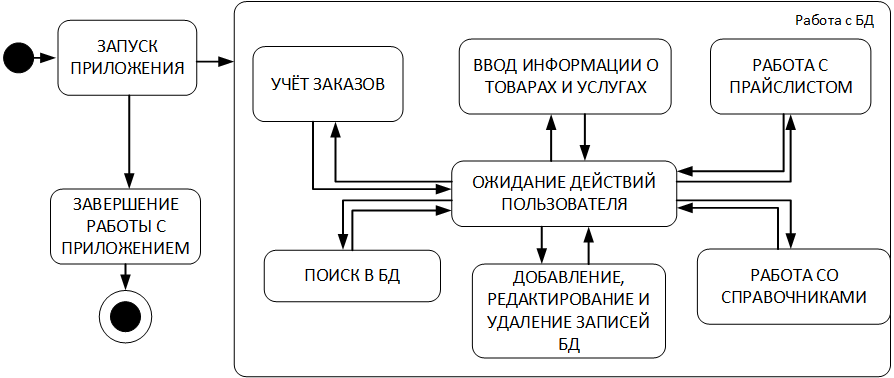


Рис.1 Декомпозиция поставленной задачи

**3.2 Общая архитектура программного средства**

Объектом моделирования является Фирма по установке и обслуживанию дверей.

Рассматриваемые процессы: учёт заказов.

Объекты моделирования представлены на диаграмме классов, рис.2.

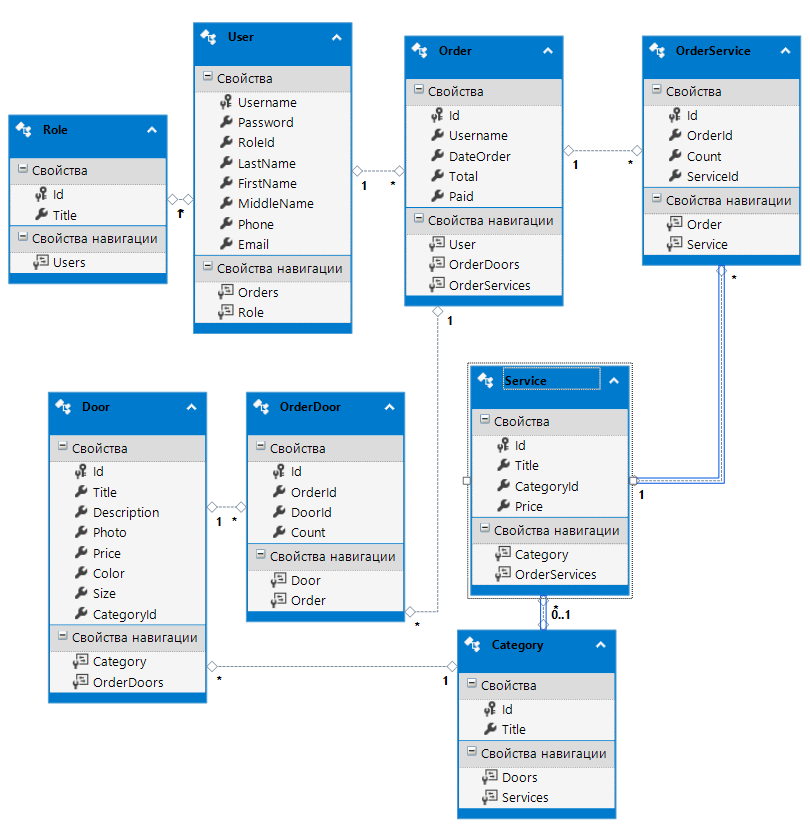


Рис.2 Диаграмма классов

По полученной декомпозиции задач была спроектирована следующая архитектура программного средства.

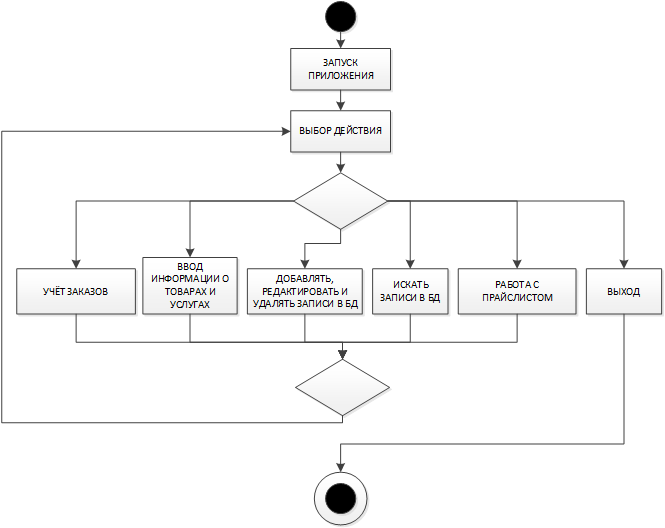


Рис.3 Схема событийно-ориентированной архитектуры

В «DoorRepairApp 1.0» используется событийно-ориентированная архитектура.(Рис.3) Роль агента (источник событий) в программе выполняют: пользователи программы, за роль стоков (потребители событий) отвечают таблицы, входящие в базу данных. Например, когда пользователь выбирает определенное действие: добавить, удалить, редактировать, вывести в Excel и т.д. система осуществит выбранные действия, и база отреагирует соответствующим образом: запись добавлена, удалена, отредактирована, либо отчет выведен в Excel. Системная архитектура пользователя рассматривает это изменение состояния как событие, создаваемое, публикуемое, определяемое и потребляемое различными приложениями в составе архитектуры.



Рис.4 Диаграмма вариантов использования

**3.3 Разработка алгоритма решения задачи**

**Общий алгоритм**

****Рис.5 Алгоритм программы

**3.4 Реализация функционального назначения программного средства**

Программа «DoorRepairApp 1.01» имеет следующий набор входных данных, такие как: товары, услуги, клиенты и Услуга-лист.

Данные вводятся администратором или пользователем в соответствующие поля ввода, снабженные всплывающими подсказками.

Выходными данными являются: каталог товаров и услуг, заказы.

Данные вводятся пользователем в соответствующие поля вывода, снабженные всплывающими подсказками.

Выходные данные редактировать вручную невозможно, т.к. они служат в качестве отчетной информации для пользователя.

**3.5 Структурная организация данных**

Для создания БД необходимо определиться с данными, которые необходимы для полноценного функционирования системы. Все эти данные указаны в реляционной модели представленной на рисунке. Любая реляционная база данных и называется реляционной, что характеризуется отношениями (relation) между таблицами. На рисунке изображены таблицы моей базы данных. При этом одна таблица является родительской (главной), а вторая – дочерней (подчиненной). Главной таблицей является «Заказ». Реляционная модель автоматизированной системы соответствует всем 12 правилам Кодда. (рис.6)

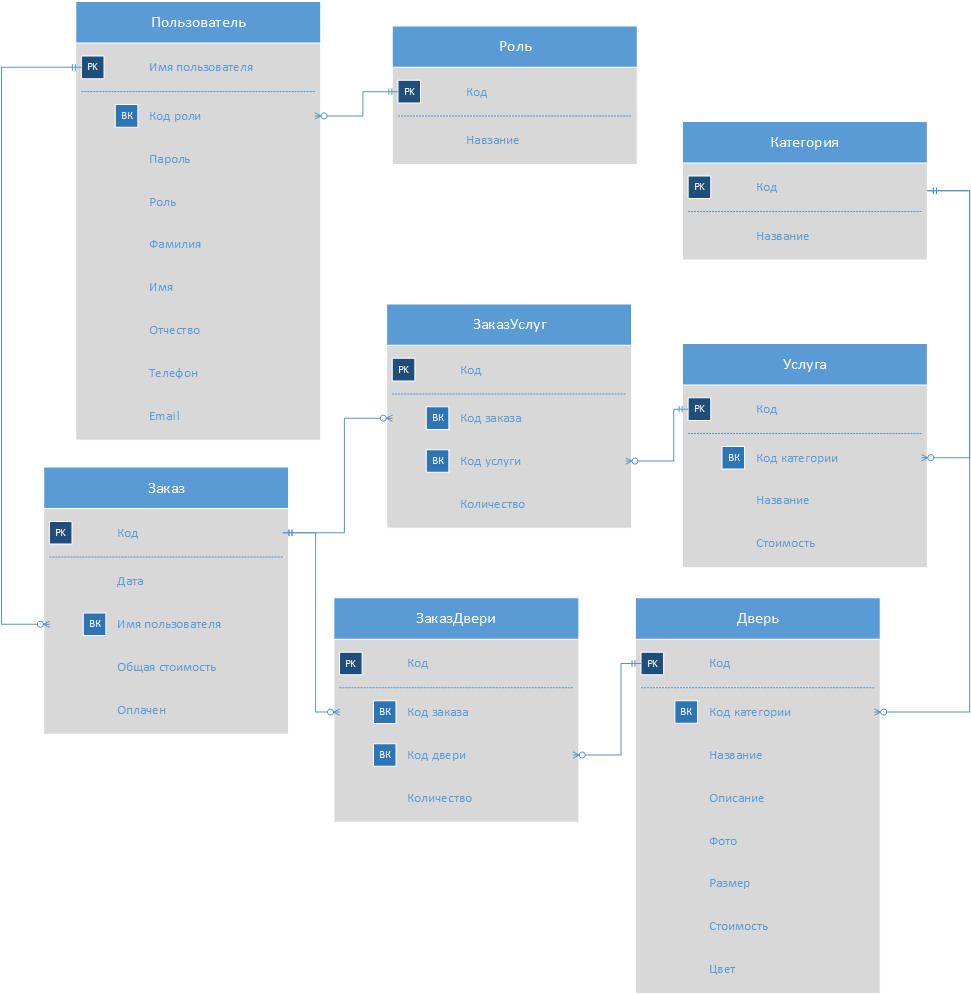


Рис.6 Реляционная модель

Первичный ключ в базе уникален, используется для организации отношений между таблицами, который не может иметь пустых и повторяющихся значений. Первичными ключами в базе являются поля: «Код» (таблица «Роль»), «Код» (Таблица «Категории»), «Код» (Таблица «ЗаказДвери»), «Код» (Таблица «Дверь»), «Код» (Таблица «Услуга»), «Код» (Таблица «Заказ услуг»), «Код» (Таблица «Заказ»), «Имя пользователя» (Таблица «Пользователь»). Остальные ID-ключи являются внешними ключами.

Для организации более эффективной обработки данных применяется нормализация. Таблицы моей БД находятся в 3НФ:

* БД находится в форме -1НФ потому, что
  + Таблица не имеет повторяющихся записей;
  + Каждый атрибут отношения хранит одно-единственное значение и не является списком, ни множеством значений;
  + Таблица не имеет повторяющихся групп полей.
    - Вторая нормальная форма(2НФ):
      * Устранены атрибуты, зависящие только от части уникального (первичного) идентификатора, т.е. ID.
        + Третья нормальная форма(3НФ):

Отсутствуют атрибуты, зависящие от атрибутов, не входящих в уникальный (первичный) ключ.

На основе реляционной модели базы данных мною в MS SQL была построена следующая база данных. (Рис.7) В ней первичным ключом является поле таблицы “Weapon” – “Id”. Связь таблиц “Category” и “Door” по полю “CategoryId” один ко многим (1:М). Связь таблиц “Door” и “OrderDoor” по полю “DoorId” один ко многим (1:1). Связь таблиц “Service” и “OrderService” по полю “ServiceId” один ко многим (1:М). Связь таблиц “Order” и “OrderService” по полю “OrderId” один ко многим (1:М). Связь таблиц “Order” и “OrderDoor” по полю “OrderId” один ко многим (1:М). Связь таблиц “Role” и “User” по полю “RoleId” один ко многим (1:М). Связь таблиц “User” и “Order” по полю “UserName” один ко многим (1:М). Все связи между таблицами базы данных типа один ко многим.

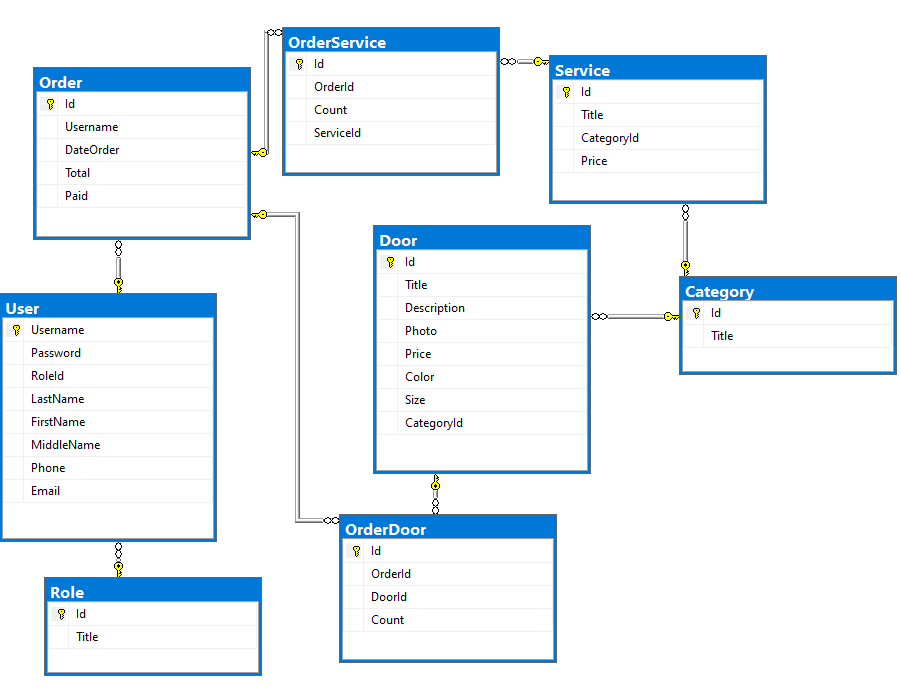


Рис.7 Схема базы данных

Таблица “Category”(Виды товаров и услуг)

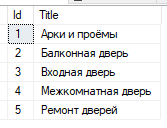


Рис.8 Таблица Category

Таблица “Service”(Услуги)

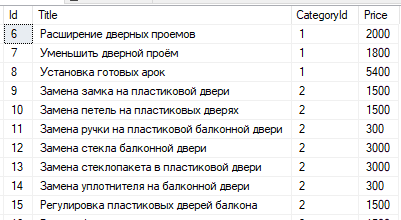


Рис.9 Таблица Service

Таблица “Door”(Дверь)

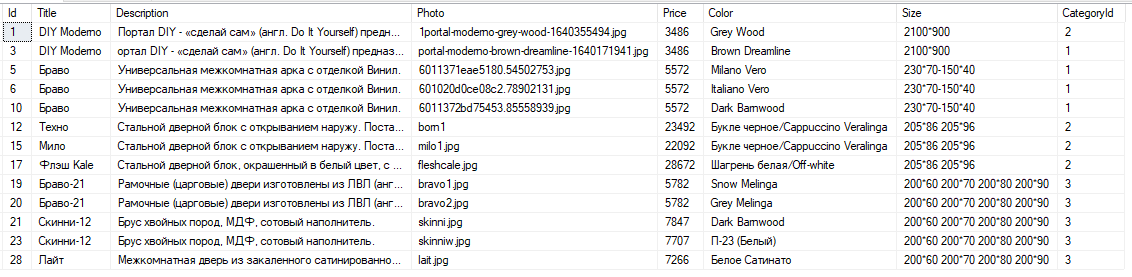


Рис.10 Таблица Weapon

Таблица “User”(Пользователи)



Рис.11 Таблица User

Таблица “Role”(Роль)

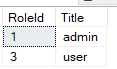


Рис.12 Таблица Role

Таблица “Order”(Заказ)

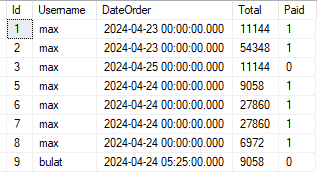


Рис.13 Таблица Order

Таблица “OrderService”(Заказ услуг)

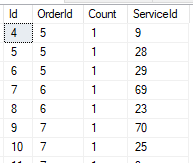


Рис.14 Таблица OrderService

Таблица “OrderDoor”(Заказ дверей )

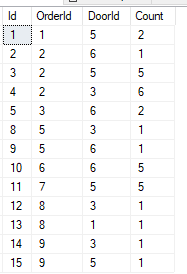


Рис.15 Таблица OrderDoor

**3.6 Разработка интерфейса ПС**

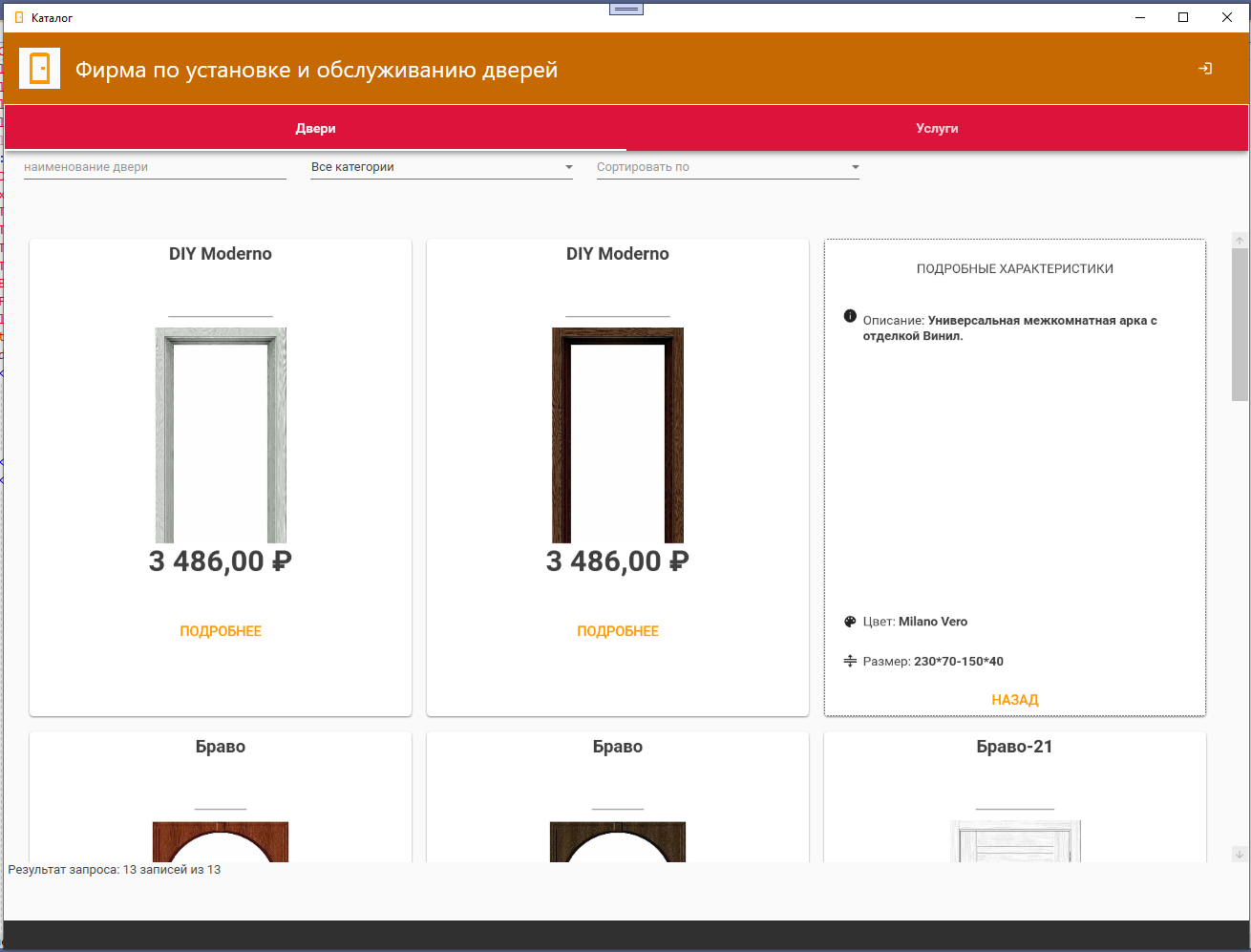
Приложение предназначено для двух типов пользователей: администратора и клиент. Для входа в панель администратора необходимо ввести свои учётные данные. Для запуска приложения достаточно щелкнуть дважды левой кнопкой мыши по ярлыку.

После запуска программы пользователь приступит к работе с главной формой программы, на которой отбражается каталог Дверьов в виде плиточного интерфейса. «DoorRepairApp 1.01» обладает удобным интерфейсом, так что даже неопытному в обращение с пользовательскими компьютерными программами пользователю не составит труда быстро привыкнуть к работе с данным программным средством.

На рабочем экране расположены несколько кнопок для комфортной работы с приложением.

**Главная форма приложения**. Основная форма приложения (Рис.16). При разработке интерфейса этой формы было использовано множество различныx компонентов:

1. ComboBox – выпадающие список для фильтрации;
2. materialDesign:Card – для отображения информации о товарах и услугах в виде карточки;
3. Buttons – кнопки для перехода на другие формы;
4. TextBox – поля для ввода.



