КОЛЛЕДЖ ПРЕЗИДЕНТСКОЙ АКАДЕМИИ

КУРСОВОЙ ПРОЕКТ

Тема: Разработка программного модуля "Система учета и анализа данных о продаже игр и развлечений".

Специальность 09.02.07 Информационные системы и программирование

Выполнила студентка группы 34ИС-21 _____ **В.В. Абрамцова**

Руководитель _____ **В.Ю. Назаров**

Москва 2023

УТВЕРЖДАЮ

Зам. директора КМПО

_____С.Ф. Гасанов

«_____» _____ 2023 г.

ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ

по дисциплине: МДК.01.01 Разработка программных модулей

Специальность 09.02.07 Информационные системы и программирование

Студентка группы 34ИС-21 В.В. Абрамцова

ТЕМА: Разработка программного модуля "Система учета и анализа данных о продаже игр и развлечений".

Дата выдачи задания «_____» _____ 2023 г.

Срок сдачи работы «_____» _____ 2023 г.

Москва 2023

Федеральное государственное бюджетное образовательное учреждение высшего образования
«РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА И ГОСУДАРСТВЕННОЙ СЛУЖБЫ ПРИ ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»
КОЛЛЕДЖ МНОГОУРОВНЕВОГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

Задание

на курсовой проект

Дисциплина: МДК.01.01 Разработка программных модулей

Тема: Разработка программного модуля "Система учета и анализа данных о продаже игр и развлечений"

Специальность: 09.02.07 Информационные системы и программирование

Группа: 34ИС-21

ФИО студента Абрамцова В.В.

ФИО руководителя Назаров В.Ю.

1. Проанализировать предметную область
2. Проанализировать готовые решения
3. Подготовить техническое задание
4. Подготовить план тестирования
5. Обосновать выбор инструментов и средств разработки
6. Описать реализацию технического задания
7. Выполнить тестирование

Задание выдано «_____» _____2023 г.

Срок выполнения«_____» _____2023 г.

Сроки защиты _____

Преподаватель: _____

Задание получил: _____

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире цифровых технологий, где индустрия развлечений и игр находится в постоянном состоянии динамичного развития, эффективное управление данными становится фундаментальным аспектом успешной деятельности компаний. Сфера развлечений, насыщенная новыми продуктами и разнообразными технологиями, требует инновационных решений в управлении информацией. Именно в этом контексте была выбрана тема курсового проекта - "Разработка программного модуля 'Система учета и анализа данных о продаже игр и развлечений'", с целью внести вклад в развитие современных инструментов управления данными в индустрии развлечений.

Актуальность курсовой работы объясняется бурным развитием индустрии развлечений, где конкуренция на рынке становится все более острой, неотъемлемым становится вопрос эффективного управления данными. Каждый момент важен, и оперативное принятие решений, основанное на анализе фактических данных, становится критически важным фактором для успешного выстраивания бизнес-процессов. Разработка программного модуля, который специализируется на учете и анализе данных о продажах в индустрии развлечений, призвана предоставить компаниям необходимый инструментарий для эффективного управления информацией и принятия обоснованных стратегических решений.

Цель данного курсового проекта – создание программного модуля "Система учета и анализа данных о продаже игр и развлечений". Этот модуль предназначен для того, чтобы компании в области развлечений могли оперативно реагировать на изменения в рыночной среде, управлять ассортиментом продуктов, а также получать аналитическую базу для выработки стратегий развития.

Задачи включают в себя не только разработку структуры базы данных, учитывающей особенности бизнеса в данной сфере, но и реализацию функционала для надежного хранения информации о клиентах, продукции и продажах. Кроме того, планируется создать инструменты для анализа данных и

формирования отчетов, чтобы обеспечить компаниям возможность оперативного и обоснованного принятия решений.

Практическая значимость заключается в том, что курсовой проект представляет собой не только теоретическую разработку, но и важный инструмент для компаний, оперирующих в сфере развлечений и игр. Созданный программный модуль может стать основой для создания собственных информационных систем, адаптированных под уникальные требования учета и анализа данных в данной отрасли. Это предоставляет возможность компаниям более гибко реагировать на изменения рынка, оперативно корректировать стратегии и принимать обоснованные решения на основе фактических данных, что в конечном итоге может определить успех в динамичной индустрии развлечений.

1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Информационное обеспечение задачи

Для успешной реализации поставленной задачи, необходимо обеспечить полноценную поддержку информационных потребностей системы учета и анализа данных о продаже игр и развлечений. Рассмотрим ключевые аспекты информационного обеспечения:

Анализ потребностей системы:

Проведение тщательного анализа текущих потребностей в управлении данными о продажах игр и развлечений.

Определение требований к функционалу, включая возможности анализа продаж, управления инвентарем, отслеживания тенденций и прогнозирования спроса.

Проектирование базы данных:

Разработка оптимальной структуры базы данных, учитывающей особенности отрасли развлечений.

Обеспечение эффективного хранения и быстрого доступа к данным о продажах, клиентах и игровом контенте.

Разработка пользовательского интерфейса:

Создание интуитивного и удобного интерфейса для взаимодействия сотрудников и аналитиков с системой.

Интеграция инструментов анализа данных для обеспечения эффективного мониторинга и принятия стратегических решений.

Роли пользователей и функциональности:

Определение ролей пользователей, таких как аналитики, менеджеры продаж, администраторы системы.

Выделение функциональностей, включая управление продуктами, мониторинг продаж, анализ данных по клиентам и рынку.

Тестирование и корректировка:

Проведение систематического тестирования на всех этапах разработки для

выявления и устранения возможных ошибок.

Коррекция функциональностей и улучшение производительности системы на основе результатов тестирования.

1.2 Обзор и анализ существующих программных решений

В данном разделе представлен подробный обзор и анализ существующих программных решений в сфере учета и анализа данных о продажах игр и развлечений. Цель данного этапа – провести всестороннее исследование рынка программных продуктов, выявить их особенности, преимущества и недостатки. Это позволит принять обоснованное решение относительно выбора оптимального программного решения, удовлетворяющего специфичным потребностям моей системы.

Далее представлен обзор уже существующих программных решений:

1. GameSalesAnalyzer Pro:

Описание: GameSalesAnalyzer Pro является интегрированным решением, специально разработанным для анализа данных о продажах игр. Он предоставляет широкий спектр инструментов для мониторинга продаж, прогнозирования трендов и определения эффективности маркетинговых стратегий.

Преимущества: гибкая система настройки отчетов и аналитических дашбордов; интеграция с ведущими игровыми платформами и онлайн-магазинами; мощные алгоритмы анализа данных для выявления скрытых паттернов.

Недостатки: относительная высокая стоимость лицензии, что может быть значимым фактором для небольших предприятий; отсутствие полной совместимости с некоторыми устаревшими операционными системами, что требует дополнительных обновлений.

2. EntertainmentSalesTracker:

Описание: EntertainmentSalesTracker предоставляет универсальное решение для учета продаж различного развлекательного контента, включая игры,

видео и музыку. Система обеспечивает полный жизненный цикл продукта, начиная от поступления на склад до анализа итоговых продаж.

Преимущества: интегрированная система управления инвентарем и поставками; многомерный анализ данных для лучшего понимания предпочтений потребителей; многопользовательские отчеты и дашборды для оперативного мониторинга.

Недостатки: неограниченные возможности масштабирования для крупных компаний с высоким объемом данных; некоторые пользователи отмечают сложность в настройке пользовательского интерфейса и отчетов без специальной поддержки.

3. SalesInsights for Gaming:

Описание: SalesInsights for Gaming предоставляет мощные инструменты анализа данных о продажах игр с акцентом на стратегическом принятии решений. Это решение поддерживает высокоуровневый анализ производительности продуктов и эффективности маркетинговых кампаний.

Преимущества: встроенные алгоритмы машинного обучения для прогнозирования продаж; интерактивные отчеты с возможностью детального бурного анализа; интеграция с облачными сервисами для максимальной гибкости.

Недостатки: отсутствие встроенных средств для работы с некоторыми специфическими форматами данных, что требует дополнительной обработки перед загрузкой; в некоторых случаях, отмечается более высокий порог вхождения для новых пользователей из-за сложности использования продвинутых функций.

2. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ

2.1 Разработка требований к проекту

Цель:

Целью данного курсового проекта является создание программного модуля для учета и анализа данных о продаже игр и развлечений «GameStatistics», который служит для эффективного управления информацией о продажах в сфере развлекательных продуктов. «GameStatistics» разрабатывается с учетом потребностей компаний, специализирующихся на продаже игр и развлечений, и призван предоставить интегрированный и мощный инструмент для комплексного учета и анализа данных.

Область применения:

Охвачен широкий спектр деятельности компаний, занимающихся продажей игр и развлекательных продуктов. Ниже представлены основные области, в которых система может быть эффективно применена:

1. Розничная торговля:

Учет продаж: «GameStatistics» обеспечивает точный учет продаж различных игр и развлекательных продуктов в розничных точках продаж.

Управление ассортиментом: Менеджеры по продажам могут эффективно управлять ассортиментом, отслеживать актуальность товаров и анализировать их популярность.

2. Аналитика и стратегическое планирование:

Формирование отчетов: Система предоставляет инструменты для создания разнообразных отчетов, что обеспечивает аналитикам информацию для стратегического планирования.

3. Управление складом:

Синхронизация с управлением складом: Интеграция с системами управления складом для обеспечения своевременного учета и контроля за наличием продукции.

