**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 21](#_Toc72446670)

[3.7 Описание структуры выходной информации 33](#_Toc72446671)

[4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 34](#_Toc72446672)

[4.1 Назначение программного средства 34](#_Toc72446673)

[4.2 Условия выполнения программного средства 34](#_Toc72446674)

[4.3 Эксплуатация программного средства 34](#_Toc72446675)

[4.4 Сообщения пользователю 46](#_Toc72446676)

[5 РАЗРАБОТКА ТЕСТОВОГО НАБОРА 47](#_Toc72446677)

[5.1 Определение верхней границы количества тестов 47](#_Toc72446678)

[5.2 Разработка тестовых вариантов 48](#_Toc72446679)

[5.3 Составление отчета о тестировании 50](#_Toc72446680)

[ЗАКЛЮЧЕНИЕ 56](#_Toc72446681)

[Оценка качества программного средства с помощью метрик 56](#_Toc72446682)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 59](#_Toc72446683)

[Приложение А Текст кода программного средства 61](#_Toc72446684)

[Приложение Б Спецификация 84](#_Toc72446685)

[Приложение B Диск с программным средством 86](#_Toc72446686)

**Аннотация**

На дипломную работу студента Карасева Н.В. на тему: Автоматизированная инфомационная система «Студия автозвука 12 Вольт». Дипломная работа представлена пояснительной запиской на 86 листов машинописного текста. Состоит из введения, 5 глав, заключения и списка литературных источников.

В главе 1 содержится общие сведения о программном средстве «АИС Студия автозвука 12 Вольт». 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 – «АИС Студия автозвука 12 Вольт», 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 инфомационной системы студии автозвука 12 Вольт, который позволяет автоматизировать деятельность студии автозвука 12 Вольт.

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. 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**

«Автоматизированная инфомационная система «Студия автозвука 12 Вольт».

**1.3 Условное обозначение программного средства**

«AvtoSound 1.0»

**1.4 Разработчики программного средства**

Программное средство разработал студент Зеленодольского Механического колледжа группы 207 Карасев К.В.

**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 «AvtoSound 1.0» работает под управление операционной системы Windows 7, Windows 8, Windows 10. Программа предназначена для автоматизации деятельности фирмы. Цель создания программы – упрощение учёта заявок. Программа позволяет:

* Создавать и редактировать базу услуг салона
* Формировать заявки
* Выгружать информацию об оказанных услугах в Excel

**2.3 Требование к программному средству**

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

«AvtoSound 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.

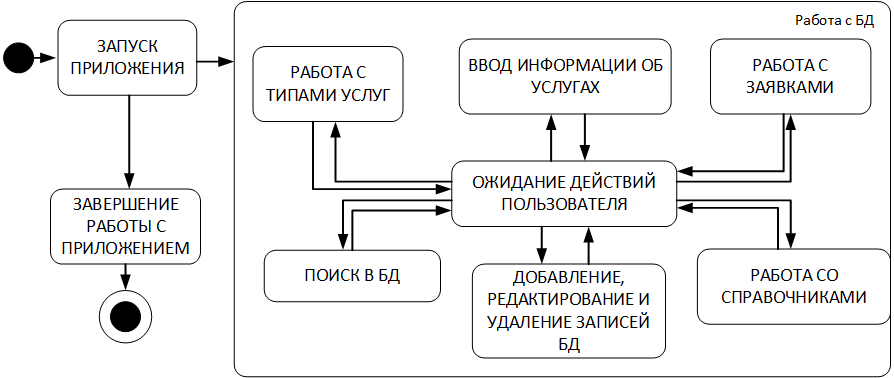


Рис.1 Декомпозиция поставленной задачи

**3.2 Общая архитектура программного средства**

Объектом моделирования является Автоматизированная инфомационная система «Студия автозвука 12 Вольт».

Рассматриваемые процессы: учёт заявок.

Объекты моделирования представлены на диаграмме классов, рис.2.

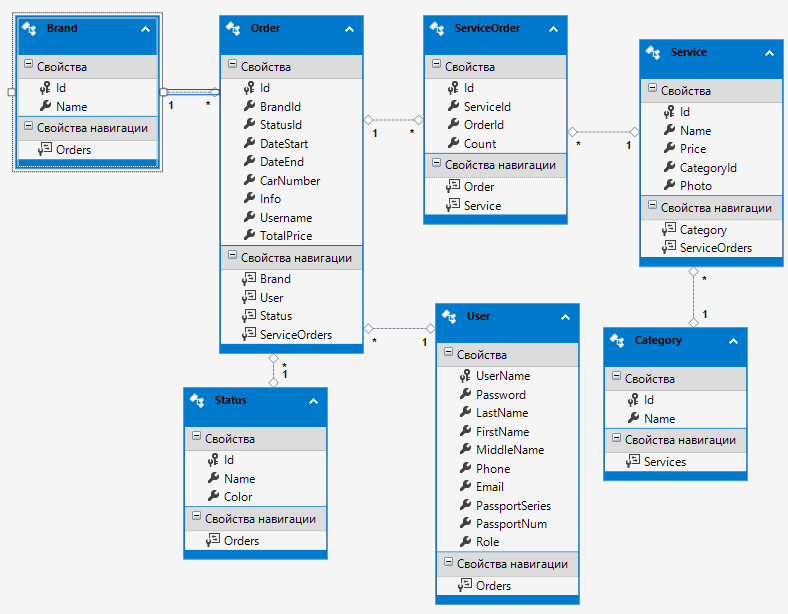


Рис.2 Диаграмма классов

По полученной декомпозиции задач была спроектирована следующая архитектура программного средства.

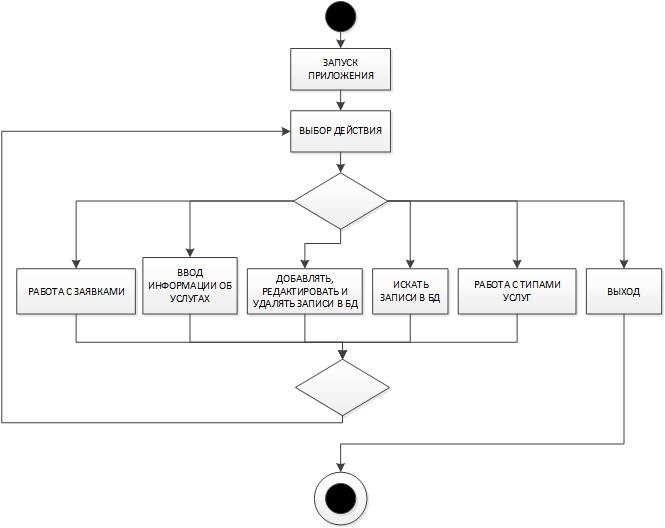


Рис.3 Схема событийно-ориентированной архитектуры

В «AvtoSound 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 Word 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.

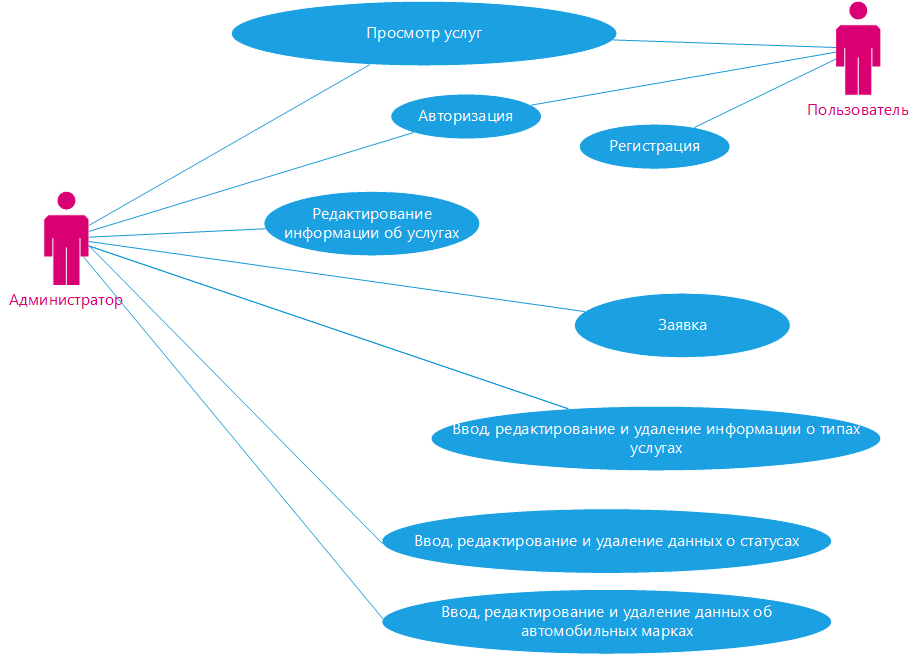


Рис.4 Диаграмма вариантов использования

**3.3 Разработка алгоритма решения задачи**

**Общий алгоритм**

****Рис.5 Алгоритм программы

**3.4 Реализация функционального назначения программного средства**

Программа «AvtoSound 1.01» имеет следующий набор входных данных, такие как: услуги, заявка, категории и бренды автомобилей.

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

Выходными данными являются: учёт заявок и квитанция.

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 «БД Автоматизированная инфомационная система «Студия автозвука 12 Вольт»» представленная 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 «Заявка». Реляционная модель автоматизированной системы соответствует всем 12 правилам Кодда. (рис.6)

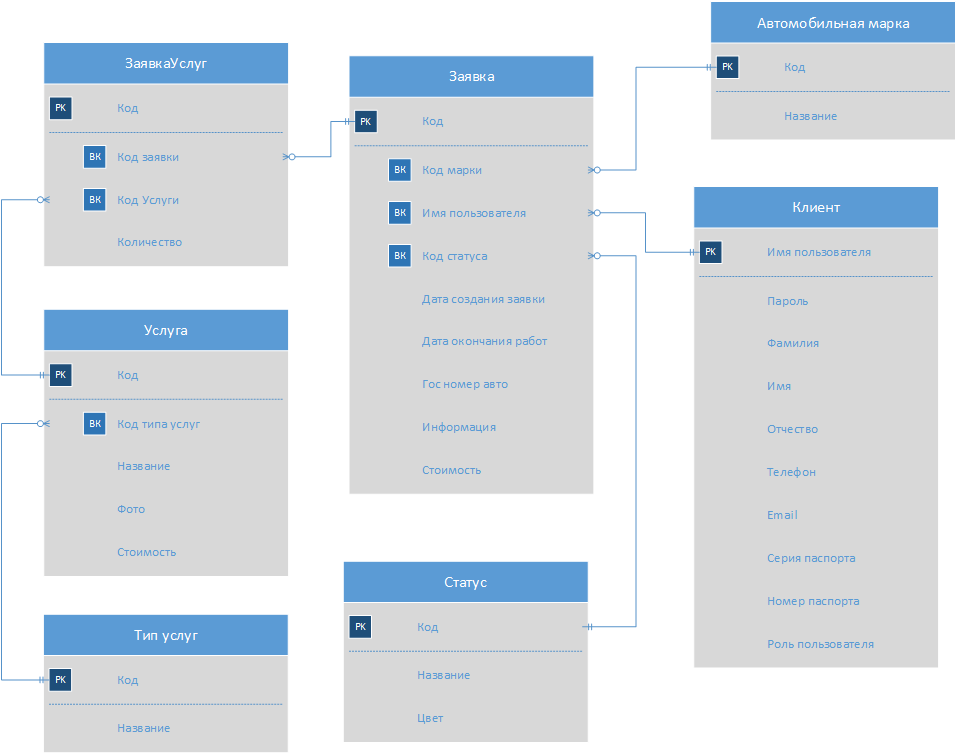


Рис.6 Реляционная модель

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 Тип услуг), Код (Таблица Услуга), Код (Таблица Австомобильная марка), Код (Таблица ЗаявкаУслуг), Код (Таблица Статус), Код (Таблица Заявка), Код (Таблица Климат система), Имя пользователя (Таблица Пользователь) и Имя пользователя(Таблица Клиент). 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 “Order” – “Id”. Связь таблиц “Category” и “Service” по полю “CategoryId” один ко многим (1:М). Связь таблиц “Status” и “Order” по полю “StatusId” один ко многим (1:М). Связь таблиц “Brand” и “Order” по полю “BrandId” один ко многим (1:М). Связь таблиц “Service” и “ServiceOrder” по полю “ServiceId” один ко многим (1:М). Связь таблиц “Order” и “ServiceOrder” по полю “OrderId” 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.

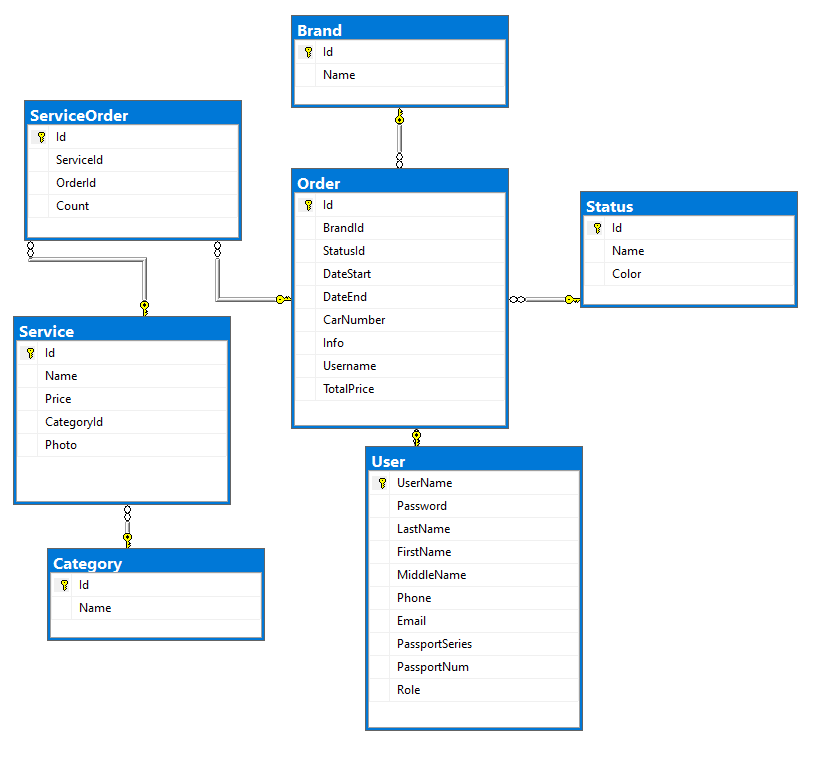


Рис.7 Схема базы данных

Таблица “Category”(Категории)

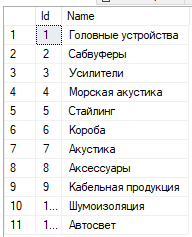


Рис.8 Таблица Category

Таблица “Brand”(Автомобильная марка)

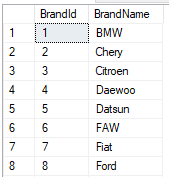


Рис.9 Таблица Brand

Таблица “Service”(Услуги)

