КОМИТЕТ ПО ОБРАЗОВАНИЮ ПРАВИТЕЛЬСТВА САНКТ-ПЕТЕРБУРГА

**САНКТ- ПЕТЕРБУРГСКОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ «КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ»**

Специальность 09.02.07 «Информационные системы и программирование»

Квалификация Программист

**УТВЕРЖДАЮ**

**Зам. директора по УМР** **\_\_\_\_\_\_\_\_\_\_\_\_ Н.А.Бондарчук**  **«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.**

ДИПЛОМНЫЙ ПРОЕКТ

***Пояснительная записка***

Тема: Разработка информационной системы для оконного бизнеса

Группа 483

Рецензент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нормоконтроль \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н. А. Бондарчук

Консультант по экономической части \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т. В. Львова

Руководитель дипломного проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И. А. Матысик

Выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Б.В. Яремко

Содержание

[Введение 4](#_Toc105355924)

[1. Аналитическая часть 6](#_Toc105355925)

[1.1 Цель разработки 6](#_Toc105355926)

[1.2 Исследование предметной области 6](#_Toc105355927)

[1.3 Обзор и анализ аналогичных решений 9](#_Toc105355928)

[1.4 Анализ средств разработки приложения 12](#_Toc105355929)

[1.4.1 Обзор методов решения 12](#_Toc105355930)

[1.4.2 Обоснование выбора средств создания приложения 18](#_Toc105355931)

[2. проектная часть 21](#_Toc105355932)

[2.1 Техническое задание 21](#_Toc105355933)

[2.1.1 Постановка задачи 21](#_Toc105355934)

[2.1.1 Назначение программного продукта 21](#_Toc105355935)

[2.1.3 Основания для разработки 22](#_Toc105355936)

[2.1.4 Требования к функциональным характеристикам 22](#_Toc105355937)

[2.1.5 Требования к программным и аппаратным средствам 23](#_Toc105355938)

[2.1.6 Требования к надежности 25](#_Toc105355939)

[2.1.7 Защита и сохранность данных 26](#_Toc105355940)

[2.2 Описание функционально-логической структуры программы 27](#_Toc105355941)

[2.3 Логическая модель данных 31](#_Toc105355942)

[2.4 Описание алгоритма системы 34](#_Toc105355943)

[2.4.1 Общий алгоритм работы программы 34](#_Toc105355944)

[2.4.2 Алгоритмы отдельных модулей 35](#_Toc105355945)

[2.5 Программная реализация 37](#_Toc105355946)

[2.5.1 Пользовательский интерфейс 37](#_Toc105355947)

[2.5.2 Тестирование и отладка программы 48](#_Toc105355948)

[2.6 Инструкция пользователя 56](#_Toc105355949)

[2.7 Анализ результатов разработки 74](#_Toc105355950)

[3. Экономическая часть 76](#_Toc105355951)

[3.1 Описание продукта 76](#_Toc105355952)

[3.2 Описание и расчет затрат на выполнение проекта 76](#_Toc105355953)

[3.2.1 Расчет трудоемкости 76](#_Toc105355954)

[3.2.2 Расчет расходов на заработную плату 78](#_Toc105355955)

[3.2.3 Расчет затрат на материалы 78](#_Toc105355956)

[3.2.4 Расчет амортизации 79](#_Toc105355957)

[3.2.5 Расчет электроэнергии 79](#_Toc105355958)

[3.2.6 Расчет смены затрат 79](#_Toc105355959)

[3.3 Расчет предполагаемой прибыли 79](#_Toc105355960)

[Заключение 81](#_Toc105355961)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 83](#_Toc105355962)

[Приложение А. Исходный текст программы 85](#_Toc105355963)

[Приложение Б. Результаты работы программы 139](#_Toc105355964)

# Введение

Ни одну компанию, поддерживаемую информационными технологиями, невозможно представить без использования баз данных и информационных систем. В наше время все больше маленьких и средних бизнесов разрабатывают для себя информационные системы, это помогает автоматизировать ручные процессы и получать быстрый доступ к информации, увеличивая тем самым продуктивность работы.

Использование информационных систем и баз данных становится неразделимой составляющей коммерческой деятельности современного человека и деятельности преуспевающих компаний.

В данном дипломном проекте разрабатывается система для фирмы по установке пластиковых окон, которая позволит хранить информацию о деятельности фирмы, о ее сотрудниках и клиентах. В любой момент информация о текущем состоянии дел компании может быть просмотрена на формах, либо распечатана из отчетов в БД.

Растущее число заказов фирмы делает очень актуальной проблему автоматизации ее деятельности. Также система предоставит менеджеру быстрый и доступный сбор, изменение обработку и предоставление информации, что приведет к сокращению затрат времени и упростит работу менеджера.

Автоматизированная информационная система учета заказов предназначена для оптимизации работы, руководства и управляющего персонала и играет огромное значение в увеличении продуктивности их труда.

Менеджер с помощью этой системы избавляется от выполнения рутинных задач при оформлении, поиске и сравнении заказов меж собой.

Разрабатываемое приложение закрепляет следующие профессиональные компетенции по профессиональным модулям ПМ 01, ПМ 02 и ПМ 11:

* ПК 1.1 Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.
* ПК 1.2 Разрабатывать программные модули в соответствии с техническим заданием.
* ПК 1.3 Выполнять отладку программных модулей с использованием специализированных программных средств.
* ПК 1.4 Выполнять тестирование программных модулей.
* ПК 1.5 Осуществлять рефакторинг и оптимизацию программного кода.
* ПК 2.1 Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент.
* ПК 2.4 Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения.
* ПК 11.1 Осуществлять сбор, обработку и анализ информации для проектирования баз данных.
* ПК 11.2 Проектировать базу данных на основе анализа предметной области.
* ПК 11.3 Разрабатывать объекты базы данных в соответствии с результатами анализа предметной области.
* ПК 11.4 Реализовывать базу данных в конкретной системе управления базами данных.

# Аналитическая часть

В данном разделе пояснительной записки к дипломному проекту описаны средства разработки, использованные при создании приложения, а также указаны цели разработки дипломного проекта. Здесь же проводится подробный анализ возможных средств и инструментов разработки приложения, дается их краткая характеристика. В завершении раздела проводится обоснование выбора средств программирования на основе проведенного анализа.

## 1.1 Цель разработки

В последние несколько лет все бизнесы в России столкнулись с проблемой цифровизации информации, поэтому каждой маленькой и средней компании приходиться разрабатывать информационную систему под свои нужды.

Целью проекта является создание и разработка автоматизированного рабочего места менеджера и директора компании, которое позволит вести учет заказов, их статусов, а также создание отчетов нужных для понимания вектора развития компании.

## Исследование предметной области

У каждого разрабатываемого программного продукта есть область применения. Для приложения «Разработка автоматизированной информационной системы по продаже пластиковых стеклопакетов» – это автоматизация работы управляющего персонала.

Сейчас установка пластиковых стеклопакетов – это то без чего не обходиться ни один офис, многоквартирный дом, а также здания предприятий.

В России существуют тысячи компаний по установке стеклопакетов, и если в регионах можно еще на несколько лет забыть о цифровизации бизнеса, то в крупных городах, таких как Москва, Санкт-Петербург, Новосибирск и Екатеринбург уже сейчас нужно думать об автоматизации рутинных задач и процессов. Конкуренция в этой отрасли растет с каждым днем и именно поэтому компании все чаще задумываются, как оставаться востребованными на рынке и улучшить свое положение. Чем быстрее и качественнее фирма выполняет свои заказы, тем выше имидж компании на рынке, тем привлекательнее она для клиентов и, что не менее важно все больше клиентов отдают предпочтение именно этой фирме, тем самым конкурентоспособность повышается и благосостояние компании увеличивается.

Важной частью каждой компании является ее репутация, она влияет на то обратиться ли человек в эту компанию снова или посоветует своим знакомым, тем самым повыситься число заказов фирмы. Для поднятия репутации компании нужно иметь высокое качество обслуживания клиентов, постоянные нововведения, которые обеспечивают продуктивную работу управляющего персонала, постоянный и эффективный контроль со стороны администрации, изучение и внедрение передового опыта, новой техники и технологии, расширение ассортимента и совершенствования качества предоставляемых услуг и продуктов.

Компания по продаже стеклопакетов – это имущественный комплекс (здание, часть здания, оборудование и иное имущество), предназначенное для встречи клиентов, а также изготовления стеклопакетов. Организационная структура состоит из комплекса отделов, которые условно можно разделить на две группы. Функция служб первой состоит в контактировании с клиентами и оперативном их обслуживании. Клиент приходит в офис компании и менеджер подбирает ему стеклопакеты в зависимости от нужд клиента, его типа дома и т.д. Процедура оформления заказов должна проходить быстро и без ошибок. Именно в упрощении и ускорении этого процесса и состоит основная задача автоматизированной информационной системы.

Несколько иная система имеет место в отделе, работа которого посвящена функционированию внутренних механизмов предприятия, а именно цеха, в котором разрабатываются и производятся стеклопакеты определенного размера и типа.

Одним из основных направлений формирования стратегических конкурентных преимуществ в оконном бизнесе является предоставление услуг более высокого качества по сравнению с конкурирующими аналогами. Ключевым здесь является предоставление таких услуг, которые удовлетворяли бы и даже превосходили ожидания клиентов. Ожидания клиентов формируются на основе уже имеющегося у них опыта, а также информации, получаемой по прямым или по массовым каналам маркетинговых коммуникаций. Исходя из этого, потребители выбирают производителя услуг и после их предоставления сравнивают свое представление о полученной услуге со своими ожиданиями. Если представление о предоставленной услуге не соответствует ожиданиям, клиенты теряют к фирме всякий интерес, если же услуга соответствует или превосходит их ожидания, они могут вновь обратиться к такому производителю услуг или посоветовать сервис своим знакомым, друзьям, родственникам и т.д.

Покупатель всегда стремится к определенному им соответствию цены услуги и ее качества. Интересно заметить, что, как правило, покупатель услуги реже жалуется на ее высокую цену, чем покупатель физического товара. Если он считает цену завышенной, то просто уходит без покупки. Неудовлетворенность услугой ведет, как правило, к большим потерям в доле рынка. Именно поэтому производитель услуг должен как можно точнее выявлять потребности и ожидания своих целевых клиентов.

Компания по продаже стеклопакетов рассчитана на клиентов среднего класса и выше среднего класса.

Целью создания автоматизированной информационной системы для управляющего персонала является улучшение качества обслуживания клиентов, автоматизация работы персонала и, как следствие, получение большей прибыли.

Соответственно, пользователями данного ПО являются директор фирмы и менеджеры. В данном проекте будет рассмотрена автоматизация рабочего места директора, главной задачей которого является работа с отчетами и менеджера, главной задачей которого является работа с заказами.

Определим функционал, который должен быть доступен директору фирмы с помощью программного продукта:

* формирует отчеты о работе менеджеров;
* формирует отчеты о доходах;
* добавляет менеджеров в систему;
* изменяет цены и товары.

Также определим функционал менеджера:

* просмотр своих заказов;
* оформление заказов;
* проектирование нестандартных окон;
* внесение заказов в БД;
* внесение информации о клиентах в БД;
* просмотр спроектированных окон.

С такой системой работа фирмы будет намного эффективнее, довольных клиентов гораздо больше, а значит и прибыль, которую получит компания – увеличится.

## Обзор и анализ аналогичных решений

Автоматизация — сложный и дорогостоящий процесс, к которому владельцы и управляющие компаний приходят постепенно, критично оценивая возможности отдачи от инвестиций и свой кадровый потенциал, тщательно выбирая функционал, поставщиков систем и состав оборудования. Поэтому современные решения автоматизации компаний по продаже стеклопакетов базируются на скоординированном взаимодействии нескольких специализированных систем, поставляемых в фирму профессиональными компаниями, которые могут не только качественно установить свои системы и обучить персонал, но также обеспечить хороший уровень круглосуточного технического и технологического сопровождения.

Все системы выбираются по следующим основным критериям:

* техническое решение;
* функциональные возможности;
* простота и эффективность работы;
* надежность;
* возможность развития;
* цена.

Наиболее распространенными системами для оконного бизнеса являются:

* 5CRM для оконного бизнеса;
* CRM для оконных компаний Altec.

Для наглядности рассмотрим несколько существующих систем автоматизированного управления оконным бизнесом.

**Система управления 5CRM**

5CRM-система эффективно автоматизирует и систематизирует все этапы в каждом отделе компании по продаже окон. CRM подходит как крупной, так и малой компании.  Система содержит все основные инструменты для качественного выполнения поставленных задач.

К преимуществам системы можно отнести хорошо исполненное техническое решение, большие функциональные возможности и удобство и простота в использовании.

К недостаткам системы отнесем достаточно высокую стоимость самого программного продукта, а также его настройке и сопровождению. Также немаловажным фактором является невозможность развития программного продукта под нужды компании.

Скриншот работы Системы управления 5CRM представлен на рисунке 1.

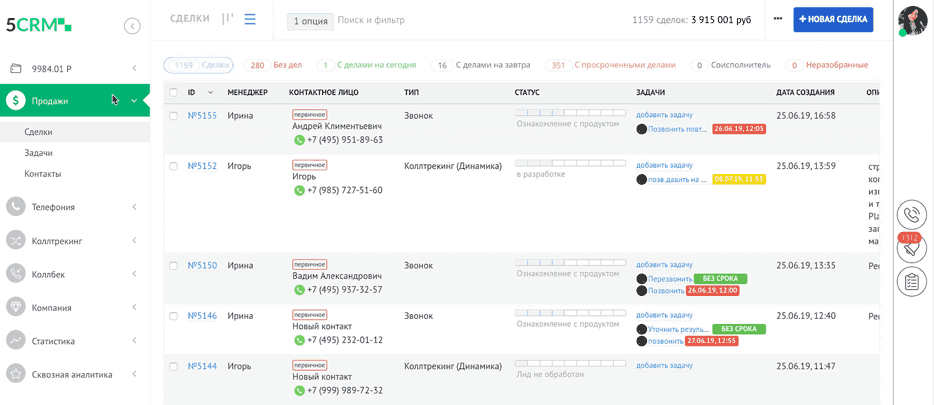


Рисунок 1 – Система управления 5CRM

**Система управления Altec**

Система для организации и контроля за взаимодействием с клиентами. Предназначена для автоматизации служб продаж и маркетинга в оконных компаниях. Выводит работу с клиентом на принципиально новый уровень, как по организации труда сотрудников, так и по возможностям контроля со стороны руководства. Тотальная фиксация всех входящих обращений и отслеживание их судьбы. Вы сможете хранить всю важную информацию о регулярности покупок, предпочитаемом ассортименте и скидках. Позволяет организовать планирование контактов с пользователем: плановый обзвон, напоминание о важных датах, календарь задач для сотрудников, ведение истории контактов с покупателями, отслеживание исполнения выполнения задач.

К недостаткам можно отнести высокую стоимость и непонятный для персонала интерфейс. Из-за неудобности использования программный продукт может только замедлить работу сотрудников, поэтому придется тратить значительное количество времени на обучение использования программы.

Скриншот работы Системы управления Altec представлен на рисунке 2.

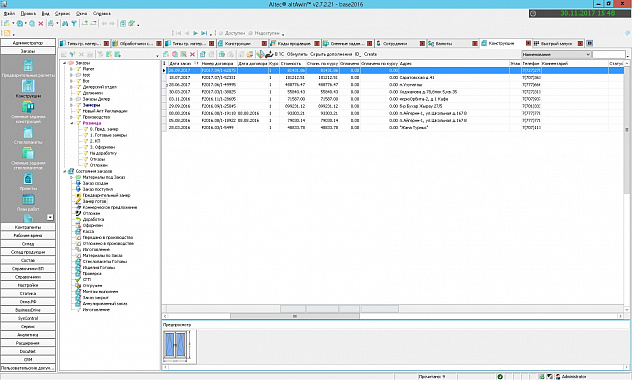


Рисунок 2 – Системы управления Altec

На основании проведенного исследования необходимо разработать приложение, которое должно обладать следующими функциональными возможностями, а именно удобный и понятный пользователю интерфейс, не требующее сложных конфигурационных настроек, иметь возможности развития программного продукта под нужды компании, иметь все необходимые функциональные возможности, а также по доступной цене.

## Анализ средств разработки приложения

В данном подразделе пояснительной записки представлен материал проведенного анализа программных средств, которыми можно разработать приложение дипломного проекта. В анализе присутствуют сравнительные характеристики программных технологий и методов, языков программирования и интегрированных сред. В результате проведенного анализа приводится обоснование выбранных технологий для разработки дипломного проекта.

### 1.4.1 Обзор методов решения

Для реализации выбранного приложения дипломного проекта потребуются следующие программные средства:

* программная платформа приложения;
* язык программирования;
* среда разработки компьютерного приложения;
* система управления базами данных.

Программных платформ для приложений существует огромное множество, но на рынке компьютерных приложений выделяют три, соответственно:

* Visual Studio;
* QtCreator;
* Eclipse IDE.

Каждая из платформ предоставляет собственные возможности для пользователя, однако, как разработчику они совершенно не интересны. Выбор платформы должен быть обоснован лишь её популярностью в разрезе области применения приложения.

Далее рассмотрим различные языки программирования, ведь именно от языка зависит и выбор среды разработки, и выбор базы данных для разрабатываемого приложения.

Рассмотрим следующие языки программирования Java и C#. Рассмотрим подробно каждый из языков, его плюсы и минусы.

**Язык программирования Java**

Java – строго типизированный объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems.

Все создаваемые программы транслируются в байт-код, который будет выполнен специальной виртуальной машиной Java (JVM). Виртуальная машина, в свою очередь, является некоторым интерпретатором. Она обрабатывает и передает инструкции оборудованию. Огромным плюсом данного выполнения программы является кроссплатформенность разрабатываемого приложения. Для запуска приложения нужно иметь устройство и виртуальную машину для него.

Второй особенностью данного языка программирования является то, что любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным), вызывают немедленное прерывание.

Ко всему прочему, язык Java активно используется для создания десктопных приложений под операционную систему Android. При этом программы компилируются в нестандартный байт-код, для использования их виртуальной машиной Dalvik (начиная с Android 5.0 виртуальная машина заменена на ART).

Для такой компиляции используется дополнительный инструмент, а именно Android SDK, разработанный компанией Google.

**Язык программирования C#**

C# разрабатывался как язык программирования прикладного уровня для CLR и, как таковой, зависит, прежде всего, от возможностей самой CLR.

Присутствие или отсутствие тех или иных выразительных особенностей языка диктуется тем, может ли конкретная языковая особенность быть транслирована в соответствующие конструкции CLR. Так, с развитием CLR от версии 1.1 к 2.0 значительно обогатился и сам C#; подобного взаимодействия следует ожидать и в дальнейшем CLR предоставляет C#, как и всем другим .NET-ориентированным языкам, многие возможности которых лишены «классические» языки программирования. Например, сборка мусора не реализована в самом C#, а производится CLR для программ, написанных на C# точно так же, как это делается для программ на VB.NET, J# и др.

Для рассматриваемых языков программирования также имеются следующие среды разработки. Для Java – это Android Studio и Eclipse» а для С# предлагается лишь одна среда разработки – это Visual Studio, так как, только она поддерживает разработку под десктоп Windows.

Рассмотрим подробно все три представленные IDE.

**IDE Android Studio**

IDE находилась в свободном доступе, начиная с версии 0.1, опубликованной в мае 2013, а затем перешла в стадию бета-тестирования, начиная с версии 0.8, которая была выпущена в июне 2014 года. Первая стабильная версия 1.0 была выпущена в декабре 2014 год.

Новые функции появляются с каждой новой версией Android Studio. На данный момент доступны следующие функции:

* расширенный редактор макетов: WYSIWYG, способность работать с UI компонентами;
* сборка приложений, основанная на Gradle;
* различные виды сборок и генерация нескольких «\*.apk» файлов;
* рефакторинг кода;
* статический анализатор кода (Lint);
* встроенный ProGuard;
* шаблоны основных макетов и компонентов Android;
* поддержка разработки приложений для Android Wear и Android TV;
* встроенная поддержка Google Cloud Platform.

Скриншот IDE Android Studio в момент работы представлен на рисунке 3.

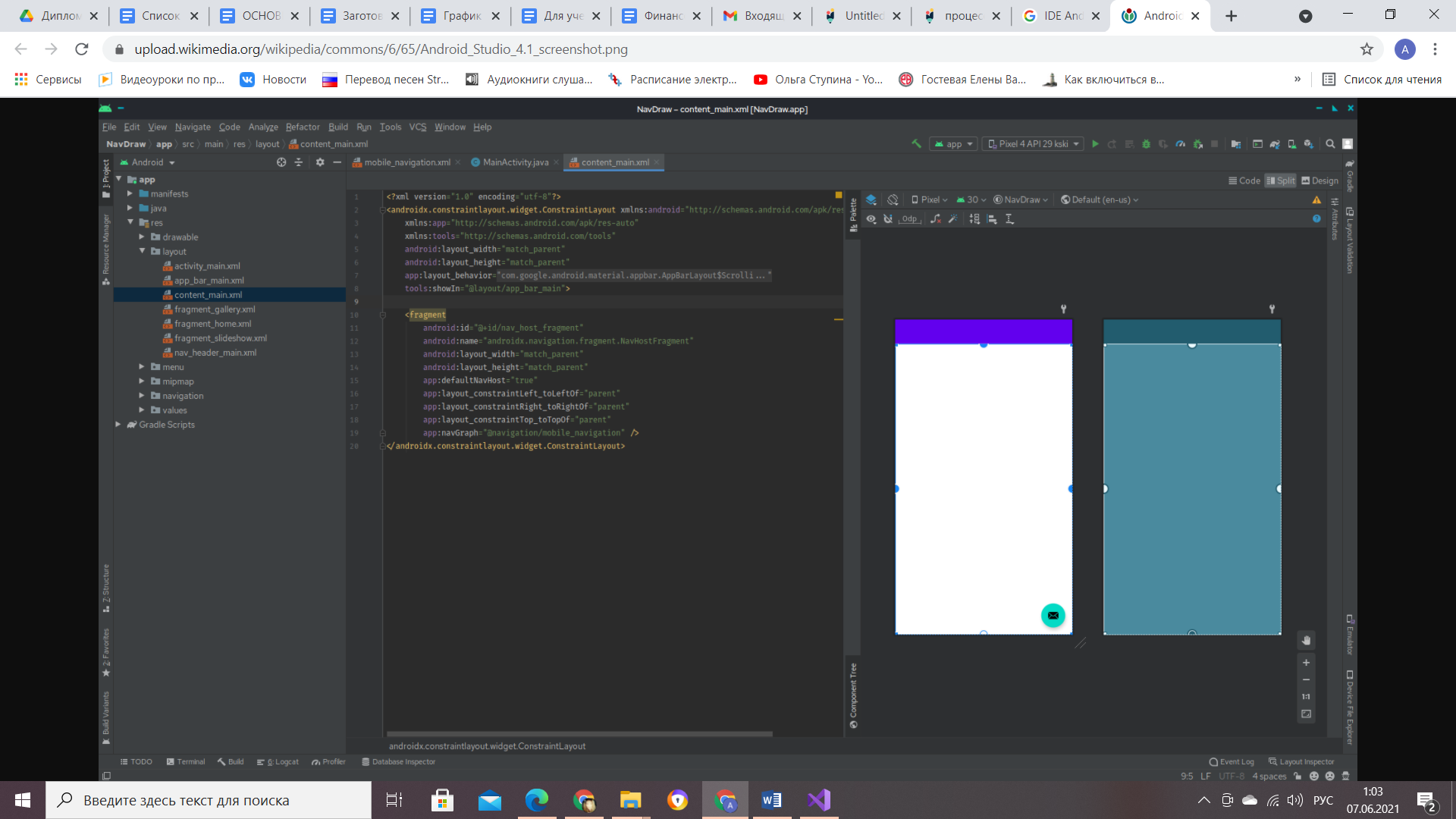


Рисунок 3 – IDE Android Studio в момент работы

**IDE Eclipse**

Eclipse – свободная интегрированная среда разработки модульных кроссплатформенных приложений. Развивается и поддерживается Eclipse Foundation.

Для среды Eclipse существует целый ряд свободных и коммерческих модулей. Первоначально среда была разработана для языка Java, но в настоящее время существуют многочисленные расширения для поддержки и других языков.

Eclipse JDT (Java Development Tools) – наиболее известный модуль, нацеленный на групповую разработку: среда интегрирована с системами управления версиями – CVS, GIT в основной поставке, для других систем (например, Subversion, MS SourceSafe) существуют плагины.

В 2014 году прекратилась поддержка плагина Android Development Tools (ADT) для Eclipse.

Скриншот IDE Eclipse в момент работы представлен на рисунке 4.

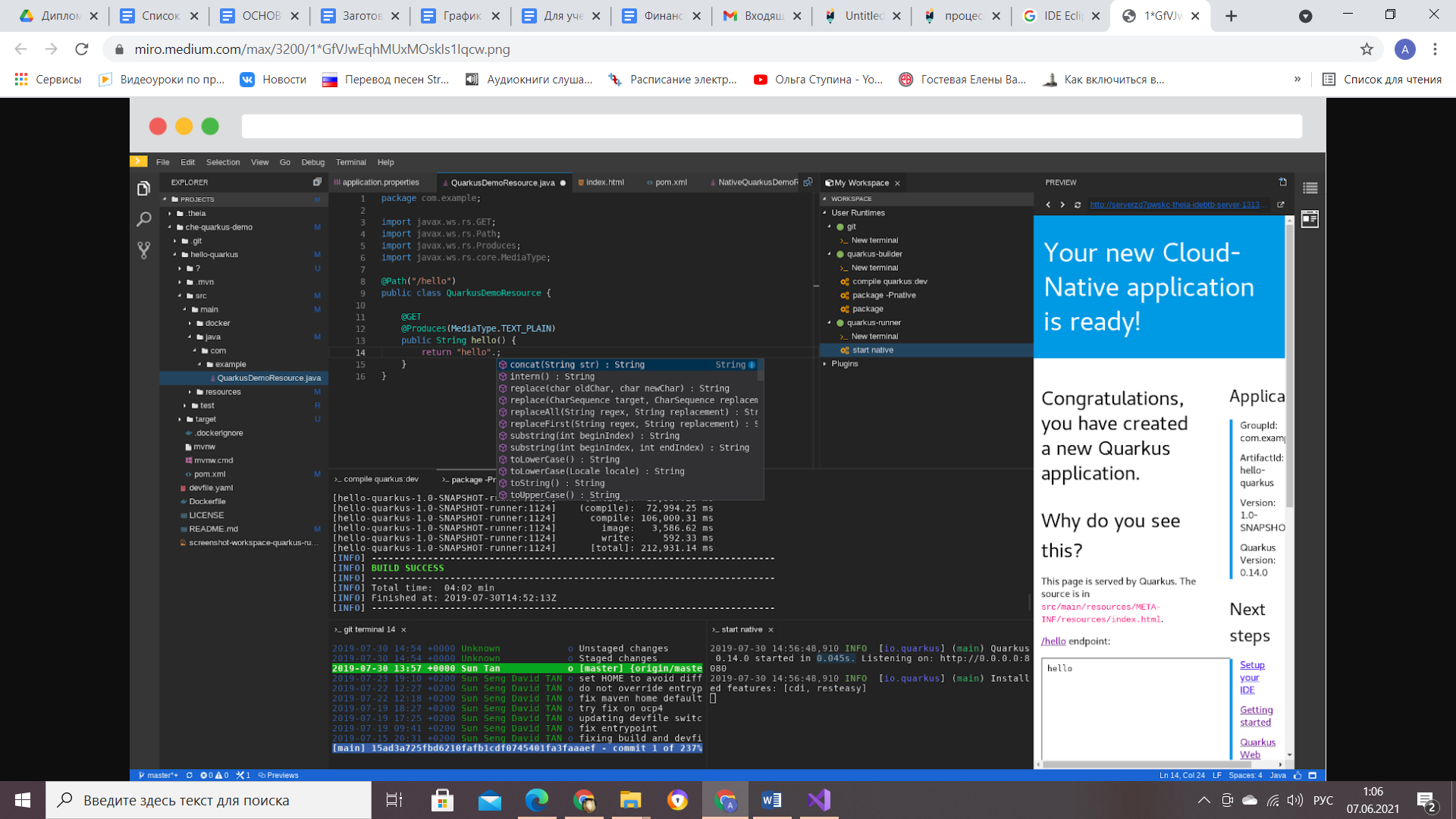


Рисунок 4 – IDE Eclipse в момент работы

**IDE Microsoft Visual Studio**

Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

После выбора языка программирования требуется выбрать БД, если в разрабатываемом приложении имеются базы данных.

Рассмотрим две системы управления базами данных, а именно:

* Microsoft Access;
* Microsoft SQL Server.

Рассмотрим каждую из представленных баз данных подробнее.

Скриншот IDE Microsoft Visual Studioв момент работы представлен на рисунке 5.