Требования к функциональности

Основные функции:

1. Добавление/удаление товаров

Администратор или менеджер должен иметь возможность добавить или удалить любую запись, хранящуюся в базе данных. Добавление затрагивает следующие поля:

– Наименование товара;
– Тип;
– Платформа;
– Цена;
– Описание;

2. Добавление/удаление клиентов

Администратор или менеджер должен иметь возможность добавить или удалить любую запись, хранящуюся в базе данных. Добавление затрагивает следующие поля:

– Имя;
– Фамилия;
– Почта;
– Номер телефона;

3. Оформление продажи

Менеджер должен иметь возможность оформить продажу, выбрав клиента, товар и дату покупки.

4. Поиск клиентов и товаров

Администратор или менеджер должен иметь возможность найти клиента или товар в базе данных по характеристикам.

5. Написание рекомендаций

Аналитик должен иметь возможность писать рекомендации, опираясь на диаграмму продаж и прошлые рекомендации.

6. Просмотр диаграммы продаж

Аналитик должен иметь возможность просмотра продаж для эффективного

визуального анализа данных и последующего написания рекомендаций.

1. Интерфейс

Графический интерфейс: программа должна иметь интуитивно понятный интерфейс, обеспечивающим удобство использования для различных пользователей. Навигационные элементы должны быть ясными и легко доступными.

Возможность быстро переходить между различными разделами для удобства работы. Важно использование понятных иконок и символов, чтобы упростить восприятие информации, однозначное отображение ключевых метрик и данных на главном экране для быстрого обзора.

2. Требования к надежности

Резервное копирование данных: программа должна обеспечивать надежное и эффективное резервное копирование данных, также значимо автоматическое резервное копирование, с учетом важности сохранения информации о продажах.

3. Защита данных

Строгие механизмы аутентификации: безопасность гарантируется механизмом аутентификации и авторизации пользователей.

4. Требования к производительности

Отклик интерфейса: программа должна быть обеспечена плавной и быстрой навигацией между различными разделами программы, уменьшая задержки и обеспечивая комфортное взаимодействие. Интерфейс должен реагировать мгновенно на ввод пользователя, обеспечивая быстрое и плавное взаимодействие при вводе данных или выполнении операций.

2.2 Построение диаграммы использования

Деятели:

1. Администратор: пользователь, ответственный за внесение информации о товарах и клиентах.

2. Аналитик: пользователь, ответственный за анализ продаж и написание рекомендаций.

3. Менеджер: пользователь, ответственный за внесение продаж и обновление данных о товарах и клиентах если в этом есть необходимость.

4. Приложение: программный модуль системы учета и анализа данных о продаже игр и развлечений.

Прецеденты для администратора:

1. Вход в систему:

– АДМИНИСТРАТОР вводит свои учетные данные (логин и пароль) для входа в систему.

– ПРИЛОЖЕНИЕ проводить проверку на корректность введенных данных и предоставляет доступ к системе.

2. Добавление товара:

– АДМИНИСТРАТОР заполняет информацию о товаре, вводя следующую информацию: тип, название, платформа, цена, описание.

– ПРИЛОЖЕНИЕ сохраняет данные о товаре в базу данных и выводит сообщение о успешном добавлении товара.

3. Добавление клиента:

– АДМИНИСТРАТОР заполняет информацию о клиенте, вводя следующую информацию: имя, фамилия, почта, номер телефона.

– ПРИЛОЖЕНИЕ сохраняет данные о клиенте в базу данных и выводит сообщение о успешном добавлении клиента.

Прецеденты для аналитика:

1. Вход в систему:

– АНАЛИТИК вводит свои учетные данные (логин и пароль) для входа в систему.

– ПРИЛОЖЕНИЕ проводит проверку на корректность введенных данных и предоставляет доступ к системе.

2. Просмотр анализа продаж:

– АНАЛИТИК выбирает характер сортировки данных.

– ПРИЛОЖЕНИЕ выводит столбчатую диаграмму с суммой продаж по

выбранной категории.

3. Написание рекомендаций:

– АНАЛИТИК пишет рекомендацию в специально выделенное для этого поле, далее сохраняет в базу данных по нажатию кнопки «Сохранить». При надобности сохраняет рекомендацию в документ с расширением .docx по нажатию кнопки «Сохранить в Word».

– По нажатию кнопки «Сохранить» ПРИЛОЖЕНИЕ сохраняет рекомендацию, дату и время ее написания в базу данных и выводит сообщение о успешном добавлении рекомендации. По нажатию кнопки «Сохранить в Word» ПРИЛОЖЕНИЕ сохраняет рекомендацию в файле с расширением .docx с датой ее написания в названии файла и выводит сообщение о успешном сохранении файла и его названием.

4. Просмотр прошлых рекомендаций:

– АНАЛИТИК выбирает рекомендацию из таблицы с рекомендациями. При надобности сохраняет рекомендацию в документ с расширением .docx по нажатию кнопки «Сохранить в Word».

– ПРИЛОЖЕНИЕ выводит выбранную рекомендацию в специально выведенное для этого поле. По нажатию кнопки «Сохранить в Word» ПРИЛОЖЕНИЕ сохраняет рекомендацию в файле с расширением .docx с датой ее написания в названии файла и выводит сообщение о успешном сохранении файла и его названием.

Прецеденты для менеджера:

1. Вход в систему:

– МЕНЕДЖЕР вводит свои учетные данные (логин и пароль) для входа в систему.

– ПРИЛОЖЕНИЕ проводит проверку на корректность введенных данных и предоставляет доступ к системе.

2. Поиск товара и просмотр данных о товаре:

– МЕНЕДЖЕР выбирает категорию (ID, название, цена, описание, тип,

платформа) поиска данных и вводит параметры поиска.

– ПРОГРАММА осуществляет поиск в базе данных по введенному параметру в выбранной категории и отображает все подходящие товары в таблице, включая следующую информацию: ID, название, цена, описание, тип, платформа.

3. Поиск клиента и просмотр информации о клиенте:

– МЕНЕДЖЕР выбирает категорию (ID, имя, фамилия, почта, номер телефона) поиска данных и вводит параметры поиска.

– ПРОГРАММА осуществляет поиск в базе данных по введенному параметру в выбранной категории и отображает всех подходящих клиентов в таблице, включая следующую информацию: ID, имя, фамилия, почта, номер телефона.

4. Удаление товара:

– МЕНЕДЖЕР выбирает нужный товар в таблице и нажимает на кнопку «Удалить».

– ПРОГРАММА удаляет товар из базы данных и выводит сообщение о успешном удалении.

5. Удаление клиента:

– МЕНЕДЖЕР выбирает нужного клиента в таблице и нажимает на кнопку «Удалить».

– ПРОГРАММА удаляет клиента из базы данных и выводит сообщение о успешном удалении.

6. Оформление продажи:

– МЕНЕДЖЕР выбирает клиента и товар из выпадающих списков, вводит дату продажи и нажимает кнопку «Оформить».

– ПРОГРАММА обрабатывает введенные данные, заносит продажу в базу данных и выводит сообщение о успешном добавлении продажи.

Отношения между прецедентами и деятелями:

Ассоциация между деятелем «Администратор» и прецедентами «Вход в

систему», «Добавление товара», «Добавление клиента». (Рисунок 1)

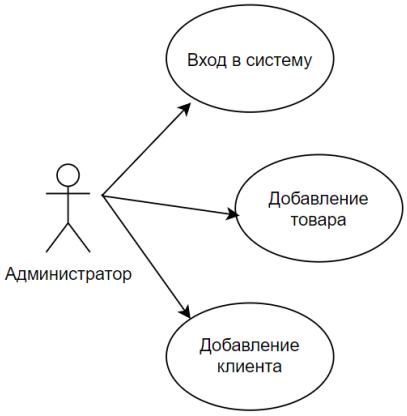

Рисунок 1 – Отношение ассоциаций между деятелями и прецедентами.

Ассоциация между деятелем «Аналитик» и прецедентами «Вход в систему», «Просмотр анализа продаж», «Написание рекомендаций». (Рисунок 2)

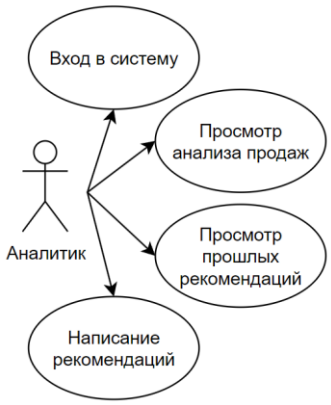

Рисунок 2 – Отношение ассоциаций между деятелями и прецедентами.

Ассоциация между деятелем «Менеджер» и прецедентами «Вход в систему», «Поиск товара и просмотр данных о товаре», «Поиск клиента и просмотр данных о клиенте», «Удаление товара», «Удаление клиента», «Оформление продаж». (Рисунок 3)

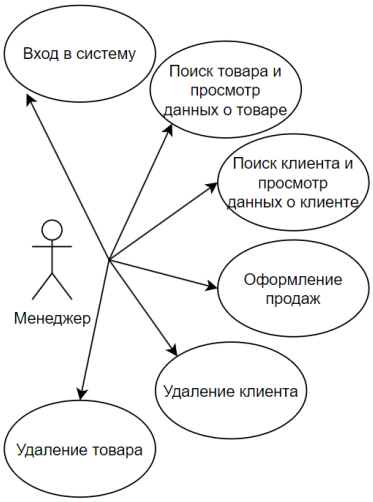

Рисунок 3 – Отношение ассоциаций между деятелями и прецедентами.

