СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc162003097)

[1 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 6](#_Toc162003098)

[1.1 Вводный инструктаж по технике безопасности во время прохождения практики 7](#_Toc162003099)

[1.1.1 Общие положения 7](#_Toc162003100)

[1.1.2 Требования охраны труда перед началом работы 7](#_Toc162003101)

[1.1.3 Требования охраны труда во время работы 8](#_Toc162003102)

[1.1.4 Требования охраны труда в аварийных ситуациях 9](#_Toc162003103)

[1.1.5 Требования охраны труда по окончании работы 10](#_Toc162003104)

[1.2 Анализ предметной области 10](#_Toc162003105)

[1.3 Выработка требований к программного обеспечению 11](#_Toc162003106)

[1.3.1 Функциональные требования 11](#_Toc162003107)

[1.3.2 Нефункциональные требования 12](#_Toc162003108)

[1.4 Разработка технического задания в соответствии с индивидуальным заданием 14](#_Toc162003109)

[1.5 Проектирование программного обеспечения 16](#_Toc162003110)

[2 СРЕДСТВА РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 28](#_Toc162003111)

[2.1 Программная реализация приложения 28](#_Toc162003112)

[2.1.1 Проектирование интерфейса приложения 29](#_Toc162003113)

[2.1.2 Программирование и интерпретация результатов 30](#_Toc162003114)

[2.1.3 Реализация алгоритма интеграции данных 34](#_Toc162003115)

[2.2 Отладка приложения 37](#_Toc162003116)

[2.3 Тестирование приложения 38](#_Toc162003117)

[2.3.1 Выбор и обоснование методики проведения тестирования 40](#_Toc162003118)

[2.3.2 Схема выполнения экспериментального тестирования 41](#_Toc162003119)

[2.3.3 Документирование результатов тестирования 42](#_Toc162003120)

[3 МОДЕЛИРОВАНИЕ В ПРОГРАММНЫХ СИСТЕМАХ 44](#_Toc162003121)

[3.1 Нахождение кратчайших путей на графе. 44](#_Toc162003122)

[3.2 Решение задачи об оптимальных ресурсах 46](#_Toc162003123)

[3.3 Определение корреляционной зависимости между рядами наблюдений 52](#_Toc162003124)

[ЗАКЛЮЧЕНИЕ 59](#_Toc162003125)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 60](#_Toc162003126)

[ПРИЛОЖЕНИЕ А ТЕХНИЧЕСКОЕ ЗАДАНИЕ 62](#_Toc162003127)

[ПРИЛОЖЕНИЕ Б РЕАЛИЗАЦИЯ ИНТЕРФЕЙСА ПРИЛОЖЕНИЯ 70](#_Toc162003139)

[ПРИЛОЖЕНИЕ В ЛИСТИНГ ПРОГРАММНОГО КОДА 77](#_Toc162003140)

# ВВЕДЕНИЕ

Учебная практика представляет собой обязательную часть образовательного процесса, направленную на закрепление теоретических знаний и приобретение практических навыков.

Основной целью практики является предоставление возможности применять полученные теоретические знания и практические навыки в реальных условиях, что позволит закрепить и углубить базовые компетенции, необходимые для успешной профессиональной деятельности. Практика дает возможность испытать знания и навыки в реальных рабочих ситуациях, что помогает лучше понять, как применять знания на практике. Кроме того, практика позволяет развивать важные профессиональные навыки, такие как коммуникация, работа в команде и решение проблем.

Задачи учебной практики охватывают широкий спектр направлений, включая:

* Изучить требования охраны труда при разработке программного обеспечения.
* Проанализировать предметную область и выработать требования к программному обеспечению.
* Разработать техническое задание на программное обеспечение.
* Спроектировать программное обеспечение.
* Реализовать программное обеспечение с использованием соответствующих средств разработки.
* Отладить и протестировать программное обеспечение.
* Углубление знаний и понимания изучаемой дисциплины: Практика позволяет применять теоретические знания на практике, закрепляя и углубляя понимание предмета;
* Изучить методы нахождения кратчайших путей на графе.
* Изучить методы решения задачи об оптимальных ресурсах.
* Изучить методы определения корреляционной зависимости между рядами наблюдений.
* Применить изученные методы для решения практических задач.
* Адаптация к профессиональной среде: Практика дает представление о реальных рабочих процессах и требованиях, помогая им адаптироваться к будущей профессиональной деятельности;
* Развитие аналитических, коммуникативных и межличностных навыков;
* Возможность применить теоретические знания на практике: Практика предоставляет возможность применить теоретические знания, полученные в ходе обучения, к решению реальных задач, что способствует развитию практических навыков;

Отчет по учебной практике является документом, демонстрирующим получение обучающегося практического опыта поиска творческих источников, участия в моделировании, создания тематической коллекции по следующим профессиональным компетенциям:

ПК 2.1. Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент.

ПК 2.2. Выполнять интеграцию модулей в программное обеспечение.

ПК 2.3. Выполнять отладку программного модуля с использованием специализированных программных средств.

ПК 2.4. Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения.

ПК 2.5. Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования.

На учебную практику по данному модулю отводится 108 часов.

Для выполнения заданий учебной практики необходимо следующее программное обеспечение:

* Microsoft Office.
* Microsoft Visio (аналог, поддерживающий язык визуального моделирования UML).
* Среда разработки (Microsoft Visual Studio).
* Графический редактор для разработки эскизов (Microsoft Visio, Figma).
* Математические онлайн калькуляторы.

# РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Разработка программного обеспечения - это процесс создания, развертывания и обслуживания программного обеспечения, которое автоматизирует процессы, повышает эффективность, создает новые возможности для бизнеса и удовлетворяет потребности клиентов.

Таблица 1. стадии разработки.

|  |  |  |
| --- | --- | --- |
| Стадии разработки | Этапы работ | Содержание работ |
| 1. Техническое задание | Этап работ, на котором определяются требования к разрабатываемому программному продукту на основе анализа предметной области и потребностей заказчика. | Содержание работ включает в себя формулировку целей и задач проекта, определение функциональных и нефункциональных требований, а также составление технического задания в соответствии с установленными стандартами. |
| 2. Эскизный проект | На этом этапе разрабатывается общая концепция программного продукта, его архитектура и основные функциональные элементы. | Содержание работ включает в себя создание макетов пользовательского интерфейса, определение основных модулей и их взаимодействие, а также проработку основных алгоритмов работы системы. |
| 3. Технический проект | На этапе технического проекта происходит детализация концепции, разработка архитектуры программного продукта, выбор технологий и инструментов разработки. | Содержание работ включает в себя создание диаграмм классов, последовательности, компонентов, а также разработку технической документации, описывающей внутреннюю структуру и особенности реализации системы. |
| 4. Рабочий проект | На этом этапе разрабатывается программный код, реализующий все ранее определенные функции и возможности системы. | Содержание работ включает в себя написание и отладку программного кода, интеграцию различных компонентов системы, а также тестирование на соответствие требованиям. |
| 5. Внедрение | На этапе внедрения программное обеспечение готовится к установке и использованию конечными пользователями. | Содержание работ включает в себя подготовку документации для пользователей, обучение персонала, установку и настройку программного обеспечения, а также оказание поддержки и сопровождение системы после ее внедрения. |

## Вводный инструктаж по технике безопасности во время прохождения практики

1.1.1 Общие положения

* Работа обучающихся в компьютерном классе разрешается только в присутствии преподавателя (лаборанта).
* Во время занятий посторонние лица могут находиться в классе только с разрешения преподавателя.
* Во время перемен между уроками проводится обязательное проветривание компьютерного кабинета с обязательным выходом обучающихся из класса.

Каждый обучающийся в ответе за состояние своего рабочего места и сохранность размещенного на нем оборудования.

Обучающийся, допустивший нарушение по технике безопасности и правилам поведения в компьютерном классе, может быть привлечен к дисциплинарной ответственности. Если нарушение техники безопасности связано с причинением имущественного ущерба, обучающийся несет и материальную ответственность в установленном законом порядке.

1.1.2 Требования охраны труда перед началом работы

Прежде чем приступать к работе за компьютером, обучающийся должен:

* убедиться в отсутствии видимых повреждений на рабочем месте;
* разместить на столе тетради, учебные пособия так, чтобы они не мешали работе на компьютере;
* принять правильною рабочую позу;
* посмотреть на индикатор монитора и системного блока и определить, включён или выключен компьютер. Переместите мышь, если компьютер находится в энергосберегающем состоянии или включить монитор, если он был выключен;
* проверить правильность подключения оборудования к электросети.

1.1.3 Требования охраны труда во время работы

Находясь в компьютерном классе, обучающиеся обязаны:

* соблюдать тишину и порядок;
* выполнять требования преподавателя и лаборанта;
* находясь в сети, работать только под своим именем и паролем;
* соблюдать режим работы (согласно санитарным правилам и нормам);
* при появлении рези в глазах, резком ухудшении видимости, невозможности сфокусировать взгляд или навести его на резкость, появления боли в пальцах и кистях рук, усиления сердцебиения немедленно покинуть рабочее место, сообщить о происшедшем преподавателю и обратиться к врачу;
* после окончания работы завершить все активные программы и корректно выключить компьютер;
* оставить рабочее место чистым.

Работая за компьютером, необходимо соблюдать правила:

* расстояние от экрана до глаз — 70 - 80 см (расстояние вытянутой руки);
* вертикально прямая спина;
* плечи опущены и расслаблены;
* ноги на полу и не скрещены;
* локти, запястья и кисти рук на одном уровне;
* локтевые, тазобедренные, коленные, голеностопные суставы под прямым углом.

При работе в компьютерном классе запрещается:

* находиться в классе в верхней одежде;
* класть одежду и сумки на столы;
* находиться в классе с напитками и едой;
* располагаться сбоку или сзади от включенного монитора;
* присоединять или отсоединять кабели, трогать разъемы, провода и розетки;
* передвигать компьютеры и мониторы;
* открывать системный блок;
* пытаться самостоятельно устранять неисправности в работе аппаратуры;
* перекрывать вентиляционные отверстия на системном блоке и мониторе;
* ударять по клавиатуре, нажимать бесцельно на клавиши;
* класть книги, тетради и другие вещи на клавиатуру, монитор и системный блок;
* приносить и запускать компьютерные игры.
* переключать разъемы интерфейсных кабелей периферийных устройств при включенном питании;
* допускать попадание влаги на поверхность системного блока (процессора), монитора, рабочую поверхность клавиатуры, дисководов, принтеров и других устройств.

1.1.4 Требования охраны труда в аварийных ситуациях

* Во всех случаях обрыва проводов питания, неисправности заземления и других повреждений, появления гари, немедленно отключить питание и сообщить об аварийной ситуации руководителю.
* Не приступать к работе до устранения неисправностей.
* При получении травм или внезапном заболевании немедленно известить своего руководителя, организовать первую доврачебную помощь или вызвать скорую медицинскую помощь.
* При появлении программных ошибок или сбоях оборудования учащийся должен немедленно обратиться к преподавателю (лаборанту).
* При появлении запаха гари, необычного звука немедленно прекратить работу, и сообщить преподавателю (лаборанту).

1.1.5 Требования охраны труда по окончании работы

* Отключить питание компьютера.
* Привести в порядок рабочее место.
* Выполнить упражнения для глаз и пальцев рук на расслабление.

## Анализ предметной области

Для разработки программного модуля, содержащего сведения о продаже компьютерных игр, важно провести анализ предметной области. Этот анализ позволит понять основные аспекты и требования к функциональности программы.

Ниже представлен анализ основных аспектов предметной области:

Игры:

* каждая игра характеризуется названием, фирмой-производителем, ценой, системными требованиями и стилем игры;
* названия игр должны быть уникальными;
* цена игры может быть выражена в определенной валюте или других единицах измерения;
* стиль игры определяет жанр или тип игры, такие как стратегия, шутер, ролевая игра и т. д.

Фирмы-производители:

* каждая игра связана с определенной фирмой-производителем;
* фирмы-производители могут иметь несколько игр в своем ассортименте.

Функциональность программного модуля:

* необходимо реализовать функции для формирования списка игр, анализа данных по производителям, поиска игр по названию игры;
* вывод информации о каждой игре должен быть структурированным и информативным, чтобы пользователь мог легко ознакомиться с основными характеристиками.

Безопасность данных:

* важно обеспечить безопасность хранения и обработки данных о продаже компьютерных игр, особенно если программа будет использоваться в коммерческих целях.

Удобство использования:

* интерфейс программного модуля должен быть интуитивно понятным и удобным для пользователя;
* пользователи должны иметь возможность легко находить необходимую информацию и выполнять различные запросы.

## Выработка требований к программного обеспечению

На основе проведенного анализа предметной области можно выработать следующие требования к программному модулю:

### Функциональные требования

* модуль должен предоставлять возможность добавления, удаления и редактирования информации о компьютерных играх;
* пользователь должен иметь возможность просматривать список всех игр, а также осуществлять поиск игр по названию;
* для каждой игры должна отображаться полная информация о её характеристиках, включая название, фирму-производителя, цену, системные требования и стиль игры.

Надежность:

* модуль должен обеспечивать сохранность данных при любых операциях с базой данных, включая добавление, удаление и редактирование записей;
* при возникновении ошибок или сбоев в работе программы, пользователь должен получать соответствующие сообщения об ошибке с указанием причины и способов их устранения.

Производительность:

* модуль должен обеспечивать быстрый доступ к информации о компьютерных играх, даже при большом объеме данных;
* поиск игр по различным критериям должен производиться эффективно и без заметных задержек.

Удобство использования:

* пользователь должен иметь возможность выполнять все основные операции с данными (добавление игр, удаление игр) с минимальным числом кликов мыши и нажатий клавиш.

Безопасность:

* доступ к функционалу модуля должен быть защищен паролем или другими методами аутентификации, чтобы предотвратить несанкционированный доступ к данным;
* модуль должен обеспечивать защиту данных от несанкционированного доступа и изменений.

Выработка требований к программному обеспечению поможет определить основные цели и функциональность модуля, а также установить стандарты качества, которые необходимо соблюдать в процессе его разработки.

### Нефункциональные требования

Нефункциональные требования определяют свойства и ограничения, касающиеся работы программного модуля, а не его функциональности. Ниже перечислены нефункциональные требования к программному модулю для управления информацией о продаже компьютерных игр:

Производительность:

* модуль должен обеспечивать высокую скорость отклика интерфейса при взаимодействии с пользователем;
* время выполнения операций по добавлению, удалению и редактированию записей должно быть минимальным даже при работе с большим объемом данных.

Надежность:

* вероятность возникновения ошибок или сбоев в работе модуля должна быть минимальной;
* модуль должен быть устойчив к непредвиденным ситуациям, таким как потеря связи с базой данных или недоступность некоторых функциональных возможностей.

Безопасность:

* доступ к административным функциям модуля должен быть ограничен только авторизованным пользователям с соответствующими правами доступа.

Удобство использования:

* интерфейс модуля должен быть интуитивно понятным и легким для освоения даже пользователями без технического образования;
* модуль должен поддерживать многозадачность и возможность одновременной работы нескольких пользователей.

Совместимость:

* программный модуль должен быть совместим с различными операционными системами, такими как Windows, MacOS и Linux;
* интерфейс модуля должен быть адаптивным и корректно отображаться на различных устройствах, включая компьютеры, планшеты и смартфоны.

Масштабируемость:

* модуль должен быть способен масштабироваться в зависимости от объема данных и количества одновременных пользователей;
* при необходимости модуль должен поддерживать расширение функциональности и добавление новых возможностей без изменения основной структуры программы.

Нефункциональные требования определяют важные аспекты работы программного модуля, которые влияют на его эффективность, безопасность и удобство использования. Учитывая эти требования, можно создать программный модуль, который будет соответствовать ожиданиям пользователей и стандартам качества.

## Разработка технического задания в соответствии с индивидуальным заданием

ГОСТ 19.201-78. Единая система программной документации. Техническое задание

Стандарт ГОСТ 19.201-78 устанавливает общие требования к содержанию и оформлению технического задания (ТЗ) на разработку программного обеспечения (ПО).

Техническое задание разрабатывается для создания программного модуля, предназначенного для управления информацией о продаже компьютерных игр. Модуль должен обеспечивать функциональность по добавлению, удалению, редактированию и поиску записей о компьютерных играх. Полное техническое задание представлено в ПРИЛОЖЕНИИ А

Общая цель разработки ПО

Целью разработки программного модуля является создание эффективного инструмента для управления информацией о продаже компьютерных игр, которая обеспечивает простоту использования, высокую производительность и безопасность.

Состав подсистем и функциональных задач

Подсистема управления играми:

* добавление новых игр в базу данных;
* удаление игр из базы данных;
* редактирование информации о играх;
* поиск игр по навазнию.

Подсистема аутентификации и авторизации:

* аутентификация пользователей перед доступом к административным функциям;
* авторизация пользователей на основе их роли и прав доступа.

Требования к подсистемам

Подсистема управления играми:

* должна быть реализована на языке программирования visual studio;
* для хранения данных использовать реляционную базу данных SQL;
* интерфейс пользователя должен быть реализован в виде консольного приложения.

Подсистема аутентификации и авторизации:

* должна обеспечивать безопасность доступа к административным функциям модуля;
* реализация должна соответствовать принципам безопасности информации.

Требования к информационной базе, математическому и программному обеспечению

Информационная база:

* база данных должна быть надежной и обеспечивать сохранность данных;
* вся информация должна храниться в нормализованном виде.

Математическое и программное обеспечение:

* программный модуль должен быть написан с использованием современных методов программирования;
* необходимо обеспечить эффективное управление ресурсами системы.

Общие требования к проектируемой системе

Производительность:

* система должна обеспечивать высокую скорость выполнения операций и отзывчивость интерфейса.

Надежность:

* работа системы должна быть стабильной и безопасной;

Безопасность:

* система должна обеспечивать защиту данных и безопасность доступа к административным функциям.

Перечень задач создания системы и исполнителей

Задачи создания системы:

* разработка архитектуры системы;
* реализация функциональности управления играми;
* реализация подсистемы аутентификации и авторизации;
* тестирование и отладка системы;
* документирование кода и подготовка документации.

Исполнители:

* программисты;
* тестировщики;
* аналитики.

Этапы создания системы и сроки их выполнения

* разработка архитектуры системы: 1 месяц.
* реализация функциональности: 2 месяца.
* тестирование и отладка: 2 недели.
* документирование и подготовка к релизу: 2 недели.