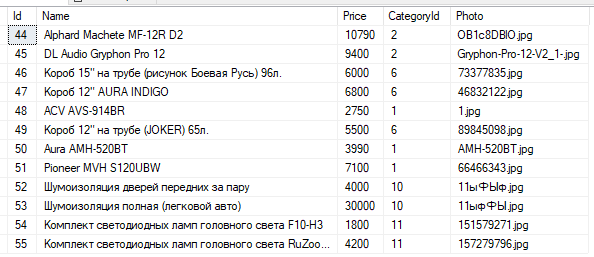


Рис.10 Таблица Service

Таблица “User”(Пользователи)

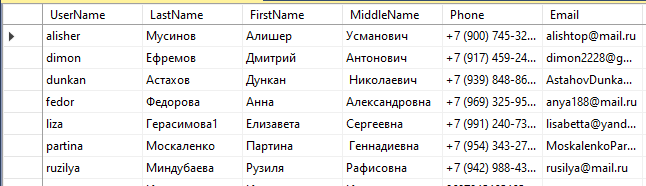


Рис.11 Таблица User

Таблица “Status”(Статус)

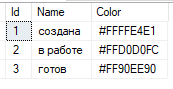


Рис.12 Таблица Status

Таблица “ServiceOrder”(Услуги заявки)

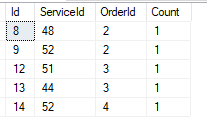


Рис.13 Таблица ServiceOrder

Таблица “Order”(Заявка)

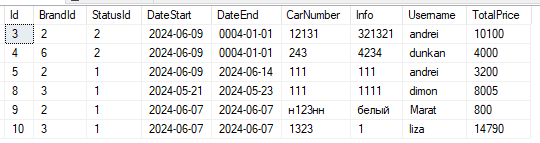


Рис.14 Таблица Order

**3.6 Разработка интерфейса ПС**

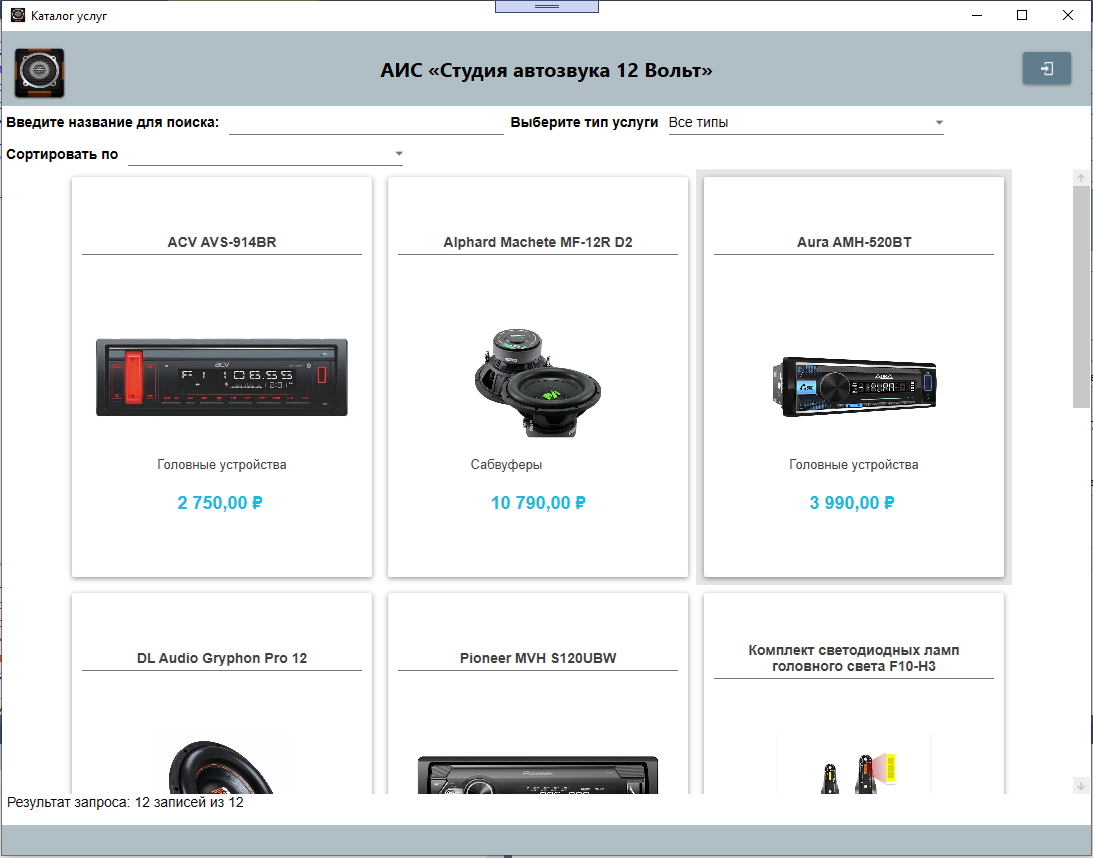
Приложение предназначено для двух типов пользователей: администратора и обычного пользователя. Для входа в панель администратора необходимо ввести свои учётные данные. Для запуска приложения достаточно щелкнуть дважды левой кнопкой мыши по ярлыку.

После запуска программы пользователь приступит к работе с главной формой программы, на которой отбражается каталог автомобилей и цен на аренду в виде плиточного интерфейса. «AvtoSound 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.

**Форма Автоматизированная инфомационная система «Студия автозвука 12 Вольт»**. Основная форма приложения (Рис.15). При разработке интерфейса этой формы было использовано множество различныx компонентов:

1. ComboBox – выпадающий список классов автомобилей для фильтрации;
2. ListBox – списки услуг и автомобильных брендов
3. materialDesign:Card – для отбражения информации об услугах в виде карточки;
4. Buttons – кнопки для перехода на другие формы;
5. TextBox – поля для ввода



Button

ComboBox

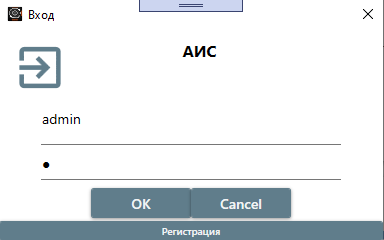
TextBox

materialDesign:Card

TextBox

Рис.15 Автоматизированная инфомационная система «Студия автозвука 12 Вольт»

**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.16)



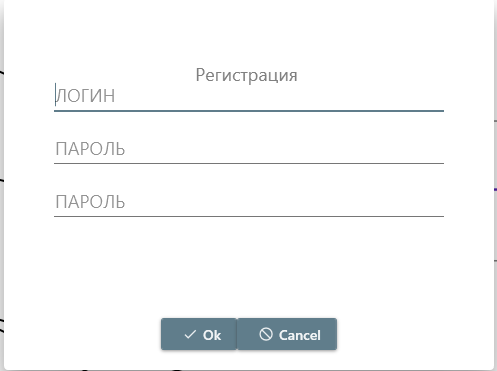
Buttons

TextBox

PasswordBox

Рис.16 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.17)



Buttons

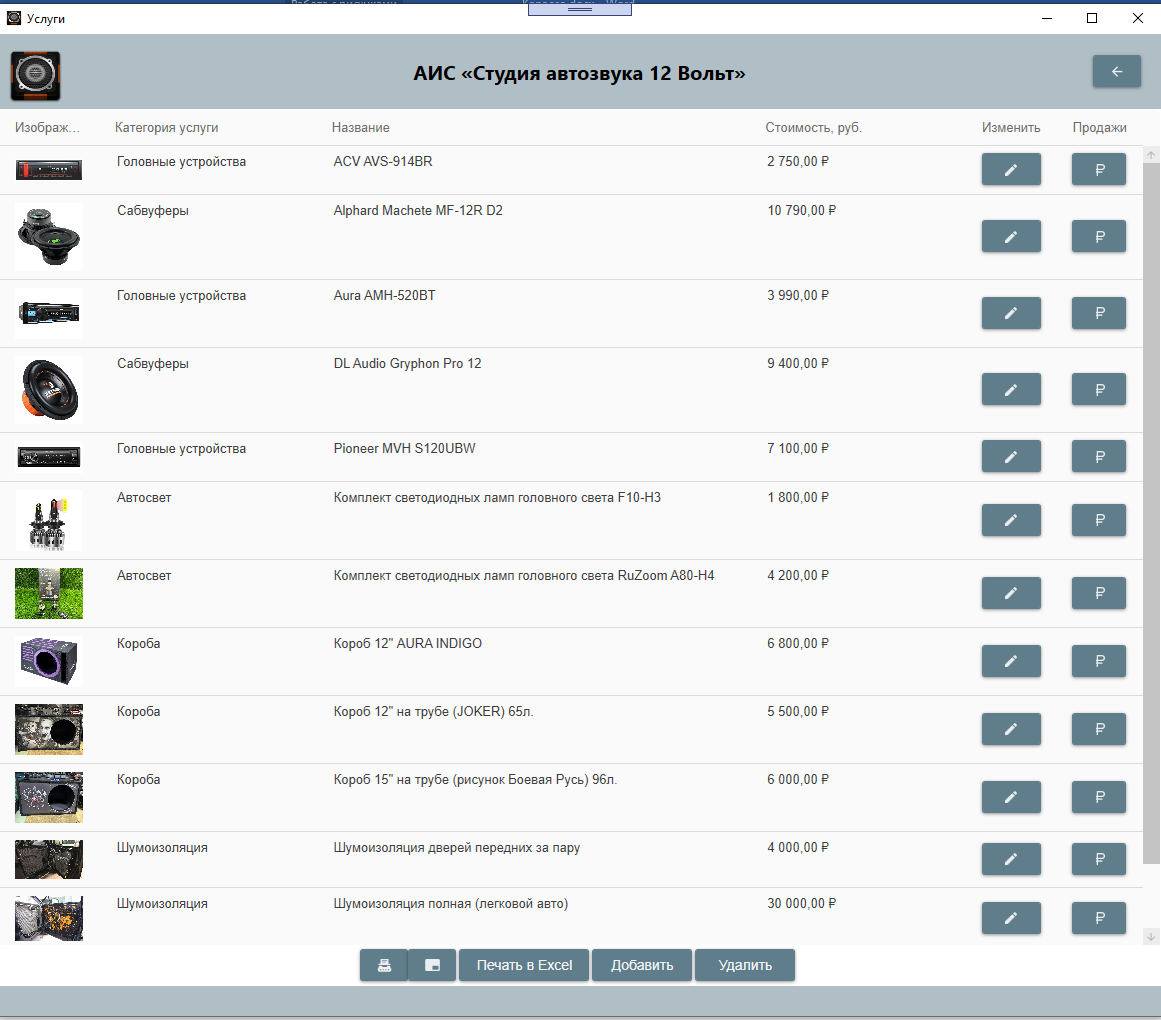
TextBox

PasswordBox

Рис.17 Форма Регистрации

**Форма Услуги(**Рис.18**)**. При разработке интерфейса этой формы были использованы компоненты:

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



Buttons

DataGrid

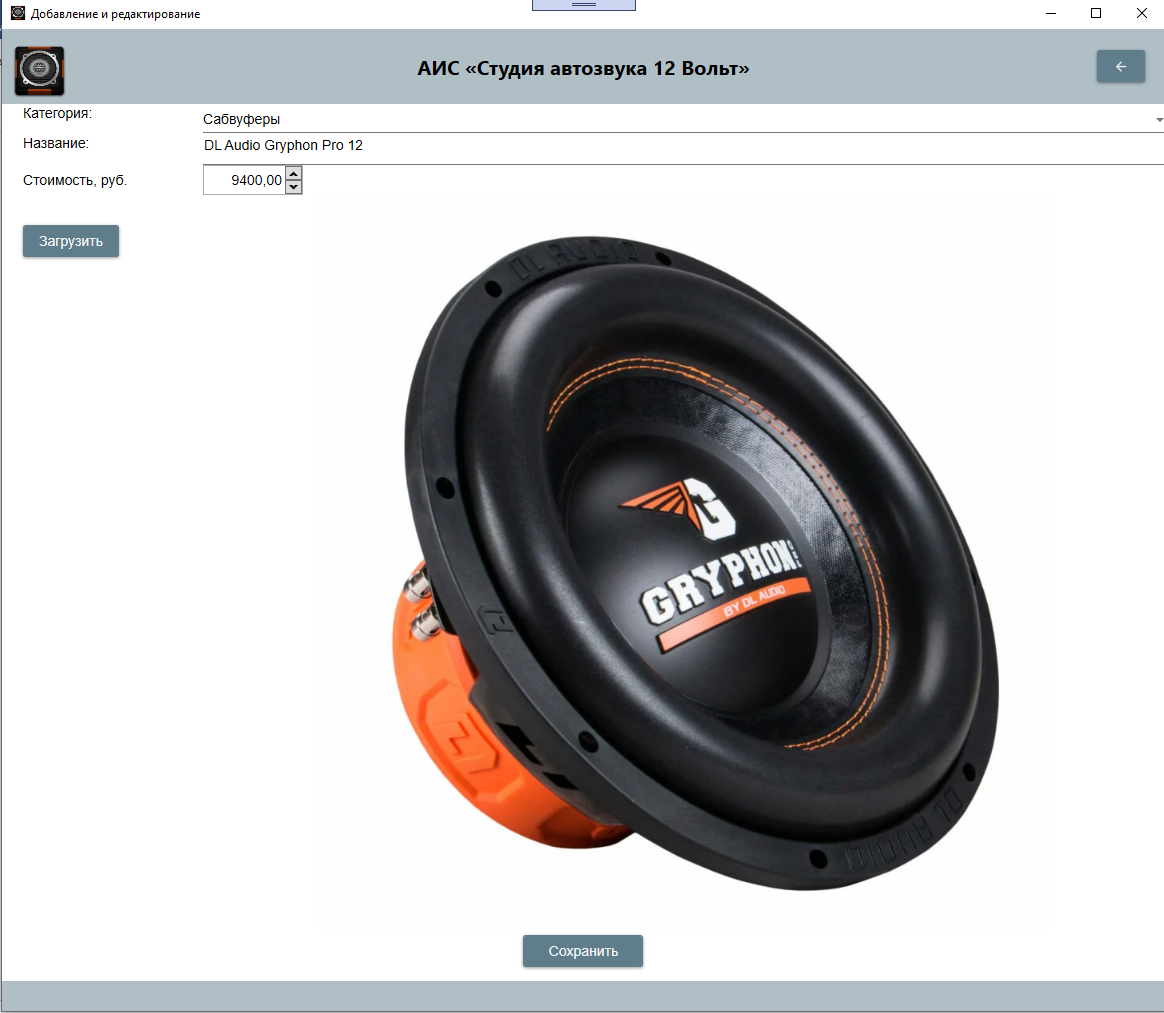
Image

TextBlock

Рис.18 Форма Услуги

**Страница добавления и редактирования(**Рис.19**)**. При разработке интерфейса этой формы были использованы компоненты:

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



Buttons

Image

TextBlock

ComboBox

TextBox

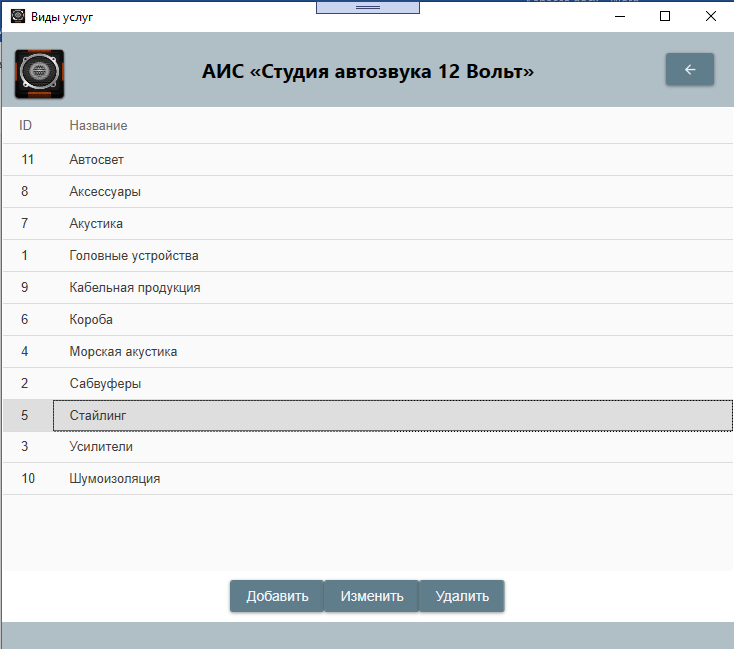
Button

DoubleUpDown

Grid

Рис.19 Страница добавления и редактирования

**Страница Виды услуг**. Интерфейс данной формы состоит из компонентов: DataGrid и кнопок: Добавить, Изменить, Удалить, Назад. (Рис.20)