Включение между прецедентами «Управление данными», «Добавить клиента», «Добавить товар», «Удалить товар», «Удалить клиента», «Оформление продажи», «Поиск товара», «Поиск клиента». (Рисунок 4)

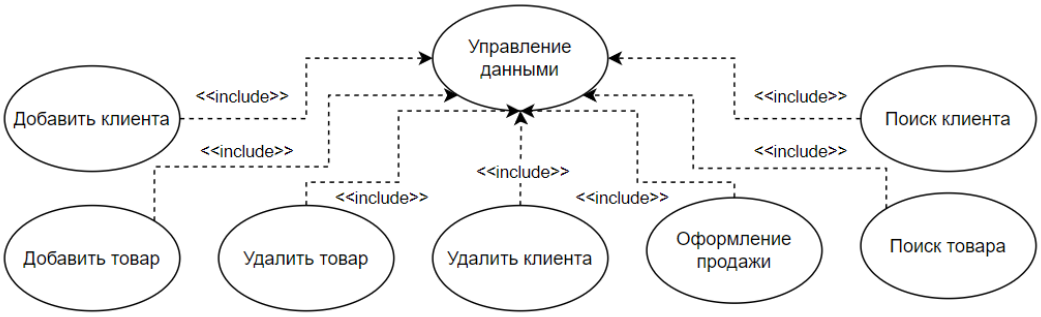

Рисунок 4 – Отношение включения между прецедентами.

Включение между прецедентами «Анализ продаж», «Просмотр анализа продаж», «Просмотр прошлых рекомендаций», «Написание рекомендаций». (Рисунок 5)

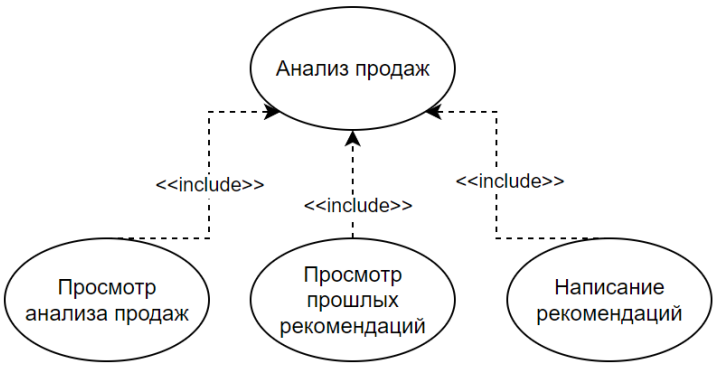

Рисунок 5 – Отношение включения между прецедентами.

Расширение между прецедентами «Добавить клиента», «Добавить товар»,

«Удалить товар», «Удалить клиента», «Оформить продажу», «Сохранить рекомендацию», «Сохранить рекомендацию в Word» и прецедентом "Подтверждение действия", который включает в себя отправку подтверждения пользователю. (Рисунок 6)

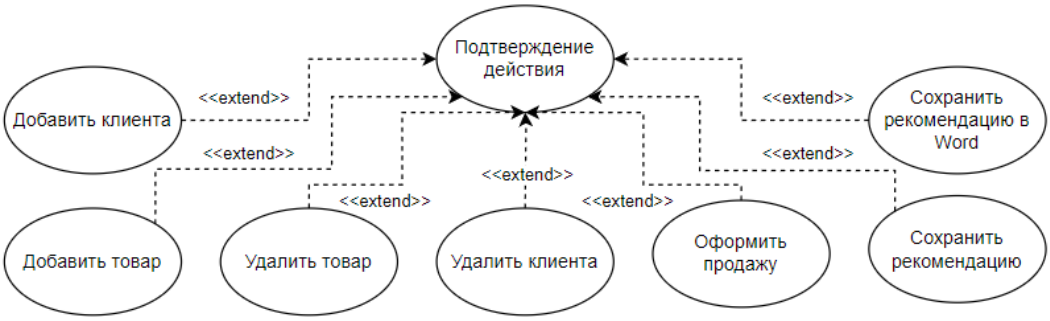

Рисунок 6 – Отношение расширения между прецедентами.

Обобщение между прецедентами «Управление GameStatistics», «Управление базой данных», «Управление рекомендациями», «Управление анализом продаж». (Рисунок 7)

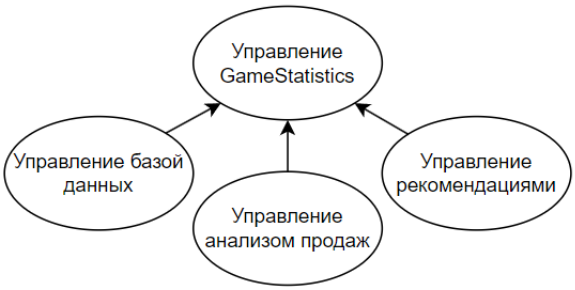

Рисунок 7 – Обобщение между прецедентами

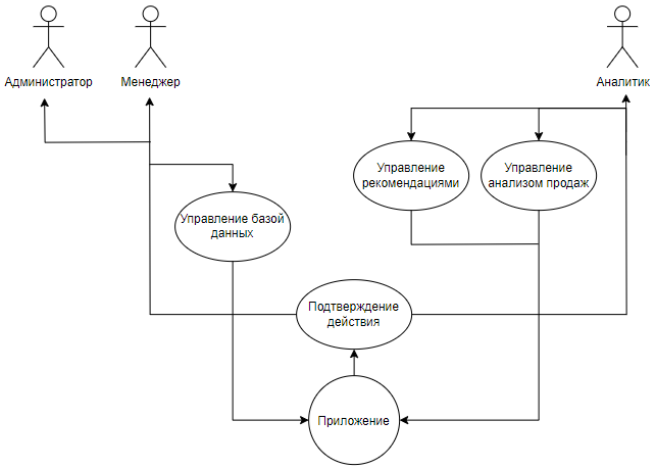

Рисунок 8 – Основная диаграмма вариантов использования.

2.3 Разработка сценария проекта

Разработка сценария проекта подразумевает создание последовательности действий, которые выполняются пользователями для достижения определенных целей в системе.

Администратор:

1. Вход в приложение:

– Пользователь открывает приложение.

– Вводит логин и пароль.

– Нажимает на кнопку «Войти».

2. Добавление товара:

– Пользователь выбирает пункт «Добавить товар» из выпадающего списка возможных действий.

– Заполняет характеристики товара (название, цена, описание, тип, платформа).

– Нажимает на кнопку «Добавить».

3. Добавление клиента:

– Пользователь выбирает пункт «Добавить клиента» из выпадающего списка возможных действий.

– Заполняет характеристики клиента (имя, фамилия, почта, номер телефона).

– Нажимает на кнопку «Добавить».

Аналитик:

1. Вход в приложение:

– Пользователь открывает приложение.

– Вводит логин и пароль.

– Нажимает на кнопку «Войти».

2. Просмотр анализа продаж:

– Пользователь нажимает на кнопку «Анализ продаж» на панели навигации.

– Пользователь выбирает характер сортировки данных.

3. Просмотр прошлых рекомендаций:

– Пользователь нажимает на кнопку «Рекомендации» на панели навигации.

– Просматривает список рекомендаций в таблице.

– Выбирает рекомендацию для ее отображения в специальном окне.

– При надобности сохраняет рекомендацию в файле с расширением .docx по нажатию кнопки «Сохранить в Word».

4. Написание рекомендаций:

– Пользователь нажимает на кнопку «Рекомендации» на панели навигации.

– Вводит рекомендации в специальное поле.

– Сохраняет рекомендацию в базе данных по нажатию кнопки «Сохранить».

– При надобности сохраняет рекомендацию в файле с расширением .docx по нажатию кнопки «Сохранить в Word».

Менеджер:

1. Вход в приложение:

– Пользователь открывает приложение.

– Вводит логин и пароль.

– Нажимает на кнопку «Войти».

2. Удаление товара:

– Пользователь выбирает пункт «Поиск/Удаление товара» из выпадающего списка возможных действий.

– Выбирает характеристику поиска товара (ID, название, цена, описание, тип, платформа).

– Вводит параметры поиска и выбирает нужный товар.

– Нажимает на кнопку «Удалить».

3. Удаление клиента:

– Пользователь выбирает пункт «Поиск/Удаление клиента» из выпадающего списка возможных действий.

– Выбирает характеристику поиска клиента (ID, имя, фамилия, почта, номер телефона).

– Вводит параметры поиска и выбирает нужного клиента.

– Нажимает на кнопку «Удалить».

4. Оформление продажи:

– Пользователь выбирает пункт «Оформить продажу» из выпадающего списка возможных действий.

– Выбирает клиента из выпадающего списка.

– Выбирает товар из выпадающего списка.

– Вводит дату оформления продажи.

– Нажимает на кнопку «Оформить».

На рисунке 9 приведен сценарий использования программы пользователем.

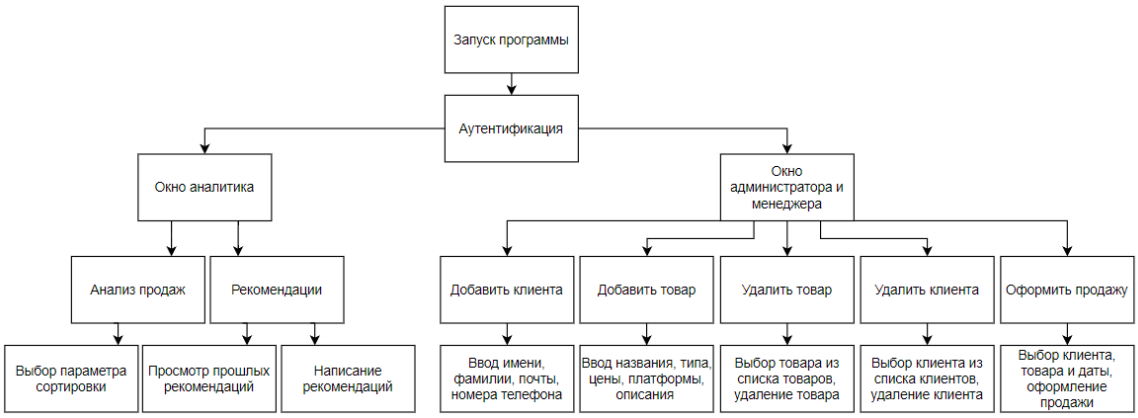

Рисунок 9 – Сценарий использования программы пользователем.

