ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»
(СПбГУТ)

АРХАНГЕЛЬСКИЙ КОЛЛЕДЖ ТЕЛЕКОММУНИКАЦИЙ
ИМ. Б.Л. РОЗИНГА (ФИЛИАЛ) СПбГУТ
(АКТ (ф) СПбГУТ)

КУРСОВОЙ ПРОЕКТ

НА ТЕМУ

РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

«НАСТОЛЬНЫЕ ИГРЫ»

Л109. 23КП01. 018 ПЗ

(Обозначение документа)

МДК.02.02 Технология разработки и защиты

баз данных

Студент	ПКС-01		12.05.2023	А.А. Добряков
	(Группа)	(Подпись)	(Дата)	(И.О. Фамилия)
Преподаватель			12.05.2023	Ю.С. Маломан
		(Подпись)	(Дата)	(И.О. Фамилия)

Архангельск 2023

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящем курсовом проекте применяют следующие сокращения и обозначения:

БД – база данных

ИС – информационная система

ОС – операционная система

ПО – программное обеспечение

СУБД – система управления базами данных

AI – автоматическая идентификация

EF – Entity Framework

ER-модель – модель «сущность-связь»

ERD – диаграмма «сущность-связь»

FK – внешний ключ

IDE – интегрированная среда разработки

LINQ – язык интегрированных запросов

NN – обязательное значение

ORM – объектно-реляционное отображение

PK – первичный ключ

SQL – язык структурированных запросов

UQ – уникальное значение

WPF – Windows Presentation Foundation

ВВЕДЕНИЕ

В настоящее время почти невозможно найти проект, работающий с большим количеством информации и не связанный с БД. БД применяются в финансовой и банковских сферах, в разных отраслях промышленности, маркетинге, многопользовательских играх и в мобильных приложениях.

Актуальность курсового проекта заключается в том, что применение БД является лучшим современным решением проблемы обработки большого количества данных. БД сильно упрощают многие действия, такие как: добавление, удаление, поиск и изменение данных. Для работы с данными и последующей корректной работы ИС раньше требовалось огромное количество времени и сложного мыслительного процесса. Теперь все процессы работы возможно упростить с помощью БД. Записи в хороших БД имеют быструю скорость обработки, не дублируют информацию и не имеют противоречивости. Дополнительные возможности разных СУБД позволяют создать триггеры, процедуры и представления, которые могут упростить и автоматизировать те процессы, которые раньше приходилось выполнять вручную.

Целью курсового проектирования является разработка многопользовательской клиент-серверной ИС для хранения, использования и редактирования информации о настольных играх, а также закрепить теоретические и практические знания и умения в проектирование БД.

Для достижения поставленной цели требуется решить следующие задачи:

- проанализировать особенности применения предметной области, относящейся к теме курсового проектирования,
- спроектировать концептуальную модель,
- спроектировать логическую модель,
- спроектировать физическую модель,
- разработать таблицы БД,

- разработать связи БД,
- заполнить таблицы БД,
- разработать представления,
- разработать функции,
- разработать процедуры,
- разработать триггеры,
- разработать приложение для доступа к БД,
- реализовать разграничение прав доступа пользователей,
- реализовать формирование отчётов,
- выполнить тестирование и отладку приложения,
- разработать руководство пользователя.

1 Анализ и разработка требований

1.1 Назначение и область применения

Разрабатываемая ИС предназначена для организации системы, направленной на представление информации и продажу настольных игр. Пользователями этой системой будут незарегистрированный пользователь, контент-менеджер и администратор системы.

1.2 Постановка задачи

Требуется разработать многопользовательскую клиент-серверную ИС с графическим интерфейсом для организации системы продаж, направленной на представление информации о настольных играх и дополнений, оформление заказов, изменение и добавление новых записей о настольных играх, дополнениях, категориях, экспорта чеков и отчётов.

Независимо от статуса пользователя должен осуществляться показ на первоначальной странице популярных игр (до десяти штук в категории), а также поиск настольных игр.

ИС должна обеспечивать выполнение следующих задач:

- авторизация под категориями: администратор системы, контент-менеджер,
- добавление и редактирование записей о товарах и дополнениях,
- добавление фотографий к информации о товарах, дополнениях,
- поиск настольных игр по различным критериям: по наименованию, по количеству игроков, по возрасту игроков, по категории и жанру,
- удаление, изменение и добавление категории,
- получение статистики о проданных играх за текущий месяц,
- получение чека при покупке игры.

