**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**«ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО»**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по дисциплине «Объектно-ориентированное программирование»

на тему: *WPF система учета продаж мебели*

Исполнитель: студент гр. ИТП-21

Бурлаков А.О.

Руководитель: преподаватель

Курочка К.С.

Дата проверки: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата допуска к защите: ­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подписи членов комиссии

по защите курсовой работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2024

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc161652339)

[1 Основные средства и технологии в разработке приложения 4](#_Toc161652340)

[1.1 Язык программирования C# 4](#_Toc161652341)

[1.2 Средства и технологии разработки 7](#_Toc161652342)

[1.3 Принципы SOLID 9](#_Toc161652343)

[1.4 Паттерны проетирования 10](#_Toc161652344)

[1.5 Техническое задание проекта 11](#_Toc161652345)

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 ОСНОВНЫЕ СРЕДСТВА И ТЕХНОЛОГИИ В РАЗРАБОТКЕ ПРИЛОЖЕНИЯ

* 1. Язык программирования C#

*C#*, как язык программирования, занимает лидирующие позиции среди мощных, динамично развивающихся и востребованных технологий в сфере информационных технологий. Этот универсальный язык применяется для создания самых разнообразных приложений: от компактных десктопных программ до масштабных веб-порталов и сервисов, обслуживающих огромное количество пользователей ежедневно.

*C#* имеет синтаксис, схожий с Си, и наследован от *С++* и *Java*. Этот объектно-ориентированный язык программирования унаследовал многое от *Java* и *С++*, включая полиморфизм, наследование, перегрузку операторов и статическую типизацию. Объектно-ориентированный подход *C*# позволяет разрабатывать крупные, гибкие, масштабируемые и легко расширяемые приложения. Постоянное обновление *C*# привносит новые интересные функции с каждой новой версией, такие как лямбды, динамическое связывание, асинхронные методы и другие.

Программы на *C*# выполняются в .*NET*, виртуальной среде выполнения, которая вызывает общеязыковую среду выполнения (*CLR*) и набор классов библиотек .*NET*. *CLR* представляет собой реализацию общеязыковой инфраструктуры (*CLI*), международного стандарта, определенного *Microsoft*. *CLI* является основой для создания сред выполнения и разработки, где языки и библиотеки могут взаимодействовать между собой

На *C*# исходный код компилируется в промежуточный язык (*IL*), соответствующий *CLI*. Код *IL* и ресурсы, такие как изображения и текстовые данные, хранятся в сборке с расширением .*dll*. Сборка содержит метаданные о типах, версиях, языках и других параметрах для сборки.

Во время выполнения программы на *C*# сборка загружается в среду *CLR*, которая выполняет *JIT*-компиляцию из *IL* в машинный код. *CLR* также управляет автоматической сборкой мусора, обработкой исключений и управлением ресурсами. Код, выполняемый *CLR*, иногда называют «управляемым кодом», чтобы подчеркнуть различия с «неуправляемым кодом», который компилируется непосредственно в машинный код для конкретной платформы.

В начале знакомства с языком программирования C# отлично подойдет официальная документация на сайте *Microsoft*, структурированная по различным категориям. Здесь вы найдете разделы как "Начало работы", "Учебники", "Основные понятия" и многое другое. Эта документация представляет собой интерактивное пособие с примерами, которые читатель может выполнять параллельно с чтением. Каждая статья включает ссылки на дополнительные ресурсы, для более глубокого изучения темы. Также для удобства пользователей предлагается возможность скачать электронную версию документации. Для тех, кто заинтересован преимущественно в практических задачах, на сайте *Microsoft* есть бесплатные курсы по *C*#.

Полезным также является сайт *Metanit* по программированию. Здесь вы найдете огромное количество информации не только о *C*#, но и о других популярных языках программирования, а также о различных системах управления базами данных. Раздел, посвященный *C*#, содержит разнообразную информацию. Например, в разделе "Общая тематика" доступно полное руководство по *C*# 9 и .*NET* 5, различные шаблоны проектирования, сетевое программирование в *C*# и .*NET*, алгоритмы, структуры данных. По завершении курса просмотра материала предлагается пройти тесты для проверки усвоенных знаний. Разработка на *C*# охватывает разные области: веб-разработку, мобильные приложения, игры. Metanit также предоставляет информацию о базах данных.

Теперь давайте рассмотрим особенности языка *C*# и их применение в ходе выполнения курсовой работы. Паттерн в программировании представляет собой определенный метод построения кода для решения типовых проблем проектирования. Он предполагает наличие шаблонных проблем, для которых паттерны предлагают ряд принципов решения этих задач.

Разнообразие паттернов программирования предлагает различные решения задач и выполняет разнообразные функции. Анализ конкретных категорий паттернов базируется на назначении или поставленной цели паттерна.