2.4 Построение диаграммы классов

1. Класс LoginPage (Приложение 1):

Назначение: управление процессом аутентификации для пользователей.

Описание: этот класс включает в себя функционал для проверки введенных учетных данных для последующего открытия основного окна приложения при успешном входе.

2. Класс ManagerPage1 (Приложение 2):

Назначение: управление интерфейсом страницы в приложении WPF для обеспечения процесса добавления/удаления клиентов, товаров и управления продажами.

Описание: класс ManagerPage1 представляет собой логику взаимодействия с пользовательским интерфейсом страницы в приложении. Он обеспечивает управление процессом добавления/удаления клиентов и товаров в базу данных, а также предоставляет функциональность для поиска клиентов и управления продажами.

3. Класс ManagerSearch (Приложение 3):

Назначение: управление поиском и отображением информации о продуктах в приложении WPF.

Описание: управляет страницей поиска и отображения информации о продуктах.

4. Класс ProductViewModel (Приложение 4):

Назначение: модель данных для удобного представления и отображения информации о продукте в пользовательском интерфейсе приложения WPF.

Описание: класс служит для инкапсуляции свойств, представляющих атрибуты продукта. Он используется для удобного и эффективного отображения данных в пользовательском интерфейсе без необходимости напрямую взаимодействовать с объектами сущности базы данных.

5. Класс ManagerSearchClnt (Приложение 5):

Назначение: управление поиском, отображением и удалением информации о клиентах в приложении WPF.

Описание: управляет страницей поиска и отображения информации о продуктах.

6. Класс AddSale (Приложение 6):

Назначение: управление добавлением информации о продажах в приложении WPF.

Описание: отвечает за отображение и обработку страницы добавления

продаж в приложении.

7. Класс AnalystPage1 (Приложение 7):

Назначение: управление навигацией на странице аналитика в приложении WPF.

Описание: представляет страницу аналитики в приложении.

8. Класс Recommendation (Приложение 8):

Назначение: управление рекомендациями аналитика в приложении WPF.

Описание: отвечает за отображение и сохранение рекомендаций аналитика.

9. Класс SalesAnalys (Приложение 9):

Назначение: проведение анализа продаж в приложении WPF.

Описание: отвечает за отображение и анализ данных по продажам.

10. Класс DB (Приложение 10):

Назначение: Класс DB служит для предоставления доступа к объекту контекста базы данных GameStatisticsEntities3 в приложении.

Описание: класс обеспечивает единственный экземпляр контекста базы данных для доступа к данным в приложении.

Диаграмма классов приведена в рисунке 10.

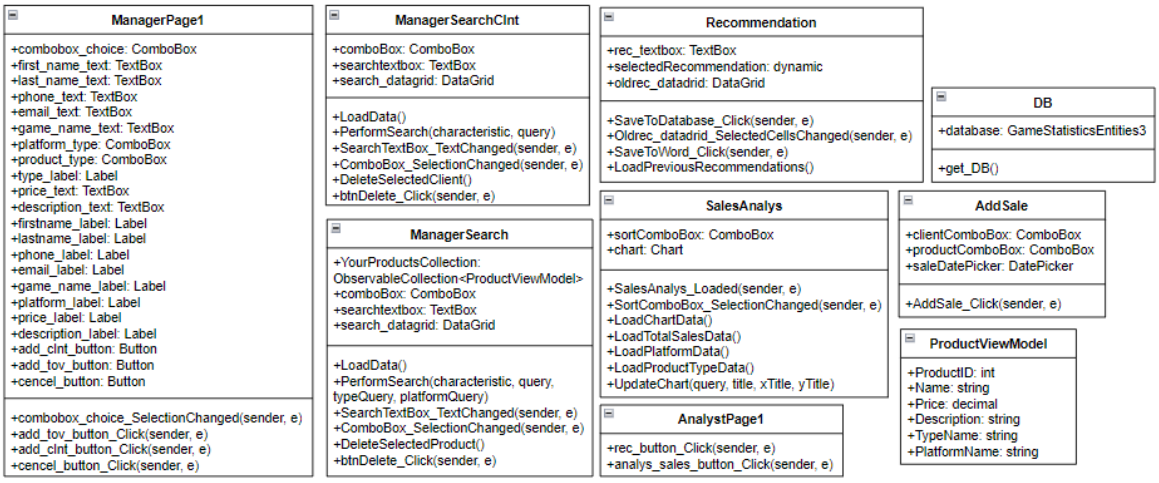

Рисунок 10 – Диаграмма классов.

2.5 Проектирование базы данных

Проектирование базы данных для данного приложения включает в себя определение таблиц и связей между ними. Ниже описано проектирование базы данных:

1. Таблица «Products»:

Структура данной сущности представлена на таблице 1.

Таблица 1 – Атрибуты сущности «Products».

Атрибут	Описание	Тип, домен
ID	Уникальный идентификатор	int
Name	Наименование товара	varchar(50)
Platform	Внешний ключ к таблице «Platform_type»	int
Price	Цена товара	decimal(10, 2)
Description	Описание товара	varchar(255)
Type	Внешний ключ к таблице «Product_type»	int

2. Таблица «Clients»:

Структура данной сущности представлена на таблице 2.

Таблица 2 – Атрибуты сущности «Clients».

Атрибут	Описание	Тип, домен
ID	Уникальный идентификатор	int
First_name	Имя клиента	varchar(50)
Last_name	Фамилия клиента	varchar(50)
Email	Почта клиента	varchar(50)
Phone	Номер телефона клиента	varchar(50)

3. Таблица «Sales»:

Структура данной сущности представлена на таблице 3.

Таблица 3 – Атрибуты сущности «Sales».

Атрибут	Описание	Тип, домен
ID	Уникальный идентификатор	int
Product_ID	Внешний ключ к таблице «Products»	int
Client_ID	Внешний ключ к таблице «Sales»	int
Sale_date	Дата продажи товара	date

4. Таблица «Platform_type»:

Структура данной сущности представлена на таблице 4.

Таблица 4 – Атрибуты сущности «Platform_type».

Атрибут	Описание	Тип, домен
ID	Уникальный идентификатор	int
PlatformName	Платформа товара	varchar(50)

5. Таблица «Product_type»:

Структура данной сущности представлена на таблице 5.

Таблица 5 – Атрибуты сущности «Product_type».

Атрибут	Описание	Тип, домен
ID	Уникальный идентификатор	int
TypeName	Тип товара	varchar(50)

6. Таблица «AnalystRec»:

Структура данной сущности представлена на таблице 6.

Таблица 6 – Атрибуты сущности «AnalystRec».

Атрибут	Описание	Тип, домен
ID	Уникальный идентификатор	int
Date	Дата продажи	date
Recommendations	Рекомендация	varchar(MAX)

Структура базы данных приведена в виде ER-диаграммы «Сущность-связь» на рисунке 11. Такая структура позволяет эффективно управлять данными, обеспечивая безопасное хранение и облегчая поиск данных. Отношения между таблицами создаются для установления связей, что обеспечивает целостность данных и удобство управления информацией.

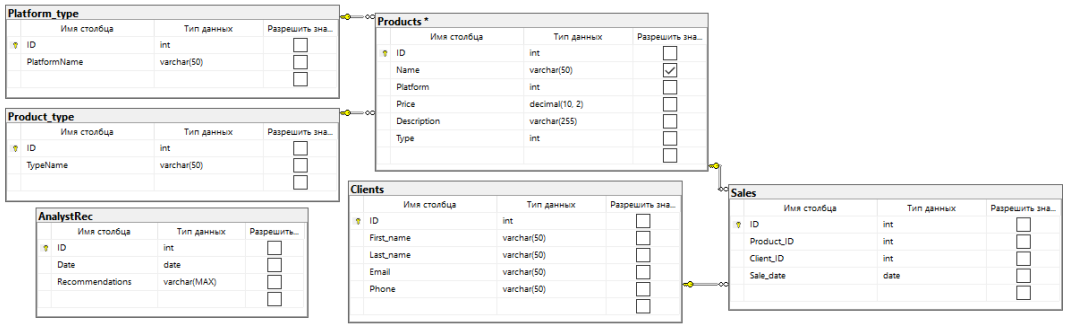

Рисунок 11 – ER-диаграмма базы данных.

3. РАЗРАБОТКА КОНСОЛЬНОГО ПРИЛОЖЕНИЯ

3.1 Описание среды разработки и программных инструментов

Для разработки программного модуля системы учета и анализа данных о продаже игр и развлечений были выбраны следующие программные инструменты:

1. Среда разработки: Visual Studio Community 2022

Описание: бесплатная интегрированная среда разработки (IDE) от Microsoft, предназначенная для создания различных приложений, включая веб-приложения, настольные приложения и мобильные приложения. Она предоставляет обширный набор инструментов и ресурсов для разработчиков, обеспечивая эффективный и удобный рабочий процесс.

2. Язык программирования: C#

Описание: высокоуровневый язык программирования, разработанный Microsoft, который позволяет создавать приложения для платформы .NET с использованием объектно-ориентированного подхода.