1.3 Описание алгоритма функционирования системы

После запуска ИС перед пользователем отображается главная станица с десятью популярными играми, список категорий и жанров, параметры для изменения количества игроков и минимального возраста, кнопки для поиска и текстовое поле ввода для ввода части имени.

При нажатии на кнопку «Войти» предоставляется возможность авторизации под одной из категорий пользователей (контент-менеджер, администратор системы).

Незарегистрированному пользователю доступны следующие функции:

- просмотр популярных настольных игр и категорий,
- поиск игр по различным категориям и части имени,
- оформление покупки настольной игры,
- получение чека при покупке игры.

Контент-менеджеру доступны следующие функции:

- просмотр информации о дополнениях и настольных играх,
- добавление и изменение записей о настольных играх и дополнениях,
- прикрепление фотографий.

Администратору системы доступны следующие функции:

- просмотр информации о дополнениях и настольных играх,
- добавление, изменение и удаление категории,
- добавление и изменение записей о настольных играх и дополнениях,
- изменение настроек подключения,
- получение статистики о продажах за текущий месяц.

1.4 Выбор состава программных и технических средств

Согласно цели проекта требуется создать многопользовательскую клиент-серверную ИС для ведения учета настольных игр.

Работа с ИС будет осуществляться на персональных компьютерах с установленной ОС Windows версии не ниже Windows 10, объединенных в локальную сеть.

В качестве системы управления БД выбрана СУБД Microsoft SQL Server 2019, т. к. она является удобной в работе и позволяет быстро создать представления.

Приложение будет написано на языке программирования C#, т. к. он позволяет быстро написать приложение с подключённой к нему БД.

Для разработки приложения будет использоваться IDE Visual Studio 2022, т. к. данная среда имеет удобный интерфейс, средства отладки и разработки оконных приложений.

Для функционирования системы на стороне сервера достаточны следующие программные и технические средства:

- ОС Windows 10 / Windows 11 / Linux,
- сервер БД: Microsoft SQL Server версии не ниже 2019 года,
- программное обеспечение для конфигурирования, управления и администрирования сервера БД: SQL Server Management Studio Management Studio 19,
- 64-разрядный процессор с тактовой частотой 1.4 ГГц.

Для функционирования системы на стороне клиента достаточны следующие программные и технические средства:

- ОС Windows 10 / Windows 11,
- установленный .NET 6.

2 Разработка серверной части информационной системы

2.1 Проектирование базы данных

В БД требуется хранить информацию о настольных играх, дополнениях к ним, категориях и покупках.

У каждой настольной игры указывается наименование (уникальное), цена (от 0,00 до 35000,00) и минимальный возраст игроков. Также записывается минимальное количество игроков (по умолчанию – 1) и максимальное (если есть), среднее время игры, изображение, описание (если есть) и количество экземпляров на складе. Все числовые значение должны быть неотрицательными.

Каждое дополнение имеет такие же, как у игры, поля наименования, цены, возраста игроков, описания и изображения. Также должно быть количество экземпляров дополнений, присутствующих на складе, и столбец с идентификатором дополнения.

Любая категория имеет уникальное название, описание и столбец, указывающий является ли данная категория жанром.

В покупках должна храниться информация о идентификаторе покупки, дате и времени покупки (не позже текущих), фамилии, имени и отчестве (если есть), месте доставки, телефоне и о электронном адресе (если есть).

В каждой покупке может быть несколько игр и дополнений и каждая игра или дополнение может быть в нескольких заказах. Каждая настольная игра может иметь несколько категорий, и каждая категория может относиться к нескольким настольным играм. У одной настольной игры может быть несколько дополнений, но дополнение относится только к одной игре.

На рисунке 1 показана концептуальная модель предметной области в виде ERD [3].