Одной из ключевых категорий паттернов являются порождающие паттерны, которые управляют процессом создания классов и объектов. В данной категории выделяются следующие паттерны:

– Абстрактная фабрика (*Abstract Factory*): обеспечивает интерфейс для создания семейств взаимосвязанных объектов с определенными интерфейсами, без привязки к конкретным типам данных объектов;

– Строитель (*Builder*): инкапсулирует создание объекта, позволяя разделить его создание на различные этапы;

– Фабричный метод (*Factory Method*): предоставляет интерфейс для создания объектов класса, оставляя решение о создании конкретных объектов на подклассах;

– Прототип (*Prototype*): дает возможность создавать объекты на основе ранее созданных прототипов, предлагая метод клонирования объектов;

– Одиночка (*Singleton*): гарантирует, что для определенного класса будет создан только один объект, а также предоставляет к нему доступ.

Из вышеописанных паттернов, наиболее широко используемым и популярным является паттерн «Одиночка». Однако помимо порождающих паттернов существуют структурные и поведенческие паттерны, которые также играют важную роль в разработке программного обеспечения.

Акцент на принципе *KISS* (*Keep It Simple, Stupid*) при написании кода следует трактовать как сохранение простоты и понятности программного кода. Соблюдение этого принципа позволяет делать код ясным и легким для понимания.

Важнейший атрибут объектно-ориентированного программирования – это наследование. Этот механизм позволяет создавать дочерние классы, которые наследуют, расширяют или изменяют возможности родительского класса. Базовый класс, чьи члены наследуются, называется родительским, а класс, который наследует члены базового класса, называется производным. Наследование часто применяется в проектах, упрощая структуру кода и избавляя от повторения одних и тех же полей и методов в каждом классе. Например, класс Product может служить основой для производных классов Chair, Table, Cupboard и других.

Также *C#* предоставляет ссылочные типы, которые могут иметь значение *NULL*, а также ссылочные типы, не допускающие значение *NULL*, утверждая определенные свойства переменных ссылочного типа. В данном проекте использование ссылочных типов, позволяющих значение *NULL*, определяет принадлежность продукта к группе. Например, определение типа гарнитуры, к которой принадлежит товар, или возврат значения *NULL* при отсутствии принадлежности.

Обработка исключений в *C#* помогает управлять непредвиденными ситуациями, которые возникают во время выполнения программы. Для этой цели используются ключевые слова *try*, *catch* и *finally*, которые позволяют обрабатывать действия, которые могут завершиться неудачно. Например, при ошибке ввода данных, проблемах с доступом к базе данных или некорректном создании объекта вызывается исключение, указывающее на возникшую проблему.

*LINQ (Language Integrated Query)* представляет собой набор технологий, интегрирующих запросы прямо в код на *C#.* Технологии *LINQ* упрощают создание запросов данных, предоставляя удобные выражения, совместимые с классами, методами и событиями. В приложении *LINQ* используется для обработки данных, обеспечивая удобный синтаксис для фильтрации, сортировки и других манипуляций с элементами данных.

*Microsoft* разработал язык запросов *LINQ*, предоставляющий удобный синтаксис для запросов данных. Этот язык позволяет легко фильтровать, сортировать и преобразовывать данные, делая всю операцию более элегантной и понятной. В использованном приложении *LINQ* активно применяется для работы с данными.

1.2 Средства и технологии разработки

XML, или расширяемый язык разметки, представляет собой спецификацию, описывающую XML-документы и частично описывающую поведение XML-процессоров. Этот язык разрабатывался для простого форматирования документов, удобного как для человека, так и для программ. Его отличительной чертой является расширяемость, позволяющая разработчику создавать разметку по своим потребностям без жесткой фиксации формата.

С точки зрения физической структуры, документ состоит из сущностей, каждая из которых может ссылаться на другую. В документе существует только один корневой элемент — документная сущность, в то время как его содержание представлено символами.

С логической точки зрения, документ содержит комментарии, объявления, элементы, ссылки на сущности и инструкции обработки. Эти компоненты структурируются разметкой, определяя структуру документа.

Элементы документа должны быть правильно вложены друг в друга. Любой элемент, начинающийся внутри другого, должен завершаться там же. Символьные данные могут содержаться как непосредственно в элементах, так и в специальных секциях "CDATA". Для связывания текста с логическими единицами используются атрибуты, представляющие пары имя-значение.

Имена в XML должны начинаться с буквы, подчеркивания или двоеточия, содержать только допустимые символы, образующие слова Unicode, арабские цифры и некоторые символы. Имена не могут начинаться с "xml" (любым регистром), зарезервированным для W3C. За пределами ASCII можно использовать родной язык для именования.

**Преимущества использования XML**

1) Расширяемость: XML позволяет разработчику создавать собственную разметку в соответствии с требованиями конкретной области.

2)Удобство для человека и программ: Благодаря простому формальному синтаксису, XML удобен как для человека, так и для программного обеспечения.

3)Базирование на Юникод: Использование кодировок Юникод обеспечивает представление содержимого документов, что способствует широкому использованию XML в различных приложениях.

4)Обработка комментариев и инструкций: XML позволяет включать комментарии и инструкции в документы для улучшения их понимания и обработки.

5)Пролог и корневой элемент: Документ начинается с пролога, содержащего объявления и инструкции, а обязательным элементом является корневой элемент, определяющий основную структуру документа.

6)Символьные данные и CDATA: XML обеспечивает способы включения символьных данных в документ, как прямо в элементы, так и с помощью расширенных CDATA-секций.

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.

1.3 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.

1.4 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.

1.5 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.