3. Платформа: .NET Framework 4.8

Описание: официальная платформа разработки от Microsoft для создания приложений под операционные системы Windows. Версия 4.8 является стабильной и поддерживает широкий спектр функциональности.

4. Библиотеки:

– Microsoft.Bcl.AsyncInterfaces
– Microsoft.Office.Interop.Word
– Newtonsoft.Json
– System.Memory
– System.Numerics.Vectors
– System.Threading.Tasks.Extensions

3.2 Обоснование выбора инструментария по разработке

Выбранный инструментарий обладает несколькими преимуществами:

1. Visual Studio Community 2022:

Мощная и Бесплатная IDE: Visual Studio Community предоставляет полноценную интегрированную среду разработки с широким спектром функциональных возможностей, что позволяет разработчикам эффективно создавать и отлаживать приложения. Бесплатность этой версии делает ее привлекательным выбором для множества проектов, особенно для стартапов, индивидуальных разработчиков и образовательных учреждений.

2. Язык программирования: C#

Эффективность и Современность: C# является мощным, современным языком программирования, разработанным специально для платформы .NET. Он обеспечивает высокую производительность, читаемый синтаксис и широкие возможности в объектно-ориентированном программировании. Выбор C# упрощает разработку и поддержку кода, а также обеспечивает доступ к современным языковым возможностям.

3. Платформа: .NET Framework 4.8

Стабильность и Богатая Функциональность: .NET Framework 4.8 представляет собой стабильную и надежную версию фреймворка, обеспечивающую высокую совместимость приложений на различных уровнях. Богатая библиотека классов и возможности платформы делают его идеальным выбором для создания разнообразных приложений, обеспечивая широкие возможности и удобство в работе с различными типами проектов.

4. Библиотеки:

– Microsoft.Bcl.AsyncInterfaces:

Асинхронные Интерфейсы: предоставляет асинхронные интерфейсы, которые полезны для работы с асинхронными паттернами в современных приложениях.

– Microsoft.Office.Interop.Word:

Интеграция с Microsoft Word: позволяет взаимодействовать с

функциональностью Microsoft Word из приложений, что полезно для автоматизации задач, связанных с обработкой документов Word.

– Newtonsoft.Json:

Обработка JSON: предоставляет мощные средства для работы с форматом данных JSON, что особенно актуально для обмена данными между клиентскими и серверными приложениями.

– System.Memory:

Управление Памятью: предоставляет типы данных для управления памятью, что важно для безопасной работы с блоками памяти в управляемой среде.

– System.Numerics.Vectors:

Векторные Операции: предоставляет поддержку векторных операций, что полезно для высокопроизводительных вычислений, таких как обработка графики или математические расчеты.

– System.Threading.Tasks.Extensions:

Расширения для Задач: предоставляет расширения для библиотеки задач .NET, облегчая работу с асинхронными операциями и улучшая читаемость кода.

3.3 Описание пользовательского интерфейса

1. Окно авторизации:

При открытии приложения пользователь видит окно авторизации. (Рисунок 12)

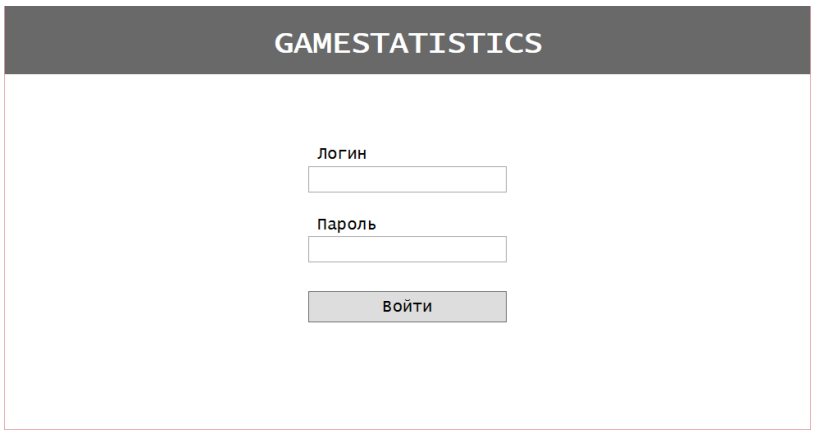

Рисунок 12 – Окно авторизации.

2. Окно администратора и менеджера (Рисунок 13):

Кнопка «Добавить клиента» (Рисунок 14): открывает окно для добавления клиента.

Кнопка «Добавить товар» (Рисунок 15): открывает окно для добавления товара.

Кнопка «Поиск/Удаление товаров» (Рисунок 16): открывает окно для поиска и удаления товаров.

Кнопка «Поиск/Удаление клиентов» (Рисунок 17): открывает окно для поиска и удаления клиентов.

Кнопка «Оформить продажу» (Рисунок 18): открывает окно для оформления продажи.

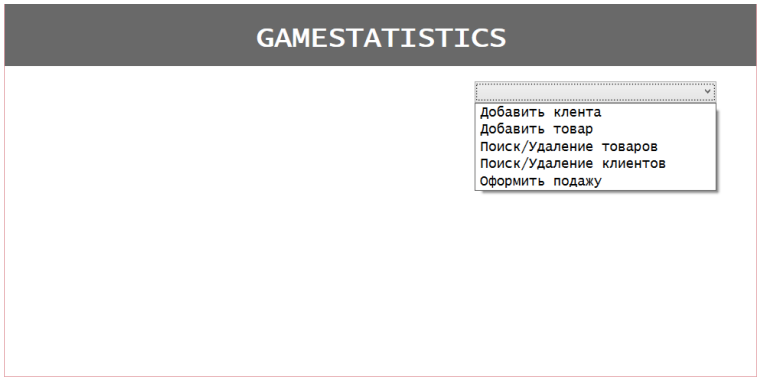

Рисунок 13 – Окно администратора и менеджера.

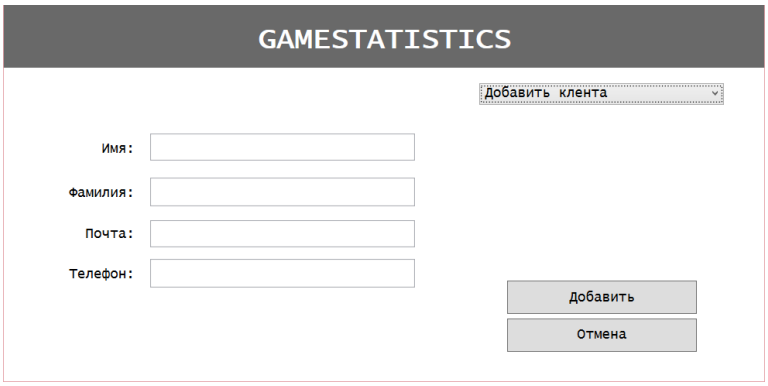

Рисунок 14 – Окно добавления клиента.

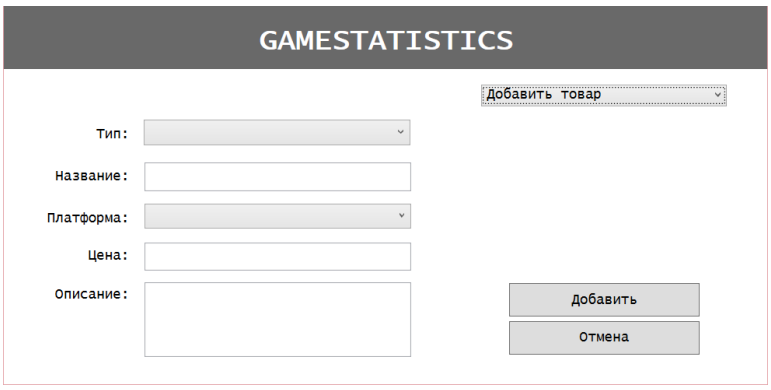

Рисунок 15 – Окно добавления товара.

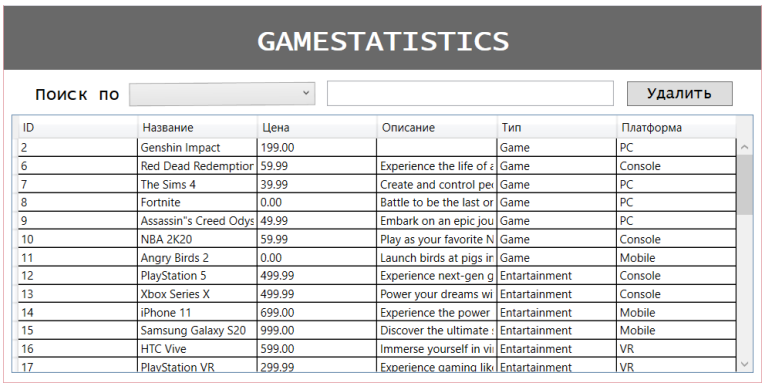

Рисунок 16 – Окно поиска/удаления товаров.

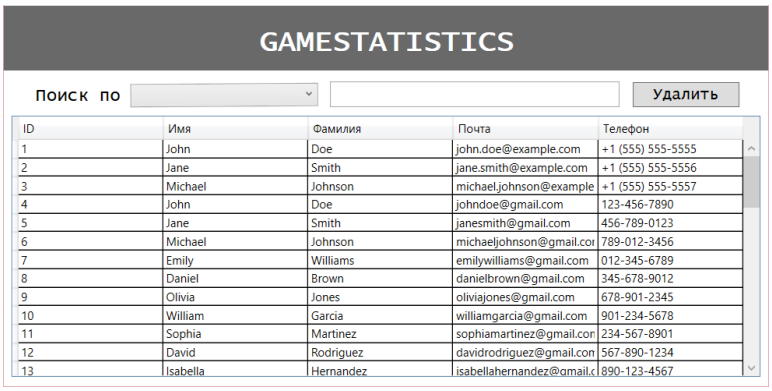

Рисунок 17 – Окно поиска/удаления клиентов.