## Проектирование программного обеспечения

Проектирование программного обеспечения для информационной системы по продажам игр должно быть сосредоточено на создании простой в использовании, надежной и масштабируемой системы.

Определение пользователей системы:

* идентификация основных пользовательских групп, включая администраторов и конечных пользователей.
* описание потребностей и ожиданий каждой пользовательской группы от системы.
* выделение основных функций и ролей пользователей, которые будут взаимодействовать с системой.
* определение интерфейсов взаимодействия между пользователями и системой, включая возможные способы доступа к функциям и данным.
* разработка пользовательских сценариев и использование диаграмм взаимодействия для наглядного представления процессов работы пользователей с системой.

На рисунке 1 представлена диаграмма вариантов использования программного обеспечения для сведений о продаже компьютерных игр

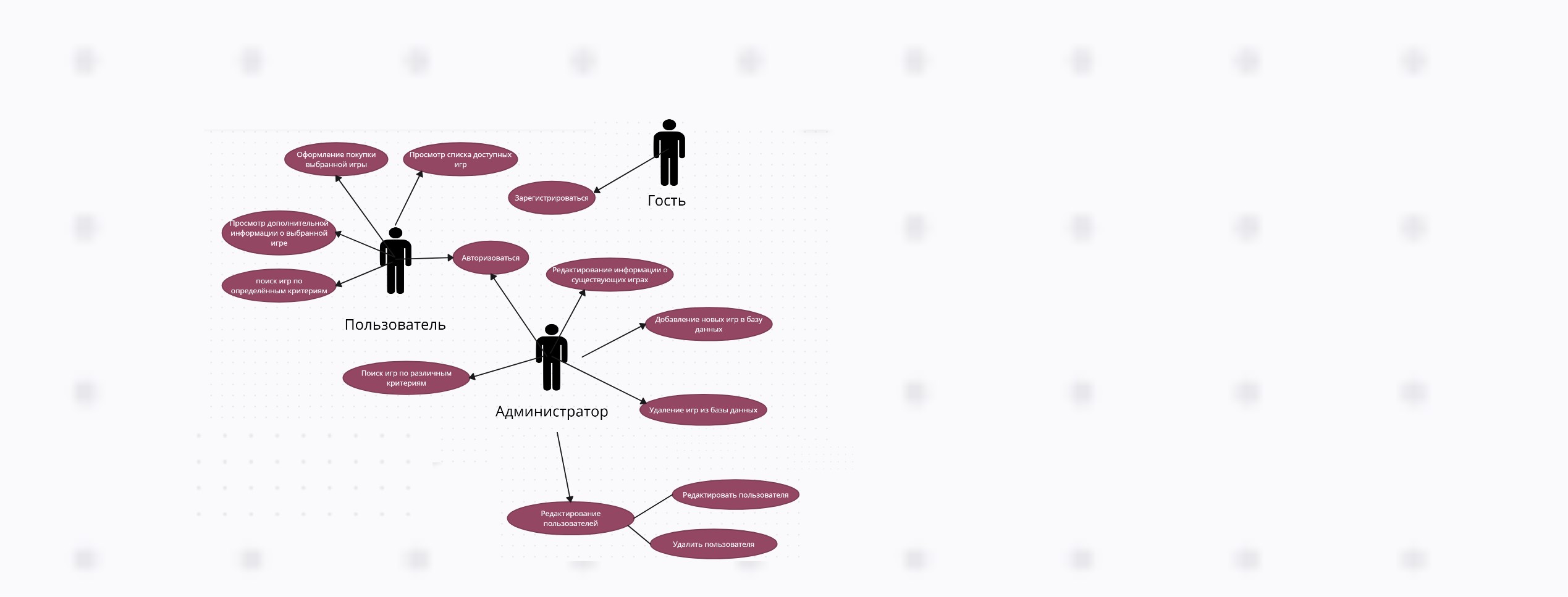


Рисунок 1 - диаграмма вариантов использования

Далее рассмотрим детальное описание прецедентов, использованных при разработке прототипа таблица 2.

Таблица 2 – Описание прецедентов

| № | Наименование | Действие |
| --- | --- | --- |
| 1 | Добавление новой игры | Администратор входит в систему.Выбирает функцию "Добавить новую игру".Вводит информацию о новой игре (название, производитель, цена, системные требования, стиль).Сохраняет данные в базе данных. |
| 2 | **Просмотр списка игр** | Пользователь входит в систему.Переходит на страницу со списком доступных игр.Получает список игр с краткой информацией о каждой. |
| 3 | Поиск игры по критериям | Пользователь входит в систему.Использует функцию поиска игр.Вводит критерии поиска (например, жанр или название).Получает результаты поиска. |
| 4 | Просмотр информации о выбранной игре | Пользователь выбирает игру из списка.Просматривает подробную информацию о выбранной игре (цена, системные требования, стиль игры). |
| 5 | Оформление покупки | Пользователь выбирает игру для покупки.Добавляет игру в корзину.Проходит процесс оформления заказа (вводит данные о доставке и оплате).Подтверждает покупку. |
| 6 | Регистрация нового пользователя | Пользователь открывает приложение " сведений о продаже компьютерных игр ".Пользователь нажимает на кнопку "Регистрация".Система отображает форму регистрации, в которую пользователь вводит свои данные, такие как имя, логин, пароль и.т.д.Пользователь заполняет форму регистрации и нажимает на кнопку "Зарегистрироваться".Система проверяет введенные данные на корректность и наличие уже существующего пользователя с такими же данными.Если данные заполнены, верно, и такого пользователя нет, система создает новую учетную запись для пользователя и перенаправляет его на страницу входа. |
| 7 | Вход в систему | Пользователь открывает приложение " сведения о продаже компьютерных игр ".Пользователь нажимает на кнопку "Войти".Система отображает форму входа, в которую пользователь вводит логин, пароль.Пользователь заполняет форму входа и нажимает на кнопку "Войти".Система проверяет введенные данные на правильность и существование пользователя с такими данными.Если данные введены правильно и пользователь существует, система авторизует пользователя и открывает ему доступ к функционалу системы. |

Диаграммы потоков данных помогают визуализировать потоки данных и процессы, которые они проходят в информационной системе. В случае системы управления информацией о продаже компьютерных игр.

На рисунке 2 изображена диаграмма потоков данных

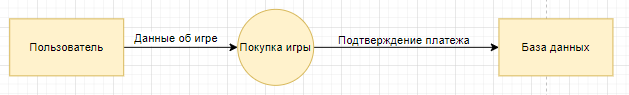


Рисунок 2 - Диаграммы потоков данных

мы можем выделить следующие уровни DFD.

Уровень 0 DFD (Общий уровень):

На уровне 0 DFD отображается общий обзор системы и ее взаимодействие с внешними сущностями.

На уровне 0 имеем следующие элементы:

1. Процесс "Управление информацией о продаже игр":

* Это главный процесс системы, который обрабатывает все действия и операции, связанные с управлением информацией о продаже игр.

2. Внешние сущности:

* Пользователи (Пользователь и Администратор) - сущности, взаимодействующие с системой;
* База данных - хранилище информации о продаже игр.

Уровень 1 DFD (Детализированный уровень).

На уровне 1 DFD можно более детально рассмотреть внутренние процессы и потоки данных в системе.

* Процесс "Управление информацией о продаже игр".

На уровне 1 мы имеем следующие дополнительные элементы:

Процессы:

* Добавление новой игры;
* Просмотр списка игр;
* Поиск игры;
* Просмотр информации о выбранной игре;
* Оформление покупки.

Хранилище данных:

* База данных, где хранится информация о продаже игр.

Диаграммы "сущность—связь" (ER-диаграмма) используются для моделирования структуры базы данных, отображая сущности и их отношения друг с другом.

На рисунке изображена диаграмма "сущность—связь".

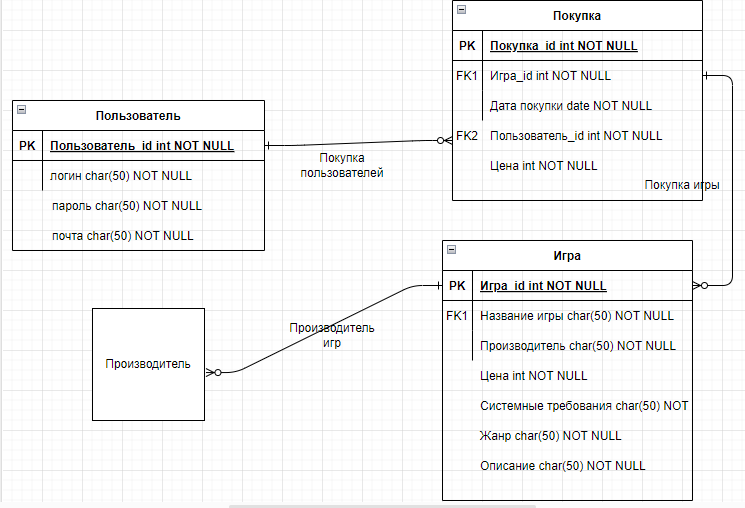


Рисунок 3 - диаграмма "сущность—связь"

Далее рассмотрим детальное описание сущность—связь, использованных при разработке прототипа.

ER-диаграмма

Сущности:

1. Игра:

* Атрибуты: Название, производитель, цена, системные требования, стиль;
* Примечание: Сущность, представляющая информацию о компьютерных играх.

2. Пользователь:

* Атрибуты: Имя, Электронная почта, Пароль;
* Примечание: Сущность, представляющая пользователей системы.

3. Покупка:

* Атрибуты: Идентификатор покупки, Дата покупки, Идентификатор игры, Идентификатор пользователя;
* Примечание: Сущность, представляющая информацию о покупках пользователей.

Связи:

1. Связь «Покупка игры»:

* Тип связи: Многие ко многим;
* Описание: Пользователи могут приобретать несколько игр, и каждая игра может быть приобретена несколькими пользователями.

2. Связь «Игра принадлежит производителю»:

* Тип связи: Один ко многим;
* Описание: Каждая игра принадлежит только одному производителю, но производитель может выпускать несколько игр.

3. Связь «Пользователь совершает покупки»:

* Тип связи: Один ко многим;
* Описание: Каждый пользователь может совершать несколько покупок, но каждая покупка относится только к одному пользователю.

Функциональные диаграммы помогают визуализировать функциональность системы, описывая взаимодействие между различными компонентами и процессами. В данном случае, для программного продукта управления информацией о продаже компьютерных игр, мы можем определить следующие функциональные диаграммы:

Диаграмма прецедентов:

Прецеденты:

1. Добавление новой игры:

* Актеры: Администратор;
* Описание: Администратор может добавлять новые игры в систему.

2. Просмотр списка игр (View Game List):

* Актеры: Пользователь, Администратор;
* Описание: Пользователь или администратор могут просматривать список доступных игр.

3. Поиск игры:

* Актеры: Пользователь, Администратор;
* Описание: Пользователь или администратор могут выполнять поиск игр по различным критериям.

4. Просмотр информации о выбранной игре:

* Актеры: Пользователь, Администратор;
* Описание: Пользователь или администратор могут просматривать подробную информацию о выбранной игре.

5. Оформление покупки:

* Актеры: Пользователь;
* Описание: Пользователь может оформлять покупку выбранных игр.

Диаграмма последовательности:

1. Просмотр списка игр:

* Пользователь отправляет запрос на просмотр списка игр;
* Система отображает список игр.

2. Поиск игры:

* Пользователь отправляет запрос на поиск игры;
* Система выполняет поиск и отображает результаты.

3. Просмотр информации о выбранной игре:

* Пользователь выбирает игру для просмотра информации;
* Система отображает подробную информацию о выбранной игре.

4. Оформление покупки:

* Пользователь добавляет выбранную игру в корзину;
* Пользователь оформляет покупку, вводя необходимые данные;
* Система обрабатывает заказ и регистрирует покупку.

На рисунке 4 изображена диаграмма последовательности пользователя

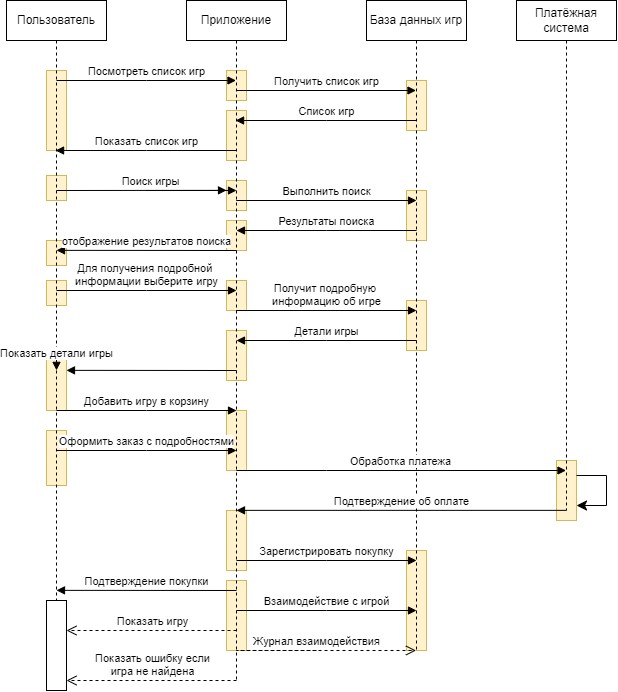


Рисунок 4 - диаграмма последовательности пользователя

Диаграммы переходов и состояний (State Transition Diagrams) помогают описать поведение объекта или системы в различных состояниях и переходах между этими состояниями. В случае программного продукта для управления информацией о продаже компьютерных игр, мы можем определить следующую диаграмму переходов состояний для процесса заказа игр.

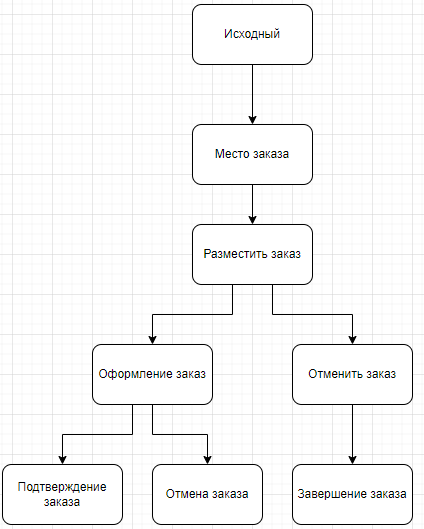


Рисунок 5 - диаграмма переходов и состояний

Состояния:

Начальное состояние:

* Начальное состояние процесса заказа игры.

Оформление заказа:

* Пользователь находится в процессе оформления заказа игры.

Подтверждение заказа:

* Заказ игры подтвержден системой.

Отмена заказа:

* Заказ игры отменен пользователем или системой.

Переходы:

Оформление заказа:

* Пользователь начинает процесс оформления заказа.

Подтверждение заказа:

* Пользователь подтверждает заказ игры.

Отмена заказа:

* Пользователь отменяет заказ игры.

Завершение заказа:

* Заказ игры успешно завершен.

Спецификации процессов описывают шаги и действия, необходимые для выполнения определенного процесса или функции в системе. Для программного продукта управления информацией о продаже компьютерных игр мы можем определить следующие спецификации процессов:

Процесс оформления заказа игры:

Цель: Позволяет пользователю выбрать и оформить заказ на приобретение игры.

Шаги:

* Пользователь просматривает список доступных игр.
* Пользователь выбирает игру для покупки.
* Пользователь добавляет выбранную игру в корзину.
* Пользователь переходит к оформлению заказа.
* Пользователь вводит необходимые данные для доставки и оплаты.
* Пользователь подтверждает заказ.

Процесс обработки заказа игры:

Цель: Обрабатывает заказ пользователя и регистрирует его в системе.

Шаги:

* Система получает данные о заказе от пользователя.
* Система проверяет наличие выбранной игры и ее доступность.
* Система создает запись о заказе в базе данных.
* Система отправляет подтверждение заказа пользователю.

Процесс отмены заказа игры:

Цель: Позволяет пользователю отменить уже оформленный заказ.

Шаги:

* Пользователь переходит в свой личный кабинет или раздел управления заказами.
* Пользователь выбирает заказ, который хочет отменить.
* Пользователь нажимает кнопку "Отменить заказ".
* Система подтверждает отмену заказа и удаляет его из базы данных.

Словарь терминов, связанных с программным обеспечением для управления информацией о продаже компьютерных игр:

* Игра (Game): Вид развлекательного программного обеспечения, предназначенный для игры пользователя.
* Фирма-производитель (Game Developer): Компания, занимающаяся разработкой и выпуском компьютерных игр.
* Цена (Price): Денежная стоимость игры, выставленная для продажи.
* Системные требования (System Requirements): Минимальные технические характеристики компьютера, необходимые для запуска игры.
* Стиль игры (Game Genre): Жанр или категория, к которой относится игра, например, экшн, стратегия, RPG и т. д.
* Пользователь (User): Человек, который использует программное обеспечение для просмотра, выбора и приобретения игр.
* Заказ (Order): Запрос пользователя на приобретение определенной игры по определенной цене.
* Корзина (Shopping Cart): Временное хранилище, в котором пользователь может добавлять выбранные игры перед оформлением покупки.
* Подтверждение заказа (Order Confirmation): Сообщение или уведомление, подтверждающее успешное оформление заказа.
* Отмена заказа (Order Cancellation): Процесс аннулирования уже оформленного заказа до его исполнения.
* База данных (Database): Организованное хранилище информации о играх, пользователях, заказах и других сущностях, используемое для работы системы управления продажами.
* Личный кабинет (User Account): Частная область на сайте или в программе, где пользователь может просматривать и управлять своими данными, заказами и другой персональной информацией.

Вывод по разделу

В результате анализа предметной области и выработки требований к программному обеспечению для управления продажами компьютерных игр, было разработано техническое задание в соответствии с ГОСТом. Проектирование программного обеспечения включило в себя определение актеров и вариантов использования системы, а также создание различных диаграмм (прецедентов, последовательности, потоков данных, сущность-связь, функциональных и диаграмм переходов состояний). В целом, данные шаги позволили определить ключевые аспекты и требования к разрабатываемому программному продукту, что является важным этапом перед переходом к фазе разработки и реализации.

# СРЕДСТВА РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Средства разработки программного обеспечения (ПО) — это инструменты и технологии, которые помогают создавать, тестировать, отлаживать и поддерживать программное обеспечение.

Интегрированные среды разработки (IDE): объединяют в себе различные инструменты разработки, такие как редактор кода, компилятор, отладчик и средства управления версиями, в единый интерфейс. Пример: Visual Studio.