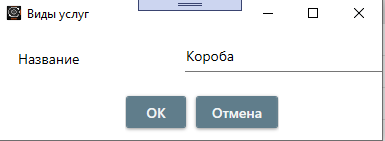
Button – кнопки Добавить, Изменить и Удалить

Button – кнопка Назад

Компонент DataGrid - таблица

Рис.20 Виды услуг

**Окно добавления и редактирования услуги**. Интерфейс данной формы состоит из компонентов: TextBox – текстовое поле ввода и кнопок Button. (Рис.21)



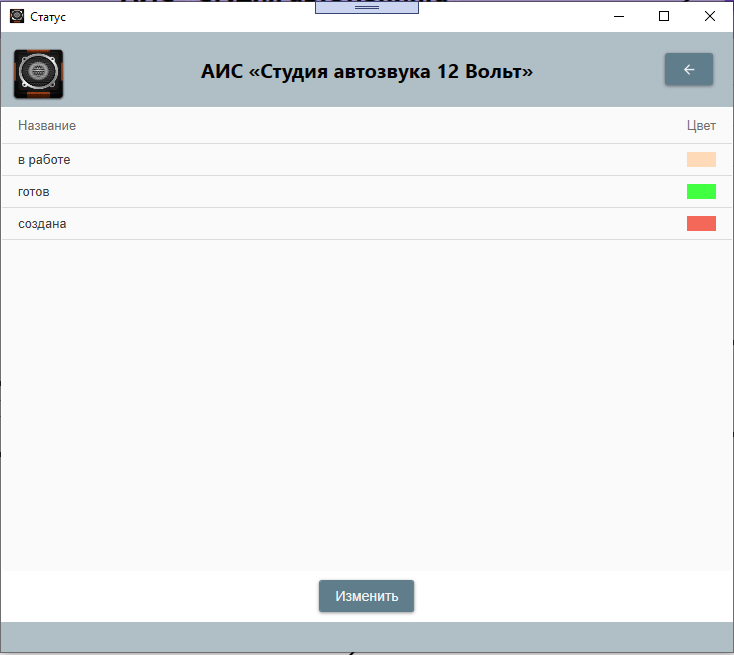
Button – кнопки Ок и Отмена

TextBox

Рис.21 Окно добавления и редактирования

**Страница Статус**. Интерфейс данной формы состоит из компонентов: DataGrid и кнопки Изменить. (Рис.22)

Компонент DataGrid - таблица

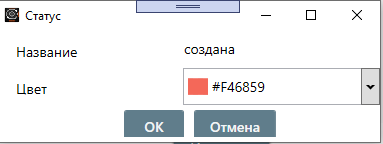


Button – кнопки Добавить, Изменить и Удалить

Button – кнопка Назад

Рис.22 Страница Статус

**Окно добавления и редактирования**. Интерфейс данной формы состоит из компонентов: TextBox – текстовое поле ввода и кнопок Button. (Рис.23)



Button – кнопки Ок и Отмена

TextBox

ColorPicker

Рис.23 Окно добавления и редактирования

**Страница Автомобильные марки**. Интерфейс данной формы состоит из компонентов: DataGrid и кнопок: Добавить, Изменить, Удалить, Назад. (Рис.24)

Button – кнопки Добавить, Изменить и Удалить

Button – кнопка Назад

Компонент DataGrid - таблица

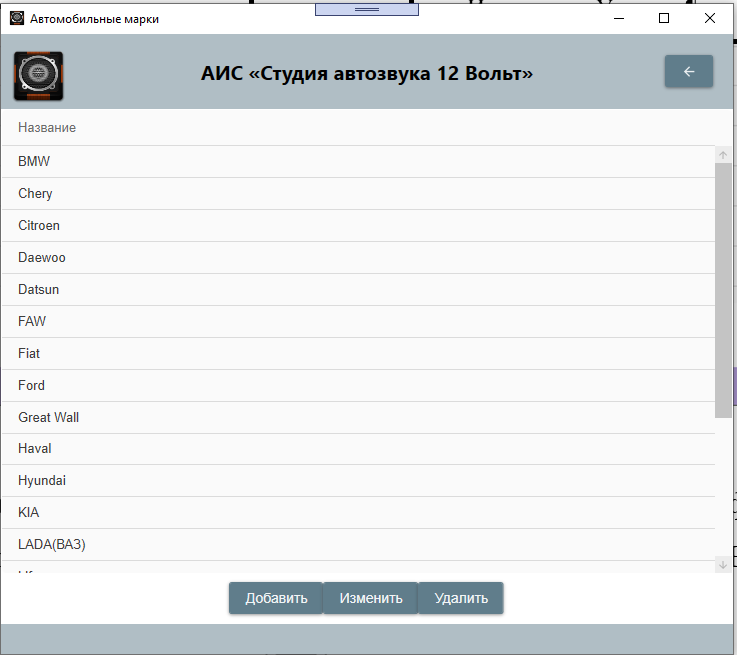
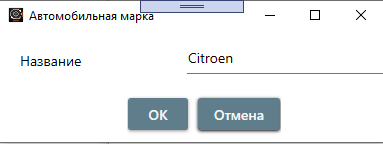


Рис.24 Страница Автомобильные марки

**Окно добавления и редактирования**. Интерфейс данной формы состоит из компонентов: TextBox – текстовое поле ввода и кнопок Button. (Рис.25)

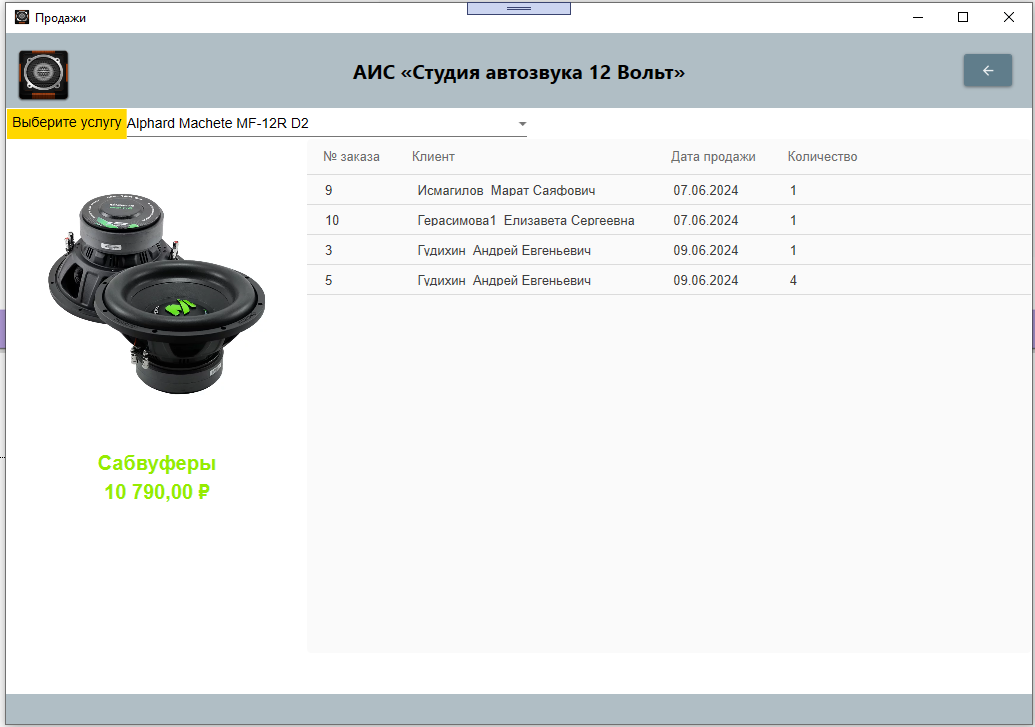


Button – кнопки Ок и Отмена

TextBox

Рис.25 Окно добавления и редактирования

**Страница Продажи**. Интерфейс данной формы состоит из компонентов: DataGrid, и Combobox. (Рис.26)



Button – кнопка Назад

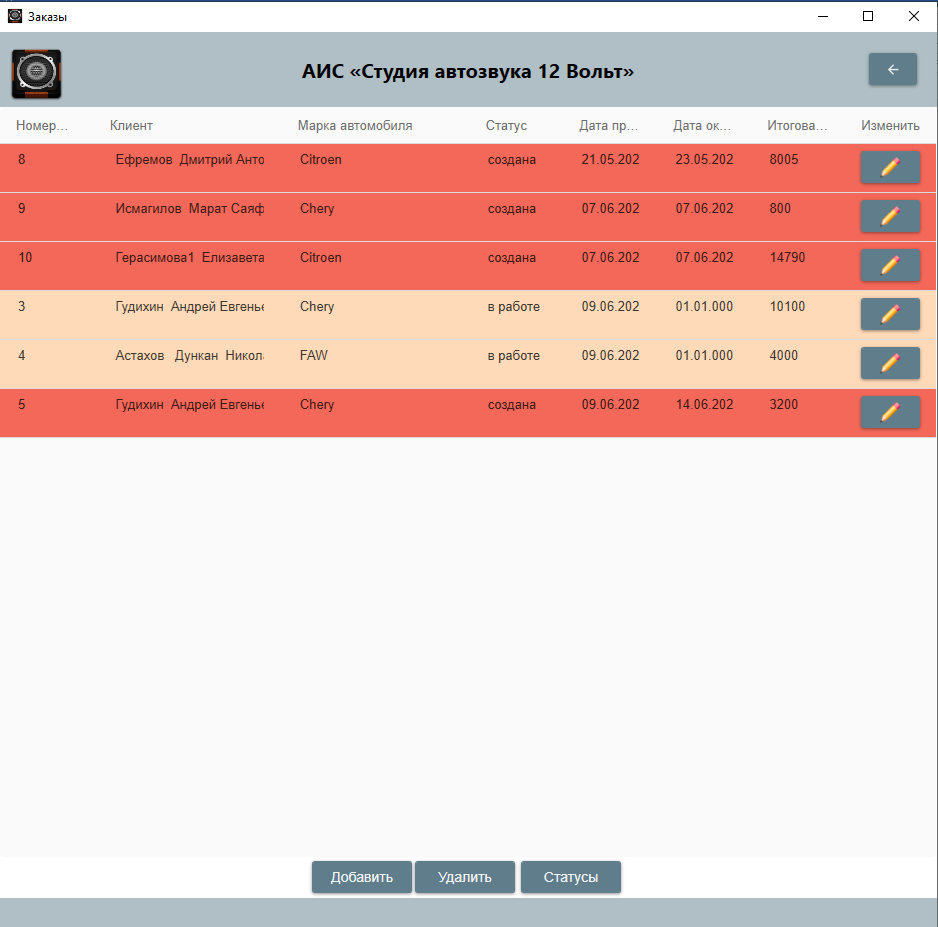
Компонент Combobox – выпадающий список

Компонент Image - изображение

Компонент DataGrid - таблица

Рис.26 Страница Продажи

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



Button – кнопки Добавить и Удалить

Button – кнопка Назад

Компонент DataGrid - таблица

Рис.27 Страница Заказы

**Страница добавления и редактирования(**Рис.28**)**. При разработке интерфейса этой формы были использованы компоненты:

1. ComboBox – выпадающий список;
2. Buttons – кнопки Сохранения, Вывод в Excel и перехода назад;
3. DatePicker – поле выбора даты;
4. DoubleUpDown – поле для ввода вевщественных чисел;
5. TextBox – поля для ввода текста;
6. TextBlock – текстовые метки.



Buttons

TextBlock

DatePicker

ComboBox

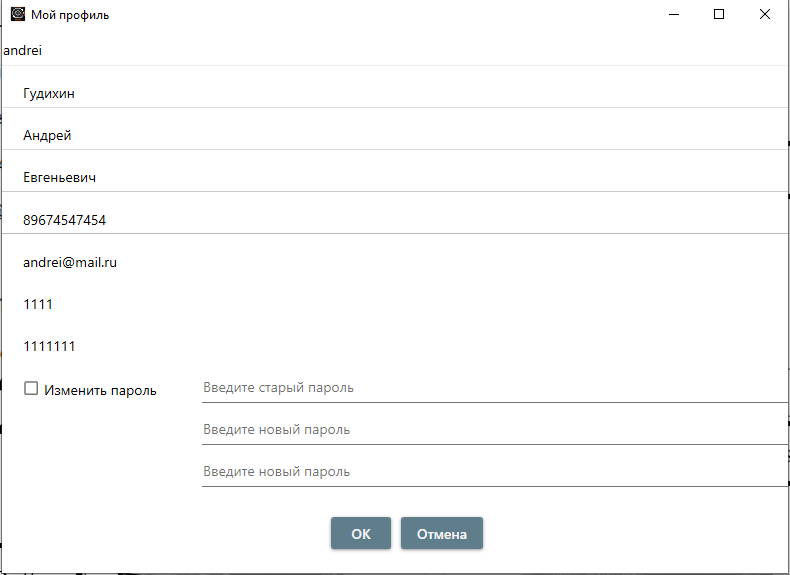
TextBox

Button

Grid

Рис.28 Страница добавления и редактирования

**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.29)



Buttons

Grid

TexBox

PasswordBox

CheckBox

Рис.29 Форма Мой профиль

**3.7 Описание структуры выходной информации**

Выходными данными являются список всех услуг и квитанция.