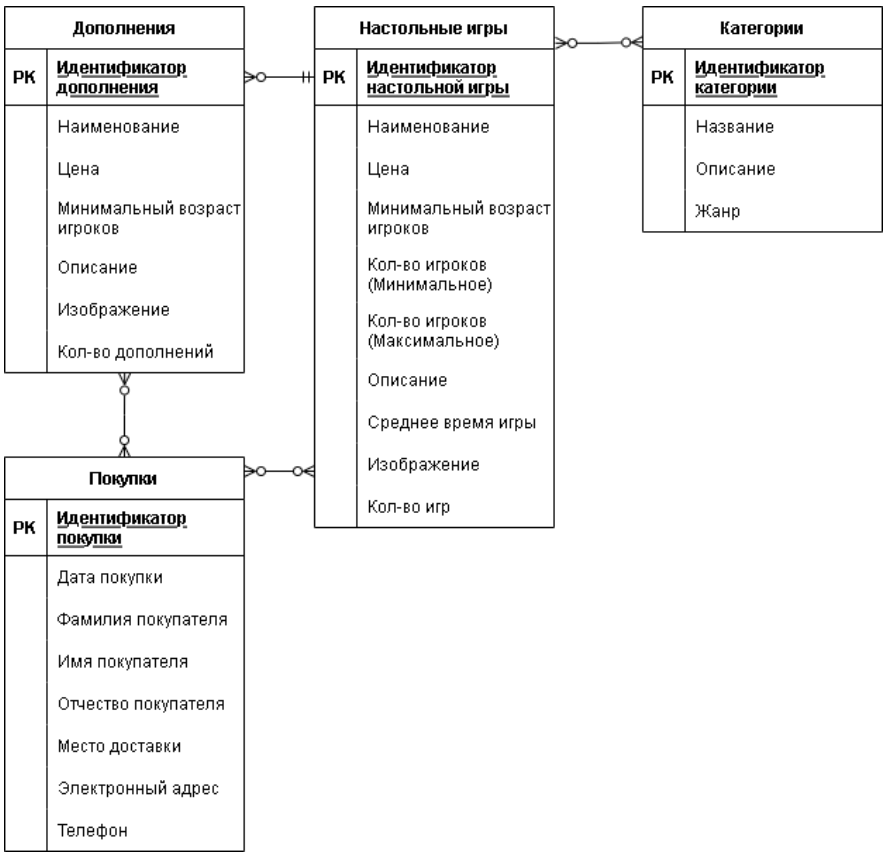

Рисунок 1 – Концептуальная модель

СУБД Microsoft SQL Server 2019, выбранная для хранения данных, является реляционной [1], [2], поэтому требуется преобразовать разработанную концептуальную модель в логическую с учетом правил преобразования ER-модели в реляционную модель данных:

- связь между сущностями «Дополнения» и «Настольные игры» M:1 и обязательна для сущности «Дополнения», поэтому формируется два отношения, соответствующие исходным сущностям, с внешним ключом в отношение «Дополнения»,

- связь между сущностями «Настольные игры» и «Категории» M:M, поэтому формируется три отношения, два соответствуют исходным сущностям, а третье — связующее с двумя внешними ключами,

- связь между сущностями «Настольные игры» и «Покупки» M:M, поэтому формируется три отношения, два соответствуют исходным сущностям, а третье — связующее с двумя внешними ключами,

- связь между сущностями «Дополнения» и «Покупки» M:M, поэтому формируется три отношения, два соответствуют исходным сущностям, а третье — связующее с двумя внешними ключами.

На рисунке 2 показана логическая модель предметной области, полученная путем преобразования концептуальной модели.