Таблица 3. Описание функций модулей ПО

|  |  |  |
| --- | --- | --- |
| № | Наименование модуля | Функции модуля |
| 1 | Глобальный модуль | Содержит глобальные процедуры и функции, предопределенные процедуры, процедуры и функции, которые необходимо выполнить при запуске системы |
| 2 | Модуль управления данными | Хранение и управление информацией об играх (название, платформа, жанр, разработчик, дата выпуска, цена, количество проданных копий, доход от продаж) |
| 3 | Модуль статистики продаж | Расчет общего дохода от продаж всех игрАнализ наиболее популярных жанров игрОтображение статистики по количеству проданных копий игр |
| 4 | Модуль отчетности | Генерация отчетов о доходе от продаж за определенный период времени |

## Программная реализация приложения

Программная реализация приложения — это процесс преобразования исходного кода приложения в исполняемый код, который может быть запущен на целевой платформе.

Исходные данные для ИС "Продажа игр":

* название игр;
* платформа (например, ПК, консоль, мобильное устройство);
* жанр игры;
* разработчик игры;
* дата выпуска;
* цена игры;
* количество проданных копий;
* доход от продаж.

Первичные документы:

* счета-фактуры;
* накладные;
* акты приема-передачи.

Обработка данных:

* расчет итоговых сумм;
* сортировка и фильтрация данных по различным критериям (например, по платформе, жанру, разработчику и т.д.);

Выходная информация и печатные документы:

* отчеты о продажах игр (в разбивке по различным критериям)
* печатные счета-фактуры, накладные и акты приема-передачи

Направление вывода отчетов (данных):

* отчеты могут выводиться на экран или экспортироваться в различные форматы (например, PDF, Excel, CSV).
* печатные документы распечатываются на принтере.

### Проектирование интерфейса приложения

Проектирование интерфейса приложения — это процесс создания пользовательского интерфейса (UI) для мобильного приложения или веб-сайта. Цель проектирования интерфейса — создать интуитивно понятный и удобный интерфейс, который позволит пользователям легко взаимодействовать с приложением.

Макет — модель объекта в уменьшенном масштабе или в натуральную величину, лишённая, как правило, функциональности представляемого объекта. Макет — модель, предварительный образец, пробный образец чего-либо. Предназначен для представления объекта; используется в тех случаях, когда представление оригинального объекта неоправданно дорого, невозможно или просто нецелесообразно.

Макетирование — процесс создания макета. Прозрачный макет — показывает внутреннюю конструкцию, принцип действия прототипа. Макет приложения был создан с помощью программы Figma. На рисунке 6 показан макет приложения.

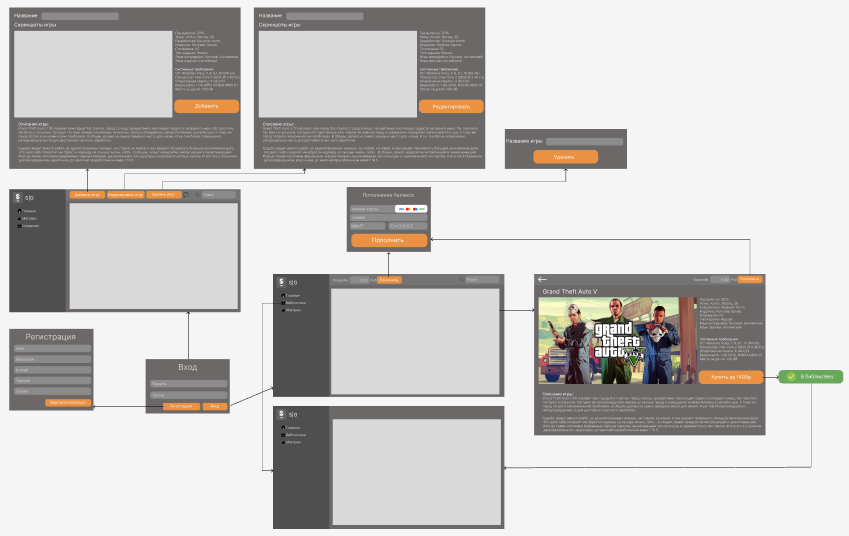


Рисунок 6 – макет приложения

Реализация интерфейса приложения представлено в ПРИЛОЖЕНИИ Б

### Программирование и интерпретация результатов

Программирование включает в себя написание кода, который определяет поведение программы.

Этот код обычно пишется на языке программирования высокого уровня, который затем компилируется или интерпретируется в машинный код, который может выполняться компьютером.

Интерпретация результатов — это процесс анализа вывода программы и определения его значения.

Это может включать проверку правильности результатов, выявление закономерностей и выводов на их основе.

На рисунке 7 представлена форма «администратор»

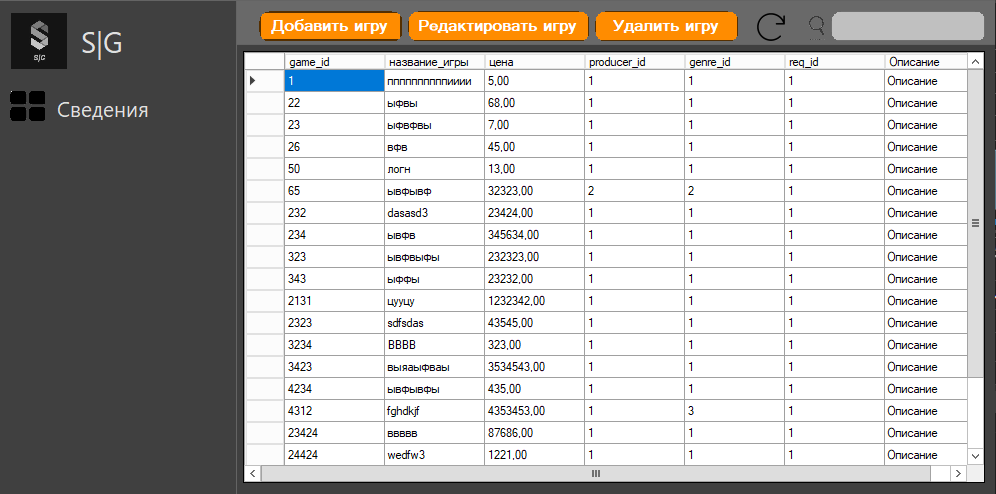


Рисунок 7 – прототип главной формы «Администратор»

Свойства элементов формы «Администратор» представлены в таблице 4.

Таблица 4 – Свойства элементов формы «Администратор»

|  |  |  |
| --- | --- | --- |
| Объект | Свойство | Значение |
| Администратор | Name | Администратор |
| BackColor | 64; 0; 0 |
| FormBorderStyle | None |
| Size | 740; 410 |
| Label1 | Font | Times New Roman; 24pt; style=Bold, Italic |
| Text | Администратор |
| ForeColor | White |
| Label2 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | White |
| Text | Сведения |
| Label3 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | silver |
| Text | S|G |
| Button1 | Text | Добавить игру |
| Location | 12; 94 |
| Button2 | Location | 12; 164 |
| Text | Редактировать игру |
| Button3 | Text | Удалить игру |
| Location | 12; 94 |
| DataGridView1 | Location | 12; 164 |
| data | Игры |

Код формы «Администратор» представлены в таблице 5.

Таблица 5 – Код формы «Администратор»

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Data.SqlClient;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  using static System.Windows.Forms.VisualStyles.VisualStyleElement;  namespace SG\_Продажа\_игр  {  public partial class Админестратор : Form  {  private int gameId;  public Админестратор()  {  InitializeComponent();  }  private void Админестратор\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "коновалов\_СВ\_УП\_02DataSet.Игры". При необходимости она может быть перемещена или удалена.  this.игрыTableAdapter.Fill(this.коновалов\_СВ\_УП\_02DataSet.Игры);  raundclass.ApplyRoundedCorners(textBox2, 10);  raundclass.ApplyRoundedCorners(button1, 10);  raundclass.ApplyRoundedCorners(button2, 10);  raundclass.ApplyRoundedCorners(button3, 10);  }  private void button1\_Click(object sender, EventArgs e)  {  this.Hide();  Добавить\_игру addGameForm = new Добавить\_игру(); // Передаем ссылку на текущий экземпляр "Администратора"  addGameForm.Show();  }  private void button2\_Click(object sender, EventArgs e)  {  this.Hide();  редактировать\_игру frm2 = new редактировать\_игру(gameId); // Передача gameId в качестве параметра конструктора  frm2.Show();  }  private void button3\_Click(object sender, EventArgs e)  {  удалить frm2 = new удалить();  frm2.Show();  }    private void textBox2\_TextChanged(object sender, EventArgs e)  {  SearchGames(textBox2.Text);  }  private void SearchGames(string gameName)  {  try  {  // Подключаемся к базе данных  string connectionString = @"Data Source=sql;Initial Catalog=Коновалов\_СВ\_УП\_02;Integrated Security=True";  using (SqlConnection connection = new SqlConnection(connectionString))  {  // Создаем SQL-запрос для поиска игр по названию  string query = "SELECT \* FROM Игры WHERE [название\_игры] LIKE @gameName";  // Создаем команду с параметрами  using (SqlCommand command = new SqlCommand(query, connection))  {  // Добавляем параметр для названия игры  command.Parameters.AddWithValue("@gameName", "%" + gameName + "%");  // Открываем соединение и выполняем команду  connection.Open();  SqlDataAdapter adapter = new SqlDataAdapter(command);  DataTable dataTable = new DataTable();  adapter.Fill(dataTable);  // Загружаем результаты поиска в DataGridView или другой контрол для отображения таблицы игр  dataGridView1.DataSource = dataTable;  }  }  }  catch (Exception ex)  {  MessageBox.Show("Ошибка при поиске игр: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  }  SqlConnection con = new SqlConnection(@"Data Source=sql;Initial Catalog=Коновалов\_СВ\_УП\_02;Integrated Security=true");  private void pictureBox1\_Click(object sender, EventArgs e)  {  try  {  string commandText = "select \* from Игры";  SqlDataAdapter itm = new SqlDataAdapter(commandText, con);  DataTable dt2 = new DataTable();  itm.Fill(dt2);  dataGridView1.DataSource = dt2;  con.Close();  }  catch  {  MessageBox.Show("Нечего обновлять");  }  }  private void label4\_Click(object sender, EventArgs e)  {  this.Close();  Сведения frm2 = new Сведения();  frm2.Show();  }  private void pictureBox2\_Click(object sender, EventArgs e)  {  this.Close();  Админестратор frm2 = new Админестратор();  frm2.Show();  }  private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)  {  }  }  } |

Листинги программного кода других формы размещён в ПРИЛОЖЕНИИ В

### Реализация алгоритма интеграции данных

Демонстрация процесса подключения SQL Server к Visual Studio и настройки соединения с базой данных представлены на рисунках №-№.

Открываем источник данных и нажимаем добавить БД, представлено на рисунке 8.



Рисунок 8 – Добавление источника данных

Далее нажимаем на «база данных», представлено на рисунке 9

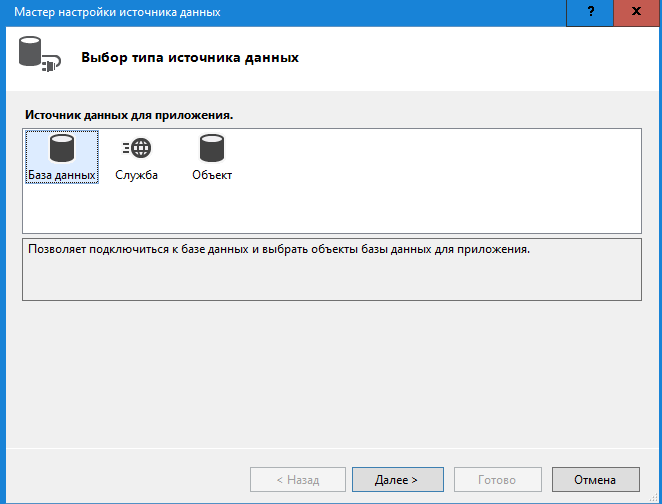


Рисунок 9 –источник данных

Далее нажимаем на «Создать подключение», представлено на рисунке 10

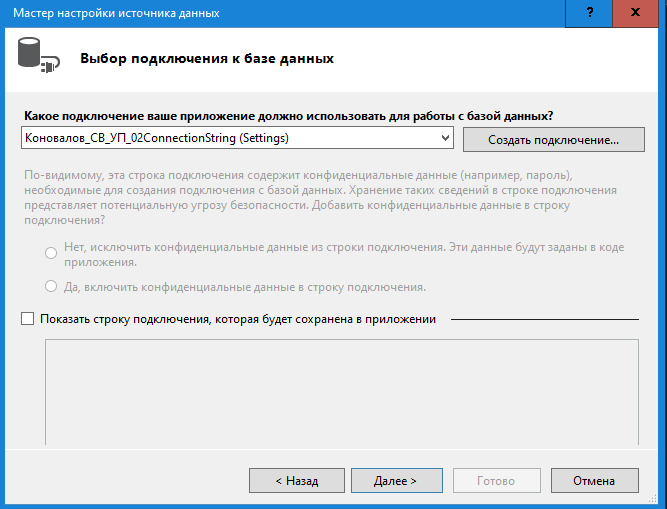


Рисунок 10 – Создать подключение

Далее нажимаем на «Другое», выбираем поставщика данных SQL,нажимаем продолжить, представлено на рисунке 11

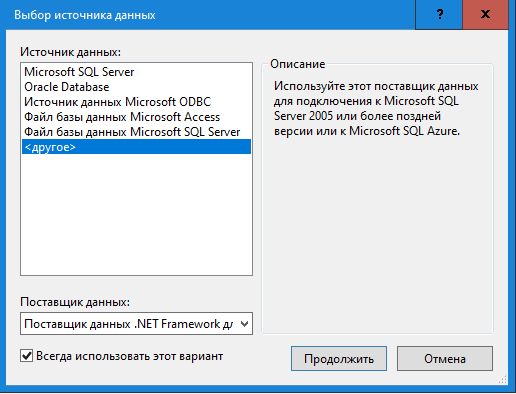


Рисунок 11 – выбор поставщика данных

Далее нажимаем на «Обновить», выбираем Базу данных SQL,нажимаем «ОК», представлено на рисунке 12

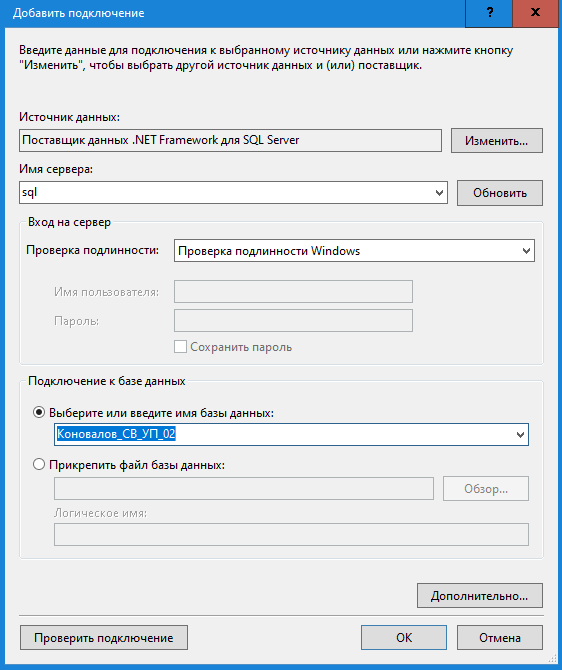


Рисунок 12 – Добавление базы данных

Далее выбираем нашу базу, нажимаем далее, отмечаем галочкой «таблицы» , «готово», представлено на рисунке 13

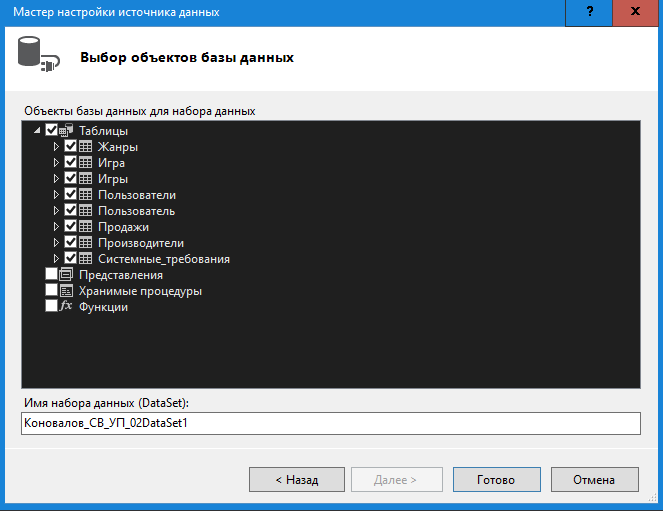


Рисунок 13 – Выбор таблиц

В результате в источниках данных появляется база данных SQL, рисунок 14.

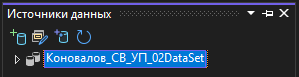


Рисунок 14 – Добавленная БД

## Отладка приложения

Отладка приложения — важный этап разработки программного обеспечения, направленный на обнаружение, анализ и устранение ошибок. В данном разделе описывается процесс отладки, используемые отладочные средства, классификация ошибок, а также приводится контрольный пример, на котором проводился процесс отладки.

На рисунке 15 представлена процесс откладки приложения

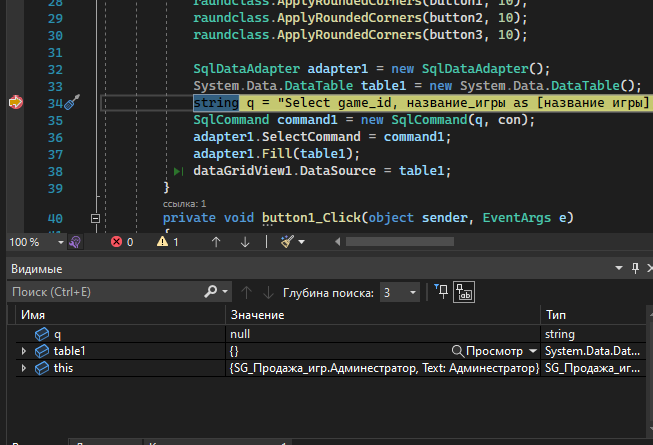


Рисунок 15 – процесс откладки приложения

## Тестирование приложения

### Выбор и обоснование методики проведения тестирования

При выборе методики тестирования для приложения, следует учитывать несколько основных подходов:

Модульное тестирование: Этот метод направлен на проверку отдельных модулей или компонентов приложения, позволяя выявить ошибки внутри отдельных блоков кода и обеспечивая легкость в рефакторинге и поддержке кода.