Button

ComboBox

TextBox

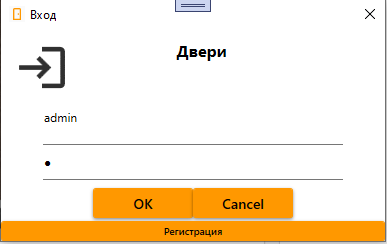
materialDesign:Card

TextBox

Button

Рис.16 Фирма по установке и обслуживанию дверей

**Форма входа в систему**. Интерфейс данной формы состоит из компонентов: Button – кнопки ОК, Cancel, Регистрация, Grid – сетка для аккуратного размещения компонентов, TextBox – поле для ввода логина, PasswordBox – поле для ввода пароля и TextBlock – текстовые метки. (Рис.17)



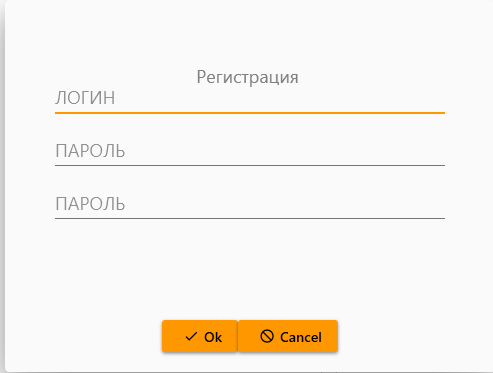
Buttons

TextBox

PasswordBox

Рис.17 Форма входа в систему

**Форма Регистрации**. Интерфейс данной формы состоит из компонентов: Button – кнопки ОК, Cancel, Grid – сетка для аккуратного размещения компонентов, TextBox – поле для ввода логина, PasswordBox – поле для ввода пароля и TextBlock – текстовые метки. (Рис.18)



Buttons

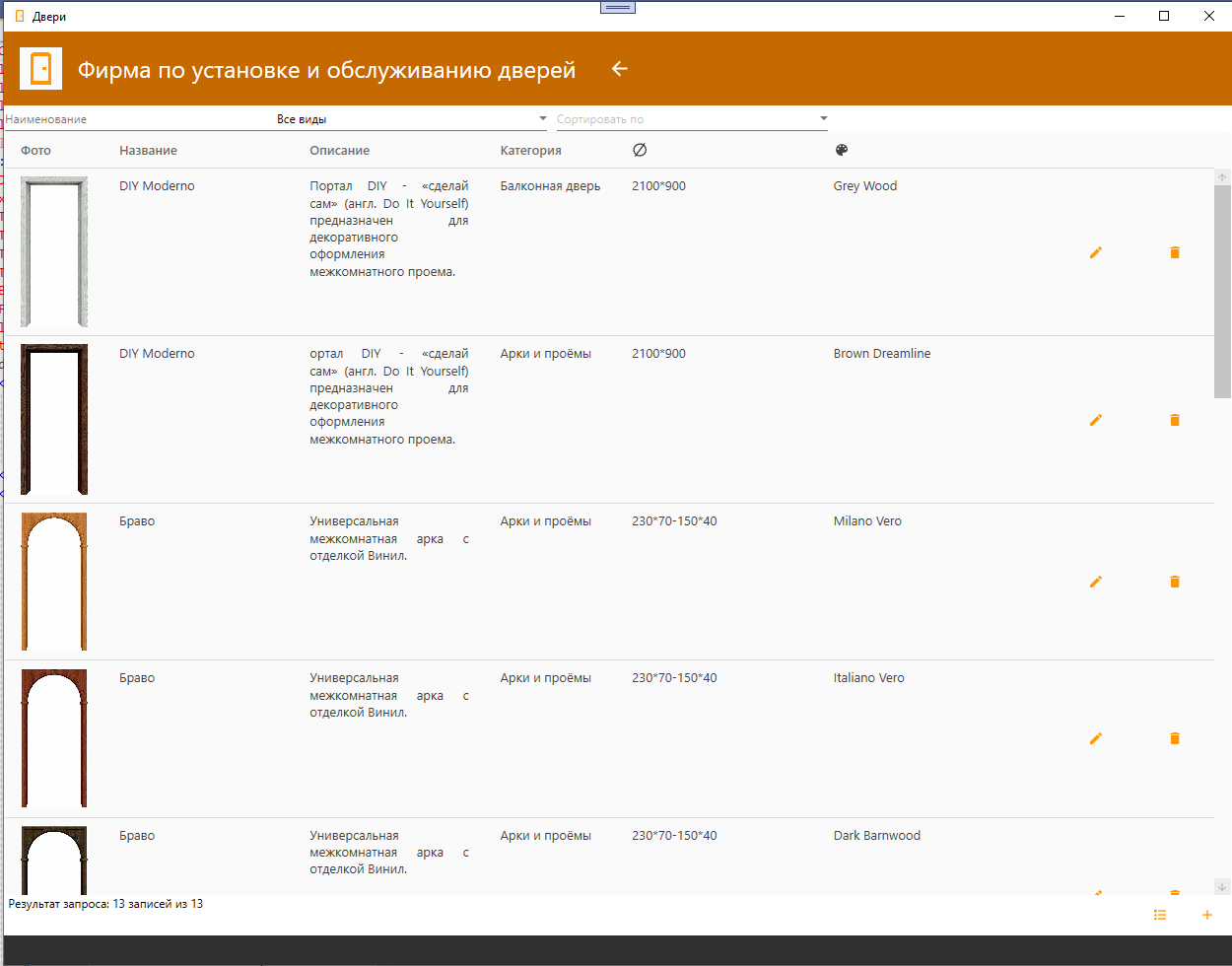
TextBox

PasswordBox

Рис.18 Форма Регистрации

**Форма Двери(**Рис.19**)**. При разработке интерфейса этой формы были использованы компоненты:

1. DataGrid – отображение списка дверей в виде таблицы;
2. Image – для отображения изображения;
3. Buttons – кнопки для перехода к справочникам, добавления, редактирования и удаления записей;
4. TextBlock – текстовая метка;



Button

DataGrid

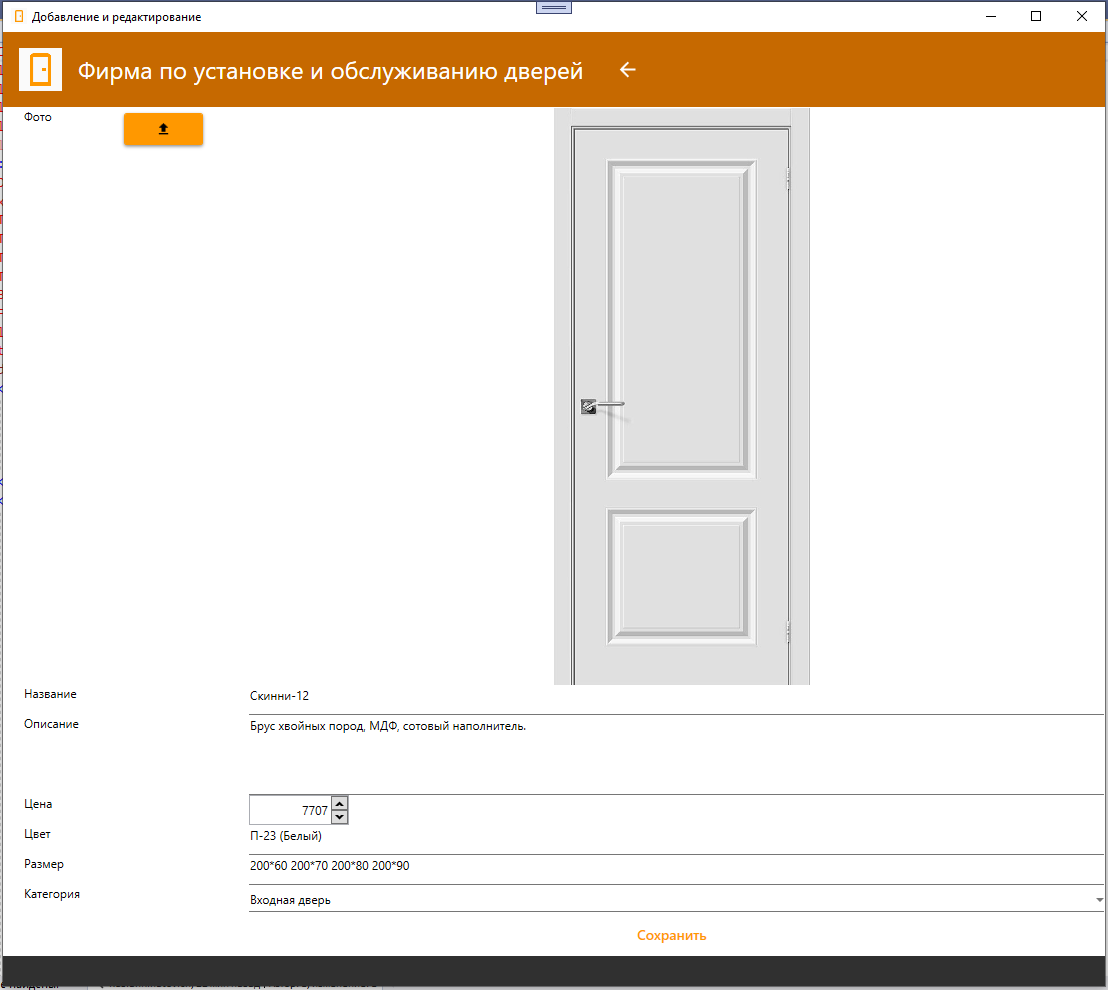
Image

TextBlock

Рис.19 Форма Двери

**Страница добавления и редактирования(**Рис.20**)**. При разработке интерфейса этой формы были использованы компоненты:

1. ComboBox – выпадающий список;
2. Buttons – кнопки Сохранения, Загрузки и перехода назад;
3. IntegerUpDown –поле для целых чисел;
4. TextBox – поля для ввода текста
5. Image – для отображения изображения;
6. TextBlock – текстовые метки;



Buttons

Image

TextBlock

ComboBox

TextBox

Button

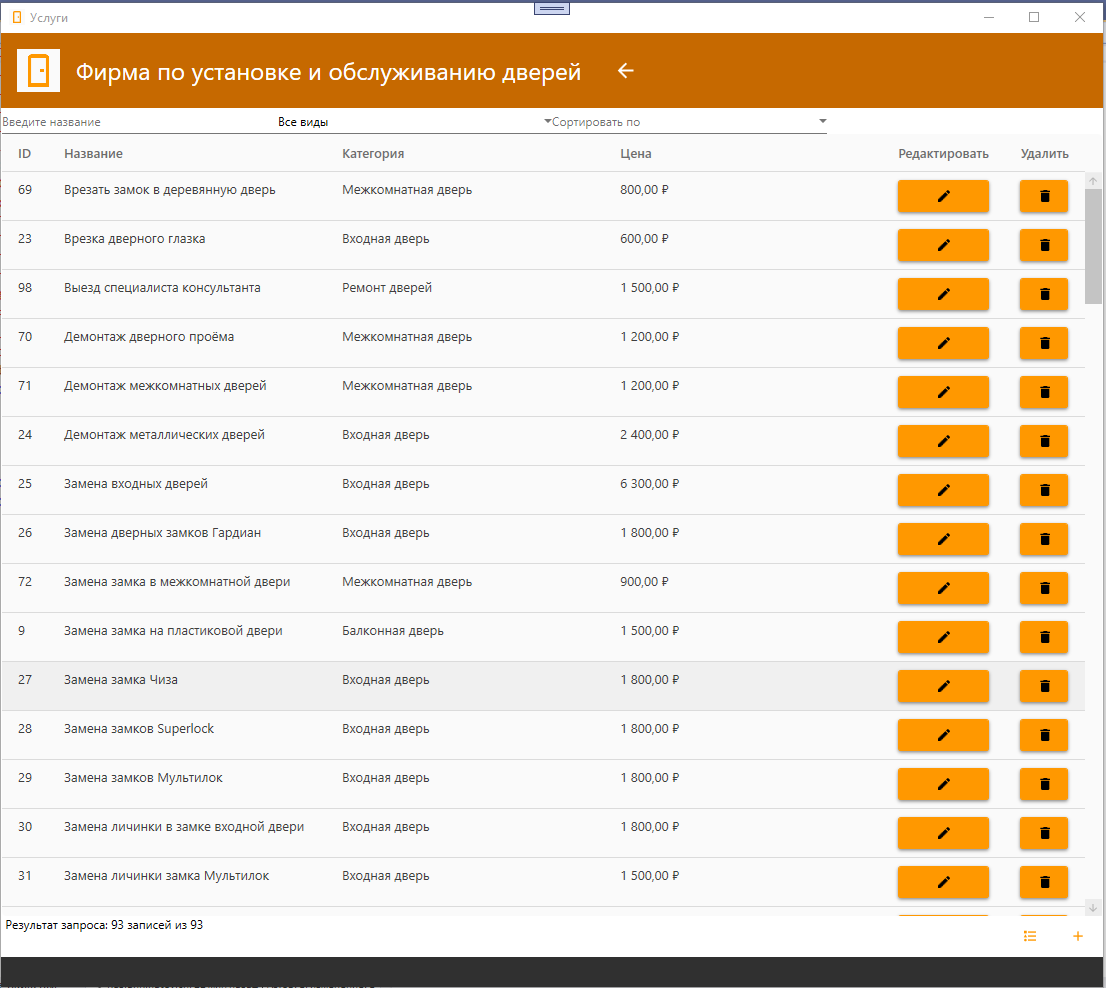
IntegerUpDown

Grid

Рис.20 Страница добавления и редактирования

**Форма Услуги(**Рис.21**)**. При разработке интерфейса этой формы были использованы компоненты:

1. DataGrid – отображение списка услуг в виде таблицы;
2. Buttons – кнопки для перехода к справочникам, добавления, редактирования и удаления записей;
3. TextBlock – текстовая метка;



Button

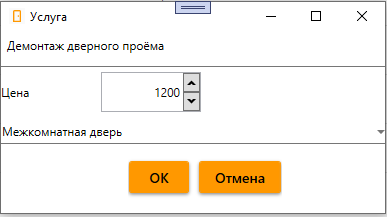
DataGrid

TextBlock

Рис.21 Форма Услуги

**Форма Услуга(**Рис.22**)**. При разработке интерфейса этой формы были использованы компоненты:

1. ComboBox – выпадающий список;
2. Buttons – кнопки Сохранения, Загрузки и перехода назад;
3. IntegerUpDown –поле для целых чисел;
4. TextBox – поля для ввода текста
5. TextBlock – текстовые метки;



TextBlock

ComboBox

TextBox

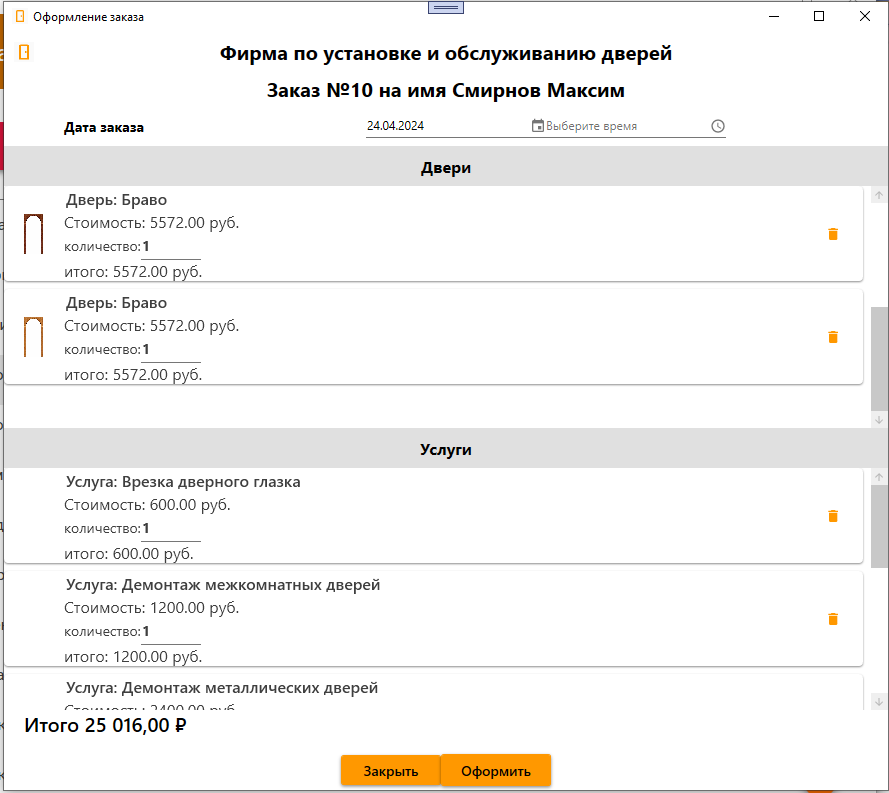
Button

IntegerUpDown

Рис.22 Форма Услуга

**Форма «Оформление заказа».** (Рис.23). При разработке интерфейса этой формы использованы компоненты:

1. DatePicker – выбор даты;
2. ListBox – списки товаров и услуг;
3. Image – для отображения изображения;
4. Buttons – кнопки для перехода на другие формы.



Button

DatePicker

TextBlock

ListBox

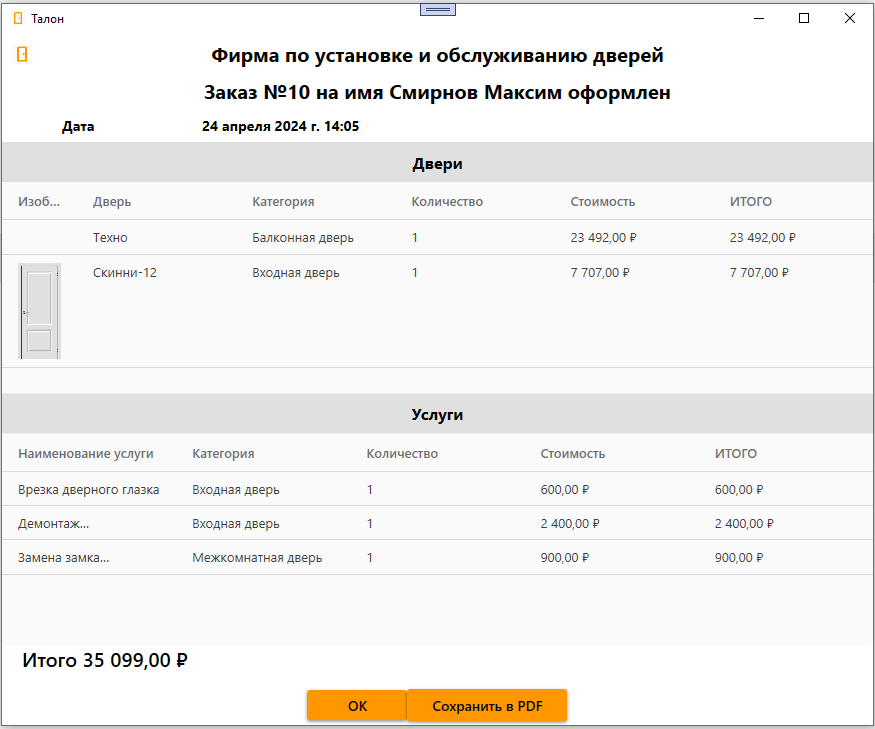
TextBlock

Image

Рис.23 Форма «Оформление заказа»

**Форма «Талон».** (Рис.24). При разработке интерфейса этой формы использованы компоненты:

1. TextBlock – текстовые метки;
2. DataGrid – список товаров;
3. Image – для отображения изображения;
4. Buttons – кнопки для перехода на другие формы.