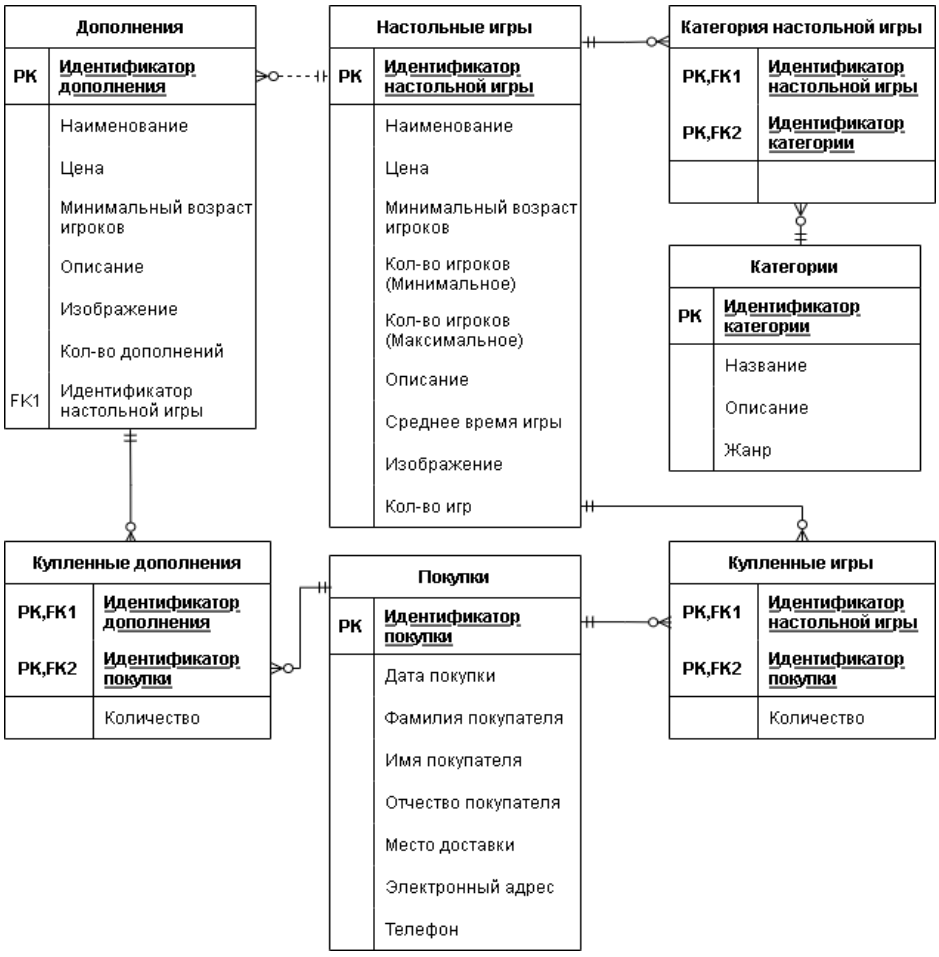

Рисунок 2 – Логическая модель

На рисунке 3 показана физическая модель для ИС «Настольные игры», разработанная на основе логической модели для СУБД Microsoft SQL Server 2019.