Рисунок 18 – Окно оформления продажи

3. Окно аналитика (Рисунок 19):

Кнопка «Анализ продаж» (Рисунок 20): открывает окно просмотра анализа продаж.

Кнопка «Рекомендации» (Рисунок 21): открывает окно для написания и просмотра прошлых рекомендаций.

Рисунок 19 – Окно аналитика.

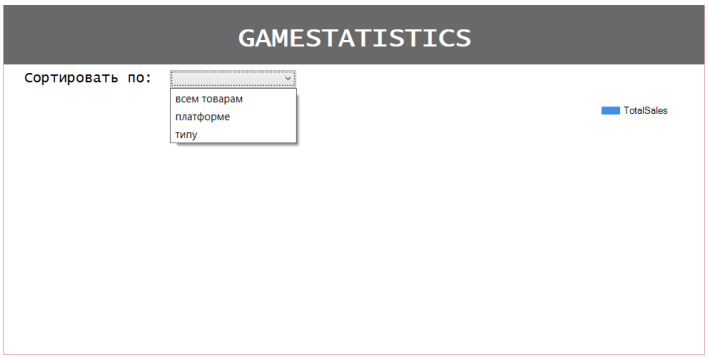

Рисунок 20 – Окно анализа продаж.

Рисунок 21 – Окно написания/просмотра рекомендаций.

3.4 Тест план для системы учета и анализа данных о продаже игр и развлечений

№	Название тест-кейса	Описание тест-кейса	Ожидаемый результат
1	Запуск приложения	Проверка успешного запуска приложения	Приложение запускается
2	Выбор параметра сортировки	Проверка возможности выбора параметра сортировки	Выбранный параметр отображается
3	Обновление данных в DataGrid	Проверка корректности отображения данных в DataGrid	Данные корректно отображаются
4	Построение диаграммы	Проверка корректности построения диаграммы	Диаграмма строится успешно
5	Навигация между страницами	Проверка корректности перехода между страницами	Переход осуществляется без ошибок
6	Работа с базой данных	Проверка корректности работы с базой данных	Данные из базы данных корректно загружаются
7	Обработка ошибок	Проверка корректности обработки ошибок	Ошибки корректно отображаются и обрабатываются

ЗАКЛЮЧЕНИЕ

В заключении курсового проекта "Система статистики игровых продаж" можно отметить успешное завершение разработки приложения, предназначенного для анализа и визуализации данных о продажах игр. Созданное приложение предоставляет удобный пользовательский интерфейс для отображения статистики продаж в разрезе платформ и типов продукции за заданный период времени.

Основной функционал включает в себя возможность выбора периода анализа (неделя, месяц, год) и сортировки данных по типу продукции или платформе. Пользователи могут в режиме реального времени отслеживать изменения в продажах и строить столбчатые диаграммы для визуального анализа данных.

Проект успешно реализован с использованием технологий .NET и WPF для создания интуитивно понятного интерфейса. Entity Framework был использован для взаимодействия с базой данных, где хранятся данные о продажах. Разработанное приложение может быть полезным для компаний, занимающихся разработкой и продажей игр, а также для аналитических отделов, занимающихся мониторингом и анализом рынка видеоигр.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Нормативно-правовые документы:

– ГОСТ 7.32–2001

– ГОСТ 7.0.5–2008

– ГОСТ 7.1–2003

– ГОСТ 7.80–2000

2. Учебники, учебные пособия, статьи:

– «C# 9.0. Карманный справочник (2021)» Джозеф Албахари, Бен Албахари

– C# 10 и .NET 6. Современная кроссплатформенная разработка (2023) Марк Прайс

3. Интернет-источники:

– Официальная документация C# [https://learn.microsoft.com/ru-ru/dotnet/csharp/]

– Полное руководство по языку программирования C# 12 и платформе .NET 8 [https://metanit.com/sharp/tutorial/]

– C# API [https://developer.unigine.com/ru/docs/latest/code/csharp/]

– Документация по Visual Studio [https://learn.microsoft.com/ru-ru/visualstudio/windows/?view=vs-2022]

– Руководство по программированию на C# [https://learn.microsoft.com/ru-ru/dotnet/csharp/programming-guide/]

– Руководство по C# [https://professorweb.ru/my/csharp/charp_theory/level1/index.php]

– Документация по средствам XAML [https://learn.microsoft.com/ru-ru/visualstudio/xaml-tools/?view=vs-2022]

– Как построить графики в Windows Forms. Компонент – Chart [https://dzen.ru/video/watch/623336aef3f62d5fcdd29ad3]

ПРИЛОЖЕНИЕ 1

Листинг 1. Файл LoginPage.xaml.cs

```
using System.Windows;
using System.Windows.Controls;
using System.Windows.Navigation;

namespace GameStatistic
{
    /// <summary>
    /// Логика взаимодействия для LoginPage.xaml
    /// </summary>
    public partial class LoginPage : Page
    {
        public LoginPage()
        {
            InitializeComponent();
        }

        private void enter_button_Click(object sender, RoutedEventArgs e)
        {
            string login = login_text.Text;
            string password = password_text.Password;

            if (login == "manager" && password == "mpass" || login == "admin" &&
password == "ampass")
            {
                NavigationService.Navigate(new ManagerPage1());
            }
            else if (login == "analyst" && password == "apass" || login == "admin" &&
password == "aapass")
            {
                NavigationService.Navigate(new AnalystPage1());
            }
            else
            {
                MessageBox.Show("Неверный логин или пароль.\nПопробуйте снова.");
                login_text.Focus();
            }

            login_text.Clear();
            password_text.Clear();
        }
    }
}
```

ПРИЛОЖЕНИЕ 2

Листинг 2. Файл ManagerPage1.xaml.cs

```
using System.Data.Entity.Infrastructure;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Navigation;
using MessageBox = System.Windows.Forms.MessageBox;

namespace GameStatistic
{
    /// <summary>
    /// Логика взаимодействия для ManagerPage1.xaml
    /// </summary>
    public partial class ManagerPage1 : Page
    {
        public ManagerPage1()
        {
            InitializeComponent();
        }

        private void combobox_choice_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {

            first_name_text.Visibility = Visibility.Collapsed;
            last_name_text.Visibility = Visibility.Collapsed;
            phone_text.Visibility = Visibility.Collapsed;
            email_text.Visibility = Visibility.Collapsed;
            game_name_text.Visibility = Visibility.Collapsed;
            platform_type.Visibility = Visibility.Collapsed;
            product_type.Visibility = Visibility.Collapsed;
            type_label.Visibility = Visibility.Collapsed;
            price_text.Visibility = Visibility.Collapsed;
            description_text.Visibility = Visibility.Collapsed;
            firstname_label.Visibility = Visibility.Collapsed;
            lastname_label.Visibility = Visibility.Collapsed;
            phone_label.Visibility = Visibility.Collapsed;
            email_label.Visibility = Visibility.Collapsed;
            game_name_label.Visibility = Visibility.Collapsed;
            platform_label.Visibility = Visibility.Collapsed;
            price_label.Visibility = Visibility.Collapsed;
            description_label.Visibility = Visibility.Collapsed;
            add_clnt_button.Visibility = Visibility.Collapsed;
            add_tov_button.Visibility = Visibility.Collapsed;
            cencel_button.Visibility = Visibility.Collapsed;


            if (combobox_choice.SelectedIndex == 0)
            {
                first_name_text.Visibility = Visibility.Visible;
                last_name_text.Visibility = Visibility.Visible;
                phone_text.Visibility = Visibility.Visible;
                email_text.Visibility = Visibility.Visible;
                firstname_label.Visibility = Visibility.Visible;
                lastname_label.Visibility = Visibility.Visible;
                phone_label.Visibility = Visibility.Visible;
                email_label.Visibility = Visibility.Visible;
                add_clnt_button.Visibility = Visibility.Visible;
                cencel_button.Visibility = Visibility.Visible;
            }
            else if (combobox_choice.SelectedIndex == 1)
            {
                game_name_text.Visibility = Visibility.Visible;
                platform_type.Visibility = Visibility.Visible;
                price_text.Visibility = Visibility.Visible;
                description_text.Visibility = Visibility.Visible;
                game_name_label.Visibility = Visibility.Visible;
                platform_label.Visibility = Visibility.Visible;
                price_label.Visibility = Visibility.Visible;
                description_label.Visibility = Visibility.Visible;
                product_type.Visibility = Visibility.Visible;
                type_label.Visibility = Visibility.Visible;
                add_tov_button.Visibility = Visibility.Visible;
                cencel_button.Visibility = Visibility.Visible;
            }
            else if (combobox_choice.SelectedIndex == 2)
            {
                NavigationService.Navigate(new ManagerSearch());
            }
            else if (combobox_choice.SelectedIndex == 3)
            {
                NavigationService.Navigate(new ManagerSearchClnt());
            }
            else if (combobox_choice.SelectedIndex == 4)
            {
                NavigationService.Navigate(new AddSale());
            }

        }

        private void add_tov_button_Click(object sender, RoutedEventArgs e)
        {
            if (platform_type.SelectedIndex < 0)
            {
                MessageBox.Show("Выберите платформу.");
                return;
            }
            if (string.IsNullOrWhiteSpace(game_name_text.Text))
            {
                MessageBox.Show("Введите название товара.");
                game_name_text.Focus();
                return;
            }
            if (product_type.SelectedIndex < 0)
            {
```

```
            MessageBox.Show("Выберите тип товара.");
            return;
        }
        if (string.IsNullOrWhiteSpace(price_text.Text))
        {
            MessageBox.Show("Введите цену товара.");
            price_text.Focus();
            return;
        }
        if (!decimal.TryParse(price_text.Text, out decimal price) || price < 0)
        {
            MessageBox.Show("Введите положительное число в поле цены.");
            price_text.Clear();
            price_text.Focus();
            return;
        }
        if (string.IsNullOrWhiteSpace(description_text.Text) || description_text.Text.Length > 255)
        {
            MessageBox.Show("Введите описание товара. Не более 255 символов.");
            description_text.Focus();
            return;
        }

        Products products = new Products();
        products.Name = game_name_text.Text;
        products.Platform = platform_type.SelectedIndex + 1;
        products.Price = price;
        products.Description = description_text.Text;
        products.Type = product_type.SelectedIndex + 1;

        try
        {
            DB.db.Products.Add(products);
            DB.db.SaveChanges();
        }
        catch (DbUpdateException ex)
        {
            MessageBox.Show($"Произошла ошибка при добавлении товара: {ex.Message}");
        }
        finally
        {
            MessageBox.Show("Товар успешно добавлен!");
            game_name_text.Clear();
            platform_type.SelectedIndex = -1;
            price_text.Clear();
            description_text.Clear();
            product_type.SelectedIndex = -1;
        }


    }
```