Button

TextBlock

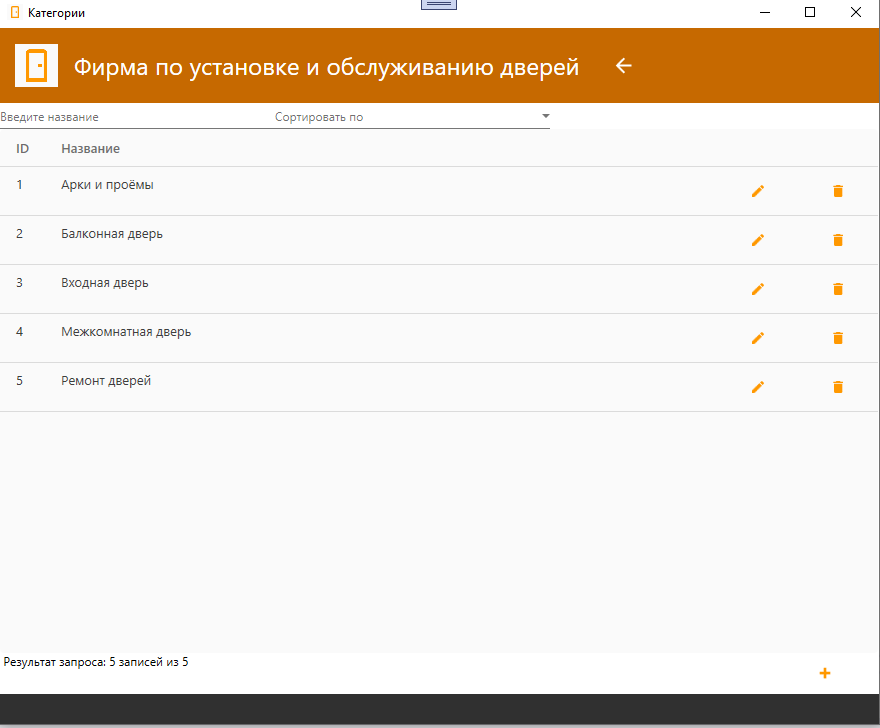
DataGrid

TextBlock

Image

Рис.24 Форма «Талон»

**Страница Категории**. Интерфейс данной формы состоит из компонентов: DataGrid и кнопок: Добавить, Изменить, Удалить, Назад. (Рис.25)



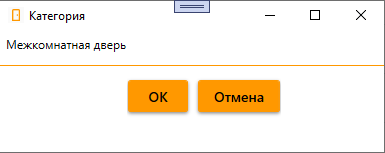
Button

Button

DataGrid

Рис.25 Категории

**Окно Категория**. Интерфейс данной формы состоит из компонентов: TextBox – текстовое поле ввода и кнопок Button. (Рис.26)

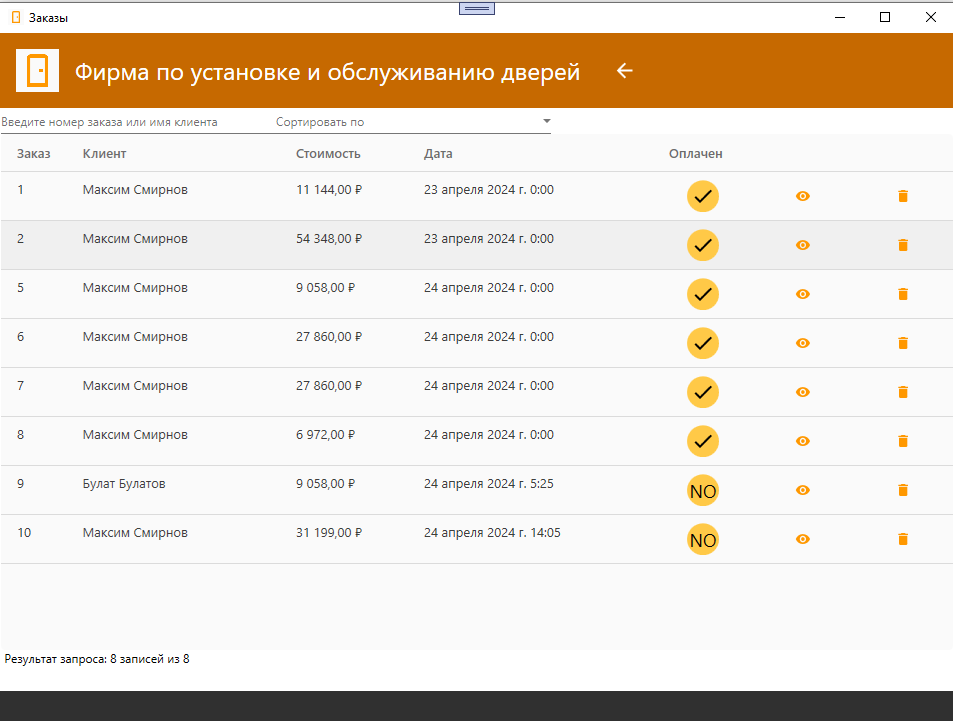


Button

TextBox

Рис.26 Окно Категория

**Страница Заказы**. Интерфейс данной формы состоит из компонентов: DataGrid, Combobox, и кнопок: Добавить, Изменить, Удалить, Назад. (Рис.27)



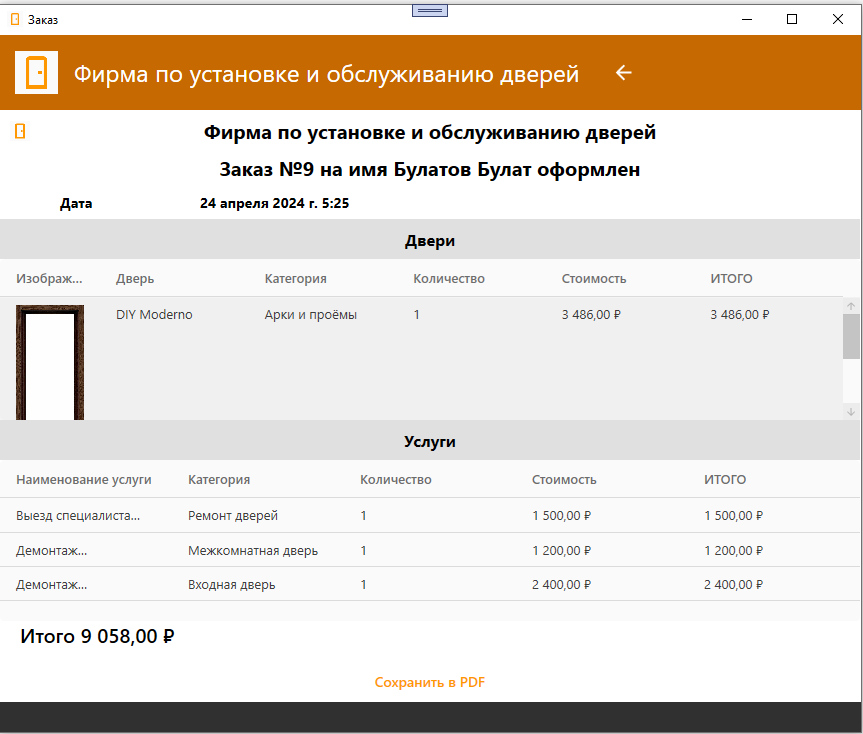
Button

ComboBox

DataGrid

Рис.27 Заказы

**Страница Заказ**. Интерфейс данной формы состоит из компонентов: DataGrid, TextBlock, и кнопок: Добавить, Изменить, Удалить, Назад. (Рис.28)



Button

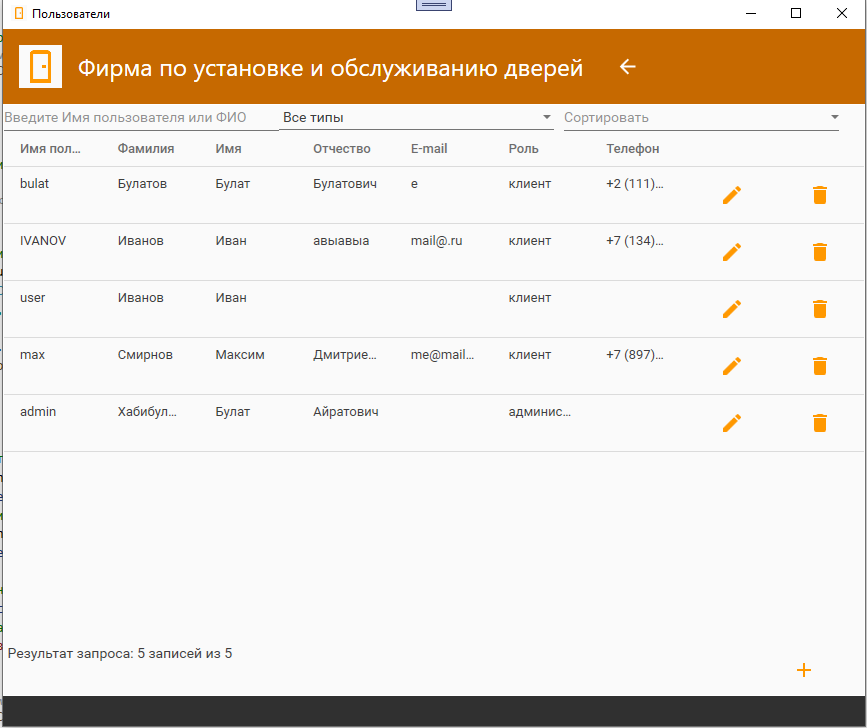
TextBlock

DataGrid

Рис.28 Заказ

**Страница Пользователи** (Рис. 29). При разработке интерфейса этой формы были использованы следующие компоненты:

* DataGrid – данные;
* ComboBox –фильтрация и сортировка пользователей;
* Button – кнопка для выполнения команд пользователя;
* TextBox – поля для ввода



Button

DataGrid

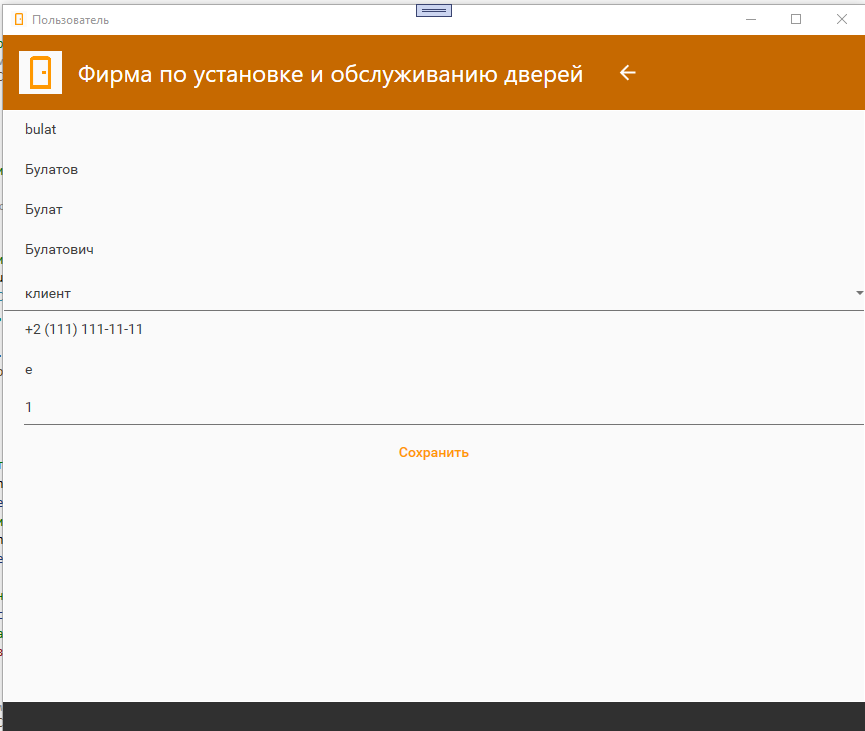
ComboBox

TextBox

Рис.29 Пользователи

**Страница Пользователь** (Рис. 30). При разработке интерфейса этой формы были использованы следующие компоненты:

* ComboBox – выпадающие списки;
* Button – кнопка для выполнения команд пользователя;
* TextBox – поля для ввода



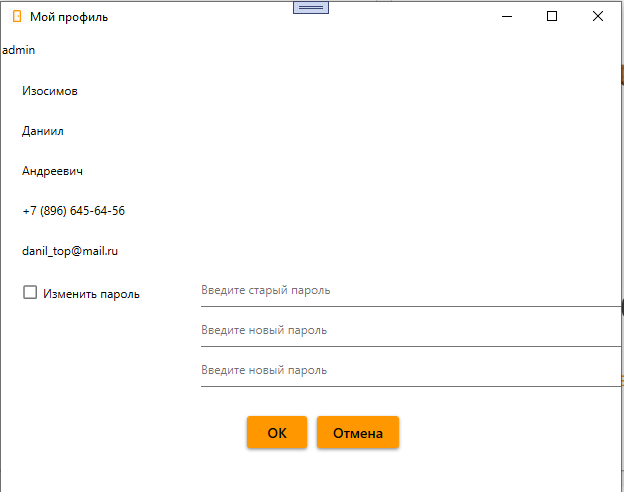
Button

ComboBox

TextBox

Рис.30 Пользователи

**Форма Мой профиль**. Интерфейс данной формы состоит из компонентов: Button – кнопки ОК и Отмена, Grid – сетка для аккуратного размещения компонентов, TextBox – поле для ввода данных, PasswordBox – поле для ввода пароля и TextBlock – текстовые метки. (Рис.31)



Buttons

Grid

TexBox

PasswordBox

CheckBox

Рис.31 Форма Мой профиль

**3.7 Описание структуры выходной информации**

Выходными данными являются список дверей.

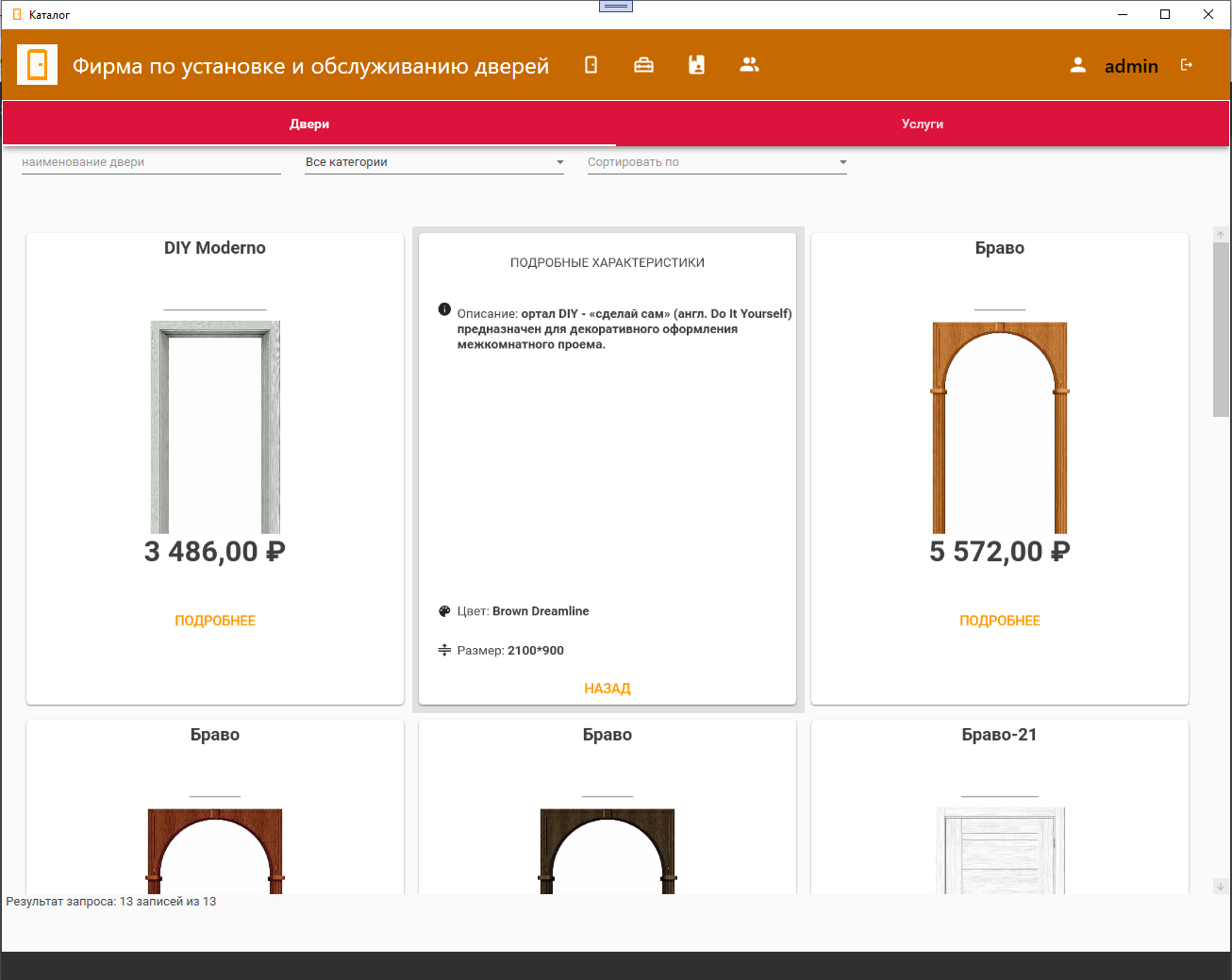


Рис.32 Каталог дверей

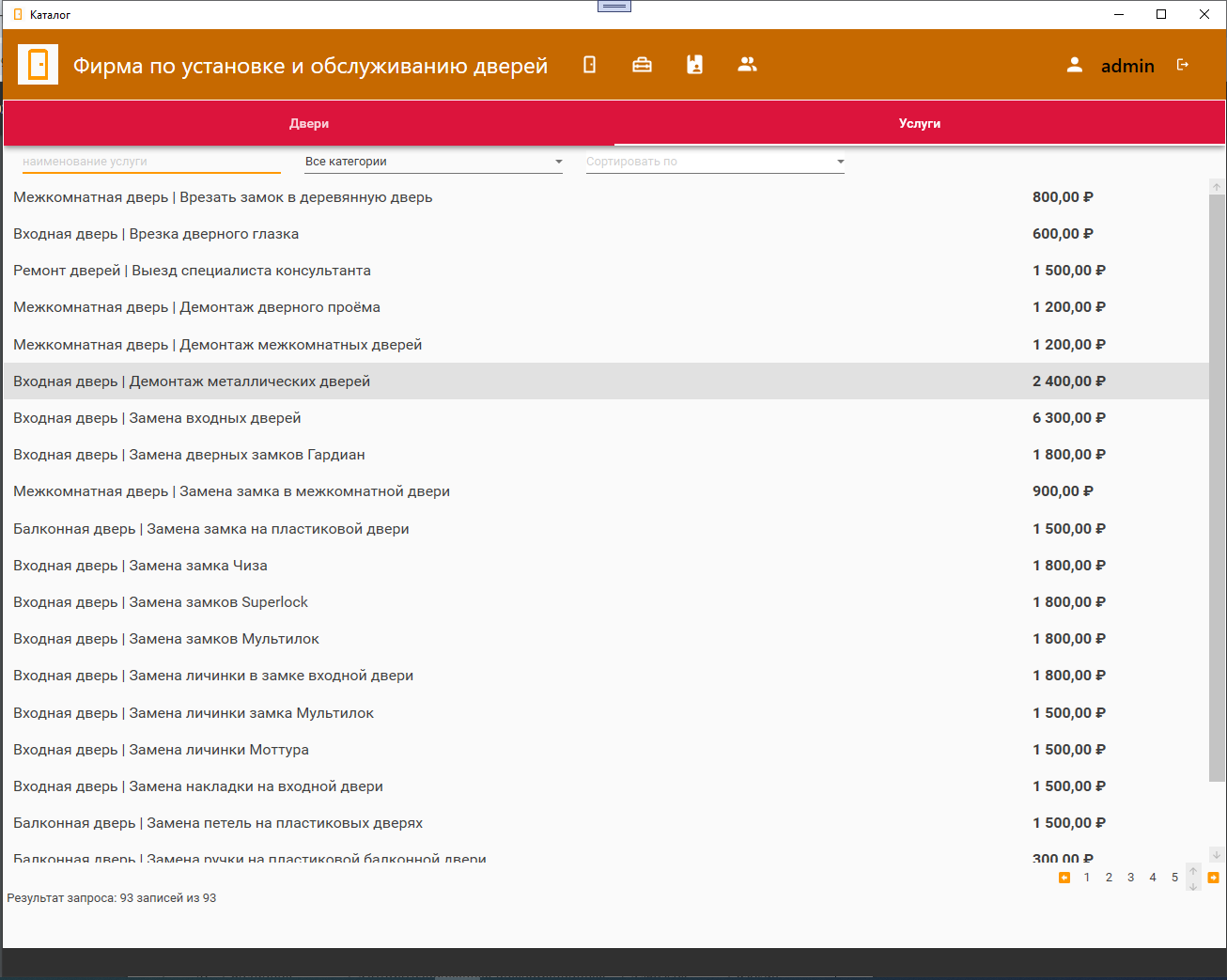


Рис.33 Каталог услуг

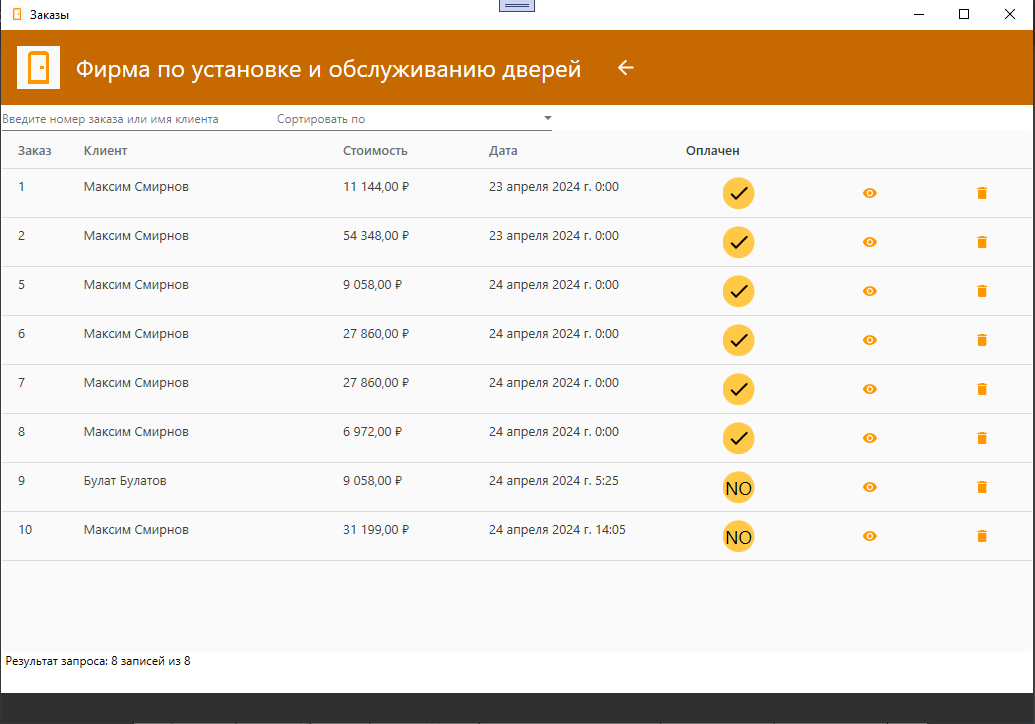


Рис.34 Заказы

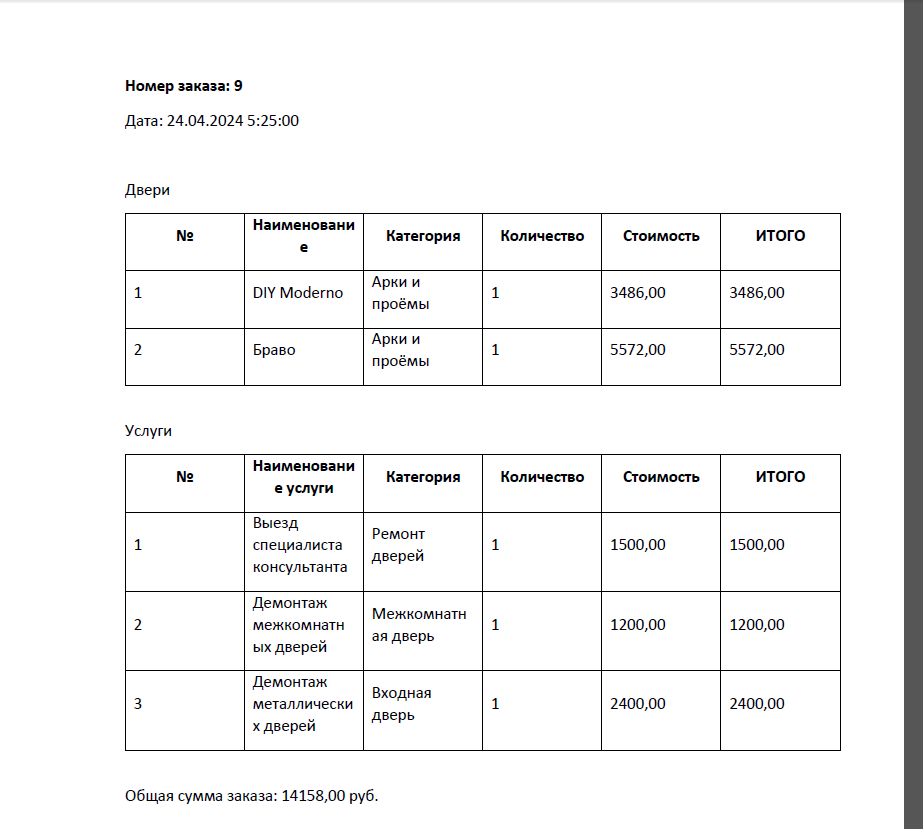


Рис.35 Заказ

**4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

**4.1 Назначение программного средства**

Назначением программного средства является автоматизация деятельности компании.

