Министерство образования Московской области

Государственное образовательное учреждение высшего образования Московской области

«Государственный гуманитарно-технологический университет»

**Ликино-Дулевский политехнический колледж – филиал ГГТУ**

**Отчет по учебной практике**

Разработка программного модуля «Ювелирная мастерская»

МДК 01.01 «Разработка программных модулей»

**Выполнила:**

Суханова Екатерина Дмитриевна\_\_\_\_\_\_\_\_\_\_

студент группы ИСП.20А\_\_\_\_\_\_\_\_\_\_\_\_\_

09.02.07 Информационные системы и программирование) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

очной формы обучения\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Руководитель:**

Кузьмина Елена Евгеньевна­­­­­­­­­­­­­­­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка

\_\_\_\_\_\_\_\_\_/\_\_\_­­­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

Подпись руководителя

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ликино-Дулево

2023 год

Оглавление

[Введение 2](#_Toc132274032)

[1. Постановка задачи 3](#_Toc132274033)

[2. Проектирование интерфейса приложения 4](#_Toc132274034)

[2.1. Правила к принципы разработки интерфейса 4](#_Toc132274035)

[2.2. Разработка макета приложения 4](#_Toc132274036)

[3. Структура хранения данных 5](#_Toc132274037)

[3.1. Проектирование структуры хранения данных 5](#_Toc132274038)

[3.2. Наполнение структуры данными 8](#_Toc132274039)

[4. Разработка технической документации 9](#_Toc132274040)

[4.1. Техническое задание 9](#_Toc132274041)

[4.2. Сообщения пользователю 14](#_Toc132274042)

[5. Тестирование программного модуля 16](#_Toc132274043)

[5.1 План тестирования 16](#_Toc132274044)

[5.2 Сценарии тестирования с результатами 17](#_Toc132274045)

[Заключение 20](#_Toc132274046)

[Список литературы: 21](#_Toc132274047)

[Приложения 22](#_Toc132274048)

[Приложение 1. Программный код приложения 22](#_Toc132274049)

# Введение

Ювелирная мастерская – это предприятие по ремонту или изготовлению драгоценных изделий, а также помещение такого предприятия.

Ювелирная мастерская выполняет любые ювелирные работы, используя при этом собственные драгоценные металлы или материалы заказчика. Клиенты могут заказать любые украшения: броши, серьги, кольца, цепочки различного плетения, браслеты, комплекты фамильных украшений с гравировкой или колье. Также мастерская осуществляет ремонт ювелирных изделий из драгоценных камней и металлов.

Ювелирные мастерские бывают замкнутые и открытого формата. В первых оставляешь заказ, уходишь, а после, забираете изделие. В открытых предприятиях позволяют остаться и наблюдать за процессом. Цена заказа формируется исходя из веса изготовленного изделия. В зависимости от сложности изделий цена работы за грамм может быть разная. Кроме этого мастерская может по желанию клиента украсить изделия камнями, имеющимися в наличии у мастерской, которые она закупает у поставщиков. Модельный ряд изделий, изготавливаемых мастерской, а также ассортимент камней постоянно расширяется. Также мастерская может добавить металл, необходимый для изготовления изделия, при нехватке его (металла) у клиента.

Ювелирные мастерские предоставляет широкий спектр услуг по следующим видам ювелирных работ, основные из них:

1. Изготовление ювелирных изделий любой сложности;
2. Ремонт ювелирных изделий;
3. Реставрация ювелирных изделий;
4. Изготовление изделий по эскизу клиента;
5. Гравировка ювелирных изделий.

Цель мастерской – получение прибыли от изготовления ювелирных изделий из материала заказчика, а также от реализации ювелирных изделий собственного производства.

1. Постановка задачи

Наименование темы разработки – «Разработка программного модуля «Ювелирная мастерская»».

Наименование работы: настольное приложение «JewelryRepair».

Программа предназначена для автоматизации ведения учёта заказов по ремонту изделий клиентов ювелирной мастерской. Приложение позволяет следующие функции для работы с записями:

1. Сортировка
2. Фильтрация
3. Поиск
4. Редактирование

Система должна эксплуатироваться сотрудниками ювелирной мастерской. Функциональным назначением программы является представление информации об изделиях (вес, состав, камни и тип изделия), клиентах (фамилия, имя, отчество и номер телефона), скидках (размер скидки), типах изделий (название типа), услугах (название услуг), заказах (дата поступления и выдачи, клиент, изделие, размер скидки, цена, услуга).

1. Проектирование интерфейса приложения
   1. Правила к принципы разработки интерфейса

Программа должна обеспечивать взаимодействие с пользователем (мастером), с помощью графического пользовательского интерфейса настольного приложения, который открывается через .ехе файл. Программа должна обеспечивать удобный и быстрый импорт данных в табличную часть из базы данных.

Входные данные программы организованы в базе данных, а именно в SQL Server. В этим данным относится не только информация, вводимая клавиатурой, но и путь к фотографиям, поэтому пользователь должен хранить на своём компьютере фотографии в папке настольного приложения «Resources». При запуске приложения, данные уже заполнены, поэтому есть возможность сразу приступить к работе, а не заниматься вводом предыдущих данных.

Выходные данные организованы в табличной форме в соответствии с столбцами в базе данных. Также присутствует возможность печать документа с имеющимися данными из таблицы «Заказы» в программе Excel.

Взаимодействие с другой программой происходит при нажатии кнопки «Excel», при выполнении этого действия происходит открытие одной из программ Microsoft Office – Excel, файл.

* 1. Разработка макета приложения

Таблица 1. «Состав технических средств и их характеристики»

|  |  |
| --- | --- |
| Процессор | Базовая частота процессора: 2.6 ГГц  Максимально поддерживаемый объем памяти: 128 ГБ  Максимальное число потоков: 12  Частота процессора: 3300 МГц  Число ядер: 4 |
| Оперативная память | Суммарный объем памяти: 32 ГБ  Тактовая частота: 3600 МГц  Пропускная способность: 28800 Мб/сек |
| Разрешение экрана | От 1600x900 |
| Размер монитора | От 19,5″ |
| Устройства ввода | Клавиатура и мышь |
| Устройства вывода | Принтер |
| Жёсткий диск | Объем накопителя: 1024 ГБ  Буферная память: 64 Мб  Потребляемая мощность: 5,6 Вт |

1. Структура хранения данных
   1. Проектирование структуры хранения данных

Разработка базы данных началась сформирования следующих диаграмм для проектирования схемы базы данных данной предметной области:

1. Use Case



Рис. 1 «Use Case диаграмма»

