**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П.О.СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

Специальность 1-40 05 01 «Информационные системы и технологии  
(по направлениям)»  
направление специальности  
1-40 05 01 12 «Информационные системы и технологии  
(в проектировании и производстве)»

Курсовая работа

по дисциплине: «Объектно-ориентированное программирование»

на тему: **«WPF система учета продаж мебели»**

Выполнил: студент гр. ИТП-21

Бурлаков А.О.

Принял: доцент

Курочка К.С.

Гомель 2024

**Оглавление**

[Введение 3](#_Toc161589659)

[1 Глава основные средства и технологии в разработке приложения 4](#_Toc161589660)

[1.1 Средства и технологии разработки 4](#_Toc161589661)

[1.2 Принципы SOLID 5](#_Toc161589662)

[1.3 Паттерны проетирования 7](#_Toc161589663)

[1.4 Техническое задание проекта 8](#_Toc161589664)

ВВЕДЕНИЕ

Повсеместное использование компьютерных технологий и техники в повседневной жизни людей позволяет автоматизировать любые производственные процессы в любой отрасли, что упрощает учет в организации и позволяет формировать отчеты за любой период и любой сложности за считанные секунды. Основная задача автоматизации бизнес-процессов – вывод качественных показателей процессов на принципиально новый уровень. Это достигается за счет того, что основным преимуществом автоматизированного труда над ручным является его надежность.

Современный рынок мебельной индустрии характеризуется высокой конкуренцией и необходимостью эффективного управления бизнесом. Учет продаж мебели играет важную роль в организации производства, оптимизации запасов, а также в принятии стратегических решений по развитию компании. Разработка приложения для учета продаж мебели на языке C# представляет собой актуальную задачу, которая позволит автоматизировать процессы управления и повысить эффективность работы предприятия.

Поэтому возникает необходимость в использовании программных продуктов, автоматизирующих основные бизнес-процессы предприятия. Но не всегда на рынке таковые можно найти, учитывая специфику каждой конкретной организации, поэтому приходится прибегать к разработке собственной информационной системы под нужды предприятия.

Практическая значимость исследования заключается в разработке информационной системы учета деятельности магазина по продаже мебели

Результатом исследования является разработанная информационная система учета деятельности магазина по продаже мебели, которая может применяться в любом розничном магазине с любым типом продукции.

1 ГЛАВА ОСНОВНЫЕ СРЕДСТВА И ТЕХНОЛОГИИ В РАЗРАБОТКЕ ПРИЛОЖЕНИЯ

1.1 Средства и технологии разработки

В данной главе будет рассмотрено обоснование выбора языка программирования C# для разработки приложения, а также рассмотрены средства и технологии, используемые в процессе разработки.

Выбор языка программирования C#:

C# является одним из наиболее распространенных языков программирования в мире, широко применяемым для создания различных типов приложений. Его интеграция с платформой .NET обеспечивает удобство разработки, высокую производительность и масштабируемость приложений.

Средства разработки:

Сохранение отчетов в XML-файлы с возможностью открытия в электронных таблицах: Для сохранения отчетов в формате XML можно использовать встроенные средства языка C#, такие как классы XmlWriter и XmlDocument. XML-формат обеспечивает удобное хранение структурированных данных, что позволяет сохранять информацию о продажах мебели в удобном и легко читаемом формате. Такие XML-файлы могут быть открыты и просмотрены в электронных таблицах, таких как Microsoft Excel, что обеспечивает удобство анализа данных.

СУБД PostgreSQL: PostgreSQL является мощной и распространенной объектно-реляционной системой управления базами данных (СУБД). Она обладает богатым набором функций, поддерживает множество типов данных и обеспечивает высокую производительность и надежность. Использование PostgreSQL позволяет эффективно хранить и управлять данными о продажах мебели, обеспечивая высокий уровень безопасности и отказоустойчивости.