**4.2 Условия выполнения программного средства**

Условия, при соблюдении которых возможно выполнение работы с программным продуктом:

-Компьютер с установленной на нём «DoorRepairApp 1.0».

Подготовительные действия:

-На иконке «DoorRepairApp 1.01» рабочего стола произвести двойной щелчок левой кнопкой мыши.

**4.3 Эксплуатация программного средства**

Этот раздел содержит обзор приложений «DoorRepairApp 1.0» и ее основных параметров.

Активировать копию программы DoorRepairApp 1.0 можно сразу после её установки, когда появится окно подтверждения подлинности Вашей версии. Ключ активации расположен внутри упаковки программного продукта. Для успешного выполнения активации ключ активации необходимо вводить именно в том виде, в котором он предоставлен.

Вводные сведения об интерфейсе пользователя

После запуска приложения отображается стартовое окно приложения (Рис.36). Главная форма представляет собой каталог дверей в виде плиточного интерфейса. Каждая дверь отображается в виде карточки с краткой информацией: название и фотография. На оборотной стороне карточки в виде списко отображается подробная информация о товаре. Пользователь имеет возможность отфильтровать список по категории.

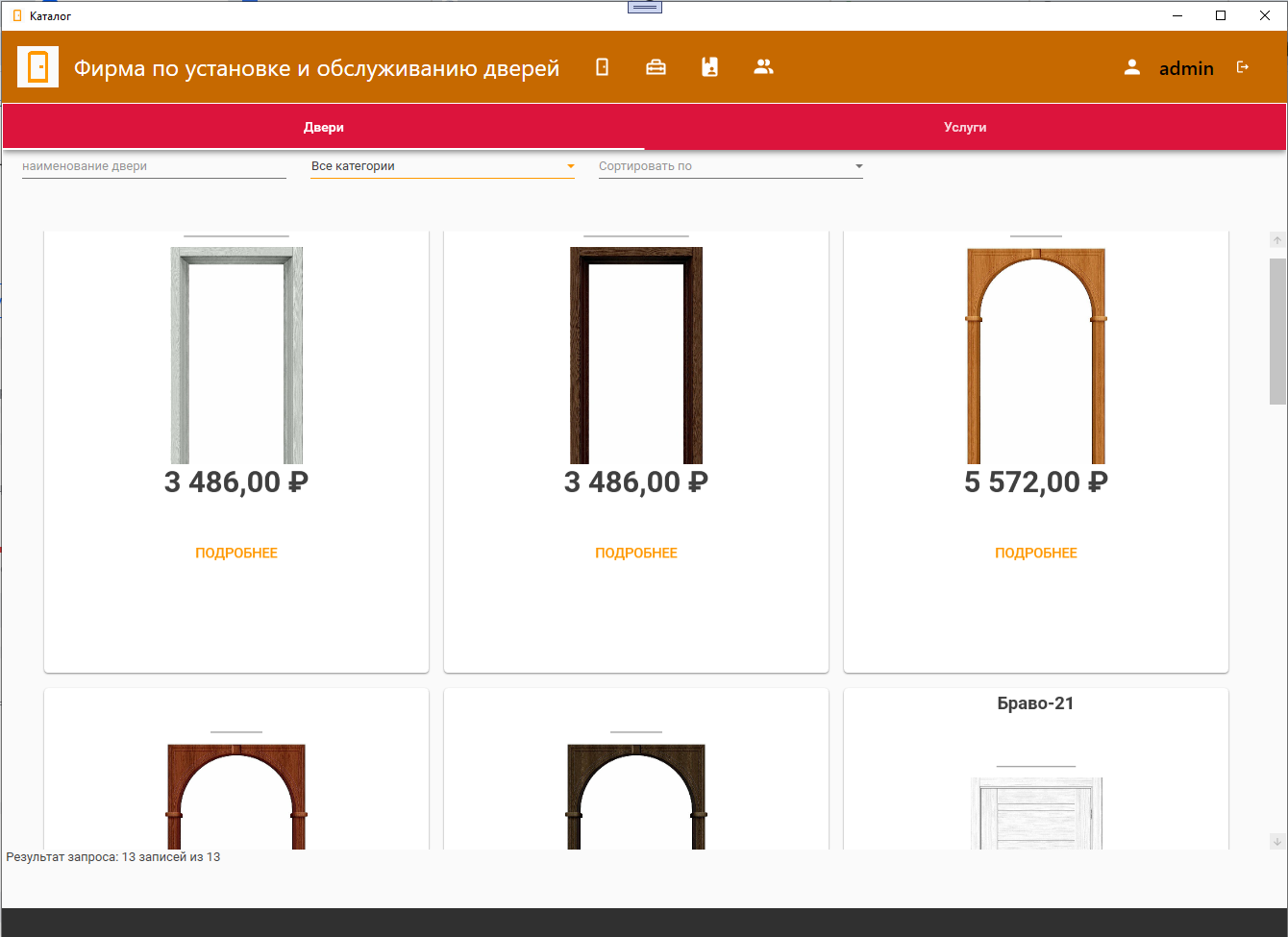


Рис.36 Окно Фирма по установке и обслуживанию дверей

Кликнув по кнопке «Закрыть» программа потребует подтверждение. (Рис.37)



Рис.37 Сообщение программы

Для входа в систему необходимо кликнуть на кнопку . Ввести логин и пароль администратора или логин и пароль пользователя и нажать OK(Рис. 38).

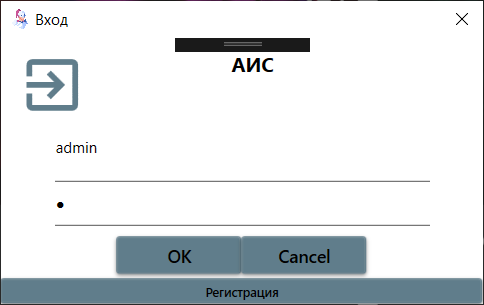


Рис.38 Форма входа в систему

Если вы являетесь новым пользователем в системе, то необходимо нажать на кнопку Регистрация. В появившейся форме требуется заполнить поля логина и пароля. При регистрации роль нового пользователя всегда клиент(Рис. 39).

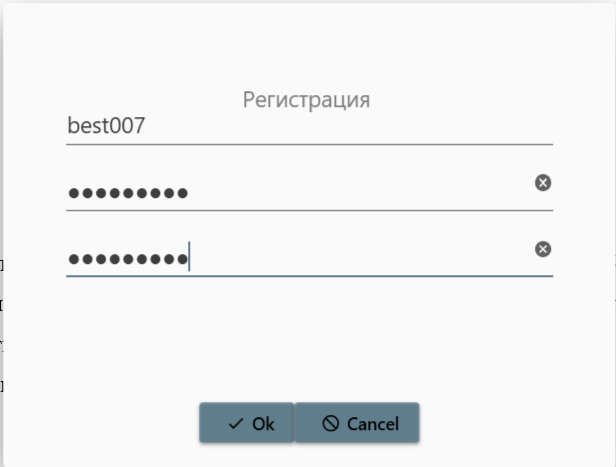


Рис.39 Форма регистрации

После входа в систему появятся кнопки для доступа к различным компонентам системы.

Для того чтобы выйти из системы нажмите на кнопку , программа потребует потдверждения(Рис.40)

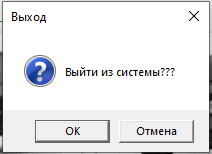


Рис.40 Сообщение программы

Для отображения подробной информации о двери нужно нажать на кнопку подробнее. Отобразится подробная информация о двери(рис. 41)

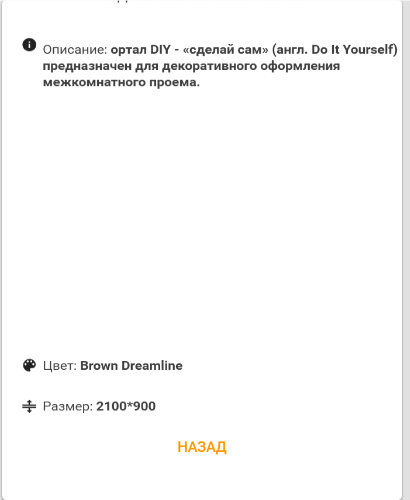


Рис.41 Подробная информация

Ниже описаны пункты меню управления и информации.

Страница «Двери». Эта форма доступна только администратору. Для того, чтобы открыть эту форму нужно нажать на кнопку «Двери» , расположенную на главной форме приложения. Данная страница предназначена для отображения информации о дверях в табличном виде, открытия страницы редактирования, добавления, удаления выделенной записи, а также для перехода на другие страницы приложения. (Рис.42)

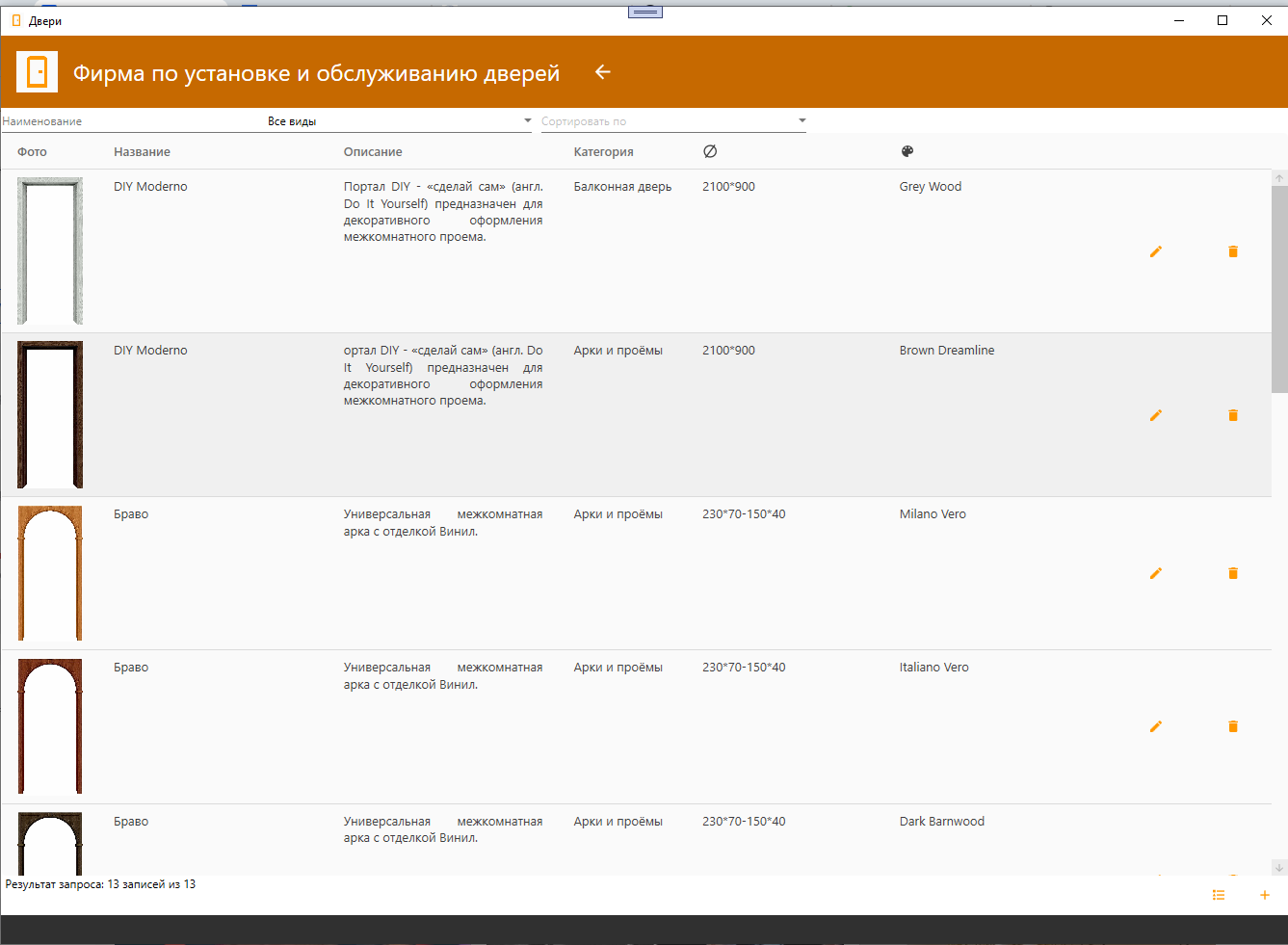
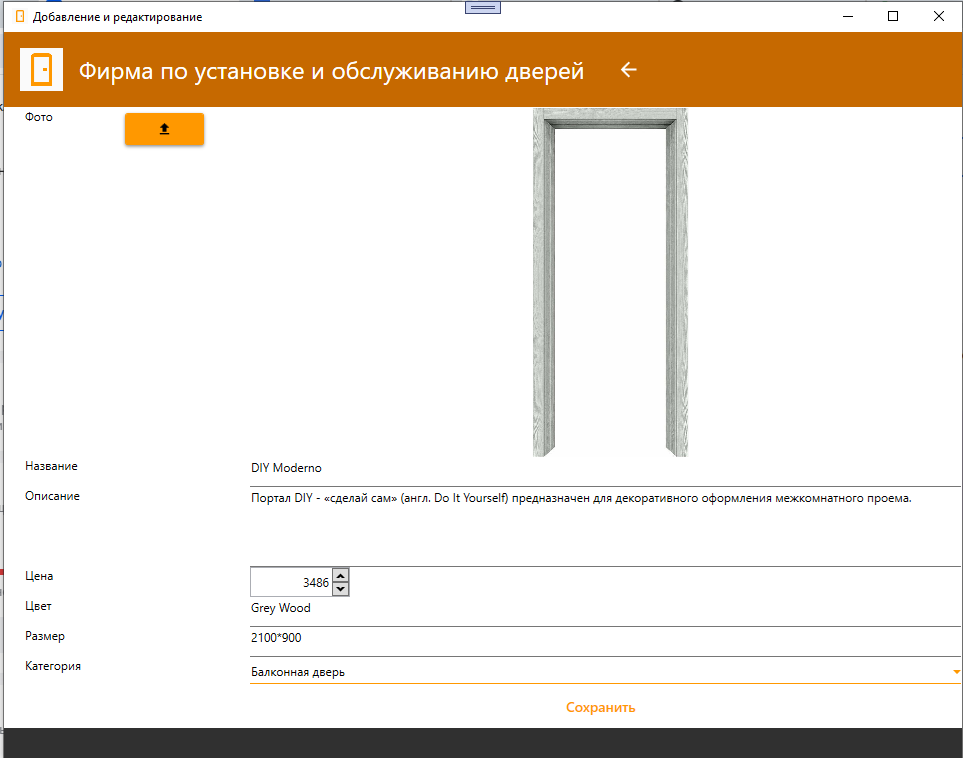


Рис.42 Страница Двери

Страница Добавление и редактирование. (Рис.43) Для открытия этой страницы нужно на странице «Двери» выбрать запись и нажать на кнопку Редактировать. Также данную страницу можно открыть в режиме добавления, нажав на соотвестсвующую кнопку на предыдущей странице. Эта страница предназначена для добавления или редактирования сведений о двери.