Листинг 3. Файл ManagerSearch.xaml.cs

```
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Windows;
using System.Windows.Controls;

namespace GameStatistic
{

    public partial class ManagerSearch : Page
    {
        public ObservableCollection<ProductViewModel> YourProductsCollection { get; set; } = new ObservableCollection<ProductViewModel>();

        public ManagerSearch()
        {
            InitializeComponent();
            comboBox.SelectionChanged += ComboBox_SelectionChanged;
            searchtextbox.TextChanged += SearchTextBox_TextChanged;

            LoadData();
        }

        private void LoadData()
        {
            var query = from product in DB.db.Products
                        join productType in DB.db.Product_type on product.Type equals productType.ID
                        join platformType in DB.db.Platform_type on product.Platform equals platformType.ID
                        select new ProductViewModel
                        {
                            ProductID = product.ID,
                            Name = product.Name,
                            Price = product.Price,
                            Description = product.Description,
                            TypeName = productType.TypeName,
                            PlatformName = platformType.PlatformName
                        };

            YourProductsCollection = new ObservableCollection<ProductViewModel>(query.ToList());
            search_datagrid.ItemsSource = YourProductsCollection;
        }

        private List<Products> PerformSearch(string characteristic, string query, string typeQuery, string platformQuery)
        {
            switch (characteristic)
            {
                case "названию":
                    return DB.db.Products.Where(p => p.Name.Contains(query)).ToList();
                case "ID" when int.TryParse(query, out int idQuery):
                    return DB.db.Products.Where(p => p.ID == idQuery).ToList();
                case "цене" when decimal.TryParse(query, out decimal price):
                    return DB.db.Products.Where(p => p.Price < price).ToList();
                case "описанию":
                    return DB.db.Products.Where(p => p.Description.Contains(query)).ToList();
                case "типу":
                    return DB.db.Products
                        .Where(p => (string.IsNullOrEmpty(typeQuery) || DB.db.Product_type.Any(pt => pt.ID == p.Type && pt.TypeName.Contains(typeQuery))) &&
                                    (string.IsNullOrEmpty(platformQuery) || DB.db.Platform_type.Any(pt => pt.ID == p.Platform && pt.PlatformName.Contains(platformQuery))))
                        .ToList();
                case "платформе":
                    return DB.db.Products

                .Where(p => (string.IsNullOrEmpty(platformQuery) || DB.db.Platform_type.Any(pt => pt.ID == p.Platform && pt.PlatformName.Contains(platformQuery))) &&
                            (string.IsNullOrEmpty(typeQuery) || DB.db.Product_type.Any(pt => pt.ID == p.Type && pt.TypeName.Contains(typeQuery))))
                .ToList();
        default:
            return new List<Products>();
    }
}

private void SearchTextBox_TextChanged(object sender, TextChangedEventArgs e)
{
    string selectedCharacteristic = (comboBox.SelectedItem as ComboBoxItem)?.Content?.ToString();
    string query = searchtextbox.Text;
    string typeQuery = (selectedCharacteristic == "типу") ? searchtextbox.Text : null;
    string platformQuery = (selectedCharacteristic == "платформе") ? searchtextbox.Text : null;

    if (!string.IsNullOrEmpty(selectedCharacteristic) && !string.IsNullOrEmpty(query))
    {
        List<Products> searchResults = PerformSearch(selectedCharacteristic, query, typeQuery, platformQuery);

        YourProductsCollection.Clear();
        foreach (var searchResult in searchResults)
        {
            YourProductsCollection.Add(new ProductViewModel
            {
                ProductID = searchResult.ID,
                Name = searchResult.Name,
                Price = searchResult.Price,
                Description = searchResult.Description,
                TypeName = DB.db.Product_type.FirstOrDefault(pt => pt.ID == searchResult.Type)?.TypeName,
                PlatformName = DB.db.Platform_type.FirstOrDefault(pt => pt.ID == searchResult.Platform)?.PlatformName
            });
        }

        search_datagrid.ItemsSource = YourProductsCollection;
    }
}

private void ComboBox_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    searchtextbox.Clear();
    searchtextbox.Focus();
}

private void DeleteSelectedProduct()
{
    // Получаем выбранный элемент из DataGrid
    ProductViewModel selectedProduct = search_datagrid.SelectedItem as ProductViewModel;
```

```
        if (selectedProduct != null)
        {
            // Получаем соответствующий объект Products из базы данных
            Products productToDelete = DB.db.Products.FirstOrDefault(p => p.ID == selectedProduct.ProductID);

            if (productToDelete != null)
            {
                // Удаляем объект из базы данных
                DB.db.Products.Remove(productToDelete);
                DB.db.SaveChanges();

                // Удаление объекта из коллекции для отображения


                    YourProductsCollection.Remove(selectedProduct);

                    MessageBox.Show("Продукт удален успешно.");
                }
            }
            else
            {
                MessageBox.Show("Выберите продукт для удаления.");
            }
        }

        private void btnDelete_Click(object sender, System.Windows.RoutedEventArgs e)
        {
            DeleteSelectedProduct();
        }
    }
}
```

ПРИЛОЖЕНИЕ 4

Листинг 4. Файл ProductViewModel.xaml.cs

```
public class ProductViewModel
    {
        public int ProductID { get; set; }
        public string Name { get; set; }
        public decimal Price { get; set; }
        public string Description { get; set; }
        public string TypeName { get; set; }
        public string PlatformName { get; set; }
    }
```

ПРИЛОЖЕНИЕ 5

Листинг 5. Файл ManagerSearchClnt.xaml.cs

```
using System.Collections.Generic;
using System.Linq;
using System.Windows;
using System.Windows.Controls;

namespace GameStatistic
{
    public partial class ManagerSearchClnt : Page
    {
        public ManagerSearchClnt()
        {
            InitializeComponent();

            comboBox.SelectionChanged += ComboBox_SelectionChanged;
            searchtextbox.TextChanged += SearchTextBox_TextChanged;

            LoadData();
        }

        private void LoadData()
        {
            search_datagrid.ItemsSource = DB.db.Clients.ToList();
        }

        private void SearchTextBox_TextChanged(object sender, TextChangedEventArgs e)
        {
            string selectedCharacteristic = (comboBox.SelectedItem as ComboBoxItem)?.Content?.ToString();
            string query = searchtextbox.Text;

            if (!string.IsNullOrEmpty(selectedCharacteristic) && !string.IsNullOrEmpty(query))
            {
                List<Clients> searchResults = PerformSearch(selectedCharacteristic, query);
                search_datagrid.ItemsSource = searchResults;
            }
            else
            {
                LoadData();
            }
        }

        private List<Clients> PerformSearch(string characteristic, string query)
        {
            switch (characteristic)
            {
                case "ID" when int.TryParse(query, out int idQuery):
                    return DB.db.Clients.Where(p => p.ID == idQuery).ToList();
                case "имени":
                    return DB.db.Clients.Where(p => p.First_name.Contains(query)).ToList();
                case "фамилии":
                    return DB.db.Clients.Where(p => p.Last_name.Contains(query)).ToList();
                case "почте":
                    return DB.db.Clients.Where(p => p.Email.Contains(query)).ToList();
                case "номеру телефона":
                    return DB.db.Clients.Where(p => p.Phone.Contains(query)).ToList();
                default:
                    break;
            }

            return new List<Clients>();
        }

        private void ComboBox_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            searchtextbox.Clear();
            searchtextbox.Focus();
        }

        private void DeleteSelectedClient()
        {
            // Получаем выбранного клиента из DataGrid
            Clients selectedClient = search_datagrid.SelectedItem as Clients;

            if (selectedClient != null)
            {
                // Удаляем клиента из базы данных
                DB.db.Clients.Remove(selectedClient);
                DB.db.SaveChanges();

                // Удаляем клиента из коллекции для отображения
                search_datagrid.ItemsSource = DB.db.Clients.ToList();

                MessageBox.Show("Клиент удален успешно.");
            }
            else
            {
                MessageBox.Show("Выберите клиента для удаления.");
            }
        }

        private void btnDelete_Click(object sender, RoutedEventArgs e)
        {
            DeleteSelectedClient();
        }
    }
}
```