Технология доступа ADO: ADO (ActiveX Data Objects) представляет собой набор компонентов и объектов, которые позволяют приложениям на платформе Windows взаимодействовать с базами данных. С помощью технологии ADO можно осуществлять подключение к базе данных, выполнение SQL-запросов, получение и обновление данных. Это обеспечивает удобство работы с данными и интеграцию с различными источниками данных, в том числе с СУБД PostgreSQL.

Доступ к данным при помощи LINQ: LINQ (Language-Integrated Query) представляет собой набор функциональных возможностей языка C#, позволяющих проводить запросы к коллекциям объектов и другим источникам данных непосредственно из кода программы. С помощью LINQ можно выполнять различные операции с данными, такие как фильтрация, сортировка, группировка и агрегация, что значительно упрощает доступ и манипулирование данными в приложении. LINQ также обеспечивает статическую типизацию и проверку на этапе компиляции, что повышает надежность и производительность приложения.

Таким образом, использование указанных технологий в процессе разработки приложения для учета продаж мебели на языке C# обеспечивает высокую производительность, надежность и удобство работы с данными, что в конечном итоге способствует повышению эффективности управления бизнесом.

Список литературных источников:

1. "C# 9.0 in a Nutshell" авторов Джозефа Албахари и Беннета Страттон
2. "Pro C# 9 with .NET 5" автора Эндрю Троелсена
3. "C# 8.0 and .NET Core 3.0 - Modern Cross-Platform Development" автора Марка Йоуссина
4. "C# 7.0 in a Nutshell" авторов Джозефа Албахари и Беннета Страттон
5. "C# 6.0 Cookbook" авторов Джейми Маннинга и Мариуса Захарие

1.2 Принципы SOLID

Принципы SOLID - это набор основных принципов объектно-ориентированного программирования, которые помогают создавать гибкие, расширяемые и поддерживаемые приложения. Давайте рассмотрим каждый из принципов подробнее и как их можно применить в разработке приложения для учета продаж мебели:

1. Принцип единственной ответственности (Single Responsibility Principle - SRP):

- Этот принцип гласит, что каждый класс должен иметь только одну причину для изменения. Он устанавливает, что класс должен быть ответственным только за одну часть функциональности.

- Применение в приложении: Классы в приложении для учета продаж мебели должны быть разделены на отдельные модули, каждый из которых отвечает только за определенную функциональность, например, управление данными о продажах, генерация отчетов, аутентификация пользователей и т.д.

1. Принцип открытости/закрытости (Open/Closed Principle - OCP):

- Согласно этому принципу, классы должны быть открыты для расширения, но закрыты для изменения. Это означает, что изменения в поведении класса должны происходить путем добавления нового кода, а не изменения существующего.

- Применение в приложении: Для добавления новых функций или изменения поведения приложения, предпочтительно создавать новые классы или модули, которые расширяют функциональность приложения, не затрагивая существующий код.

1. Принцип подстановки Барбары Лисков (Liskov Substitution Principle - LSP):

- Этот принцип утверждает, что объекты в программе должны быть заменяемыми на экземпляры их подтипов без изменения корректности выполнения программы.

- Применение в приложении: При разработке приложения следует учитывать, что все подклассы должны быть совместимы с базовым классом, и их поведение не должно нарушать ожидаемую функциональность базового класса.

1. Принцип разделения интерфейса (Interface Segregation Principle - ISP):

- Согласно этому принципу, клиенты не должны зависеть от методов, которые они не используют. Интерфейсы должны быть разделены на более мелкие, специфические для клиентов, чтобы минимизировать зависимости.

- Применение в приложении: Интерфейсы в приложении должны быть разделены на более мелкие, специализированные интерфейсы, чтобы клиенты могли использовать только те методы, которые им нужны, без необходимости реализации всех методов большого интерфейса.

1. Принцип инверсии зависимостей (Dependency Inversion Principle - DIP):

- Этот принцип гласит, что модули верхнего уровня не должны зависеть от модулей нижнего уровня. Оба должны зависеть от абстракций. Конкретные классы должны зависеть от абстракций, а не наоборот.