1. ER-диаграмма

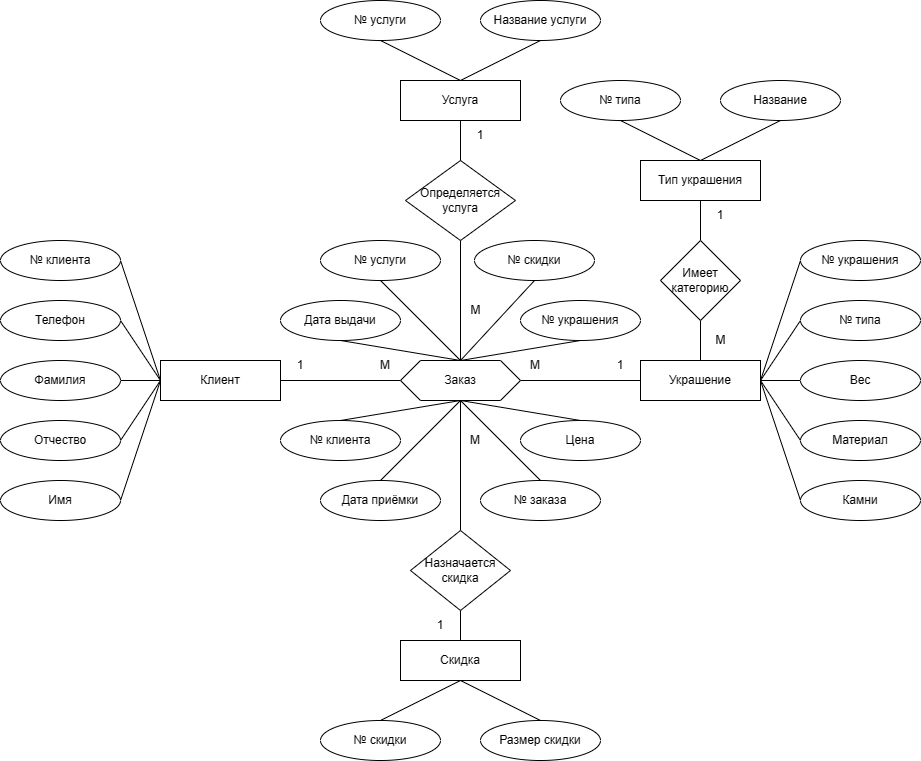


Рис. 2 «ER-диаграмма»

При помощи диаграмм получили такую схему базы данных:

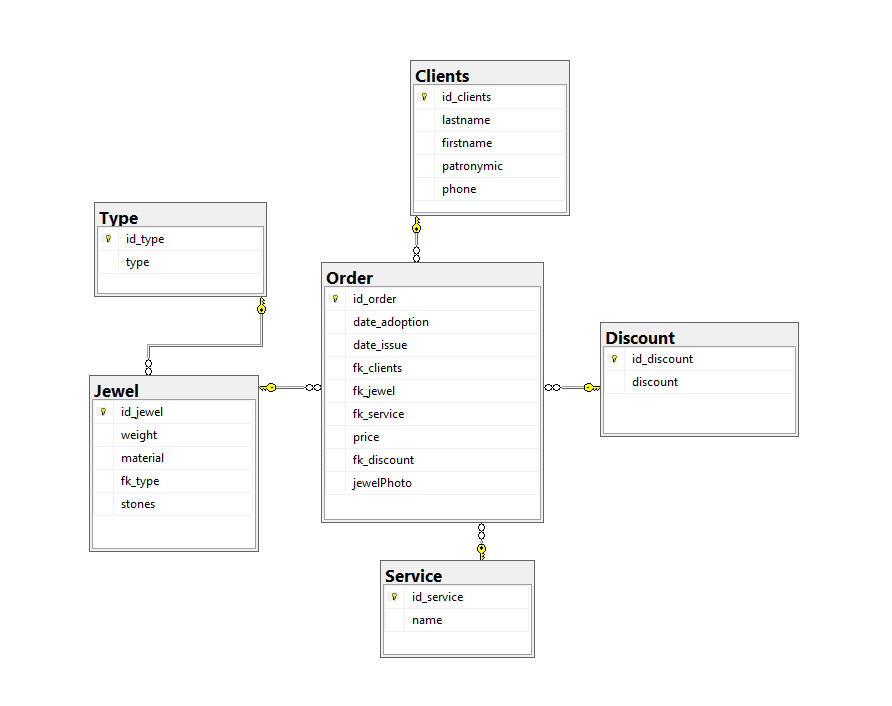


Рис. 3 «Схема базы данных в SQL Server»

Эти таблицы содержат следующие атрибуты и типы данных:

1. Type

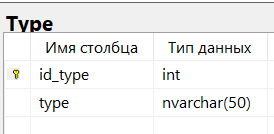


Рис. 4 «Таблица Type»

1. Order

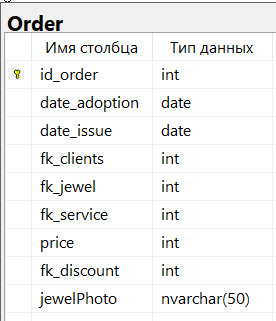


Рис. 5 «Таблица Order»

1. Discount

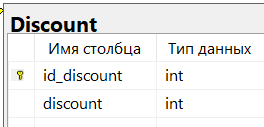


Рис. 6 «Таблица Discount»

1. Clients

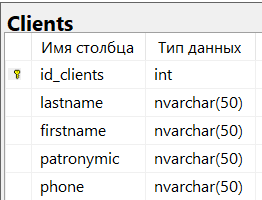


Рис. 7 «Таблица Clients»

1. Jewel



Рис. 8 «Таблица Jewel»

1. Service

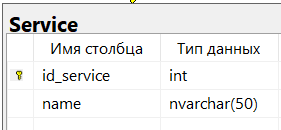


Рис. 9 «Таблица Service»

* 1. Наполнение структуры данными

Для разработки использовалась база данных SQL Server и программа Visual Studio 2022. В данной программе был создан проект «Приложение WPF (.Net Framework)» на языке C#.

Для подключения базы данных использовалась технология Entities Framework.

Проект состоит из окна FrameWindow, класса ClassFrame, и подключенной базы данных JewelryShop. Содержит следующие страницы для редактирования и добавления информации в таблицы: AddEditPageOrder, AddEditPageClients, AddEditPageDiscount, AddEditPageJewels, AddEditPageService, AddEditPageTypeJewel. Страницы для отображения данных таблиц: PageOrder, PageDiscount, PageJewels, PageTypeJewel, PageClients, PageService.

1. Разработка технической документации
   1. Техническое задание

Для начала пользователю необходимы на ПК следующие программы: Microsoft SQL Server Management Studio 18, Visual Studio 2022 и Microsoft Excel.

Для выполнения работы настольного приложения, пользователю необходимо загрузить базу данных в SQL Server через скрипт или bak файл, при этом клиент должен назвать базу данных JewelryShop. Следующим шагом является открытие двойным щелчком файла JewelryRepair.sln, после загрузки откроется программа. Чтобы её запустить следует нажать кнопку «Пуск» на панели инструментов.

При открытии настольного приложения, пользователя приветствует главный экран, где требуется выбрать кнопку, которая направит клиента на страницу с таблицей, название которой совпадает с описанием кнопки.