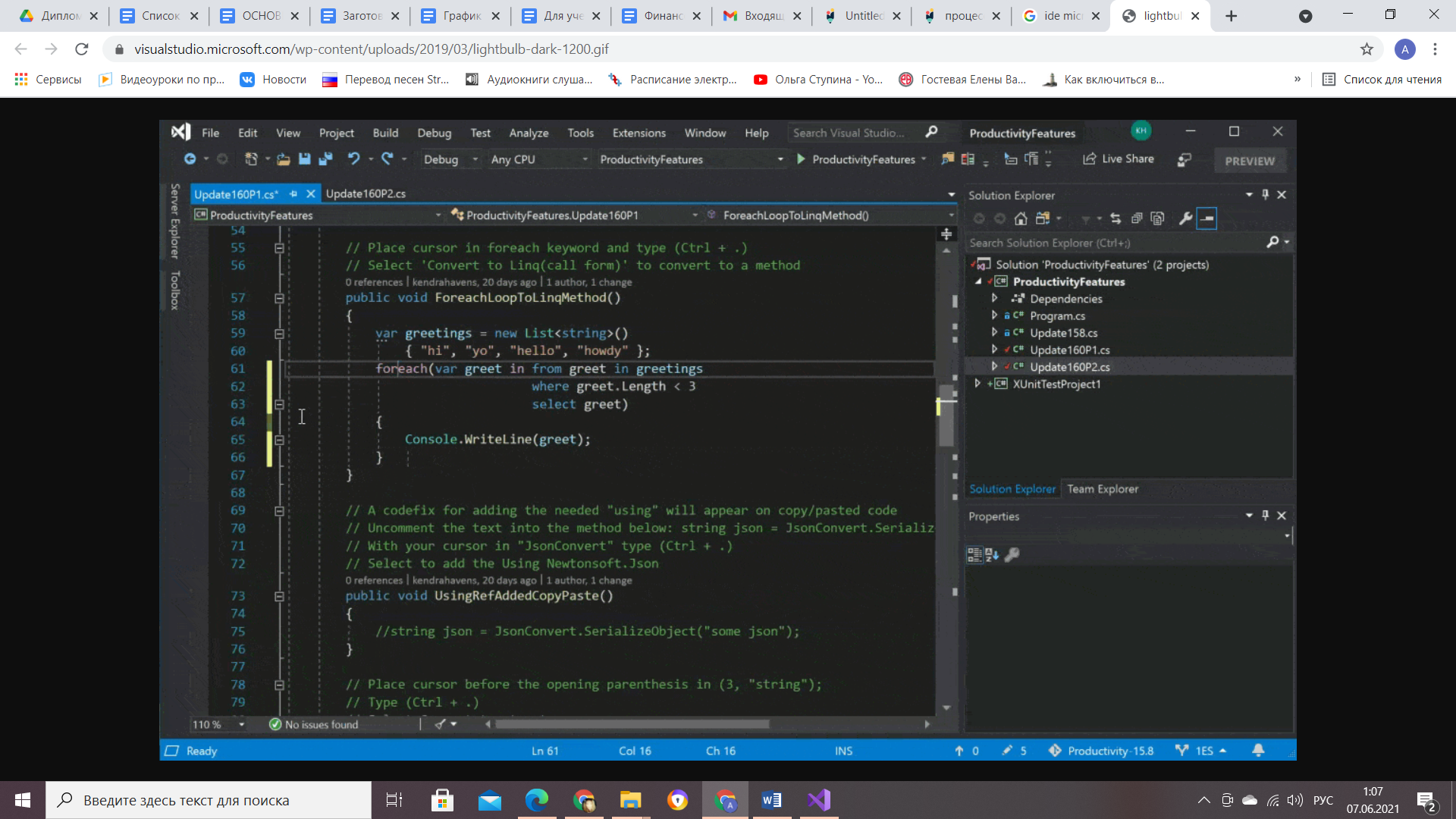


Рисунок 5 – IDE Microsoft Visual Studioв момент работы

**База данных Microsoft Access**

Microsoft Access — реляционная система управления базами данных корпорации Microsoft. Входит в состав пакета Microsoft Office.

Основные компоненты MS Access:

* конструктор таблиц;
* построитель экранных форм;
* построитель MS SQL Server -запросов (язык MS SQL Server в MS Access не соответствует стандарту Ansi);
* построитель отчётов, выводимых на печать.

Они могут вызывать скрипты на языке VBA, поэтому MS Access позволяет разрабатывать приложения и базы данных практически «с нуля» или написать оболочку для внешней базы данных.

**База данных Microsoft SQL Server**

Microsoft SQL Server является качественным инструментом для создания, разработки и редактирования баз данных. Функции базы данных:

* создание файлов баз данных;
* создание, изменение и удаление таблицы;
* создание, изменение и удаление индексов;
* поиск, редактирование, добавление и удаление записей;
* импорт и экспорт записей в виде текста;
* импорт и экспорт таблиц из / в CSV-файлы;
* импорт и экспорт базы данных из / в файлы SQL;
* исполнение SQL запросов и проверка результатов;
* журнал SQL команд, выполняемых как пользователем, так и приложением.

База данных Microsoft SQL Server обладает высоким уровнем отказоустойчивости и надежности. После первоначальной установки и настройки обычно не требует дополнительных затрат на обслуживание.

### 1.4.2 Обоснование выбора средств создания приложения

Еще раз приведем список рассматриваемых программных средств:

* программная платформа приложения (Операционная система);
* язык программирования;
* среда разработки компьютерного приложения;
* система управления базами данных.

Выбор всех программных средств напрямую зависит от той операционной системы, для которой будет написано приложение. По статистике в России за последние годы самой популярной системой является Windows.

Исходя из данных статистики, была определена операционная система разрабатываемому приложению – Windows.

Из рассматриваемых языков программирования был выбран C#. Он был выбран по следующему ряду причин:

* большие возможности программировать под Windows;
* популярность языка;
* доступность библиотек и решений;
* простота в изучении.

Как описывалось ранее, C# достаточно функционален и хорошо приспособлен для программирование компьютерных приложений. Этот язык, несомненно, пользуется большим спросом.

Программисты на C# разрабатывают приложения, игры и веб-ресурсы любой сложности на одном из самых развитых языков с использованием платформы .NET. Платформа и язык поддерживаются IT-гигантом Microsoft.

Большое количество инструментов и технологий позволяет с помощью C# решать практически любые задачи, поэтому спрос на разработчиков C# постоянно растёт. Как отмечалось выше, для выбранного нами языка С#, Visual Studio подходит лучше всего. Visual Studio представляет собой полностью интегрированную среду разработки. Она спроектирована таким образом, чтобы делать процесс написания кода, его отладки и компиляции в сборку для поставки конечным потребителям как можно более простым. На практике это означает, что Visual Studio является очень сложным приложением с многодокументным интерфейсом, в котором можно делать практически все, что касается разработки кода. Однако, выбор IDE Visual Studio был обоснован также следующими факторами:

* мощный редактор кода;
* возможность работы с Window Form и компонентами;
* рефакторинг кода;
* статический анализатор кода (в самой среде);
* наличие стандартных компонентов и основных макетов для Windows.

Таким образом, данная среда разработки не только является единственным выбором, но и самым удобным из представленных, ввиду того, что разметка графической части приложения является очень простой и удобной, а также работа с кодом не доставит разработчику особых сложностей.

В разрабатываемом приложении информация будет храниться в базе данных. Однако, выбранный язык программирования, и среда разработки ставят разработчику особые условия, так как для работы с базой данных должна быть разработана база данных, которая может хранить большие объёмы информации. Из представленных ранее, таким функционалом наделена СУБД Microsoft SQL Server. Она предоставляет абсолютно все требуемые функции и имеет интуитивно понятный интерфейс.

После проведенного исследования, из всего перечня предлагаемых программных средств и технологий был определен следующий стек технологий: операционная система (программная платформа приложения) – Windows, язык программирования – C#, среда разработки – Visual Studio 2022, система управления базами данных – Microsoft SQL Server Management Studio 17.

Хотелось бы отметить, что всё определенные средства выше, были выбраны из-за предоставляемого функционала, распространённости, доступности, простоты понимания и использования.

# проектная часть

В данном разделе технической документации к дипломному проекту описываются процессы практических мероприятий при решении поставленных целей, требования к аппаратным ресурсам для развертывания системы, описание связи с другими программами.

## 2.1 Техническое задание

В данном подразделе четко указывается постановка задачи, описываются функциональные возможности, которые поставлены перед программой, требования к программе, смысл разработки данного дипломного проекта.

### 2.1.1 Постановка задачи

Требуется разработать многофункциональное быстродействующее приложение по теме: «Разработка информационной системы для оконного бизнеса», которое позволит повысить эффективность работы менеджеров и директора компании за счет снижения трудозатрат на работу с клиентами и составления заказов.

В приложении должен присутствовать следующий функционал:

* составление заказов;
* внесение заказов в базу данных;
* проектирование нестандартных стеклопакетов;
* внесение макетов нестандартных окон в базу;
* редактирование цен и товаров;
* добавление менеджеров в систему;
* просмотр заказов;
* просмотр отчетности;
* внесение информации о клиентах в базу.

### 2.1.1 Назначение программного продукта

Разрабатываемый программный продукт предназначен для автоматизации отдельных процессов в оконном бизнесе. Менеджеры компании, различного уровня компьютерной грамотности смогут эффективно использовать приложение в своей работе, так как оно имеет дружелюбный интерфейс. Целью создания программного продукта является повышение эффективности работы персонала фирмы, за счет автоматизации ручного составления, оформления заказов и ведения БД.

### 2.1.3 Основания для разработки

Основанием для разработки данного компьютерное комплекса является задание на дипломное проектирование и задание на преддипломную практику, представленное в технической документации. Задание было выдано руководителем дипломного проекта и утверждено на заседании кафедры специальных дисциплин колледжа информационных технологий

### 2.1.4 Требования к функциональным характеристикам

Как правило, в оконных компаниях работает несколько менеджеров, тогда приложение должно персонифицировать своего пользователя. Для этого добавлен блок функций управления персоналом. В него входят: создание учетных записей менеджеров, редактирование информации по каждому пользователю, назначение паролей.

Приложение работает с БД клиентов. Для эффективной защиты и хранения данных о клиентах приложение использует СУБД Microsoft SQL Server. С помощью встроенных сервисов Microsoft SQL Server предоставляет нам возможность резервного сохранения данных или переноса информации на другое рабочее место.

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

* управление списком стеклопакетов (добавление и редактирование информации в БД о товарах);
* управление БД клиентов компании (добавление информации в БД о клиентах, их личные данные);
* управления деталями заказа (добавление информации в БД о деталях заказа);
* просмотр заказов;
* просмотр отчетности о деятельности менеджеров;
* проектирование нестандартных окон;
* сохранение макета нестандартных окон в БД.

Входные данные программы – это информация о клиентах компании и детали заказа – вводится пользователем с клавиатуры в процессе работы приложения.

Выходные данные программы должны быть организованны в виде текста на формах приложения и информации в таблицах базы данных.

### 2.1.5 Требования к программным и аппаратным средствам

Для полноценной работы компьютерного приложения пользователю потребуется персональный компьютер с ОС Windows, а также установленный Microsoft SQL Server 2018. Наиболее требовательным к программным и аппаратным средствам является MS SQL Server. Используем официальный сайт Microsoft для ознакомления с требованиями. Требования к оборудованию показаны в таблице 1.

**Таблица 1 – Требования к оборудованию**

|  |  |
| --- | --- |
| Компонент | Требование |
| Жесткий диск | Для SQL Server требуется как минимум 6 ГБ свободного места на диске.  Требования к месту на диске определяются набором устанавливаемых компонентов SQL Server. Рекомендуется устанавливать SQL Server на компьютерах с файловой системой NTFS.  Диски, доступные только для чтения, подключенные и сжатые диски блокируются во время установки. |
| Накопитель | Для установки с DVD-диска необходим соответствующий дисковод. |
| Монитор | ДляSQL Server требуется монитор Super VGA с разрешением 800x600 пикселей или более высоким. |
| Интернет | Для поддержки функциональных средств Интернета требуется доступ к Интернету (могут применяться дополнительные тарифы). |

**Окончание таблицы 1**

|  |  |
| --- | --- |
| Память | Минимальные: Экспресс-выпуски: 512 МБ Все другие выпуски: 1 ГБ Рекомендуется: Экспресс-выпуски: 1 ГБ Все другие выпуски: Для обеспечения оптимальной производительности требуется не менее 4 ГБ с последующим увеличением по мере роста размера базы данных. |
| Быстродействие процессора | Минимум: процессор x64 с тактовой частотой 1,4 ГГц Рекомендуется: 2,0 ГГц и выше |
| Тип процессора | Процессор x64: AMD Opteron, AMD Athlon 64, Intel Xeon с поддержкой Intel EM64T, Intel Pentium IV с поддержкой EM64T. |

Требования к программному обеспечению показаны в таблице 2.

**Таблица 2 – Требования к программному обеспечению**

|  |  |
| --- | --- |
| Компонент | Требование |
| .NET Framework | Для установки SQL Server 2016 и более поздних версий требуется .NET Framework 4.6 для следующих компонентов: ядро СУБД, Master Data Services и репликация. Во время установки SQL Server автоматически устанавливается .NET Framework. Также вы можете вручную установить .NET Framework со страницы Microsoft .NET Framework 4.6 для Windows. Дополнительные сведения, рекомендации и руководство для платформы .NET Framework 4.6 см. в статье [Руководство по развертыванию .NET Framework для разработчиков](https://msdn.microsoft.com/library/ee942965(v=vs.110).aspx). В Windows 8.1и Windows Server 2012 R2 нужно установить обновление [KB2919355](https://support.microsoft.com/kb/2919355) перед установкой .NET Framework 4.6. |
| Сетевое программное обеспечение | Поддерживаемые операционные системы для SQL Server содержат встроенное сетевое программное обеспечение. Именованные экземпляры и экземпляры по умолчанию изолированной установки поддерживают следующие сетевые протоколы: общая память, именованные каналы, TCP/IP и VIA. |

Из вышеизложенного можно сделать вывод, что для полноценной работы программного продукта необходимо выполнить минимальные требования к программным и аппаратным средствам:

* процессор x64: AMD Opteron, AMD Athlon 64, Intel Xeon с поддержкой Intel EM64T, Intel Pentium IV с поддержкой EM64T;
* минимум 6 ГБ свободного места на диске;
* оперативная память 1Gb;
* устройства ввода: дисковод, клавиатура и мышь;
* монитор Super VGA с разрешением 800x600 пикселей или более высоким;
* операционная система Microsoft Windows 7 SP1 и выше;
* Microsoft .NET Framework 4.6;
* Microsoft SQL Server 2018.

### 2.1.6 Требования к надежности

Надёжность программного обеспечения обуславливается наличием в программах разного рода ошибок, внесенных в неё, как правило, при разработке. Под надёжностью ПО понимают способность выполнять заданные функции, сохраняя во времени значения установленных эксплуатационных показателей в заданных пределах, соответствующих заданным режимам и условиям исполнения. Под ошибкой понимают всякое невыполнение программой заданных функций. Появление ошибки является отказом программы. Оценка надежности ПО по ГОСТ 28195-99 констатирует фактическую надёжность по опыту эксплуатации программного комплекса P(t) 1-n/N, где n – число отказов при испытаниях ПО; N – число экспериментов при испытаниях.

Данный раздел определяет требуемый уровень отказоустойчивости разрабатываемого приложения.

**Требования к надежному функционированию системы**

Корректное функционирование программы обеспечивается путем выполнения следующих рекомендаций:

* бесперебойное питание ПК пользователя;
* все файлы и папки должны находиться в общей папке;
* запрещено менять имена файлов и папок, находящиеся в папке с базой данных;
* запрещено удалять файлы и папки, находящиеся в папке с базой данных.

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем операционной системы, не должно превышать времени, необходимого на перезагрузку операционной системы и запуск программы, при условии соблюдения условий эксплуатации технических и программных средств.

**Отказы из-за некорректных действий пользователей системы**

В процессе работы приложения осуществляется контроль входных данных на соответствие требуемому формату.

Доступ к приложению имеют только зарегистрированные пользователи.

Для устранения сбоев при некорректных действиях пользователя, требуется ограничить его доступ к базе данных.

**Требования к пользователям системы**

Пользователями приложения являются директор и менеджеры компании. Так как приложение имеет простой и интуитивно понятный интерфейс, дополнительные навыки и знания, помимо общей эрудиции, не требуются.

### 2.1.7 Защита и сохранность данных

В системе должен быть обеспечен надлежащий уровень защиты информации в соответствии с законом о защите персональной информации и программного комплекса в целом от несанкционированного доступа – «Об информации, информатизации и защите информации» РФ N 24-ФЗ от 20.02.95.

Доступ к приложению осуществляется через авторизацию пользователей с помощью логина и пароля.

Целостность и защиту данных в СУБД SQL Server настраивает администратор с помощью встроенных средств СУБД.

## Описание функционально-логической структуры программы

В данном разделе пояснительной записки представлены разработанные диаграммы, которые описывают алгоритмы разрабатываемого приложения.

В рамках методологии функционального моделирования IDEF0 бизнес-процесс представляется в виде набора функций, которые взаимодействуют между собой, а также показываются информационные, людские и производственные ресурсы, требуемые для каждой функции. Выделяют два типа функциональных моделей. Модель AS-IS предназначена для описания существующих бизнес – процессов на предприятии. Модель ТО-ВЕ описывает идеальное положение вещей – то, к чему нужно стремиться.

**Контекстная диаграмма**

В данной дипломной работе на основе нотации IDEF0 была разработана контекстная диаграмма, которая показывает входные и выходные ресурсы, правила управления и механизм управления. Контекстная диаграмма ТО-ВЕ представлена на рисунке 6.

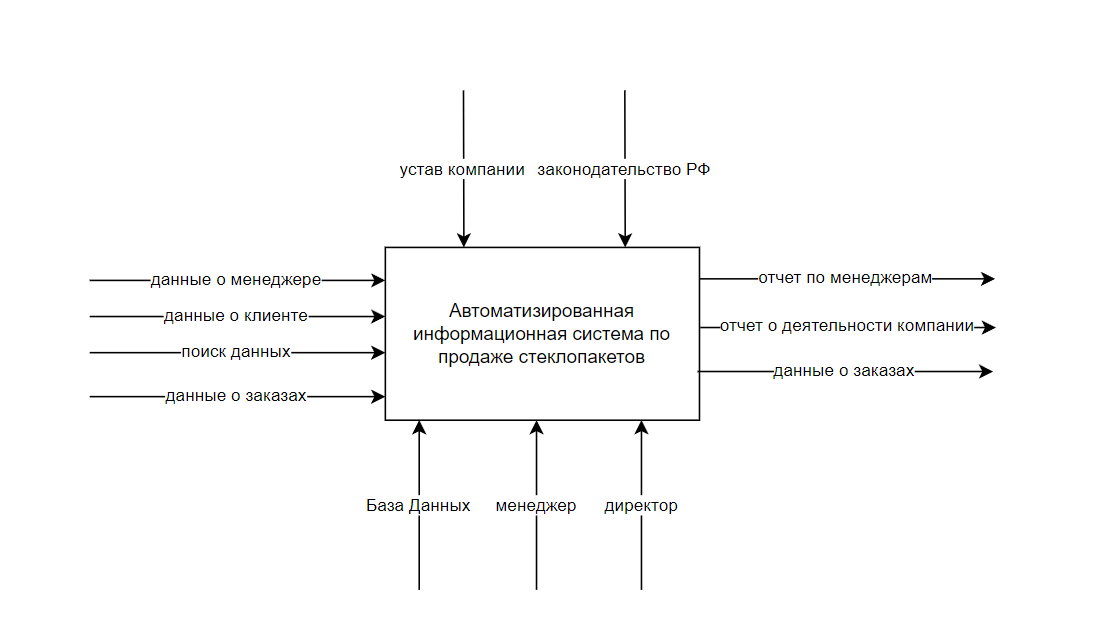


Рисунок 6 – Контекстная диаграмма TO-BE

Взаимодействие системы с окружающей средой описывается в терминах входа (на рис.6 – это данные о менеджерах, клиентах, заказах, справочной информации), выхода (результаты работы приложения – это информация о заказах, отчеты по деятельности компании, отчеты по менеджерам), управления («Законодательство РФ» и «Устав компании») и механизмов («База данных», «менеджер и директор» – это ресурсы, необходимые для процесса функционирования автоматизированной информационной системы по продаже стеклопакетов).

«Клиенты» – те участники, для которых работает компания. Они платят деньги в качестве платы за оказываемые услуги и товары. Отчеты о деятельности фирмы показывают наличие прибыли предприятия. Получение прибыли – цель коммерческой деятельности. Значит, чтобы добиться этой цели, компания должна оказать услуги клиентам.

«Законы РФ» и «Устав компании» – это правила, которыми управляется процесс функционирования фирмы, как предприятия со своими внутренними правилами, и также обязанного «жить» согласно законодательству конкретной страны.

В оказании услуг принимает участие менеджер компании. База данных позволяет предоставлять клиентам более высокий уровень обслуживание за счет сокращения времени. Модель описывает деятельность менеджера и директора компании, а именно следующий функционал: составление заказов и внесение их в базу данных, внесение данных клиентов в базу данных, просмотр отчетности о деятельности фирмы и менеджеров.

**Диаграмма вариантов использования (Use-case diagram)**

Для описания функционального назначения системы построена диаграмма вариантов использования (use case diagram). Диаграмма вариантов использования является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки.

Разработка диаграммы вариантов использования преследует следующие цели:

* определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы;
* сформулировать общие требования к функциональному поведению проектируемой системы;
* разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в форме так называемых вариантов использования, с которыми взаимодействуют некоторые внешние сущности или актеры. При этом актером или действующим лицом называется любой объект, субъект или система, взаимодействующая с моделируемой системой извне. В свою очередь вариант использования служит для описания сервисов, которые система предоставляет актеру.

Построение диаграммы вариантов использования является первым этапом процесса объектно-ориентированного анализа и проектирования, цель которого – представить совокупность функциональных требований к поведению проектируемой системы.

В рамках проекта было определено, что приложение будут использовать несколько пользователей (менеджеры и директор компании). Разработанная диаграмма вариантов использования представлена на рисунке 7.

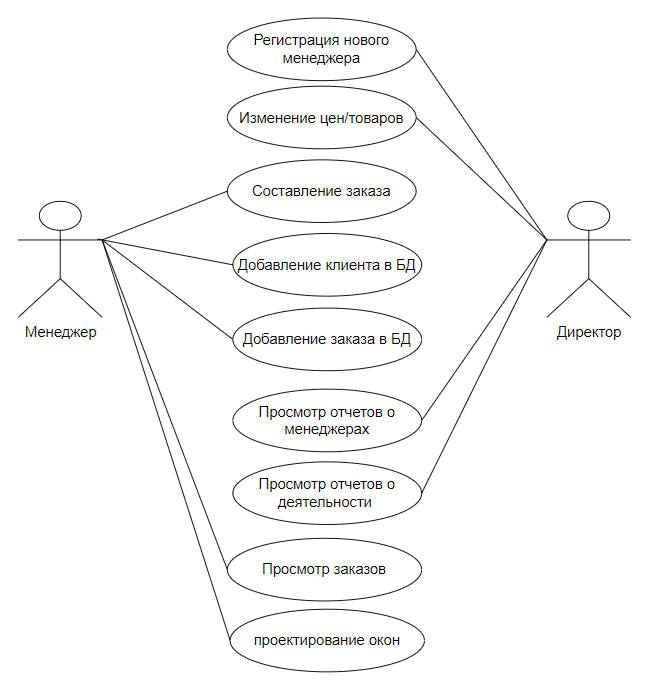


Рисунок 7 – Диаграмма вариантов использования

На рисунке 8 представлена диаграмма деятельности для процедуры оформления заказа. Данная процедура выбрана для демонстрации эффективной работы менеджера. За счет автоматизации данный процесс позволит избежать временных затрат и ошибок при составлении заказа.

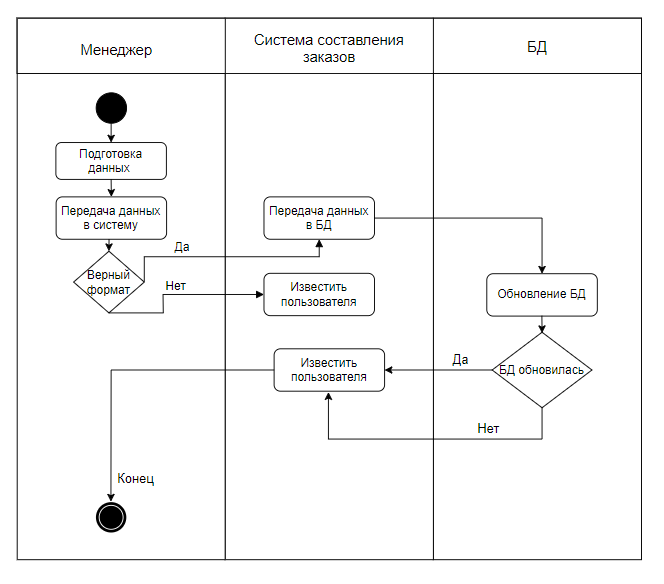


Рисунок 8 – Диаграмма деятельности для процедуры оформления заказа

## 2.3 Логическая модель данных

Построение информационной модели предметной области предполагает выделение сущностей, их атрибутов и первичных ключей, идентификацию связей между сущностями. Общепринятым видом графического изображения реляционной модели данных является ER-диаграмма, на которой сущности изображаются прямоугольниками, соединенные между собой связями. Такое графическое представление облегчает восприятие структуры базы данных по сравнению с текстовым описанием.

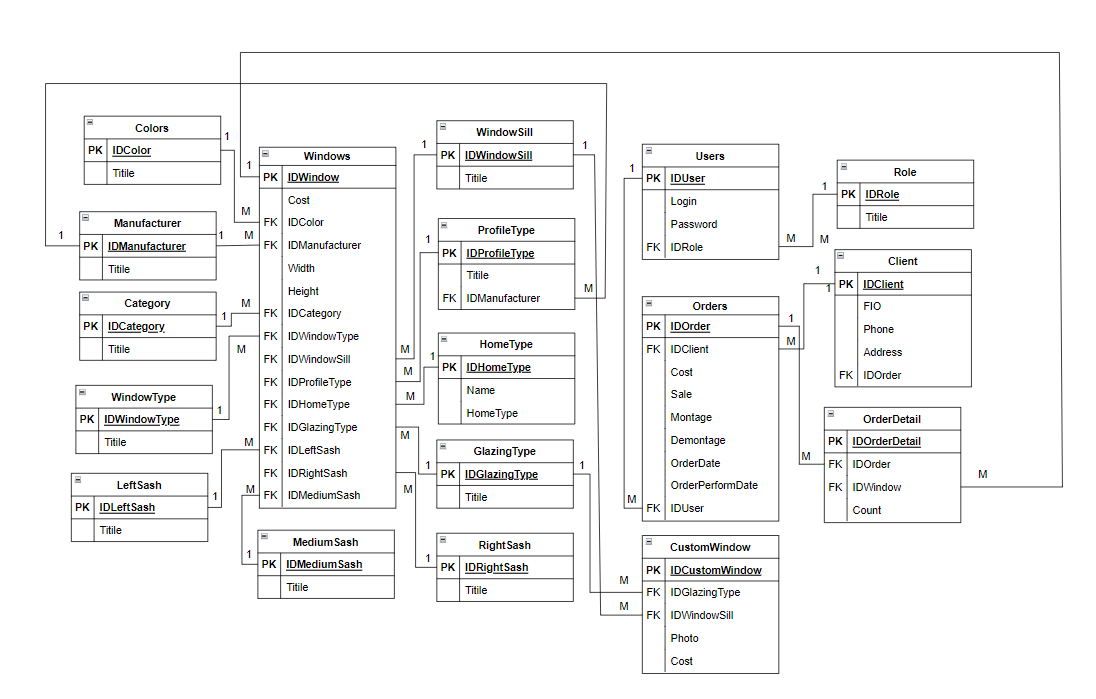


Рисунок 9 – ER диаграмма

Основными понятиями модели данных являются поле, логическая запись, логический файл. Слово «логический» введено, чтобы отличать понятия, относящиеся к логической модели данных, от понятий, относящихся к физической модели данных. Основными понятиями физической модели данных, используемыми для представления логической модели данных, являются поле, физическая запись, физический файл. В частности, логическая запись, состоящая из полей, может быть представлена в виде физической записи (из тех же полей), логический файл – в виде физического файла.

Рассмотрим таблицы подробнее.

В таблице Windows содержится подробная информация об окнах с указанием типа и назначения полей. Подробное описание данной таблицы базы данных представлено в таблице 3.

**Таблица 3 – Windows (Окна)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Key | Field name | Data type | Nullable | Default | Notes |
| PK | IDWindow | int | нет |  | Идентификатор окна |
|  | Cost | decimal(18,2) | нет |  | Цена окна |
| FK | IDColor | int | нет |  | Идентификатор цвета |
| FK | IDManufacturer | int | нет |  | Идентификатор производителя |
|  | Width | int | нет |  | Ширина |
|  | Height | int | нет |  | Высота |
| FK | IDCategory | int | нет |  | Идентификатор категории |
| FK | IDWindowType | int | нет |  | Идентификатор типа окна |
| FK | IDWindowSill | int | нет |  | Идентификатор подоконника |
| FK | IDProfileType | int | нет |  | Идентификатор профиля окна |
| FK | IDHomeType | int | нет |  | Идентификатор дома |
| FK | IDGlazingType | int | нет |  | Идентификатор типа стеклопакета |
| FK | IDLeftSash | int | нет |  | Идентификатор левой створки |
| FK | IDRightSash | int | нет |  | Идентификатор правой створки |
| FK | IDMediumSash | int | нет |  | Идентификатор центральной створки |

В таблице Orders содержится подробная информация о заказах с указанием типа и назначения полей. Подробное описание данной таблицы базы данных представлено в таблице 4.

**Таблица 4 – Orders (Заказы)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Key | Field name | Data type | Nullable | Default | Notes |
| PK | IDOrder | int | нет |  | Идентификатор заказа |
| FK | IDClient | int | нет |  | Идентификатор клиента |
|  | Cost | decimal(18,2) | нет |  | Стоимость заказа |
|  | Sale | decimal(18,2) | нет |  | Скидка |
|  | Montage | bit | да |  | Монтаж |
|  | Demontage | bit | да |  | Демонтаж |
|  | OrderDate | date | нет |  | Дата заказа |
|  | OrderPerformDate | date | да |  | Дата выполнения |
| FK | IDUser | int | да |  | Идентификатор менеджера |

## Описание алгоритма системы

В данном разделе пояснительной записки представлены алгоритмы работы приложения, оформленные в графическом виде. Здесь содержатся укрупненная блок-схема работы приложения в целом и блок-схемы отдельных значимых алгоритмов.

### 2.4.1 Общий алгоритм работы программы

Общий укрупненный алгоритм работы приложения «Автоматизированная информационная система по продаже стеклопакетов» приведен на рисунке 10.