Интеграционное тестирование: Здесь проверяется взаимодействие между различными модулями или компонентами приложения, что помогает выявить ошибки в интеграции между разными частями приложения.

Системное тестирование: Этот вид тестирования проверяет приложение в целом, когда все компоненты объединены, для соответствия требованиям и выявления ошибок, связанных с взаимодействием компонентов.

Тестирование безопасности: Направлено на обнаружение уязвимостей в приложении, обеспечивая защиту данных от потенциальных атак и обеспечивая конфиденциальность и целостность информации.

Рекомендуется использовать комплексный подход, включающий все эти методики тестирования для приложения. Это поможет обеспечить надежность, безопасность, эффективность и соответствие требованиям пользователей. Каждая методика тестирования важна для обеспечения качества приложения и выявления различных видов ошибок и проблем.

### Схема выполнения экспериментального тестирования

Схема выполнения экспериментального тестирования — это детально разработанный план или методика, которая применяется для проверки и оценки качества, эффективности, безопасности и других характеристик продукта, системы или компонента. В контексте разработки программного обеспечения, это включает в себя серию действий, направленных на идентификацию и устранение ошибок, проверку соответствия функциональных и нефункциональных требований, а также оценку пользовательского опыта и производительности приложения.

1.Определение целей тестирования:

* Определение основных целей и задач тестирования.
* Определение критериев успеха.

2. Планирование тестирования:

* Выбор методов тестирования (например, функциональное, нагрузочное, юзабилити).
* Определение тестовых случаев и сценариев.

3. Подготовка к тестированию:

* Установка необходимых инструментов и средств для тестирования.
* Подготовка тестовых данных.
* Подготовка среды для проведения тестирования.

4. Выполнение тестирования:

* Запуск тестовых сценариев и случаев.
* Регистрация результатов тестирования.
* Оценка производительности, надежности, безопасности и других характеристик.

5. Анализ результатов:

* Оценка соответствия продукта требованиям.
* Идентификация и устранение выявленных ошибок.
* Формирование отчета о проведенном тестировании.

6. Повторное тестирование (при необходимости):

* Повторное выполнение тестов для проверки исправлений и улучшений.
* Повторная оценка результатов и подготовка отчета.

7. Завершение тестирования:

* Подготовка итогового отчета о проведенном тестировании.
* Оценка эффективности и полноты проведенного тестирования.

Проведение нагрузочного тестирования

На рисунке 16 представлен путь к профилировщику производительности

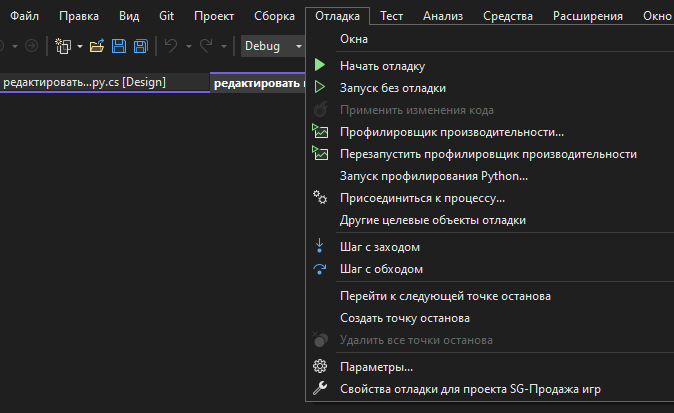


Рисунок 16 – Выбор профилировщика производительности

Далее на рисунке 17 представлен выбор инструментов для тестирования

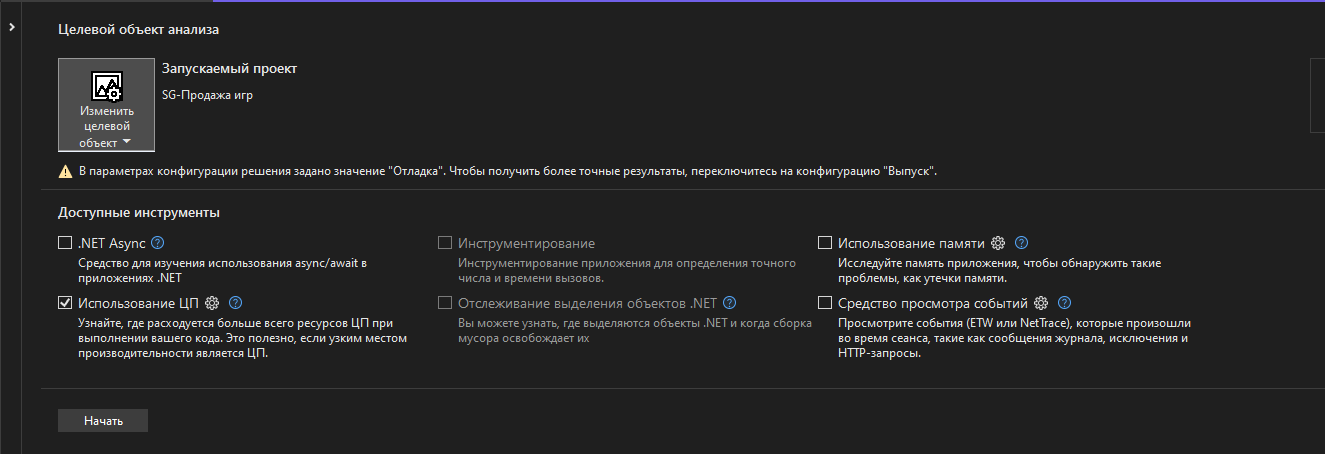


Рисунок 17 – Выбор инструментов

### Документирование результатов тестирования

Для проверки нашего приложения мы можем использовать различные методы тестирования, такие как юзабилити, нагрузочное тестирование.

Юзабилити тестирование

Прежде чем начинать юзабилити-тестирование необходимо собрать группу пользователей, которые будут тестировать данную информационную систему. Количество привлеченных пользователей должно быть не менее пяти человек. Пользователям будет представлен список некоторых вопросов, относящихся к графическому интерфейсу.

Таблица 6– Информация о пользователях

| ФИО | Пол | Возраст | Образование |
| --- | --- | --- | --- |
| Леснеев Марк Сергеевич | Мужской | 20 | Основное общее |
| Якименко Ян Борисович | Мужской | 20 | Полное общее |
| Савин Марк Леонидович | Мужской | 19 | Основное общее |
| Березин Даниил Александрович | Мужской | 19 | Среднее специальное |
| Коновалов Сергей Владимирович | Мужской | 20 | Среднее специальное |

Результаты проведения юзабилити-тестирования рисунок 16

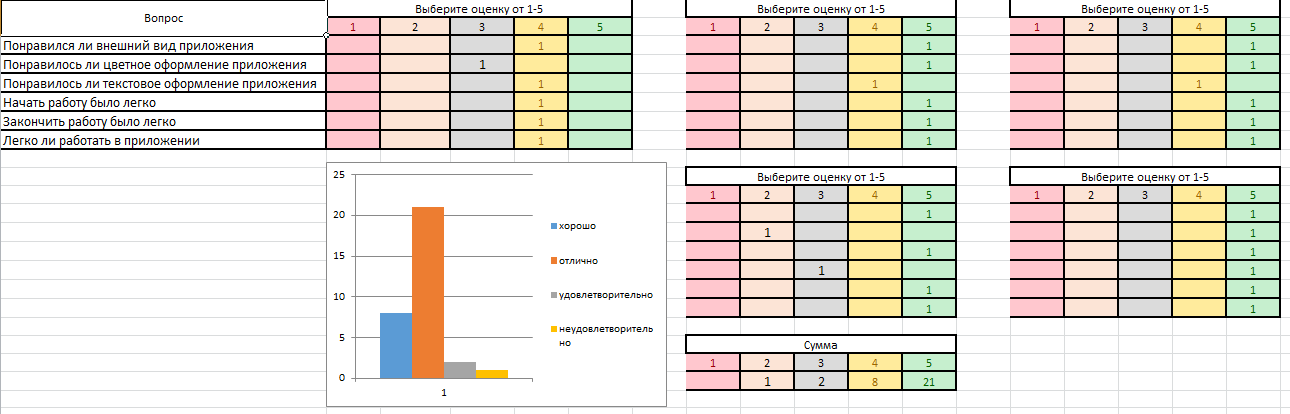


Рисунок 16 – Результат юзабилити-тестирования

Исходя из средней оценки информационной системы, рассчитанной на основе общих оценок всех пользователей с учетом ответов на все поставленные вопросы, можно сделать вывод о том, что внешние характеристики графического интерфейса находятся на довольно высоком уровне. Также в информационной системе нет помех для пользователя в виде отвлекающей информации, так как все отображаемые данные находятся в рамках предметной области.

Проведение нагрузочного тестирования

Для проведения нагрузочного тестирования составлена таблица 10, в которой указано количество запущенных копий программного продукта, нагрузка на процесс и оперативную память.

На рисунке 17-18 изображены тесты нагрузочного тестирования

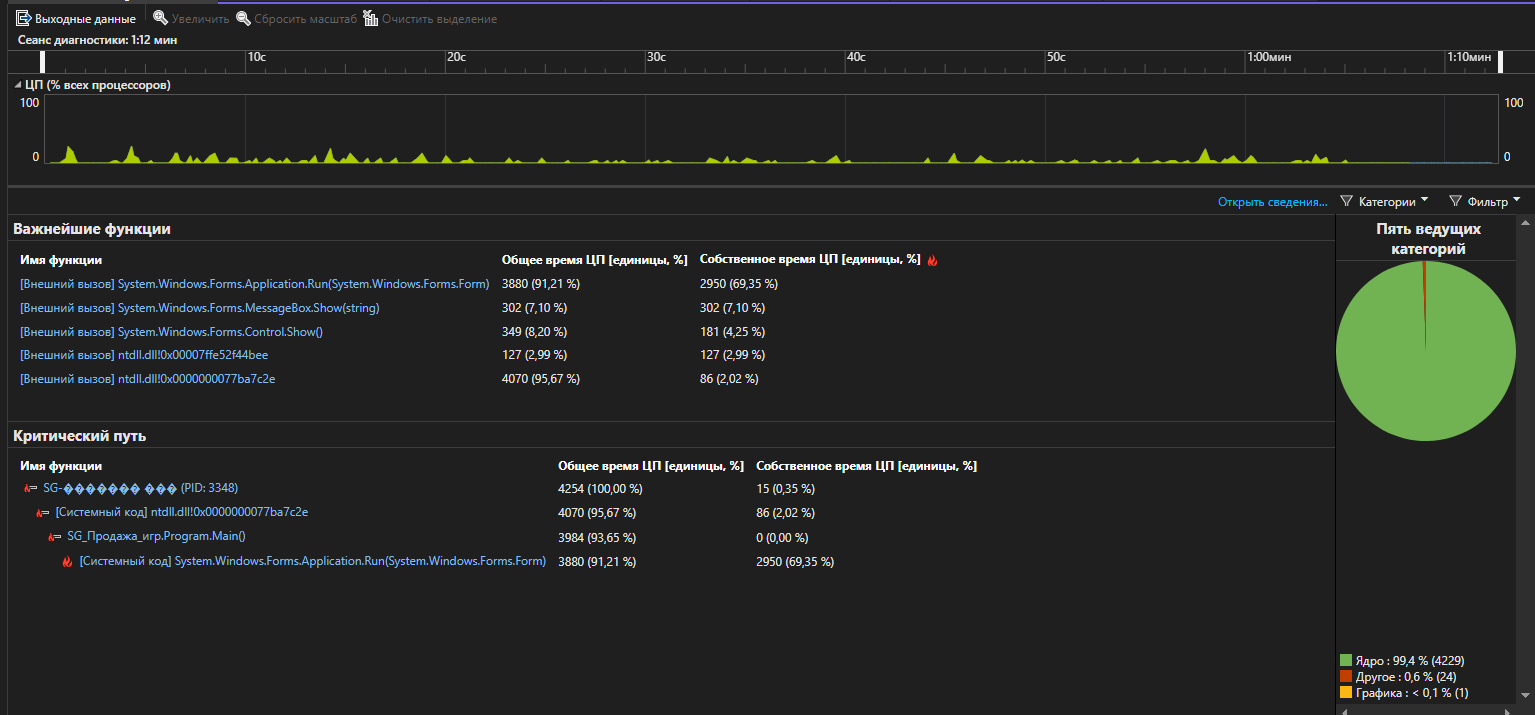


Рисунок 17 – тест Процессора

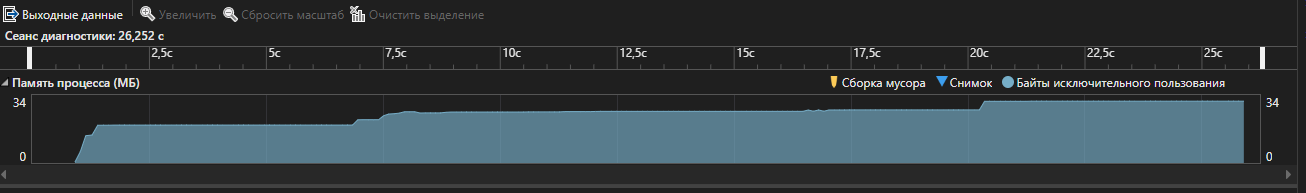


Рисунок 18 – тест Оперативной памяти

Таблица 7 – Результаты нагрузочного тестирования

|  |  |  |
| --- | --- | --- |
| Число копий | Нагрузка на процессор(%) | Нагрузка на память(%) |
| 1 | 0,1 | 0,2 |
| 5 | 0,2 | 2 |
| 10 | 2,5 | 3 |
| 15 | 3,3 | 4 |
| 25 | 4,6 | 4 |

Исходя из результатов тестирования, можно прийти к выводу, что нагрузка, оказываемая на центральный процессор и память, не имеет критичных значений. Это значит, что показатель производительности информационной системы на приемлемом уровне, то есть сбои и длительная загрузка не должны помешать пользователю при работе с данным программным продуктом.

В ходе тестирования приложения, созданного в Visual Studio, были использованы различные методики и подходы, включая функциональное тестирование, проверку пользовательского интерфейса, адаптивность к различным устройствам, совместимость с различными версиями Windows, а также тестирование производительности и безопасности.

Был разработан ряд тестовых сценариев, нацеленных на проверку правильной работы основных функций приложения, его отзывчивости и удобства использования.

Интерфейс: Интерфейс приложения оценен как интуитивно понятный и удобный. Однако выявлены проблемы с отображением некоторых элементов на мобильных устройствах, требующие исправления.

Совместимость: Приложение успешно работает на последних версиях операционной системы Windows, но имеет проблемы совместимости с устаревшими версиями Windows, что требует доработки для поддержки более широкого диапазона версий.

Производительность: Результаты тестирования производительности показали, что время отклика приложения находится в приемлемых пределах. Рекомендуется оптимизация изображений для ускорения загрузки приложения.

Исходя из проведенного тестирования, можно сделать вывод, что приложение в целом соответствует заданным требованиям, за исключением некоторых обнаруженных недочетов. Наиболее значимыми из них являются проблемы с формой обратной связи, совместимостью с устаревшими версиями операционной системы Windows и уязвимости безопасности, которые требуют внимания и исправлений.

Выводы по разделу.

После тщательного анализа требований к разработке приложения и определения ключевых целей был разработан подробный план действий, включающий проектирование интерфейса, программирование функциональности и интеграцию данных. Процесс реализации проекта включал использование современных технологий и методологий разработки, что способствовало созданию гибкого и масштабируемого приложения. Особое внимание уделялось отладке и тестированию приложения, что обеспечило высокое качество и надежность разработанного программного обеспечения. Результаты тестирования подтвердили соответствие приложения всем заранее определенным требованиям, что позволяет уверенно заявить о готовности проекта к использованию.

# МОДЕЛИРОВАНИЕ В ПРОГРАММНЫХ СИСТЕМАХ

Моделирование в программных системах является важным инструментом для анализа, проектирования и оптимизации различных аспектов системы. Методы математического моделирования играют ключевую роль в этом процессе, позволяя абстрагировать сложные системы, описать их поведение с помощью математических уравнений и алгоритмов, а затем анализировать эти модели для принятия обоснованных решений.

## Нахождение кратчайших путей на графе.

Нахождение кратчайших путей в графе - это задача поиска наиболее эффективных маршрутов между двумя или более вершинами графа.

Условия задачи: Определите длину кратчайшего пути между пунктами A и Z. Постройте данный граф, найдите матрицу смежности и матрицу инцидентности, для полученного графа.

Исходные данные для построения графа представлены в весовой матрице на рисунке 19

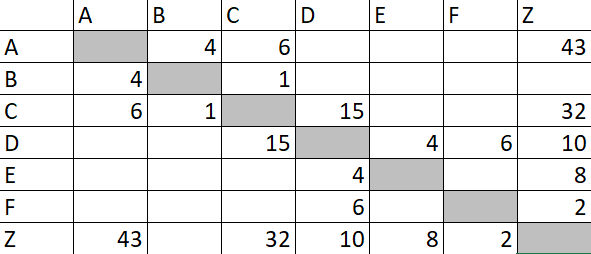


Рисунок 19 - Исходные данные

Построенный граф представлен на рисунке 20

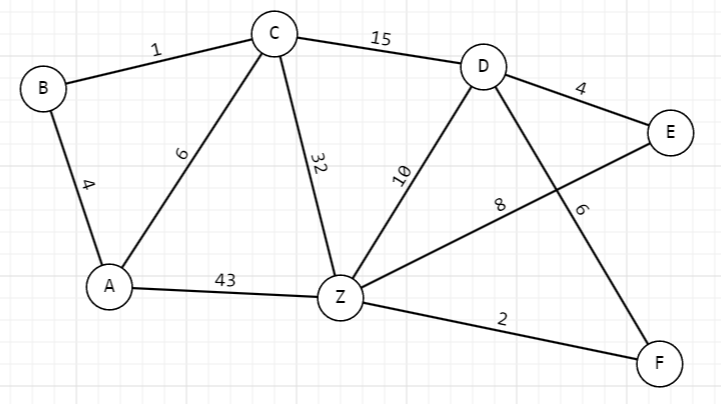


Рисунок 20 - Построенный граф

Нахождение кратчайшего пути между пунктами A и Z.

Для нахождения кратчайшего пути необходимо определить все возможные пути из пункта A в пункт Z. Таких путей всего 8. Далее необходимо рассчитать расстояние каждого пути и сравнить конечный результат.