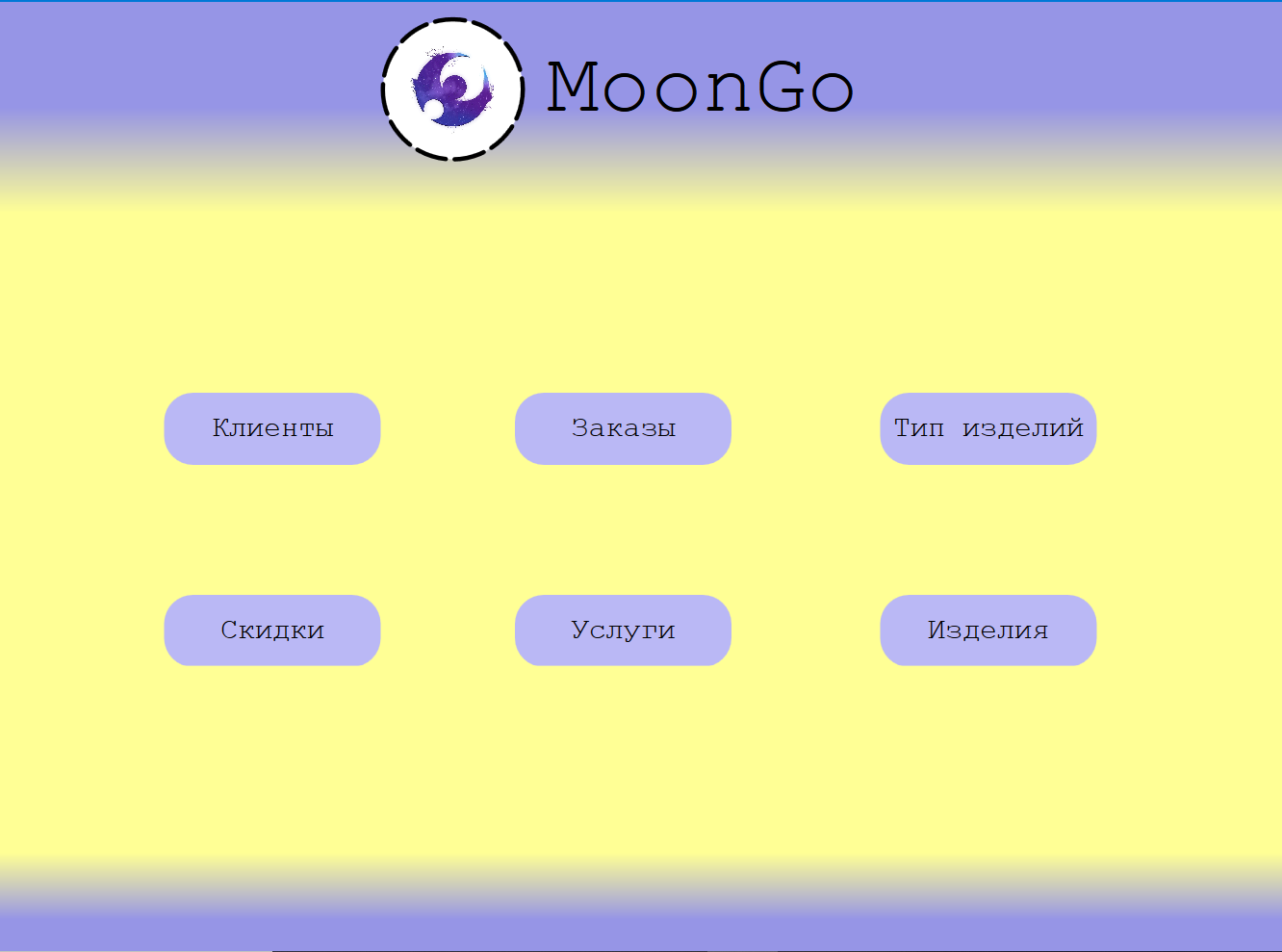


Рис. 10 «Главное окно»

После нажатия на одну из кнопок пользователь попадает на страницу, на которой есть таблица с заполненными данными, кнопки для перехода к другим таблицам и возвращения на предыдущую страницу, элементы управления для взаимодействия с информацией, которые находятся на правой части страницы, а также кнопки снизу для добавления и удаления информации из таблицы. В качестве примера рассмотрим одну из страниц со всеми функциями.



Рис. 11 «Окно с таблицей»

С помощью комбинированного списка пользователь может фильтровать информацию по какому-либо критерию. Чтобы использовать данный элемент управления следует нажать на него, а затем выбрать элемент. В качестве примера здесь используется комбинированный список по услуге.



Рис. 12 «Вызов комбинированного списка»