Рисунок 10 – Общий алгоритм работы приложения

### 2.4.2 Алгоритмы отдельных модулей

В данном подпункте рассматриваются наиболее важные алгоритмы, которые отражают взаимодействие системы в целом, а также информационный обмен с базой данных.

При первом запуске приложения необходимо ввести логин и пароль директора, каждый пользователь имеет свой пароль. Затем директор регистрирует менеджеров в системе. Это необходимо для удобства начала пользования приложением.

На рисунке 11 приведен алгоритм работы процедуры добавления менеджера.

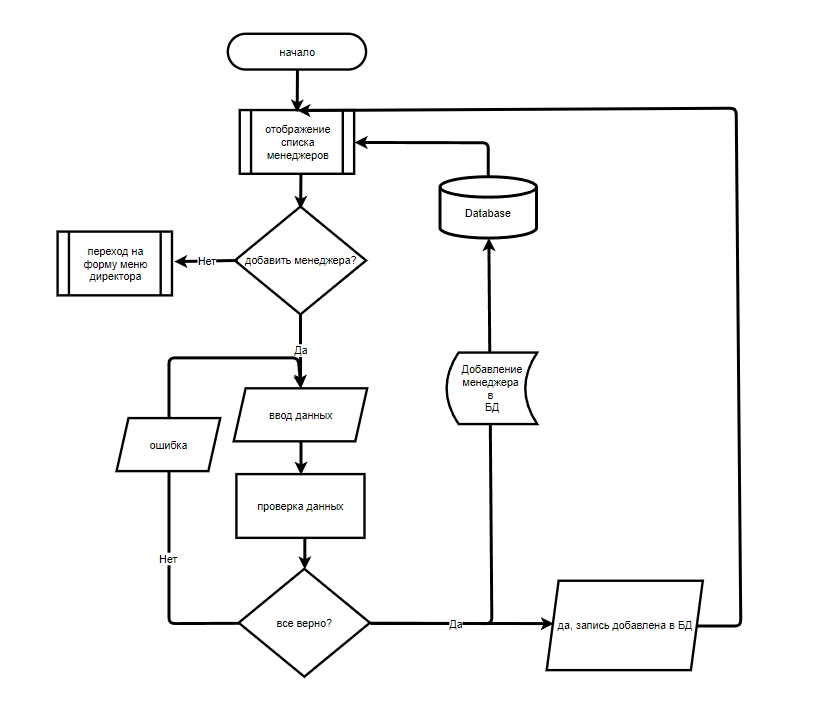


Рисунок 11 – Алгоритм работы процедуры добавления менеджера

Наиболее важной процедурой приложения является составление заказа. Менеджер при оформлении заказа добавляет данные клиента в базу или выбирает клиента, который уже занесен в базу, менеджер составляет заказа и добавляет в корзину окна нужные клиенту и применяет скидку по надобности, приложение рассчитывает стоимость заказа. Затем в БД записывается информация о заказе.

Схема алгоритма работы процедуры составления заказа приведена на рисунке 12.

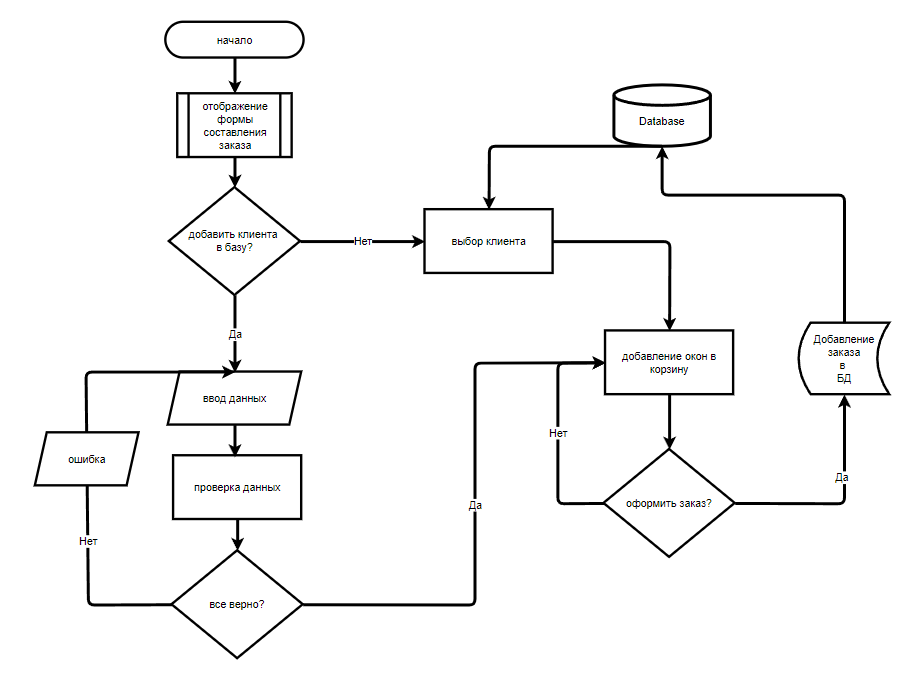


Рисунок 12 – Алгоритм составления заказа

## Программная реализация

В данном разделе пояснительной записки рассматривается программная реализация разрабатываемого приложения, разработка пользовательского интерфейса, а также проведение тестирования и отладки приложения с применением специализированных средств.

### 2.5.1 Пользовательский интерфейс

Пользовательский интерфейс должен отвечать двум основным требованиям:

* обладать свойствами естественности, согласованности и т.д.
* быть узнаваемым или интуитивно-понятным.

Интуитивно-понятным интерфейсом считается такой интерфейс, в котором пользователь может разобраться самостоятельно без помощи сторонних лиц. Чем понятнее интерфейс пользователю, тем меньше времени он потратит на изучение работы приложения.

Обычно заказчики маленьких бизнесов не столько обращают внимание на стиль интерфейса, главное, чтобы он был интуитивно понятен работникам компании.

Рассмотрим принцип работы с цветом по кругу Иттена.

Монохромные цвета.

За основу берётся один цвет, разная его насыщенность и прозрачность. Монохромные сочетания весьма просты в использовании и довольно мягкие на восприятие. Но им часто не хватает выразительности.

Близкие цвета.

Цвета, которые находятся рядом на цветовом круге. Эти цвета обладают схожими характеристиками световых волн, поэтому сочетаются очень просто.

Комплементарные.

Это цвета, расположенные строго друг напротив друга. Самый большой контраст дают именно они. Это всё броское, живое, дерзкое.

Триадные цвета.

Расположены через три сектора. Обычно, контрастные цвета сложно сочетаются, хотя, чистом виде сморятся весьма эффектно в кардинально разных пропорциях. Однако, чем меньше яркость цвета, тем больше возможностей. Триады первичных цветов слишком резкие. Вторичные и третичные триады более мягкие.

Принцип работы с цветом по кругу Иттена прост: подбираются 2 – 4 цвета через принципы цветовых гармоний и экспериментируют с яркостью-контрастностью и площадью заполнения.

Стоит отдельно отметить ахроматические цвета. Никаких красок, никаких оттенков. Только градация от черного к белому. Полный спектр серости.

Наличие чистого белого или чистого чёрного в цветовом решении основных шести гармоний, усиливает контраст, соответственно, интерфейс будет очень «живым», если не переборщить с площадью покрытия цветов. В противном случае, чрезмерная активность цвета будет раздражать.

Немаловажное правило в цветоподборе интерфейса: минимализм. Пять цветов – это максимальный предел. Нельзя перегружать приложение всеми цветами радуги. Задача – донести информацию, а не пугать людей пестростью.

Wireframe – основа структурированного цифрового проекта, один из самых ранних и наиболее важных этапов проектирования. Wireframe использует заполнители, такие как поля с метками, для представления контента, который будет добавлен позже.

Дизайнеры используют Wireframe из-за следующих преимуществ:

– структурированный дизайн. Вы знаете, где все будет размещаться, еще до перехода к конкретным техническим деталям;

– создание основы на раннем этапе. Меню навигация и макет определяют, как будет разрабатываться остальная часть проекта;

– дизайн сфокусирован на контенте.

Wireframe – это скелет дизайна.

Wireframe могут быть созданы с помощью редактора изображений, специализированных инструментов или даже нарисованы на бумаге.

Wireframe форма составления заказов приведена на рисунке 13.

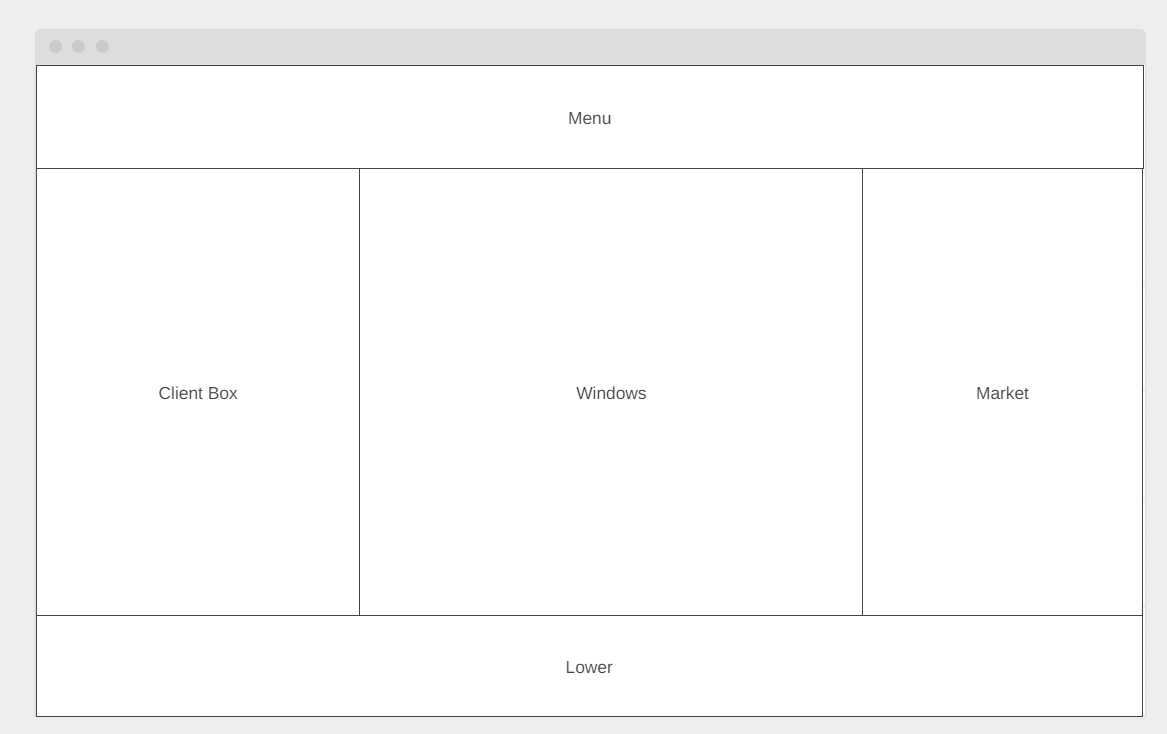


Рисунок 13 – Wireframe

Рассмотрим подробнее созданный Wireframe. Верхняя строчка "Menu" — это то место, где отображается заголовок формы, логотип компании и кнопка возвращения на предыдущую форму. В рамке «Client Box» будет располагаться информация о клиенте. По центру экрана присутствует рамка "Windows" в ней будет находиться все для расчета стоимости окон, а также кнопки добавления окна в базу и корзину. "Market" — это корзина, в которой находятся окна, из этой рамки можно будет оформить заказа и внести его в базу.

Итак, после изучения стандартов и основных принципов проектирования пользовательского интерфейса, рассмотрим формы разрабатываемого программного продукта:

* форма авторизации;
* форма составления заказа;
* форма просмотра заказов;
* форма добавления менеджера;
* форма меню менеджера;
* форма меню директора;
* форма отчетности по менеджерам;
* форма отчетности по доходам;
* форма проектирования окон;
* форма просмотра спроектированных окон;
* форма изменения цены/товара.

Перейдем к рассмотрению форм.

Для начала рассмотрим форму первичной авторизации, так как это первая форма, с которой пользователь начинает своё знакомство с приложением. На рисунке 14 продемонстрирован интерфейс данной формы.



Рисунок 14 – Форма авторизации

Следующая форма позволяет оформлять заказы и вносить их в базу данных. Каждому заказу присваивается свой номер, сумма заказа, менеджер, который его оформил, размер скидки, необходим ли монтаж и демонтаж окон, f также дата заказа и дата выполнения. Пример данной формы показан на рисунке 15.

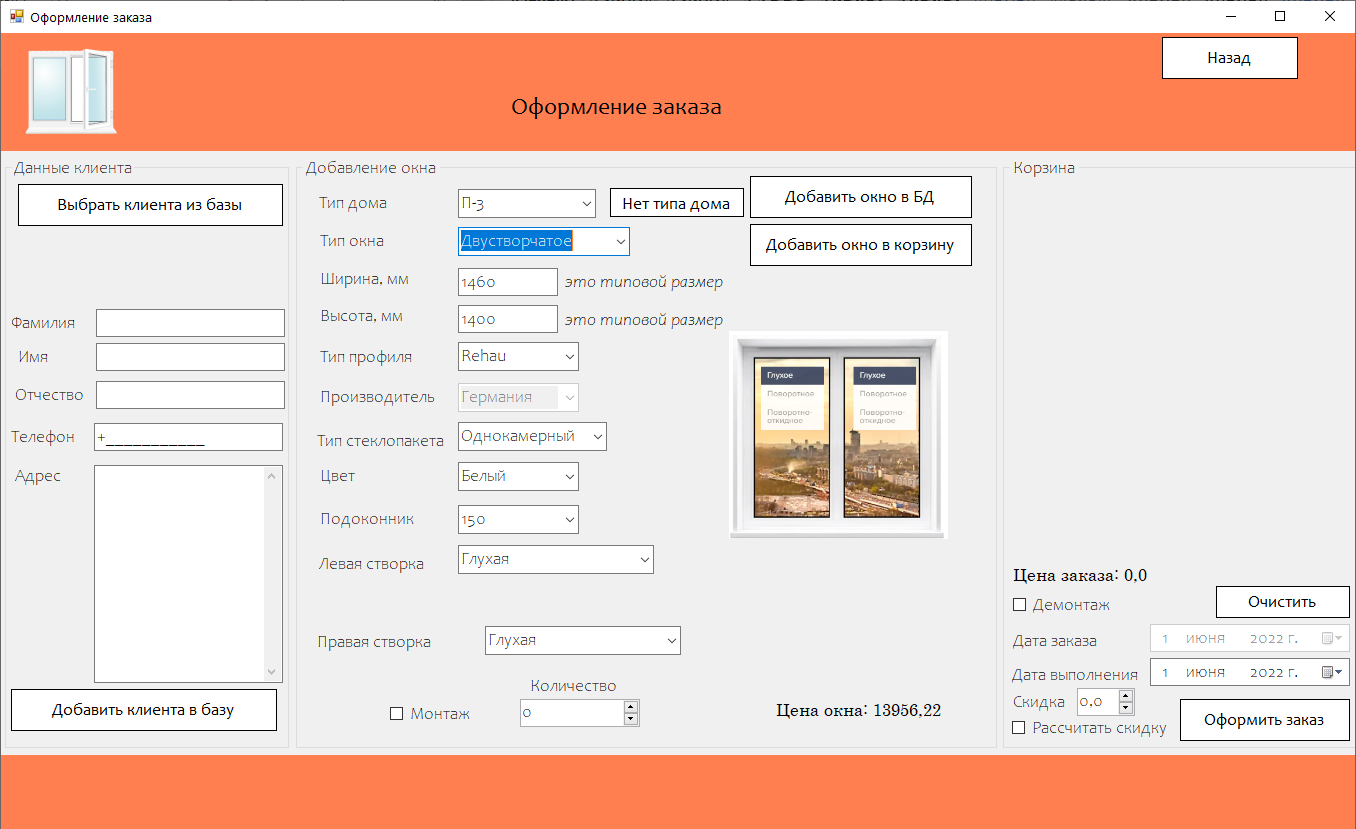


Рисунок 15 – Форма составления заказов

Форма просмотра заказов предназначена, чтобы отслеживать заказы из базы данных. На этой форме менеджер может просмотреть свои оформленные заказы и посмотреть дополнительную информацию по ним. На рисунке 16 представлена форма просмотра заказов.

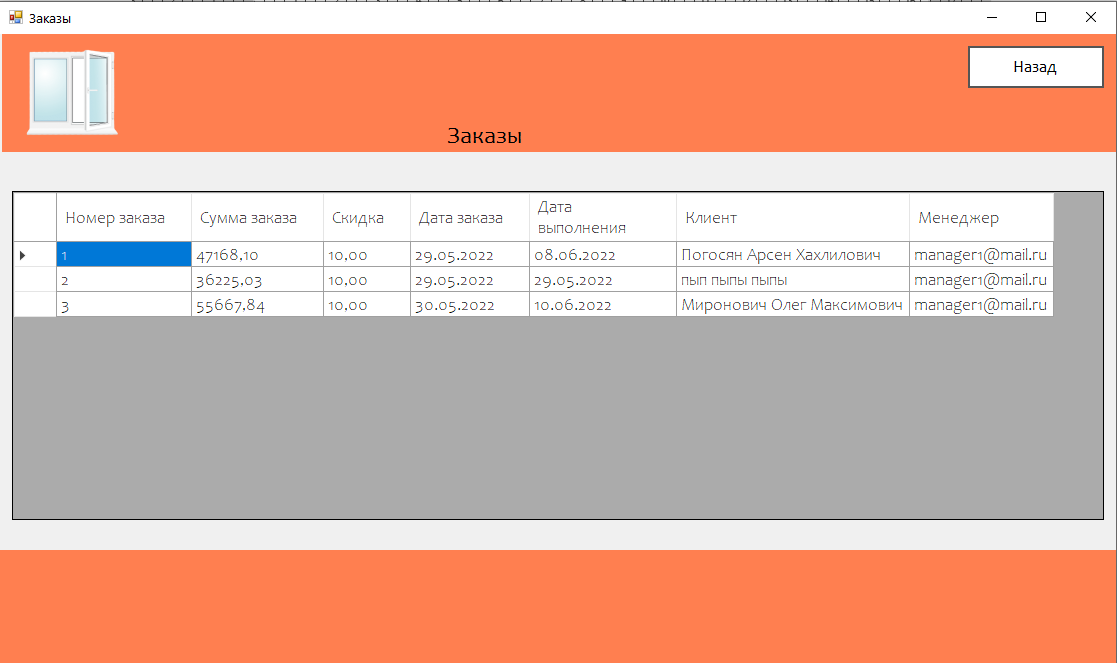


Рисунок 16 – Форма просмотра заказов

Следующая форма позволяет директору просматривать зарегистрированных менеджеров и добавлять новых в систему. На этой форме директор создает данные для входа новому менеджеру. На рисунке 17 продемонстрирована форма добавления менеджера.

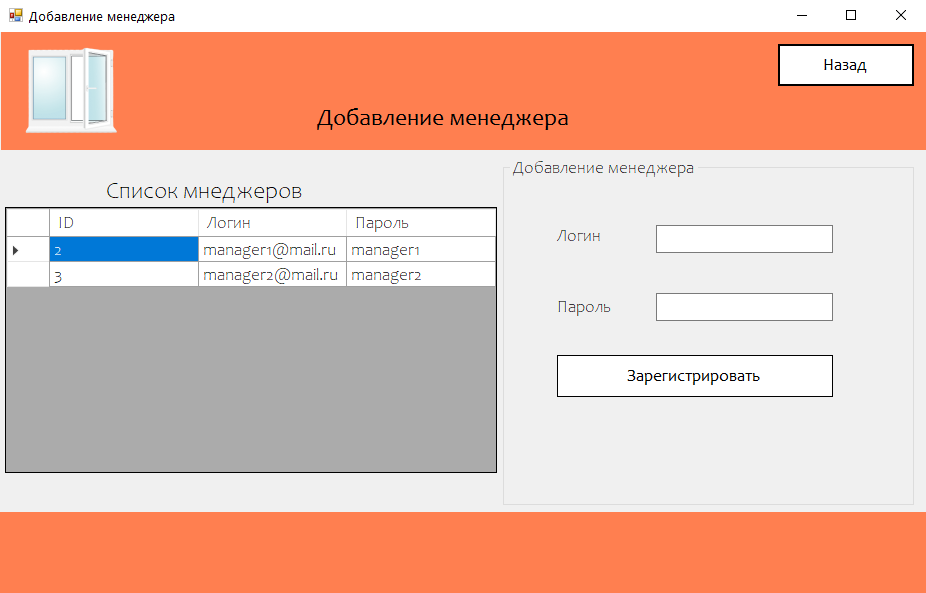


Рисунок 17 – Форма регистрации клиентов

Форма меню менеджера служит менеджерам для навигации по приложению, на ней менеджер может увидеть весь доступный ему функционал и перейти к нужной форме. Форма меню менеджера продемонстрирована на рисунке 18.

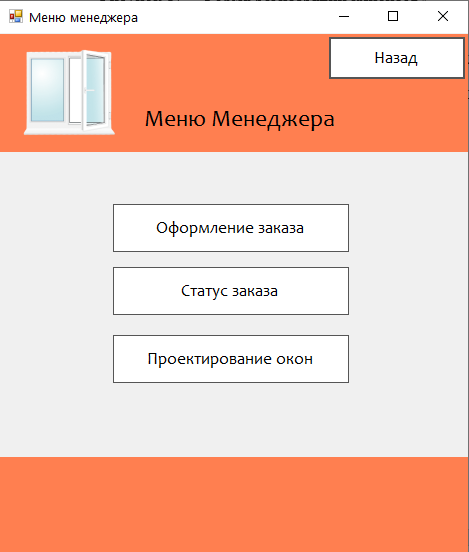


Рисунок 18 – Форма меню менеджера

Форма меню директора служит директору для навигации по приложению, на ней директор может увидеть весь доступный ему функционал и перейти к нужной форме. Форма меню директора представлена на рисунке 19.

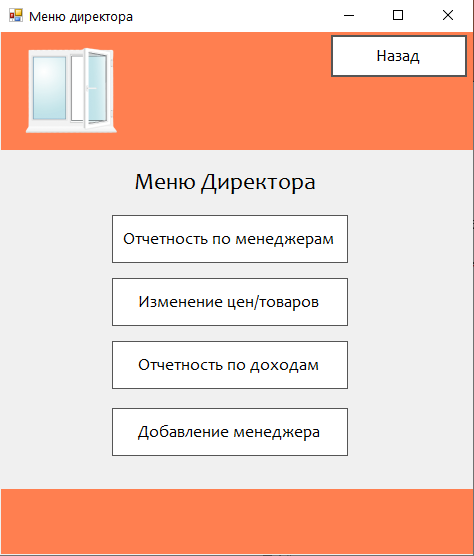


Рисунок 19 – Форма меню директора

На следующей форме представлена отчетность по менеджерам, а именно заказы, которые они оформленные, суммы этих заказов и даты, а также директор может выбрать менеджера, по которому ему нужна статистика и период времени, что продемонстрированно на рисунке 20.

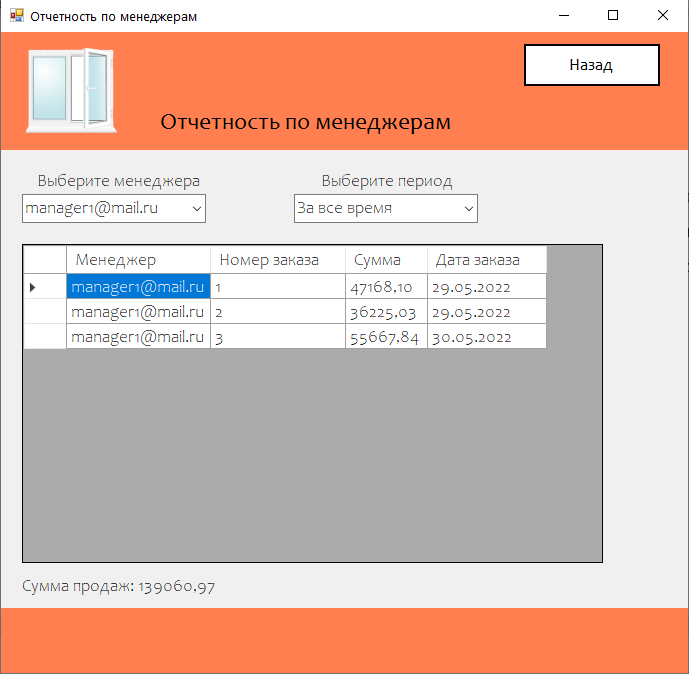


Рисунок 20 – Форма отчетности по менеджерам

На форме отчетности по доходам показана информация о деятельности фирмы, а именно сколько заказов было принято, на какую сумму и число реализованной продукции, также можно отфильтровать эти данные по временному периоду, что показано на рисунке 21.

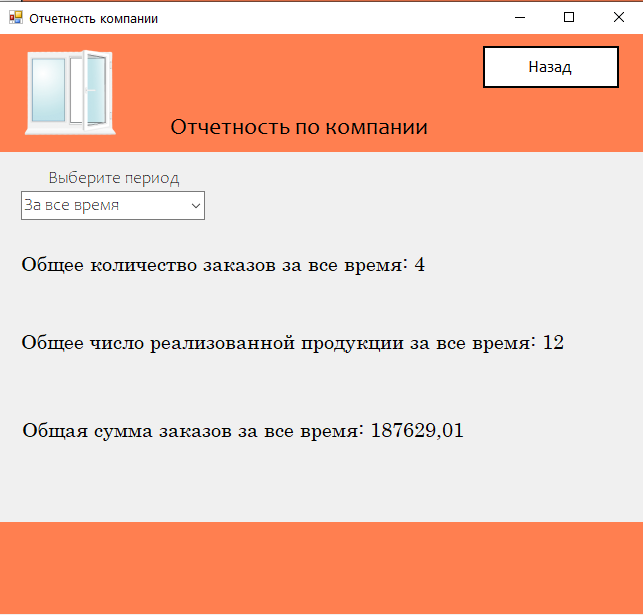


Рисунок 21 – Форма отчетности по доходам

Форма изменения цен и товаров позволяет пользователю изменить цены окон и их характеристики. Данная форма показана на рисунке 22.

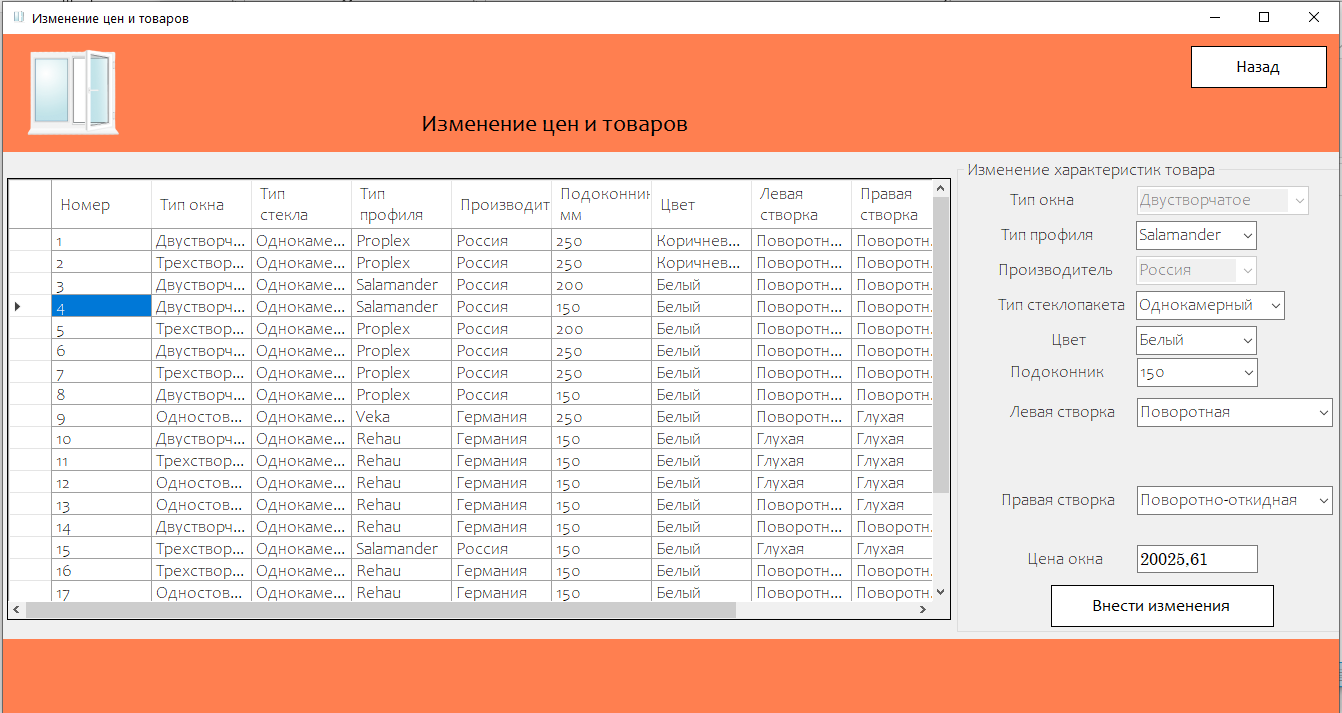


Рисунок 22 – Форма изменения цен и товаров

Следующая форма служит для проектирования нестандартных окон и внесения их в базу данных, также форма позволяет сохранять макеты нестандартных окон, что показано на рисунке 23.