Возможные пути и расчет его расстояния:

1. AZ = 43.
2. ABCZ = 4 + 5 + 32 = 41
3. ABCDZ = 4 + 5 + 15 + 10 = 34
4. ABCDEZ = 4 + 5 + 15 + 4 + 8 = 36
5. ABCDFZ = 4 + 5 + 15 + 6 + 2 = 32
6. ACDZ = 6 + 15 + 10 = 31
7. ACDEZ = 6 + 15 + 4 + 8 = 33
8. ACDFZ = 6 + 15 + 6 + 2 = 29

По результат расчета кратчайшего пути из всех возможных было выявлено, что самым коротким путем из пункта A в пункт Z является путь ACDFZ и его расстояние равно 29.

Для построенного графа были построены матрица смежности и инцидентности.

Матрица смежности графа - это таблица, в которой указываются связи между вершинами графа. Элемент матрицы a[i, j] равен количеству ребер, соединяющих вершины i и j. Если ребра между вершинами нет, то элемент равен 0. Матрица смежности графа представлена на рисунке 21

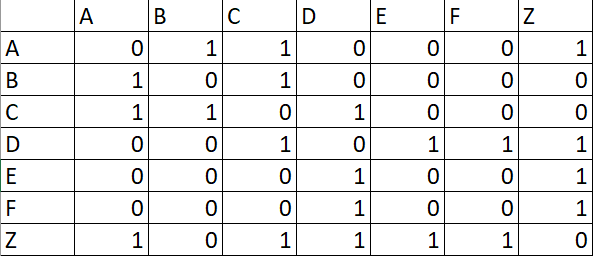
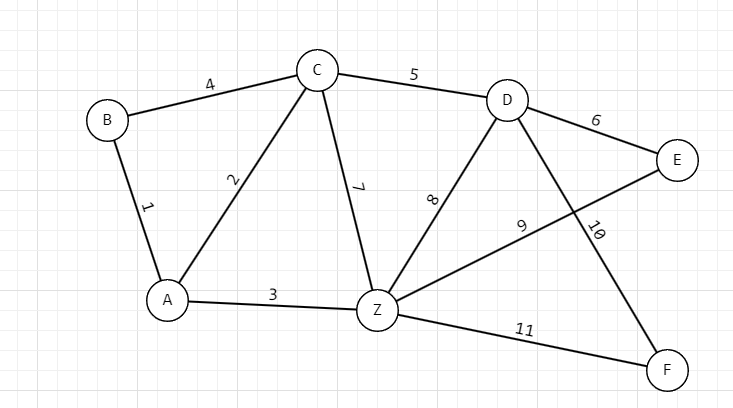


Рисунок 21 - Матрица смежности графа

Матрица инцидентности имеет вид представления графа в виде матрицы, в которой каждый столбец задаёт отдельную дугу. Строки матрицы при этом задают вершины. Положительное число в столбце задаёт вершину, из которой выходит дуга, а отрицательное - в которую входит. Если оба числа положительные, то дуга неориентированная. Матрица инцидентности представлена на рисунке 22



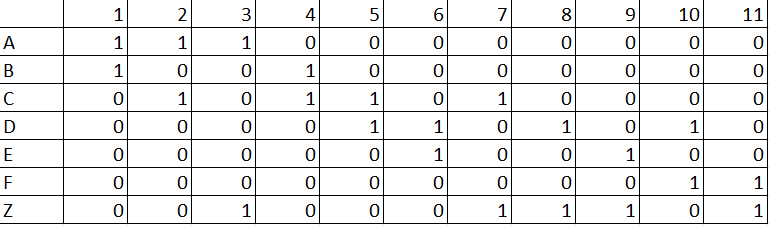


Рисунок 22 - Матрица инцидентности

## Решение задачи об оптимальных ресурсах

Особенностью задач с учетом стоимости ресурсов является тот факт, что бывают случаи, когда в разных точках оптимального для целевой функции отрезка экономятся разные ресурсы. Чтобы выбрать оптимальный вариант плана производства, нужно учитывать стоимость сберегаемых ресурсов.

Исходные данные для расчета представлены в таблице.

|  |  |  |
| --- | --- | --- |
| Целевая функция | Ограничения | Стоимость ресурсов |
| Z=1,8х1+х2 | х1≥ 0, х2≥ 0  5х1+11х2≤ 99  4х1+х2≤ 34  9х1+5х2≤ 82 | c1 = 2,5  c2 = 24  c3 = 8,7 |

Необходимо найти максимальное значение целевой функции F = 1,8х1+х2→ max, при системе ограничений:

5х1+11х2≤ 99,

4х1+х2≤ 34,

9х1+5х2≤ 82,

x1 ≥ 0,

x2 ≥ 0,

где x1, x2 - целые числа.

Решение задачи графическим методом:

Шаг 1. Построение области допустимых значений, то есть решение графически системы неравенства. Для этого необходимо построить каждую прямую и определить полуплоскости, заданные неравенствами. График области допустимых значений представлен на рисунке 23

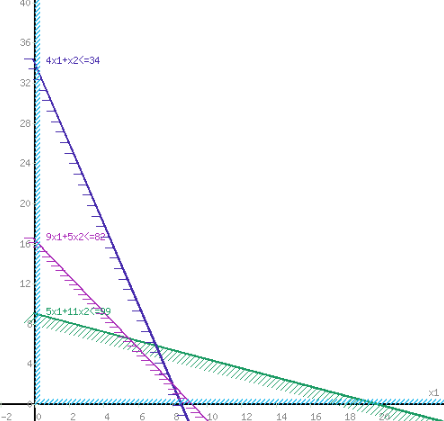


Рисунок 23 - Построенная область допустимых значений

Шаг 2. Границы области допустимых решений.

Пересечением полуплоскостей будет являться область, координаты точек которого удовлетворяют условию неравенствам системы ограничений задачи. Обозначить границы области многоугольника решений. Границы области представлены на рисунке 24

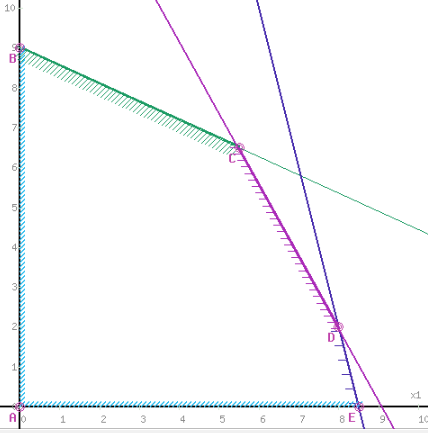


Рисунок 24 - Границы области допустимых решений

Шаг №3. Рассмотрим целевую функцию задачи F = 1.8x1+x2 → max. Построим прямую, отвечающую значению функции F = 1.8x1+x2 = 0. Вектор-градиент, составленный из коэффициентов целевой функции, указывает направление максимизации F(X). Начало вектора – точка (0; 0), конец – точка (1.8;1). Будем двигать эту прямую параллельным образом. Поскольку нас интересует максимальное решение, поэтому двигаем прямую до последнего касания обозначенной области. На графике эта прямая обозначена пунктирной линией.. График представлен на рисунке 25

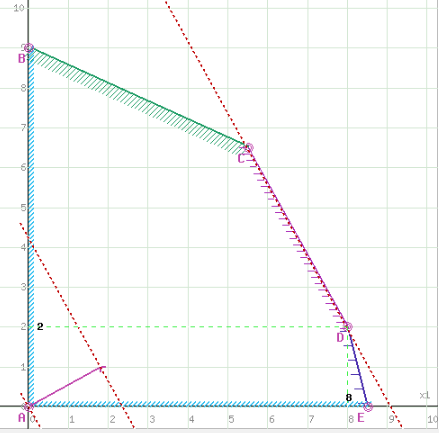


Рисунок 25 - Целевая функция задачи

Прямая F(x) = const пересекает область в точке D. Так как точка D получена в результате пересечения прямых (2) и (3), то ее координаты удовлетворяют уравнениям этих прямых:

4x1+x2=34

9x1+5x2=82

Решив систему уравнений, получим: x1 = 8, x2 = 2 Откуда найдем максимальное значение целевой функции: F(x) = 1.8\*8 + 1\*2 = 16.4 Поскольку функция цели F(x) параллельна прямой (3), то на отрезке DC функция F(x) будет принимает одно и тоже максимальное значение. Для определения координат точки C решим систему двух линейных уравнений:

5x1+11x2=99

9x1+5x2=82

Решив систему уравнений, получим: x1 = 8, x2 = 2 Откуда найдем максимальное значение целевой функции: F(x) = 1.8\*5.5 + 1\*6.5 = 16.4

Решение получилось не целочисленным.

Множество допустимых решений задачи с отмеченными на нем целочисленными точками представлено на рисунке 26

Перемещение линии уровня целевой функции F(X) в направлении, задаваемом ее градиентом, показывает, что наибольшее значение F(X)= 16 она примет в точке (2, 8).

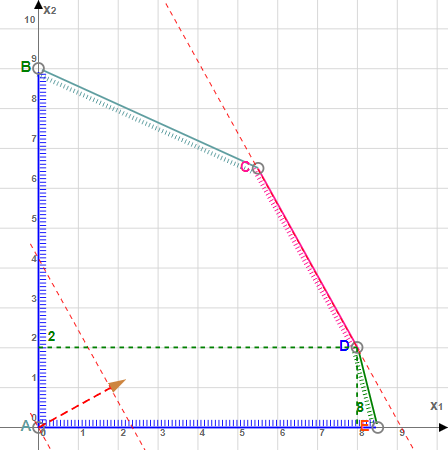


Рисунок 26 - Целевая функция

Решение задачи с помощью приложения MS Excel. Первым этапом решения необходимо найти максимальное значение целевой функции. В приложении MS Excel необходимо составить таблицу, таблица представлена на рисунке 27

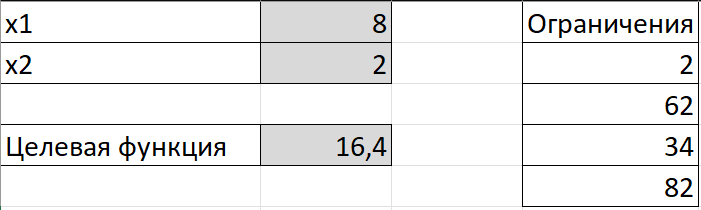


Рисунок 27 - Нахождение максимального значения целевой функции

Далее необходимо заполнить окно Поиск решений с учетом исходных данных. Окно представлено на рисунке 28

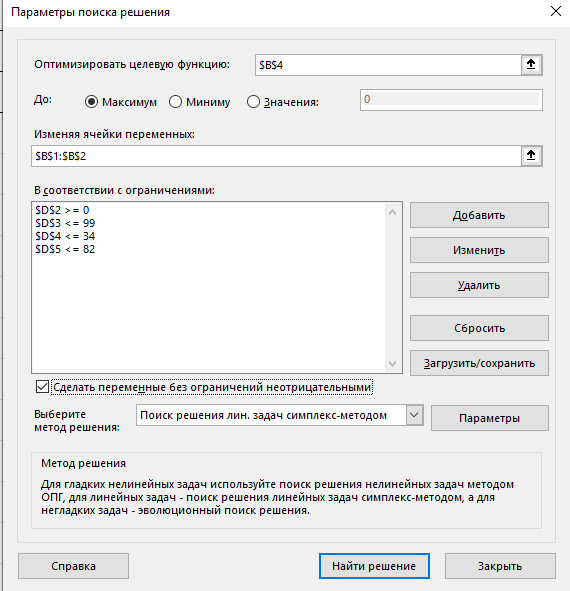


Рисунок 28 - Окно поиск решений

В результате выполнения поиска решения будет максимальное значение целевой функции (Z=16,4), а так же координаты точек х1 и х2.

Вторым шагом в решении этой задачи будет выполнение учета расхода денежных средств, с учетом максимальной экономии. Для этого необходимо дополнить таблиц данными о стоимость и расходе денежных средств. Результат представлен на рисунке 29

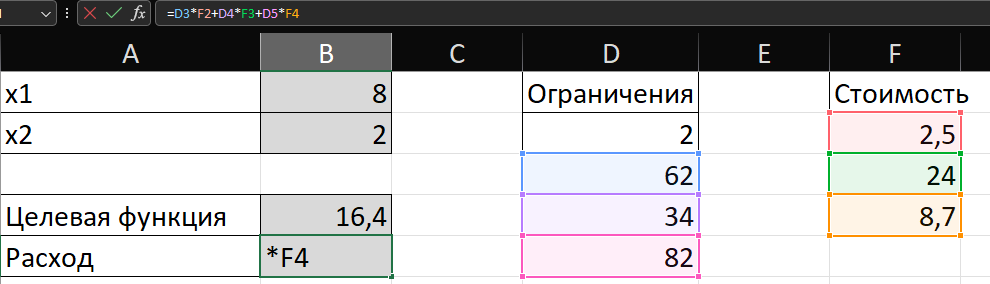


Рисунок 29 - Данные о стоимость и расходе денежных средств

Теперь в окне Поиск решений необходимо внести необходимые корректировки. Окно Поиск решений представлено на рисунке.30

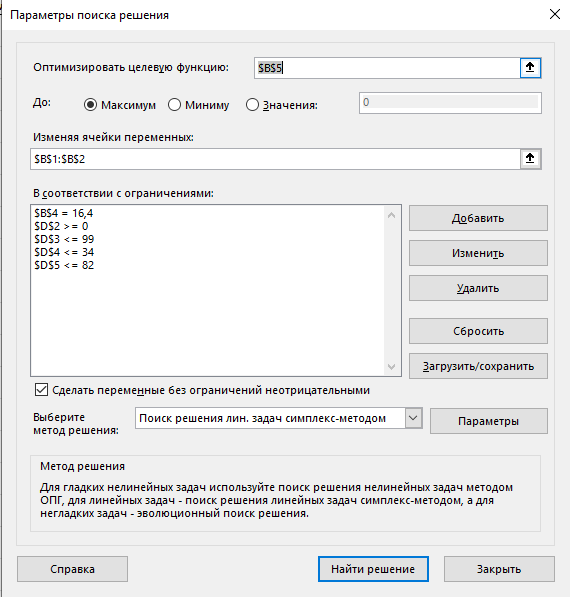


Рисунок 30 - Окно поиск решений

В результате вычислений координаты точек совпадают с аналитическими расчетами. Результат вычислений представлен на рисунке 31

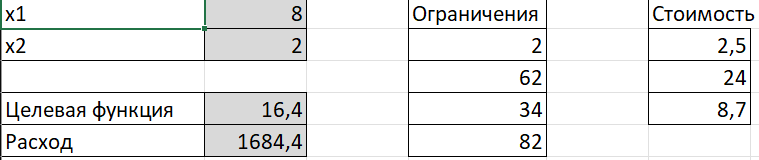


Рисунок 31 - Результат вычислений

## Определение корреляционной зависимости между рядами наблюдений

Корреляционный анализ – это совокупность методов обнаружения так называемой корреляционной зависимости между случайными величинами.

Для решения задачи корреляционной зависимости между рядами наблюдений необходимо выполнить последовательные шаги для достижения результата. Шаги решения представлены ниже.

1 Построить график зависимости между переменными (точечная диаграмма), по которой подобрать модель уравнение регрессии.

Значение переменных представлены на рисунке 32

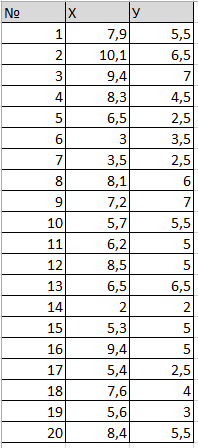


Рисунок 32 - Значения переменных

График зависимости представлен на рисунке 33 .

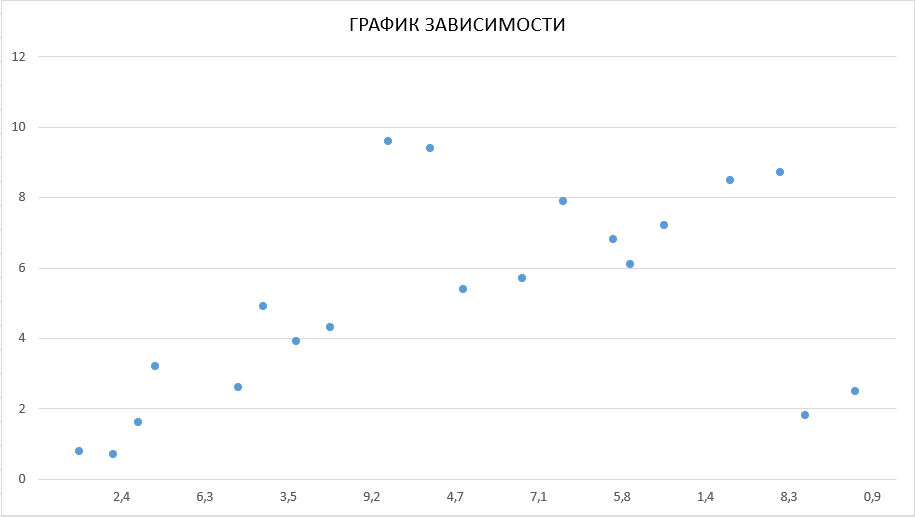


Рисунок 33 - График зависимости

На основании поля корреляции можно выдвинуть гипотезу о том, что связь между всеми возможными значениями X и Y носит линейный характер.

Для вычисления оценки математических ожиданий случайных величин X и Y была построена таблица. Результат представлен на рисунке 34.

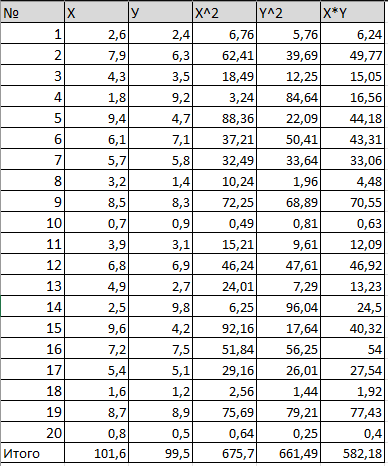


Рисунок 34 - Случайные величины X и Y

Оценка математических ожиданий случайной величины X (среднее арифметическое):

Оценка математических ожиданий случайной величины Y (среднее арифметическое):

1. Оценка средних квадратичных отклонений.

Расчёт выборочной дисперсии:

Оценка средних квадратичных отклонений:

1. Оценка коэффициента корреляции между X и Y и определение его значимости и надёжности.