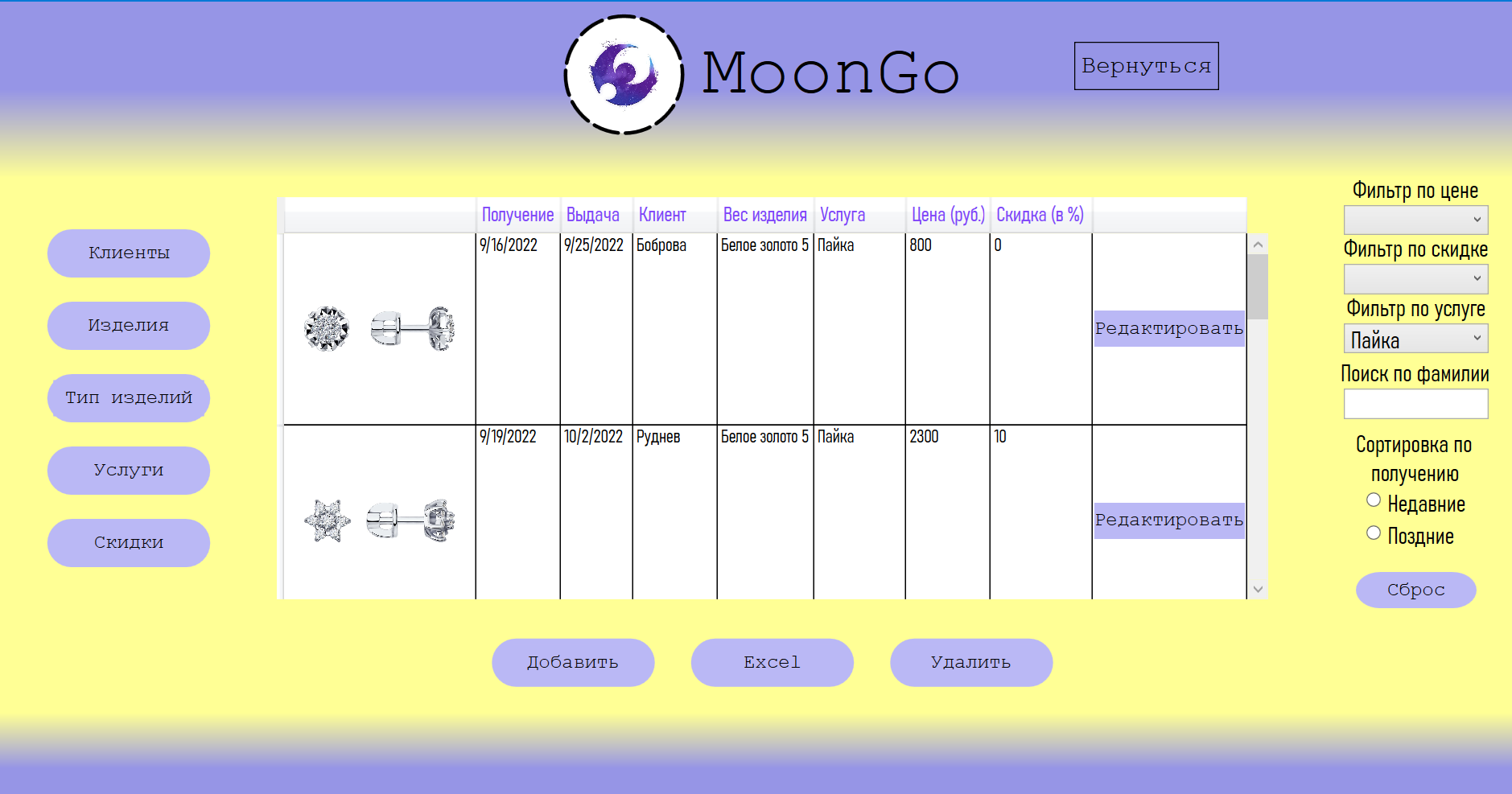


Рис. 13 «Результат использования комбинированного списка»

В приложении присутствует поиск для нахождения данных по атрибуту таблицы. Чтобы использовать поиск нужно ввести буквы, которые присутствуют в записи одного из атрибутов.



Рис. 14 «Использование поисковой строки»

Чтобы сортировать записи по атрибуту используются радиокнопки, которые выводят в начале записи, соответствующие условию.



Рис. 15 «Результат сортировки»

Чтобы убрать результат операций, пользователю нужно нажать на кнопку «Сброс». Для удаления информации из таблицы, клиент должен сперва выделить нужные записи, а затем нажать на кнопку «Удалить». Если следует изменить данные, то для этого требуется нажать «Редактировать». Впоследствии пользователь окажется на странице для редактирования данных.



Рис. 16 «Страница редактирования»

При необходимости добавления информации клиенту следует нажать на кнопку «Добавить». Затем появится страница для добавления с пустыми значениями, которые потребуется заполнить.



Рис. 17 «Окно добавления информации»

Чтобы получить информацию из таблицы в программе Excel в виде документа, пользователь должен воспользоваться кнопкой «Excel». Затем клиент попадет уже на лист этой программы.

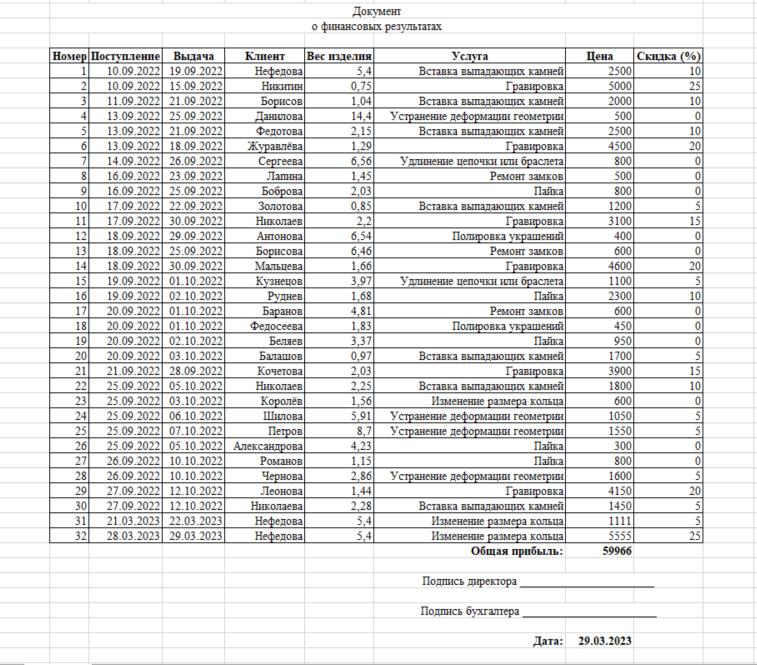


Рис. 18 «Печать в Excel»

Для завершения работы приложения нужно нажать на символ крестика, который находится в верхнем правом углу окна.

* 1. Сообщения пользователю

Если пользователь не пропускает при добавлении записи какое-либо поле, то при нажатии на кнопку «Сохранить» появляется окно с описанием того, какую информацию нужно заполнить.

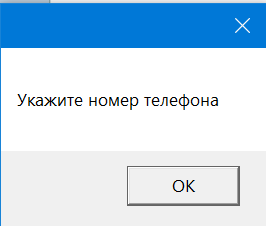


Рис. 19 «Окно с напоминанием о том, что нужно заполнить ещё поле»

Когда при нажатии на кнопку «Сохранить» пользователь ввёл отрицательное значение скидки, появляется окно с предупреждением о том, что скидка не может быть отрицательной.

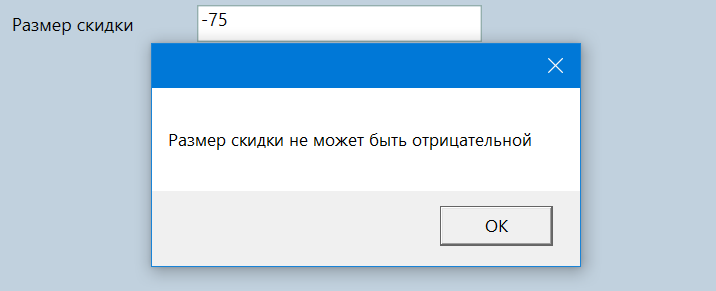


Рис. 20 «Окно с предупреждением о неправильном значении»

При сохранении новой записи в любой таблице появляется окно с уведомлением, что новая запись добавлена.

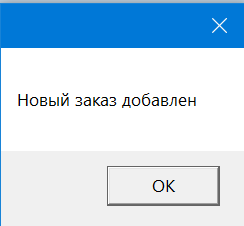


Рис. 21 «Окно с уведомлением о добавлении новой записи»

Когда пользователь собирается удалить запись, появляется предупреждающее окно.

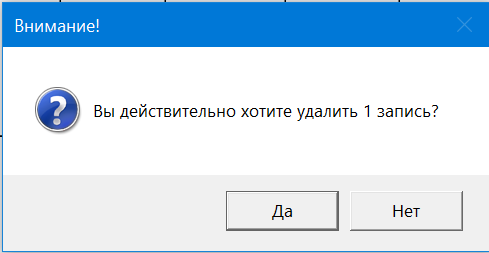


Рис. 22 «Предупреждающее окно»

При подтверждении об удалении, снова отображается окно о выполненной операции.