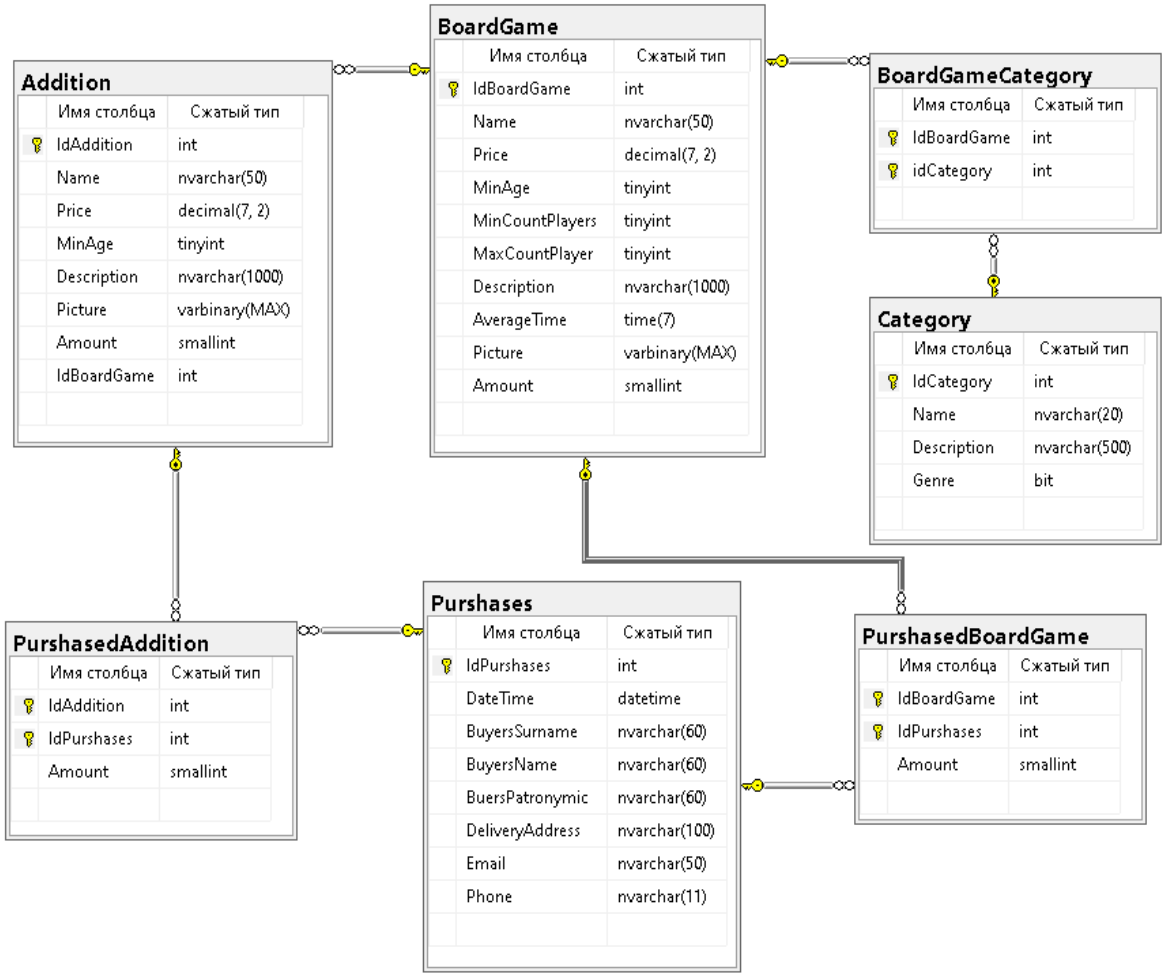

Рисунок 3 – Физическая модель

2.2 Создание объектов базы данных

В таблице 1 в виде словаря данных представлено описание созданных в СУБД Microsoft SQL Server 2019 таблиц и ограничений целостности БД.

Таблица 1 – Словарь данных

Ключ	Поле	Тип данных	NN	Примечание
BoardGame				
PK	IdBoardGame	int	+	AI, UQ
	Name	narchar(50)	+	UQ
	MinAge	tinyint	+	По умолчанию: 0

Продолжение таблицы 1

Ключ	Поле	Тип данных	NN	Примечание
	Price	decimal(7,2)	+	По умолчанию: 0.00, >=0.00 и <=35000.00
	MinCountPlayers	tinyint	+	По умолчанию:1
	MaxCountPlayers	tinyint	-	
	Description	nvarchar(1000)	-	
	AverageTime	time(7)	+	
	Picture	varbinary(Max)	-	
	Amount	smallint	+	По умолчанию: 0, >=0
Category				
PK	IdCategory	int	+	AI, UQ
	Name	nvarchar(20)	+	UQ
	Description	nvarchar(500)	-	
	Genre	bit	+	По умолчанию: 0
BoardGameCategory				
PK, FK1	IdBoardGame	int	+	Совместно UQ с idCategory
PK, FK2	idCategory	int	+	Совместно UQ с IdBoardGame
Addition				
PK	IdAddition	int	+	AI, UQ
	Name	nvarchar(50)	+	UQ
	Price	decimal(7,2)	+	По умолчанию: 0.00, >=0.00 и <=35000.00
	MinAge	tinyint	+	По умолчанию: 0
	Description	nvarchar(1000)	-	
	Picture	varbinary(Max)	-	
	Amount	smallint		По умолчанию: 0, >=0
FK1	IdBoardGame	int	+	
Purshases				
PK	IdPurshases	int	+	AI, UQ
	DateTime	datetime	+	По умолчанию: GETDATE() , <=GETDATE()

Продолжение таблицы 1

Ключ	Поле	Тип данных	NN	Примечание
	BuyersSurname	nvarchar(60)	+	
	BuyersName	nvarchar(60)	+	
	BuersPatronymic	nvarchar(60)	-	
	DeliveryAddress	nvarchar(100)	+	
	Email	nvarchar(50)	-	
	Phone	nvarchar(11)	+	
PurshasedBoardGame				
PK, FK1	IdBoardGame	int	+	Совместно UQ с IdPurshases
PK, FK2	IdPurshases	int	+	Совместно UQ с IdBoardGame
	Amount	smallint	+	По умолчанию: 1, >=1
PurshasedAddition				
PK, FK1	IdAddition	int	+	Совместно UQ с IdPurshases
PK, FK2	IdPurshases	int	+	Совместно UQ с IdAddition
	Amount	smallint	+	По умолчанию: 1, >=1

Для решения поставленных задач в БД созданы представления:

- представление MonthOrders, возвращающее информацию о всех заказах за текущий месяц с датой, полным именем заказчика, суммой покупки за настольные игры, дополнения и общую сумму (код создания представления показан в листинге 1),

- представление Top10BoardGame, возвращающее десять самых продаваемых игр.