Расчет ковариации:

Расчет тесноты связи:

Линейный коэффициент корреляции принимает значения от -1 до +1.

1. Определение параметра уравнения линейной регрессии.

Из выполненных расчетов получаем эмпирические коэффициенты регрессии: b = 0.48, a = 5.531.

Уравнение регрессии принимает следующий вид: y = 0.48x + 5.531

Далее необходимо построить прямую на графике. Результат представлен на рисунке 35.

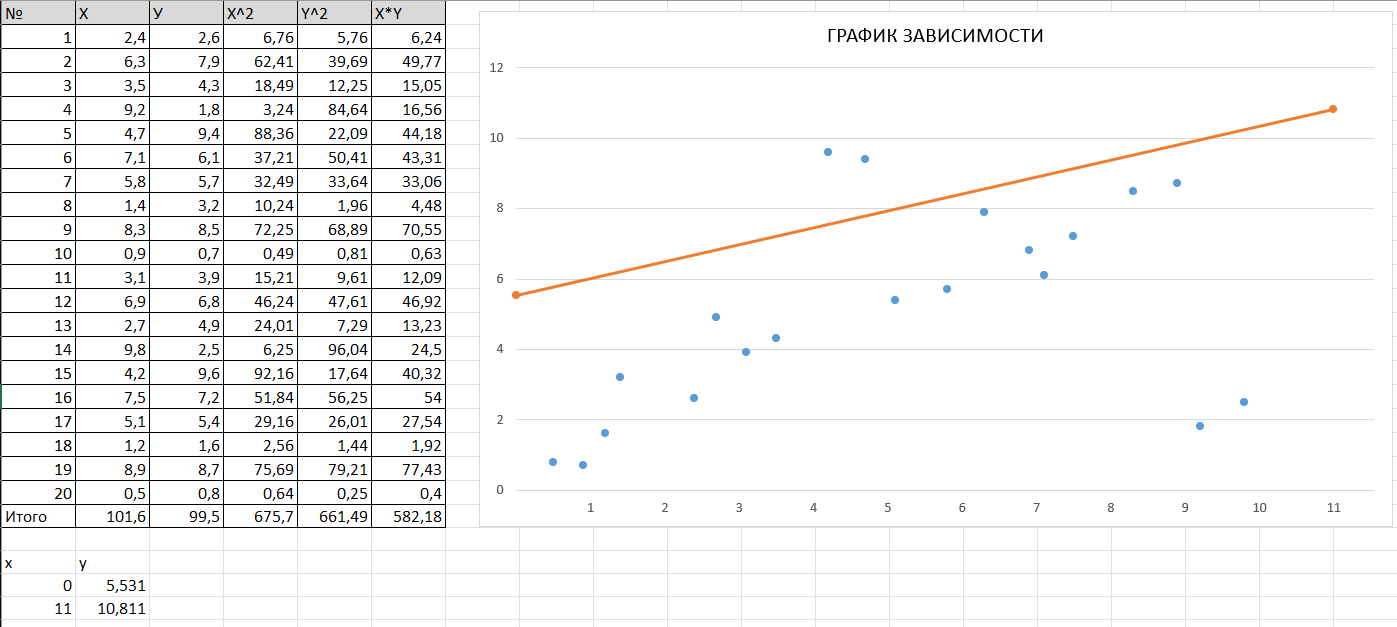


Рисунок 35 - построенная прямая

1. Оценка качества уравнения регрессии с помощью ошибки аппроксимации, характеризующей различие между фактическим значением и расчетным по уравнению регрессии.

Ошибка аппроксимации в пределах 52%-56% свидетельствует о хорошем подборе уравнения регрессии к исходным данным. Поскольку ошибка больше 55%, то данное уравнение не желательно использовать в качестве регрессии.

7 Степень тесноты связи между переменными

Связи между признаками могут быть слабыми и сильными (тесными). Их критерии оцениваются по шкале Чеддока:

0.1 < < 0.3: слабая;

0.3 < < 0.5: умеренная;

0.5 < < 0.7: заметная;

0.7 < < 0.9: высокая;

0.9 < < 1: весьма высокая;

Связь между признаком Y и фактором X умеренная и прямая.

8 Значимость коэффициента корреляции

Так как исходные данные являются выборочными, необходимо оценить значимость величины коэффициента корреляции. Для этого выдвинем нулевую гипотезу о незначимости коэффициента корреляции.

Выдвигаем гипотезы:

H0: rxy = 0, нет линейной взаимосвязи между переменными;

H1: rxy ≠ 0, есть линейная взаимосвязь между переменными;

Для того чтобы при уровне значимости α проверить нулевую гипотезу о равенстве нулю генерального коэффициента корреляции нормальной двумерной случайной величины при конкурирующей гипотезе H1 ≠ 0, надо вычислить наблюдаемое значение критерия (величина случайной ошибки).

и по таблице критических точек распределения Стьюдента, по заданному уровню значимости α и числу степеней свободы k = n - 2 найти критическую точку tкрит двусторонней критической области. Если tнабл < tкрит оснований отвергнуть нулевую гипотезу. Если |tнабл| > tкрит — нулевую гипотезу отвергают.

По таблице Стьюдента с уровнем значимости α=0.05 и степенями свободы k=18 находим tкрит:

tкрит(n-m-1;α/2) = tкрит(18;0.025) = 2.445

где m = 1 - количество объясняющих переменных.

Если |tнабл| > tкрит, то полученное значение коэффициента корреляции признается значимым (нулевая гипотеза, утверждающая равенство нулю коэффициента корреляции, отвергается).

Поскольку |tнабл| > tкрит, то отклоняем гипотезу о равенстве 0 коэффициента корреляции. Другими словами, коэффициент корреляции статистически значим.

9 Коэффициент эластичности.

Коэффициенты регрессии нежелательно использовать для непосредственной оценки влияния факторов на результативный признак в том случае, если существует различие единиц измерения результативного показателя у и факторного признака х.

Для этих целей вычисляются коэффициенты эластичности.

Средний коэффициент эластичности E показывает, на сколько процентов в среднем по совокупности изменится результат у от своей средней величины при изменении фактора x на 1% от своего среднего значения.

Коэффициент эластичности находится по формуле:

Коэффициент эластичности меньше 1. Следовательно, при изменении Х на 1%, Y изменится менее чем на 1%. Другими словами - влияние Х на Y не существенно.

Выводы

Изучена зависимость Y от X. На этапе спецификации была выбрана парная линейная регрессия. Оценены её параметры. Возможна экономическая интерпретация параметров модели - увеличение X на 1 ед.изм. приводит к увеличению Y в среднем на 0.48 ед.изм.

Вывод по разделу.

"Моделирование в программных системах" было рассмотрено значение моделирования для анализа, проектирования и оптимизации программных систем.

Методы математического моделирования были выделены как ключевой инструмент в этом процессе, позволяющий абстрагировать сложные системы и анализировать их поведение с помощью математических уравнений и алгоритмов.

# ЗАКЛЮЧЕНИЕ

Учебная практика по модулю "ПМ 02 Осуществление интеграции программных модулей" представляет важное звено в процессе формирования специалистов в области информационных технологий. Этот этап обучения не только предоставляет ценный опыт работы в реальной профессиональной среде, но и развивает ключевые навыки, необходимые для успешной карьеры в сфере разработки программного обеспечения.

Ознакомление с основными аспектами разработки программного обеспечения «Продажа компьютерных игр» на практике, включая анализ требований, интеграцию модулей, отладку, тестирование и инспектирование компонент, позволяющий углубить свои знания и применить их на практике.

Использование современных инструментов разработки, таких как Microsoft Visual Studio и Microsoft Visio, дает возможность студентам освоить профессиональные инструменты и приобрести опыт работы в реальных проектах.

Полученный опыт на учебной практике способствует развитию аналитических умений, коммуникативных навыков, умения работать в команде и решать проблемы эффективно. Эти компетенции являются важными для успешной карьеры в области информационных технологий, где важно не только техническое мастерство, но и способность к адаптации к изменениям и коллективной работе.

Таким образом, учебная практика по модулю "ПМ 02 Осуществление интеграции программных модулей" играет ключевую роль в подготовке будущих специалистов в области информационных технологий, обогащая их опыт и навыки, необходимые для успешной карьеры в данной сфере. Этот этап обучения является важным шагом на пути к профессиональному росту и достижению успеха в IT-индустрии.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Информационные технологии в профессиональной деятельности: учебник / Филимонова Е.В. — Москва: КноРус, 2021. — 482 с.
2. Э. Дастин, Тестирование программного обеспечения. Внедрение, управление и автоматизация / Э. Дастин, Д. Рэшка, Д. Пол; Пер. с англ. М. Павлов. — М.: Лори, 2017. — 567 c.
3. И.Г.Семакин, Основы алгоритмизации и программирования: учебник для студ. учреждений сред. проф. образования / И.Г.Семакин, А.П.Шестаков. — М.: Издательский центр «Академия», 2017. — 304
4. Иванов, А.А. Автоматизация технологических процессов и производств: Учебное пособие / А.А. Иванов. — М.: Форум, 2018. — 224 c
5. Компьютерное моделирование и исследование систем автоматического управления: Учебно-методическое пос : учебнометодическое пособие / Федоров С.Е. — Москва : Русайнс, 2020. — 92 с.

Маркин, А. В. Программирование на sql в 2 ч. Часть 2: учебник и практикум для бакалавриата и магистратуры / А. В. Маркин. — М.: Издательство Юрайт, 2019. — 292 с.

1. Ангарский промышленно-экономический техникум – ГБПОУ ИО «АПЭТ» [Электронный ресурс]: Официальный сайт ГБПОУ ИО «Ангарский промышленно-экономический техникум», 2008-2022. URL: <http://a-pet.ru/> (дата обращения: 22.02.2024).
2. Добро пожаловать в интегрированную среду разработки Visual Studio [Электронный ресурс] – Microsoft, 2022. Режим доступа: https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2022 (дата обращения: 22.02.2024
3. Microsoft Visual Studio [Электронный ресурс] – Wikipedia, 2022. Режим доступа: https://ru.wikipedia.org/wiki/Microsoft\_Visual\_Studio (дата обращения: 22.02.2024).
4. SQL [Электронный ресурс] – htmlacademy.ru, 2022. Режим доступа: https://htmlacademy.ru/tutorial/php/sql (дата обращения 22.02.2024).
5. Создание макетов [Электронный ресурс] – maket-master.ru, 2022. Режим доступа: http://maket-master.ru (дата обращения: 22.02.2024).
6. Расчет экономической эффективности [Электронный ресурс] – business.ru, 2022. Режим доступа: https://www.business.ru/article/1810-raschet-ekonomicheskoy-effektivnosti (дата обращения: 22.02.2024).
7. Интеграция приложений [Электронный ресурс] – osp.ru, 2022. Режим доступа: https://www.osp.ru/os/2006/09/3776464 (дата обращения: 22.02.2024).
8. Электронный фонд правовых и нормативно-технических документов [электронный ресурс] URL: https://docs.cntd.ru/document/1200082859(дата обращения: 22.02.2024).
9. IT MASTER GURU [электронный ресурс] URL:https://itmaster.guru/nastrojka-interneta/kak-po-lokalnoy-seti-podklyuchit printer.html(дата обращения: 22.02.2024).
10. Справочник [электронный ресурс] URL:https://spravochnick.ru/informatika/soprovozhdenie\_i\_obsluzhivanie\_progra mnogo\_obespecheniya\_kompyuternyh\_sistem(дата обращения: 22.02.2024).

# ПРИЛОЖЕНИЕ А

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Министерство образования Иркутской области

Государственное бюджетное профессиональное образовательное учреждение

Иркутской области

**«Ангарский промышленно-экономический техникум»**

(ГБПОУ ИО «АПЭТ»)

|  |
| --- |
| Автоматизация системы сведения о продаже компьютерных игр |
| Наименование ИС |
| ПРОДАЖА КОМПЬЮТЕРНЫХ ИГР |
| Сокращенное наименование ИС |
| **ТЕХНИЧЕСКОЕ ЗАДАНИЕ** |
| Действует с 05.03.2024 |

|  |
| --- |
| Составитель |
| Коновалов Сергей Валадимирович |
| ФИО сотавителя |
| Разработчик |
| Должность составителя |

г. Ангарск, 2024 г.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ. 3](file:///\\apetfs\ИСПП-3\Коновалов_СВ\4%20курс\Учебная%20практика\ИС%20Автотрансопорт\Доки\Приложения\1%20ТЗ..docx#_Toc28338856)

[1.Основание для разработки 3](file:///\\apetfs\ИСПП-3\Коновалов_СВ\4%20курс\Учебная%20практика\ИС%20Автотрансопорт\Доки\Приложения\1%20ТЗ..docx#_Toc28338857)

[2.Назначение разработки 3](file:///\\apetfs\ИСПП-3\Коновалов_СВ\4%20курс\Учебная%20практика\ИС%20Автотрансопорт\Доки\Приложения\1%20ТЗ..docx#_Toc28338858)

[3.Технические требования к программе или программному изделию 4](file:///\\apetfs\ИСПП-3\Коновалов_СВ\4%20курс\Учебная%20практика\ИС%20Автотрансопорт\Доки\Приложения\1%20ТЗ..docx#_Toc28338859)

[3.1. Требования к функциональным характеристикам 4](file:///\\apetfs\ИСПП-3\Коновалов_СВ\4%20курс\Учебная%20практика\ИС%20Автотрансопорт\Доки\Приложения\1%20ТЗ..docx#_Toc28338860)

[3.2. Требования к надежности 4](file:///\\apetfs\ИСПП-3\Коновалов_СВ\4%20курс\Учебная%20практика\ИС%20Автотрансопорт\Доки\Приложения\1%20ТЗ..docx#_Toc28338861)

[3.3. Условия эксплуатации 4](file:///\\apetfs\ИСПП-3\Коновалов_СВ\4%20курс\Учебная%20практика\ИС%20Автотрансопорт\Доки\Приложения\1%20ТЗ..docx#_Toc28338862)

[3.4. Требования к составу и параметрам технических средств 5](file:///\\apetfs\ИСПП-3\Коновалов_СВ\4%20курс\Учебная%20практика\ИС%20Автотрансопорт\Доки\Приложения\1%20ТЗ..docx#_Toc28338863)

[3.5. Требования к информационной и программной совместимости 5](file:///\\apetfs\ИСПП-3\Коновалов_СВ\4%20курс\Учебная%20практика\ИС%20Автотрансопорт\Доки\Приложения\1%20ТЗ..docx#_Toc28338864)

[3.6. Требования к транспортированию и хранению 5](file:///\\apetfs\ИСПП-3\Коновалов_СВ\4%20курс\Учебная%20практика\ИС%20Автотрансопорт\Доки\Приложения\1%20ТЗ..docx#_Toc28338865)

[3.7. Специальные требования 5](file:///\\apetfs\ИСПП-3\Коновалов_СВ\4%20курс\Учебная%20практика\ИС%20Автотрансопорт\Доки\Приложения\1%20ТЗ..docx#_Toc28338866)

# ВВЕДЕНИЕ

В настоящее время автоматизация и создание информационных систем считаются одними из самых трудоемких областей в игровой индустрии. Одной из главных причин такого активного развития этой области является то, что автоматизация является основой для революционных изменений в процессах продажи игр, которые играют ключевую роль в деятельности игровых компаний и игроков.

В результате возникают системы управления продажами игр, которые направлены на поддержание и улучшение процесса продажи с помощью устройства управления (комплекса средств сбора, обработки, передачи информации и формирования управляющих сигналов или команд).

Информационная система по продажам игр, в свою очередь, представляет собой систему, предоставляющую авторизованному персоналу данные или информацию, связанную с продажами игр.

Основная цель информационных систем по продажам игр заключается в том, чтобы производить необходимую информацию для эффективного управления всеми ресурсами, связанными с продажей игр.

# Основание для разработки

Основным основанием для разработки информационной системы по продажам игр является необходимость автоматизации и оптимизации процессов продажи игр. Такая система может помочь игровым компаниям в следующих задачах:

* Управление каталогом игр и данными о продуктах;
* Обработка заказов и платежей;
* Управление запасами и доставкой;
* Отслеживание продаж и аналитика;
* Управление отношениями с клиентами (CRM);
* Маркетинг и продвижение игр.

# Назначение разработки

Основным назначением разработки информационной системы по продажам игр является предоставление игровым компаниям комплексного решения для автоматизации и оптимизации процессов продажи игр. Такая система должна выполнять следующие функции:

* Управление каталогом игр и данными о продуктах: Хранение и управление информацией о играх, включая названия, описания, цены, скриншоты;
* Обработка заказов и платежей: Прием и обработка заказов на игры, включая сбор платежей и выставление счетов;
* Управление запасами и доставкой: Отслеживание запасов игр и управление доставкой заказов клиентам;
* Отслеживание продаж и аналитика: Сбор и анализ данных о продажах для получения ценной информации о тенденциях продаж, предпочтениях клиентов и эффективности маркетинговых кампаний;
* Управление отношениями с клиентами (CRM): Хранение и управление данными о клиентах, включая историю заказов, предпочтения и отзывы;
* Маркетинг и продвижение игр: Предоставление инструментов для маркетинга и продвижения игр, таких как создание и рассылка электронных писем, управление социальными сетями и проведение рекламных кампаний.

# Технические требования к программе или программному изделию

## Требования к функциональным характеристикам.

Информационная система по продажам игр должна обладать следующими функциональными характеристиками:

1. Управление каталогом игр и данными о продуктах:

* Добавление, редактирование и удаление игр из каталога
* Управление информацией о продуктах, включая названия, описания, цены, скриншоты и видео
* Категоризация и фильтрация игр по различным критериям

2. Обработка заказов и платежей:

* Прием и обработка заказов на игры
* Интеграция с платежными шлюзами для безопасного сбора платежей
* Выставление счетов и отслеживание статуса оплаты

3. Управление запасами и доставкой:

* Отслеживание запасов игр в режиме реального времени
* Управление доставкой заказов, включая выбор способа доставки и отслеживание статуса доставки
* Обработка возвратов и обменов

4. Отслеживание продаж и аналитика:

* Сбор и анализ данных о продажах, включая количество проданных игр, средний размер заказа и доход
* Создание отчетов и графиков для визуализации данных о продажах
* Идентификация тенденций продаж и возможностей для роста

5. Управление отношениями с клиентами (CRM):

* Хранение и управление данными о клиентах, включая имена, адреса электронной почты, историю заказов и предпочтения
* Сегментация клиентов на основе различных критериев
* Отправка персонализированных электронных писем и сообщений клиентам

6. Маркетинг и продвижение игр:

* Создание и рассылка электронных писем для продвижения игр
* Управление социальными сетями и проведение рекламных кампаний
* Отслеживание эффективности маркетинговых кампаний и оптимизация их для повышения рентабельности инвестиций

## Требования к надежности

Для обеспечения надежности необходимо проверять корректность вводимых данных пользователя.