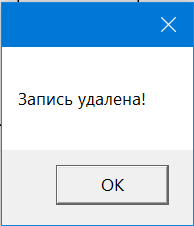


Рис. 23 «Окно при удалении записи»

При добавлении данных о номере телефона пользователь должен строго ввести 11 цифр, иначе появится предупреждающее сообщение.

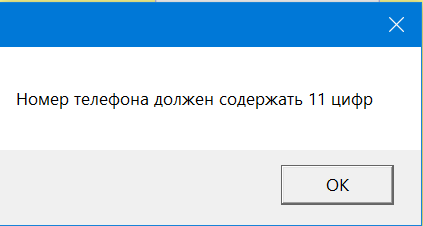


Рис. 24 «Окно, если номер телефона содержит не 11 цифр при вводе»

4.2.1

* + 1. Выбор средств разработки
    2. Технологии доступа к данным
    3. Программные методы разработки интерфейса
    4. Реализация функционала приложения
    5. Обработки исключений
  1. Руководство пользователя

1. Тестирование программного модуля
   1. План тестирования

Чтобы реализовать модульное тестирование были созданы библиотека классов и проект модульного теста. Для модульного теста для взяты атрибуты номер телефона, имя клиента и вес изделия. Для атрибутов проверяются следующие показатели:

1. Номер телефона должен содержать 11 символов

public static int GetNumberStrength(string phone)

{

int result = 0;

if (phone.Length == 11)

{

result++;

}

return result;

}

В качестве образцов возьмем следующие номера телефонов: 89345671223, 8934567122. В данной ситуации тест пройдет только первый номер телефона.

1. В номере телефона не должно быть букв

public static int GetNumberLetter(string phone)

{

int result = 0;

if (Regex.Match(phone, "[a-z]").Success)

{

result++;

}

return result;

}

В качестве примера возьмем номер телефон, содержащий букву: 8934567234s. Этот номер телефона прошел тест, тем самым отобразив неправильный ввод.

1. Имя клиента не должно содержать цифры

public static int GetNameNumber(string name)

{

int result = 0;

if (Regex.Match(name, "[0-9]").Success)

{

result++;

}

return result;

}

Здесь мы проверим имя клиента, в котором содержится цифра: Michae1. Он покажет, что данный является невереным.

1. Вес не должен быть отрицательным

public static int GetNumberMinus(double weight)

{

int result = 0;

if (weight>0)

{

result++;

}

return result;

}

В данной ситуации мы будем проверять отрицательное значение веса: -2.6. В результате мы получим значение «0», что тест означает прошел успешно.

Подводя итоги тестов можно сказать что результаты положительный статус проведения.

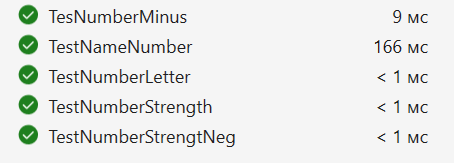


Рис. 25 «Результаты модульного тестирования»

* 1. Сценарии тестирования с результатами

Таблица 2 «Аннотация теста»

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| Название проекта | JewelryRepair |
| Рабочая версия | 1.0 |
| Имя тестирующего | Суханова Екатерина Дмитриевна |
| Дата(ы) теста | 29.03.2023 |

Таблица 3 «Тестовый пример №1»

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| Тестовый пример # | TC\_SStr\_1 |
| Приоритет тестирования  (Низкий/Средний/Высокий) | Низкий |
| Заголовок/название теста | Ввод в поисковую строку цифр с буквами. |
| Краткое изложение теста | Если при вводе в поисковую строку мы вводим два символа, один из которых есть в записях, а другого нет, тогда должны отобразиться записи, в которых должны быть один из символов. |
| Этапы теста | В поисковую строку вводим символ, который точно есть в записи, затем, которого нет ни в одной записи (например: цифру, если поиск происходит по атрибуту фамилии). |
| Тестовые данные | В поисковую строку введём значение «п2» |
| Ожидаемый результат | Должны отобразиться в таблице две записи, где в фамилии встречается буква п. |
| Фактический результат | В таблице отобразилось ни одной записи. |
| Предварительное условие | В таблице должна быть информация, чтобы осуществить поиск. |
| Постусловие | В таблице отобразилось ни одной записи. |
| Статус (Зачет/Незачет) | Незачет |

Таблица 4 «Тестовый пример №2»

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| Тестовый пример # | TC\_NullV\_1 |
| Приоритет тестирования  (Низкий/Средний/Высокий) | Средний |
| Заголовок/название теста | Добавить запись, не указав одну из характеристик. |
| Краткое изложение теста | Попытка добавления записи в таблице «Заказы», если не указать один из характеристик. |
| Этапы теста | Зайти на страницу добавления, указать следующие значения: дата поступления, дата выдачи, клиент, услуга, цена, вес изделия. Но оставить значение поля со скидкой оставить пустой. |
| Тестовые данные | Дата поступления: 27.03.2023. Дата выдачи: 28.03.2023. Клиент: Нефедова. Услуга: полировка украшений. Цена: 1500. Вес изделия: 3.6. Скидка: null. |
| Ожидаемый результат | Предупреждающее сообщение о незаполненном поле «Скидка». |
| Фактический результат | Сообщение о том, что скидка должна быть указана. |
| Предварительное условие | Должна быть кнопка для перехода на страницу добавления и сохранения данных, элементы управления куда будут заполняться данные. |
| Постусловие | В таблицу не добавляется запись. |
| Статус (Зачет/Незачет) | Зачет |

Таблица 5 «Тестовый пример №3»

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| Тестовый пример # | TC\_MinV\_1 |
| Приоритет тестирования  (Низкий/Средний/Высокий) | Средний |
| Заголовок/название теста | Отрицательная сумма. |
| Краткое изложение теста | Не должно осуществляться сохранение записи, в которой указана отрицательная цена заказа. |
| Этапы теста | Зайти на страницу добавления, указать следующие значения: дата поступления, дата выдачи, клиент, услуга, цена, вес изделия и скидка. Но значение для цены должно быть отрицательным. |
| Тестовые данные | Дата поступления: 27.03.2023. Дата выдачи: 28.03.2023. Клиент: Нефедова. Услуга: полировка украшений. Цена: -1500. Вес изделия: 3.6. Скидка: null. |
| Ожидаемый результат | Предупреждающее сообщение об отрицательном значении поля «Цена». |
| Фактический результат | Предупреждающее сообщение об отрицательном значении поля «Цена». |
| Предварительное условие | Должна быть кнопка для перехода на страницу добавления и сохранения данных, элементы управления куда будут заполняться данные. |
| Постусловие | Запись не была добавлена. |
| Статус (Зачет/Незачет) | Зачем |

Таблица 6 «Тестовый пример №4»