- Применение в приложении: Вместо прямой зависимости между классами, предпочтительно использовать абстракции и интерфейсы для уменьшения связанности компонентов и облегчения тестирования и поддержки кода.

- Применение принципов SOLID в разработке приложения для учета продаж мебели поможет создать более гибкий, расширяемый и легко поддерживаемый код, что повысит его качество и эффективность.

1.3 Паттерны проетирования

В приложении для учета продаж мебели можно применить различные паттерны проектирования для облегчения разработки, улучшения архитектуры и увеличения гибкости приложения. Рассмотрим некоторые из них:

1. Паттерн MVC (Model-View-Controller):

Модель (Model) представляет бизнес-логику и данные приложения. В данном случае это могут быть классы для работы с базой данных и обработки данных о продажах.

Представление (View) отвечает за отображение данных пользователю. Здесь могут быть реализованы формы для ввода и отображения информации о продажах мебели.

Контроллер (Controller) обрабатывает пользовательские запросы, вызывает соответствующие методы модели и обновляет представление. Это может быть слой приложения, который обрабатывает взаимодействие с пользователем и управляет бизнес-логикой.

1. Паттерн Repository:

Этот паттерн используется для инкапсуляции доступа к данным. В приложении для учета продаж мебели репозиторий может быть использован для абстрагирования работы с базой данных. Каждый тип данных (например, продажи, категории, производители) может иметь свой репозиторий, который обеспечивает доступ к данным этого типа.

1. Паттерн Фабрика (Factory):

Фабричный метод используется для создания объектов без явного указания их класса. В приложении для учета продаж мебели фабрика может быть использована для создания объектов различных типов мебели в зависимости от параметров, таких как категория или производитель.

1. Паттерн Стратегия (Strategy):

Стратегия позволяет выбирать алгоритм выполнения во время выполнения программы. В приложении для учета продаж мебели этот паттерн может быть использован для выбора различных стратегий обработки запросов или генерации отчетов в зависимости от требований пользователя или контекста выполнения.

1. Паттерн Наблюдатель (Observer):

Наблюдатель используется для реализации механизма подписки на изменения объекта и автоматического уведомления наблюдателей о таких изменениях. В приложении для учета продаж мебели этот паттерн может быть использован для реализации механизма уведомления об изменениях в базе данных или данных о продажах.

1. Паттерн Декоратор (Decorator):

Декоратор позволяет добавлять новую функциональность объектам динамически. В приложении для учета продаж мебели этот паттерн может быть использован для добавления дополнительной функциональности, например, для расширения возможностей работы с данными о продажах или для добавления дополнительной проверки данных перед их сохранением.

Применение указанных выше паттернов проектирования поможет создать более гибкую, расширяемую и поддерживаемую архитектуру приложения для учета продаж мебели. Каждый паттерн решает определенные проблемы проектирования и может быть использован в соответствии с требованиями и особенностями приложения.

1.4 Техническое задание проекта

Цель проекта: Разработать приложение для учета продаж мебели.

Исходные данные: для каждой единицы товара известны ее наименование, категория, розничная цена и производитель. Мебель может быть представлена как поштучно, так и в составе гарнитуров.

Функциональные требования:

1. Возможность добавления, удаления и редактирования информации о продажах мебели:

- Пользователь должен иметь возможность добавлять новые записи о продажах мебели, удалять существующие записи и редактировать информацию о продажах, включая наименование товара, категорию, цену и производителя.

- Для добавления новой записи пользователь должен заполнить обязательные поля формы, такие как наименование товара и его цена, а также может указать категорию и производителя.

- При редактировании существующей записи пользователь может изменять любые данные о продаже мебели.

- Для удаления записи пользователь должен подтвердить свое намерение удалить запись.

1. Ведение базы данных о проданных товарах:

- Приложение должно вести базу данных о проданных товарах, включая информацию о наименовании товара, его категории, цене и производителе.