Листинг 1 – Код создания представления MonthOrders

```
CREATE VIEW MonthOrders -- код создания представления MonthOrders
AS
SELECT
     Purshases.IdPurshases, -- Индефикатор заказа
```

```
     Purshases.DateTime, -- Дата заказа
     CONCAT_WS(' ', Purshases.BuyersName,
           Purshases.BuyersSurname, Purshases.BuyersPatronymic)
          AS Buyer, -- Полное имя заказчика
     ISNULL(SUM(BoardGame.Price*PurshasedBoardGame.Amount),0)
          AS BoardGamesPrice, -- Сумма покупки настольных игр
     ISNULL(SUM(Addition.Price*PurshasedAddition.Amount),0)
          AS AdditionGamesPrice, -- Сумма покупки дополнений
     ISNULL(SUM(BoardGame.Price*PurshasedBoardGame.Amount),0)
          +ISNULL(SUM(Addition.Price * PurshasedAddition.Amount
          ),0) AS OrderPrice -- Общая сумма покупки
FROM BoardGame RIGHT JOIN
     PurshasedBoardGame ON BoardGame.IdBoardGame =
         PurshasedBoardGame.IdBoardGame RIGHT JOIN
     Purshases ON PurshasedBoardGame.IdPurshases =
         Purshases.IdPurshases LEFT JOIN
     PurshasedAddition ON Purshases.IdPurshases =
         PurshasedAddition.IdPurshases LEFT JOIN
     Addition ON PurshasedAddition.IdAddition =
         Addition.IdAddition
-- Проверка, что месяц и год заказа совпадает с текущими
WHERE MONTH(Purshases.DateTime)=MONTH(GETDATE())AND
YEAR(Purshases.DateTime)=YEAR(GETDATE())
GROUP BY Purshases.IdPurshases, Purshases.DateTime,
    Purshases.BuyersName, Purshases.BuyersSurname,
    Purshases.BuyersPatronymic
```

Для решения поставленных задач в БД созданы функции пользователя:

- скалярная функция CountAdditions, возвращающая количество дополнений на основе параметра имени настольной игр (код создания функции представлен листингом 2),

- табличная функция GetBoardGameByPrice, возвращающая таблицу отсортированных по цене настольных игр на основе указанного ценового диапазона.

Листинг 2 – Код создания функции пользователя CountAdditions

```
-- код создания функции пользователя CountAdditions
CREATE FUNCTION CountAdditions
(
     -- Вводимый параметр — имя настольной игры
     @NameBoardGame nvarchar(50)
)
RETURNS INT -- Возвращаемое значение INT
```

```
AS BEGIN
     -- Создание переменной кол-ва дополнений
     DECLARE @count INT
     -- Получение кол-ва дополнений
     SELECT @count = Count(*)
     FROM BoardGame JOIN
           Addition ON BoardGame.IdBoardGame =
               Addition.IdBoardGame
     WHERE BoardGame.Name = @NameBoardGame
     RETURN @count -- Возвращение кол-ва
END
```

Для решения поставленных задач в БД созданы хранимые процедуры:

- хранимая процедура AddCategory, добавляющая категорию AddCategory на основе заданного имени, описания и указание, является ли категория жанром, и возвращает идентификатор новой категории (код создания хранимой процедуры представлен листингом 3),

- хранимая процедура ChangePriceBoardGameAndHisAddition, изменяющая цену на настольную игру и все её дополнений по идентификатору настольной игры и изменению цены (в рублях) и возвращающая количество затронутого товара.