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| Тестовый пример # | TC\_OpDt\_1 |
| Приоритет тестирования  (Низкий/Средний/Высокий) | Низкий |
| Заголовок/название теста | Взаимодействие элементов управления |
| Краткое изложение теста | При последовательном использовании элементов управления, для выполнения операций над записями (фильтрация, сортировка и т.п.), предыдущая операция не должна сбрасываться. |
| Этапы теста | В комбинированном списке выбираем любое значение. Затем нажимаем на радиокнопку, выполняющую сортировку по критерию. |
| Тестовые данные | Элемент комбинированного списка: браслет. Радиокнопка: от наименьшего. |
| Ожидаемый результат | Отобразятся все записи, к которых имеется тип изделия «Браслет» и отсортированные от наименьших по весу. |
| Фактический результат | После нажатия на радиокнопку, таблица вернулась в прежнее состояние, только теперь была выполнена сортировка по весу изделия. |
| Предварительное условие | Должны быть элементы управления, позволяющие применять операции над данными в таблице. |
| Постусловие | После нажатия на радиокнопку, таблица вернулась в прежнее состояние, только теперь была выполнена сортировка по весу изделия. |
| Статус (Зачет/Незачет) | Незачет |

Таблица 7 «Тестовый пример №5»

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| Тестовый пример # | TC\_LimV\_1 |
| Приоритет тестирования  (Низкий/Средний/Высокий) | Средний |
| Заголовок/название теста | Проверка номера телефона. |
| Краткое изложение теста | При сохранении информации о номере телефона, должна быть проверка количества цифр. |
| Этапы теста | Зайти на страницу добавления клиентов, указать следующие значения: фамилия, имя, отчество, номер телефона. |
| Тестовые данные | Фамилия: Нефедова. Имя: Олеся. Отчество: Сергеевна. Номер телефона: 845623454 |
| Ожидаемый результат | Сообщение о том, что номер телефона должен содержать 11 цифр. |
| Фактический результат | Сообщение о том, что номер телефона должен содержать 11 цифр. |
| Предварительное условие | Должна быть кнопка для перехода на страницу добавления и сохранения данных, элементы управления куда будут заполняться данные. |
| Постусловие | Информация не была добавления в талицу. |
| Статус (Зачет/Незачет) | Зачет |

# Заключение

В данной курсовой работе был создан программный модуль «Ювелирная мастерская». Для достижения цели была создана база данных в SQL Server и проект в Visual Studio 2022. В результате выполнения курсовой работы по разработке приложения были построены различные представления настольного приложения, сформированы функциональная, программная архитектура и архитектура данных.

При разработке данного программного модуля была освоена технология для подключения базы данных в SQL Server к Visual Studio 2022 Entities Framework. Формирование шаблона в Excel для отображения информации при помощи языка C#. Из-за типа данных, с которым не приходилось раньше работать были изучены новые элементы управления и способы форматирования строк.

На данный момент программный модуль «Ювелирная мастерская» требует усовершенствования. Поэтому я вижу следующие решения, которые помогут улучшить программный модуль:

1. Чтобы его улучшить следует добавить печать для отдельных заказов, с целью упрощения учета заявок.
2. Добавить модуль авторизации для безопасности данных в базе.
3. Разработать больше предупреждающих сообщений пользователю при заполнении данных, потому что на данный момент выявлены не все ситуации, при которых информация может сохраниться в неверном формате.
4. Создать функцию добавления изображения в таблицу «Заказы».
5. Совместная работа элементов управления для операций над таблицей с данными.
6. Создание программы на других платформах

При написании документации для программного модуля, в которой описывается как пользоваться программным продуктом. Были сформированы требования к составу и параметрам технических средств, к информационной и программной совместимости, к функциональным характеристикам. Чтобы обеспечить уверенность в качестве приложения были выполнены модульное тестирование и тестовые сценарии. Документирование позволило рассмотреть программный модуль с точки зрения программиста, пользователя и тестировщика.

# Список литературы:

1. Дейтел П. Как программировать на Visual C# 2012; Питер - М., 2014. - **316** c.
2. Джозеф Албахари C# 5.0. Справочник. Полное описание языка; Диалектика / Вильямс - М., 2014. - **545** c.
3. Колесников А. П. Методы численного анализа, изложенные на языке формул и алгоритмическом языке C#; **Высшая школа** - Москва, 2017. - 414 c.
4. Культин Никита Борисович Основы программирования в Microsoft Visual C# 2010; БХВ-Петербург - М., 2011. - **384** c.
5. Бен-Ган, Ицик Microsoft SQL Server 2012. Основы T-SQL / Ицик Бен-Ган. - М.: Эксмо, **2016**. - **722** c.
6. Вишневский, Алексей Microsoft SQL Server. Эффективная работа / Алексей Вишневский. - М.: Питер, **2015**. - **374** c.
7. Дэвидсон, Луис Проектирование баз данных на SQL Server 2000 / Луис Дэвидсон. - М.: Бином. Лаборатория знаний, **2016**. - 662 c.
8. Тернстрем, Т. Microsoft SQL Server 2008. Разработка баз данных. Учебный курс Microsoft / Т. Тернстрем. - М.: Русская Редакция, **2017**. - **781** c.
9. Великий Фаберже. Искусство ювелиров придворной фирмы. - Москва**: Машиностроение, 2019**. - 144 c.
10. Плешев А.М. Классификация и ассортимент ювелирных изделий. Кооперативная торговля ювелирных изделий. - М.: 2008
11. [Программные решения для бизнеса](https://nationalteam.worldskills.ru/skills/programmnye-resheniya-dlya-biznesa/?ysclid=lfv9mfgh1u464821596)

# Приложения

## Приложение 1. Программный код приложения

Для привязки базы данных используется элемент проекта для создания модели ADO.NET EDM.

После привязки базы данных следует добавить следующий код для обработки полей в таблицах:

private static JewelryShopEntities \_context;

public JewelryShopEntities()

: base("name=JewelryShopEntities")

{

}

public static JewelryShopEntities GetContext() {

if (\_context == null)

\_context = new JewelryShopEntities();

return \_context;

}

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{

throw new UnintentionalCodeFirstException();

}

Класс, при помощи которого в FrameWindow будут отображаться страницы. Класс ClassFrame – это элемент, который осуществляет навигацию между составляющими проекта.

internal class ClassFrame

{

public static Frame frmObj { get; set; }

}

Страницы с отображением данных таблиц делаются по одной схеме, поэтому разберёмся на примере странице главной таблицей. Полный текст программы смотрите в приложении.

Чтобы уменьшить код с интерфейсом страницы были использованы стили. В данном коде можно просмотреть стили применимые для кнопок, комбинированных списков и таблицы. Стили ко всем остальным элементам можно увидеть в приложении.