 Рис.43 Страница Добавление и редактирование

«Категории»: эта страница предназначена для просмотра и удаления информации о категориях. Добавление и редактирование осуществляется через дополнительную форму. (Рис.44)

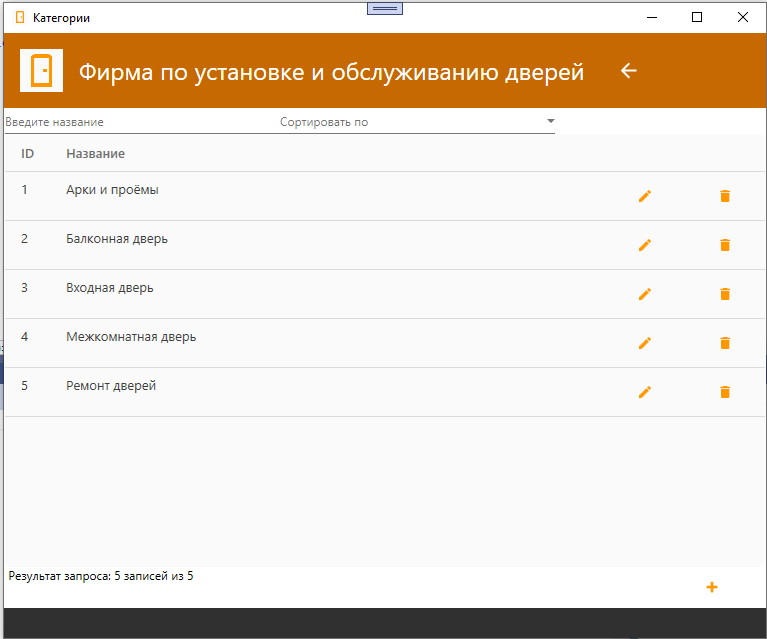


Рис.44 Категории

«Категория»: эта форма предназначена для добавления и редактирования записи. (Рис.45)

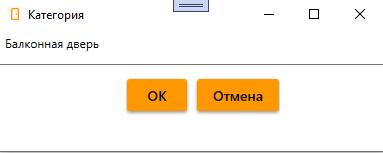


Рис.45 Категория

«Услуги»: эта страница предназначена для просмотра и удаления информации о стоимости услуг. Добавление и редактирование осуществляется через дополнительную форму. (Рис.46)

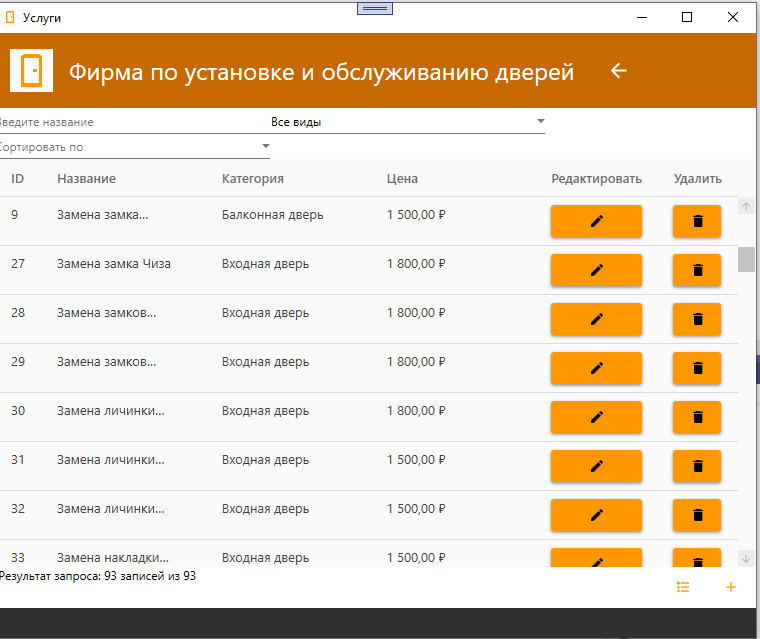


Рис.46 Страница Услуги

«Услуга»: эта форма предназначена для добавления и редактирования записи о стоимости услуги. (Рис.47)

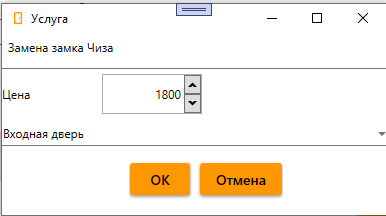


Рис.47 Форма услуга

«Заказы»: эта страница предназначена для просмотра заказов и изменения информации об оплате. (Рис.48)

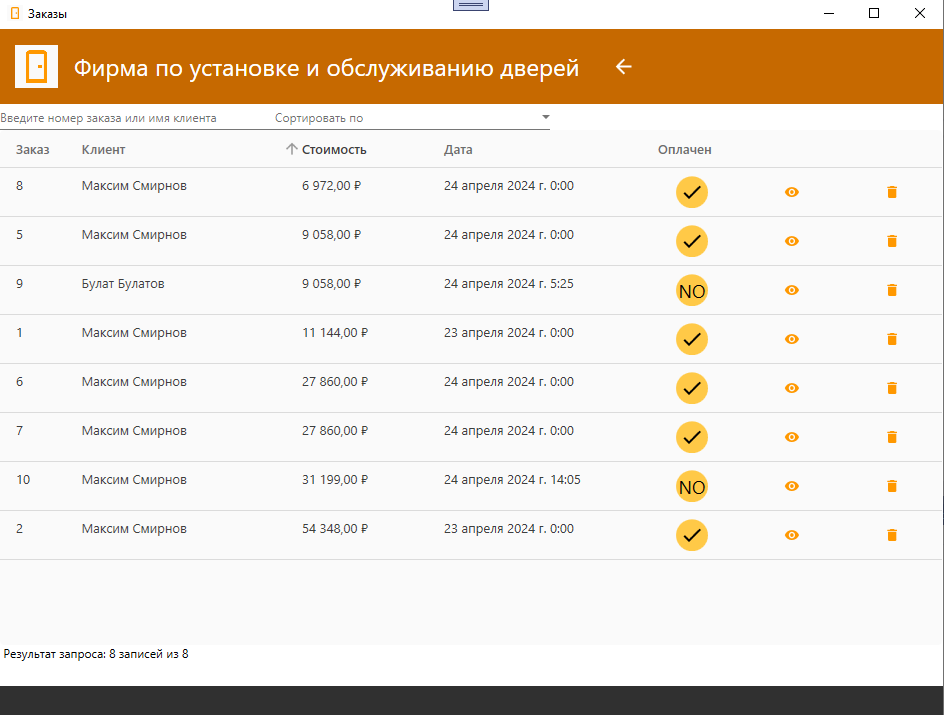


Рис.48 Страница Заказы

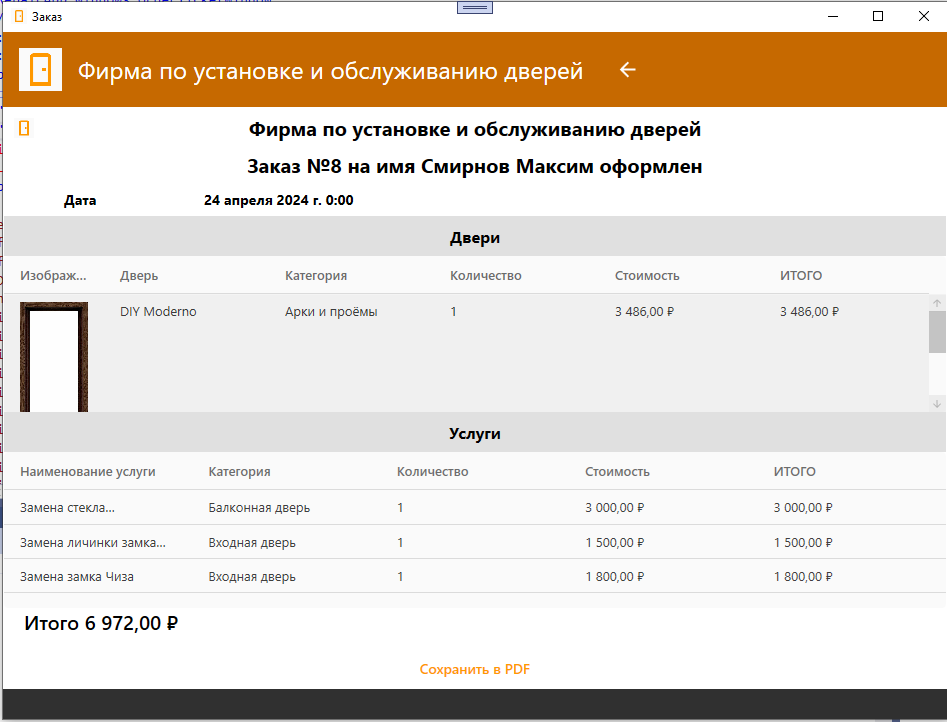


Рис.49 Страница Заказ

«Пользователи»: эта страница предназначена для просмотра и удаления информации о пользователях. Добавление и редактирование осуществляется через дополнительную форму. (Рис.50)

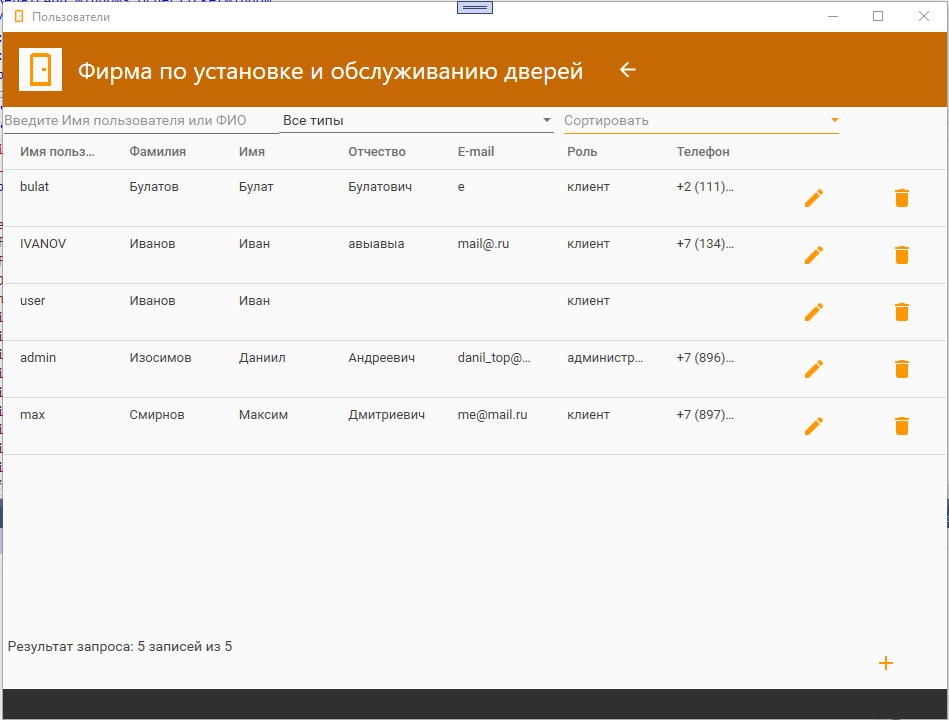


Рис.50 Страница Пользователи

«Пользователь»: эта форма предназначена для добавления и редактирования записи о пользователе. (Рис.51)

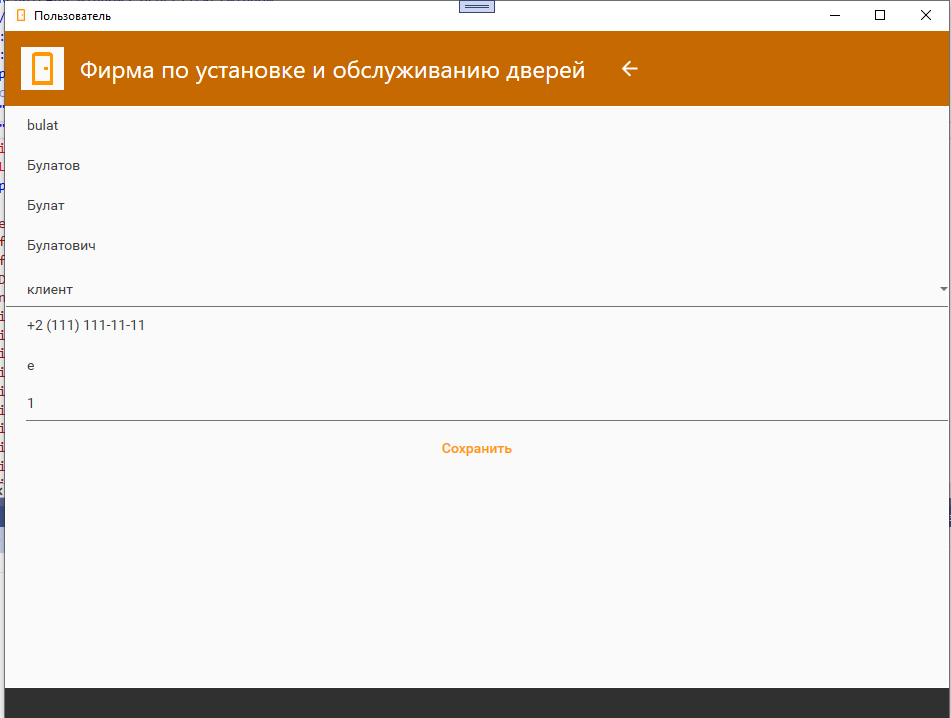


Рис.51 Форма Пользователь

«Мой профиль»: эта форма предназначена для просмотра и редактирования информации о пользователе. (Рис.52)

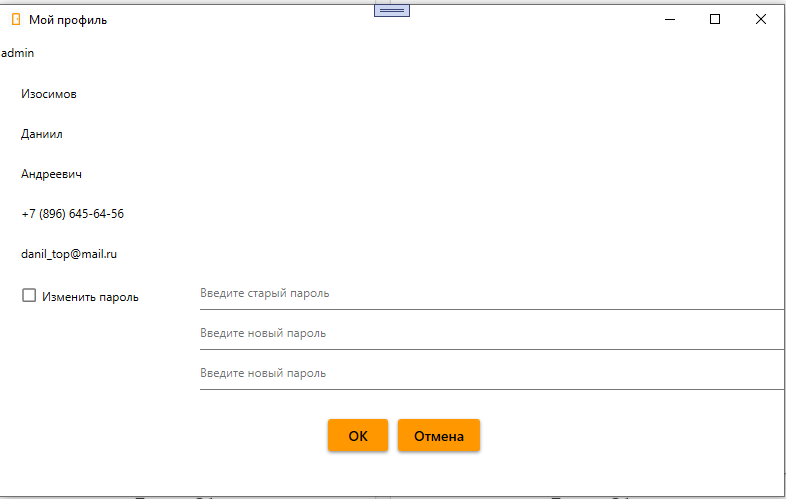


Рис.52 Форма Мой профиль

Работа с DoorRepairApp 1.01

После запуска программы Вы перейдете на главную форму программы. В формах «Дверь», «Категории», «Услуга-лист», заполняем все необходимые для работы пункты.

После этого кликаем по кнопке «Добавить запись».

Если Вы желаете удалить любую введенную ранее информацию, Вам необходимо кликнуть по нужной кнопке.

Для того чтобы изменить запись, введите в редактируемые поля данные и нажмите кнопку сохранить. Данные формы редактированию и управлению со стороны пользователя не подлежат.

После того, как Вы закончите сеанс работы с программой, если Вы хотите выйти из DoorRepairApp 1.01 кликните на соответствующей кнопке «Выход».

Основные кнопки DoorRepairApp 1.01

Кнопки управления «Добавить запись», «Удалить запись», «Сохранить» расположены на каждой конкретной форме в базе данных.

Действия этих кнопок приведены в таблице 1:

Таблица 1 – Действие кнопок

|  |  |
| --- | --- |
| Добавить запись | После нажатия на эту кнопку выйдет вспомогательная форма для добавления записей |
| Удалить запись | Кнопка «Удалить запись» позволяет удалить выделенную запись. |
| Сохранить данные | Кнопка «Сохранить данные» добавляет запись введенную пользователем в БД, либо обновляет существующую. |
| Изменить | Открывает форму в режиме редактирования информации |
|  | Позволяет Загрузить файл |
|  | Переход на предыдущую страницу |
|  | Перейти на страницу Услуга-лист |

**4.4 Сообщения пользователю**

Для уверенной и комфортной работы пользователю важно понимать поведение программы, а программе, в свою очередь, поведение пользователя. Для информирования и уточнения намерений пользователей в программах используются сообщения.

При попытке удалить запись, пользователю выводится сообщение с запросом на подтверждение или отмену действия. (Рис.53)



Рис.53 Сообщение Подтверждение удаления

После добавления новой записи, пользователю выводится сообщение. (Рис.54)



Рис.54 Сообщение Запись добавлена

После обновления записи, пользователю выводится сообщение. (Рис.55)



Рис.55 Сообщение Запись обновлена

При попытке удалить связанные записи, программа выводит пользователю сообщение о невозможности действия. (Рис.56)



Рис.56 Сообщение Ошибка удаления

**5 РАЗРАБОТКА ТЕСТОВОГО НАБОРА**

**5.1 Определение верхней границы количества тестов**

Для разработанного алгоритма был построен граф.

****

Рис.57 Потоковый граф

Необходимое количество тестов определяется по формуле V(G) = E-N+2, где E – это число ребер, а N–это число вершин данного графа. Согласно представленному графу E=34, N=25. Далее по формуле рассчитаем необходимое количество тестовых сценариев (тест-кейсов).

V(G)=34-25+2=11.

Для данного программного средства необходимо разработать не менее 11 тестовых сценариев.

Чтобы определить количество тестовых прогонов для программного продукта DoorRepairApp 1.0 необходимо выделить требования. Из требований к приложению выделим поддержку 2-х операционных систем с одним основным языком локализации и выполнения 6-ти основных функций. Кроме того, приложение позволяет выполнять 5 функций по взаимодействию с системой (запуск приложения, выход из приложения, сохранение результатов в файл, работа с буфером и т.п.).

Таким образом, полное покрытие требований задаёт набор из 2\*1\*(6+5) =22 тестовых прогонов.

**5.2 Разработка тестовых вариантов**

Таблица 2 – Тестирование корректного добавления информации в Категории

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного добавления информации в таблицу Категории | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Категории | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Название = Межкомнатные двери | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Категории открыта | | Отображается форма Категории |  |

Таблица 3 – Тестирование некорректного добавления информации в таблицу Категории

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации в таблицу Категории | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Категории | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Название = | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, так как поле пустое | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Категории открыта | | Отображается форма Категории |  |

Таблица 4 – Тестирование корректного редактирования информации в таблице Категории

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного редактирования информации в таблице Категории | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Категории | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  Название = Межкомнатные двери | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Категории  открыта | | Отображается форма Категории |  |

Таблица 5 – Тестирование некорректного редактирования информации в таблице Категории

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного редактирования информации в таблице Категории | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Категории | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  Название = | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, поля пустые | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Категории  открыта | | Отображается форма Категории |  |

Таблица 6 – Тестирование корректного добавления информации в таблицу Дверь

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного добавления информации в таблицу Дверь | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Дверь | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Изображение= загружено  Название = DIY Moderno  Категория = Балконная дверь  Цвет = белый  Цена = 3748  Размер = 8 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Дверь открыта | | Отображается форма Дверь |  |

Таблица 7 – Тестирование некорректного добавления информации в таблицу Дверь

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации в таблицу Дверь | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Дверь | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Изображение= загружено  Название =  Категория = Балконная дверь  Цвет = белый  Цена = 3748  Размер = 8 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, поля пустые | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Дверь  открыта | | Отображается форма Дверь |  |

Таблица 8 – Тестирование корректного редактирования информации в таблице Дверь

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного редактирования информации в таблице Дверь | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Дверь | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  Изображение= загружено  Название = DIY Moderno  Категория = Балконная дверь  Цвет = белый  Цена = 3748  Размер = 8 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Дверь открыта | | Отображается форма Дверь |  |

Таблица 9 – Тестирование некорректного редактирования информации в таблице Дверь

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного редактирования информации в таблице Дверь | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Дверь | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  Изображение= загружено  Название =  Категория = Балконная дверь  Цвет = белый  Цена = 3748  Размер = 8 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, поля пустые | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Дверь открыта | | Отображается форма Дверь |  |

Таблица 10 – Тестирование корректного добавления информации в Услуга

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного добавления информации в таблицу Услуга | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Услуга | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Категория = Входная дверь  Услуга = Врезка дверного глазка  Стоимость = 600 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Услуга открыта | | Отображается форма Услуга |  |

Таблица 11 – Тестирование некорректного добавления информации в таблицу Услуга

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации в таблицу Услуга | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Услуга | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Категория = Входная дверь  Услуга =  Стоимость = 600 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, так как поле пустое | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Услуга открыта | | Отображается форма Услуга |  |

Таблица 12 – Тестирование корректного редактирования информации в таблице Услуга

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного редактирования информации в таблице Услуга | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Услуга | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  Категория = Входная дверь  Услуга = Врезка дверного глазка  Стоимость = 600 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Услуга открыта | | Отображается форма Услуга |  |

Таблица 13 – Тестирование некорректного редактирования информации в таблице Услуга

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного редактирования информации в таблице Услуга | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Услуга | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  Категория =  Услуга = Врезка дверного глазка  Стоимость = 600 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, поля пустые | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Услуга открыта | | Отображается форма Услуга |  |

**5.3 Составление отчета о тестировании**

Для форм проекта были созданы и проведены тесты, результаты сведены в таблицу:

Таблица 14 – Результаты тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Прохождение | | |
| Форма | Число ТП | пройдено | | не пройдено |
| Главная | 4 | 4 | | 0 |
| Дверь | 4 | 4 | | 0 |
| Категории | 4 | 4 | | 0 |
| Услуга | 4 | 4 | | 0 |
| Заказы | 4 | 4 | | 0 |
| Профиль | 4 | 4 | | 0 |
| ИТОГО | 24 | 24 | 0 | |

По результатам тестирования были выявлены и устранены следующие проблемы:

При выполнении теста таблица №15 Тестирование формы «Услуга», Некорректный запрос

**Ошибка**



Рис.57 Ошибка теста №8

**Причина**

Поле стоимость имеет тип вещественное число, было введено строковое значение.

**Принятые меры**

Ошибка устранена. В поле широта можно ввести только цифры и запятую

**ЗАКЛЮЧЕНИЕ**

**Оценка качества программного средства с помощью метрик**

Оценка качества программного средства проводится исходя из стандарта оценки качества ISО 9126.

На компьютере следующей конфигурации:

Установленная операционная система – Windows 10 Professional.

Процессор: Intel Pentium G2030 3.0 GHz.