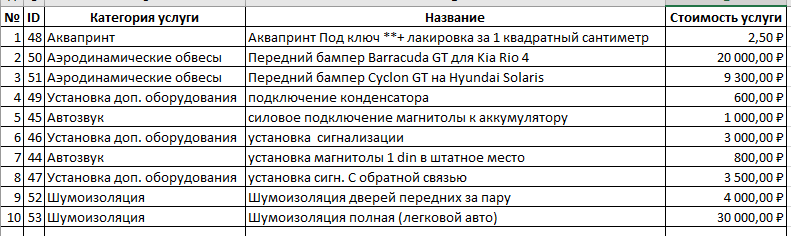


Рис.30 Список услуг

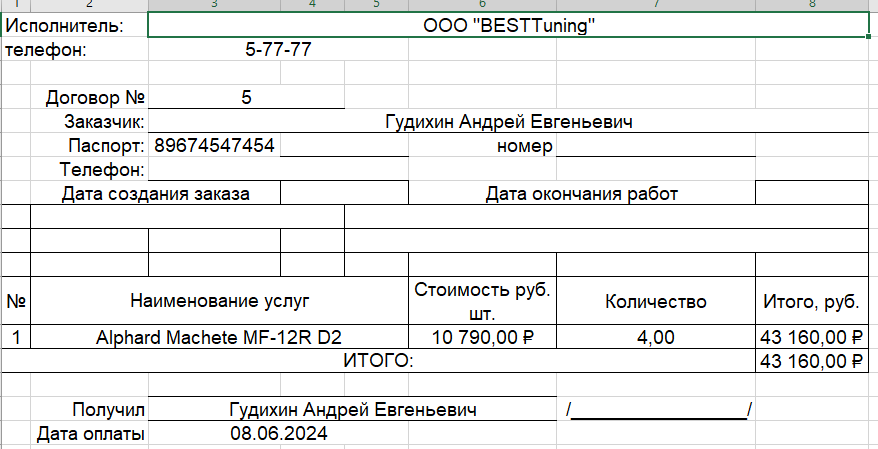


Рис.31 Квитанция

**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 «AvtoSound 1.0».

Подготовительные действия:

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

**4.3 Эксплуатация программного средства**

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

Активировать копию программы AvtoSound 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.32). Форма представляет собой каталог услуг. Каждая запись об услуге представляет собой карточку с краткой информацией об услуге.

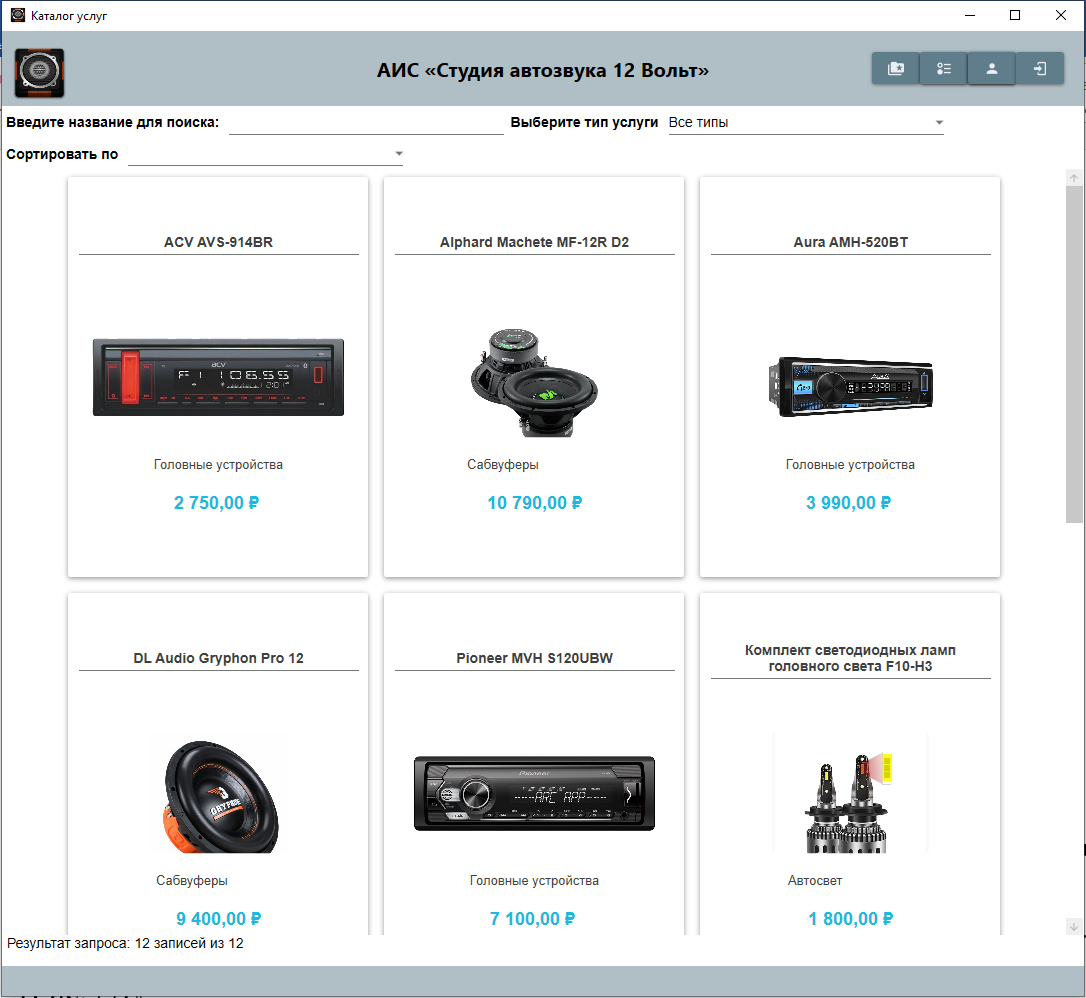


Рис.32 Окно Автоматизированная инфомационная система «Студия автозвука 12 Вольт»

Кликнув по кнопке «Закрыть» программа потребует подтверждение. (Рис.33)



Рис.43 Сообщение программы

Для входа в систему необходимо кликнуть на кнопку . Ввести логин и пароль администратора или логин и пароль пользователя и нажать OK(Рис. 34).

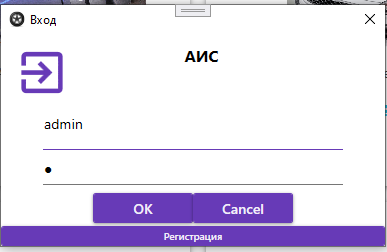


Рис.34 Форма входа в систему

Если вы являетесь новым пользователем в систем, то необходимо нажать на кнопку Регистрация. В появившейся форме требуется заполнить поля логина и пароля. При регистрации роль нового пользователя всегда пользователь(Рис. 35).

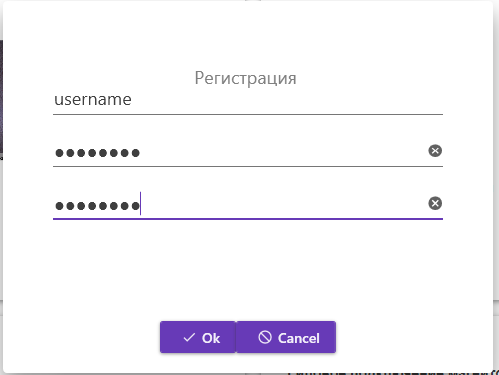


Рис.35 Форма регистрации

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.36)

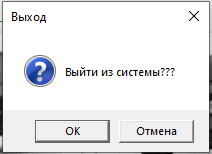


Рис.36 Сообщение программы

Ниже описаны пункты меню управления и информации.

Страница «Услуги». Эта форма доступна только администратору. Для того, чтобы открыть эту форму нужно нажать на кнопку , расположенную на главной форме приложения в правом верхнем углу. Данная страница предназначена для отображения информации об услугах в табличном виде, открытия страницы редактирования, добавления, удаления выделенной записи, а также для перехода на другие страницы приложения. (Рис.37)

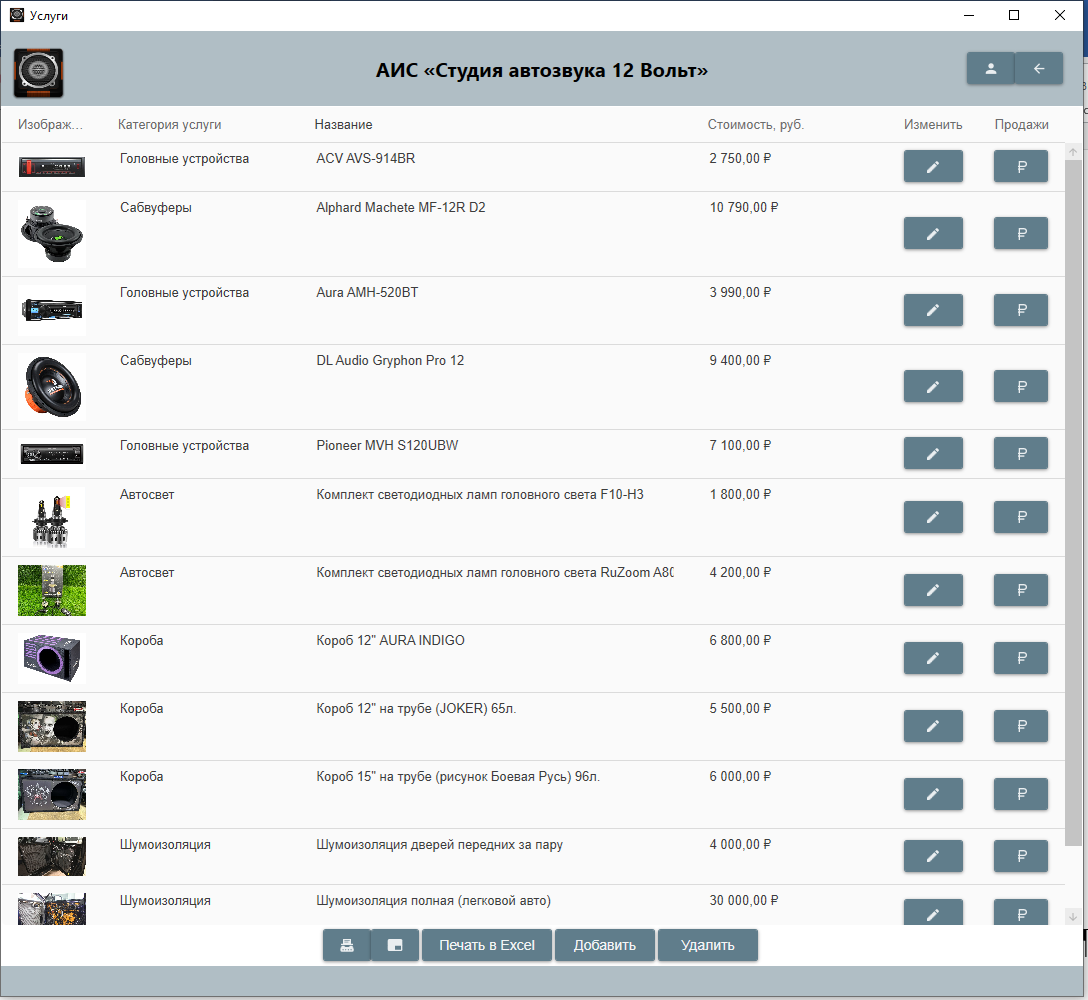


Рис.37 Страница Услуги

Страница Добавление и редактирование. (Рис.38) Для открытия этой страницы нужно на странице услуги выбрать запись и нажать на кнопку Редактировать. Также данную страницу можно открыть в режиме добавления, нажав на соответствующую кнопку на предыдущей странице. Эта страница предназначена для добавления или редактирования сведений об услугах студии.



Рис.38 Страница Добавление и редактирование

«Виды услуг»: эта страница предназначена для просмотра и удаления информации о типах услуг. Добавление и редактирование осуществляется через дополнительную форму. (Рис.39)

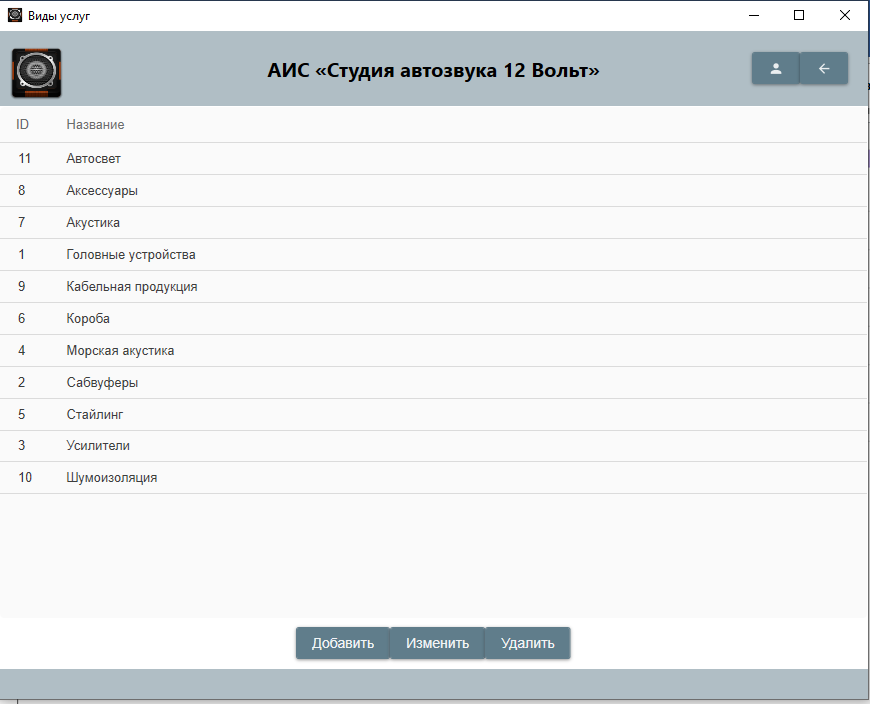


Рис.39 Виды услуг

«Вид услуги»: эта форма предназначена для добавления и редактирования записи. (Рис.40)

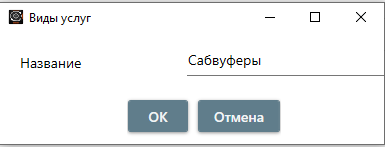


Рис.40 Вид услуги

«Автомобильные марки»: эта страница предназначена для просмотра и удаления информации об автомобильных марках. Добавление и редактирование осуществляется через дополнительную форму. (Рис.41)