<Page.Resources>

<Style x:Key="StyleButton">

<Setter Property="Button.Background" Value="#BAB8F5" />

<Setter Property="Button.BorderBrush" Value="#BAB8F5" />

<Setter Property="Button.FontSize" Value="16" />

<Setter Property="Button.FontFamily" Value="Courier New" />

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

<Setter Property="Button.Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<Border Background="{TemplateBinding Background}">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Setter.Value>

</Setter>

<Style.Triggers>

<Trigger Property="Button.IsMouseOver" Value="True">

<Setter Property="Button.Background" Value="#CECDF5"/>

</Trigger>

</Style.Triggers>

</Style>

<Style x:Key="StyleComboBox">

<Setter Property="ComboBox.FontSize" Value="20"/>

<Setter Property="ComboBox.FontFamily" Value="Bahnschrift Light Condensed"/>

<Setter Property="ComboBox.Margin" Value="2"/>

</Style>

<Style x:Key="DataGridStyle" TargetType="{x:Type DataGrid}">

<Setter Property="ColumnHeaderStyle" Value="{DynamicResource ColumnHeaderStyle}"/>

<Setter Property="BorderBrush" Value="Transparent"/>

<Setter Property="Background" Value="Transparent"/>

</Style>

<Style x:Key="ColumnHeaderStyle" TargetType="DataGridColumnHeader">

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

<Setter Property="Foreground" Value="#8044F9"/>

<Setter Property="FontSize" Value="17"/>

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

<Setter Property="FontFamily" Value="Bahnschrift Light Condensed"/>

</Style>

</Page.Resources>

Кнопка для перехода на другую страницу с таблицей. В ней прописывается интерфейс при помощи параметров.

<Button Name="BtnClients"

Content="Клиенты"

Style="{StaticResource StyleButton}"

Width="100"

Click="BtnClients\_Click"/>

Код .cs для осуществления перехода на другую страницу с таблицей.

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

{

ClassFrame.frmObj.Navigate(new PageClients());

}

Код .cs кнопки для возвращения на предыдущую страницу.

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

{

ClassFrame.frmObj.GoBack();

}

Код .cs чтобы выполнить переход на другую страницу для добавления данных в таблицу.

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

{

ClassFrame.frmObj.Navigate(new AddEditPageOrder(null));

}

Код интерфейса для таблицы, в которой указываются параметры для отображения информации из источника данных и интерфейс.

<DataGrid Name="DGridOrder" Grid.Row="0" Grid.Column="1"

AutoGenerateColumns="False"

IsReadOnly="True"

HorizontalAlignment="Center"

Margin="15"

Style="{DynamicResource DataGridStyle}">

Код интерфейса для отображения картинок в таблице. Для этого нужно добавить все картинки в папку проекта «Images».

<DataGridTemplateColumn Width="Auto">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Image Width="150"

Grid.Row="1"

Stretch="UniformToFill"

HorizontalAlignment="Left"

Margin="3">

<Image.Source>

<Binding Path="jewelPhoto">

<Binding.TargetNullValue>

<ImageSource>\Images\jewelry.png</ImageSource>

</Binding.TargetNullValue>

</Binding>

</Image.Source>

</Image>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

Код интерфейса для одного из столбца таблицы. В нем указывается привязка к атрибуту из таблицы в базе данных. При необходимости можно указать формат строки.

<DataGridTextColumn

Header="Получение" Width="70"

FontFamily="Bahnschrift Light Condensed"

FontSize="15"

Binding="{Binding date\_adoption, StringFormat=d}"/>

Код для столбца из таблицы, если он является вторичным ключом.

<DataGridTextColumn

Header="Клиент" Width="70"

FontFamily="Bahnschrift Light Condensed"

FontSize="15"

Binding="{Binding Clients.lastname}"/>

Код интерфейса для кнопки редактирования информации в таблице.

<DataGridTemplateColumn Width="Auto">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button Name="BtnEdit"

Click="BtnEdit\_Click"

Style="{StaticResource StyleButton}"

Content="Редактировать"

Height="30"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

Код .cs для кнопки редактирования информации из таблицы.

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

{

ClassFrame.frmObj.Navigate(new AddEditPageOrder((sender as Button).DataContext as Order));

}

Код .cs для вывода данных таблицы.

DGridOrder.ItemsSource=JewelryShopEntities.GetContext().Order.ToList();

Код для элемента, который может не иметь связанного визуального элемента. А также для работы с комбинированными списками, в которых участвуют вторичные ключи.

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

{

if (Visibility == Visibility.Visible)

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

DGridOrder.ItemsSource=JewelryShopEntities.GetContext().Order.ToList(); CmbFiltrDiscount.ItemsSource=JewelryShopEntities.GetContext().Discount.ToList();

CmbFiltrDiscount.SelectedValuePath = "id\_discount";

CmbFiltrDiscount.DisplayMemberPath = "discount1";

CmbFiltrService.ItemsSource=JewelryShopEntities.GetContext().Service.ToList();

CmbFiltrService.SelectedValuePath = "id\_service";

CmbFiltrService.DisplayMemberPath = "name";

}

}

Код интерфейса для комбинированного списка по вторичному ключу.

<ComboBox Name="CmbFiltrDiscount"

Height="25"

Width="120"

FontSize="20"

FontFamily="Bahnschrift Light Condensed"

SelectionChanged="CmbFiltrDiscount\_SelectionChanged"/>

Код .cs для комбинированного списка по вторичному ключу.

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

{

int id = CmbFiltrDiscount.SelectedIndex + 1;

DGridOrder.ItemsSource = JewelryShopEntities.GetContext().Order.Where(x => x.fk\_discount == id).ToList();

}

Код интерфейса для текстового поля, который служит поисковиком для информации в таблице.

<TextBox Name="TxbSearch"

FontSize="20"

FontFamily="Bahnschrift Light Condensed"

Width="120"

Height="25"

TextChanged="TxbSearch\_TextChanged"/>

Код .cs для текстового поля, который находит записи таблицы по фамилии.

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

{

DGridOrder.ItemsSource = JewelryShopEntities.GetContext().Order.Where(x => x.Clients.lastname.ToLower().Contains(TxbSearch.Text.ToLower())).ToList();

}

Код радиокнопки на странице. Она служит для сортировки данных в таблице.

<RadioButton Content="Недавние"

Style="{StaticResource StyleRadioButton}"

Name="RbUp"

Width="85"

Checked="RbUp\_Checked"/>

Код .cs для радиокнопки, которая отбирает недавние заказы.

private void RbUp\_Checked(object sender, RoutedEventArgs e)

{

DGridOrder.ItemsSource=JewelryShopEntities.GetContext().Order.OrderBy(x => x.date\_adoption).ToList();

}

Чтобы реализовать печать в Excel, следует прописать директиву и подключить в ссылках её библиотеку.

using Excel = Microsoft.Office.Interop.Excel;

Код .cs для печати информации в Excel из таблицы в виде документа.