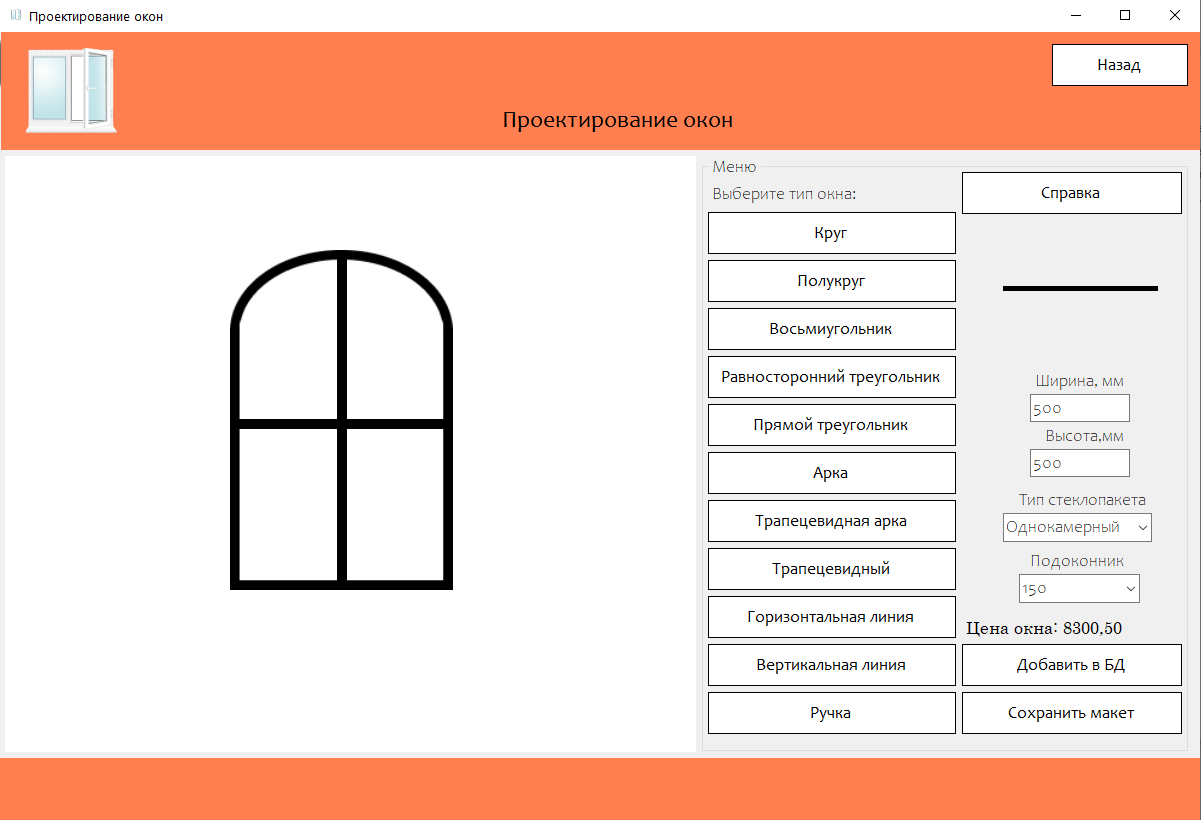


Рисунок 23 – Форма проектирования нестандартных окон

На форме просмотра спроектированных окон пользователь может увидеть макет спроектированного окна из базы, а также его стоимость и характеристики, что представлено на рисунке 24.

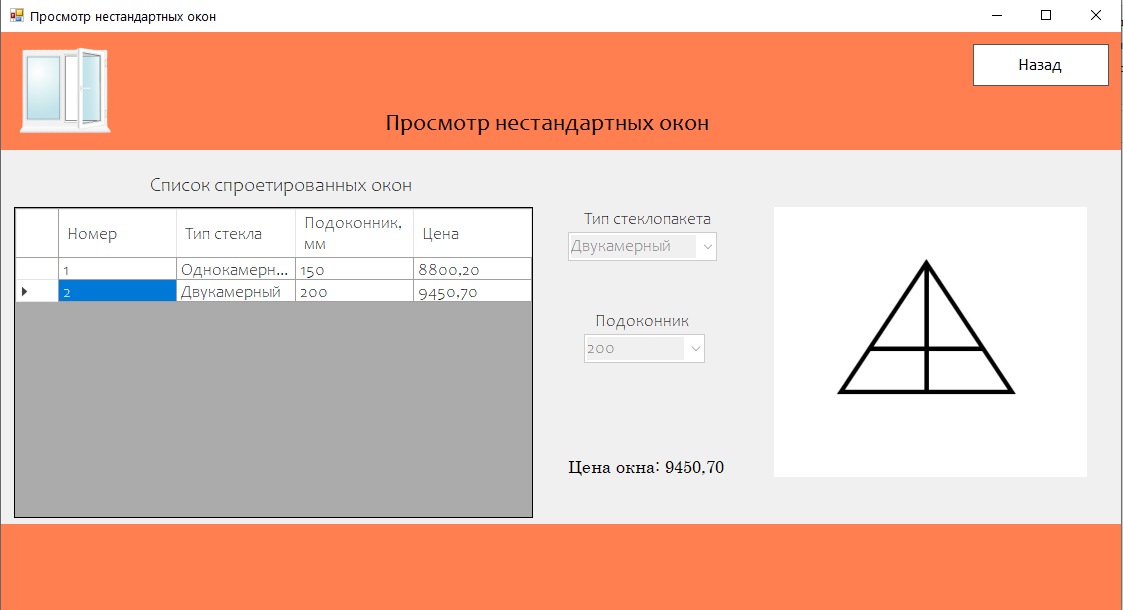


Рисунок 24 – Форма просмотра спроектированных окон

Итак, разработанный пользовательский интерфейс является дружелюбным, так как основные требования соблюдены. Цветовое решение приятно для восприятия, управление в приложение осуществляется прямым манипулированием мыши. Все некорректные действия пользователей сопровождаются соответствующими всплывающими сообщениями. При заполнении полей форм предусмотрена программная проверка на соответствие формату. Использованы выпадающие списки значений для стандартных полей, чтобы минимизировать ошибки заполнения.

### 2.5.2 Тестирование и отладка программы

Отладка — это поиск (локализация), анализ и устранение ошибок в программном обеспечении, которые были найдены во время тестирования.

Виды ошибок, при которых помогает отладка программы:

– ошибки компиляции;

– ошибки компоновки;

– ошибки выполнения.

Ошибки компиляции — это простые ошибки, которые в компилируемых языках программирования выявляет компилятор (программа, которая преобразует текст на языке программирования в набор машинных кодов). Если компилятор показывает несколько ошибок, отладку кода начинают с исправления самой первой, так как она может быть причиной других.

В интерпретируемых языках (например Python) текст программы команда за командой переводится в машинный код и сразу исполняется. К моменту обнаружения ошибки часть программы уже может исполниться.

Ошибки компоновки связаны с разрешением внешних ссылок. Выявляет компоновщик (редактор связей) при объединении модулей программы. Простой пример — ситуация, когда требуется обращение к подпрограмме другого модуля, но при компоновке она не найдена. Ошибки также просто найти и устранить.

Ошибки выполнения обнаруживает операционная система, аппаратные средства или пользователи при выполнении программы. Они считаются непредсказуемыми и проявляются после успешной компиляции и компоновки.

Рассмотрим методы отладки программы.

Метод ручного тестирования.

Отладка программы заключается в тестировании вручную с помощью тестового набора, при работе с которым была допущена ошибка. Несмотря на эффективность, метод не получится использовать для больших программ или программ со сложными вычислениями. Ручное тестирование применяется как составная часть других методов отладки.

Метод индукции.

В основе отладки системы — тщательный анализ проявлений ошибки. Это могут быть сообщения об ошибке или неверные результаты вычислений. Например, если во время выполнения программы завис компьютер, то, чтобы найти фрагмент проявления ошибки, нужно проанализировать последние действия пользователя. На этапе отладки программы строятся гипотезы, каждая из них проверяется. Если гипотеза подтвердилась, информация об ошибке детализируется, если нет — выдвигаются новые.

Важно, чтобы выдвинутая гипотеза объясняла все проявления ошибки. Если объясняется только их часть, то либо гипотеза неверна, либо ошибок несколько.

Метод дедукции.

Сначала специалисты предлагают множество причин, по которым могла возникнуть ошибка. Затем анализируют их, исключают противоречащие имеющимся данным. Если все причины были исключены, проводят дополнительное тестирование. В обратном случае наиболее вероятную причину пытаются доказать.

Рассмотрим отладку программы с помощью точки остановы. Пример работы приложения в отладочном режиме отображен на рисунке 25.

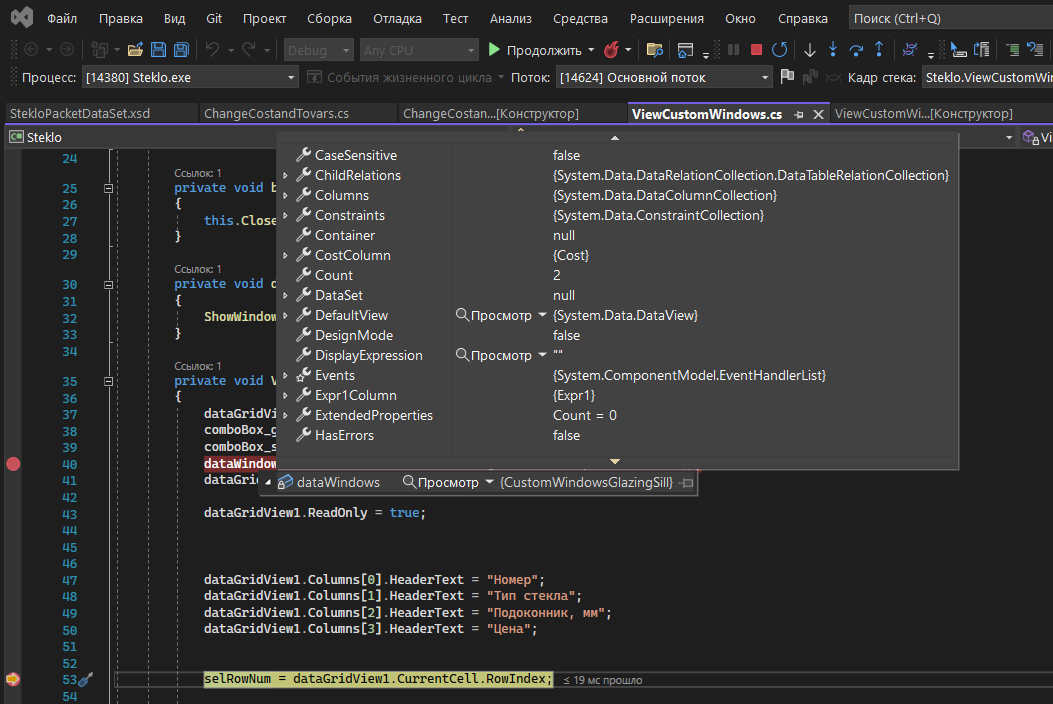


Рисунок 25 – Отладка приложения

Из представленной выше отладки можно сделать вывод, что запрос к базе сформировался корректно, о чем свидетельствует заполненная форма просмотра спроектированных окон, представленная на рисунке 26.

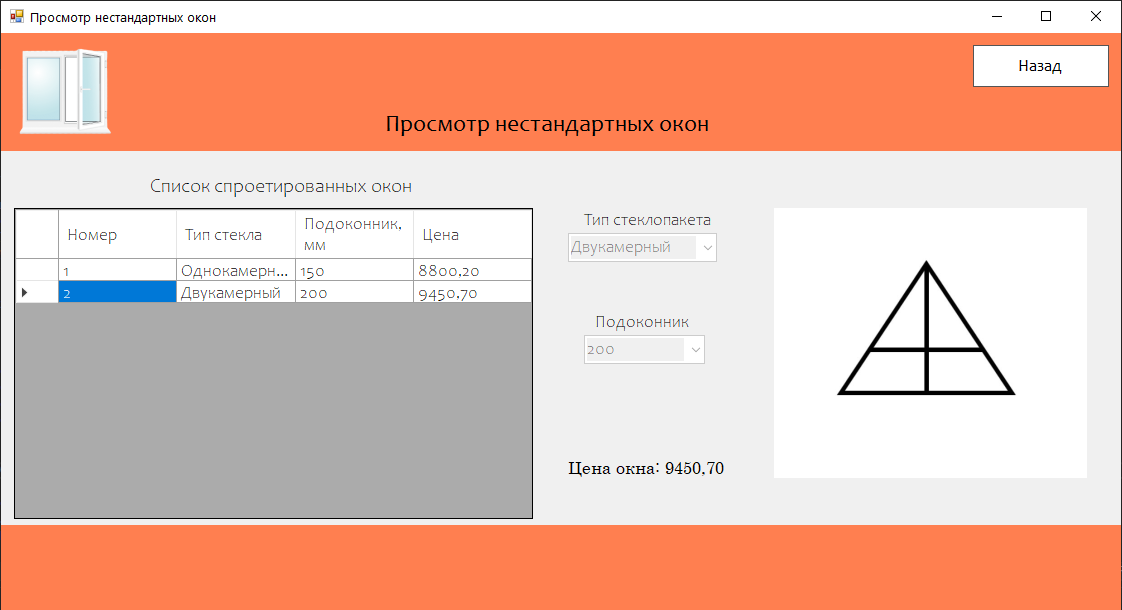


Рисунок 26 – Заполненная форма просмотра спроектированных окон

Тестирование – это процесс выполнения программы на некотором наборе данных, для которого заранее известен результат или известны правила поведения этих программ.

Тестирование компонентов — тестируется минимально возможный для тестирования компонент, например, отдельный класс или функция. Часто тестирование компонентов осуществляется разработчиками программного обеспечения.

Интеграционное тестирование — тестируются интерфейсы между компонентами, подсистемами или системами. При наличии резерва времени на данной стадии тестирование ведётся итерационно, с постепенным подключением последующих подсистем.

Системное тестирование — тестируется интегрированная система на её соответствие требованиям.

Альфа-тестирование — имитация реальной работы с системой штатными разработчиками, либо реальная работа с системой потенциальными пользователями/заказчиком. Чаще всего альфа-тестирование проводится на ранней стадии разработки продукта, но в некоторых случаях может применяться для законченного продукта в качестве внутреннего приёмочного тестирования. Иногда альфа-тестирование выполняется под отладчиком или с использованием окружения, которое помогает быстро выявлять найденные ошибки. Обнаруженные ошибки могут быть переданы тестировщикам для дополнительного исследования в окружении, подобном тому, в котором будет использоваться программа.

Бета-тестирование — в некоторых случаях выполняется распространение предварительной версии (в случае проприетарного программного обеспечения иногда с ограничениями по функциональности или времени работы) для некоторой большей группы лиц с тем, чтобы убедиться, что продукт содержит достаточно мало ошибок. Иногда бета-тестирование выполняется для того, чтобы получить обратную связь о продукте от его будущих пользователей.

Проведем тестирования входных данных в программе, для этого создадим или отредактируем запись в таблице. В таблице 5 перечислены основные поля для проведения тестирования методом кейс-тест.

**Таблица 5 – Описание информационных полей для тестирования**

|  |  |
| --- | --- |
| Наименование | Описание |
| Наименование проекта | Наименование проекта проверено |
| Номер версии | Версия проекта (первый номер можно принять как 1.0) |
| Имя тестера | Имя тестера, который выполнял эти тесты |
| Даты тестирования | Даты, когда проводили тестирование – это может быть один тест или несколько. |
| Test Case # | Уникальный ID для каждого test case. |
| Приоритет тестирования (Малы /Средний /высокий) | Насколько важен каждый тест. |
| Название тестирования /Имя | Название тестирования. Например, проверка формы авторизации с правильным логином и паролем. |
| Резюме испытания | Описание, чего нужно достигнуть при тестировании. |
| Шаги тестирования | Перечислите детально все шаги тестирования. Напишите, в каком порядке должны быть выполнены эти шаги. |
| Данные тестирования | Напишите тестовые данные, используемые для этого тестирования. Таким образом, актуальные данные, которые будут предложены, будут использоваться для проведения тестирования. |
| Ожидаемый результат | Опишите подробно ожидаемый результат, включая любые сообщения и ошибки, которые должны быть выданы на экран. |
| Фактический результат | Опишите любое соответствующее поведение системы после выполнения тестирования. |
| Предпосылки | Любые предварительные действия, которые должны быть выполнены перед проведением тестирования. |
| Постусловия | Состояние системы после выполнения тестирования |
| Статус (Pass/Fail) | Если фактический результат не соответствует ожидаемым результатам отметка, что тест провалился (fail). В противном случае как прошло (pass) |
| Комментарии | Используйте эту область для любых дополнительных записей или комментариев. |

Для тестирования правильной работы приложения был разработан тест-кейс, который описывает авторизацию пользователя. Тест-кейс приведен в таблице 6.

**Таблица 6 – Успешная авторизация пользователя**

|  |  |
| --- | --- |
| Наименование | Описание |
| Наименование | АИС по продаже стеклопакетов |
| Номер версии | 1.0 |
| Имя тестера | Яремко Богдан |
| Даты тестирования | 25.05.2022 |
| Test Case # | TC\_UI\_1 |
| Приоритет тестирования | Высокий |
| Название тестирования /Имя | Проверка корректной авторизации пользователя системы |
| Резюме испытания | Убедиться в корректной авторизации пользователей системы |
| Шаги тестирования | 1. Запускаем приложение 2. В поле «Логин» вводим логин пользователя 3. В поле «Пароль» вводим пароль пользователя |
| Данные тестирования | Данные для входа в приложение:  Логин: manager1@mail.ru  Пароль: manager1 |
| Ожидаемый результат | Всплывающее сообщение о том, что менеджер авторизовался в системе |
| Фактический результат | Полностью соответствует ожидаемому |
| Предпосылки | Некорректная авторизация |
| Постусловия | 1. Запущено приложение 2. Отображена форма «Авторизация» 3. Все поля заполнены 4. После нажатия кнопки «Войти» пользователь получает сообщение об успешной авторизации |
| Статус*(Pass/Fail)* | Pass |
| Комментарии |  |

Далее создадим тест-кейс добавления менеджера в систему. Тест-кейс продемонстрирован в таблице 7.

**Таблица 7 – Успешное добавление менеджера в систему**

|  |  |
| --- | --- |
| Наименование | Описание |
| Наименование проекта | АИС по продаже стеклопакетов |
| Номер версии | 1.0 |
| Имя тестера | Яремко Богдан |
| Даты тестирования | 28.05.2022 |
| Test Case # | TC\_UI\_2 |
| Приоритет тестирования | Средний |
| Название тестирования | Проверка корректного добавления менеджера в систему |
| Резюме испытания | Убедиться в корректном добавлении нового менеджера |
| Шаги тестирования | 1. Запускаем приложение 2. В поле «Логин» вводим логин директора 3. В поле «Пароль» вводим пароль директора 4. В открывшемся меню директора выбираем пункт «Добавление менеджера» 5. На форме добавления менеджера в поле «Логин» вводим логин которого нет в базе 6. В поле «Пароль» вводим любой пароль 7. Нажимаем кнопку «Добавить менеджера» |
| Данные тестирования | Данные для входа в приложение:  Логин: director@gmail.com  Пароль: director  Данные для проверки:  Логин: manager6@mail.ru  Пароль: manager6 |
| Ожидаемый результат | Всплывающее сообщение о том, что менеджер добавлен в систему |
| Фактический результат | Полностью соответствует ожидаемому |

**Окончание таблицы 7**

|  |  |
| --- | --- |
| Предпосылки | Некорректное добавление новых менеджеров |
| Постусловия | 1. Запущено приложение 2. Отображена форма «Добавление менеджера» 3. Все поля заполнены 4. После нажатия кнопки «Добавить менеджера» пользователь получает сообщение об успешном добавлении нового менеджера |
| Статус *(Pass/Fail)* | Pass |
| Комментарии |  |

Юнит-тест или модульный тест, — это программа, которая проверяет работу небольшой части кода. Разработчики регулярно обновляют сайты и приложения, добавляют фичи, рефакторят код и вносят правки, а затем проверяют, как всё работает.

Тестировать систему целиком после каждого обновления — довольно муторно и неэффективно. Поэтому обновлённые или исправленные части кода прогоняют через юнит-тесты.

Для юнит-тестирования подключают тестовые фреймворки — они позволяют имитировать функции. В коде больших проектов много зависимостей: одна функция вызывает другую и влияет на разные части программы. Но, как правило, достаточно проверить функции «в вакууме», отдельно от остального кода. Для этого и нужен тестовый фреймворк — он моделирует условия, в которых функция А вызывает функцию Б изолированно от других функций.

Рассмотрим два юнит-теста, первый тест проверяет создание нового менеджера с данными которых нет в базе, то есть такой менеджер должен добавиться в базу без каких-либо проблем. Второй тест проверяет добавиться ли в систему менеджер данные которого уже есть в базе. Юнит-тесты показаны на рисунке 27.

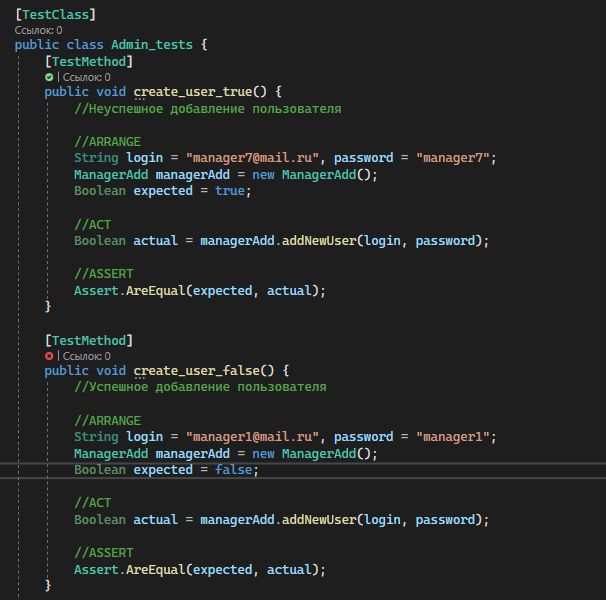


Рисунок 27 – Юнит-тесты

Как и предполагалось первый тест выполнился без каких-либо проблем, а второй не смог выполниться из-за того, что такие данные уже есть в базе.

Подводя итоги по разделу, отладка и тестирование, можно сказать, что при отладке приложения сбоев не выявлено, данные обработаны корректно, приложение работает исправно. При создании тестов и при их прохождении все работало корректно, и данные были без ошибок и промедления как добавлены в базу данных, так и извлечены из нее на формы приложения.

## 2.6 Инструкция пользователя

В данном разделе будут отражены рекомендации и правила использования разработанного приложения «Автоматизированная информационная система по продаже стеклопакетов». Для начала работы запускаем ярлык – Steklo.exe.

**Форма авторизации пользователя приложения**

После запуска программы открывается окно аутентификации, где нужно ввести логин и пароль для входа в систему. На рисунке 28 отображено данное окно. Также на форме присутствует кнопка «Выйти», которая закрывает приложение. Фрагмент кода авторизации представлен в приложении А.

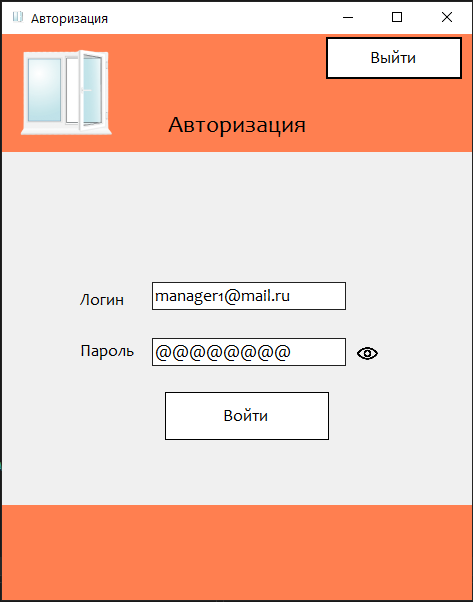


Рисунок 28 – Форма для ввода данных

При ошибочном вводе пароля или логина вы увидите надпись: «Таких данных нет». Следовательно, вы должны проверить правильность ввода. Сообщение было показано, отслеживание ошибок ввода пароля работает. На рисунке 29 представлена описываемая ситуация. Если пользователь 3 раза подряд введет неправильные данные, то система заблокируется на 15 секунд, что продемонстрировано в приложении Б на рисунке Б.1.

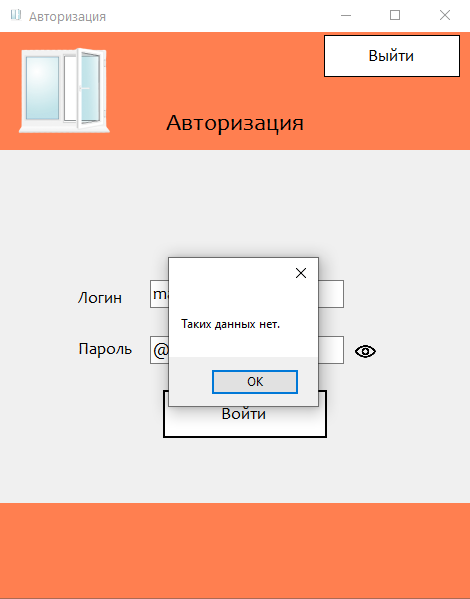


Рисунок 29 – Результат неправильного ввода логина или пароля

В случае корректного ввода данных, программа даст доступ к меню менеджера или директора, в зависимости от введенных данных. При правильном вводе пароля и логина аутентификация проходит успешно. Рассмотрим меню менеджера.

**Форма «Меню менеджера»**

Переходим на меню менеджера, оно служит для навигации менеджера в приложении. На рисунке 30 продемонстрированно как выглядит данная форма. На форме присутствует 5 кнопок. Кнопка «Назад» перемещает пользователя обратно на окно авторизации, кнопка «Оформление заказа» открывает форму составления заказов, кнопка «Статус заказа» перемещает менеджера на форму просмотра оформленных им заказов, кнопка «Проектирование окон» перемещает пользователя на окно проектирование нестандартных стеклопакетов, кнопка «Просмотр нестандартных стеклопакетов» открывает форму со всеми спроектированными окнами. Фрагмент кода формы представлен в приложении А.

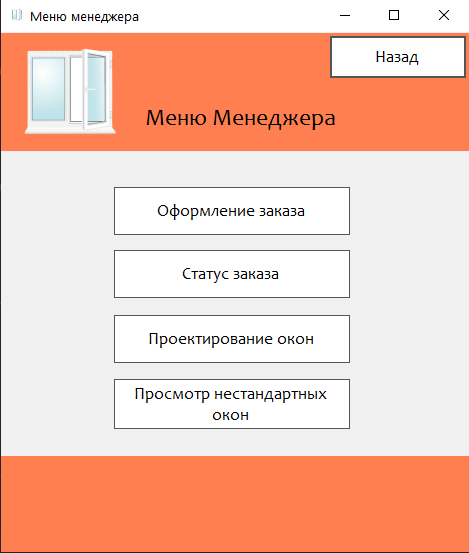


Рисунок 30 – Меню менеджера

**Форма «Оформление заказа»**

Рассмотрим форму оформления заказа. Форма представлена на рисунке 31. В правом верхнем углу есть кнопка «Назад», как и на любой форме кроме авторизации, данная кнопка позволяет вернуться к предыдущему окну приложения.

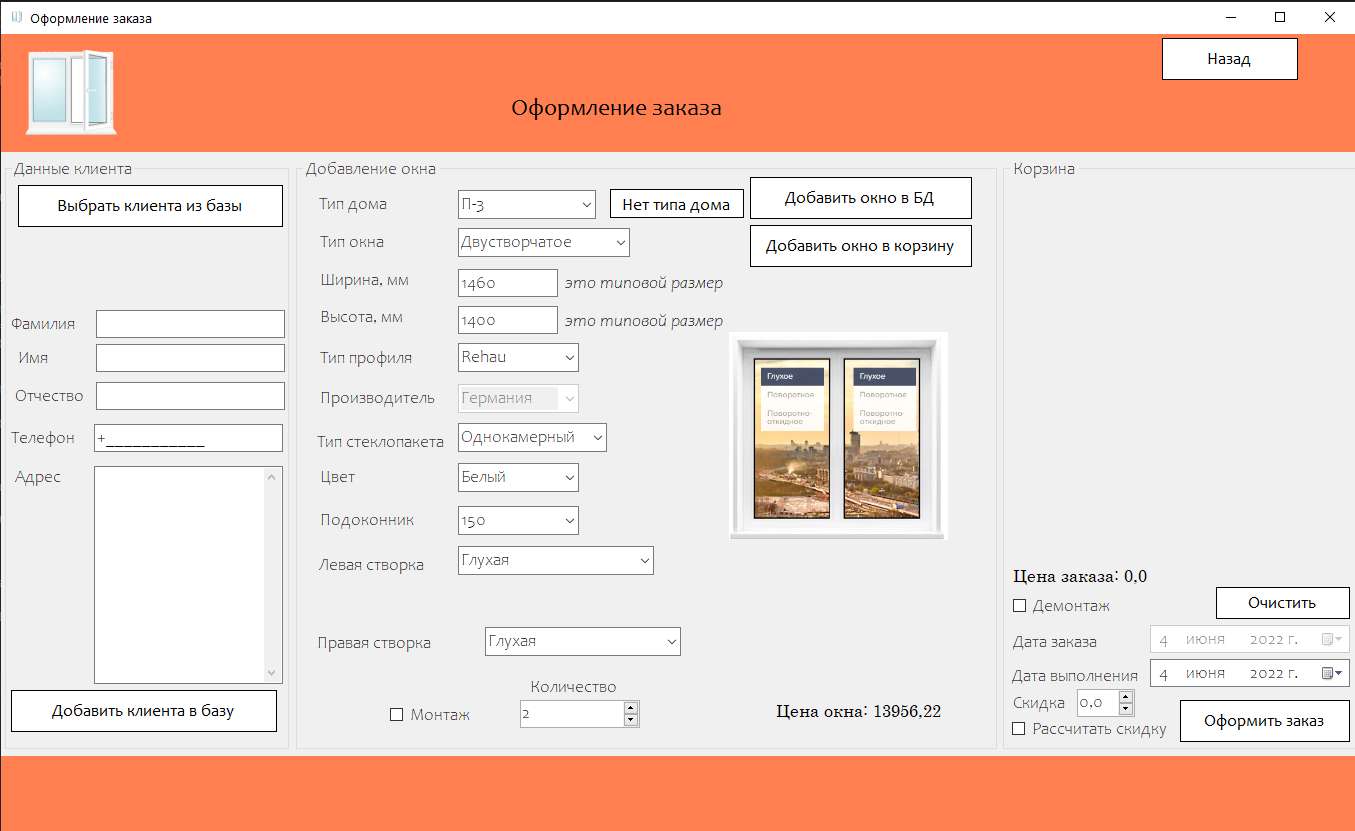


Рисунок 31 **–** Форма оформления заказа

В левой части формы расположен блок данных клиента, где менеджер может выбрать клиента из базы нажав на кнопку «Выбрать клиента из базы», что показано на рисунке 32 или добавить нового клиента в базу, для этого он должен заполнить следующие поля: фамилия, имя, отчество, телефон, адрес, а затем нажать на кнопку «Добавить клиента в базу». Фрагмент кода формы «Оформление заказа» представлен в приложении А.