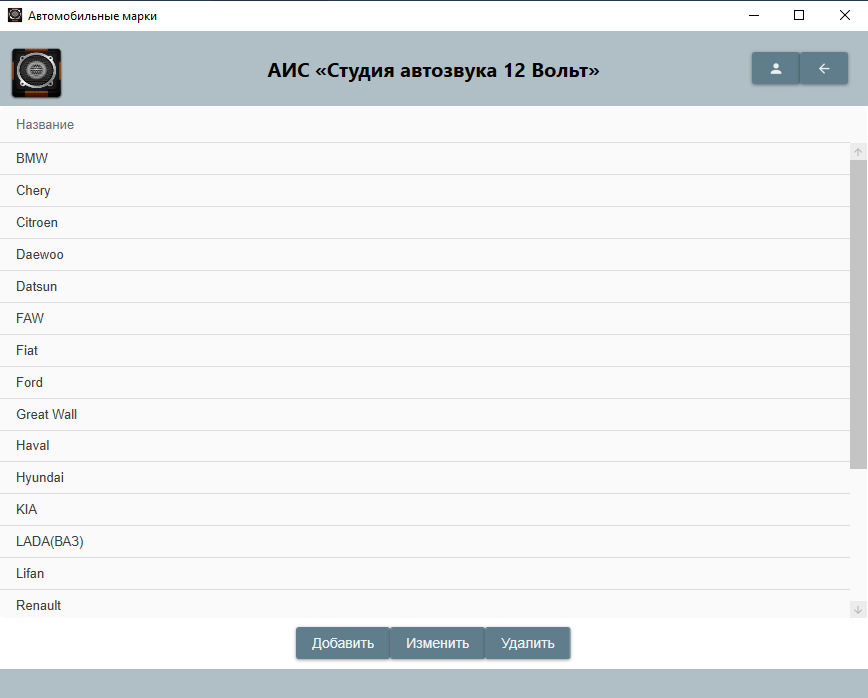


Рис.41 Автомобильные марки

«Автомобильная марка»: эта форма предназначена для добавления и редактирования записи об автомобильной марке. (Рис.42)

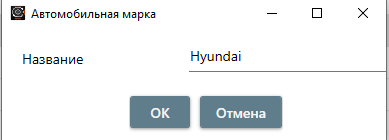


Рис.42 Автомобильная марка

«Продажи»: эта страница предназначена для просмотра информации о продажах услуг.. (Рис.43)

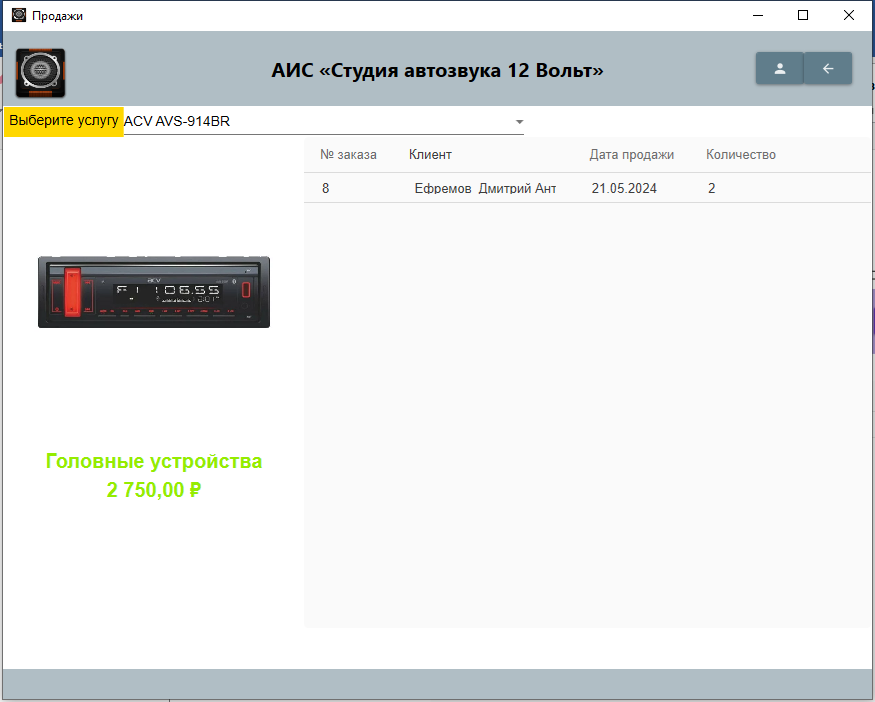


Рис.43 Продажи

«Заказы»: эта страница страница предназначена для отображения информации обзаказах в табличном виде, открытия страницы редактирования, добавления, удаления выделенной записи.(Рис.44)

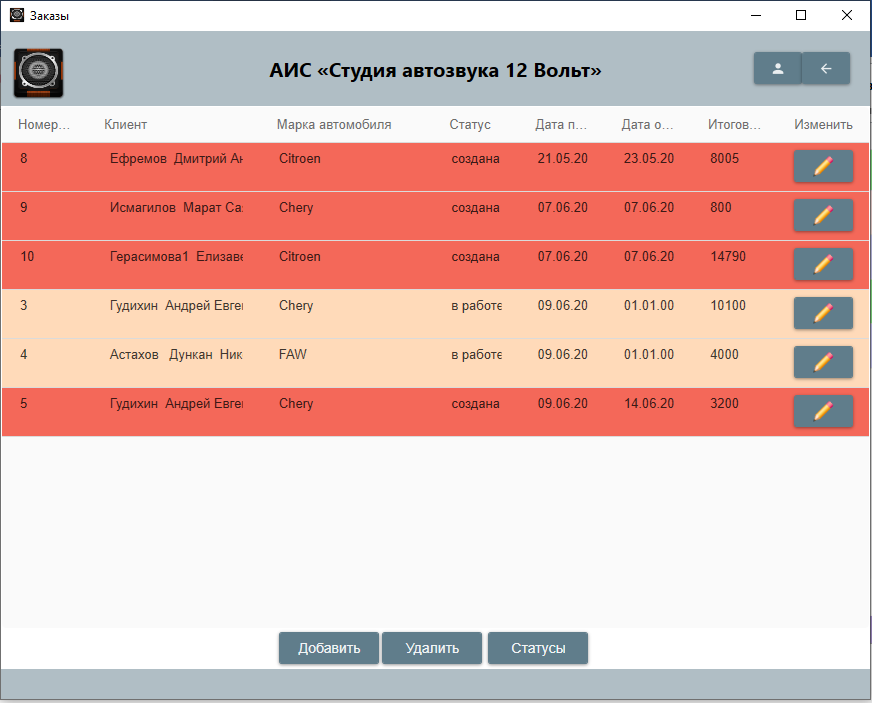


Рис.44 Страница Заказы

Страница Добавление и редактирование. (Рис.45) Для открытия этой страницы нужно на странице заказы 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 прокате.

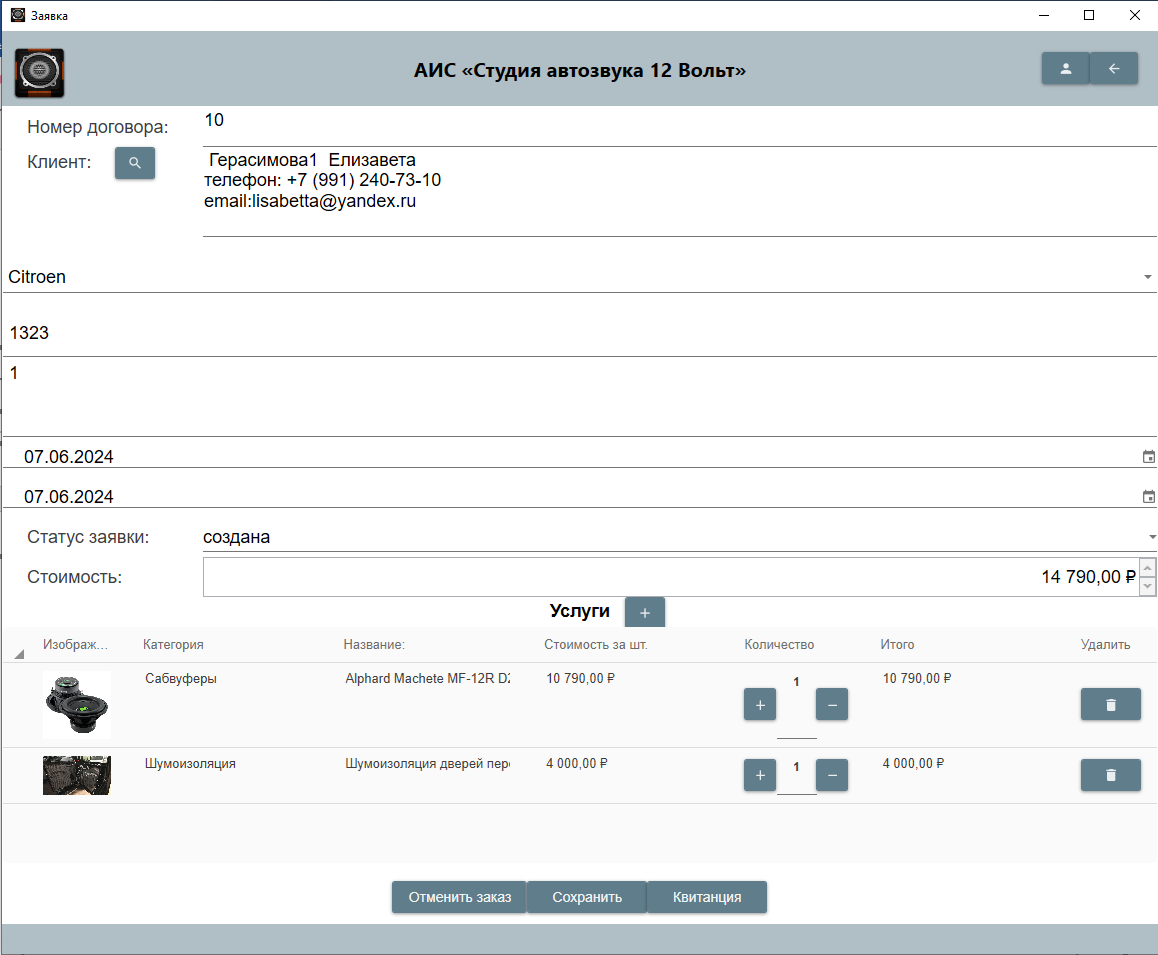


Рис.45 Страница Добавление и редактирование

«Статусы»: эта страница предназначена для просмотра информации о статусах. Редактирование осуществляется через дополнительную форму. (Рис.46)

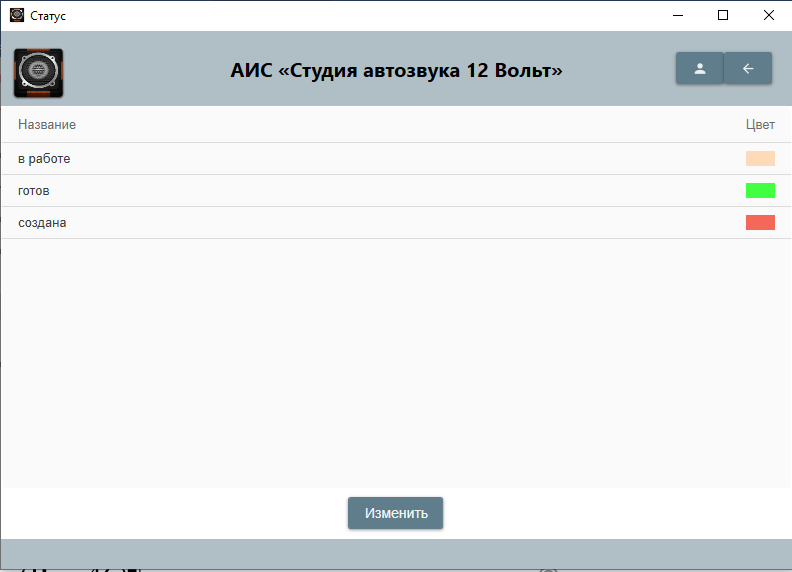


Рис.46 Статусы

«Статус»: эта форма предназначена для редактирования цвета выбранного статуса. (Рис.47)

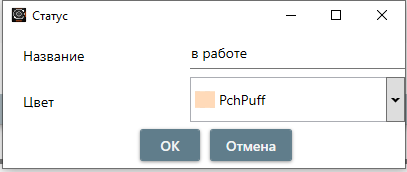


Рис.47 Статус

«Мой профиль»: эта форма предназначена для просмотра и редактирования информации о пользователе. (Рис.48)

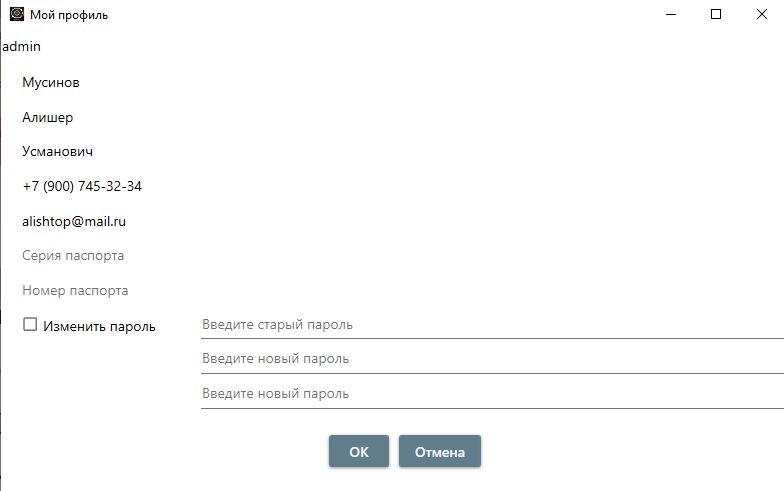


Рис.48 Форма Мой профиль

Работа с AvtoSound 1.01

После запуска программы Вы перейдете на главную форму программы. В формах «Услуги», «Заказы», «Автомобильные марки», «Виды услуг», «Мой профиль», «Статусы» 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 AvtoSound 1.01 кликните на соответствующей кнопке «Выход».

Основные кнопки AvtoSound 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 Прайс-лист |
|  | Выгрузить информацию в Excel |

**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.49)



Рис.49 Сообщение Подтверждение удаления

После добавления новой записи, пользователю выводится сообщение. (Рис.50)



Рис.50 Сообщение Запись добавлена

После обновления записи, пользователю выводится сообщение. (Рис.51)



Рис.51 Сообщение Запись обновлена

При попытке удалить связанные записи, программа выводит пользователю сообщение о невозможности действия. (Рис.52)



Рис.52 Сообщение Ошибка удаления

**5 РАЗРАБОТКА ТЕСТОВОГО НАБОРА**

**5.1 Определение верхней границы количества тестов**

Для разработанного алгоритма был построен граф.