## Условия эксплуатации

Данный программный продукт предполагается эксплуатировать в условиях, благоприятных для оператора ЭВМ. Предполагается, что обслуживать данный программный продукт будет один человек –администратор. Администратор должен обеспечивать нормальное функционирование программного продукта и ввести базу о продажах игр пользующийся этим программным продуктом.

## Требования к составу и параметрам технических средств

Для нормального функционирования программного средства должны соответствовать:

* процессор Intel Pentium III с тактовой частотой от 75 Мгц и выше;
* оперативная память от 128 Мб;
* жесткий диск c объемом памяти не менее 60 Мб свободного дискового пространства;
* видеокарта, с объемом оперативной памяти 512 Мб.

## Требования к информационной и программной совместимости

Системные программные средства, используемые программой, должны быть представлены лицензионной локализованной версией операционной системы:

* Windows NT 10.0 (2018);
* Windows NT 10.0 (2019);
* Windows NT 10.0 (2022).

## Требования к транспортированию и хранению

Программа поставляется на разных носителях таких как Usb флеш накопителе или на лазерном носителе информации.

## Специальные требования.

Разрабатываемый программный продукт должен обладать следующими требованиями:

* программа должна иметь простой интуитивно понятный интерфейс;
* редактировать устаревшую информацию;
* обеспечивать целостность хранимой информации.

# ПРИЛОЖЕНИЕ Б

РЕАЛИЗАЦИЯ ИНТЕРФЕЙСА ПРИЛОЖЕНИЯ

Разработанные в MS Visual Studio формы продемонстрированы на рисунках 1-6.

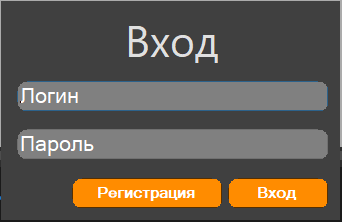


Рисунок 1 – Прототип авторизация

Свойства элементов формы Авторизация представлены в таблице 1.

Таблица 1 – Свойства элементов формы «Авторизация»

|  |  |  |
| --- | --- | --- |
| Объект | Свойство | Значение |
| Form1 | Name | Form1 |
| BackColor | 64; 64; 64 |
| FormBorderStyle | None |
| Size | 435; 430 |
| Label1 | Font | Times New Roman; 27,75pt; style=Italic |
| Text | Вход |
| ForeColor | White |
| Label2 | Font | Microsoft Sans Serif; 15pt; style=Bold |
| Text | Логин |
| ForeColor | White |
| Label3 | Font | Microsoft Sans Serif; 15pt; style=Bold |
| Text | Пароль |
| ForeColor | White |
| Button1 | Text | войти |
| Location | 12; 94 |
| Button1 | Location | 12; 164 |
| Text | Зарегистироваться |

Форма «Добавить игру» представлена на рисунке 2

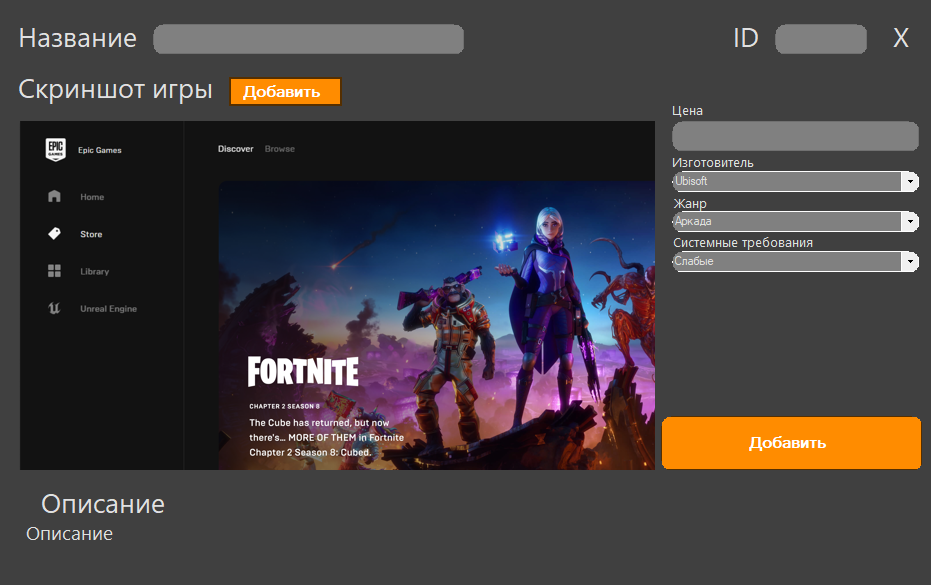


Рисунок 2 –форма «Добавить игру»

Свойства элементов формы «Добавить игру» представлены в таблице 5.

Таблица 5 – Свойства элементов формы «Добавить игру»

|  |  |  |
| --- | --- | --- |
| Объект | Свойство | Значение |
| Добавить игру | Name | Добавить игру |
| BackColor | 64; 0; 0 |
| FormBorderStyle | None |
| Size | 740; 410 |
| Label1 | Font | Times New Roman; 24pt; style=Bold, Italic |
| Text | Название |
| ForeColor | White |
| Label2 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | White |
| Text | Скриншот игры |
| Label3 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | silver |
| Text | ID |
| Button1 | Text | Добавить скриншот |
| Location | 12; 94 |
| Button2 | Location | 12; 164 |
| Text | Добавить |
| Label4 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | silver |
| Text | Цена |
| Label5 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | silver |
| Text | Изготовитель |
| Label6 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | silver |
| Text | Жанр |
| Label7 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | silver |
| Text | Системные требования |
| Label8 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | silver |
| Text | Описание |
| TextBox1 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | silver |
| TextBox2 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | silver |
| TextBox3 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | silver |
| ComboBox1 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | silver |
| ComboBox1 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | silver |
| ComboBox1 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | silver |

Форма «Сведения» представлена на рисунке 3

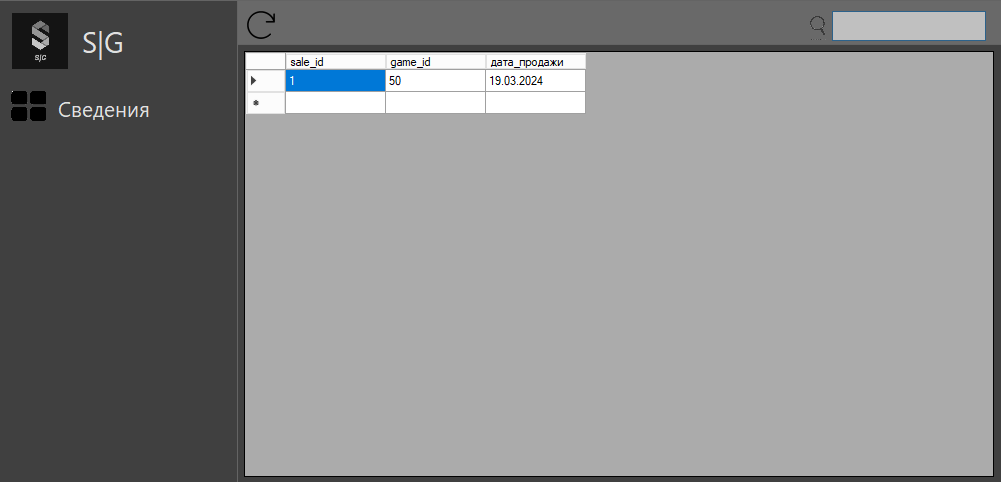


Рисунок 3 –форма «Сведения»

Свойства элементов формы «Сведения» представлены в таблице 7.

Таблица 7 – Свойства элементов формы «Сведения»

|  |  |  |
| --- | --- | --- |
| Объект | Свойство | Значение |
| Сведения | Name | Сведения |
| BackColor | 64; 0; 0 |
| FormBorderStyle | None |
| Size | 740; 410 |
| Label1 | Font | Times New Roman; 24pt; style=Bold, Italic |
| Text | Название |
| ForeColor | White |
| Label2 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | White |
| Text | Скриншот игры |
| TextBox1 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | silver |
| DataGridView1 | Location | 12; 164 |
| data | Продажи |

Форма «Пользователь» представлена на рисунке 4

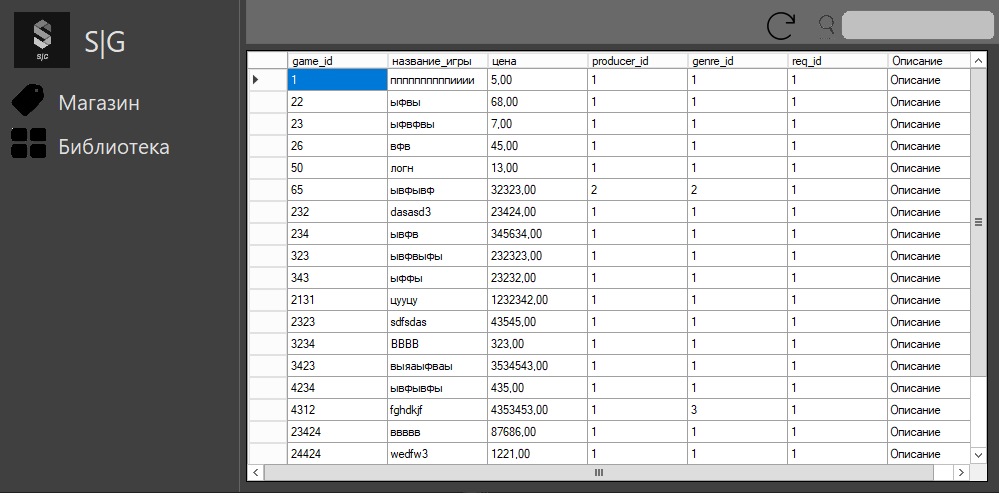


Рисунок 4 – прототип главной формы «Пользователь»

Свойства элементов формы «Пользователь» представлены в таблице 9.

Таблица 9 – Свойства элементов формы «Пользователь»

|  |  |  |
| --- | --- | --- |
| Объект | Свойство | Значение |
| Пользователь | Name | Пользователь |
| BackColor | 64; 0; 0 |
| FormBorderStyle | None |
| Size | 740; 410 |
| Label1 | Font | Times New Roman; 24pt; style=Bold, Italic |
| Text | Магазин |
| ForeColor | White |
| Label2 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | White |
| Text | Библеотека |
| DataGridView1 | Location | 12; 164 |
| data | Игры |
| TextBox1 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | White |

Форма «Игра» представлена на рисунке 5

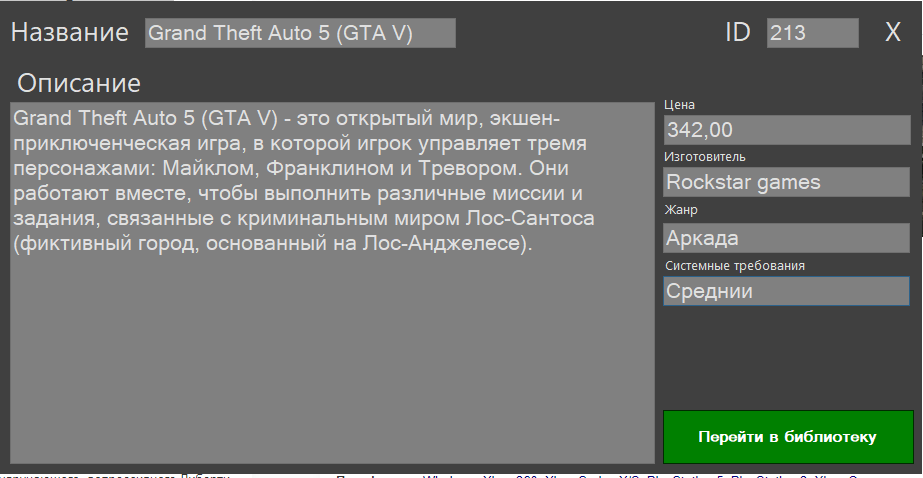


Рисунок 5 – прототип главной формы «Игра»

Свойства элементов формы «Игра» представлены в таблице 11.

Таблица 11 – Свойства элементов формы «Игра»

|  |  |  |
| --- | --- | --- |
| Объект | Свойство | Значение |
| Игра | Name | Игра |
| BackColor | 64; 0; 0 |
| FormBorderStyle | None |
| Size | 740; 410 |
| Label1 | Font | Times New Roman; 24pt; style=Bold, Italic |
| Text | Название |
| ForeColor | White |
| Label2 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | White |
| Text | Скриншот игры |
| Label3 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | silver |
| Text | ID |
| Button1 | Location | 12; 164 |
| Text | Добавить в библеотеку |
| Label4 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | silver |
| Text | Цена |
| Label5 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | silver |
| Text | Изготовитель |
| Label6 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | silver |
| Text | Жанр |
| Label7 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | silver |
| Text | Системные требования |
| Label8 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | silver |
| Text | Описание |
| TextBox1 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | silver |
| TextBox2 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | silver |
| TextBox3 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | silver |
| TextBox4 | Font | Times New Roman; 24pt;style=Bold, Italic |
| ForeColor | silver |
| TextBox5 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | silver |
| TextBox6 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | silver |

Форма «Библиотека» представлена на рисунке 6

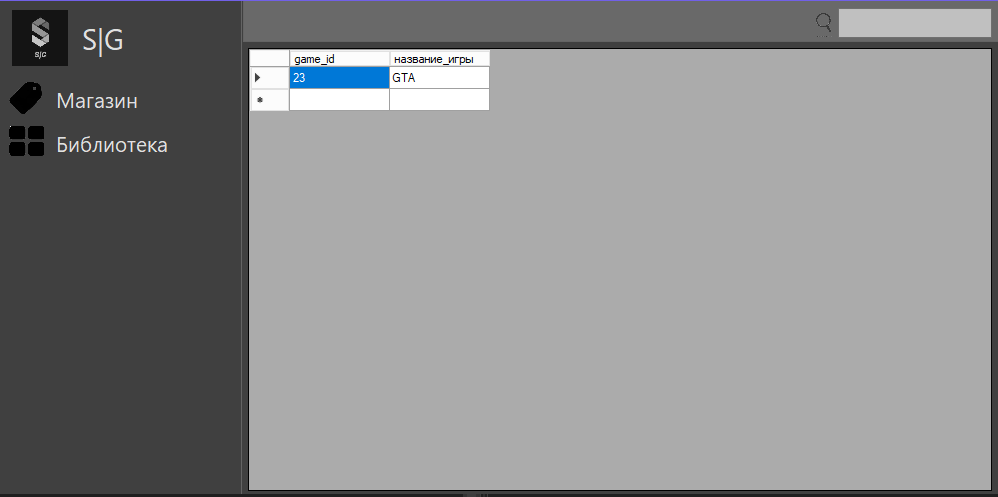


Рисунок 6 –форма «Библиотека»

Свойства элементов формы «Библиотека» представлены в таблице 12.

Таблица 12 – Свойства элементов формы «Библиотека»

|  |  |  |
| --- | --- | --- |
| Объект | Свойство | Значение |
| Библиотека | Name | Библиотека |
| BackColor | 64; 0; 0 |
| FormBorderStyle | None |
| Size | 740; 410 |
| Label1 | Font | Times New Roman; 24pt; style=Bold, Italic |
| Text | Магазин |
| ForeColor | White |
| Label2 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | White |
| Text | Библеотека |
| TextBox1 | Font | Times New Roman; 24pt; style=Bold, Italic |
| ForeColor | silver |
| DataGridView1 | Location | 12; 164 |
| data | Игра |

# ПРИЛОЖЕНИЕ В

ЛИСТИН ПРОГРАММНОГО КОДА

Код формы авторизации представлены в таблице 2

Таблица 2 – Код формы «Авторизация»

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Data.Common;  using System.Data.SqlClient;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  namespace SG\_Продажа\_игр  {  public partial class Form1 : Form  {  public Form1()  {  InitializeComponent();  ShowPlaceholder();  }  private void ShowPlaceholder()  {  // Показываем надпись только если TextBox пустой  if (string.IsNullOrEmpty(textBox1.Text))  {  textBox1.Text = "Логин";  textBox1.ForeColor = System.Drawing.Color.White; // Цвет текста серый  }  else if (string.IsNullOrEmpty(textBox2.Text))  {  textBox2.Text = "Пароль";  textBox2.ForeColor = System.Drawing.Color.White; // Цвет текста серый  }  }  public static string rol = "";  public string CmdText = "SELECT login,Name\_ak,Pas,buy + sale AS 'Стоимость аккаунта',guard,phon,phon\_date FROM [ak]";  public string ConnString = "Data Source=192.168.1.3;" +  "Initial Catalog=Коновалов\_СВ\_УП\_02;" +  "User id=КоноваловСергей;" +  "Password=1202;";//Глобальная БД  public static string connectString = @"Data Source=sql;Initial Catalog=Коновалов\_СВ\_УП\_02;Integrated Security=true";  public static string a, b;  private SqlConnection myConnection;  public SqlDataAdapter da = null;  public DataSet ds = new DataSet();  public SqlConnection connP = new SqlConnection();  private void Form1\_Load(object sender, EventArgs e)  {  raundclass.ApplyRoundedCorners(textBox1, 10);  raundclass.ApplyRoundedCorners(textBox2, 10);  raundclass.ApplyRoundedCorners(button1, 10);  raundclass.ApplyRoundedCorners(button2, 10);  }  private void button2\_Click(object sender, EventArgs e)  {  this.Hide();  регистрация frm2 = new регистрация();  frm2.Show();  }  private void textBox1\_MouseEnter(object sender, EventArgs e)  {  if (textBox1.Text == "Логин")  {  textBox1.Text = "";  textBox1.ForeColor = Color.White;  }  }  private void textBox1\_MouseLeave(object sender, EventArgs e)  {  ShowPlaceholder();  }  private void textBox2\_MouseEnter(object sender, EventArgs e)  {  if (textBox2.Text == "Пароль")  {  textBox2.Text = "";  textBox2.ForeColor = Color.White;  }  }  private void textBox2\_MouseLeave(object sender, EventArgs e)  {  ShowPlaceholder();  }  private void button1\_Click(object sender, EventArgs e)  {  if (string.IsNullOrEmpty(textBox1.Text) || string.IsNullOrEmpty(textBox2.Text))  {  MessageBox.Show("Все поля должны быть заполнены");  return;  }  myConnection = new SqlConnection(connectString);  myConnection.Open();  string query = "SELECT Логин, пароль, Роль From Пользователи";  DataTable dt = new DataTable();  da = new SqlDataAdapter(query, myConnection);  da.Fill(dt);  Boolean flag = false;  foreach (DataRow row in dt.Rows)  {  if (row["Логин"].ToString() == textBox1.Text && row["пароль"].ToString() == textBox2.Text)  {  flag = true;  if (row["Роль"].ToString() == "Админестратор")  {  this.Hide();  Админестратор frm2 = new Админестратор();  frm2.Show();  }  else if (row["Роль"].ToString() == "Пользователь")  {  this.Hide();  Пользователь frm2 = new Пользователь();  frm2.Show();  }  break;  }  }  if (flag == false)  {  MessageBox.Show("Неверный логин или пароль");  }  myConnection.Close();  }  }  } |

Код формы «Добавить игру» представлены в таблице 6.