Рисунок 32 – Выбор клиента из базы

Если менеджер выбрал клиента из базы, то данные клиента занесутся в соответствующие поля, как показано на рисунке 33.

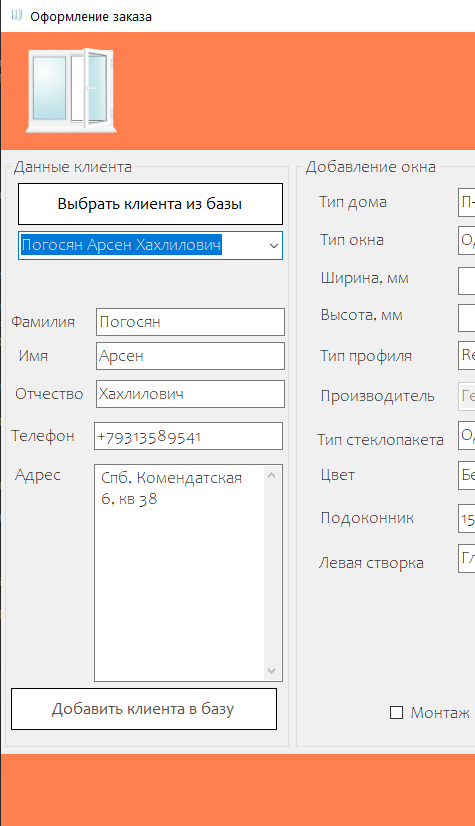


Рисунок 33 – Данные клиента из базы

Если все же клиент отсутствует в базе и менеджер заполнил все поля и нажал кнопку «Добавить клиента в базу», то будет выведено сообщение о том, что клиент успешно добавлен в базу. Добавление клиента в базу показано на рисунке 34.



Рисунок 34 – Добавление клиента в базу

В центральной части формы расположен блок «Добавление окна». В этом блоке менеджер задает ширину и высоту окна, а также выбирает его характеристики, такие как тип окна, тип дома, тип профиля, тип стеклопакета, цвет рамы, размер подоконника и типы створок окна. Также в этом блоке расположена картинка окна для лучшей наглядности. На рисунке 35 показан блок «Добавление окна».



Рисунок 35 – Блок «Добавления окна»

Если в базе отсутствует тип дома клиента, то менеджер может нажать на кнопку «Нет типа дома», тогда откроется окно добавления типа дома. В окне добавления типа дома пользователю нужно заполнить поля «Имя типа дома» и «Тип дома», далее нажать кнопку «Добавить», что показано на рисунке 36.

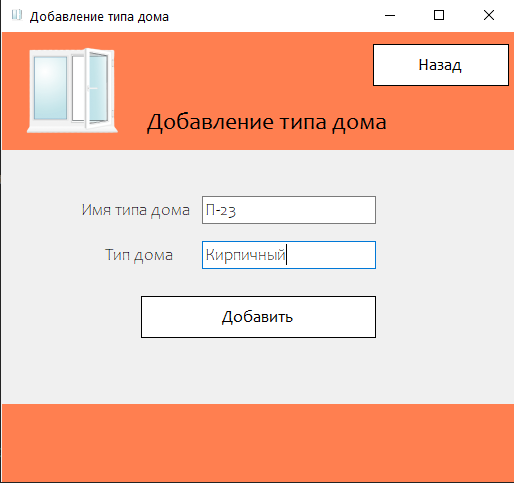


Рисунок 36 – Добавление типа дома

Чтобы начать оформление заказа менеджеру нужно выбрать количество окон, которые он хочет добавить в корзину и указать нужен ли монтаж. Затем пользователь должен нажать кнопку «Добавить окно в корзину». Если такое окно есть в базе, то оно сразу добавиться в корзину, иначе на экране появиться сообщение о том, что окно сначала нужно добавить в базу нажав на кнопку «Добавить окно в БД» и после успешного добавление окна в базу, добавить окно в корзину можно нажав соответствующую кнопку. Сообщение о том, что нужно добавить окно в базу показано на рисунке 37.

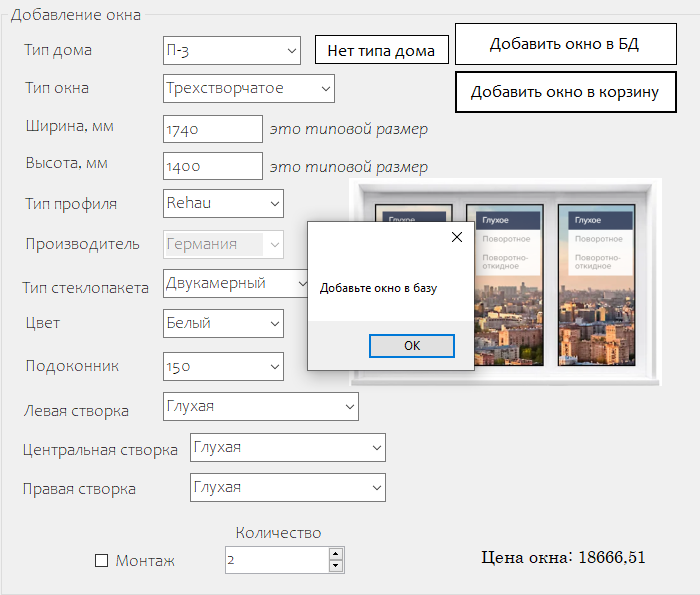


Рисунок 37 – Сообщение о добавлении окна

В правой части формы расположен блок «Корзина», в котором отображаются окна, которые добавлены в заказ, их стоимость и характеристики, а также общая цена заказа, размер скидки, флаг расчета скидки, кнопка очищения корзины, дата заказа, дата выполнения заказа и кнопка «Оформить заказ». Работа программного блока «Корзина» продемонстрирована на рисунке 38.

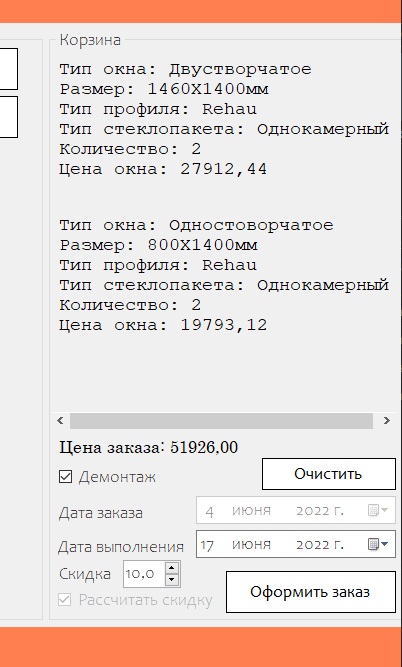


Рисунок 38 – Корзина

После того как менеджер добавил окна в корзину, он может либо оформить заказ, либо очистить корзину с помощью кнопки «Очистить». Чтобы оформить заказа пользователю нужно указать дату выполнения заказа, при желании клиента установить флаг «Демонтаж» и, если есть необходимость рассчитать скидку, затем нажать кнопку «Оформить заказ», после чего выведется сообщение о том, что заказа внесен в базу, как показано на рисунке 39.

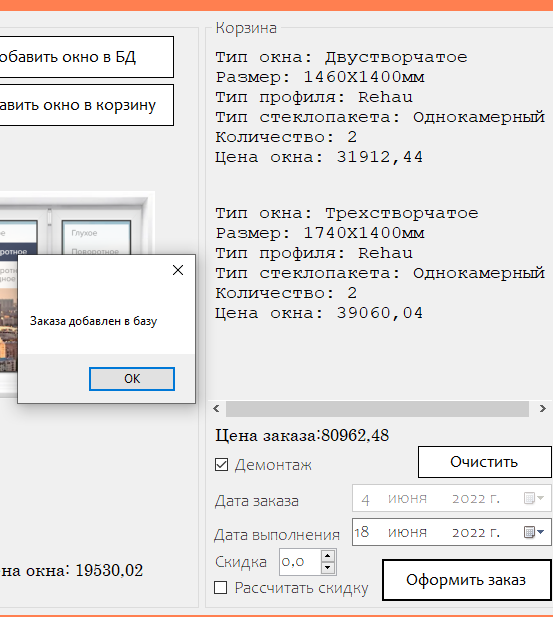


Рисунок 39 – Сообщение о внесении заказа в базу

**Форма «Статус заказа»**

На данной форме менеджер может просмотреть заказы, оформленные им, что продемонстрированно на рисунке 40. Данная форма не имеет элементов управления кроме кнопки «Назад» и служит лишь для отображения заказов. Фрагмент кода формы «Статус заказа» представлен в приложении А.

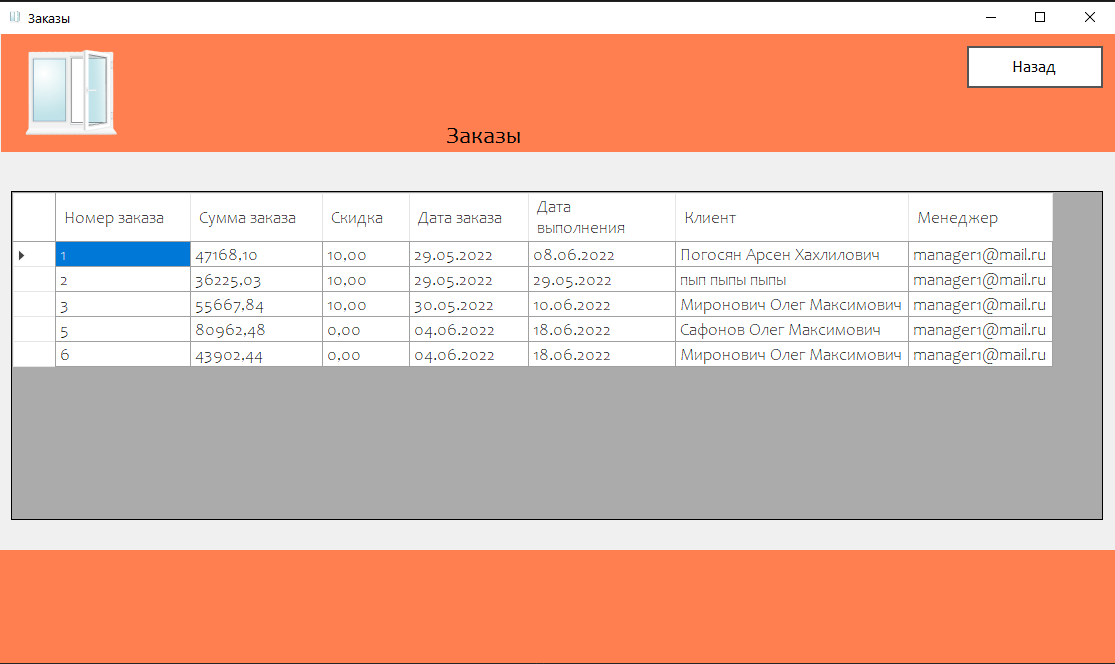


Рисунок 40 – Форма «Статус заказа»

**Форма «Проектирование окон»**

Форма предназначена для проектирования окон нестандартного типа, что представлено на рисунке 41. На форме есть кнопки геометрических фигур, а также кнопки линий и ручка они добавляют картинки на форму. На форме есть кнопка «Справка», нажав на которую можно увидеть подсказку по работе в редакторе. Также на форме есть поля характеристик окна, а именно ширина и высота, тип стеклопакета и размер подоконника. Еще на форме присутствуют кнопки «Добавить в БД» и «Сохранить макет». Фрагмент кода формы продемонстрирован в приложении А.

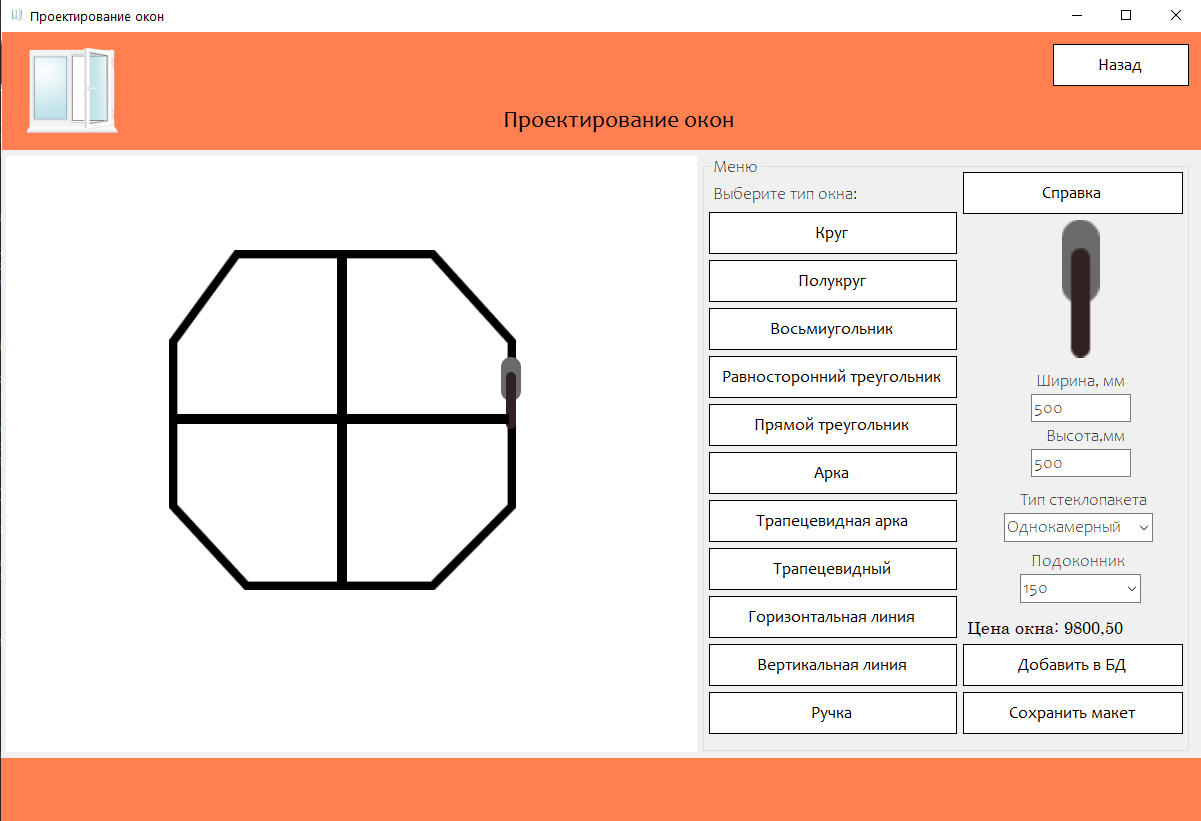


Рисунок 41 – Проектирование окон

Чтобы просмотреть информацию по работе с редактором нужно нажать кнопку «Справка». Результат работы кнопки показан на рисунке Б.2 в приложении Б.

Чтобы добавить окно в базу нужно нажать кнопку «Добавить в БД», после чего должно вывести сообщение об успешном добавлении в базу, что продемонстрировано на рисунке 42.

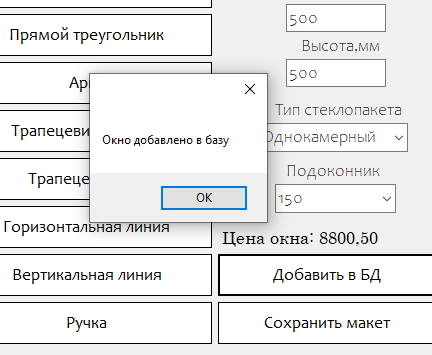


Рисунок 42 – Успешное добавление окна в базу

Если нажать кнопку «Сохранить макет», то откроется диалоговое окно сохранения файла в формате изображения, что представлено в приложении Б на рисунке Б.3.

**Форма «Просмотр нестандартных окон»**

На форме можно просмотреть все спроектированные окна, которые были добавлены в базу, их характеристики и цену, что показано на рисунке 43. Чтобы просмотреть определенное окно нужно лишь нажать на его номер в таблице. Фрагмент кода данной формы представлен в приложении А.

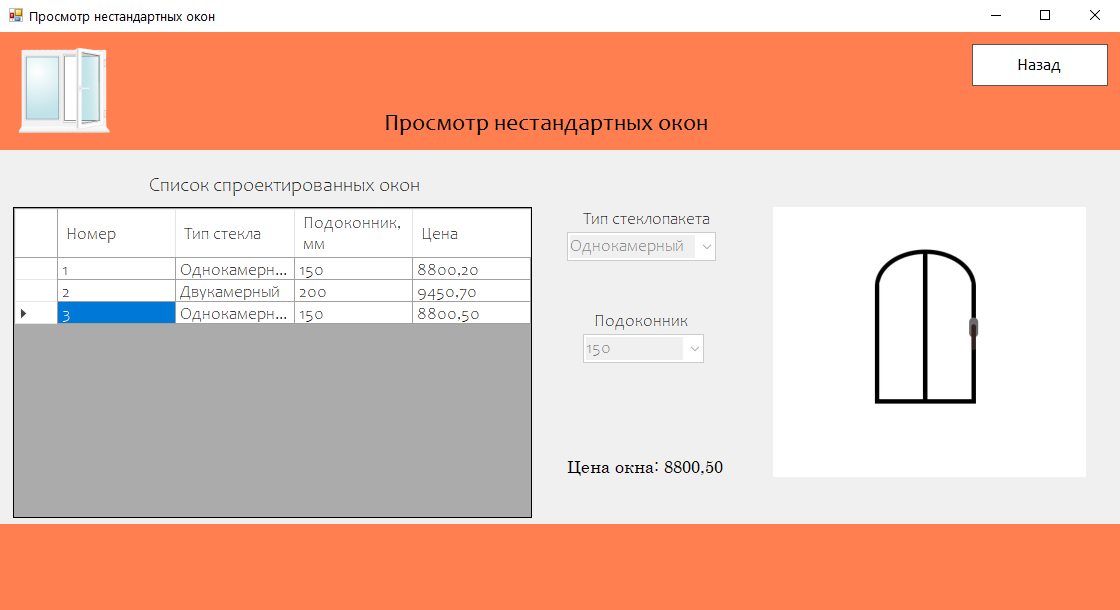


Рисунок 43 – Просмотр нестандартных окон

**Форма «Меню директора»**

Переходим на меню директора, оно служит для навигации директора в приложении. На рисунке 44 продемонстрированно как выглядит данная форма. На форме присутствует 5 кнопок. Кнопка «Назад» перемещает пользователя обратно на окно авторизации, кнопка «Отчетность по менеджерам» открывает форму отчетности менеджеров, кнопка «Изменение цен/товаров» перемещает директора на форму изменения цен и товаров, кнопка «Отчетность по доходам» перемещает пользователя на форму просмотра отчетности компании, кнопка «Добавление менеджера» открывает форму благодаря которой можно добавить нового менеджера в систему. Фрагмент кода формы продемонстрирован в приложении А.

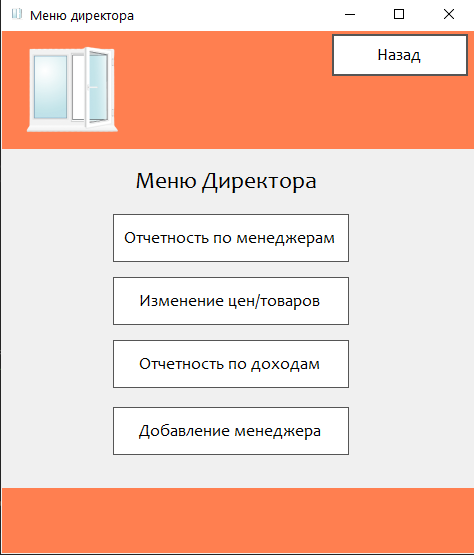


Рисунок 44 – Меню директора

**Форма «Отчетность по менеджерам»**

На этой форме можно выбрать менеджера и временной период, по которому нужна отчетность, информация о менеджере отображается в таблице, что представлено на рисунке 45. Фрагмент кода формы показан в приложении А.

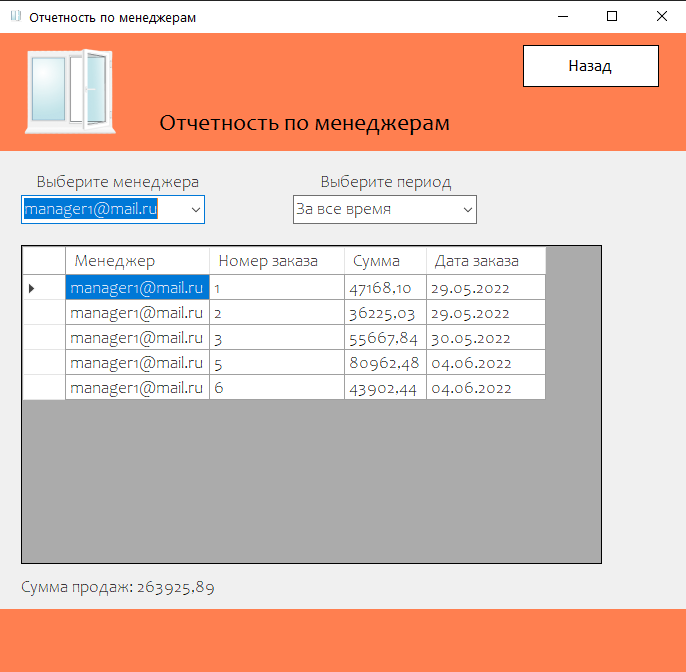


Рисунок 45 – Отчетность по менеджерам

**Форма «Изменение цен/товаров»**

На форме «Изменение/цен товаров» директор может выбрать из таблицы товар, нажав на его номер и тогда информация о товаре отобразиться в блоке «Изменение характеристик товара». В этом блоке пользователь может изменить характеристики окна и его цену, а затем внести изменения, нажав кнопку «Внести изменения», тогда выведется сообщение о том, что изменения внесены. Работа данной формы показана на рисунке 46. Фрагмент кода продемонстрирован в приложении А.

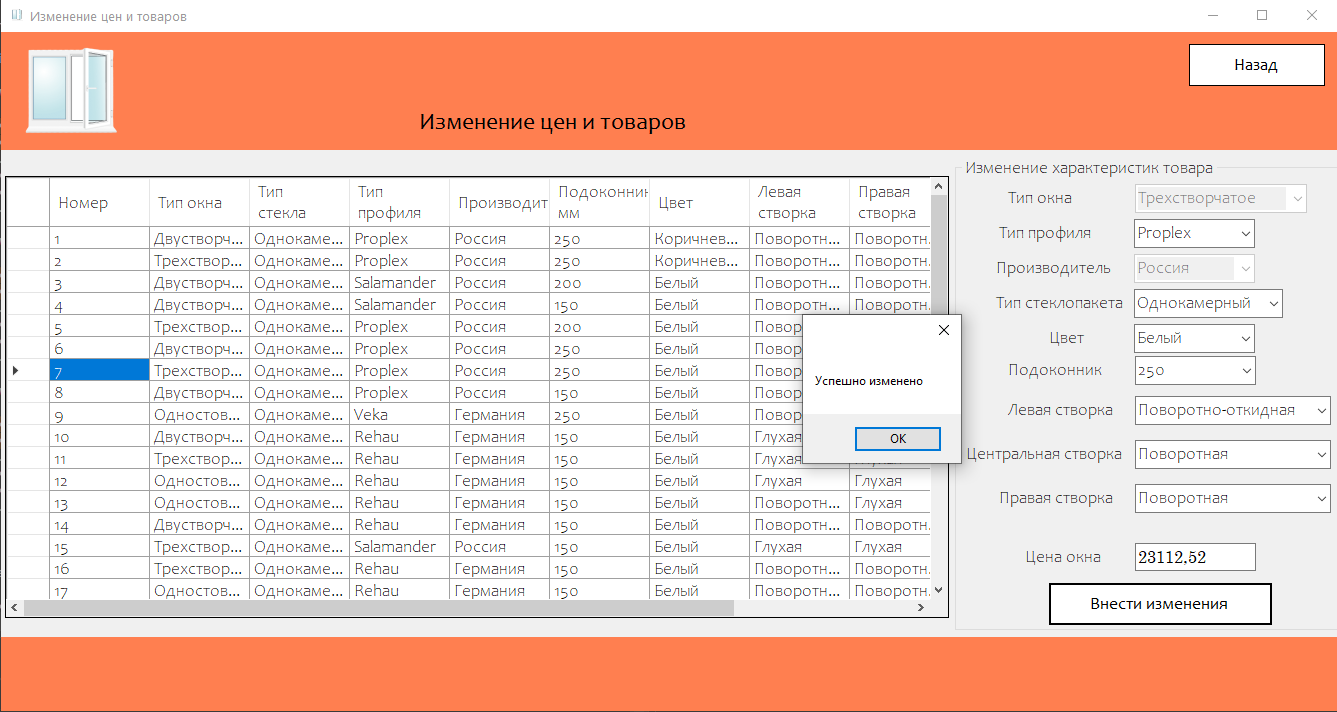


Рисунок 46 – Изменение цен и товаров

**Форма «Отчетность по доходам»**

Данная форма является информационной, на ней отображается количество заказов, количество реализованной продукции и общая выручка компании. Пользователь может выбрать за какой временной период ему нужна информация. Форма отчетности по доходам представлена на рисунке 47. Фрагмент кода данной формы представлен в приложении А.

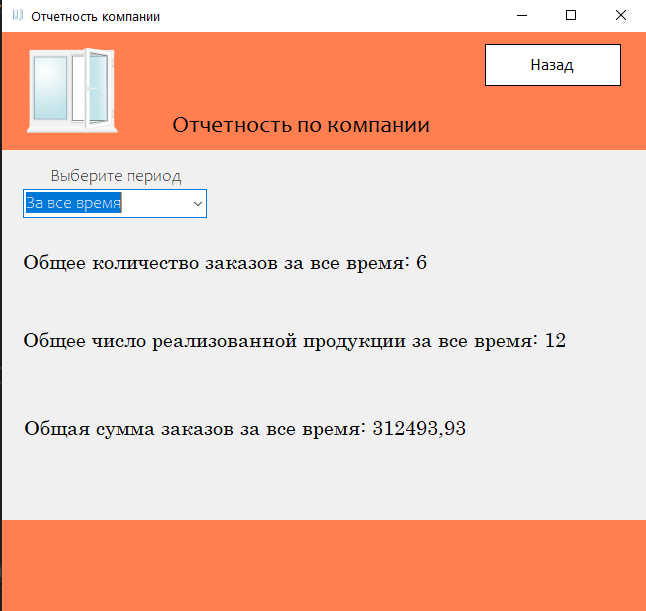


Рисунок 47 – Форма «Отчетность по доходам»

**Форма «Добавления менеджера»**

Данная форма используется директором для того, чтобы добавлять новых менеджеров в систему. В левой части формы есть таблица, в которой отображаются все менеджеры системы, а в правой части расположен блок «Добавления менеджера». В данном блоке есть поле логина и пароля, которые должен заполнить пользователь, чтобы добавить нового менеджера в систему. После заполнения полей нужно нажать «Зарегистрировать», тогда выведется сообщение об успешной регистрации, и пользователь добавится в таблицу, что показано на рисунке 48. Фрагмент кода формы «Добавление менеджера» продемонстрирован в приложении А.

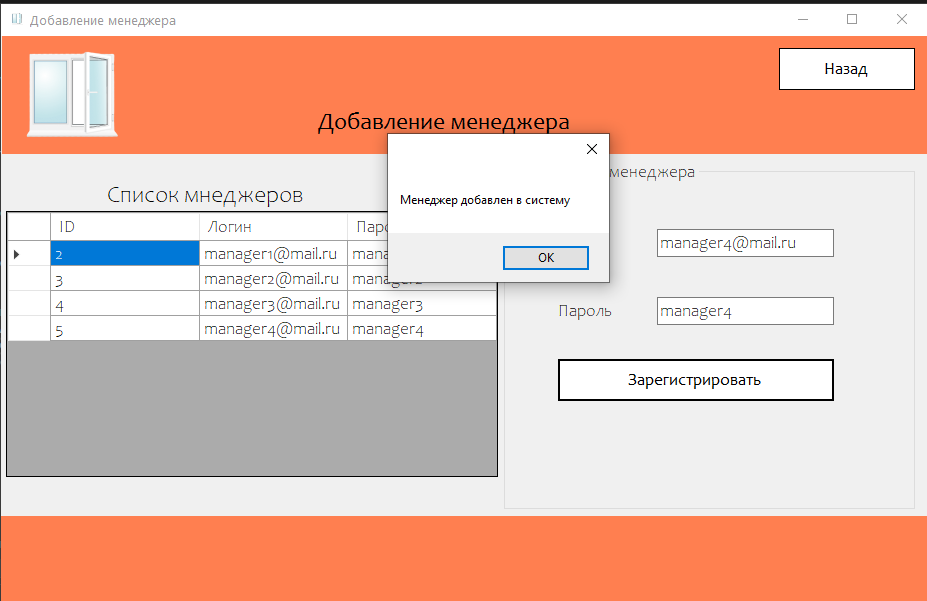


Рисунок 48 – Добавление менеджера

На этом рассмотрение основного функционала приложения законченно. Как можно заметить, работа с приложением не требует дополнительных навыков пользователя. Дружелюбность интерфейса реализована за счет интуитивно понятных переходов между формами через меню и стандартизованных элементов интерфейса.