****

Рис.53 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 AvtoSound 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 Статусы

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

Таблица 3 – Тестирование некорректного добавления информации в таблицу Статусы

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

Таблица 4 – Тестирование корректного редактирования информации в таблице Статусы

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

Таблица 5 – Тестирование некорректного редактирования информации в таблице Статусы

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

Таблица 6 – Тестирование корректного добавления информации в таблицу Услуги

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного добавления информации в таблицу Услуги | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Услуги | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Фото= загружено  Название = ACV AVS-914BR,  Тип услуги = Головные устройства,  Стоимость = 2750 р. | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Услуги открыта | | Отображается форма Услуги |  |

Таблица 7 – Тестирование некорректного добавления информации в таблицу Услуги

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации в таблицу Услуги | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Услуги | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Фото=  Название =  Тип услуги = Головные устройства,  Стоимость = 2750 р. | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, поля пустые | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Услуги  открыта | | Отображается форма Услуги |  |

Таблица 8 – Тестирование корректного редактирования информации в таблице Услуги

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного редактирования информации в таблице Услуги | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Услуги | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  Фото= загружено  Название = ACV AVS-914BR,  Тип услуги = Головные устройства,  Стоимость = 2750 р. | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Услуги открыта | | Отображается форма Услуги |  |

Таблица 9 – Тестирование некорректного редактирования информации в таблице Услуги

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного редактирования информации в таблице Услуги | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Услуги | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  Фото=  Название =  Тип услуги = Головные устройства,  Стоимость = 2750 р. | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, поля пустые | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Услуги открыта | | Отображается форма Услуги |  |

Таблица 10 – Тестирование корректного добавления информации в Вид услуги

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

Таблица 11 – Тестирование некорректного добавления информации в таблицу Вид услуги

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

Таблица 12 – Тестирование корректного редактирования информации в таблице Вид услуги

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

Таблица 13 – Тестирование некорректного редактирования информации в таблице Вид услуги

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

**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 |
| Услуги | 4 | 4 | | 0 |
| Статусы | 4 | 4 | | 0 |
| Вид услуги | 4 | 4 | | 0 |
| Продажи | 4 | 4 | | 0 |
| Заказы | 4 | 4 | | 0 |
| Мой профиль | 4 | 4 | | 0 |
| Автомобильные марки | 4 | 4 | | 0 |
| ИТОГО | 32 | 36 | 0 | |

По результатам тестирования были выявлены и устранены следующие проблемы:

При выполнении теста таблица №8. Тестирование формы «Услуга», Некорректный запрос

**Ошибка**



Рис.68 Ошибка теста №8

**Причина**

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

**Принятые меры**

Ошибка устранена. В поле стоимость 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 = 5000 тысяч строк кода/1 месяца=5000 тысяч строк кода/месяц

Качество = 30 ошибок /5 тысяч строк кода=6 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, 2022. - 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,2022,604с.

5. Енин А., Енин Н. Локальная СУБД своими руками. Учимся на примерах.-М.:СОЛОН-ПРЕСС,2022,464с.,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, 2022-512с.

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#. 2022 г.

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

14. Никита Культин. Основы программирования в MicrosoftVisualC# 2021 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#. 2022 г.

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

**Приложение А**

«Автоматизированная инфомационная система «Студия автозвука 12 Вольт»»

Текст кода программного средства

460.ЗМК.00013-01 12 01

**Текст кода программного средства**

<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.DeepPurple.xaml" />

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

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</Application.Resources>

</Application>

<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="#95EC00"/>

<!--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>

<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/1.ico"

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}"/>

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

Margin="10,10,0,0"

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

<TextBlock Grid.Row="0" Text="АИС «Студия автозвука 12 Вольт»"

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" Visibility="Collapsed">

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

</Button>

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

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

</Button>

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

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

</Button>

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

<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.Question);

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;

if (Manager.CurrentUser is null)

return;

if (Manager.CurrentUser.Role ==true)

BtnEditGoods.Visibility = Visibility.Collapsed;

else

BtnMyAccount.Visibility = Visibility.Collapsed;}

else

{ BtnBack.Visibility = Visibility.Collapsed;

BtnEnter.Visibility = Visibility.Visible;

if (Manager.CurrentUser is null)

return;

BtnOrder.Visibility = Visibility.Visible;

if (Manager.CurrentUser.Role == true)

BtnEditGoods.Visibility = Visibility.Visible;

else

BtnMyAccount.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.Question);

if (messageBoxResult == MessageBoxResult.OK)

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

Manager.CurrentUser = null;

BtnEditGoods.Visibility = Visibility.Collapsed;

BtnOrder.Visibility = Visibility.Collapsed;

BtnMyAccount.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;}

else

{ BtnMyAccount.Visibility = Visibility.Visible;

BtnOrder.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());

}

}

}

<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 = AutoTuningBDEntities.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 = AutoTuningBDEntities.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 = "";

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

AutoTuningBDEntities.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="Каталог услуг"

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" Margin="3">

<TextBlock Text="Введите название для поиска:" VerticalAlignment="Center" FontWeight="Bold"

/>

<TextBox Width="275" x:Name="TBoxSearch" Margin="10 0 0 0"

TextChanged="TBoxSearchTextChanged"/>

</StackPanel>

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

<TextBlock Text="Выберите тип услуги" VerticalAlignment="Center" FontWeight="Bold"

/>

<ComboBox Width="275" Margin="10 0 0 0" x:Name="ComboCategory"

SelectionChanged="ComboTypeSelectionChanged"

DisplayMemberPath="Name"/>

</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 Content="По возрастанию цены"/>

<ComboBoxItem Content="По убыванию цены"/>

</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="400" >

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

<Grid.RowDefinitions>

<RowDefinition Height="\*" />

<RowDefinition Height="140" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<TextBox IsReadOnly="True" Text="{Binding Name}" TextWrapping="Wrap"

HorizontalAlignment="Center" FontWeight="Bold" VerticalAlignment="Center" HorizontalContentAlignment="Center"

FontSize="14" Margin="10" Width="280"/>

<Image Source="{Binding Path=GetPhoto}" Grid.Row="1" Height="140" Width="260" Margin="10" Stretch="Uniform" />

<StackPanel Grid.Row="2" Margin="10" HorizontalAlignment="Center">

<TextBlock TextWrapping="Wrap" VerticalAlignment="Center"

Text="{Binding Category.Name}"/>

<TextBlock TextWrapping="Wrap" VerticalAlignment="Center" FontSize="18" Foreground="#FF18BADE" Margin="20" FontWeight="Bold"

Text="{Binding Price, StringFormat='c',

ConverterCulture='ru-RU'}" />

<!--<StackPanel HorizontalAlignment="Right" Orientation="Horizontal" Margin="8">

<Button Style="{StaticResource MaterialDesignToolButton}" Width="30" x:Name="BtnShowMore" ToolTip="Показать на карте" Padding="2 0 2 0"

materialDesign:RippleAssist.IsCentered="True" Click="BtnShowMore\_Click">

<materialDesign:PackIcon Kind="Pin" />

</Button>

</StackPanel>-->

</StackPanel>

</Grid>

</materialDesign:Card>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

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

</Grid>

</Page>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

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 categories = AutoTuningBDEntities.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 = AutoTuningBDEntities.GetContext().Services.OrderBy(p => p.Name).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)

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

LViewGoods.ItemsSource = AutoTuningBDEntities.GetContext().Services.OrderBy(p => p.Name).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 = AutoTuningBDEntities.GetContext().Services.OrderBy(p => p.Name).ToList();

// выбор только тех товаров, которые принадлежат данному производителю

if (ComboCategory.SelectedIndex > 0)

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

// выбор тех товаров, в названии которых есть поисковая строка

currentGoods = currentGoods.Where(p => p.Name.ToLower().Contains(TBoxSearch.Text.ToLower())).ToList();

// сортировка

if (ComboSort.SelectedIndex >= 0)

{ // сортировка по возрастанию цены

if (ComboSort.SelectedIndex == 0)

currentGoods = currentGoods.OrderBy(p => p.Price).ToList();

// сортировка по убыванию цены

if (ComboSort.SelectedIndex == 1)

currentGoods = currentGoods.OrderByDescending(p => p.Price).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());}}

}

<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="Услуги" 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="Изображение">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

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

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTextColumn Binding="{Binding Category.Name}" Header="Категория услуги" Width="\*"/>

<DataGridTextColumn Binding="{Binding Name}" Header="Название" 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="Продажи">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

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

<materialDesign:PackIcon Kind="CurrencyRub" />

</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="Typewriter" Padding="0"/>

</Button>

<Button x:Name="BtnBrands"

Click="BtnBrands\_Click" ToolTip="Автомобильные марки">

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

</Button>

<Button x:Name="BtnExcel" Width="130" Margin="3"

Content="Печать в Excel" Click="BtnExcel\_Click"/>

<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<Service> services;

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 Service));}

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

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

services = AutoTuningBDEntities.GetContext().Services.OrderBy(p => p.Name).ToList();

DataGridGood.ItemsSource = services;}}

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<Service>().ToList();

// вывод сообщения с вопросом Удалить запись?

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

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

//если пользователь нажал ОК пытаемся удалить запись

if (messageBoxResult == MessageBoxResult.OK)

{ try

{ // берем из списка удаляемых товаров один элемент

Service x = selectedGoods[0];

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

// если да, то выбрасывается исключение и удаление прерывается

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

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

// удаляем товара

AutoTuningBDEntities.GetContext().Services.Remove(x);

//сохраняем изменения

AutoTuningBDEntities.GetContext().SaveChanges();

MessageBox.Show("Записи удалены");

services.Clear();

services = AutoTuningBDEntities.GetContext().Services.OrderBy(p => p.Name).ToList();

DataGridGood.ItemsSource = null;

DataGridGood.ItemsSource = services;}

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 Service));}

// отображение номеров строк в DataGrid

private void DataGridGoodLoadingRow(object sender, DataGridRowEventArgs e)

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

private void PrintExcel()

{ string fileName = AppDomain.CurrentDomain.BaseDirectory + "\\" + "Services" + ".xltx";

Excel.Application xlApp = new Excel.Application();

Excel.Worksheet xlSheet = new Excel.Worksheet();

try

{ //добавляем книгу

xlApp.Workbooks.Open(fileName, System.Type.Missing, System.Type.Missing, System.Type.Missing, System.Type.Missing,

System.Type.Missing, System.Type.Missing, System.Type.Missing, System.Type.Missing,

System.Type.Missing, System.Type.Missing, System.Type.Missing, System.Type.Missing,

System.Type.Missing, System.Type.Missing);

//делаем временно неактивным документ

xlApp.Interactive = false;

xlApp.EnableEvents = false;

Excel.Range xlSheetRange;

//выбираем лист на котором будем работать (Лист 1)

xlSheet = (Excel.Worksheet)xlApp.Sheets[1];

//Название листа

xlSheet.Name = "Список товаров";

int row = 2;

int i = 0;

foreach (Service car in services)

{ xlSheet.Cells[row, 1] = (i + 1).ToString();

string s;

//// DateTime y = Convert.ToDateTime(dtOrders.Rows[i].Cells[1].Value);

xlSheet.Cells[row, 2] = car.Id.ToString();

s = "";

if (car.CategoryId != null) s = car.Category.Name.ToString();

xlSheet.Cells[row, 3] = s;

//if (car.BrandId != null) s = car.Brand.Name.ToString();

//xlSheet.Cells[row, 4] = s;

//s = "";

xlSheet.Cells[row, 4] = car.Name.ToString();

//s = "";

//if (car.EngineCapacity != null) s = car.EngineCapacity.ToString();

//xlSheet.Cells[row, 6] = s;

//s = "";

//if (car.FuelRate != null) s = car.FuelRate.ToString();

//xlSheet.Cells[row, 7] = s;

//s = "";

//if (car.TrunkVolume != null) s = car.TrunkVolume.ToString();

//xlSheet.Cells[row, 8] = s;

s = "";

if (car.Price != null) s = car.Price.ToString();

xlSheet.Cells[row, 5] = s;

row++;

Excel.Range r = xlSheet.get\_Range("A" + row.ToString(), "E" + row.ToString());

r.Insert(Excel.XlInsertShiftDirection.xlShiftDown);

i++;}

row--;

xlSheetRange = xlSheet.get\_Range("A2:E" + (row + 1).ToString(), System.Type.Missing);

xlSheetRange.Borders.LineStyle = true;

//xlSheet.Cells[row + 1, 9] = "=SUM(I2:I" + row.ToString() + ")";

//xlSheet.Cells[row + 1, 8] = "ИТОГО:";

row++;

//выбираем всю область данных\*/

xlSheetRange = xlSheet.UsedRange;

//выравниваем строки и колонки по их содержимому

xlSheetRange.Columns.AutoFit();

xlSheetRange.Rows.AutoFit();}

catch (Exception ex)

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

finally

{ //Показываем ексель

xlApp.Visible = true;

xlApp.Interactive = true;

xlApp.ScreenUpdating = true;

xlApp.UserControl = true;}}

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

{ PrintExcel();}

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

{ Manager.MainFrame.Navigate(new BrandPage());}

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 SellPage((sender as Button).DataContext as Service));}}

}

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

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="Заказы" 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 договора" Width="\*"/>

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

<DataGridTextColumn Binding="{Binding Brand.Name}" Header="Марка автомобиля" Width="2\*"/>

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

<DataGridTextColumn Binding="{Binding DateStart, StringFormat='d', ConverterCulture=ru-RU}" Header="Дата приёма заявки" Width="\*"/>

<DataGridTextColumn Binding="{Binding DateEnd, StringFormat='d', ConverterCulture=ru-RU}" Header="Дата окончания работ" Width="\*"/>

<DataGridTextColumn Binding="{Binding TotalPrice}" Header="Итоговая цена" Width="\*"/>

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

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

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

<Image Source="/WpfAssortmentCheck;component/Resources/edit.png" />

</Button>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

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

<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="Статусы" Click="BtnStatus\_Click"/>

</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 WpfAssortmentCheck.Windows;

using Excel = Microsoft.Office.Interop.Excel;

namespace WpfAssortmentCheck.Pages

{

/// <summary>

/// Логика взаимодействия для OrderPage.xaml

/// </summary>

public partial class OrderPage : Page

{ List<Order> rents;

User \_currentUser;

public OrderPage()

{ 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;

DataGridGood.Columns[8].Visibility = Visibility.Collapsed;

//загрузка обновленных данных

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

rents = AutoTuningBDEntities.GetContext().Orders.OrderBy(p => p.DateStart).ToList();

DataGridGood.ItemsSource = rents;}

else

{ BtnStatus.Visibility = Visibility.Collapsed;

DataGridGood.Columns[7].Visibility = Visibility.Collapsed;

DataGridGood.ItemsSource = null;

//загрузка обновленных данных

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

rents = AutoTuningBDEntities.GetContext().Orders.Where(x => x.Username == \_currentUser.UserName).OrderBy(p => p.DateStart).ToList();

DataGridGood.ItemsSource = rents;}}

private void PageIsVisibleChanged(object sender, DependencyPropertyChangedEventArgs e)

{ //событие отображения данного Page

// обновляем данные каждый раз когда активируется этот Page

if (Visibility == Visibility.Visible)

{ LoadData();}}

private void BtnAddClick(object sender, RoutedEventArgs e)

{ // открытие AddGoodPage для добавления новой записи

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

private void BtnDeleteClick(object sender, RoutedEventArgs e)

{ // удаление выбранного товара из таблицы

//получаем все выделенные товары

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

// вывод сообщения с вопросом Удалить запись?