ПРИЛОЖЕНИЕ 6

Листинг 6. Файл AddSale.xaml.cs

```
using System.Linq;
using System.Windows;
using System.Windows.Controls;

namespace GameStatistic
{
    public partial class AddSale : Page
    {
        public AddSale()
        {
            InitializeComponent();

            // Заполните выпадающие списки клиентов и продуктов
            clientComboBox.ItemsSource = DB.db.Clients.ToList();
            productComboBox.ItemsSource = DB.db.Products.ToList();
        }

        private void AddSale_Click(object sender, RoutedEventArgs e)
        {
            // Получите выбранных клиента, продукт и дату продажи
            var selectedClient = clientComboBox.SelectedItem as Clients;
            if (selectedClient != null)
            {
                // Access the first and last names separately
                string firstName = selectedClient.First_name;
                string lastName = selectedClient.Last_name;

                // Use the concatenated full name if needed
                string fullName = $"{lastName} {firstName}";
            }

            var selectedProduct = productComboBox.SelectedItem as Products;
            var saleDate = saleDatePicker.SelectedDate;

            if (selectedClient != null && selectedProduct != null && saleDate.HasValue)
            {

                var sale = new Sales
                {
                    Product_ID = selectedProduct.ID,
                    Client_ID = selectedClient.ID,
                    Sale_date = saleDate.Value
                };

                DB.db.Sales.Add(sale);
                DB.db.SaveChanges();

                MessageBox.Show("Продажа успешно оформлена!");

                saleDatePicker.SelectedDate = null;
                clientComboBox.SelectedItem = null;
                productComboBox.SelectedItem = null;
            }
            else
            {
                MessageBox.Show("Пожалуйста, выберите клиента, продукт и дату продажи.");
            }
        }
    }
}
```

ПРИЛОЖЕНИЕ 7

Листинг 7. Файл AnalystPage1.xaml.cs

```
using System.Windows;
using System.Windows.Controls;
using System.Windows.Navigation;

namespace GameStatistic
{
    /// <summary>
    /// Логика взаимодействия для AnalystPage1.xaml
    /// </summary>
    public partial class AnalystPage1 : Page
    {
        public AnalystPage1()
        {
            InitializeComponent();
        }

        private void rec_button_Click(object sender, RoutedEventArgs e)
        {
            NavigationService.Navigate(new Recommendation());
        }

        private void analys_sales_button_Click(object sender, RoutedEventArgs e)
        {
            NavigationService.Navigate(new SalesAnalys());
        }

    }
}
```

ПРИЛОЖЕНИЕ 8

Листинг 8. Файл Recommendation.xaml.cs

```
using System;
using System.Linq;
using System.Windows;
using System.Windows.Controls;

namespace GameStatistic
{
    public partial class Recommendation : System.Windows.Controls.Page
    {
        public Recommendation()
        {
            InitializeComponent();

            // Загрузка прошлых рекомендаций из базы данных
            LoadPreviousRecommendations();
        }

        private void SaveToDatabase_Click(object sender, RoutedEventArgs e)
        {
            string recommendation = rec_textbox.Text;

            if (!string.IsNullOrEmpty(recommendation))
            {
                // Сохранение в базу данных
                DB.db.AnalystRec.Add(new AnalystRec
                {
                    Recommendations = recommendation,
                    Date = DateTime.Now
                });

                DB.db.SaveChanges();

                // Обновление списка прошлых рекомендаций
                LoadPreviousRecommendations();
            }
        }

        private dynamic selectedRecommendation;

        private void Oldrec_datadrid_SelectedCellsChanged(object sender, SelectedCellsChangedEventArgs e)
        {
            // Получаем выбранную строку
            selectedRecommendation = oldrec_datadrid.SelectedItem as dynamic;

            // Обновляем TextBox выбранной рекомендацией
            if (selectedRecommendation != null)
            {
                rec_textbox.Text = $"{selectedRecommendation.Recommendations}";
            }
        }

        private void SaveToWord_Click(object sender, RoutedEventArgs e)
        {
            // Проверяем, выбрана ли рекомендация
            if (selectedRecommendation != null)
            {
                string recommendation = selectedRecommendation.Recommendations;
                DateTime date = selectedRecommendation.Date;

                if (!string.IsNullOrEmpty(recommendation))

                {
                    // Сохранение в Word с использованием даты добавления
                    SaveToWordDocument(date, recommendation);
                }
            }
            else
            {
                MessageBox.Show("Выберите рекомендацию для сохранения.", "Предупреждение");
            }
        }

        private void SaveToWordDocument(DateTime date, string recommendation)
        {
            // Имя файла Word
            string fileName = $"Recommendation_{date.ToString("dd.MM.yy")}.docx";

            try
            {
                // Создание объекта Word
                var wordApp = new Microsoft.Office.Interop.Word.Application();
                var doc = wordApp.Documents.Add();

                // Добавление текста в документ
                doc.Content.Text = $"{date} - {recommendation}";

                // Сохранение документа
                doc.SaveAs2(fileName);

                // Закрытие Word
                doc.Close();
                wordApp.Quit();

                MessageBox.Show($"Рекомендации сохранены в файле {fileName}", "Сохранено");
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка при сохранении в Word: {ex.Message}", "Ошибка");
            }
        }


        private void LoadPreviousRecommendations()
        {
            // Загрузка прошлых рекомендаций из базы данных
            var previousRecommendations = DB.db.AnalystRec.Select(rec => new { rec.ID, rec.Date, rec.Recommendations }).ToList();

            // Отображение в DataGrid
            oldrec_datadrid.ItemsSource = previousRecommendations;
        }

    }
}
```

ПРИЛОЖЕНИЕ 9

Листинг 9. Файл SalesAnalys.xaml.cs

```
using System.Collections.Generic;
using System.Linq;
using System.Windows;
using System.Windows.Controls;

namespace GameStatistic
{
    public partial class SalesAnalys : System.Windows.Controls.Page
    {
        public SalesAnalys()
        {
            InitializeComponent();
            Loaded += SalesAnalys_Loaded;
        }
        private void SalesAnalys_Loaded(object sender, RoutedEventArgs e)
        {
            LoadChartData();
        }
        private void SortComboBox_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            LoadChartData();
        }
        private void LoadChartData()
        {
            var selectedIndex = sortComboBox.SelectedIndex;

            switch (selectedIndex)
            {
                case 0:
                    LoadTotalSalesData();
                    break;
                case 1:
                    LoadPlatformData();
                    break;
                case 2:
                    LoadProductTypeData();
                    break;
            }
        }
        private void LoadTotalSalesData()
        {
            var query = from sale in DB.db.Sales
                        join product in DB.db.Products on sale.Product_ID equals product.ID
                        group new { sale, product } by new { Year = sale.Sale_date.Year, Month = sale.Sale_date.Month } into grouped
                        orderby grouped.Key.Year, grouped.Key.Month
                        select new ChartData { Label = grouped.Key.Month.ToString(), Value = grouped.Sum(s => s.product.Price) };

            UpdateChart(query, "Сумма продаж по месяцам", "Месяц", "Сумма продаж");
        }
        private void LoadPlatformData()
        {
            var query = from sale in DB.db.Sales
                        join product in DB.db.Products on sale.Product_ID equals product.ID
                        join platform in DB.db.Platform_type on product.Platform equals platform.ID
                        group new { sale, product, platform } by new { Platform = platform.PlatformName } into grouped
                        orderby grouped.Key.Platform
                        select new ChartData { Label = grouped.Key.Platform, Value = grouped.Sum(s => s.product.Price) };

            UpdateChart(query, "Сумма продаж по платформам", "Платформа", "Сумма продаж");
        }
        private void LoadProductTypeData()
        {
            var query = from sale in DB.db.Sales
                        join product in DB.db.Products on sale.Product_ID equals product.ID
                        join productType in DB.db.Product_type on product.Type equals productType.ID
                        group new { sale, product, productType } by new { ProductType = productType.TypeName } into grouped
```

```
                    orderby grouped.Key.ProductType
                    select new ChartData { Label = grouped.Key.ProductType, Value = grouped.Sum(s => s.product.Price) };

            UpdateChart(query, "Сумма продаж по типам товаров", "Тип товара", "Сумма продаж");
        }
        private void UpdateChart(IEnumerable<ChartData> query, string title, string xTitle, string yTitle)
        {
            // Очистка предыдущих данных
            chart.Series["TotalSales"].Points.Clear();

            // Преобразование данных для использования в диаграмме
            var chartData = query.ToList();
            var series = chart.Series["TotalSales"];

            foreach (var dataPoint in chartData)
            {
                series.Points.AddXY(dataPoint.Label, dataPoint.Value);
            }
            chart.ChartAreas[0].AxisX.Interval = 1;
            chart.ChartAreas[0].AxisX.MajorGrid.Enabled = false;
            chart.ChartAreas[0].AxisX.Title = xTitle;
            chart.ChartAreas[0].AxisY.Title = yTitle;
            chart.Legends[0].Enabled = true;
            chart.Titles.Clear();
            chart.Titles.Add(title);
        }
        public class ChartData
        {
            public string Label { get; set; }
            public decimal Value { get; set; }
        }
    }
}
```

ПРИЛОЖЕНИЕ 10

Листинг 10. Файл DB.xaml.cs

```
namespace GameStatistic
{
    public static class DB
    {
        private static GameStatisticsEntities3 database;
        public static GameStatisticsEntities3 db
        {
            get
            {
                if (database == null)
                    database = new GameStatisticsEntities3();

                return database;

            }
        }
    }
}
```