## 2.7 Анализ результатов разработки

В ходе разработки дипломного проекта приложение «Разработка автоматизированной информационной системы по продаже стеклопакетов», была подробно изучена предметная область разрабатываемого приложения, выявлены основные стоимостные характеристи предмета автоматизации, были изучены требования к работе персонала и их функциональные обязанности, определены функции, которые нуждались в автоматизации.

В итоге был, получен опыт в работе по данной теме, так же были освоены основные приемы работы с базами данных в среде программирования Visual Studio, изучены особенности развертывания, требования по защите данных и способы ее реализации в СУБД MS SQL Server. Важным аспектом, было освоение новых компонентов, с помощью которых осуществлялась связь и отображение информации базы данных. Были реализованы основные способы манипуляции с базами данных, а именно: просмотр, редактирование, поиск и фильтрация по заданным критериям.

База данных, кроме числовых и строковых полей, позволяет хранить и использовать двоичные поля, которые представляются изображениями отдельных записей. Программа снабжена простым интерфейсом, но, тем не менее, были реализованы все необходимые операции с таблицами баз данных.

Особенностью дипломного проекта является применение графических библиотек и классов для реализации программного модуля Конструктора нестандартных окон.

Для реализации приложения были разработаны следующие алгоритмы:

* алгоритм первичной авторизации;
* алгоритм формирования заказов;
* алгоритм добавления менеджеров;
* алгоритм проектирования окон;
* алгоритм добавления спроектированных окон в базу данных.

Для всех форм приложения был разработан интуитивно-понятный интерфейс с классическими элементами управления интерфейсом. Это в значительной степени облегчает использование разработанного проекта даже неопытными пользователями – сотрудниками компании по продаже стеклопакетов.

Были изучены и применены новые технологии, такие как:

* язык программирования С#;
* Субд Microsoft SQL Server.
* возможности новой версии интегрированной среды разработки прикладных программ Visual Studio 2022.

Подводя итоги второго раздела, можно сделать вывод, что разработанный проект представляет собой полностью работоспособное приложение, рабочий функционал которого соответствует требованиям технического задания дипломного проекта.

# Экономическая часть

В данном разделе предоставлено экономическое обоснование созданного программного продукта, а также рассчитана предполагаемая прибыль от его реализации.

## Описание продукта

Данный программный продукт является информационной системой, которая может использоваться директором и менеджерами компании по продаже стеклопакетов. Приложение позволит составлять и отслеживать статусы заказов, просматривать отчетность деятельности фирмы и упростит проектирование окон нестандартных размеров.

Работа с программой начинается с регистрации пользователя, где директор создает своим работникам данные для входа в систему. После получения данных менеджер может авторизоваться в системе и выполнять свои рабочие обязанности, а именно составлять заказы и вносить информацию о них в базу данных.

Директор, же войдя в систему может просматривать отчетность фирмы, а также изменять цены и характеристики товаров.

Программный продукт призван автоматизировать рутинные процессы сотрудников компании, тем самым улучшив их производительность труда. Приложение имеет интуитивно-понятный интерфейс, что позволяет не тратить время на обучение персонала пользованию программой.

## 3.2 Описание и расчет затрат на выполнение проекта

Разработка данной программы, так же, как и разработка любого другого программного продукта, требует определенных материальных, временных и трудовых затрат и, следовательно, должна соответственно окупаться. С экономической точки зрения затраты, связанные с выполнением проекта, должны быть покрыты доходами от реализации конечного продукта.

### 3.2.1 Расчет трудоемкости

Трудоемкость выполнения программного продукта характеризуется перечнем основных этапов и видов работ, которые должны быть выполнены. Упорядочен данный перечень в соответствии со смысловым содержанием каждого вида работ и взаимосвязями между всеми видами работ.

Перед тем как приступить к разработке программного продукта необходимо сформировать техническое задание, которые может быть выдано заказчиком или сформировано самим производителем основываясь на потребности клиентов.

Выделим основные этапы разработки приложения:

1. Разработка и получение технического задания (ТЗ):

2. Подготовительный этап:

– сбор информации;

– определение возможных направлений решения;

– выбор объектного построения программы;

– разработка общей методики создания продукта.

3. Основной этап:

– разработка основного алгоритма;

– создание интерфейса;

– отладка.

4. Завершающий этап:

– подготовка технической документации;

– сдача продукта.

Трудоемкость выполнения работы оценивается в человеко-часах.

Трудоёмкость по видам работ представлена на таблице 8.

**Таблица 8 – Расчет трудоемкости**

|  |  |  |
| --- | --- | --- |
| № п/п | Виды работ | Трудоёмкость |
| 1 | Получение ТЗ | 2 |
| 2 | Сбор информации и ознакомление с предметной областью | 26 |
| 3 | Определение возможных направлений решения | 15 |
| 4 | Выбор объектного построения программы | 10 |
| 5 | Разработка общей методики создания продукта | 10 |

**Окончание таблицы 8**

|  |  |  |
| --- | --- | --- |
| 6 | Разработка основного алгоритма | 67 |
| 7 | Создание интерфейса | 35 |
| 8 | Отладка | 27 |
| 9 | Подготовка технической документации | 12 |
| 10 | Сдача продукта | 2 |
|  | Итого | 206 |

Суммарная трудоемкость составила 206 чел./час.

### 3.2.2 Расчет расходов на заработную плату

На основе данных о трудоемкости и средней заработной плате по отрасли рассчитываем основную заработную плату. Предположим, что заработная плата программиста без опыта работы составляет 24000 тыс. руб. в месяц. (21 рабочий день, 8 часовой рабочий день) или 142,85 руб./час.

Таким образом, расходы на заработную плату по проекту составляют:

ЗП = 142,85 \* 206 = 29427,1 руб.

Отчисления на ЗП (Пенсионный Фонд, Фонд социального страхования, Фонд обязательного мед. страхования, территориальные фонды мед. страхования) составляют 30%.

В денежном выражении составляют: 29427,1 \*0,3 = 8828,13 руб.

### 3.2.3 Расчет затрат на материалы

Материалы, затраченные на создание проекта, приведены в таблице 9.

**Таблица 9 – Расчет затрат на материалы**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Материальные ресурсы | Единицы измерения | Потребляемое количество | Цена за единицу, руб. | Сумма, руб. |
| 1 | Картридж | Шт | 1 | 2150 | 2150 |
| 2 | Бумага формата А4 | упаковка | 1 | 593 | 593 |
| 3 | Папка | Шт | 1 | 258 | 258 |
| 4 | Файлы | упаковка | 2 | 259 | 518 |
| 5 | Диск | Шт | 1 | 120 | 120 |
|  | Итого |  |  |  | 3639 |

### 3.2.4 Расчет амортизации

Стоимость компьютера 53500 тыс. руб. Используем его в течении 5 лет (60 мес.). Годовая амортизация составит в месяц 891,66 руб. В час: 5,31 руб. Умножив на трудоёмкость, определим:

АО= 5,31\*206= 1093,86 руб.

### 3.2.5 Расчет электроэнергии

ПК в среднем употребляет 0,5 кВт/ч. 0,5 \* 206 = 103 кВт. Примерная стоимость 1 кВт/ч около 4,98 руб.

Итого: 103 \* 4,98 =512,94 руб.

### 3.2.6 Расчет смены затрат

Все результаты расчета затрат приведены в таблице 10.

**Таблица 10 – Смета всех затрат**

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование статей затрат | Сумма, руб. |
| 1 | Расходные материалы | 3639 |
| 2 | Основная заработная плата | 29427,1 |
| 3 | Расходы на электроэнергию | 512,94 |
| 4 | Амортизационные отчисления | 1093,86 |
| 5 | Отчисления на ЗП | 8828,13 |
|  | Итого | 43501,03 |

## Расчет предполагаемой прибыли

Экономическим эффектом данного проекта является предполагаемая прибыль, полученная в результате реализации программного продукта.

Суммарная стоимость затрат на создание данного программного продукта составляет 43501,03 руб.

Заказчиком является компания, занимающаяся производством стеклопакетов и их установкой. Внедрение программного продукта позволит автоматизировать задачи управляющего персонала компании, что позволит увеличить трудоемкость работы сотрудников и более эффективно использовать рабочее время.

У данного приложения нет бесплатных аналогов, однако они не позволяют в полной мере реализовать задачу, поставленную заказчиком.

Созданное приложение имеет ряд уникальных особенностей:

– создан удобный и понятный интерфейс;

– удобный в эксплуатации;

– реализована возможность проектирования окон нестандартного размера.

Минимальная стоимость программ конкурентов составляет 80000 руб. Предположим, что потенциальный заказчик готов приобрести созданный программный продукт за 60000 руб. Затраты на разработку проекта составили 43501,03 руб., в случае приобретения клиентом программного продукта за 60000 руб. предполагаемая прибыль составит 16498,97 руб. Рентабельность проекта будет составлять 72% (43501,03 руб. / 60000 руб.).

Данная программа при сравнительно небольших затратах 43501,03 рублей, позволяет автоматизировать, максимально упростить и увеличить производительность работы персонала. Использование данного программного продукта в значительной мере упрощает процесс составления заказа и сокращает время внесения его в базу данных.

Выгодой заказчика является экономия денежных средств при заказе подобного приложения у стороннего лица. Средняя стоимость заказа подобного приложения колеблется от 60000 руб. до 80000. Таким образом, заказчик экономит на создании приложения 20000 руб.

# Заключение

Тема дипломного проекта «Разработка информационной системы для оконного бизнеса» связана с проектированием и реализацией информационной системы, выполняещей авматизацию хранения и обработки информации компании, занимающейся производством и продажей стеклолпакетов.

В результате работы над дипломным проектом была разработана автоматизированная информационная система для выбранной предметной области. В программе разработано одиннадцать основных программных модулей, выполняющих различные действия:

– модуль «Вход в систему»;

– модуль «Меню менеджера»;

– модуль «Меню директора»;

– модуль «Оформления заказа»;

– модуль «Статус заказа»;

– модуль «Проектирование нестандартных окон»;

– модуль «Просмотр заказов на нестандартные окна»;

– модуль «Добавление менеджера»;

– модуль «Отчетность по доходам»;

– модуль «Отчетность по менеджерам»;

– модуль «Изменения цен и товаров».

Программа позволяет систематизировать деятельность менеджера, снизить к минимуму ошибки во время оформления заказов, а также ускорить консультирование клиентов, благодаря наглядной демонстрации описания товара и его изображению. Также программа позволяет директору просматривать статистику менеджеров и отчетность фирмы.

Достоинствами программы являются:

– простой и интуитивно понятный интерфейс;

– минимальное количество ручного ввода информации;

– занимает небольшое пространство на жестком диске;

– выводит на формы всю необходимую информацию;

– обеспечивает фильтрацию и поиск данных таблиц.

К недостаткам программы можно отнести малое количество функций для роли директора. Однако, данный недочет можно устранить в дальнейшей версии данного программного продукта.

При написании дипломного проекта были изучены свойства и методы в среде программирования Microsoft Visual Studio C#, которые позволяют создавать приложения, работающие с полноценными базами данных, обращающиеся к данным через ADO технологию.

Разработанный проект может использоваться компанией по продаже стеклопакетов для консультирования клиентов и реализации продукции.

В дальнейшем в приложение планируется добавить более детализированную систему проектирования окон и новые функции для директора.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Рудаков А. В. Технология разработки программных продуктов. Практикум: учеб. пособие для студ. учреждений сред. проф. образования/А. В. Рудаков, Г. Н. Федорова. – 4-е изд., стер. – М.:Издательский центр «Академия»; 2014. — 192 с.
2. Г. Н. Федорова Разработка, внедрение и адаптация программного обеспечения отраслевой направленности: учебное пособие/Г. Н. Федорова. – Москва: КУРС:ИНФРА-М, 2020 – 336 с. (Среднее профессиональное образование).
3. Гагарина Л. Г. Технология разработки программного обеспечения:учебное пособие/Л. Г. Гагарина, Е. В. Кокорина, Б. Д. Сидорова-Виснадул:под ред. Л. Г. Гагариной. – Москва:ИД «ФОРУМ»:ИНФРА-М, 2021. – 400 с. – (Среднее профессиональное образование).
4. Федорова Г. Н. Осуществление интеграции программных модулей: учебник для студ. учреждений сред. проф. образования / Г. Н. Федорова. 2-е изд., стер. – М.:Издательский центр «Академия», 2018. – 288 с.
5. Перлова О. Н. Проектирование и разработка информационных систем : учебник для студ. учреждений сред. проф. образования/О. П. Ляпина, А. В. Гусева. - 3-е изд., испр. – М.:Издательский центр «Академия», 2020. – 256 с.
6. Зверева В.П. Сопровождение и обслуживание программного обеспечения компьютерных систем : учебник для студ. учреждений сред. проф. образования/В. П. Зверева, А. В. Назаров. – 2-е изд., испр. – М.:Издательский центр «Академия», 2020. -256 с.
7. Голицына О.Л. Основы алгоритмизации и программирования: учебное пособие/О.Л. Голицына, И.И. Попов. — 4-е изд., испр. и доп. — Москва:ФОРУМ:ИНФРА-М, 2021. — 431 с. — (Среднее профессиональное образование).
8. Хорев П.Б. Объектно-ориентированное программирование с примерами на С#:учебное пособие/П.Б. Хорев. — Москва:ФОРУМ: ИНФРА-М, 2021. — 200 с. — (Среднее профессиональное образование).
9. Гуриков С. Р. Введение в программирование на языке Visual C# :  учеб. пособие/С.Р. Гуриков. — М.:ФОРУМ:ИНФРА-М, 2019. — 447 с. — (Среднее профессиональное образование).
10. <https://metanit.com/> - справочные материалы по языкам программирования.
11. <https://docs.microsoft.com/ru-ru/> - техническая документация Майкрософт.
12. <https://habr.com> – справочные материалы по технологиям программирования.
13. <https://nationalteam.worldskills.ru/skills/programmnye-resheniya-dlya-biznesa/> - сайт движения WSR по компетенции «Программные решения для бизнеса».

# Приложение А. Исходный текст программы

**ClassTotal.cs**

namespace Steklo

{

static class ClassTotal

{

static public int idUser;

static public int idRole;

static public string login;

static public int idClient;

static public decimal TotalCost;

static public bool Demontage;

static public bool Montage;

static public decimal sale;

static public int idOrder;

}

}

**Authorization.cs**

namespace Steklo

{

public partial class Form\_Auth : Form

{

StekloPacketDataSet.UsersDataTable dataUsers;

bool visible\_pass = false;

int error\_input;

public Form\_Auth()

{

InitializeComponent();

dataUsers = this.usersTableAdapter1.GetData();

//textBox\_login.Text = "manager1@mail.ru";

//textBox\_pass.Text = "manager1";

textBox\_login.Text = "director@gmail.com";

textBox\_pass.Text = "director";

StartPosition = FormStartPosition.CenterScreen;

}

private void button\_enter\_Click(object sender, EventArgs e)

{

string log, pas;

pas = this.textBox\_pass.Text;

log = this.textBox\_login.Text;

dataUsers = this.usersTableAdapter1.GetData();

var filter = dataUsers.Where(rec => rec.Login == log && rec.Password == pas);

if (filter.Count() == 0) //Нет записей – совпадение логина+пароля не найдено

{

MessageBox.Show("Таких данных нет.");

error\_input++;

switch (error\_input)

{

case 3:

MessageBox.Show("Вы 3 раза подряд ввели неправильные данные для входа. Система заблокирована на 15 секунд");

button\_enter.Enabled = false;

timer1.Tick += new EventHandler(timer1\_Tick);

timer1.Interval = 15000;

timer1.Start();

break;

case 4:

MessageBox.Show("Вы 4 раза подряд ввели неправильные данные для входа. Система заблокирована на 20 секунд");

button\_enter.Enabled = false;

timer1.Tick += new EventHandler(timer1\_Tick);

timer1.Interval = 20000;

timer1.Start();

break;

case 5:

MessageBox.Show("Вы 5 раз подряд ввели неправильные данные для входа. Система автоматически закроется");

Application.Exit();

break;

}

}

else

ClassTotal.idUser = filter.ElementAt(0).IDUser;

ClassTotal.idRole = filter.ElementAt(0).IDRole;

ClassTotal.login = filter.ElementAt(0).Login;

//Переход к формам в зависимости от роли

switch (ClassTotal.idRole)

{

case 2:

MessageBox.Show("Вы успешно авторизовались как менеджер.");

ManagerMenu managerMenu = new ManagerMenu();

this.Hide();

managerMenu.StartPosition = FormStartPosition.CenterScreen;

managerMenu.ShowDialog();

this.Show();

break;

case 1:

MessageBox.Show("Вы успешно авторизовались как директор.");

DirectorMenu directorMenu = new DirectorMenu();

this.Hide();

directorMenu.StartPosition = FormStartPosition.CenterScreen;

directorMenu.ShowDialog();

this.Show();

break;

}

}

private void button\_exit\_Click(object sender, EventArgs e)

{

this.Close();

}

private void pictureBox\_pass\_Click(object sender, EventArgs e)

{

if (visible\_pass == false)

{

this.textBox\_pass.PasswordChar = '\0';

visible\_pass = true;

}

else

{

this.textBox\_pass.PasswordChar = '@';

visible\_pass = false;

}

}

private void timer1\_Tick(object sender, EventArgs e)

{

timer1.Stop();

button\_enter.Enabled = true;

}

}

**ManagerMenu.cs**

namespace Steklo

{

public partial class ManagerMenu : Form

{

public ManagerMenu()

{

InitializeComponent();

}

private void button\_close\_Click(object sender, EventArgs e)

{

this.Close();

}

private void button\_make\_order\_Click(object sender, EventArgs e)

{

MakeOrder makeOrder = new MakeOrder();

this.Hide();

makeOrder.StartPosition = FormStartPosition.CenterScreen;

makeOrder.ShowDialog();

this.Show();

}

private void button\_toOrders\_Click(object sender, EventArgs e)

{

FormOrders formOrders = new FormOrders();

this.Hide();

formOrders.StartPosition = FormStartPosition.CenterScreen;

formOrders.ShowDialog();

this.Show();

}

private void button2\_Click(object sender, EventArgs e)

{

WindowProject win = new WindowProject();

this.Hide();

win.StartPosition = FormStartPosition.CenterScreen;

win.ShowDialog();

this.Show();

}

private void button\_ViewCustomWindows\_Click(object sender, EventArgs e)

{

ViewCustomWindows view = new ViewCustomWindows();

this.Hide();

view.StartPosition = FormStartPosition.CenterScreen;

view.ShowDialog();

this.Show();

}

}

}

**DirectorMenu.cs**

namespace Steklo

{

public partial class DirectorMenu : Form

{

public DirectorMenu()

{

InitializeComponent();

}

private void button\_exit\_Click(object sender, EventArgs e)

{

this.Close();

}

private void button\_manageradd\_Click(object sender, EventArgs e)

{

ManagerAdd managerAdd = new ManagerAdd();

this.Hide();

managerAdd.StartPosition = FormStartPosition.CenterScreen;

managerAdd.ShowDialog();

this.Show();

}

private void button\_otchetManager\_Click(object sender, EventArgs e)

{

ManagersCheck managersCheck = new ManagersCheck();

this.Hide();

managersCheck.StartPosition = FormStartPosition.CenterScreen;

managersCheck.ShowDialog();

this.Show();

}

private void button\_otchetSum\_Click(object sender, EventArgs e)

{

CompanyCheck companyCheck = new CompanyCheck();

this.Hide();

companyCheck.StartPosition = FormStartPosition.CenterScreen;

companyCheck.ShowDialog();

this.Show();

}

private void button\_redakt\_Click(object sender, EventArgs e)

{

ChangeCostandTovars changeCostand = new ChangeCostandTovars();

this.Hide();

changeCostand.StartPosition = FormStartPosition.CenterScreen;

changeCostand.ShowDialog();

this.Show();

}

}

}

**ChangeCostandTovars.cs**

namespace Steklo

{

public partial class ChangeCostandTovars : Form

{

StekloPacketDataSet.WindowsAllDataTable dataWindows;

int selRowNum = 0;

int idWindow, idGlazing, idWindowType, idProfile, idManufacturer, idSill, idColor, idLeftSash, idRightSash, idMediumSash;

private void textBox\_cost\_TextChanged(object sender, EventArgs e)

{

if (textBox\_cost.Text == "")

{

MessageBox.Show("Стоимость не может быть нулевой");

return;

}

}

private void comboBox\_profiletype\_SelectionChangeCommitted(object sender, EventArgs e)

{

idProfile = (int)comboBox\_profiletype.SelectedValue;

//idmanufacturer = (int)profileTypeTableAdapter1.SelectManufacturer(idprofiletype);

if (idProfile == 1 || idProfile == 2)

{

comboBox\_manufacturer.SelectedValue = 1;

comboBox\_manufacturer.Enabled = false;

}

else

{

comboBox\_manufacturer.SelectedValue = 2;

comboBox\_manufacturer.Enabled = false;

}

}

private void textBox\_cost\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if (!Char.IsDigit(number) && number != 8 && number != 44)

{

e.Handled = true;

}

}

private void dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)

{

ShowWindowDetails();

}

decimal cost;

public ChangeCostandTovars()

{

InitializeComponent();

}

private void ChangeCostandTovars\_Load(object sender, EventArgs e)

{

dataWindows = this.windowsAllTableAdapter1.GetData();

dataGridView1.DataSource = dataWindows.CopyToDataTable();

dataGridView1.ReadOnly = true;

comboBox\_windowstype.Enabled = false;

dataGridView1.Columns[0].HeaderText = "Номер";

dataGridView1.Columns[1].HeaderText = "Тип окна";

dataGridView1.Columns[2].HeaderText = "Тип стекла";

dataGridView1.Columns[3].HeaderText = "Тип профиля";

dataGridView1.Columns[4].HeaderText = "Производитель";

dataGridView1.Columns[5].HeaderText = "Подоконник, мм";

dataGridView1.Columns[6].HeaderText = "Цвет";

dataGridView1.Columns[7].HeaderText = "Левая створка";

dataGridView1.Columns[8].HeaderText = "Правая створка";

dataGridView1.Columns[9].HeaderText = "Центральная створка";

dataGridView1.Columns[10].HeaderText = "Цена";

selRowNum = dataGridView1.CurrentCell.RowIndex;

comboBox\_windowstype.DataSource = windowsTypeTableAdapter1.GetData();

comboBox\_windowstype.DisplayMember = "Title";

comboBox\_windowstype.ValueMember = "IDWindowType";

comboBox\_profiletype.DataSource = profileTypeTableAdapter1.GetData();

comboBox\_profiletype.DisplayMember = "Title";

comboBox\_profiletype.ValueMember = "IDProfileType";

comboBox\_manufacturer.DataSource = manufacturerTableAdapter1.GetData();

comboBox\_manufacturer.DisplayMember = "Title";

comboBox\_manufacturer.ValueMember = "IDManufacturer";

comboBox\_manufacturer.Enabled = false;

comboBox\_glazingtype.DataSource = glazingTypeTableAdapter1.GetData();

comboBox\_glazingtype.DisplayMember = "Title";

comboBox\_glazingtype.ValueMember = "IDGlazingType";

comboBox\_glazingtype.SelectedIndex = 0;

comboBox\_color.DataSource = colorsTableAdapter1.GetData();

comboBox\_color.DisplayMember = "Title";

comboBox\_color.ValueMember = "IDColor";

comboBox\_color.SelectedIndex = 0;

comboBox\_sill.DataSource = windowSillTableAdapter1.GetData();

comboBox\_sill.DisplayMember = "Title";

comboBox\_sill.ValueMember = "IDWindowSill";

comboBox\_sill.SelectedIndex = 0;

comboBox\_leftsash.DataSource = leftSashTableAdapter1.GetData();

comboBox\_leftsash.DisplayMember = "Title";

comboBox\_leftsash.ValueMember = "IDLeftSash";

comboBox\_rigthsash.DataSource = rightSashTableAdapter1.GetData();

comboBox\_rigthsash.DisplayMember = "Title";

comboBox\_rigthsash.ValueMember = "IDRightSash";

comboBox\_centrsash.DataSource = mediumSashTableAdapter1.GetDataByMediumSash();

comboBox\_centrsash.DisplayMember = "Title";

comboBox\_centrsash.ValueMember = "IDMediumSash";

comboBox\_centrsash.SelectedValue = 1;

ShowWindowDetails();

}

public void ShowWindowDetails()

{

selRowNum = dataGridView1.CurrentCell.RowIndex;

idWindow = (int)(dataGridView1.Rows[selRowNum].Cells[0].Value);

idWindowType = (int)windowsTypeTableAdapter1.SelectIDWindowType((dataGridView1.Rows[selRowNum].Cells[1].Value).ToString());

idGlazing = (int)glazingTypeTableAdapter1.SelectIDGlazing((dataGridView1.Rows[selRowNum].Cells[2].Value).ToString());

idProfile = (int)profileTypeTableAdapter1.SelectProfileID((dataGridView1.Rows[selRowNum].Cells[3].Value).ToString());

idManufacturer = (int)profileTypeTableAdapter1.SelectManufacturer(idProfile);

idSill = (int)windowSillTableAdapter1.SelectIDSill((dataGridView1.Rows[selRowNum].Cells[5].Value).ToString());

idColor = (int)colorsTableAdapter1.SelectIDColor((dataGridView1.Rows[selRowNum].Cells[6].Value).ToString());

idLeftSash = (int)leftSashTableAdapter1.SelectIDLeft((dataGridView1.Rows[selRowNum].Cells[7].Value).ToString());

idRightSash = (int)rightSashTableAdapter1.SelectIDRight((dataGridView1.Rows[selRowNum].Cells[8].Value).ToString());

idMediumSash = (int)mediumSashTableAdapter1.SelectIDMedium((dataGridView1.Rows[selRowNum].Cells[9].Value).ToString());

cost = (decimal)(dataGridView1.Rows[selRowNum].Cells[10].Value);

if (idWindowType == 1)

{

comboBox\_centrsash.Visible = false;

comboBox\_centrsash.Enabled = true;

label16.Visible = false;

comboBox\_rigthsash.Visible = false;

comboBox\_rigthsash.Enabled = true;

label17.Visible = false;

}

if (idWindowType == 2)

{

comboBox\_centrsash.Visible = false;

comboBox\_centrsash.Enabled = true;

label16.Visible = false;

comboBox\_rigthsash.Visible = true;

comboBox\_rigthsash.Enabled = true;

label17.Visible = true;

}

if (idWindowType == 3)

{

comboBox\_centrsash.Visible = true;

comboBox\_centrsash.Enabled = true;

label16.Visible = true;

comboBox\_rigthsash.Visible = true;

comboBox\_rigthsash.Enabled = true;

label17.Visible = true;

}

if (idManufacturer == 1)

{

comboBox\_manufacturer.SelectedValue = 1;

comboBox\_manufacturer.Enabled = false;

}

else

{

comboBox\_manufacturer.SelectedValue = 2;

comboBox\_manufacturer.Enabled = false;

}

comboBox\_windowstype.SelectedValue = idWindowType;

comboBox\_profiletype.SelectedValue = idProfile;

comboBox\_sill.SelectedValue = idSill;

comboBox\_color.SelectedValue = idColor;

comboBox\_manufacturer.SelectedValue = idManufacturer;

comboBox\_glazingtype.SelectedValue = idGlazing;

comboBox\_rigthsash.SelectedValue = idRightSash;

comboBox\_leftsash.SelectedValue = idLeftSash;

comboBox\_centrsash.SelectedValue = idMediumSash;

textBox\_cost.Text = cost.ToString();

}

private void button\_exit\_Click(object sender, EventArgs e)

{

this.Close();

}

private void button\_update\_Click(object sender, EventArgs e)

{

idWindow = (int)(dataGridView1.Rows[selRowNum].Cells[0].Value);

idWindowType = (int)comboBox\_windowstype.SelectedValue;

idGlazing = (int)comboBox\_glazingtype.SelectedValue;

idProfile = (int)comboBox\_profiletype.SelectedValue;

idManufacturer = (int)comboBox\_manufacturer.SelectedValue;

idSill = (int)comboBox\_sill.SelectedValue;

idColor = (int)comboBox\_color.SelectedValue;

idLeftSash = (int)comboBox\_leftsash.SelectedValue;

idRightSash = (int)comboBox\_rigthsash.SelectedValue;

idMediumSash = (int)comboBox\_centrsash.SelectedValue;

cost = Convert.ToDecimal(textBox\_cost.Text);

if (idWindowType == 1)

{

try

{

windowsTableAdapter1.UpdateWindow(cost, idColor, idManufacturer, idSill, idProfile, idGlazing, idLeftSash, 1, 1, idWindow);

MessageBox.Show("Успешно изменено");

}

catch (Exception ex)

{

MessageBox.Show("Не удалось обновить данные");

}

}

if (idWindowType == 2)

{

try

{

windowsTableAdapter1.UpdateWindow(cost, idColor, idManufacturer, idSill, idProfile, idGlazing, idLeftSash, idRightSash, 1, idWindow);

MessageBox.Show("Успешно изменено");

}

catch (Exception ex)

{

MessageBox.Show("Не удалось обновить данные");

}

}

if (idWindowType == 3)

{

try

{

windowsTableAdapter1.UpdateWindow(cost, idColor, idManufacturer, idSill, idProfile, idGlazing, idLeftSash, idRightSash, idMediumSash, idWindow);

MessageBox.Show("Успешно изменено");

}

catch (Exception ex)

{

MessageBox.Show("Не удалось обновить данные");

}

}

}

}

}

**CompanyCheck.cs**

namespace Steklo

{

public partial class CompanyCheck : Form

{

public CompanyCheck()

{

InitializeComponent();

}

private void button\_exit\_Click(object sender, EventArgs e)

{

this.Close();

}

private void CompanyCheck\_Load(object sender, EventArgs e)

{

comboBox\_period.SelectedIndex = 0;

label\_kolvoorders.Text = "Общее количество заказов за все время: " + ordersCompanyTableAdapter1.SelectAllOrders();

label\_kolvo\_windows.Text = "Общее число реализованной продукции за все время: " + ordersCompanyTableAdapter1.SelectAllCount();

label\_summ.Text = "Общая сумма заказов за все время: " + ordersCompanyTableAdapter1.SelectAllSum();