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

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

//если пользователь нажал ОК пытаемся удалить запись

if (messageBoxResult == MessageBoxResult.OK)

{ try

{ // берем из списка удаляемых товаров один элемент

Order x = selectedGoods[0];

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

// удаляем товара

AutoTuningBDEntities.GetContext().Orders.Remove(x);

//сохраняем изменения

AutoTuningBDEntities.GetContext().SaveChanges();

MessageBox.Show("Записи удалены");

LoadData();}

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 Order));}

// отображение номеров строк в DataGrid

private void DataGridGoodLoadingRow(object sender, DataGridRowEventArgs e)

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

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

{ Manager.MainFrame.Navigate(new StatusPage());}

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

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

}

<Page x:Class="WpfAssortmentCheck.Windows.AddNewOrderPage"

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.Windows"

mc:Ignorable="d"

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

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

d:DesignHeight="600" d:DesignWidth="800" Style="{StaticResource base\_page}"

Title="Заявка" IsVisibleChanged="Page\_IsVisibleChanged">

<Grid >

<Grid.RowDefinitions>

<RowDefinition Height="0"/>

<RowDefinition Height="40" x:Name="rwOrderNum"/>

<RowDefinition Height="90"/>

<RowDefinition Height="80"/>

<RowDefinition Height="40"/>

<RowDefinition Height="80"/>

<RowDefinition Height="40"/>

<RowDefinition Height="40"/>

<RowDefinition Height="40"/>

<RowDefinition Height="40" x:Name="rwPrice"/>

<RowDefinition Height="30" x:Name="rwServicesName"/>

<RowDefinition Height="\*" x:Name="rwServices"/>

<RowDefinition Height="60" x:Name="rwButton"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="200"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Rectangle Grid.Row="0" Grid.ColumnSpan="2" />

<Label Grid.Row="1" Grid.Column="0" Content="Номер договора:" VerticalAlignment="Center" HorizontalAlignment="Left" Margin="20,0" FontSize="18"/>

<TextBox x:Name="tbOrderId" Grid.Row="1" Grid.Column="1" IsReadOnly="True" Text="{Binding Id}" materialDesign:HintAssist.Hint="Номер договора" FontSize="18" Padding="0,2"/>

<ComboBox x:Name="ComBoBrand" Grid.Row="3" Grid.Column="0" Grid.ColumnSpan="2" SelectedItem="{Binding Brand}" DisplayMemberPath="Name"

SelectedValuePath="Id" materialDesign:HintAssist.Hint="Марка автомобиля" FontSize="18" Padding="5" />

<TextBox x:Name="tbCarNumber" Grid.Row="4" Grid.Column="0" Grid.ColumnSpan="2" Text="{Binding CarNumber}" materialDesign:HintAssist.Hint="Номер автомобиля" FontSize="18" Padding="5"/>

<TextBox x:Name="tbCarInfo" Grid.Row="5" Grid.Column="0" Grid.ColumnSpan="2" Text="{Binding Info}" materialDesign:HintAssist.Hint="Информация" FontSize="18" Padding="5"/>

<!--<TextBox x:Name="tbCarNumber" Grid.Row="4" Grid.Column="1" Text="{Binding CarNumber}" materialDesign:HintAssist.Hint="Номер автомобиля" FontSize="18" Padding="0,2"/>-->

<TextBox x:Name="tbClient" Grid.Row="2" Grid.Column="1" IsReadOnly="True" materialDesign:HintAssist.Hint="Клиент" FontSize="18" Padding="0,2"/>

<StackPanel Grid.Row="2" Grid.Column="0" Orientation="Horizontal">

<Label Content="Клиент:" VerticalAlignment="Top" HorizontalAlignment="Left" Margin="20,0" FontSize="18"/>

<Button x:Name="btnLoadClient" VerticalAlignment="Top" HorizontalAlignment="Right" Click="btnLoadClient\_Click" Padding="0" Width="40" >

<materialDesign:PackIcon Kind="Search"/>

</Button>

</StackPanel>

<DatePicker x:Name="tbStartDate" Grid.Row="6" Grid.Column="0" Padding="20 0 0 0" Grid.ColumnSpan="2" SelectedDate="{Binding DateStart}" materialDesign:HintAssist.Hint="Дата подачи" FontSize="18" />

<DatePicker x:Name="tbEndDate" Grid.Row="7" Grid.Column="0" Grid.ColumnSpan="2" SelectedDate="{Binding DateEnd}" materialDesign:HintAssist.Hint="Дата завершения работ" FontSize="18" Padding="20 0 0 0" />

<Label Grid.Row="8" Grid.Column="0" Content="Статус заявки:" VerticalAlignment="Center" HorizontalAlignment="Left" Margin="20,0" FontSize="18"/>

<ComboBox x:Name="ComboStatus" SelectedItem="{Binding Status}" Grid.Row="8" Grid.Column="1" DisplayMemberPath="Name" FontSize="18"

SelectedValuePath="Id" />

<Label Grid.Row="9" Grid.Column="0" Content="Стоимость:" VerticalAlignment="Center" HorizontalAlignment="Left" Margin="20,0" FontSize="18"/>

<xctk:DoubleUpDown x:Name="TbTotalPrice" Grid.Row="9" Grid.Column="1" materialDesign:HintAssist.Hint="Стоимость" FontSize="18" Padding="0,2" FormatString="C" DefaultValue="0" DisplayDefaultValueOnEmptyText="True"

Value="{Binding TotalPrice}" IsReadOnly="True" Minimum="0"/>

<TextBlock Grid.Row="10" Grid.ColumnSpan="2" FontSize="18" Text="Услуги" TextAlignment="Center" Padding="0,3,0,0" FontWeight="Bold"/>

<DataGrid x:Name="DtOrderPriceList" Grid.Row="11" Grid.ColumnSpan="2" VerticalAlignment="Stretch" HorizontalAlignment="Stretch" AutoGenerateColumns="False" IsReadOnly="True" HeadersVisibility="All" CanUserAddRows="False">

<DataGrid.Columns>

<materialDesign:DataGridTextColumn Binding="{Binding Id}" Header="ID" Width="\*" Visibility="Hidden"/>

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

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

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

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<materialDesign:DataGridTextColumn Binding="{Binding Key.Category.Name}" Header="Категория " Width="\*"/>

<materialDesign:DataGridTextColumn Binding="{Binding Key.Name}" Header="Название: " Width="\*"/>

<materialDesign:DataGridTextColumn Binding="{Binding Key.Price, StringFormat='c', ConverterCulture='ru-RU'}" Header="Стоимость за шт." Width="\*"/>

<!--<materialDesign:DataGridTextColumn Binding="{Binding Value.Count}" Header="Количество" Width="\*"/>-->

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

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<StackPanel Orientation="Horizontal">

<Button x:Name="BtnAdd" Content="{materialDesign:PackIcon Kind=Plus}" Width="32" Padding="0" DataContext="{Binding Key}" Click="BtnAdd\_Click"/>

<TextBox Text="{Binding Value.Count}" IsReadOnly="True" Width="40" FontSize="12" FontWeight="Bold" HorizontalContentAlignment="Center" HorizontalAlignment="Center"/>

<Button x:Name="BtnDelete" Content="{materialDesign:PackIcon Kind=Minus}" Padding="0" Width="32" DataContext="{Binding Key}" Click="BtnDelete\_Click\_1"/>

</StackPanel>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<materialDesign:DataGridTextColumn Binding="{Binding Value.Count}" Header="Количество" Width="\*" Visibility="Collapsed"/>

<materialDesign:DataGridTextColumn Binding="{Binding Value.Total, StringFormat='c', ConverterCulture=ru-RU}" Header="Итого" Width="\*"/>

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

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button x:Name="BtnDeleteItem" Content="{materialDesign:PackIcon Kind=Delete}" Width="60" DataContext="{Binding Key}" Click="BtnDeleteItem\_Click"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

<StackPanel Grid.Row="10" Grid.ColumnSpan="2" HorizontalAlignment="Center" Orientation="Vertical" Width="40" Margin="130 0 0 0">

<Button x:Name="btnAddService" Padding="0" Click="btnAddService\_Click" >

<materialDesign:PackIcon Kind="Add"/>

</Button>

</StackPanel>

<StackPanel Orientation="Horizontal" Grid.Row="12" Grid.ColumnSpan="2" Margin="10" HorizontalAlignment="Center">

<Button x:Name="btnCancel" Content="Отменить заказ" HorizontalAlignment="Center" VerticalAlignment="Bottom" Width="auto" Click="BtnCancel\_Click" />

<Button x:Name="btnSave" Content="Сохранить" HorizontalAlignment="Center" VerticalAlignment="Bottom" Width="120" Click="btnSave\_Click" />

<Button x:Name="btnExcel" Content="Квитанция" HorizontalAlignment="Center" VerticalAlignment="Bottom" Width="120" Click="btnExcel\_Click\_1" />

</StackPanel>

<materialDesign:DialogHost Grid.Row="8" Grid.ColumnSpan="2" x:Name="AddServiceHost"

BorderThickness="1"

HorizontalAlignment="Center"

VerticalAlignment="Center" MaxHeight="800" MinWidth="600">

<materialDesign:DialogHost.DialogContent>

<StackPanel Margin="20" >

<TextBlock Text="Прайс-лист" FontSize="16"/>

<StackPanel Orientation="Vertical" Margin="2" Height="100">

<ComboBox x:Name="CmbCategory" FontSize="16" SelectedItem="{Binding Category}" SelectedValuePath="Id" DisplayMemberPath="Name"

VerticalAlignment="Stretch" materialDesign:HintAssist.Hint="Выберите вид услуг"

VerticalContentAlignment="Center" Height="30"

Width="300" SelectedIndex="0" Padding="10,0,0,0" SelectionChanged="CmbCategory\_SelectionChanged"/>

</StackPanel>

<Rectangle Width="Auto" Height="10"/>

<ListBox x:Name="LbPriceList" ScrollViewer.HorizontalScrollBarVisibility="Auto"

SelectedValuePath="Id" Height="400" IsTextSearchCaseSensitive="True" Width="500">

<ListBox.ItemTemplate>

<DataTemplate>

<StackPanel>

<Image Width="60" Height="60" Source="{Binding Path=GetPhoto}" />

<StackPanel Margin="10 0 0 0">

<TextBlock FontSize="14" Text="{Binding Path=Name, StringFormat={}Услуга: {0}}" Width="350" HorizontalAlignment="Left" FontWeight="Bold" />

<TextBlock FontSize="14" Text="{Binding Path=Price, StringFormat={}цена за шт.: {0:F2} руб.}" Width="300" HorizontalAlignment="Left" />

</StackPanel>

</StackPanel>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

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

<Button x:Name="btnServiceOK" Content="Ok" Click="btnServiceOK\_Click"/>

<Button x:Name="btnServiceCancel" Content="Cancel" Click="btnServiceCancel\_Click"/>

</StackPanel>

</StackPanel>

</materialDesign:DialogHost.DialogContent>

</materialDesign:DialogHost>

<materialDesign:DialogHost Grid.Row="8" Grid.ColumnSpan="2" x:Name="hostLoadClient"

BorderThickness="1"

HorizontalAlignment="Center"

VerticalAlignment="Center" >

<materialDesign:DialogHost.DialogContent>

<StackPanel Margin="20" >

<ListBox x:Name="lbClient" Grid.Row="2" ScrollViewer.HorizontalScrollBarVisibility="Auto" SelectedValuePath="AirID" Height="600">

<ListBox.ItemTemplate>

<DataTemplate>

<StackPanel>

<TextBlock FontSize="18" Text="{Binding Path=UserName}" HorizontalAlignment="Left" />

<TextBlock FontSize="18" Text="{Binding Path=LastName}" HorizontalAlignment="Left" />

<TextBlock FontSize="18" Text="{Binding Path=FirstName}" HorizontalAlignment="Left" />

<TextBlock FontSize="14" Text="{Binding Path=Phone}" HorizontalAlignment="Left" TextWrapping="WrapWithOverflow" Width="500" />

</StackPanel>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

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

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

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

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

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

</StackPanel>

</Button>

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

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

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

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

</StackPanel>

</Button>

</StackPanel>

</StackPanel>

</materialDesign:DialogHost.DialogContent>

</materialDesign:DialogHost>

</Grid>

</Page>

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.Data.Entity;

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;

using Excel = Microsoft.Office.Interop.Excel;

namespace WpfAssortmentCheck.Windows

{

/// <summary>

/// Логика взаимодействия для AddNewOrderPage.xaml

/// </summary>

public partial class AddNewOrderPage : Page

{ //текущий товар

private Order \_currentItem = new Order();

public struct BuyItem

{ public int Count { get; set; }

public double Total { get; set; }}

//public string GetPhoto

//{

// get

// {

// if (Photo is null)

// return null;

// return System.IO.Directory.GetCurrentDirectory() + @"\Images\" + Photo.Trim();

// }

//}

//public int Id { get; set; }

//public string Name { get; set; }

//public double Price { get; set; }

//public int CategoryId { get; set; }

//public string Photo { get; set; }

public static Dictionary<Service, BuyItem> buyServices = new Dictionary<Service, BuyItem>();

string \_userName = "";

public AddNewOrderPage(Order selected)

{ InitializeComponent();

buyServices.Clear();

\_currentItem = selected;

CmbCategory.ItemsSource = AutoTuningBDEntities.GetContext().Categories.ToList();

CmbCategory.SelectedIndex = -1;

ComBoBrand.ItemsSource = AutoTuningBDEntities.GetContext().Brands.ToList();

ComboStatus.ItemsSource = AutoTuningBDEntities.GetContext().Status.ToList();

lbClient.ItemsSource = AutoTuningBDEntities.GetContext().Users.ToList();

PreparingData();

//ComboStatus.SelectedIndex = 0;

LoadData();}

void PreparingData()

{ // если это окно открыл пользователь в режиме добавления новой записи

if ((Manager.CurrentUser.Role != true) &&(\_currentItem == null))

{ \_userName = Manager.CurrentUser.UserName;

ComboStatus.SelectedIndex = 1;

btnLoadClient.Visibility = Visibility.Hidden;

ComboStatus.Visibility = Visibility.Collapsed;

tbClient.Text = Manager.CurrentUser.GetInfo;

\_currentItem = new Order();

\_currentItem.DateStart = DateTime.Today;

\_currentItem.DateEnd = DateTime.Today;

btnExcel.Visibility = Visibility.Collapsed;

\_currentItem.StatusId = 1;

\_currentItem.Username = Manager.CurrentUser.UserName;

btnCancel.Visibility = Visibility.Collapsed;

DataContext = \_currentItem;

ComboStatus.SelectedIndex = 0;

return;}

// если это окно открыл пользователь в режиме просмотра записи

if ((Manager.CurrentUser.Role != true) && (\_currentItem != null))

{ MessageBox.Show("2");

\_userName = Manager.CurrentUser.UserName;

btnLoadClient.Visibility = Visibility.Hidden;

ComboStatus.IsEnabled = false;

tbClient.Text = Manager.CurrentUser.GetInfo;

btnExcel.Visibility = Visibility.Visible;

DtOrderPriceList.Columns[5].Visibility = Visibility.Collapsed;

DtOrderPriceList.Columns[6].Visibility = Visibility.Visible;

//DtOrderPriceList.Columns[7].Visibility = Visibility.Collapsed;

DtOrderPriceList.Columns[8].Visibility = Visibility.Collapsed;

DtOrderPriceList.IsReadOnly = true;

btnSave.Visibility = Visibility.Collapsed;

if (\_currentItem.StatusId == 1)

{ btnCancel.Visibility = Visibility.Visible;}

else

btnCancel.Visibility = Visibility.Collapsed;

DataContext = \_currentItem;

return;}

if ((Manager.CurrentUser.Role == true) && (\_currentItem == null))