ОЗУ: 4 Gb

Таблица 15 – Требования к количественным характеристикам качества программного средства

|  |  |  |  |
| --- | --- | --- | --- |
| **Характеристики качества** | **Мера** | **Требуемое значение** | **Реальное значение** |
| **Надежность** |  | | |
| *Завершенность:*  *-* наработка на отказ при отсутствии рестарта; | Часы | 10 | 10 |
| *Устойчивость:*  - относительные ресурсы на обеспечение надежности и рестарта. | % | 0,5 | 0,5 |
| *Восстанавливаемость:*  -длительность восстановления. | Минуты | 2 | 2 |
| *Доступность-готовность:*  -относительное время работоспособного функционирования. | Вероятность | 0,998 | 0,998 |
| **Эффективность** |  | | |
| *Временная эффективность:*  *-*время отклика - получение результатов на типовое задание;  - пропускная способность число типовых заданий, исполняемых в единицу времени. | Секунды  Число в минуту | 3  60 | 3  60 |
| *Используемость ресурсов:*  *-* относительная величина использования ресурсов ЭВМ при нормальном функционировании программного средства. |  | 0,001 | 0,001 |

Таблица 16 – Требования к качественным характеристикам программного средства.

|  |  |  |  |
| --- | --- | --- | --- |
| **Характеристики качества** | **Мера** | **Требуемое значение** | **Реальное значение** |
| **Практичность** |  | | |
| *Простота использования:*  **-**среднее время ввода заданий;  -среднее время отклика на задание. | Секунды  Секунды | 10  3 | 10  3 |
| *Изучаемость :*  - трудоемкость изучения применения ПС;  -продолжительность изучения;  -объем эксплуатационной документации; | Чел.-часы  Часы  Страницы | 5  2  15 | 5  2  15 |
| **Сопровождаемость** |  | | |
| *Изменяемость:*  -трудоемкость подготовки изменений;  -длительность подготовки изменений. | Чел.-часы  Часы | 5  5 | 5  5 |
| *Тестируемость:*  *-трудоемкость тестирования изменений;*  *- длительность тестирования изменений.* | Чел.-часы  Часы | 3  3 | 3  3 |
| **Мобильность** |  | | |
| *Адаптируемость:*  - трудоемкость адаптации;  -длительность адаптации. | Чел.-часы  Часы | 0,2  1 | 0,2  1 |
| *Простота установки:*  -трудоемкость инсталляции;  -длительность инсталляции. | Чел.-часы  Часы | 0,2  1 | 0,2  1 |
| *Замещаемость:*  -трудоемкость замены компонентов;  - длительность замены компонентов. | Чел.-часы  Часы | 0,2  0,2 | 0,2  0,2 |

Таблица 17 – Сложность

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Тип объекта | Количество | Вес | | | Итого |
|  |  | Простой | Средний | Сложный |  |
| Экран | 17 | x1 | x2 | **x3** | =51 |
| Отчет | 3 | x2 | x5 | **x8** | =24 |

Таблица 18 – Сложность экрана

|  |  |  |  |
| --- | --- | --- | --- |
| Экраны | Количество клиентских (клт) таблиц данных | | |
| Количество представлений | Всего <4 | Всего <8 | **Всего >8** |
|  | (<2 срв, <3 клт) | (2-3 срв, 3-5 клт) | (>3 срв, >5 клт) |
| <3 | Простой | Простой | Средний |
| 3-7 | Простой | Средний | Сложный |
| **>8** | Средний | Сложный | **Сложный** |

Таблица 19 – Сложность отчета

|  |  |  |  |
| --- | --- | --- | --- |
| Отчеты | Количество Возвратских (клт) таблиц данных | | |
| Количество представлений | Всего <4 | Всего <8 | Всего >8 |
|  | (<2 срв, <3 клт) | (2-3 срв, 3-5 клт) | **(>3 срв, >5 клт)** |
| 0 или 1 | Простой | Простой | Средний |
| **2 или 3** | Простой | Средний | **Сложный** |
| >4 | Средний | Сложный | Сложный |

Производительность = 6000 тысяч строк кода/1 месяца=6000 тысяч строк кода/месяц

Качество = 30 ошибок /6 тысяч строк кода=5 ошибок/тысяч строк кода

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Г. Шилдт C#: Учебный курс. - Спб.: Питер, 2021. - 512с

2. Либерти, Д. Программирование на Си Шарп. - М.: Символ-плюс, 2018. - 684 с.

3. Ватсон К. Си Шарп/К.Ватсон.-М.:Лори, 2021.-862 с

4. Балена Ф., Димауро Д. Современная практика программирования на Visual Basic и Visual C#.-М.:Русская редакция, 2021,604с.

5. Енин А., Енин Н. Локальная СУБД своими руками. Учимся на примерах.-М.:СОЛОН-ПРЕСС, 2021,464с.,СД

6. Лабор В.В. Си Шарп: СоВозврат приложений для Windows/-Мн.: Харвест, 2021-384с.

7. Фролов А.В., Фролов Г.В. Визуальное проектирование приложений C#.-М.:КУДИЦ-ОБРАЗ, 2021-512с.

8. ГОСТ 19.201-78.ЕСПД. Техническое задание. Требования к содержанию и оформлению

9. ГОСТ 19.102-77.ЕСПД. Стадии разработки

10. ГОСТ 19.505-79.ЕСПД. Руководство оператора. Требования к содержанию и оформлению

11. ГОСТ 19.002-80.ЕСПД. Схемы алгоритмов и программ. Правила выполнения

12. ГОСТ 19.404-79.ЕСПД. Пояснительная записка. Требования к содержанию и оформлению

13. Владимир Биллинг. Основы программирования на С#. 2021 г.

[Электронный ресурс http://mexalib.com/view/42671]

14. Никита Культин. Основы программирования в MicrosoftVisualC# 2019. 2019 г.

[Электронный ресурс http://www.litres.ru/nikita-kultin/osnovy-programmirovaniya-v-microsoft-visual-c-2010-2]

15. Фролов А.В., Фролов Г.В. Визуальное проектирование приложений C#.-М.:КУДИЦ-ОБРАЗ, 2021.

[Электронный ресурс http://www.twirpx.com/file/27852]

16. Йен Гриффитс. Программирование на С#. 2020г.

[Электронный ресурс http://rutracker.org/forum/viewtopic.php?t=4772853]

**Приложение А**

Фирма по установке и обслуживанию дверей

Текст кода программного средства

460.ЗМК.00010-01 12 01

**Текст кода программного средства**

<?xml version="1.0" encoding="utf-8"?>

<configuration>

<configSections>

<!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkID=237468 -->

<section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false" />

</configSections>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />

</startup>

<entityFramework>

<defaultConnectionFactory type="System.Data.Entity.Infrastructure.SqlConnectionFactory, EntityFramework" />

<providers>

<provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />

</providers>

</entityFramework>

<connectionStrings>

<add name="YogaFeatPilatesBDEntities" connectionString="metadata=res://\*/Models.YogaPilatesModel.csdl|res://\*/Models.YogaPilatesModel.ssdl|res://\*/Models.YogaPilatesModel.msl;provider=System.Data.SqlClient;provider connection string=&quot;data source=.\sqlexpress;initial catalog=YogaFeatPilatesBD;integrated security=True;MultipleActiveResultSets=True;App=EntityFramework&quot;" providerName="System.Data.EntityClient" /></connectionStrings>

</configuration>

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

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

xmlns:local="clr-namespace:WpfAssortmentCheck">

<!--цвета приложения для соответсвия стилю-->

<SolidColorBrush x:Key="main" Color="#FFF"/>

<SolidColorBrush x:Key="additional" Color="#FFD700"/>

<SolidColorBrush x:Key="akcent" Color="#FFF19208"/>

<!--Стиль для окна-->

<Style TargetType="Window" x:Key="base\_window">

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

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

<Setter Property="MinHeight" Value="600"/>

<Setter Property="MinWidth" Value="800"/>

<!--<Setter Property="Background" Value="{StaticResource main}"/>-->

</Style>

<Style TargetType="Window" x:Key="base\_window\_1">

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

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

<!--<Setter Property="Background" Value="{StaticResource main}"/>-->

</Style>

<!--стиль для страницы-->

<Style TargetType="Page" x:Key="base\_page">

<Setter Property="FontFamily" Value="Arial"/>

<!--<Setter Property="Background" Value="{StaticResource main}"/>-->

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

</Style>

<!--стиль для полей ввода TextBox-->

<Style TargetType="TextBox">

<Setter Property="Height" Value="30"/>

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

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

<Setter Property="Padding" Value="3 0"/>

<Setter Property="Background" Value="{StaticResource main}"/>

</Style>

<!--стиль для полей кнопок Button-->

<Style TargetType="Button">

<Setter Property="Width" Value="auto"/>

<Setter Property="Height" Value="auto"/>

<Setter Property="Background" Value="{StaticResource akcent}"/>

<Setter Property="Foreground" Value="{StaticResource main}"/>

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

<Setter Property="FontWeight" Value="Bold"/>

<Setter Property="Margin" Value="3"/>

<Setter Property="Padding" Value="5"/>

<Setter Property="Height" Value="30"/>

</Style>

<!--стиль для меток TextBlock -->

<Style TargetType="TextBlock" x:Key="base\_textblock">

<Setter Property="Height" Value="30"/>

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

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

<Setter Property="Padding" Value="5"/>

<Setter Property="Background" Value="{StaticResource additional}"/>

</Style>

<!--стиль для меток TextBlock -->

<Style TargetType="TextBlock" x:Key="item\_textblock">

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

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

<Setter Property="TextWrapping" Value="Wrap"/>

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

<Setter Property="Padding" Value="3"/>

</Style>

<!--стиль для меток ComboBoxItem -->

<Style TargetType="ComboBoxItem">

<Setter Property="Background" Value="{StaticResource main}"/>

<Setter Property="Height" Value="40"/>

</Style>

<!--стиль для меток ListViewItem -->

<Style TargetType="ListViewItem" x:Key="good\_item">

<Setter Property="Background" Value="{Binding GetColor}" />

<Setter Property="BorderBrush" Value="{StaticResource akcent}"/>

<Setter Property="BorderThickness" Value="1"/>

</Style>

</ResourceDictionary>

<Application x:Class="WpfAssortmentCheck.App"

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

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

xmlns:local="clr-namespace:WpfAssortmentCheck"

StartupUri="windows/MainWindow.xaml">

<!--строчка StartUpUrl отвечает за запускаемое со стартом приложения окно-->

<!--Первое запускаемое окно будет главным в приложении.-->

<Application.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="Dictionary1.xaml"/>

<ResourceDictionary Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/Themes/MaterialDesignTheme.PasswordBox.xaml" />

<ResourceDictionary Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/Themes/MaterialDesignTheme.Light.xaml" />

<ResourceDictionary Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/Themes/MaterialDesignTheme.Defaults.xaml" />

<ResourceDictionary Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/Themes/MaterialDesignTheme.Button.xaml" />

<ResourceDictionary Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/Themes/MaterialDesignTheme.Menu.xaml" />

<ResourceDictionary Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/Themes/MaterialDesignTheme.ToolTip.xaml" />

<ResourceDictionary Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/Themes/MaterialDesignTheme.PopupBox.xaml" />

<ResourceDictionary Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/Themes/MaterialDesignTheme.TextBox.xaml" />

<ResourceDictionary Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/Themes/MaterialDesignTheme.TextBlock.xaml" />

<ResourceDictionary Source="pack://application:,,,/MaterialDesignColors;component/Themes/Recommended/Primary/MaterialDesignColor.Amber.xaml" />

<!--<ResourceDictionary Source="pack://application:,,,/MaterialDesignColors;component/Themes/Recommended/Accent/MaterialDesignColor.LightBlue.xaml" />-->

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</Application.Resources>

</Application>

<Window x:Class="WpfAssortmentCheck.MainWindow"

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

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

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

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

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns:local="clr-namespace:WpfAssortmentCheck"

mc:Ignorable="d"

Title="{Binding ElementName=MainFrame, Path=Content.Title}"

Height="450" Width="800" Icon="/WpfAssortmentCheck;component/Resources/icons8-пилатес-16.png"

Closing="WindowClosing" WindowStartupLocation="CenterScreen" Style="{StaticResource base\_window\_1}"

Closed="WindowClosed" >

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="75"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="30"/>

</Grid.RowDefinitions>

<Grid/>

<Rectangle Fill="{StaticResource PrimaryHueLightBrush}"/>

<Image Source="/WpfAssortmentCheck;component/Resources/icons8-пилатес-50.png"

Margin="10,10,0,0"

HorizontalAlignment="Left" Height="55" Width="62"/>

<TextBlock Grid.Row="0" Text="АИС «Фирма по установке и обслуживанию дверей»"

HorizontalAlignment="Center"

VerticalAlignment="Center"

FontSize="20"

FontWeight="Bold"/>

<Frame x:Name="MainFrame" Grid.Row="1"

NavigationUIVisibility="Hidden"

BorderThickness="1" ContentRendered="MainFrameContentRendered"/>

<StackPanel Grid.Row="0" Orientation="Horizontal" HorizontalAlignment="Right" Margin="0 0 20 0">

<Button x:Name="BtnEditGoods" Click="BtnEditGoodsClick" ToolTip="Тренеры" Visibility="Collapsed">

<materialDesign:PackIcon Kind="AccountMultiplePlus" Padding="0"/>

</Button>

<Button x:Name="BtnOrder" Click="BtnOrder\_Click" ToolTip="Заявки" Visibility="Collapsed">

<materialDesign:PackIcon Kind="OrderBoolAscending" Padding="0"/>

</Button>

<Button x:Name="BtnBuyes" Click="BtnBuyes\_Click" ToolTip="Услугаы" Visibility="Collapsed">

<materialDesign:PackIcon Kind="Cash" Padding="0"/>

</Button>

<Button x:Name="BtnMyAccount" Click="BtnMyAccount\_Click" ToolTip="Мой профиль" Visibility="Collapsed">

<materialDesign:PackIcon Kind="Account" Padding="0"/>

</Button>

<Button x:Name="BtnMyOrders" Click="BtnMyOrders\_Click" ToolTip="Оставить заявку" Visibility="Collapsed">

<materialDesign:PackIcon Kind="StarBorder" Padding="0"/>

</Button>

<Button x:Name="BtnEnter" Click="BtnEnter\_Click" ToolTip="Войти">

<materialDesign:PackIcon x:Name="IconBtnKey" Kind="Login" Padding="0"/>

</Button>

<Button x:Name="BtnBack"

Click="BtnBackClick" ToolTip="Назад" Visibility="Collapsed">

<materialDesign:PackIcon Kind="ArrowBack" Padding="0"/>

<!--Key-->

</Button>

</StackPanel>

<Rectangle Grid.Row="2" Fill="{StaticResource PrimaryHueLightBrush}"/>

</Grid>

</Window>

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using WpfAssortmentCheck.Models;

using WpfAssortmentCheck.Pages;

using WpfAssortmentCheck.Windows;

namespace WpfAssortmentCheck{

///<summary>

/// Логика взаимодействия для MainWindow.xaml

///</summary>

public partial class MainWindow : Window{

public MainWindow(){

InitializeComponent();

MainFrame.Navigate(new CatalogOfGoods());

Manager.CurrentUser = null;

Manager.MainFrame = MainFrame;}

private void WindowClosed(object sender, EventArgs e){

App.Current.Shutdown();}

//событие попытки закрытия окна,

// если пользователь выберет Cancel, то форму не закроем

private void WindowClosing(object sender, System.ComponentModel.CancelEventArgs e){

MessageBoxResult x = MessageBox.Show("Вы действительно хотите выйти?",

"Выйти", MessageBoxButton.OKCancel, MessageBoxImage.Weaponion);

if (x == MessageBoxResult.Cancel)

e.Cancel = true;}

// Кнопка назад

private void BtnBackClick(object sender, RoutedEventArgs e){

Manager.MainFrame.GoBack();}

// Кнопка навигации

private void BtnEditGoodsClick(object sender, RoutedEventArgs e){

MainFrame.Navigate(new GoodsPage());}

// Событие отрисовки страницы

// Скрываем или показываем кнопку Назад

// Скрываем или показываем кнопки Для перехода к остальным страницам

private void MainFrameContentRendered(object sender, EventArgs e){

if (MainFrame.CanGoBack){

BtnBack.Visibility = Visibility.Visible;

BtnEnter.Visibility = Visibility.Collapsed;

BtnOrder.Visibility = Visibility.Collapsed;

BtnBuyes.Visibility = Visibility.Collapsed;

if (Manager.CurrentUser is null)

return;

if (Manager.CurrentUser.Role == true)

BtnEditGoods.Visibility = Visibility.Collapsed;

else{

BtnMyAccount.Visibility = Visibility.Collapsed;

BtnMyOrders.Visibility = Visibility.Collapsed;}}

else{

BtnBack.Visibility = Visibility.Collapsed;

BtnEnter.Visibility = Visibility.Visible;

if (Manager.CurrentUser is null)

return;

BtnOrder.Visibility = Visibility.Visible;

BtnBuyes.Visibility = Visibility.Visible;

if (Manager.CurrentUser.Role == true)

BtnEditGoods.Visibility = Visibility.Visible;

else{

BtnMyAccount.Visibility = Visibility.Visible;

BtnMyOrders.Visibility = Visibility.Visible;}

}}

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

{}

private void BtnEditGroups\_Click(object sender, RoutedEventArgs e){

MainFrame.Navigate(new CategoryPage());}

private void BtnEnter\_Click(object sender, RoutedEventArgs e){

if (Manager.CurrentUser != null){

MessageBoxResult messageBoxResult = MessageBox.Show($"Выйти из системы??? ", "Выход", MessageBoxButton.OKCancel,

MessageBoxImage.Weaponion);

if (messageBoxResult == MessageBoxResult.OK){

IconBtnKey.Kind = MaterialDesignThemes.Wpf.PackIconKind.Login;

Manager.CurrentUser = null;

BtnEditGoods.Visibility = Visibility.Collapsed;

BtnOrder.Visibility = Visibility.Collapsed;

BtnBuyes.Visibility = Visibility.Collapsed;

BtnMyAccount.Visibility = Visibility.Collapsed;

BtnMyOrders.Visibility = Visibility.Collapsed;

return;}}

LoginWindow window = new LoginWindow();

if (window.ShowDialog() == true){

if (Manager.CurrentUser.Role == true){

BtnBack.Visibility = Visibility.Collapsed;

BtnEditGoods.Visibility = Visibility.Visible;

BtnOrder.Visibility = Visibility.Visible;

BtnBuyes.Visibility = Visibility.Visible;}

else {

BtnMyAccount.Visibility = Visibility.Visible;

BtnMyOrders.Visibility = Visibility.Visible;

BtnOrder.Visibility = Visibility.Visible;

BtnBuyes.Visibility = Visibility.Visible;}

}

}

private void BtnMyOrder\_Click(object sender, RoutedEventArgs e){

//MainFrame.Navigate(new StatusPage());

}

private void BtnMyAccount\_Click(object sender, RoutedEventArgs e){

try{

ClientWindow window = new ClientWindow();

if (window.ShowDialog() == true){

MessageBox.Show("Запись изменена", "Внимание", MessageBoxButton.OK, MessageBoxImage.Information);}}

catch{

MessageBox.Show("Ошибка");}}

private void BtnOrder\_Click(object sender, RoutedEventArgs e){

MainFrame.Navigate(new OrderPage());

}

private void BtnMyOrders\_Click(object sender, RoutedEventArgs e){

try{

MakeOrderWindow window = new MakeOrderWindow();

if (window.ShowDialog() == true){

MessageBox.Show("Запись изменена", "Внимание", MessageBoxButton.OK, MessageBoxImage.Information);}}

catch{

MessageBox.Show("Ошибка");}}

private void BtnBuyes\_Click(object sender, RoutedEventArgs e){

MainFrame.Navigate(new BuyPage());}

}}

<Window x:Class="WpfAssortmentCheck.Windows.LoginWindow"

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

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

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

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

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns:local="clr-namespace:WpfAssortmentCheck.Windows"

mc:Ignorable="d"

Title="Вход" Height="250"

Width="400" MinHeight="250" MinWidth="400"

WindowStartupLocation="CenterScreen" Style="{StaticResource base\_window\_1}"

ResizeMode="NoResize">

<!--NoResize - нужен, чтобы нельзя было изменить зааднные размеры окна-->

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="75"/>

<RowDefinition Height="40"/>

<RowDefinition Height="40"/>

<RowDefinition/>

<RowDefinition Height="20"/>

</Grid.RowDefinitions>

<Grid />

<materialDesign:PackIcon Kind="LoginVariant" Foreground="{StaticResource PrimaryHueMidBrush}"

Margin="10,10,0,0"

HorizontalAlignment="Left" Height="55" Width="62"/>

<TextBlock Grid.Row="0" Text="АИС"

HorizontalAlignment="Center"

VerticalAlignment="Center"

FontSize="16" TextWrapping="Wrap" TextAlignment="Center"

FontWeight="Bold" Height="60" Margin="65,10,48,5" Width="281" />

<TextBox x:Name="TbLogin" Grid.Row="1"

Grid.Column="1" materialDesign:HintAssist.Hint="Логин"

Width="300" Text="admin"/>

<PasswordBox x:Name="TbPass" Height="30" Width="300" materialDesign:HintAssist.Hint="Пароль" Password="1" Grid.Row="2" />

<StackPanel Orientation="Horizontal" Grid.Row="3" HorizontalAlignment="Center">

<Button Width="100" Height="30"

Content="OK"

x:Name="BtnOk"

Click="BtnOkClick"/>

<Button Width="100"

Height="30"

Content="Cancel"

x:Name="BtnCancel"

Click="BtnCancelClick"/>

</StackPanel>

<Button Grid.Row="4" Content="Регистрация" Padding="0" FontSize="10" Height="20" Click="Button\_Click"/>

<materialDesign:DialogHost x:Name="DialogHost"

BorderThickness="1">

<materialDesign:DialogHost.DialogContent>

<StackPanel>

<Grid VerticalAlignment="Top" Margin="35, 60" Height="180" Width="420">

<Label Content="Регистрация " HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0 0 0 20" Foreground="Gray" FontSize="18"/>

<StackPanel VerticalAlignment="Center" Margin="15">

<TextBox x:Name="tbRegLogin" Margin="0, 10" TextAlignment="Left" materialDesign:HintAssist.Hint="ЛОГИН" FontSize="18"/>

<PasswordBox x:Name="psbPassword1" Margin="0,10" materialDesign:HintAssist.Hint="ПАРОЛЬ" materialDesign:TextFieldAssist.HasClearButton="True" FontSize="18"/>

<PasswordBox x:Name="psbPassword2" Margin="0,10" materialDesign:HintAssist.Hint="ПАРОЛЬ" materialDesign:TextFieldAssist.HasClearButton="True" FontSize="18"/>