- Для каждой продажи мебели должна быть создана соответствующая запись в базе данных с указанием всех сопутствующих данных.

- База данных должна быть структурирована и удобна для поиска и фильтрации данных по различным критериям.

1. Поддержка различных типов мебели:

- Приложение должно поддерживать различные типы мебели, включая как отдельные единицы товара, так и гарнитуры.

- Для каждого типа мебели должна быть предусмотрена соответствующая структура данных и интерфейс взаимодействия с пользователем.

1. Генерация отчетов о продажах за определенный период времени:

- Пользователь должен иметь возможность генерировать отчеты о продажах мебели за определенный период времени.

- Отчет должен содержать информацию о каждой продаже, включая наименование товара, категорию, цену и производителя, а также дату и время продажи.

- Отчет может быть сохранен в различных форматах, включая XML, CSV или PDF, с возможностью открытия в электронных таблицах для анализа и обработки данных.

1. Интерфейс пользователя с удобным и интуитивно понятным интерфейсом:

- Приложение должно иметь удобный и интуитивно понятный интерфейс, позволяющий пользователям легко и быстро освоить его функционал.

- Интерфейс должен быть доступен на русском и английском языках, с возможностью выбора пользователем предпочтительного языка.

- Пользовательский интерфейс должен быть адаптивным и поддерживать различные разрешения экранов, а также быть доступным для использования как на стационарных, так и на мобильных устройствах.

Нефункциональные требования:

1. Производительность:

- Приложение должно обеспечивать высокую производительность при работе с базой данных, обработке запросов и генерации отчетов.

- Время ответа на запросы пользователя должно быть минимальным, не превышающим 1-2 секунды.

- Приложение должно быть способно обрабатывать большие объемы данных без замедления производительности.

1. Надежность:

- Приложение должно быть стабильным и надежным, минимизируя возможность возникновения сбоев и ошибок.

- В случае возникновения ошибок приложение должно предоставлять информативные сообщения об ошибках и возможные способы их устранения.

- Все данные, вводимые пользователем, должны быть корректно валидированы, чтобы избежать ошибочного ввода и сохранить целостность данных.

1. Безопасность:

- Доступ к приложению и данным должен быть защищен от несанкционированного доступа.

- Пользовательская аутентификация должна быть реализована с использованием сильных паролей и методов шифрования.

- Приложение должно обеспечивать разграничение прав доступа пользователей в зависимости от их ролей и полномочий.

1. Масштабируемость:

- Приложение должно быть легко масштабируемым и способным обрабатывать рост числа пользователей и объема данных.

- В случае необходимости приложение должно поддерживать горизонтальное и вертикальное масштабирование без простоя в работе.

1. Удобство использования:

- Интерфейс пользователя должен быть интуитивно понятным и легко освоимым даже для пользователей без технического образования.

- Для удобства пользователей приложение должно предоставлять контекстную справку и подсказки о функциональности и использовании различных элементов интерфейса.

1. Локализация:

- Приложение должно поддерживать локализацию на различные языки, включая русский и английский.

­- Пользователи должны иметь возможность выбрать предпочтительный язык интерфейса приложения из доступных опций.

1. Отказоустойчивость:

- Приложение должно быть отказоустойчивым и способным восстанавливаться после сбоев и аварийных ситуаций.

- Резервное копирование данных и регулярное обслуживание базы данных должны быть организованы для предотвращения потери данных в случае сбоев.

1. Эффективное использование ресурсов:

- Приложение должно эффективно использовать ресурсы компьютера, включая процессор, память и сетевое соединение.

- Приложение не должно потреблять излишние ресурсы компьютера и должно обеспечивать оптимальную производительность при минимальном потреблении ресурсов.

Таким образом, выполнение функциональных требований позволит разработать полнофункциональное приложение для учета продаж мебели, обеспечивающее удобство использования, точность и надежность в учете данных о продажах.