{ btnLoadClient.Visibility = Visibility.Visible;

\_currentItem = new Order();

\_currentItem.DateStart = DateTime.Today;

\_currentItem.DateEnd = DateTime.Today;

btnExcel.Visibility = Visibility.Collapsed;

btnCancel.Visibility = Visibility.Collapsed;

DataContext = \_currentItem;

return;}

if ((Manager.CurrentUser.Role == true) && (\_currentItem != null))

{ btnLoadClient.Visibility = Visibility.Visible;

tbClient.Text = \_currentItem.User.GetInfo;

\_userName = \_currentItem.User.UserName;

btnExcel.Visibility = Visibility.Visible;

btnCancel.Visibility = Visibility.Visible;

DataContext = \_currentItem;

return;}}

void LoadData()

{ if (\_currentItem != null)

{ List<ServiceOrder> serviceOrders = AutoTuningBDEntities.GetContext().ServiceOrders.Where(p => p.OrderId == \_currentItem.Id).ToList();

List<Service> services = AutoTuningBDEntities.GetContext().Services.ToList();

foreach (ServiceOrder order in serviceOrders)

{ Service service = services.Where(p => p.Id == order.ServiceId).FirstOrDefault();

buyServices[service] = new BuyItem { Count = order.Count, Total = service.Price \* order.Count };}

DtOrderPriceList.ItemsSource = buyServices;

TbTotalPrice.Value = \_currentItem.TotalPrice;}

CalculateTotalPrice();

//

// DtOrderPriceList.ItemsSource = AutoTuningBDEntities.GetContext().ServiceOrders.Where(p => p.OrderId == \_currentItem.Id).ToList();}

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

{ hostLoadClient.IsOpen = true;}

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

{ try

{ if (lbClient.SelectedItems.Count > 0)

{ for (int i = 0; i < lbClient.SelectedItems.Count; i++)

{ User xair = lbClient.SelectedItems[i] as User;

if (xair != null)

{ \_userName = xair.UserName;

\_currentItem.Username = \_userName;

tbClient.Text = xair.GetInfo;}

}}

//MaterialDesignThemes.Wpf.DialogHost.Show("Запись вфыафыва");}

catch (Exception ex)

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

hostLoadClient.IsOpen = false;}

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

{ hostLoadClient.IsOpen = false;}

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

{ AddServiceHost.IsOpen = true;}

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

{ try

{ if (LbPriceList.SelectedItems.Count > 0)

{ for (int i = 0; i < LbPriceList.SelectedItems.Count; i++)

{ Service x = LbPriceList.SelectedItems[i] as Service;

if (x != null)

{ int id = x.Id;

if (buyServices.ContainsKey(x))

{ int k = buyServices[x].Count + 1;

double p = x.Price \* k;

buyServices[x] = new BuyItem { Count = k, Total = p};}

else

{ int k = 1;

double p = x.Price \* k;

buyServices[x] = new BuyItem { Count = k, Total = p };}

DtOrderPriceList.ItemsSource = null;

DtOrderPriceList.ItemsSource = buyServices;}

}}

CalculateTotalPrice();

AddServiceHost.IsOpen = false;

//MaterialDesignThemes.Wpf.DialogHost.Show("Запись вфыафыва");}

catch (Exception ex)

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

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

{ AddServiceHost.IsOpen = false;}

void CalculateTotalPrice()

{ //if (\_currentItem.Id == 0)

//{

double total = 0;

foreach (KeyValuePair<Service, BuyItem> valuePair in buyServices)

{ total += valuePair.Value.Total;}

TbTotalPrice.Value = total;

if (buyServices.Count == 0)

{ btnSave.IsEnabled = false;

btnExcel.IsEnabled = false;}

else

{ btnSave.IsEnabled = true;

btnExcel.IsEnabled = true;}

//}}

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

{ // вывод сообщения с вопросом Удалить запись?

MessageBoxResult messageBoxResult = MessageBox.Show($"Удалить товар из корзины???", "Удаление", MessageBoxButton.OKCancel, MessageBoxImage.Question);

//если пользователь нажал ОК пытаемся удалить запись

if (messageBoxResult == MessageBoxResult.OK)

{ if (DtOrderPriceList.SelectedIndex >= 0)

{ var x = (DtOrderPriceList.SelectedValue as Service);

buyServices.Remove(x);

DtOrderPriceList.ItemsSource = null;

DtOrderPriceList.ItemsSource = buyServices;}}

CalculateTotalPrice();}

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

{ PrintExcel();}

private void PrintExcel()

{ string fileName = AppDomain.CurrentDomain.BaseDirectory + "\\" + "Check" + ".xltx";

Excel.Application xlApp = new Excel.Application();

Excel.Worksheet xlSheet = new Excel.Worksheet();

try

{ //добавляем книгу

xlApp.Workbooks.Open(fileName, Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing);

//делаем временно неактивным документ

xlApp.Interactive = false;

xlApp.EnableEvents = false;

Excel.Range xlSheetRange;

//выбираем лист на котором будем работать (Лист 1)

xlSheet = (Excel.Worksheet)xlApp.Sheets[1];

//Название листа

xlSheet.Name = "Список заявок";

int row = 13;

int i = 0;

xlSheet.Cells[4, 3] = tbOrderId.Text;

xlSheet.Cells[5, 3] = $"{\_currentItem.User.LastName} {\_currentItem.User.FirstName} {\_currentItem.User.MiddleName}";

xlSheet.Cells[6, 3] = \_currentItem.User.Phone;

//ServiceName = service.ServiceName,

// DryOrderID = dryorder.DryOrderID,

// OrderID = dryorder.OrderID,

// ServiceID = service.ServiceID,

// DryOrderContent = dryorder.DryOrderContent,

// ServicePrice = dryorder.ServicePrice,

// DryOrderCount = dryorder.DryOrderCount,

// DryOrderPrice = dryorder.DryOrderPrice

if (buyServices.Count > 0)

{ foreach (KeyValuePair<Service, BuyItem> valuePair in buyServices)

{

xlSheet.Cells[row, 1] = (i + 1).ToString();

// DateTime y = Convert.ToDateTime(dtOrders.Rows[i].Cells[1].Value);

xlSheet.Cells[row, 2] = valuePair.Key.Name;

xlSheet.Cells[row, 6] = valuePair.Key.Price.ToString();

xlSheet.Cells[row, 7] = valuePair.Value.Count.ToString();

xlSheet.Cells[row, 8] = valuePair.Value.Total.ToString();

//xlSheet.Cells[row, 8] = order.PriceList.Price.ToString();

row++;

Excel.Range r = xlSheet.get\_Range("A" + row.ToString(), "H" + row.ToString());

r.Insert(Excel.XlInsertShiftDirection.xlShiftDown);

Excel.Range x = xlSheet.get\_Range("B" + row.ToString(), "E" + row.ToString());

x.Merge();

i++;}}

row--;

xlSheetRange = xlSheet.get\_Range("A9:H" + (row + 1).ToString(), Type.Missing);

xlSheetRange.Borders.LineStyle = true;

row++;

Excel.Range t = xlSheet.get\_Range("A" + row.ToString(), "G" + row.ToString());

t.Merge();

xlSheet.Cells[row, 8] = "=SUM(H9:H" + (row - 1).ToString() + ")";

xlSheet.Cells[row, 1] = "ИТОГО:";

t = xlSheet.get\_Range("A" + row.ToString());

t.Style.HorizontalAlignment = Excel.XlHAlign.xlHAlignRight;

row += 2;

xlSheet.Cells[row, 3] = $"{\_currentItem.User.LastName} {\_currentItem.User.FirstName} {\_currentItem.User.MiddleName}"; ;

row++;

xlSheet.Cells[row, 3] = DateTime.Today.ToShortDateString();

//выбираем всю область данных\*/

xlSheetRange = xlSheet.UsedRange;

//выравниваем строки и колонки по их содержимому

xlSheetRange.Columns.AutoFit();

xlSheetRange.Rows.AutoFit();}

catch (Exception ex)

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

finally

{ //Показываем ексель

xlApp.Visible = true;

xlApp.Interactive = true;

xlApp.ScreenUpdating = true;

xlApp.UserControl = true;}}

// проверка полей

private StringBuilder CheckFields()

{ StringBuilder s = new StringBuilder();

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

s.AppendLine("Выберите клиента");

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

s.AppendLine("Заполните информацию об автомобиле");

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

s.AppendLine("Укажите гос номер автомобиля");

if (ComBoBrand.SelectedIndex == -1)

s.AppendLine("Выберите бренд");

//if ((Manager.CurrentUser.Role == false) && (\_currentItem.StatusId != 1))

//{

// s.AppendLine("Выберите бренд");

//}

return s;}

private void btnSave\_Click(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)

{ AutoTuningBDEntities.GetContext().Orders.Add(\_currentItem);

try

{ \_currentItem.TotalPrice = Convert.ToDouble(TbTotalPrice.Value);

AutoTuningBDEntities.GetContext().SaveChanges();

int id = \_currentItem.Id;

\_currentItem.Username = \_userName;

tbOrderId.Text = id.ToString();

List<ServiceOrder> orderGoods = new List<ServiceOrder>();

double total = 0;

foreach (KeyValuePair<Service, BuyItem> valuePair in buyServices)

{ ServiceOrder orderGood = new ServiceOrder();

orderGood.OrderId = id;

orderGood.ServiceId = valuePair.Key.Id;

orderGood.Count = valuePair.Value.Count;

total += valuePair.Value.Total;

orderGoods.Add(orderGood);}

AutoTuningBDEntities.GetContext().ServiceOrders.AddRange(orderGoods);

AutoTuningBDEntities.GetContext().SaveChanges();

MessageBox.Show($"Ваш заказ номер {\_currentItem.Id} создан"); ;

tbOrderId.Visibility = Visibility.Visible;

btnExcel.IsEnabled = true;

btnSave.Visibility = Visibility.Collapsed;

buyServices.Clear();

PreparingData();

LoadData();

catch (Exception ex)

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

else

{ \_currentItem.TotalPrice = Convert.ToDouble(TbTotalPrice.Value);

\_currentItem.Username = \_userName;

List<ServiceOrder> delItems = AutoTuningBDEntities.GetContext().ServiceOrders.Where(p => p.OrderId == \_currentItem.Id).ToList();

AutoTuningBDEntities.GetContext().ServiceOrders.RemoveRange(delItems);

List<ServiceOrder> orderGoods = new List<ServiceOrder>();

double total = 0;

foreach (KeyValuePair<Service, BuyItem> valuePair in buyServices)

{ ServiceOrder orderGood = new ServiceOrder();

orderGood.OrderId = \_currentItem.Id;

orderGood.ServiceId = valuePair.Key.Id;

orderGood.Count = valuePair.Value.Count;

total += valuePair.Value.Total;

orderGoods.Add(orderGood);}

AutoTuningBDEntities.GetContext().ServiceOrders.AddRange(orderGoods);

AutoTuningBDEntities.GetContext().SaveChanges();

//AutoTuningBDEntities.GetContext().SaveChanges();

MessageBox.Show("Запись Изменена");

PreparingData();

LoadData();}}

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

{ PrintExcel();}

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

{ if (CmbCategory.SelectedIndex != -1)

{ Category category = (CmbCategory.SelectedItem) as Category;

LbPriceList.ItemsSource = AutoTuningBDEntities.GetContext().Services.Where(p => p.CategoryId == category.Id).ToList();}

else

{ LbPriceList.ItemsSource = null;}}

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

{ var x = (sender as Button).DataContext as Service;

if (buyServices.ContainsKey(x))

{ int k = buyServices[x].Count + 1;

double p = x.Price \* k;

buyServices[x] = new BuyItem { Count = k, Total = p};

DtOrderPriceList.ItemsSource = null;

DtOrderPriceList.ItemsSource = buyServices;}

CalculateTotalPrice();}

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

{ var x = (sender as Button).DataContext as Service;

if (buyServices.ContainsKey(x))

{ int k = buyServices[x].Count;

if (k > 0) k--;

if (k == 0)

{ MessageBoxResult messageBoxResult = MessageBox.Show($"Удалить услугу?", "Удаление", MessageBoxButton.OKCancel, MessageBoxImage.Question);

//если пользователь нажал ОК пытаемся удалить запись

if (messageBoxResult == MessageBoxResult.OK)

{ buyServices.Remove(x);

DtOrderPriceList.ItemsSource = null;

DtOrderPriceList.ItemsSource = buyServices;}

else

{ k = 1;

double p = x.Price \* k;

buyServices[x] = new BuyItem { Count = k, Total = p };

DtOrderPriceList.ItemsSource = null;

DtOrderPriceList.ItemsSource = buyServices;}}

else

{ double p = x.Price \* k;

buyServices[x] = new BuyItem { Count = k, Total = p };

DtOrderPriceList.ItemsSource = null;

DtOrderPriceList.ItemsSource = buyServices;}}

CalculateTotalPrice();}

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

{ // if ((Manager.CurrentUser.Role == false) && ())

var x = (sender as Button).DataContext as Service;

MessageBoxResult messageBoxResult = MessageBox.Show($"Удалить услугу?", "Удаление", MessageBoxButton.OKCancel, MessageBoxImage.Question);

if (messageBoxResult == MessageBoxResult.OK)

{ buyServices.Remove(x);

DtOrderPriceList.ItemsSource = null;

DtOrderPriceList.ItemsSource = buyServices;}

CalculateTotalPrice();}

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

{}

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

{ MessageBoxResult messageBoxResult = MessageBox.Show($"Отменить заявку?", "Отмена", MessageBoxButton.OKCancel, MessageBoxImage.Question);

if (messageBoxResult == MessageBoxResult.OK)

{ List<ServiceOrder> delItems = AutoTuningBDEntities.GetContext().ServiceOrders.Where(p => p.OrderId == \_currentItem.Id).ToList();

AutoTuningBDEntities.GetContext().ServiceOrders.RemoveRange(delItems);

AutoTuningBDEntities.GetContext().Orders.Remove(\_currentItem);

AutoTuningBDEntities.GetContext().SaveChanges();

MessageBox.Show("Заявка отменена");

Manager.MainFrame.GoBack();}

}}}

}}

**Приложение Б**

«Автоматизированная инфомационная система «Студия автозвука 12 Вольт»»

Спецификация

460.ЗМК.00019-01 01

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Обозначение** | | |  | | --- | | **Наименование** | | |  | | --- | | **Примечание** | |
|  | |  | | --- | | Документация | |  |
| 460.ЗМК.00013-01 12 01 | «Автоматизированная инфомационная система «Студия автозвука 12 Вольт»» |  |
|  | Текст программы |  |
| 460.ЗМК.00013-01 13 01 | «Автоматизированная инфомационная система «Студия автозвука 12 Вольт»» |  |
|  | Описание программы |  |
| 460.ЗМК.00013-01 34 01 | «Автоматизированная инфомационная система «Студия автозвука 12 Вольт»» |  |
|  | Руководство пользователя |  |
| 460.ЗМК.00013-01 81 01 | «Автоматизированная инфомационная система «Студия автозвука 12 Вольт»» |  |
|  | Пояснительная записка |  |

**Приложение B**

**Диск с программным средством**