}

private void comboBox\_period\_SelectionChangeCommitted(object sender, EventArgs e)

{

if (comboBox\_period.SelectedIndex == 0)

{

label\_kolvoorders.Text = "Общее количество заказов за все время: " + ordersCompanyTableAdapter1.SelectAllOrders();

label\_kolvo\_windows.Text = "Общее число реализованной продукции за все время: " + ordersCompanyTableAdapter1.SelectAllCount();

label\_summ.Text = "Общая сумма заказов за все время: " + ordersCompanyTableAdapter1.SelectAllSum();

}

if (comboBox\_period.SelectedIndex == 1)

{

DateTime now = DateTime.Now;

var startDate = new DateTime(now.Year, now.Month, 1);

label\_kolvoorders.Text = "Общее количество заказов за месяц: " + ordersCompanyTableAdapter1.SelectMonthOrders(startDate.ToString("yyyy-dd-MM"));

label\_kolvo\_windows.Text = "Общее число реализованной продукции за месяц: " + ordersCompanyTableAdapter1.SelectAllCount();

label\_summ.Text = "Общая сумма заказов за месяц: " + ordersCompanyTableAdapter1.SelectMonthSum(startDate.ToString("yyyy-dd-MM"));

}

}

}

}

**FormOrders.cs**

namespace Steklo

{

public partial class FormOrders : Form

{

StekloPacketDataSet.OrdersUsersClientDataTable dataOrders;

public FormOrders()

{

InitializeComponent();

}

private void button\_close\_Click(object sender, EventArgs e)

{

this.Close();

}

private void FormOrders\_Load(object sender, EventArgs e)

{

dataOrders = this.ordersUsersClientTableAdapter1.GetData(ClassTotal.idUser);

dataGridView1.DataSource = dataOrders.CopyToDataTable();

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;

dataGridView1.AllowUserToAddRows = false;

dataGridView1.AutoGenerateColumns = false;

dataGridView1.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells;

dataGridView1.Columns[0].HeaderText = "Номер заказа";

dataGridView1.Columns[1].HeaderText = "Сумма заказа";

dataGridView1.Columns[2].HeaderText = "Скидка";

dataGridView1.Columns[3].HeaderText = "Дата заказа";

dataGridView1.Columns[4].HeaderText = "Дата выполнения";

dataGridView1.Columns[5].HeaderText = "Клиент";

dataGridView1.Columns[6].HeaderText = "Менеджер";

}

}

}

**MakeOrder.cs**

namespace Steklo

{

public partial class MakeOrder : Form

{

StekloPacketDataSet.ClientDataTable clients;

StekloPacketDataSet.HomeTypeDataTable homeType;

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

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

Panel panel;

Label label\_windowtype, label\_size, label\_profiletype, label\_glazingtype, label\_costwindow, label\_count\_window;

Button deleteFrommarket;

string dateOrder;

string datePerform;

int IDwindow;

int idwindowinmarket = 0;

int panelid = 0;

int montage = 0, demontage = 0;

int count = 0;

int idClient;

int IDOrder;

string nameClient, surnameClient, patronymicClient, phoneClient, addressClient;

bool refresh = true;

bool clientadded = false;

decimal cost = 0;

int idhometype = 1, idwindowtype = 1, idprofiletype, idmanufacturer, width = 500, height = 500, idglazingtype, idcolor, idsill, idleftsash, idrightsash, idmediumsash;

decimal squarewidth = 0.0M;

decimal squareheight = 0.0M;

decimal summsquare = 0.0M;

decimal totalcost = 0.0M;

decimal sumsale = 0.0M;

private void MakeOrder\_Load(object sender, EventArgs e)

{

comboBox\_client.Enabled = false;

comboBox\_client.Visible = false;

this.flowLayoutPanel1.FlowDirection = FlowDirection.TopDown;

this.flowLayoutPanel1.AutoScroll = true;

this.flowLayoutPanel1.WrapContents = false;

this.flowLayoutPanel1.Font = new Font(FontFamily.GenericMonospace, 12);

this.flowLayoutPanel1.Controls.Clear();

checkBox\_demontage.Checked = true;

pictureBox\_dvyx.Visible = false;

pictureBox\_trx.Visible = false;

pictureBox\_odn.BackgroundImage = Properties.Resources.odn\_glx;

pictureBox\_odn.BackgroundImageLayout = ImageLayout.Zoom;

pictureBox\_dvyx.BackgroundImageLayout = ImageLayout.Zoom;

pictureBox\_trx.BackgroundImageLayout = ImageLayout.Zoom;

ShowHomeTypes();

idhometype = (int)comboBox\_hometype.SelectedValue;

comboBox\_windowstype.DataSource = windowsTypeTableAdapter1.GetData();

comboBox\_windowstype.DisplayMember = "Title";

comboBox\_windowstype.ValueMember = "IDWindowType";

idwindowtype = (int)comboBox\_windowstype.SelectedValue;

comboBox\_profiletype.DataSource = profileTypeTableAdapter1.GetData();

comboBox\_profiletype.DisplayMember = "Title";

comboBox\_profiletype.ValueMember = "IDProfileType";

comboBox\_windowstype.SelectedIndex = 0;

idprofiletype = (int)comboBox\_profiletype.SelectedValue;

comboBox\_manufacturer.DataSource = manufacturerTableAdapter1.GetData();

comboBox\_manufacturer.DisplayMember = "Title";

comboBox\_manufacturer.ValueMember = "IDManufacturer";

comboBox\_windowstype.SelectedIndex = 0;

comboBox\_manufacturer.Enabled = false;

idmanufacturer = (int)comboBox\_manufacturer.SelectedValue;

comboBox\_glazingtype.DataSource = glazingTypeTableAdapter1.GetData();

comboBox\_glazingtype.DisplayMember = "Title";

comboBox\_glazingtype.ValueMember = "IDGlazingType";

comboBox\_glazingtype.SelectedIndex = 0;

idglazingtype = (int)comboBox\_glazingtype.SelectedValue;

comboBox\_color.DataSource = colorsTableAdapter1.GetData();

comboBox\_color.DisplayMember = "Title";

comboBox\_color.ValueMember = "IDColor";

comboBox\_color.SelectedIndex = 0;

idcolor = (int)comboBox\_color.SelectedValue;

comboBox\_sill.DataSource = windowSillTableAdapter1.GetData();

comboBox\_sill.DisplayMember = "Title";

comboBox\_sill.ValueMember = "IDWindowSill";

comboBox\_sill.SelectedIndex = 0;

idsill = (int)comboBox\_sill.SelectedValue;

comboBox\_leftsash.DataSource = leftSashTableAdapter1.GetData();

comboBox\_leftsash.DisplayMember = "Title";

comboBox\_leftsash.ValueMember = "IDLeftSash";

idleftsash = (int)comboBox\_leftsash.SelectedValue;

comboBox\_rigthsash.DataSource = rightSashTableAdapter1.GetData();

comboBox\_rigthsash.DisplayMember = "Title";

comboBox\_rigthsash.ValueMember = "IDRightSash";

comboBox\_centrsash.DataSource = mediumSashTableAdapter1.GetDataByMediumSash();

comboBox\_centrsash.DisplayMember = "Title";

comboBox\_centrsash.ValueMember = "IDMediumSash";

comboBox\_centrsash.Visible = false;

comboBox\_client.DataSource = clientTableAdapter1.GetData();

comboBox\_client.DisplayMember = "FIO";

comboBox\_client.ValueMember = "IDClient";

label\_cost.Text = "Цена окна: " + cost;

}

public void WindowCost()

{

idhometype = (int)comboBox\_hometype.SelectedValue;

idwindowtype = (int)comboBox\_windowstype.SelectedValue;

idglazingtype = (int)comboBox\_glazingtype.SelectedValue;

idsill = (int)comboBox\_sill.SelectedValue;

idcolor = (int)comboBox\_color.SelectedValue;

idleftsash = (int)comboBox\_leftsash.SelectedValue;

idrightsash = (int)comboBox\_rigthsash.SelectedValue;

idmediumsash = (int)comboBox\_centrsash.SelectedValue;

idprofiletype = (int)comboBox\_profiletype.SelectedValue;

idmanufacturer = (int)comboBox\_manufacturer.SelectedValue;

squarewidth = Convert.ToDecimal(width) / 1000;

squareheight = Convert.ToDecimal(height) / 1000;

summsquare = squareheight \* squarewidth;

if (idwindowtype == 1)

{

idrightsash = 1;

idmediumsash = 1;

}

if (summsquare <= 1)

{

switch (idglazingtype)

{

case 1:

cost = 1 \* 5800.50M;

break;

case 2:

cost = 1 \* 6800.70M;

break;

case 3:

cost = 1 \* 8100.10M;

break;

}

}

else

{

switch (idglazingtype)

{

case 1:

cost = summsquare \* 5800.50M;

break;

case 2:

cost = summsquare \* 6800.70M;

break;

case 3:

cost = summsquare \* 8100.10M;

break;

}

}

switch (idsill)

{

case 1:

cost += 500;

break;

case 2:

cost += 650;

break;

case 3:

cost += 750;

break;

case 4:

cost += 900;

break;

case 5:

cost += 1000;

break;

case 6:

cost += 1150;

break;

}

switch (idcolor)

{

case 1:

cost += 0;

break;

case 2:

cost += 250;

break;

case 3:

cost += 500;

break;

case 4:

cost += 1000;

break;

case 5:

cost += 1000;

break;

}

switch (idleftsash)

{

case 1:

cost += 0;

break;

case 2:

cost += 1500;

break;

case 3:

cost += 2000;

break;

}

switch (idrightsash)

{

case 1:

cost += 0;

break;

case 2:

cost += 1500;

break;

case 3:

cost += 2000;

break;

}

switch (idmediumsash)

{

case 1:

cost += 0;

break;

case 2:

cost += 200;

break;

case 3:

cost += 1500;

break;

case 4:

cost += 2000;

break;

}

switch (idprofiletype)

{

case 1:

cost += 1400;

break;

case 2:

cost += 2000;

break;

case 3:

cost += 500;

break;

case 4:

cost += 1000;

break;

}

if (checkBox\_montage.Checked)

{

montage = 1;

}

else

{

montage = 0;

}

switch (montage)

{

case 0:

cost += 0;

break;

case 1:

cost += 2500;

break;

}

label\_cost.Text = "Цена окна: " + Math.Round(cost, 2).ToString();

}

public void ShowHomeTypes()

{

if (refresh == true)

{

comboBox\_hometype.DataSource = homeTypeTableAdapter1.GetData();

comboBox\_hometype.DisplayMember = "Name";

comboBox\_hometype.ValueMember = "IDHomeType";

comboBox\_hometype.SelectedIndex = 0;

refresh = false;

}

}

private void button\_no\_hometype\_Click(object sender, EventArgs e)

{

refresh = true;

AddHomeType addHomeType = new AddHomeType();

this.Hide();

addHomeType.ShowDialog();

this.Show();

}

private void comboBox\_glazingtype\_SelectionChangeCommitted(object sender, EventArgs e)

{

WindowCost();

}

private void comboBox\_sill\_SelectionChangeCommitted(object sender, EventArgs e)

{

WindowCost();

}

private void comboBox\_color\_SelectionChangeCommitted(object sender, EventArgs e)

{

WindowCost();

}

private void comboBox\_leftsash\_SelectionChangeCommitted(object sender, EventArgs e)

{

WindowCost();

WindowShow();

}

private void comboBox\_centrsash\_SelectionChangeCommitted(object sender, EventArgs e)

{

WindowCost();

WindowShow();

}

private void checkBox\_montage\_CheckedChanged(object sender, EventArgs e)

{

WindowCost();

}

private void checkBox\_demontage\_CheckStateChanged(object sender, EventArgs e)

{

if (checkBox\_demontage.Checked == true)

{

demontage = 1;

}

else if (checkBox\_demontage.Checked == false)

{

demontage = 0;

}

switch (demontage)

{

case 0:

totalcost -= 9990;

label\_totalcost.Text = "Цена заказа: " + Math.Round(totalcost, 2);

break;

case 1:

totalcost += 9990;

label\_totalcost.Text = "Цена заказа: " + Math.Round(totalcost,2);

break;

}

}

private void numericUpDown\_sale\_ValueChanged(object sender, EventArgs e)

{

sumsale = numericUpDown\_sale.Value;

}

private void button\_chooseclient\_Click(object sender, EventArgs e)

{

comboBox\_client.Enabled = true;

comboBox\_client.Visible = true;

button\_add\_to\_base.Enabled = false;

}

private void comboBox\_client\_SelectionChangeCommitted(object sender, EventArgs e)

{

idClient = (int)comboBox\_client.SelectedValue;

ClassTotal.idClient = idClient;

clientadded = true;

DataTable dt = new DataTable();

dt = clientTableAdapter1.GetDataByClient(idClient);

maskedTextBox\_phone.Text = dt.Rows[0][2].ToString();

textBox\_address.Text = dt.Rows[0][3].ToString();

string fio = dt.Rows[0][1].ToString();

string[] words = fio.Split(' ');

textBox\_surname.Text = words[0];

textBox\_name.Text = words[1];

textBox\_patron.Text = words[2];

}

private void checkBox1\_CheckedChanged(object sender, EventArgs e)

{

sumsale = numericUpDown\_sale.Value;

sumsale = totalcost \* numericUpDown\_sale.Value / 100;

totalcost = totalcost - sumsale;

label\_totalcost.Text = "Цена заказа: " + Math.Round(totalcost, 2);

checkBox1.Enabled = false;

}

private void comboBox\_profiletype\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void button\_oform\_order\_Click(object sender, EventArgs e)

{

if (clientadded == true)

{

ClassTotal.TotalCost = totalcost;

if (demontage == 0)

{

ClassTotal.Demontage = false;

}

else

{

ClassTotal.Demontage = true;

}

if (demontage == 0)

{

ClassTotal.Montage = false;

}

else

{

ClassTotal.Montage = true;

}

dateOrder = dateTimePicker\_order.Value.ToShortDateString();

datePerform = dateTimePicker\_perform.Value.ToShortDateString();

ClassTotal.sale = numericUpDown\_sale.Value;

ClassTotal.TotalCost = totalcost;

if (ordersTableAdapter1.SelectLastID().ToString() == "")

{

IDOrder = 1;

ClassTotal.idOrder = IDOrder;

}

else

{

IDOrder = (int)(ordersTableAdapter1.SelectLastID() + 1);

ClassTotal.idOrder = IDOrder;

}

try

{

ordersTableAdapter1.InsertOrder(IDOrder, ClassTotal.idClient, ClassTotal.TotalCost, ClassTotal.sale, ClassTotal.Montage, ClassTotal.Demontage, dateOrder.ToString(), datePerform.ToString(), ClassTotal.idUser);

MessageBox.Show("Заказа добавлен в базу");

this.flowLayoutPanel1.Controls.Clear();

IdWindow.Clear();

CountWindow.Clear();

totalcost = 0;

label\_totalcost.Text = "Цена заказа: ";

checkBox1.Enabled = true;

textBox\_address.Text = "";

textBox\_name.Text = "";

textBox\_patron.Text = "";

textBox\_surname.Text = "";

maskedTextBox\_phone.Text = "";

}

catch (Exception ex)

{

MessageBox.Show("Не удалось внести заказ в базу");

}

for (int i = 0; i < IdWindow.Count; i++)

{

orderDetailTableAdapter1.InsertOrderDetail(ClassTotal.idOrder, IdWindow[i], CountWindow[i]);

}

}

else

{

MessageBox.Show("Сначала добавьте клиента");

return;

}

}

private void comboBox\_rigthsash\_SelectionChangeCommitted(object sender, EventArgs e)

{

WindowCost();

WindowShow();

}

private void button\_delAllinmarket\_Click(object sender, EventArgs e)

{

this.flowLayoutPanel1.Controls.Clear();

IdWindow.Clear();

CountWindow.Clear();

totalcost = 0;

label\_totalcost.Text = "Цена заказа: ";

checkBox1.Enabled = true;

}

private void button\_addwindowtoDB\_Click(object sender, EventArgs e)

{

decimal costadded;

DataTable dt = windowsTableAdapter1.GetDataByWindow(idcolor,width,height,idwindowtype,idsill,idprofiletype,idhometype,idglazingtype,idleftsash,idrightsash,idmediumsash);

if (dt.Rows.Count == 0)

{

if (width != 0 && height != 0 && width >= 500 && width <= 2800 && height >= 500 && height <= 2000)

{

costadded = cost;

if (montage == 1)

{

costadded -= 2500;

}

int windowID;

windowID = (int)(windowsTableAdapter1.SelectLastID() + 1);

windowsTableAdapter1.InsertWindow(windowID, cost, idcolor, idmanufacturer, width, height, 1, idwindowtype, idsill, idprofiletype, idhometype, idglazingtype, idleftsash, idrightsash, idmediumsash);

MessageBox.Show("Окно успешно добавлено в базу");

}

else

{

MessageBox.Show("Не удалось добавить в базу");

}

}

else

{

MessageBox.Show("Такое окно есть в базе");

return;

}

}

private void button\_addtomarket\_Click(object sender, EventArgs e)

{

if (width != 0 && height != 0 && width >= 500 && width <= 2800 && height >= 500 && height <= 2000 && textBox\_height.Text != "" && textBox\_width.Text != "" && numericUpDown1.Value !=0)

{

DataTable dt = windowsTableAdapter1.GetDataByWindow(idcolor, width, height, idwindowtype, idsill, idprofiletype, idhometype, idglazingtype, idleftsash, idrightsash, idmediumsash);

if (dt.Rows.Count != 0)

{

IDwindow = (int)windowsTableAdapter1.SelectWindowID(idcolor, width, height, idwindowtype, idsill, idprofiletype, idhometype, idglazingtype, idleftsash, idrightsash, idmediumsash);

count = 0;

count = (int)numericUpDown1.Value;

DataTable dat = windowTypeGlazingandProfile1.GetDataMarket(idglazingtype, idprofiletype, idwindowtype);

string glazing = dat.Rows[0][2].ToString();

string profile = dat.Rows[0][1].ToString();

string window = dat.Rows[0][0].ToString();

cost = cost \* count;

totalcost += cost;

totalcost = Math.Round(totalcost, 2);

label\_totalcost.Text = "Цена заказа:" + totalcost;

//panels.Add(panel);

panel = new Panel();

panel.Size = new Size(this.flowLayoutPanel1.Width, 150);

label\_windowtype = new Label();

label\_windowtype.Location = new Point(0, 0);

label\_windowtype.Size = new Size(this.flowLayoutPanel1.Width, 20);

label\_windowtype.AutoSize = false;

label\_windowtype.Text = "Тип окна: " + window;

panel.Controls.Add(label\_windowtype);

label\_size = new Label();

label\_size.Location = new Point(0, 20);

label\_size.Size = new Size(this.flowLayoutPanel1.Width, 20);

label\_size.AutoSize = false;

label\_size.Text = "Размер: " + width + "X" + height + "мм";

panel.Controls.Add(label\_size);

label\_profiletype = new Label();

label\_profiletype.Location = new Point(0, 40);

label\_profiletype.Size = new Size(this.flowLayoutPanel1.Width, 20);

label\_profiletype.AutoSize = false;

label\_profiletype.Text = "Тип профиля: " + profile;

panel.Controls.Add(label\_profiletype);

label\_glazingtype = new Label();

label\_glazingtype.Location = new Point(0, 60);

label\_glazingtype.Size = new Size(this.flowLayoutPanel1.Width, 20);

label\_glazingtype.AutoSize = false;

label\_glazingtype.Text = "Тип стеклопакета: " + glazing;

panel.Controls.Add(label\_glazingtype);

label\_count\_window = new Label();

label\_count\_window.Location = new Point(0, 80);

label\_count\_window.Size = new Size(this.flowLayoutPanel1.Width, 20);

label\_count\_window.AutoSize = false;

label\_count\_window.Text = "Количество: " + count;

panel.Controls.Add(label\_count\_window);

label\_costwindow = new Label();

label\_costwindow.Location = new Point(0, 100);

label\_costwindow.Size = new Size(this.flowLayoutPanel1.Width, 20);

label\_costwindow.AutoSize = false;

cost = Math.Round(cost, 2);

label\_costwindow.Text = "Цена окна: " + cost;

panel.Controls.Add(label\_costwindow);

IdWindow.Add(IDwindow);

CountWindow.Add(count);

this.flowLayoutPanel1.Controls.Add(panel);

cost = 0;

WindowCost();

}

else

{

MessageBox.Show("Добавьте окно в базу");

return;

}

}

else

{

MessageBox.Show("Укажите правильные значения ширины, высоты и количества окон");

return;

}

}

private void comboBox\_hometype\_SelectionChangeCommitted(object sender, EventArgs e)

{

idhometype = (int)comboBox\_hometype.SelectedValue;

textBox\_height.Text = "";

height = 500;

textBox\_width.Text = "";

width = 500;

ShowHeightAndWidth();

}

private void textBox\_width\_TextChanged(object sender, EventArgs e)

{

if (textBox\_width.Text != "")

{

width = Convert.ToInt32(textBox\_width.Text);

}

if (width < 500)

{

label\_widthtype.Text = "";

label\_width.Text = "Значение ширины не может быть меньше 500";

label\_width.ForeColor = Color.Red;

}

if (width >= 500)

{

label\_width.Text = "";

}

if (width > 2800)

{

label\_widthtype.Text = "";

label\_width.Text = "Значение ширины не может быть больше 2800";

label\_width.ForeColor = Color.Red;

}

if (width <= 2800 && width >= 500)

{

label\_width.Text = "";

}

WindowCost();

}

private void textBox\_height\_TextChanged(object sender, EventArgs e)

{

if (textBox\_height.Text != "")

{

height = Convert.ToInt32(textBox\_height.Text);

}

if (height < 500)

{

label\_heighttype.Text = "";

label\_height.Text = "Значение высоты не может быть меньше 500";

label\_height.ForeColor = Color.Red;

}

if (height >= 500)

{

label\_height.Text = "";

}

if (height > 2000)

{

label\_heighttype.Text = "";

label\_height.Text = "Значение высоты не может быть больше 2000";

label\_height.ForeColor = Color.Red;

}

if (height <= 2000 && height >= 500)

{

label\_height.Text = "";

}

WindowCost();

}

private void textBox\_width\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if (!Char.IsDigit(number) && number != 8)

{

e.Handled = true;

}

}

private void textBox\_height\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if (!Char.IsDigit(number) && number != 8)

{

e.Handled = true;

}

}

private void comboBox\_windowstype\_SelectedIndexChanged(object sender, EventArgs e)

{

if (idwindowtype == 1)

{

comboBox\_centrsash.Visible = false;

comboBox\_centrsash.Enabled = false;

label16.Visible = false;

comboBox\_rigthsash.Visible = false;

comboBox\_rigthsash.Enabled = false;

label17.Visible = false;

}

if (idwindowtype == 2)

{

comboBox\_centrsash.Visible = false;

comboBox\_centrsash.Enabled = false;

label16.Visible = false;

comboBox\_rigthsash.Visible = true;

comboBox\_rigthsash.Enabled = true;

label17.Visible = true;

}

if (idwindowtype == 3)

{

comboBox\_centrsash.Visible = true;

comboBox\_centrsash.Enabled = true;

label16.Visible = true;

comboBox\_rigthsash.Visible = true;

comboBox\_rigthsash.Enabled = true;

label17.Visible = true;

}

}

public void WindowShow()

{

idleftsash = (int)comboBox\_leftsash.SelectedValue;

idrightsash = (int)comboBox\_rigthsash.SelectedValue;

idmediumsash = (int)comboBox\_centrsash.SelectedValue;

idwindowtype = (int)comboBox\_windowstype.SelectedValue;

if (idwindowtype == 1)

{

pictureBox\_odn.Visible = true;

pictureBox\_dvyx.Visible = false;

pictureBox\_trx.Visible = false;

switch (idleftsash)

{

case 1:

pictureBox\_odn.BackgroundImage = Properties.Resources.odn\_glx;

break;

case 2:

pictureBox\_odn.BackgroundImage = Properties.Resources.odn\_pov;

break;

case 3:

pictureBox\_odn.BackgroundImage = Properties.Resources.odn\_pov\_otkd;

break;

}

}

if (idwindowtype == 2)

{

pictureBox\_odn.Visible = false;

pictureBox\_dvyx.Visible = true;

pictureBox\_trx.Visible = false;

if (idleftsash == 1 && idrightsash == 1)

{

pictureBox\_dvyx.BackgroundImage = Properties.Resources.dv\_glx\_glx;

}

else if (idleftsash == 1 && idrightsash == 2)

{

pictureBox\_dvyx.BackgroundImage = Properties.Resources.dv\_glx\_pov;

}

else if (idleftsash == 1 && idrightsash == 3)

{

pictureBox\_dvyx.BackgroundImage = Properties.Resources.dv\_glx\_povotkd;

}

else if (idleftsash == 2 && idrightsash == 1)

{

pictureBox\_dvyx.BackgroundImage = Properties.Resources.dv\_pox\_glx;

}

else if (idleftsash == 3 && idrightsash == 1)

{

pictureBox\_dvyx.BackgroundImage = Properties.Resources.dv\_povotkd\_glx;

}

else if (idleftsash == 2 && idrightsash == 2)

{

pictureBox\_dvyx.BackgroundImage = Properties.Resources.dv\_pov\_pov;

}

else if (idleftsash == 3 && idrightsash == 3)

{

pictureBox\_dvyx.BackgroundImage = Properties.Resources.dv\_povotkd\_povotkd;

}

else if (idleftsash == 2 && idrightsash == 3)

{

pictureBox\_dvyx.BackgroundImage = Properties.Resources.dv\_pov\_povotkd;

}

else if (idleftsash == 3 && idrightsash == 2)

{

pictureBox\_dvyx.BackgroundImage = Properties.Resources.dv\_povotkd\_pov;

}

}

if (idwindowtype == 3)

{

pictureBox\_odn.Visible = false;

pictureBox\_dvyx.Visible = false;

pictureBox\_trx.Visible = true;

if (idleftsash == 1 && idrightsash == 1 && idmediumsash == 2)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.glx\_glx\_glx;

}

else if (idleftsash == 2 && idrightsash == 2 && idmediumsash == 3)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.pov\_pov\_pov;

}

else if (idleftsash == 3 && idrightsash == 3 && idmediumsash == 4)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.otkd\_otkd\_otkd;

}

else if (idleftsash == 1 && idrightsash == 3 && idmediumsash == 4)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.glx\_otkd\_otkd;

}

else if (idleftsash == 1 && idrightsash == 2 && idmediumsash == 3)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.glx\_pov\_pov;

}

else if (idleftsash == 1 && idrightsash == 2 && idmediumsash == 2)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.glx\_glx\_pov;

}

else if (idleftsash == 1 && idrightsash == 3 && idmediumsash == 2)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.glx\_glx\_otkd;

}

else if (idleftsash == 2 && idrightsash == 2 && idmediumsash == 2)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.pov\_glx\_pov;

}

else if (idleftsash == 3 && idrightsash == 3 && idmediumsash == 2)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.otkd\_glx\_otkd;

}

else if (idleftsash == 2 && idrightsash == 3 && idmediumsash == 2)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.pov\_glx\_otkd;

}

else if (idleftsash == 3 && idrightsash == 2 && idmediumsash == 2)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.otkd\_glx\_pov;

}

else if (idleftsash == 2 && idrightsash == 1 && idmediumsash == 3)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.pov\_pov\_glx;

}

else if (idleftsash == 3 && idrightsash == 1 && idmediumsash == 4)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.otkd\_otkd\_glx;

}

else if (idleftsash == 1 && idrightsash == 3 && idmediumsash == 3)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.glx\_pov\_otkd;

}

else if (idleftsash == 2 && idrightsash == 3 && idmediumsash == 4)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.pov\_otkd\_otkd;

}

else if (idleftsash == 2 && idrightsash == 3 && idmediumsash == 3)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.pov\_pov\_otkd;

}

else if (idleftsash == 3 && idrightsash == 2 && idmediumsash == 4)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.otkd\_otkd\_pov;

}

else if (idleftsash == 3 && idrightsash == 2 && idmediumsash == 3)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.otkd\_pov\_pov;

}

else if (idleftsash == 3 && idrightsash == 1 && idmediumsash == 3)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.otkd\_pov\_glx;

}

else if (idleftsash == 2 && idrightsash == 2 && idmediumsash == 4)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.pov\_otkd\_pov;

}

else if (idleftsash == 1 && idrightsash == 1 && idmediumsash == 3)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.glx\_pov\_glx;

}

else if (idleftsash == 3 && idrightsash == 3 && idmediumsash == 3)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.otkd\_pov\_otkd;

}

else if (idleftsash == 3 && idrightsash == 1 && idmediumsash == 2)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.otkd\_glx\_glx;

}

else if (idleftsash == 2 && idrightsash == 1 && idmediumsash == 4)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.pov\_otkd\_glx;

}

else if (idleftsash == 2 && idrightsash == 1 && idmediumsash == 2)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.pov\_glx\_glx;

}

else if (idleftsash == 1 && idrightsash == 2 && idmediumsash == 4)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.glx\_otkd\_pov;

}

else if (idleftsash == 1 && idrightsash == 1 && idmediumsash == 4)

{

pictureBox\_trx.BackgroundImage = Properties.Resources.glx\_otkd\_glx;

}

}

}

private void comboBox\_windowstype\_SelectionChangeCommitted(object sender, EventArgs e)

{

idwindowtype = (int)comboBox\_windowstype.SelectedValue;

label\_widthtype.Text = "";

label\_heighttype.Text = "";

ShowHeightAndWidth();

WindowCost();

WindowShow();

}

public void ShowHeightAndWidth()

{

string str\_height;

string str\_width;

str\_width = windowsTableAdapter1.SelectWidth(idhometype, idwindowtype).ToString();

str\_height = windowsTableAdapter1.SelectHeight(idhometype, idwindowtype).ToString();

if (str\_width != "")

{

textBox\_width.Text = str\_width;

label\_widthtype.Text = "это типовой размер";

}

if (str\_height != "")

{

textBox\_height.Text = str\_height;

label\_heighttype.Text = "это типовой размер";