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

{

var app = new Excel.Application();

Excel.Workbook wb = app.Workbooks.Add();

Excel.Worksheet worksheet = app.Worksheets.Item[1];

int indexRows = 5;

worksheet.Cells[3][indexRows] = "Номер";

worksheet.Cells[4][indexRows] = "Поступление";

worksheet.Cells[5][indexRows] = "Выдача";

worksheet.Cells[6][indexRows] = "Клиент";

worksheet.Cells[7][indexRows] = "Вес изделия";

worksheet.Cells[8][indexRows] = "Услуга";

worksheet.Cells[9][indexRows] = "Цена";

var printItems = DGridOrder.Items;

Excel.Range Bold = worksheet.Range[worksheet.Cells[3][indexRows], worksheet.Cells[9][indexRows]];

Bold.Style.Font.Name = "Times New Roman";

Bold.Font.Bold = true;

Excel.Range Borders = worksheet.Range[worksheet.Cells[3][indexRows], worksheet.Cells[9][indexRows]];

Borders.Borders[Excel.XlBordersIndex.xlEdgeBottom].LineStyle =

Borders.Borders[Excel.XlBordersIndex.xlEdgeLeft].LineStyle =

Borders.Borders[Excel.XlBordersIndex.xlEdgeRight].LineStyle =

Borders.Borders[Excel.XlBordersIndex.xlEdgeTop].LineStyle =

Borders.Borders[Excel.XlBordersIndex.xlInsideHorizontal].LineStyle =

Borders.Borders[Excel.XlBordersIndex.xlInsideVertical].LineStyle = Excel.XlLineStyle.xlContinuous;

Borders.HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter;

worksheet.Columns.AutoFit();

Excel.Range DocumentRange = worksheet.Range[worksheet.Cells[3][2], worksheet.Cells[9][2]];

DocumentRange.Merge();

DocumentRange.Value = "Документ";

DocumentRange.HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter;

Excel.Range TitleRange = worksheet.Range[worksheet.Cells[3][3], worksheet.Cells[9][3]];

TitleRange.Merge();

TitleRange.Value = "о финансовых результатах";

TitleRange.HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter;

TitleRange.Style.Font.Name = "Times New Roman";

foreach (Order item in printItems)

{

worksheet.Cells[3][indexRows + 1] = indexRows-4;

worksheet.Cells[4][indexRows + 1] = item.date\_adoption;

worksheet.Cells[5][indexRows + 1] = item.date\_issue;

worksheet.Cells[5][indexRows+1].ColumnWidth = 10;

worksheet.Cells[6][indexRows + 1]=item.Clients.lastname.ToString();

worksheet.Cells[6][indexRows].ColumnWidth = 13;

worksheet.Cells[7][indexRows + 1] = item.Jewel.weight.Value;

worksheet.Cells[8][indexRows + 1] = item.Service.name.ToString();

worksheet.Cells[8][indexRows].ColumnWidth = 35;

worksheet.Cells[9][indexRows + 1] = item.price;

worksheet.Cells[9][indexRows].ColumnWidth = 11;

indexRows++;

}

Excel.Range Borders1 = worksheet.Range[worksheet.Cells[3][indexRows - printItems.Count], worksheet.Cells[9][indexRows]];

Borders1.Borders[Excel.XlBordersIndex.xlEdgeBottom].LineStyle =

Borders1.Borders[Excel.XlBordersIndex.xlEdgeLeft].LineStyle =

Borders1.Borders[Excel.XlBordersIndex.xlEdgeRight].LineStyle =

Borders1.Borders[Excel.XlBordersIndex.xlEdgeTop].LineStyle =

Borders1.Borders[Excel.XlBordersIndex.xlInsideHorizontal].LineStyle =

Borders1.Borders[Excel.XlBordersIndex.xlInsideVertical].LineStyle = Excel.XlLineStyle.xlContinuous;

worksheet.Cells[8][indexRows + 1] = "Общая прибыль:";

worksheet.Cells[8][indexRows + 1].Style.HorizontalAlignment = Excel.XlHAlign.xlHAlignRight;

worksheet.Cells[8][indexRows + 1].Font.Bold = true;

worksheet.Cells[9][indexRows + 1].Formula = $"=SUM(I{indexRows}:" + $"I{indexRows - printItems.Count + 1})";

worksheet.Cells[9][indexRows + 1].Font.Bold = true;

Excel.Range DirectorRange = worksheet.Range[worksheet.Cells[8][indexRows + 3], worksheet.Cells[10][indexRows + 3]];

DirectorRange.Merge();

DirectorRange.Value = "Подпись директора \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

DirectorRange.HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter;

Excel.Range AccountantRange = worksheet.Range[worksheet.Cells[8][indexRows + 5], worksheet.Cells[10][indexRows + 5]];

AccountantRange.Merge();

AccountantRange.Value = "Подпись бухгалтера \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

AccountantRange.HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter;

worksheet.Cells[8][indexRows + 7].Value= "Дата:";

worksheet.Cells[8][indexRows + 7].Font.Bold = true;

worksheet.Cells[8][indexRows + 7].HorizontalAlignment = Excel.XlHAlign.xlHAlignRight;

worksheet.Cells[9][indexRows + 7].Formula = $"=TODAY()";

worksheet.Cells[9][indexRows + 7].Font.Bold = true;

app.Visible = true;

}

Код .cs для удаления выделенных записей из таблицы.

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

{

var orderForRemoving = DGridOrder.SelectedItems.Cast<Order>().ToList();

if (MessageBox.Show($"Вы действительно хотите удалить orderForRemoving.Count()} запись?", "Внимание!", MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

try

{ JewelryShopEntities.GetContext().Order.RemoveRange(orderForRemoving);

JewelryShopEntities.GetContext().SaveChanges();

MessageBox.Show("Запись удалена!");

DGridOrder.ItemsSource=JewelryShopEntities.GetContext().Order.ToList();

}

catch (Exception ex)

{

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

}

}

}

Код интерфейса комбинированного списка с вторичными ключами для добавления информации в таблицу.

<ComboBox Name="CmbService"

Grid.Row="3"

Grid.Column="1"

Style="{StaticResource StyleComboBox}"

SelectedValue="{Binding fk\_service}"/>

Код .cs комбинированного списка с вторичными ключами.

CmbService.ItemsSource = JewelryShopEntities.GetContext().Service.ToList();

CmbService.SelectedValuePath = "id\_service";

CmbService.DisplayMemberPath = "name";

Код .cs для сохранения изменений или добавления информации в таблицу.

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

{

StringBuilder error = new StringBuilder();

if (\_currentOrder.date\_adoption == null)

error.AppendLine("Укажите дату поступления");

if (\_currentOrder.date\_issue == null)

error.AppendLine("Укажите дату выдачи");

if (\_currentOrder.fk\_clients == null)

error.AppendLine("Укажите фамилию клиента");

if (\_currentOrder.fk\_jewel == null)

error.AppendLine("Укажите вес изделия");

if (\_currentOrder.fk\_service == null)

error.AppendLine("Укажите услугу");

if (\_currentOrder.price == null || \_currentOrder.price < 1)

error.AppendLine("Укажите цену");

if (\_currentOrder.fk\_discount == null || \_currentOrder.fk\_discount < 0)

error.AppendLine("Укажите скидку");

if (error.Length > 0)

{

MessageBox.Show(error.ToString());

return;

}

if (\_currentOrder.id\_order == 0)

JewelryShopEntities.GetContext().Order.Add(\_currentOrder);

try

{

JewelryShopEntities.GetContext().SaveChanges();

MessageBox.Show("Новый заказ добавлен");

}

catch (Exception ex)

{

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

}