</StackPanel>

</Grid>

<StackPanel HorizontalAlignment="Center" Margin="10 20">

<StackPanel Orientation="Horizontal" HorizontalAlignment="Center">

<Button x:Name="btnOk" Click="btnOk\_Click">

<StackPanel Orientation="Horizontal" HorizontalAlignment="Center">

<materialDesign:PackIcon Kind="Check" Margin="5" VerticalAlignment="Center"/>

<TextBlock Text="Ok" VerticalAlignment="Center"/>

</StackPanel>

</Button>

<Button x:Name="btnCancel" Click="BtnCancel\_Click">

<StackPanel Orientation="Horizontal" HorizontalAlignment="Center">

<materialDesign:PackIcon Kind="Cancel" Margin="5" VerticalAlignment="Center"/>

<TextBlock Text="Cancel" VerticalAlignment="Center"/>

</StackPanel>

</Button>

</StackPanel>

</StackPanel>

</StackPanel>

</materialDesign:DialogHost.DialogContent>

</materialDesign:DialogHost>

</Grid>

</Window>

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 WpfAssortmentCheck.Models;

namespace WpfAssortmentCheck.Windows{

///<summary>

/// Логика взаимодействия для LoginWindow.xaml

///</summary>

public partial class LoginWindow : Window{

public LoginWindow(){

InitializeComponent();}

private void BtnOkClick(object sender, RoutedEventArgs e){

try

{ //загрузка всех пользователей из БД в список

List<User> users = YogaFeatPilatesBDEntities.GetContext().Users.ToList();

//попытка найти пользователя с указанным паролем и логином

//если такого пользователя не будет обнаружено то переменная u будет равна null

User u = users.FirstOrDefault(p => p.Password == TbPass.Password && p.UserName == TbLogin.Text);

if (u != null){

// логин и пароль корректные запускаем главную форму приложения

//MainWindow mainWindow = new MainWindow();

//mainWindow.Owner = this;

//this.Hide();

//mainWindow.Show();

Manager.CurrentUser = u;

this.DialogResult = true;}

else{

MessageBox.Show("Не верный логин или пароль");}}

catch (Exception ex){

MessageBox.Show(ex.Message.ToString());}}

//код кнопки Cancel

private void BtnCancelClick(object sender, RoutedEventArgs e){

this.DialogResult = false;}

private void Button\_Click(object sender, RoutedEventArgs e){

DialogHost.IsOpen = true;}

private void btnOk\_Click(object sender, RoutedEventArgs e){

if ((tbRegLogin.Text == "") || (psbPassword1.Password == "") || (psbPassword2.Password == "")){

MessageBox.Show("Поля пустые", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;}

List<User> users = YogaFeatPilatesBDEntities.GetContext().Users.ToList();

//попытка найти пользователя с указанным паролем и логином

//если такого пользователя не будет обнаружено то переменная u будет равна null

User u = users.FirstOrDefault(p => p.UserName == tbRegLogin.Text);

if (u != null){

MessageBox.Show("Данный логин занят, выберите другой логин", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;}

if (psbPassword1.Password != psbPassword2.Password){

MessageBox.Show("Пароли не совпадают", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;}

User user = new User();

user.UserName = tbRegLogin.Text;

user.Password = psbPassword1.Password;

user.Role = false;

user.FirstName = "";

user.LastName = "";

user.MiddleName = "";

user.Phone = "";

user.Email = "";

user.PassportNum = "";

user.PassportSeries = "";

YogaFeatPilatesBDEntities.GetContext().Users.Add(user);

YogaFeatPilatesBDEntities.GetContext().SaveChanges();

MessageBox.Show("Регистраця прошла успешно", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

DialogHost.IsOpen = false;}

private void BtnCancel\_Click(object sender, RoutedEventArgs e){

DialogHost.IsOpen = false;}

}}

<Page x:Class="WpfAssortmentCheck.Pages.CatalogOfGoods"

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

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

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

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

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns:local="clr-namespace:WpfAssortmentCheck.Pages"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="Фирма по установке и обслуживанию дверей ШАКИРА"

IsVisibleChanged="PageIsVisibleChanged" Style="{StaticResource base\_page}"

>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="30"/>

</Grid.RowDefinitions>

<WrapPanel Grid.Row="0"

HorizontalAlignment="Stretch"

>

<StackPanel Orientation="Horizontal">

<TextBlock Text="Выберите направление" FontWeight="Bold" FontSize="14" Padding="3"

/>

<ComboBox Width="275" x:Name="ComboCategory" FontSize="14" Padding="3"

SelectionChanged="ComboTypeSelectionChanged"

DisplayMemberPath="Name"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<TextBlock Text="Выберите Дверь" FontSize="14" Padding="3" FontWeight="Bold"/>

<ComboBox Width="275" x:Name="ComboTrainer" FontSize="14" Padding="3" SelectionChanged="ComboTrainer\_SelectionChanged">

<ComboBox.ItemTemplate>

<DataTemplate>

<StackPanel Orientation="Horizontal">

<Image Source="{Binding GetPhoto}" Width="15" Margin="5 0 5 0" />

<TextBlock Text="{Binding GetFio}" />

</StackPanel>

</DataTemplate>

</ComboBox.ItemTemplate>

</ComboBox>

</StackPanel>

<StackPanel Orientation="Horizontal" Margin="3">

<TextBlock Text="Сортировать по" VerticalAlignment="Center" FontWeight="Bold"

/>

<ComboBox Width="275"

x:Name="ComboSort"

SelectionChanged="ComboSortSelectionChanged" Margin="10 0 0 0" >

<ComboBoxItem >

<StackPanel Orientation="Horizontal">

<TextBlock Text="Направление"/>

<materialDesign:PackIcon Kind="SortAscending" Margin="5 0 0 0"/>

</StackPanel>

</ComboBoxItem>

<ComboBoxItem >

<StackPanel Orientation="Horizontal">

<TextBlock Text="Направление"/>

<materialDesign:PackIcon Kind="SortDescending" Margin="5 0 0 0"/>

</StackPanel>

</ComboBoxItem>

<ComboBoxItem >

<StackPanel Orientation="Horizontal">

<TextBlock Text="Тренер"/>

<materialDesign:PackIcon Kind="SortAlphabeticalAscending" Margin="5 0 0 0"/>

</StackPanel>

</ComboBoxItem>

<ComboBoxItem >

<StackPanel Orientation="Horizontal">

<TextBlock Text="Тренер"/>

<materialDesign:PackIcon Kind="SortAlphabeticalDescending" Margin="5 0 0 0"/>

</StackPanel>

</ComboBoxItem>

</ComboBox>

</StackPanel>

</WrapPanel>

<ListView x:Name="LViewGoods" Grid.Row="1"

ScrollViewer.HorizontalScrollBarVisibility="Disabled"

>

<ListView.ItemsPanel >

<ItemsPanelTemplate>

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

</ItemsPanelTemplate>

</ListView.ItemsPanel>

<ListView.ItemTemplate>

<DataTemplate>

<materialDesign:Card Width="300" Height="500" >

<Grid materialDesign:ColorZoneAssist.Mode="SecondaryMid">

<Grid.RowDefinitions>

<RowDefinition Height="40" />

<RowDefinition Height="200" />

<RowDefinition Height="30" />

<RowDefinition Height="30" />

<RowDefinition Height="200" />

</Grid.RowDefinitions>

<StackPanel Grid.Row="0" Margin="5" HorizontalAlignment="Center">

<TextBlock Text="{Binding Category.Name}" Foreground="{StaticResource akcent}" TextWrapping="Wrap" HorizontalAlignment="Center"

FontWeight="Bold" Margin="10"

FontSize="16"/>

</StackPanel>

<Image Source="{Binding Path=Trainer.GetPhoto}" Grid.Row="1" Height="200" Width="260" Margin="10"

Stretch="Uniform" />

<Button Style="{StaticResource MaterialDesignFloatingActionMiniAccentButton}"

HorizontalAlignment="Right" VerticalAlignment="Bottom" Margin="0 0 40 0"

Grid.Row="1"

ToolTip="Услуга занятий"

x:Name="BtnMoreInfo"

Padding="2 0 2 0"

Click="BtnMoreInfo\_Click">

<materialDesign:PackIcon Kind="DotsVertical" />

</Button>

<TextBlock Grid.Row="2" Margin="5" TextAlignment="Center"

Text="{Binding Trainer.GetFio}" TextWrapping="Wrap" HorizontalAlignment="Center"

FontWeight="Bold" FontSize="14" />

<TextBlock Grid.Row="3" Margin="5" TextAlignment="Center"

Text="Стоимость Услугаов" TextWrapping="Wrap" HorizontalAlignment="Center"

FontWeight="Bold" FontSize="12" Foreground="{StaticResource akcent}" />

<ScrollViewer HorizontalScrollBarVisibility="Disabled" VerticalScrollBarVisibility="Auto" Grid.Row="4">

<ListBox x:Name="ListBoxPrices" Height="200" ItemsSource="{Binding Abonements}" >

<ListBox.ItemTemplate>

<DataTemplate>

<StackPanel Margin="5" Orientation="Horizontal">

<TextBlock FontSize="14" Margin="20 0 0 0" Text="{Binding Path=LessonCount, StringFormat={}{0} занятий}" HorizontalAlignment="Left" Width="150"/>

<TextBlock FontSize="14" Text="{Binding Path=Price, StringFormat='c', ConverterCulture='ru-RU'}" HorizontalAlignment="Right" Width="120"/>

</StackPanel>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

</ScrollViewer>

</Grid>

</materialDesign:Card>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

<TextBlock Grid.Row="2" x:Name="TextBlockCount" />

<materialDesign:DialogHost x:Name="DialogHostMoreInformation"

Background="{DynamicResource PrimaryHueDarkBrush}" MinWidth="800"

MinHeight="400"

BorderThickness="1" HorizontalAlignment="Center" VerticalAlignment="Center" Grid.RowSpan="2" Grid.ColumnSpan="2"

>

<materialDesign:DialogHost.DialogContent >

<StackPanel Margin="20" >

<TextBlock x:Name="TbCategoryName" TextAlignment="Center" Foreground="Red"

TextWrapping="Wrap" HorizontalAlignment="Center" FontWeight="Bold" Margin="10"

FontSize="20"/>

<Image Source="{Binding Path=GetPhoto}" Height="240" Width="320" Stretch="Uniform" />

<Grid >

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<TextBlock FontWeight="Bold" Text="{Binding GetFio}" TextAlignment="Center" TextWrapping="Wrap" Grid.Row="0" Margin="3" FontSize="18" />

<ScrollViewer Grid.Row="1" Height="100" HorizontalAlignment="Stretch" Width="400">

<TextBlock TextWrapping="Wrap" VerticalAlignment="Center" Margin="5" TextAlignment="Center"

Text="{Binding Info}" FontSize="14" Grid.Row="1" />

</ScrollViewer>

<TextBlock Text="Услуга занятий:" TextWrapping="Wrap" Grid.Row="2" Grid.Column="0" Grid.ColumnSpan="2" FontWeight="Bold"/>

<ListBox Height="200" x:Name="ListBoxTimeTable" Grid.Row="3" >

<ListBox.ItemTemplate>

<DataTemplate>

<StackPanel Margin="5" Orientation="Horizontal">

<TextBlock FontSize="14" Text="{Binding Path=DayWeek}" HorizontalAlignment="Left" Width="120"/>

<TextBlock FontSize="14" Text="{Binding Path=DayTime, StringFormat='t', ConverterCulture='ru-RU'}" HorizontalAlignment="Right" />

</StackPanel>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

</Grid>

<Button x:Name="btnOk" Click="BtnOk\_Click" Width="100" HorizontalAlignment="Center">

<StackPanel Orientation="Horizontal" HorizontalAlignment="Center">

<materialDesign:PackIcon Kind="Check" Foreground="#FFF" Margin="5" VerticalAlignment="Center"/>

<TextBlock Text="OK" VerticalAlignment="Center"/>

</StackPanel>

</Button>

</StackPanel>

</materialDesign:DialogHost.DialogContent>

</materialDesign:DialogHost>

</Grid>

</Page>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Data.Entity;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using WpfAssortmentCheck.Models;

namespace WpfAssortmentCheck.Pages{

///<summary>

/// Логика взаимодействия для CatalogOfGoods.xaml

///</summary>

public partial class CatalogOfGoods : Page{

int \_itemcount = 0;

public CatalogOfGoods(){

InitializeComponent();

// загрузка данных в combobox + добавление дополнительной строки

var trainers = YogaFeatPilatesBDEntities.GetContext().Trainers.OrderBy(p => p.LastName).ToList();

trainers.Insert(0, new Trainer{

FirstName = "Все"}

);

ComboTrainer.ItemsSource = trainers;

ComboTrainer.SelectedIndex = 0;

var categories = YogaFeatPilatesBDEntities.GetContext().Categories.OrderBy(p => p.Name).ToList();

categories.Insert(0, new Category{

Name = "Все типы"}

);

ComboCategory.ItemsSource = categories;

ComboCategory.SelectedIndex = 0;

// загрузка данных в listview сортируем по названию

//LViewGoods.ItemsSource = YogaFeatPilatesBDEntities.GetContext().Abonements.OrderBy(p => p.CategoryTrainer.Trainer.LastName).ToList();

LViewGoods.ItemsSource = YogaFeatPilatesBDEntities.GetContext().CategoryTrainers.OrderBy(p => p.Trainer.LastName).ToList();

\_itemcount = LViewGoods.Items.Count;

// отображение количества записей

TextBlockCount.Text = $" Результат запроса:{\_itemcount} записей из{\_itemcount}";}

private void PageIsVisibleChanged(object sender, DependencyPropertyChangedEventArgs e){

//обновление данных после каждой активации окна

if (Visibility == Visibility.Visible){

ComboTrainer.ItemsSource = null;

var trainers = YogaFeatPilatesBDEntities.GetContext().Trainers.OrderBy(p => p.LastName).ToList();

trainers.Insert(0, new Trainer{

FirstName = "Все"}

);

ComboTrainer.ItemsSource = trainers;

ComboTrainer.SelectedIndex = 0;

ComboCategory.ItemsSource = null;

var categories = YogaFeatPilatesBDEntities.GetContext().Categories.OrderBy(p => p.Name).ToList();

categories.Insert(0, new Category{

Name = "Все типы"}

);

ComboCategory.ItemsSource = categories;

ComboCategory.SelectedIndex = 0;

YogaFeatPilatesBDEntities.GetContext().ChangeTracker.Entries().ToList().ForEach(p => p.Reload());

// LViewGoods.ItemsSource = YogaFeatPilatesBDEntities.GetContext().Abonements.OrderBy(p => p.CategoryTrainer.Trainer.LastName).ToList();

LViewGoods.ItemsSource = YogaFeatPilatesBDEntities.GetContext().CategoryTrainers.OrderBy(p => p.Trainer.LastName).ToList();}}

// Поиск товаров, которые содержат данную поисковую строку

private void TBoxSearchTextChanged(object sender, TextChangedEventArgs e){

UpdateData();}

// Поиск товаров конкретного производителя

private void ComboTypeSelectionChanged(object sender, SelectionChangedEventArgs e){

UpdateData();}

///<summary>

/// Метод для фильтрации и сортировки данных

///</summary>

private void UpdateData(){

// получаем текущие данные из бд

//var currentGoods = YogaFeatPilatesBDEntities.GetContext().Abonements.OrderBy(p => p.CategoryTrainer.Trainer.LastName).ToList();

var currentGoods = YogaFeatPilatesBDEntities.GetContext().CategoryTrainers.OrderBy(p => p.Trainer.LastName).ToList();

// выбор только тех товаров, которые принадлежат данному производителю

if (ComboTrainer.SelectedIndex > 0)

currentGoods = currentGoods.Where(p => p.TrainerId == (ComboTrainer.SelectedItem as Trainer).Id).ToList();

if (ComboCategory.SelectedIndex > 0)

currentGoods = currentGoods.Where(p => p.CategoryId == (ComboCategory.SelectedItem as Category).Id).ToList();

if (ComboSort.SelectedIndex >= 0){

// сортировка по возрастанию цены

if (ComboSort.SelectedIndex == 0)

currentGoods = currentGoods.OrderBy(p => p.Category.Name).ToList();

if (ComboSort.SelectedIndex == 1)

currentGoods = currentGoods.OrderByDescending(p => p.Category.Name).ToList();

// сортировка по убыванию цены

if (ComboSort.SelectedIndex == 2)

currentGoods = currentGoods.OrderBy(p => p.Trainer.LastName).ToList();

if (ComboSort.SelectedIndex == 3)

currentGoods = currentGoods.OrderByDescending(p => p.Trainer.LastName).ToList();}

// В качестве источника данных присваиваем список данных

LViewGoods.ItemsSource = currentGoods;

// отображение количества записей

TextBlockCount.Text = $" Результат запроса:{currentGoods.Count} записей из{\_itemcount}";}

// сортировка товаров

private void ComboSortSelectionChanged(object sender, SelectionChangedEventArgs e){

UpdateData();}

private void BtnShowMore\_Click(object sender, RoutedEventArgs e){

// Service selected = (sender as Button).DataContext as Service;

// MessageBox.Show(selected.PriceLists.Count.ToString());}

private void ComboTrainer\_SelectionChanged(object sender, SelectionChangedEventArgs e){

UpdateData();}

private void BtnMoreInfo\_Click(object sender, RoutedEventArgs e){

CategoryTrainer edu = (sender as Button).DataContext as CategoryTrainer;

Trainer trainer = YogaFeatPilatesBDEntities.GetContext().Trainers.FirstOrDefault(p => p.Id == edu.TrainerId);

List<TimeTable> timeTables = YogaFeatPilatesBDEntities.GetContext().TimeTables.Where(p => p.CategoryTrainerId == edu.Id).ToList();

ListBoxTimeTable.ItemsSource = timeTables;

TbCategoryName.Text = edu.Category.Name;

DialogHostMoreInformation.DataContext = trainer;

DialogHostMoreInformation.IsOpen = true;}

private void BtnOk\_Click(object sender, RoutedEventArgs e){

DialogHostMoreInformation.IsOpen = false;}

}}

<Page x:Class="WpfAssortmentCheck.Pages.GoodsPage"

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

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

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

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

xmlns:local="clr-namespace:WpfAssortmentCheck.Pages"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800" Style="{StaticResource base\_page}"

Title="Дверь" IsVisibleChanged="PageIsVisibleChanged" >

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="40"/>

</Grid.RowDefinitions>

<DataGrid x:Name="DataGridGood" Grid.Row="0"

AutoGenerateColumns="False"

IsReadOnly="True"

SelectionMode="Single"

LoadingRow="DataGridGoodLoadingRow">

<DataGrid.Columns>

<DataGridTemplateColumn Width="100" Header="Фото">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Image Source="{Binding Path=GetPhoto}" />

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTextColumn Binding="{Binding GetFio}" Header="ФИО" Width="\*"/>

<DataGridTextColumn Binding="{Binding Birthday, StringFormat='d', ConverterCulture='ru-RU'}" Header="Дата рождения" Width="2\*"/>

<DataGridTextColumn Binding="{Binding WorkExperience}" Header="Стаж, лет" Width="2\*"/>

<!--<DataGridTextColumn Binding="{Binding Price, StringFormat='c', ConverterCulture='ru-RU'}" Header="Стоимость, руб." Width="\*"/>-->

<DataGridTemplateColumn Width="auto" Header="Изменить">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button x:Name="BtnEdit" Click="ButtonClick" Margin="0">

<materialDesign:PackIcon Kind="Edit" />

</Button>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Width="auto" Header="Услугаы">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button x:Name="BtnPrice" Click="BtnPrice\_Click" Margin="0">

<materialDesign:PackIcon Kind="CurrencyRub" />

</Button>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Width="auto" Header="Услуга">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button x:Name="BtnTimeTable" Click="BtnTimeTable\_Click" Margin="0">

<materialDesign:PackIcon Kind="Timelapse" />

</Button>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

<StackPanel Orientation="Horizontal" Grid.Row="1" HorizontalAlignment="Center">

<Button x:Name="BtnCategories"

Click="BtnCategories\_Click" ToolTip="Виды услуг">

<materialDesign:PackIcon Kind="FormatListBulletedType" Padding="0"/>

</Button>

<Button x:Name="BtnAdd" Grid.Row="1" Width="100"

Content="Добавить" Click="BtnAddClick" />

<Button x:Name="BtnDelete" Grid.Row="1" Width="100" Margin="3"

Content="Удалить" Click="BtnDeleteClick"/>

</StackPanel>

</Grid>

</Page>

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.OleDb;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using WpfAssortmentCheck.Models;

using WpfAssortmentCheck.Pages;

using Excel = Microsoft.Office.Interop.Excel;

namespace WpfAssortmentCheck.Pages{

///<summary>

/// Логика взаимодействия для GoodsPage.xaml

///</summary>

public partial class GoodsPage : Page{

List<Trainer> trainers;

public GoodsPage(){

InitializeComponent();}

private void ButtonClick(object sender, RoutedEventArgs e){

// открытие редактирования товара

// передача выбранного товара в AddGoodPage

Manager.MainFrame.Navigate(new AddGoodPage((sender as Button).DataContext as Trainer));}

private void PageIsVisibleChanged(object sender, DependencyPropertyChangedEventArgs e){

//событие отображения данного Page

// обновляем данные каждый раз когда активируется этот Page

if (Visibility == Visibility.Visible){

DataGridGood.ItemsSource = null;

//загрузка обновленных данных

YogaFeatPilatesBDEntities.GetContext().ChangeTracker.Entries().ToList().ForEach(p => p.Reload());

trainers = YogaFeatPilatesBDEntities.GetContext().Trainers.OrderBy(p => p.LastName).ToList();

DataGridGood.ItemsSource = trainers;}}

private void BtnAddClick(object sender, RoutedEventArgs e){

// открытие AddGoodPage для добавления новой записи

Manager.MainFrame.Navigate(new AddGoodPage(null));}

private void BtnDeleteClick(object sender, RoutedEventArgs e){

// удаление выбранного товара из таблицы

//получаем все выделенные товары

var selectedGoods = DataGridGood.SelectedItems.Cast<Trainer>().ToList();

// вывод сообщения с вопросом Удалить запись?