}

}

private void comboBox\_profiletype\_SelectionChangeCommitted(object sender, EventArgs e)

{

idprofiletype = (int)comboBox\_profiletype.SelectedValue;

idmanufacturer = (int)profileTypeTableAdapter1.SelectManufacturer(idprofiletype);

if (idmanufacturer == 1)

{

comboBox\_manufacturer.SelectedValue = 1;

comboBox\_manufacturer.Enabled = false;

}

else

{

comboBox\_manufacturer.SelectedValue = 2;

comboBox\_manufacturer.Enabled = false;

}

WindowCost();

}

private void comboBox\_hometype\_DropDown(object sender, EventArgs e)

{

ShowHomeTypes();

}

private void textBox\_phone\_KeyPress(object sender, KeyPressEventArgs e)

{

}

public MakeOrder()

{

InitializeComponent();

}

private void button\_close\_Click(object sender, EventArgs e)

{

this.Close();

}

private void button\_add\_to\_base\_Click(object sender, EventArgs e)

{

surnameClient = textBox\_surname.Text;

nameClient = textBox\_name.Text;

patronymicClient = textBox\_patron.Text;

phoneClient = maskedTextBox\_phone.Text;

addressClient = textBox\_address.Text;

string FIOClient;

if (surnameClient == "" || nameClient == "" || patronymicClient == "" || phoneClient == "" || addressClient == "")

{

MessageBox.Show("Вы не ввели одно из значений!!");

return;

}

else

{

FIOClient = surnameClient + " " + nameClient + " " + patronymicClient;

try

{

clientTableAdapter1.InsertClient(FIOClient, phoneClient, addressClient);

MessageBox.Show("Клиент добавлен в базу");

clientadded = true;

ClassTotal.idClient = (int)clientTableAdapter1.SelectIDClient(phoneClient,addressClient,FIOClient);

}

catch (Exception ex)

{

MessageBox.Show("Ошибка добавления клиента в базу");

return;

}

}

}

}

}

**ManagerAdd.cs**

namespace Steklo

{

public partial class ManagerAdd : Form

{

StekloPacketDataSet.UsersDataTable dataUsers;

public ManagerAdd()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

string pass;

string login;

if (textBox\_login.Text != "" || textBox\_password.Text != "")

{

login = textBox\_login.Text;

pass = textBox\_password.Text;

DataTable dt = usersTableAdapter1.GetDataByManager(login,pass);

if (dt.Rows.Count == 0)

{

int idmanager = (int)(usersTableAdapter1.SelectLastID() + 1);

try

{

usersTableAdapter1.InsertManager(idmanager, login, pass, 2);

dataUsers = this.usersTableAdapter1.GetData();

dataGridView1.DataSource = dataUsers.Where(x => x.IDRole > 1).CopyToDataTable();

MessageBox.Show("Менеджер добавлен в систему");

}

catch (Exception ex)

{

MessageBox.Show("Не удалось добавить пользователя");

}

}

else

{

MessageBox.Show("Менеджер с такими данными уже есть");

return;

}

}

else

{

MessageBox.Show("Введите логин и пароль");

return;

}

}

private void ManagerAdd\_Load(object sender, EventArgs e)

{

dataUsers = this.usersTableAdapter1.GetData();

dataGridView1.DataSource = dataUsers.Where(x => x.IDRole > 1).CopyToDataTable();

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

dataGridView1.AllowUserToAddRows = false;

dataGridView1.AutoGenerateColumns = false;

dataGridView1.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells;

dataGridView1.Columns[0].HeaderText = "ID";

dataGridView1.Columns[1].HeaderText = "Логин";

dataGridView1.Columns[2].HeaderText = "Пароль";

dataGridView1.Columns[3].Visible = false;

}

private void button\_exit\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

**ManagersCheck.cs**

namespace Steklo

{

public partial class ManagersCheck : Form

{

StekloPacketDataSet.OrdersManagersDataTable ordersManagers;

StekloPacketDataSet.UsersDataTable dataUsers;

int idManager;

decimal summ;

public ManagersCheck()

{

InitializeComponent();

}

private void button\_exit\_Click(object sender, EventArgs e)

{

this.Close();

}

private void ManagersCheck\_Load(object sender, EventArgs e)

{

dataUsers = this.usersTableAdapter1.GetData();

comboBox\_manager.DataSource = dataUsers.Where(x => x.IDRole > 1).CopyToDataTable();

comboBox\_manager.DisplayMember = "Login";

comboBox\_manager.ValueMember = "IDUser";

comboBox\_manager.SelectedIndex = 0;

idManager = (int)comboBox\_manager.SelectedValue;

comboBox\_period.SelectedIndex = 0;

ordersManagers = this.ordersManagersTableAdapter1.GetData(idManager);

dataGridView1.DataSource = ordersManagers.CopyToDataTable();

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;

dataGridView1.AllowUserToAddRows = false;

dataGridView1.AutoGenerateColumns = false;

dataGridView1.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells;

dataGridView1.Columns[0].HeaderText = "Менеджер";

dataGridView1.Columns[1].HeaderText = "Номер заказа";

dataGridView1.Columns[2].HeaderText = "Сумма";

dataGridView1.Columns[3].HeaderText = "Дата заказа";

summ = 0;

for (int i = 0; i < dataGridView1.RowCount; i++)

{

summ += (decimal)dataGridView1.Rows[i].Cells[2].Value;

}

label\_summ.Text = "Сумма продаж: " + summ;

}

private void comboBox\_manager\_SelectionChangeCommitted(object sender, EventArgs e)

{

idManager = (int)comboBox\_manager.SelectedValue;

ordersManagers = this.ordersManagersTableAdapter1.GetData(idManager);

dataGridView1.DataSource = ordersManagers.CopyToDataTable();

summ = 0;

for (int i = 0; i < dataGridView1.RowCount; i++)

{

summ += (decimal)dataGridView1.Rows[i].Cells[2].Value;

}

label\_summ.Text = "Сумма продаж: " + summ;

}

private void comboBox\_period\_SelectionChangeCommitted(object sender, EventArgs e)

{

DateTime now = DateTime.Now;

var startDate = new DateTime(now.Year, now.Month, 1);

var endDate = startDate.AddMonths(1).AddDays(-1);

if (comboBox\_period.SelectedIndex == 0)

{

idManager = (int)comboBox\_manager.SelectedValue;

ordersManagers = this.ordersManagersTableAdapter1.GetData(idManager);

dataGridView1.DataSource = ordersManagers.CopyToDataTable();

summ = 0;

for (int i = 0; i < dataGridView1.RowCount; i++)

{

summ += (decimal)dataGridView1.Rows[i].Cells[2].Value;

}

label\_summ.Text = "Сумма продаж: " + summ;

}

if (comboBox\_period.SelectedIndex == 1)

{

idManager = (int)comboBox\_manager.SelectedValue;

ordersManagers = this.ordersManagersTableAdapter1.GetDataByMonth(startDate.ToString("yyyy-dd-MM"), idManager);

dataGridView1.DataSource = ordersManagers.CopyToDataTable();

summ = 0;

for (int i = 0; i < dataGridView1.RowCount; i++)

{

summ += (decimal)dataGridView1.Rows[i].Cells[2].Value;

}

label\_summ.Text = "Сумма продаж: " + summ;

}

}

}

}

**ViewCustomWindows.cs**

namespace Steklo

{

public partial class ViewCustomWindows : Form

{

StekloPacketDataSet.CustomWindowsGlazingSillDataTable dataWindows;

int selRowNum = 0;

int idSill, idGlazing, idWindow;

decimal cost;

public ViewCustomWindows()

{

InitializeComponent();

}

private void button\_close\_Click(object sender, EventArgs e)

{

this.Close();

}

private void dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)

{

ShowWindowDetails();

}

private void ViewCustomWindows\_Load(object sender, EventArgs e)

{

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

comboBox\_glazingtype.Enabled = false;

comboBox\_sill.Enabled = false;

dataWindows = this.customWindowsGlazingSillTableAdapter1.GetData();

dataGridView1.DataSource = dataWindows.CopyToDataTable();

dataGridView1.ReadOnly = true;

dataGridView1.Columns[0].HeaderText = "Номер";

dataGridView1.Columns[1].HeaderText = "Тип стекла";

dataGridView1.Columns[2].HeaderText = "Подоконник, мм";

dataGridView1.Columns[3].HeaderText = "Цена";

selRowNum = dataGridView1.CurrentCell.RowIndex;

comboBox\_glazingtype.DataSource = glazingTypeTableAdapter1.GetData();

comboBox\_glazingtype.DisplayMember = "Title";

comboBox\_glazingtype.ValueMember = "IDGlazingType";

comboBox\_sill.DataSource = windowSillTableAdapter1.GetData();

comboBox\_sill.DisplayMember = "Title";

comboBox\_sill.ValueMember = "IDWindowSill";

ShowWindowDetails();

}

public void ShowWindowDetails()

{

selRowNum = dataGridView1.CurrentCell.RowIndex;

idWindow = (int)dataGridView1.Rows[selRowNum].Cells[0].Value;

idGlazing = (int)glazingTypeTableAdapter1.SelectIDGlazing((dataGridView1.Rows[selRowNum].Cells[1].Value).ToString());

idSill = (int)windowSillTableAdapter1.SelectIDSill((dataGridView1.Rows[selRowNum].Cells[2].Value).ToString());

cost = (decimal)(dataGridView1.Rows[selRowNum].Cells[3].Value);

comboBox\_sill.SelectedValue = idSill;

comboBox\_glazingtype.SelectedValue = idGlazing;

byte[] photo = customWindowsTableAdapter1.SelectPhoto(idWindow);

MemoryStream stream = new MemoryStream(photo);

Image bit = Image.FromStream(stream);

pictureBox2.BackgroundImage = bit;

pictureBox2.BackgroundImageLayout = ImageLayout.Zoom;

label\_cost.Text = "Цена окна: " + cost;

}

}

}

**WindowProject.cs**

namespace Steklo

{

public partial class WindowProject : Form

{

PictureBox pb;

int x0, y0;

bool drag = false;

bool picadd = false;

int sashCount = 0;

int idglazingtype = 1;

int idsill = 1;

int horizLineCount = 0;

int vertikLineCount = 0;

int Width = 500, Height = 500;

Point last\_pos = new Point();

decimal cost;

decimal summsquare, squareheight, squarewidth;

public WindowProject()

{

InitializeComponent();

textBoxHeight.Text = "500";

textBoxWidth.Text = "500";

}

private void button\_info\_Click(object sender, EventArgs e)

{

MessageBox.Show("Цена любого типа окна расчитывается по площади прямоугольника в большую сторону." + Environment.NewLine +

"Чтобы увеличить линию нажмите на нее и прокрутите колесико вверх, а" + Environment.NewLine + "чтобы уменьшить прокрутите колесико вниз." + Environment.NewLine +

"Чтобы удалить ручку или линию нажмите на нее быстро два раза.");

}

private void button\_exit\_Click(object sender, EventArgs e)

{

this.Close();

}

private void button\_elipse\_Click(object sender, EventArgs e)

{

picadd = false;

try

{

pictureBox3.Controls.Clear();

sashCount = 0;

vertikLineCount = 0;

horizLineCount = 0;

cost = 0;

WindowCost();

labelCost.Text = "Цена окна: " + Math.Round(cost, 2);

pictureBox3.BackgroundImage = Properties.Resources.elipse;

pictureBox3.BackgroundImageLayout = ImageLayout.Zoom;

pictureBox3.SendToBack();

pictureBox2.BackgroundImageLayout = ImageLayout.Zoom;

pictureBox2.BackgroundImage = Properties.Resources.elipse;

picadd = true;

}

catch

{

MessageBox.Show("Вы не добавили предмет", "Ошибка");

}

}

private void Pb\_doubleClick(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

PictureBox pb = sender as PictureBox;

if ( pb.Width == 20 && sashCount != 0)

{

cost -= 1500;

labelCost.Text = "Цена окна: " + Math.Round(cost,2);

sashCount--;

}

if (pb.Height == 10 && horizLineCount != 0)

{

cost -= 1000;

labelCost.Text = "Цена окна: " + Math.Round(cost, 2);

horizLineCount--;

}

if (pb.Width == 10 && vertikLineCount != 0)

{

cost -= 1000;

labelCost.Text = "Цена окна: " + Math.Round(cost, 2);

vertikLineCount--;

}

this.pictureBox3.Controls.Remove(pb);

}

}

private void Pb\_MouseMove(object sender, MouseEventArgs e)

{

PictureBox pb = (PictureBox)sender;

if (drag)

{

pb.Left += (e.X - x0);

pb.Top += (e.Y - y0);

}

}

private void Pb\_MouseDown(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

drag = true;

x0 = e.X;

y0 = e.Y;

last\_pos = e.Location;

}

}

private void Pb\_MouseUp(object sender, MouseEventArgs e)

{

drag = false;

}

private void button\_halfelipse\_Click(object sender, EventArgs e)

{

picadd = false;

try

{

pictureBox3.Controls.Clear();

sashCount = 0;

vertikLineCount = 0;

horizLineCount = 0;

cost = 0;

WindowCost();

labelCost.Text = "Цена окна: " + Math.Round(cost, 2);

pictureBox3.BackgroundImage = Properties.Resources.halfelipse;

pictureBox3.BackgroundImageLayout = ImageLayout.Zoom;

pictureBox3.SendToBack();

pictureBox2.BackgroundImageLayout = ImageLayout.Zoom;

pictureBox2.BackgroundImage = Properties.Resources.halfelipse;

picadd = true;

}

catch

{

MessageBox.Show("Вы не добавили предмет", "Ошибка");

}

}

private void button\_eight\_Click(object sender, EventArgs e)

{

picadd = false;

try

{

pictureBox3.Controls.Clear();

sashCount = 0;

vertikLineCount = 0;

horizLineCount = 0;

cost = 0;

WindowCost();

labelCost.Text = "Цена окна: " + Math.Round(cost, 2);

pictureBox3.BackgroundImage = Properties.Resources.eight1;

pictureBox3.BackgroundImageLayout = ImageLayout.Zoom;

pictureBox3.SendToBack();

pictureBox2.BackgroundImageLayout = ImageLayout.Zoom;

pictureBox2.BackgroundImage = Properties.Resources.eight1;

picadd = true;

}

catch

{

MessageBox.Show("Вы не добавили предмет", "Ошибка");

}

}

private void button\_triangleravn\_Click(object sender, EventArgs e)

{

picadd = false;

try

{

pictureBox3.Controls.Clear();

sashCount = 0;

vertikLineCount = 0;

horizLineCount = 0;

cost = 0;

WindowCost();

labelCost.Text = "Цена окна: " + Math.Round(cost, 2);

pictureBox3.BackgroundImage = Properties.Resources.treugravn;

pictureBox3.BackgroundImageLayout = ImageLayout.Zoom;

pictureBox3.SendToBack();

pictureBox2.BackgroundImageLayout = ImageLayout.Zoom;

pictureBox2.BackgroundImage = Properties.Resources.treugravn;

picadd = true;

}

catch

{

MessageBox.Show("Вы не добавили предмет", "Ошибка");

}

}

private void button\_trianglepryam\_Click(object sender, EventArgs e)

{

picadd = false;

try

{

pictureBox3.Controls.Clear();

sashCount = 0;

vertikLineCount = 0;

horizLineCount = 0;

cost = 0;

WindowCost();

labelCost.Text = "Цена окна: " + Math.Round(cost, 2);

pictureBox3.BackgroundImage = Properties.Resources.treugpryam1;

pictureBox3.BackgroundImageLayout = ImageLayout.Zoom;

pictureBox3.SendToBack();

pictureBox2.BackgroundImageLayout = ImageLayout.Zoom;

pictureBox2.BackgroundImage = Properties.Resources.treugpryam1;

picadd = true;

}

catch

{

MessageBox.Show("Вы не добавили предмет", "Ошибка");

}

}

private void button\_arc\_Click(object sender, EventArgs e)

{

picadd = false;

try

{

pictureBox3.Controls.Clear();

sashCount = 0;

vertikLineCount = 0;

horizLineCount = 0;

cost = 0;

WindowCost();

labelCost.Text = "Цена окна: " + Math.Round(cost, 2);

pictureBox3.BackgroundImage = Properties.Resources.arc;

pictureBox3.BackgroundImageLayout = ImageLayout.Zoom;

pictureBox3.SendToBack();

pictureBox2.BackgroundImageLayout = ImageLayout.Zoom;

pictureBox2.BackgroundImage = Properties.Resources.arc;

picadd = true;

}

catch

{

MessageBox.Show("Вы не добавили предмет", "Ошибка");

}

}

private void button\_trapezoidarc\_Click(object sender, EventArgs e)

{

picadd = false;

try

{

pictureBox3.Controls.Clear();

sashCount = 0;

vertikLineCount = 0;

horizLineCount = 0;

cost = 0;

WindowCost();

labelCost.Text = "Цена окна: " + Math.Round(cost, 2);

pictureBox3.BackgroundImage = Properties.Resources.trapezy1;

pictureBox3.BackgroundImageLayout = ImageLayout.Zoom;

pictureBox3.SendToBack();

pictureBox2.BackgroundImageLayout = ImageLayout.Zoom;

pictureBox2.BackgroundImage = Properties.Resources.trapezy1;

picadd = true;

}

catch

{

MessageBox.Show("Вы не добавили предмет", "Ошибка");

}

}

private void button\_trapezoid\_Click(object sender, EventArgs e)

{

picadd = false;

try

{

pictureBox3.Controls.Clear();

sashCount = 0;

vertikLineCount = 0;

horizLineCount = 0;

cost = 0;

WindowCost();

labelCost.Text = "Цена окна: " + Math.Round(cost, 2);

pictureBox3.BackgroundImage = Properties.Resources.trapezypryam1;

pictureBox3.BackgroundImageLayout = ImageLayout.Zoom;

pictureBox3.SendToBack();

pictureBox2.BackgroundImageLayout = ImageLayout.Zoom;

pictureBox2.BackgroundImage = Properties.Resources.trapezypryam1;

picadd = true;

}

catch

{

MessageBox.Show("Вы не добавили предмет", "Ошибка");

}

}

private void button\_horizline\_Click(object sender, EventArgs e)

{

try

{

pb = new PictureBox();

pb.Width = 270;

pb.Height = 10;

pb.Parent = pictureBox3;

pb.Image = Properties.Resources.png\_line\_hor;

pb.Left = 175;

pb.Top = 173;

pb.BackgroundImageLayout = ImageLayout.Zoom;

pictureBox2.BackgroundImageLayout = ImageLayout.Zoom;

pictureBox2.BackgroundImage = Properties.Resources.png\_line\_hor;

pb.BackColor = Color.Transparent;

pb.BringToFront();

this.pictureBox3.Controls.Add(pb);

horizLineCount++;

WindowCost();

pb.MouseClick += Pb\_Horiz\_Click;

pb.MouseDoubleClick += Pb\_doubleClick;

pb.MouseMove += Pb\_MouseMove;

pb.MouseDown += Pb\_MouseDown;

pb.MouseUp += Pb\_MouseUp;

pb.MouseWheel += Pb\_Mouse\_WheelHoriz;

picadd = true;

}

catch

{

MessageBox.Show("Вы не добавили предмет", "Ошибка");

}

}

public void WindowCost()

{

squarewidth = Convert.ToDecimal(Width) / 1000;

squareheight = Convert.ToDecimal(Height) / 1000;

summsquare = squareheight \* squarewidth;

if (summsquare <= 1)

{

switch (idglazingtype)

{

case 1:

cost = 1 \* 5800.50M;

break;

case 2:

cost = 1 \* 6800.70M;

break;

case 3:

cost = 1 \* 8100.10M;

break;

}

}

else

{

switch (idglazingtype)

{

case 1:

cost = summsquare \* 5800.50M;

break;

case 2:

cost = summsquare \* 6800.70M;

break;

case 3:

cost = summsquare \* 8100.10M;

break;

}

}

switch (idsill)

{

case 1:

cost += 500;

break;

case 2:

cost += 650;

break;

case 3:

cost += 750;

break;

case 4:

cost += 900;

break;

case 5:

cost += 1000;

break;

case 6:

cost += 1150;

break;

}

cost += 1500.00M \* sashCount;

cost += 1000.00M \* horizLineCount;

cost += 1000.00M \* vertikLineCount;

labelCost.Text = "Цена окна: " + Math.Round(cost, 2);

}

private void button\_verticline\_Click(object sender, EventArgs e)

{

try

{

pb = new PictureBox();

pb.Width = 10;

pb.Height = 270;

pb.Parent = pictureBox3;

pb.Image = Properties.Resources.png\_line\_vert;

pb.Left = 175;

pb.Top = 173;

pb.BackgroundImageLayout = ImageLayout.Zoom;

pictureBox2.BackgroundImageLayout = ImageLayout.Zoom;

pictureBox2.BackgroundImage = Properties.Resources.png\_line\_vert;

pb.BackColor = Color.Transparent;

pb.BringToFront();

this.pictureBox3.Controls.Add(pb);

vertikLineCount ++;

WindowCost();

pb.MouseClick += Pb\_Vertik\_Click;

pb.MouseDoubleClick += Pb\_doubleClick;

pb.MouseMove += Pb\_MouseMove;

pb.MouseDown += Pb\_MouseDown;

pb.MouseUp += Pb\_MouseUp;

pb.MouseWheel += Pb\_Mouse\_WheelVertik;

picadd = true;

}

catch

{

MessageBox.Show("Вы не добавили предмет", "Ошибка");

}

}

private void button\_sash\_Click(object sender, EventArgs e)

{

try

{

pb = new PictureBox();

pb.Width = 20;

pb.Height = 72;

pb.Image = Properties.Resources.sash;

pb.Left = 175;

pb.Top = 173;

pb.BackgroundImageLayout = ImageLayout.Zoom;

pictureBox2.BackgroundImageLayout = ImageLayout.Zoom;

pb.BringToFront();

pb.Parent = pictureBox3;

pictureBox2.BackgroundImage = Properties.Resources.sash;

pb.BackColor = Color.Transparent;

this.pictureBox3.Controls.Add(pb);

sashCount++;

WindowCost();

pb.MouseClick += Pb\_Click;

pb.MouseDoubleClick += Pb\_doubleClick;

pb.MouseMove += Pb\_MouseMove;

pb.MouseDown += Pb\_MouseDown;

pb.MouseUp += Pb\_MouseUp;

picadd = true;

}

catch

{

MessageBox.Show("Вы не добавили предмет", "Ошибка");

}

}

protected override void OnPaint(PaintEventArgs e)

{

e.Graphics.DrawImage(Properties.Resources.sash, Point.Empty);

}

private void textBoxWidth\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if (!Char.IsDigit(number) && number != 8)

{

e.Handled = true;

}

}

private void textBoxHeight\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if (!Char.IsDigit(number) && number != 8)

{

e.Handled = true;

}

}

private void textBoxWidth\_TextChanged(object sender, EventArgs e)

{

Width = Convert.ToInt32(textBoxWidth.Text);

WindowCost();

}

private void textBoxHeight\_TextChanged(object sender, EventArgs e)

{

Height = Convert.ToInt32(textBoxHeight.Text);

WindowCost();

}

private void PaintHoriz(object sender, PaintEventArgs e)

{

e.Graphics.DrawImage(Properties.Resources.png\_line\_hor, new Point(175,173));

}

private void WindowProject\_Load(object sender, EventArgs e)

{

textBoxWidth.Text = "500";

textBoxHeight.Text = "500";

comboBox\_glazingtype.DataSource = glazingTypeTableAdapter1.GetData();

comboBox\_glazingtype.DisplayMember = "Title";

comboBox\_glazingtype.ValueMember = "IDGlazingType";

comboBox\_glazingtype.SelectedIndex = 0;

idglazingtype = (int)comboBox\_glazingtype.SelectedValue;

comboBox\_sill.DataSource = windowSillTableAdapter1.GetData();

comboBox\_sill.DisplayMember = "Title";

comboBox\_sill.ValueMember = "IDWindowSill";

comboBox\_sill.SelectedIndex = 0;

idsill = (int)comboBox\_sill.SelectedValue;

}

private void Pb\_Click(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Right)

{

PictureBox pb = sender as PictureBox;

pb.Focus();

}

if (e.Button == MouseButtons.Left)

{

PictureBox pb = sender as PictureBox;

pb.Focus();

}

}

private void Pb\_Horiz\_Click(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Right)

{

PictureBox pb = sender as PictureBox;

pb.Focus();

}

if (e.Button == MouseButtons.Left)

{

PictureBox pb = sender as PictureBox;

pb.Focus();

}

}

private void button\_save\_Click(object sender, EventArgs e)

{

if (pictureBox3.BackgroundImage != null || pictureBox3.Image != null)

{

SaveFileDialog pic = new SaveFileDialog();

pic.Title = "Сохранение картинки";

pic.OverwritePrompt = true;

pic.CheckPathExists = true;

pic.Filter = "Image Files(\*.BMP)|\*.BMP|Image Files(\*.JPG)|\*.JPG|Image Files(\*.PNG)|\*.PNG|All files (\*.\*)|\*.\*";

pic.ShowHelp = true;

if (pic.ShowDialog() == DialogResult.OK)

{

try

{

int width, height;

width = panel3.Width;

height = panel3.Height;

Bitmap save = new Bitmap(width, height);

panel3.DrawToBitmap(save, panel3.ClientRectangle);

save.Save(pic.FileName);

MessageBox.Show("Изображение сохранено", "Save");

}

catch

{

MessageBox.Show("Не удалось сохранить изшбражение","Ошибка",MessageBoxButtons.OK,MessageBoxIcon.Error);

}

}

}

}

private void comboBox\_glazingtype\_SelectionChangeCommitted(object sender, EventArgs e)

{

idglazingtype = (int)comboBox\_glazingtype.SelectedValue;

WindowCost();

}

private void comboBox\_sill\_SelectionChangeCommitted(object sender, EventArgs e)

{

idsill = (int)comboBox\_sill.SelectedValue;

WindowCost();

}

private void button\_addtoBase\_Click(object sender, EventArgs e)

{

int windowID;

byte[] photo = null;

idglazingtype = (int)comboBox\_glazingtype.SelectedValue;

idsill = (int)comboBox\_sill.SelectedValue;

if (customWindowsTableAdapter1.SelectLastID().ToString() == "")

{

windowID = 1;

}

else

{

windowID = (int)(customWindowsTableAdapter1.SelectLastID() + 1);

}

if (pictureBox3.Image != null || pictureBox3.BackgroundImage != null)

{

try

{

MemoryStream stream = new System.IO.MemoryStream();

int width, height;

width = panel3.Width;

height = panel3.Height;

Bitmap save = new Bitmap(width, height);

panel3.DrawToBitmap(save, panel3.ClientRectangle);

save.Save(stream, System.Drawing.Imaging.ImageFormat.Jpeg);

photo = stream.ToArray();

customWindowsTableAdapter1.Insert(windowID, idsill, idglazingtype, cost ,photo);

MessageBox.Show("Окно добавлено в базу");

}

catch

{

MessageBox.Show("Не удалось добавить в базу");

}

}

else

{

MessageBox.Show("Сначала спроектируйте окно");

return;

}

}

private void Pb\_Vertik\_Click(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Right)

{

PictureBox pb = sender as PictureBox;

pb.Focus();

}

if (e.Button == MouseButtons.Left)

{

PictureBox pb = sender as PictureBox;

pb.Focus();

}

}

private void Pb\_Mouse\_WheelHoriz(object sender, MouseEventArgs e)

{

PictureBox pb = sender as PictureBox;

if (e.Delta > 0)

pb.Width += 5;

else

pb.Width -= 5;

}

private void Pb\_Mouse\_WheelVertik(object sender, MouseEventArgs e)

{

PictureBox pb = sender as PictureBox;

if (e.Delta > 0)

pb.Height += 5;

else

pb.Height -= 5;

}

}

}

# Приложение Б. Результаты работы программы

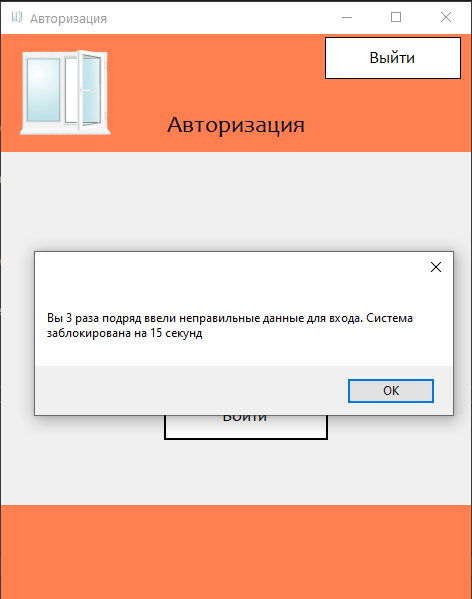


Рисунок Б.1 – Система заблокирована

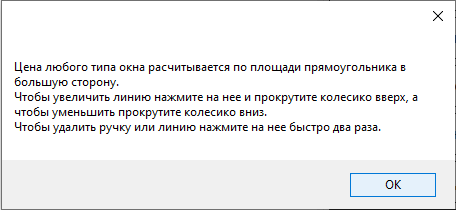


Рисунок Б.2 – Справка

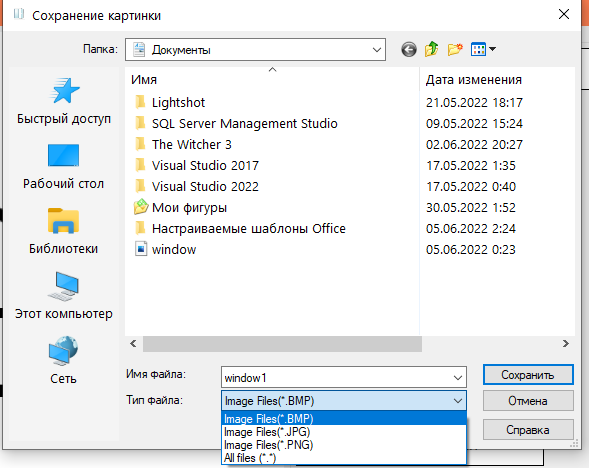


Рисунок Б.3 – Сохранение макета