Таблица 6 – Код формы «Добавить игру»

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Data.SqlClient;  using System.Drawing;  using System.Drawing.Imaging;  using System.IO;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  using static System.Windows.Forms.VisualStyles.VisualStyleElement;  namespace SG\_Продажа\_игр  {  public partial class Добавить\_игру : Form  {  // Измененный конструктор, который принимает аргумент типа Админестратор  public Добавить\_игру()  {  InitializeComponent();  }  private void Добавить\_игру\_Load(object sender, EventArgs e)  {  raundclass.ApplyRoundedCorners(textBox2, 10);  raundclass.ApplyRoundedCorners(textBox1, 10);  raundclass.ApplyRoundedCorners(textBox4, 10);  raundclass.ApplyRoundedCorners(button1, 10);  raundclass.ApplyRoundedCorners(comboBox2, 10);  raundclass.ApplyRoundedCorners(comboBox3, 10);  raundclass.ApplyRoundedCorners(comboBox1, 10);  string connectionString = @"Data Source=sql;Initial Catalog=Коновалов\_СВ\_УП\_02;Integrated Security=True";  // Получаем данные о производителях и загружаем их в comboBox1  string query1 = "SELECT producer\_id, название\_фирмы FROM Производители";  using (SqlConnection connection = new SqlConnection(connectionString))  using (SqlCommand command = new SqlCommand(query1, connection))  {  connection.Open();  SqlDataAdapter adapter = new SqlDataAdapter(command);  DataTable dataTable = new DataTable();  adapter.Fill(dataTable);  comboBox1.DataSource = dataTable;  comboBox1.ValueMember = "producer\_id";  comboBox1.DisplayMember = "название\_фирмы";  }  // Получаем данные о жанрах игр и загружаем их в comboBox2  string query2 = "SELECT genre\_id, название\_жанра FROM Жанры";  using (SqlConnection connection = new SqlConnection(connectionString))  using (SqlCommand command = new SqlCommand(query2, connection))  {  SqlDataAdapter adapter = new SqlDataAdapter(command);  DataTable dataTable = new DataTable();  adapter.Fill(dataTable);  comboBox2.DataSource = dataTable;  comboBox2.ValueMember = "genre\_id";  comboBox2.DisplayMember = "название\_жанра";  }  // Получаем данные о требованиях к системе и загружаем их в comboBox3  string query3 = "SELECT req\_id, Видеокарта FROM Системные\_требования";  using (SqlConnection connection = new SqlConnection(connectionString))  using (SqlCommand command = new SqlCommand(query3, connection))  {  SqlDataAdapter adapter = new SqlDataAdapter(command);  DataTable dataTable = new DataTable();  adapter.Fill(dataTable);  comboBox3.DataSource = dataTable;  comboBox3.ValueMember = "req\_id";  comboBox3.DisplayMember = "Видеокарта";  }  pictureBox1.DragEnter += new DragEventHandler(pictureBox1\_DragEnter);  pictureBox1.DragDrop += new DragEventHandler(pictureBox1\_DragDrop);  }  private void button1\_Click(object sender, EventArgs e)  {  try  {  if (textBox1.Text == "" || comboBox1.Text == "" || textBox2.Text == "" || textBox4.Text == "" || comboBox2.Text == "" || comboBox3.Text == "")  {  MessageBox.Show("Заполните все поля");  }  else  {  string connectionString = @"Data Source=sql;Initial Catalog=Коновалов\_СВ\_УП\_02;Integrated Security=True";  string query = "INSERT INTO [Игры] ([game\_id],[название\_игры],[цена],[producer\_id],[genre\_id],[req\_id],[Описание]) VALUES (@game\_id, @название\_игры, @цена, @producer\_id, @genre\_id, @req\_id, @Описание)";  using (SqlConnection connection = new SqlConnection(connectionString))  using (SqlCommand command = new SqlCommand(query, connection))  {  int game\_id = Convert.ToInt32(textBox4.Text);  string название\_игры = Convert.ToString(textBox2.Text);  decimal цена = Convert.ToDecimal(textBox1.Text);  int producer\_id = Convert.ToInt32(comboBox1.SelectedValue);  int genre\_id = Convert.ToInt32(comboBox2.SelectedValue);  int req\_id = Convert.ToInt32(comboBox3.SelectedValue);  string Описание = Convert.ToString(label5.Text);  command.Parameters.AddWithValue("@game\_id", game\_id);  command.Parameters.AddWithValue("@название\_игры", название\_игры);  command.Parameters.AddWithValue("@цена", цена);  command.Parameters.AddWithValue("@producer\_id", producer\_id);  command.Parameters.AddWithValue("@genre\_id", genre\_id);  command.Parameters.AddWithValue("@req\_id", req\_id);  command.Parameters.AddWithValue("@Описание", Описание);  connection.Open();  command.ExecuteNonQuery();  }  this.Hide();  Админестратор frm2 = new Админестратор();  frm2.Show();  }  }  catch (Exception ex)  {  MessageBox.Show("Ошибка: " + ex.Message);  }  }  private void pictureBox1\_Click(object sender, EventArgs e)  {  pictureBox1.AllowDrop = true;  }  private void button2\_Click(object sender, EventArgs e)  {  OpenFileDialog openFileDialog = new OpenFileDialog();  openFileDialog.Filter = "Image files (\*.jpg, \*.jpeg, \*.png, \*.gif)|\*.jpg; \*.jpeg; \*.png; \*.gif";  openFileDialog.Title = "Выберите изображение";  DialogResult result = openFileDialog.ShowDialog();  if (result == DialogResult.OK)  {  string imagePath = openFileDialog.FileName;  try  {  pictureBox1.Image = Image.FromFile(imagePath);  }  catch (Exception ex)  {  MessageBox.Show("Ошибка загрузки изображения: " + ex.Message);  }  }  }  private void label10\_Click(object sender, EventArgs e)  {  this.Hide();  Админестратор frm2 = new Админестратор();  frm2.Show();  }  }  } |

Код формы «Сведения» представлены в таблице 8.

Таблица 8 – Код формы «Сведения»

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Data.SqlClient;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  namespace SG\_Продажа\_игр  {  public partial class Сведения : Form  {  public Сведения()  {  InitializeComponent();  }  private void textBox2\_TextChanged(object sender, EventArgs e)  {  SearchGames(textBox2.Text);  }  private void SearchGames(string gameName)  {  try  {  // Подключаемся к базе данных  string connectionString = @"Data Source=sql;Initial Catalog=Коновалов\_СВ\_УП\_02;Integrated Security=True";  using (SqlConnection connection = new SqlConnection(connectionString))  {  // Создаем SQL-запрос для поиска игр по названию  string query = "SELECT \* FROM Игры WHERE [название\_игры] LIKE @gameName";  // Создаем команду с параметрами  using (SqlCommand command = new SqlCommand(query, connection))  {  // Добавляем параметр для названия игры  command.Parameters.AddWithValue("@gameName", "%" + gameName + "%");  // Открываем соединение и выполняем команду  connection.Open();  SqlDataAdapter adapter = new SqlDataAdapter(command);  DataTable dataTable = new DataTable();  adapter.Fill(dataTable);  // Загружаем результаты поиска в DataGridView или другой контрол для отображения таблицы игр  dataGridView1.DataSource = dataTable;  }  }  }  catch (Exception ex)  {  MessageBox.Show("Ошибка при поиске игр: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  }  private void Сведения\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "коновалов\_СВ\_УП\_02DataSet.Продажи". При необходимости она может быть перемещена или удалена.  this.продажиTableAdapter.Fill(this.коновалов\_СВ\_УП\_02DataSet.Продажи);  }  SqlConnection con = new SqlConnection(@"Data Source=sql;Initial Catalog=Коновалов\_СВ\_УП\_02;Integrated Security=true");  private void pictureBox1\_Click(object sender, EventArgs e)  {  try  {  string commandText = "select \* from Продажи";  SqlDataAdapter itm = new SqlDataAdapter(commandText, con);  DataTable dt2 = new DataTable();  itm.Fill(dt2);  dataGridView1.DataSource = dt2;  con.Close();  }  catch  {  MessageBox.Show("Нечего обновлять");  }  }  private void pictureBox2\_Click(object sender, EventArgs e)  {  this.Close();  Админестратор frm2 = new Админестратор();  frm2.Show();  }  }  } |

Код формы «Пользователь» представлены в таблице 10.

Таблица 10 – Код формы «Пользователь»

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Data.SqlClient;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  namespace SG\_Продажа\_игр  {  public partial class Пользователь : Form  {  public Пользователь()  {  InitializeComponent();  dataGridView1.CellMouseClick += dataGridView1\_CellMouseClick;  }  private void Пользователь\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "коновалов\_СВ\_УП\_02DataSet.Игры". При необходимости она может быть перемещена или удалена.  this.игрыTableAdapter.Fill(this.коновалов\_СВ\_УП\_02DataSet.Игры);  raundclass.ApplyRoundedCorners(textBox1, 10);  }  SqlConnection con = new SqlConnection(@"Data Source=sql;Initial Catalog=Коновалов\_СВ\_УП\_02;Integrated Security=true");  private void SearchGames(string gameName)  {  try  {  // Подключаемся к базе данных  string connectionString = @"Data Source=sql;Initial Catalog=Коновалов\_СВ\_УП\_02;Integrated Security=True";  using (SqlConnection connection = new SqlConnection(connectionString))  {  // Создаем SQL-запрос для поиска игр по названию  string query = "SELECT \* FROM Игры WHERE [название\_игры] LIKE @gameName";  // Создаем команду с параметрами  using (SqlCommand command = new SqlCommand(query, connection))  {  // Добавляем параметр для названия игры  command.Parameters.AddWithValue("@gameName", "%" + gameName + "%");  // Открываем соединение и выполняем команду  connection.Open();  SqlDataAdapter adapter = new SqlDataAdapter(command);  DataTable dataTable = new DataTable();  adapter.Fill(dataTable);  // Загружаем результаты поиска в DataGridView или другой контрол для отображения таблицы игр  dataGridView1.DataSource = dataTable;  }  }  }  catch (Exception ex)  {  MessageBox.Show("Ошибка при поиске игр: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  }  private void pictureBox3\_Click(object sender, EventArgs e)  {  try  {  string commandText = "select \* from Игры";  SqlDataAdapter itm = new SqlDataAdapter(commandText, con);  DataTable dt2 = new DataTable();  itm.Fill(dt2);  dataGridView1.DataSource = dt2;  con.Close();  }  catch  {  MessageBox.Show("Нечего обновлять");  }  }  private void dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)  {  this.Hide();  int rowIndex = e.RowIndex;  // Проверяем, что индекс строки действителен  if (rowIndex >= 0 && rowIndex < dataGridView1.Rows.Count)  {  // Получаем данные из выбранной строки  DataGridViewRow selectedRow = dataGridView1.Rows[rowIndex];  string data1 = selectedRow.Cells[0].Value.ToString();  string data2 = selectedRow.Cells[1].Value.ToString();  string data3 = selectedRow.Cells[2].Value.ToString();  string data4 = selectedRow.Cells[3].Value.ToString();  string data5 = selectedRow.Cells[4].Value.ToString();  string data6 = selectedRow.Cells[5].Value.ToString();  // Открываем форму игры, передавая данные  OpenGameForm(data1, data2, data3, data4, data5, data6);  }  }  private void OpenGameForm(string data1, string data2, string data3, string data4, string data5, string data6 )  {  Игра gameForm = new Игра(data1, data2, data3, data4,data5, data6 );  gameForm.ShowDialog();  }  private void dataGridView1\_CellMouseClick(object sender, DataGridViewCellMouseEventArgs e)  {    }  private void label4\_Click(object sender, EventArgs e)  {  this.Close();  библеотека а = new библеотека();  а.Show();  }  private void label5\_Click(object sender, EventArgs e)  {  this.Close();  Пользователь а = new Пользователь();  а.Show();  }  }  } |

Код формы «Игра» представлены в таблице 12.

Таблица 12 – Код формы «Игра»

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Data.SqlClient;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  namespace SG\_Продажа\_игр  {  public partial class Игра : Form  {  private string data1;  private string data2;  private string data3;  private string data4;  private string data5;  private string data6;  public Игра(string data1, string data2, string data3, string data4, string data5, string data6 /\* и так далее \*/)  {  InitializeComponent();  this.data1 = data1;  this.data2 = data2;  this.data3 = data3;  this.data4 = data4;  this.data5 = data5;  this.data6 = data6;  textBox4.Text = data1;  textBox2.Text = data2;  textBox1.Text = data3;  textBox3.Text = data4;  textBox5.Text = data5;  textBox6.Text = data6;  }  private void button1\_Click(object sender, EventArgs e)  {  button1.Visible = false;  button2.Visible = true;  try  {  string connectionString = @"Data Source=sql;Initial Catalog=Коновалов\_СВ\_УП\_02;Integrated Security=True";  string gameInsertQuery = "INSERT INTO [Игра] ([game\_id], [название\_игры]) VALUES (@game\_id, @название\_игры)";  string salesInsertQuery = "INSERT INTO [Продажи] ([sale\_id], [game\_id], [дата\_продажи]) VALUES (@sale\_id, @game\_id, @дата\_продажи)";  using (SqlConnection connection = new SqlConnection(connectionString))  {  connection.Open();  // Добавляем игру в таблицу Игра  using (SqlCommand command = new SqlCommand(gameInsertQuery, connection))  {  int game\_id = Convert.ToInt32(textBox4.Text);  string название\_игры = Convert.ToString(textBox2.Text);  command.Parameters.AddWithValue("@game\_id", game\_id);  command.Parameters.AddWithValue("@название\_игры", название\_игры);  command.ExecuteNonQuery();  }  // Добавляем информацию о продаже в таблицу Продажи  using (SqlCommand command = new SqlCommand(salesInsertQuery, connection))  {  int sale\_id;  int game\_id = Convert.ToInt32(textBox4.Text);  DateTime дата\_продажи = DateTime.Now;  // Получаем максимальное значение sale\_id из таблицы Продажи и увеличиваем его на 1  string maxSaleIdQuery = "SELECT MAX(sale\_id) FROM Продажи";  using (SqlCommand maxSaleIdCommand = new SqlCommand(maxSaleIdQuery, connection))  {  object result = maxSaleIdCommand.ExecuteScalar();  if (result != DBNull.Value)  {  sale\_id = Convert.ToInt32(result) + 1;  }  else  {  sale\_id = 1; // Если таблица пуста, начинаем с 1  }  }  command.Parameters.AddWithValue("@sale\_id", sale\_id);  command.Parameters.AddWithValue("@game\_id", game\_id);  command.Parameters.AddWithValue("@дата\_продажи", дата\_продажи);  command.ExecuteNonQuery();  }  }  }  catch (Exception ex)  {  MessageBox.Show("Ошибка: " + ex.Message);  }  }  private void label10\_Click(object sender, EventArgs e)  {  this.Hide();  }  private void button2\_Click(object sender, EventArgs e)  {  this.Close();  библеотека f = new библеотека();  f.Show();  }  }  } |

Код формы «Библиотека» представлены в таблице 13.

Таблица 13 – Код формы «Библиотека»

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Data.SqlClient;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  namespace SG\_Продажа\_игр  {  public partial class Сведения : Form  {  public Сведения()  {  InitializeComponent();  }  private void textBox2\_TextChanged(object sender, EventArgs e)  {  SearchGames(textBox2.Text);  }  private void SearchGames(string gameName)  {  try  {  // Подключаемся к базе данных  string connectionString = @"Data Source=sql;Initial Catalog=Коновалов\_СВ\_УП\_02;Integrated Security=True";  using (SqlConnection connection = new SqlConnection(connectionString))  {  // Создаем SQL-запрос для поиска игр по названию  string query = "SELECT \* FROM Игры WHERE [название\_игры] LIKE @gameName";  // Создаем команду с параметрами  using (SqlCommand command = new SqlCommand(query, connection))  {  // Добавляем параметр для названия игры  command.Parameters.AddWithValue("@gameName", "%" + gameName + "%");  // Открываем соединение и выполняем команду  connection.Open();  SqlDataAdapter adapter = new SqlDataAdapter(command);  DataTable dataTable = new DataTable();  adapter.Fill(dataTable);  // Загружаем результаты поиска в DataGridView или другой контрол для отображения таблицы игр  dataGridView1.DataSource = dataTable;  }  }  }  catch (Exception ex)  {  MessageBox.Show("Ошибка при поиске игр: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  }  private void Сведения\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "коновалов\_СВ\_УП\_02DataSet.Продажи". При необходимости она может быть перемещена или удалена.  this.продажиTableAdapter.Fill(this.коновалов\_СВ\_УП\_02DataSet.Продажи);  }  SqlConnection con = new SqlConnection(@"Data Source=sql;Initial Catalog=Коновалов\_СВ\_УП\_02;Integrated Security=true");  private void pictureBox1\_Click(object sender, EventArgs e)  {  try  {  string commandText = "select \* from Продажи";  SqlDataAdapter itm = new SqlDataAdapter(commandText, con);  DataTable dt2 = new DataTable();  itm.Fill(dt2);  dataGridView1.DataSource = dt2;  con.Close();  }  catch  {  MessageBox.Show("Нечего обновлять");  }  }  private void pictureBox2\_Click(object sender, EventArgs e)  {  this.Close();  Админестратор frm2 = new Админестратор();  frm2.Show();  }  }  } |