MessageBoxResult messageBoxResult = MessageBox.Show($"Удалить{selectedGoods.Count()} записей???",

"Удаление", MessageBoxButton.OKCancel, MessageBoxImage.Weaponion);

//если пользователь нажал ОК пытаемся удалить запись

if (messageBoxResult == MessageBoxResult.OK){

try{

// берем из списка удаляемых товаров один элемент

Trainer x = selectedGoods[0];

// проверка, есть ли у товара в таблице о продажах связанные записи

// если да, то выбрасывается исключение и удаление прерывается

if ((x.CategoryTrainers.Count > 0))

throw new Exception("Ошибка удаления, есть связанные записи");

// удаляем товара

YogaFeatPilatesBDEntities.GetContext().Trainers.Remove(x);

//сохраняем изменения

YogaFeatPilatesBDEntities.GetContext().SaveChanges();

MessageBox.Show("Записи удалены");

trainers.Clear();

trainers = YogaFeatPilatesBDEntities.GetContext().Trainers.OrderBy(p => p.LastName).ToList();

DataGridGood.ItemsSource = null;

DataGridGood.ItemsSource = trainers;}

catch (Exception ex){

MessageBox.Show(ex.Message.ToString(), "Ошибка удаления", MessageBoxButton.OK, MessageBoxImage.Error);}

}}

private void BtnSellClick(object sender, RoutedEventArgs e){

// открытие страницы о продажах SellGoodsPage

// передача в него выбранного товара

//Manager.MainFrame.Navigate(new SellGoodsPage((sender as Button).DataContext as Trainer));}

// отображение номеров строк в DataGrid

private void DataGridGoodLoadingRow(object sender, DataGridRowEventArgs e){

e.Row.Header = (e.Row.GetIndex() + 1).ToString();}

private void BtnCategories\_Click(object sender, RoutedEventArgs e){

Manager.MainFrame.Navigate(new CategoryPage());}

private void BtnPrice\_Click(object sender, RoutedEventArgs e){

Manager.MainFrame.Navigate(new AbonementPage((sender as Button).DataContext as Trainer));}

private void BtnTimeTable\_Click(object sender, RoutedEventArgs e){

Manager.MainFrame.Navigate(new TimeTablePage((sender as Button).DataContext as Trainer));}

}}

<Page x:Class="WpfAssortmentCheck.Pages.AddGoodPage"

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

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

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

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

xmlns:xctk="http://schemas.xceed.com/wpf/xaml/toolkit"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns:local="clr-namespace:WpfAssortmentCheck.Pages" Style="{StaticResource base\_page}"

mc:Ignorable="d"

d:DesignHeight="680" d:DesignWidth="800"

Title="Добавление и редактирование"

>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="30"/>

<RowDefinition Height="30"/>

<RowDefinition Height="30"/>

<RowDefinition Height="100"/>

<RowDefinition Height="30"/>

<RowDefinition Height="30"/>

<RowDefinition Height="30"/>

<RowDefinition/>

<RowDefinition Height="50"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="160"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Rectangle Grid.Row="4" Grid.ColumnSpan="2" />

<TextBlock Grid.Row="6" Grid.Column="0" Text="Направления" Margin="20 0"/>

<xctk:CheckComboBox x:Name="ComboCategories" Grid.Row="6" Grid.Column="1"

DisplayMemberPath="Name" SelectedMemberPath="Selected" TabIndex="34" MaxDropDownHeight="150"/>

<TextBlock Grid.Row="4" Grid.Column="0" Text="Дата рождения:" Margin="20 0"/>

<DatePicker x:Name="DatePickerBirthday" Grid.Row="4" Grid.Column="1" SelectedDate="{Binding Birthday}" Margin="0,2" />

<TextBox x:Name="TextBoxGoodId" Text="{Binding GoodId}" Grid.Row="4" Grid.Column="1" Padding="0,2" IsReadOnly="True" Visibility="Collapsed"/>

<TextBlock Text="Стаж работы, лет." Grid.Row="5" Grid.Column="0" VerticalAlignment="Center" Margin="20,7,20,6" />

<xctk:IntegerUpDown x:Name="UpDownWorkExperience"

Width="100"

HorizontalAlignment="Left" Increment="1"

Value="{Binding WorkExperience}" Grid.Row="5" Minimum="0"

Grid.Column="1" Padding="0,2" />

<Image x:Name="ImagePhoto" Source="{Binding GetPhoto}" Grid.Row="7" Grid.Column="1"

VerticalAlignment="Stretch" HorizontalAlignment="Stretch"/>

<Button x:Name="BtnLoad" Grid.Row="7" Grid.Column="0" Margin="20,30,0,0"

HorizontalAlignment="Left" VerticalAlignment="Top"

Click="BtnLoadClick" Content="Загрузить"/>

<Button x:Name="BtnSave" Grid.Row="8" Content="Сохранить" HorizontalAlignment="Center" Margin="140,5,340,0" VerticalAlignment="Top" Width="120"

Click="BtnSaveClick" Grid.Column="1"/>

<TextBlock Grid.Column="0" Text="Фамилия:" Margin="20 0"/>

<TextBox Text="{Binding LastName}" materialDesign:HintAssist.Hint="Фамилия" Padding="20 5 0 0" Grid.Column="1" VerticalAlignment="Center" Grid.Row="0" Height="30" Margin="0,7" />

<TextBox Text="{Binding FirstName}" materialDesign:HintAssist.Hint="Имя" Padding="20 5 0 0" Grid.Column="1" VerticalAlignment="Center" Grid.Row="1" Height="30" Margin="0,7" />

<TextBox Text="{Binding MiddleName}" materialDesign:HintAssist.Hint="Отчество" Padding="20 5 0 0" Grid.Column="1" VerticalAlignment="Center" Grid.Row="2" Height="30" Margin="0,7" />

<TextBlock Grid.Column="0" Text="Имя:" Grid.Row="1" Margin="20 0"/>

<TextBlock Grid.Column="0" Text="Отчество:" Grid.Row="2" Margin="20 0"/>

<TextBlock Grid.Column="0" Text="Информация:" Grid.Row="3" Margin="20,0"/>

<TextBox Text="{Binding Info}" materialDesign:HintAssist.Hint="Информация"

Padding="20 5 0 0" Grid.Column="1"

VerticalAlignment="Stretch" Grid.Row="3" TextWrapping="Wrap" Margin="0,7" ScrollViewer.HorizontalScrollBarVisibility="Disabled"

ScrollViewer.VerticalScrollBarVisibility="Auto"/>

</Grid>

</Page>

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using WpfAssortmentCheck.Models;

namespace WpfAssortmentCheck.Pages{

///<summary>

/// Логика взаимодействия для AddGoodPage.xaml

///</summary>

public partial class AddGoodPage : Page

{ //текущий товар

private Trainer \_currentItem = new Trainer();

// путь к файлу

private string \_filePath = null;

// название текущей главной фотографии

private string \_photoName = null;

// текущая папка приложения

private static string \_currentDirectory = Directory.GetCurrentDirectory() + @"\Images\";

// передача в AddGoodPage товара

List<CategoryTrainer> currentItems = new List<CategoryTrainer>();

class ComboBoxTypeItem{

public int Id{ get; set;}

public int CategoryTrainerId{ get; set;}

public string Name{ get; set;}

public bool Selected{ get; set;}}

List<ComboBoxTypeItem> servicesItems = new List<ComboBoxTypeItem>();

private string GetComboBoxItemContent(List<ComboBoxTypeItem> items){

List<string> s = new List<string>();

foreach (ComboBoxTypeItem item in items){

if (item.Selected == true){

s.Add(item.Name);}}

return String.Join(", ", s);}

// загрузка

private void LoadServices(){

servicesItems.Clear();

List<CategoryTrainer> xlist = YogaFeatPilatesBDEntities.GetContext().CategoryTrainers.Where(i => i.TrainerId == \_currentItem.Id).ToList();

List<Category> items = YogaFeatPilatesBDEntities.GetContext().Categories.ToList();

List<int> datas = new List<int>();

foreach (CategoryTrainer c in xlist){

datas.Add(c.CategoryId);

currentItems.Add(c);}

// MessageBox.Show(currentItems.Count.ToString());

foreach (Category item in items){

if (datas.Contains(item.Id)){

servicesItems.Add(new ComboBoxTypeItem

{

Id = item.Id,

Name = item.Name,

Selected = true

});

}

else

servicesItems.Add(new ComboBoxTypeItem

{

Id = item.Id,

Name = item.Name,

Selected = false

});}

ComboCategories.ItemsSource = null;

ComboCategories.ItemsSource = servicesItems;}

//сохраytybt

void SaveServices(){

List<CategoryTrainer> saveItems = new List<CategoryTrainer>();

foreach (ComboBoxTypeItem item in servicesItems){

if (item.Selected == true){

saveItems.Add(new CategoryTrainer

{

TrainerId = \_currentItem.Id,

CategoryId = item.Id,

}); ;}}

// MessageBox.Show(currentItems.Count.ToString());

// MessageBox.Show(saveItems.Count.ToString());

List<CategoryTrainer> delItems = new List<CategoryTrainer>();

List<CategoryTrainer> addItems = new List<CategoryTrainer>();

foreach (CategoryTrainer x in currentItems){

bool b = false;

foreach (CategoryTrainer y in saveItems){

if ((y.CategoryId == x.CategoryId) && (y.TrainerId == x.TrainerId))

{

b = true;

break;

}

}

if (!b)

delItems.Add(x);}

foreach (CategoryTrainer x in saveItems){

bool b = false;

foreach (CategoryTrainer y in currentItems){

if ((y.CategoryId == x.CategoryId) && (y.TrainerId == x.TrainerId))

{

b = true;

break;

}

}

if (!b)

addItems.Add(x);}

string s = "";

foreach (CategoryTrainer x in delItems){

if ((x.Abonements.Count == 0) && (x.TimeTables.Count == 0))

YogaFeatPilatesBDEntities.GetContext().CategoryTrainers.Remove(x);

else

s += "\n" + x.Category.Name;}

if (s != "")

MessageBox.Show($"Не удалось удалить у данного Дверь следующие " +

$"направления:{s}, поскольку у данного Дверь есть Услугаы" +

$" и занятия по данным направлениям", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

YogaFeatPilatesBDEntities.GetContext().CategoryTrainers.AddRange(addItems);

YogaFeatPilatesBDEntities.GetContext().SaveChanges();}

public AddGoodPage(Trainer selectedItem){

InitializeComponent();

// если передано null, то мы добавляем новый товар

if (selectedItem != null){

\_currentItem = selectedItem;

TextBoxGoodId.Visibility = Visibility.Hidden;

int x = selectedItem.Id;

//\_filePath = \_currentDirectory + \_currentItem.Photo;}

else{

\_currentItem.Birthday = DateTime.Today;}

DataContext = \_currentItem;

LoadServices();

\_photoName = \_currentItem.Photo;}

// проверка полей

private StringBuilder CheckFields(){

StringBuilder s = new StringBuilder();

if (string.IsNullOrWhiteSpace(\_currentItem.LastName))

s.AppendLine("Поле фамилия пустое");

if (string.IsNullOrWhiteSpace(\_currentItem.FirstName))

s.AppendLine("Поле имя пустое");

if (string.IsNullOrWhiteSpace(\_currentItem.MiddleName))

s.AppendLine("Поле отчество пустое");

if (\_currentItem.Birthday == null)

s.AppendLine("Выберите дату рождения");

if (\_currentItem.WorkExperience == 0)

s.AppendLine("Укадите стаж");

if (string.IsNullOrWhiteSpace(\_currentItem.Info))

s.AppendLine("Заполните информацию о тренере");

if (string.IsNullOrWhiteSpace(\_photoName))

s.AppendLine("фото не выбрано пустое");

return s;}

// сохранение

private void BtnSaveClick(object sender, RoutedEventArgs e){

StringBuilder \_error = CheckFields();

// если ошибки есть, то выводим ошибки в MessageBox

// и прерываем выполнение

if (\_error.Length > 0){

MessageBox.Show(\_error.ToString());

return;}

// проверка полей прошла успешно

if (\_currentItem.Id == 0){

// добавление нового товара

// формируем новое название файла картинки,

// так как в папке может быть файл с тем же именем

string photo = ChangePhotoName();

// путь куда нужно скопировать файл

string dest = \_currentDirectory + photo;

File.Copy(\_filePath, dest);

\_currentItem.Photo = photo;

// добавляем товар в БД

YogaFeatPilatesBDEntities.GetContext().Trainers.Add(\_currentItem);}

try{

if (\_filePath != null){

string photo = ChangePhotoName();

string dest = \_currentDirectory + photo;

File.Copy(\_filePath, dest);

\_currentItem.Photo = photo;}

// Сохраняем изменения в БД

YogaFeatPilatesBDEntities.GetContext().SaveChanges();

SaveServices();

MessageBox.Show("Запись Изменена");

// Возвращаемся на предыдущую форму

Manager.MainFrame.GoBack();}

catch (Exception ex){

MessageBox.Show(ex.Message.ToString());}}

// загрузка фото

private void BtnLoadClick(object sender, RoutedEventArgs e){

try{

//Диалог открытия файла

OpenFileDialog op = new OpenFileDialog();

op.Title = "Select a picture";

op.Filter = "JPEG Files (\*.jpeg)|\*.jpeg|PNG Files (\*.png)|\*.png|JPG Files (\*.jpg)|\*.jpg|GIF Files (\*.gif)|\*.gif";

// диалог вернет true, если файл был открыт

if (op.ShowDialog() == true){

// проверка размера файла

// по условию файл дожен быть не более 2Мб.

FileInfo fileInfo = new FileInfo(op.FileName);

if (fileInfo.Length > (1024 \* 1024 \* 2))

{

// размер файла меньше 2Мб. Поэтому выбрасывается новое исключение

throw new Exception("Размер файла должен быть меньше 2Мб");

}

ImagePhoto.Source = new BitmapImage(new Uri(op.FileName));

\_photoName = op.SafeFileName;

\_filePath = op.FileName;}}

catch (Exception ex){

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

\_filePath = null;}}

//подбор имени файла

string ChangePhotoName(){

string x = \_currentDirectory + \_photoName;

string photoname = \_photoName;

int i = 0;

if (File.Exists(x)){

while (File.Exists(x)){

i++;

x = \_currentDirectory + i.ToString() + photoname;}

photoname = i.ToString() + photoname;}

return photoname;}

}}

<Page x:Class="WpfAssortmentCheck.Pages.BuyPage"

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

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

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

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

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns:local="clr-namespace:WpfAssortmentCheck.Pages"

mc:Ignorable="d" Style="{StaticResource base\_page}"

d:DesignHeight="450" d:DesignWidth="800"

Title="Покупка Услугаов" IsVisibleChanged="PageIsVisibleChanged" >

<Page.Resources>

<Style TargetType="DataGridRow" x:Key="grid\_item">

<Setter Property="Background" Value="{Binding Color}" />

<Setter Property="BorderThickness" Value="1"/>

</Style>

</Page.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="40"/>

</Grid.RowDefinitions>

<DataGrid x:Name="DataGridGood" Grid.Row="0"

AutoGenerateColumns="False"

IsReadOnly="True"

SelectionMode="Single"

ItemContainerStyle="{StaticResource grid\_item}"

LoadingRow="DataGridGoodLoadingRow"

>

<DataGrid.RowStyle>

<Style TargetType="DataGridRow">

<Setter Property="Background" Value="{Binding Status.Color}"/>

</Style>

</DataGrid.RowStyle>

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding Id}" Header="Номер Услугаа" Width="\*"/>

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

<DataGridTextColumn Binding="{Binding User.GetFio}" Header="Клиент" Width="2\*"/>

<DataGridTextColumn Binding="{Binding Abonement.CategoryTrainer.Category.Name}" Header="Направление" Width="2\*"/>

<DataGridTextColumn Binding="{Binding Abonement.CategoryTrainer.Trainer.GetFio}" Header="Тренер" Width="\*"/>

<DataGridTextColumn Binding="{Binding LessonsLeftCount}" Header="Оставшееся количество занятий" Width="\*"/>

<DataGridTextColumn Binding="{Binding DateTime, StringFormat='d', ConverterCulture=ru-RU}" Header="Дата покупки Услугаа" Width="\*"/>

<DataGridTextColumn Binding="{Binding Abonement.Price}" Header="Стоимость" Width="\*"/>

<DataGridTemplateColumn Width="auto" Header="Подробнее" x:Name="EditColumn">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button x:Name="BtnEdit" Click="BtnEdit\_Click" Margin="0">

<materialDesign:PackIcon Kind="DotsSquare"/>

</Button>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Width="auto" Header="Посещение" >

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button x:Name="BtnLook" Click="BtnLook\_Click" Content="{materialDesign:PackIcon Kind=Eye}" Margin="0">

</Button>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

<StackPanel Orientation="Horizontal" Grid.Row="1" HorizontalAlignment="Center">

<Button x:Name="BtnAdd" Grid.Row="1" Width="100"

Content="Добавить" Click="BtnAddClick" />

<Button x:Name="BtnDelete" Grid.Row="1" Width="100" Margin="3"

Content="Удалить" Click="BtnDeleteClick"/>

<Button x:Name="BtnStatus" Grid.Row="1" Width="100" Margin="3"

Content="Категории" Click="BtnStatus\_Click"/>

<StackPanel Orientation="Horizontal">

<ComboBox x:Name="cmbSearchType" Background="White" VerticalAlignment="Stretch" VerticalContentAlignment="Center" Width="200" SelectedIndex="0" Padding="10,0,0,0" SelectionChanged="cmbSearchType\_SelectionChanged">

<ComboBoxItem Content="Поиск по договору"/>

<ComboBoxItem Content="Поиск по клиенту"/>

</ComboBox>

<TextBox x:Name="tbSearchID" Width=" 100" BorderThickness="1" Background="White"/>

<Button x:Name="btnSearch" Width="60" Click="BtnSearch\_Click" Background="White" BorderBrush="Black" >

<materialDesign:PackIcon Kind="Search" Foreground="#FF0000" VerticalAlignment="Center"/>

</Button>

<Button x:Name="btnClearSearch" Width="60" Click="btnClearSearch\_Click" Background="White" BorderBrush="Black" >

<materialDesign:PackIcon Kind="Cancel" Foreground="#FF0000" VerticalAlignment="Center"/>

</Button>

<Button x:Name="btnExcel" Width="60" Background="White" BorderBrush="Black" Click="BtnExcel\_Click" >

<materialDesign:PackIcon Kind="FileExcel" Foreground="#FF087008" VerticalAlignment="Center"/>

</Button>

</StackPanel>

</StackPanel>

</Grid>

</Page>

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.OleDb;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using WpfAssortmentCheck.Models;

using WpfAssortmentCheck.Pages;

using WpfAssortmentCheck.Windows;

using Excel = Microsoft.Office.Interop.Excel;

namespace WpfAssortmentCheck.Pages{

///<summary>

/// Логика взаимодействия для BuyPage.xaml

///</summary>

public partial class BuyPage : Page{

List<Buy> buys;

User \_currentUser;

ICollectionView collectionView;

public BuyPage(){

InitializeComponent();

\_currentUser = Manager.CurrentUser;

LoadData();}

private void ButtonClick(object sender, RoutedEventArgs e){

// открытие редактирования товара

// передача выбранного товара в AddGoodPage

// Manager.MainFrame.Navigate(new AddNewOrderPage((sender as Button).DataContext as Order));}

void LoadData(){

if (\_currentUser.Role == true){

DataGridGood.ItemsSource = null;

//загрузка обновленных данных

FilterByClient(tbSearchID.Text);

break;

}}

**Приложение Б**

Фирма по установке и обслуживанию дверей

Спецификация

460.ЗМК.00010-01 01

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Обозначение** | | |  | | --- | | **Наименование** | | |  | | --- | | **Примечание** | |
|  | |  | | --- | | Документация | |  |
| 460.ЗМК.00010-01 12 01 | Фирма по установке и обслуживанию дверей |  |
|  | Текст программы |  |
| 460.ЗМК.00010-01 13 01 | Фирма по установке и обслуживанию дверей |  |
|  | Описание программы |  |
| 460.ЗМК.00010-01 34 01 | Фирма по установке и обслуживанию дверей |  |
|  | Руководство пользователя |  |
| 460.ЗМК.00010-01 81 01 | Фирма по установке и обслуживанию дверей |  |
|  | Пояснительная записка |  |

**Приложение B**

**Диск с программным средством**