Листинг 3 – Код создания хранимой процедуры AddCategory

```
-- код создания хранимой процедуры AddCategory
CREATE PROCEDURE dbo.AddCategory
   @Name nvarchar(20), -- Название категории
   -- Описание, по умолчанию - Null
   @Decription nvarchar(500) = null,
   @Genre bit = 0, -- Битовое поле Жанр, по умолчанию - false
   -- Выходной параметр — полученный ID новой категории
   @Id INT OUTPUT
AS
BEGIN
     -- Добавление в таблицу новой категории
     -- с указанным названием, описанием и битовым полем
     INSERT INTO Category(Name,Description,Genre)
     VALUES(@Name, @Decription, @Genre);
     -- Присвоение выходному параметру значения
     SELECT @Id = SCOPE_IDENTITY()
END
```

Для решения поставленных задач в БД созданы следующие триггеры:

- триггер trChangeBoardGameCount, выполняющийся перед операцией INSERT в таблице PurshasedBoardGame и проверяющий, есть ли у настольной игры приобретаемое количество экземпляров (код создания триггера представлен листингом 4),

- триггер trDeleteCategory, выполняющийся перед операцией DELETE в таблице Category, который удаляет сначала связанные с категорией записи в таблице BoardGameCategory, а затем и в таблице Category (код создания триггера представлен листингом 5).

Листинг 4 – Код создания триггера trChangeBoardGameCount

```
-- код создания триггера trChangeBoardGameCount
CREATE TRIGGER trChangeBoardGameCount
ON PurshasedBoardGame
-- Триггер срабатывает перед добавлением записи
INSTEAD OF INSERT
AS
BEGIN
DECLARE @count int; -- Поле кол-ва покупаемых игр
DECLARE @amount int; -- Поле кол-ва игр в наличии
-- Заполнение полей значениями из таблиц
SELECT @count = PurshasedBoardGame.Amount,
        @amount = BoardGame.Amount
FROM PurshasedBoardGame JOIN
      BoardGame ON PurshasedBoardGame.IdBoardGame =
          BoardGame.IdBoardGame
-- Проверка на наличие нужного кол-ва игр
IF(@count<=@amount) BEGIN
     -- При успешной проверки происходит добавление
     -- в таблицу PurshasedBoardGame
     INSERT INTO PurshasedBoardGame(IdBoardGame, IdPurshases,
          Amount)
     SELECT IdBoardGame, IdPurshases, Amount
     FROM inserted;
     UPDATE BoardGame --Уменьшение кол-ва экземпляров на складе
     SET BoardGame.Amount -= inserted.Amount
     FROM BoardGame JOIN
          inserted ON BoardGame.IdBoardGame =
          inserted.IdBoardGame;
     END
ELSE -- Иначе вызывать исключение
     RAISERROR('Нет такого кол-ва',18,1)
END;
```

Листинг 5 – Код создания триггера trDeleteCategory

```
-- код создания триггера trDeleteCategory
CREATE TRIGGER [dbo].[trDeleteCategory]
ON [dbo].[Category]
-- Триггер срабатывает перед удалением записи
INSTEAD OF DELETE
AS
-- Удаление связанных записей в таблице BoardGameCategory
DELETE
FROM BoardGameCategory
WHERE BoardGameCategory.idCategory IN (SELECT idCategory
                                       FROM deleted)
-- Удаление категории
DELETE
FROM Category
WHERE Category.idCategory IN (SELECT idCategory
                                     FROM deleted)
```

3 Разработка клиентской части информационной системы

3.1 Разработка приложения для доступа к базе данных

Разработано оконное приложение WPF [5], т. к. к проекту WPF легко подключить EF Core, а технология WPF позволяет создавать приятный интерфейс. Для общего вида элементов приложения в документе App.xaml прописаны стили для всего приложения.

При создании приложения использована технология ORM, а именно модуль EF Core. Подключение к БД происходит в файле BoardGameContext.cs в методе OnConfiguring. Данные подключения к БД хранятся в настройках проекта. Если приложение не может подключиться к БД, то вызывается диалоговое окно настроек. Смена настроек подключения происходит в обработчике события ConnectButton_Click (листинг 6).

Листинг 6 – Код обработчика события ConnectButton_Click

```
//Смена имени сервера
Properties.Settings.Default.ServerName =
     ServerNameTextBox.Text;
//Смена имени БД
Properties.Settings.Default.BDName = BDNameTextBox.Text;
//Смена логина подключения
Properties.Settings.Default.BDUserLogin =
     BdUserNameTextBox.Text;
//Смена пароля подключения
Properties.Settings.Default.Password = PasswordTextBox.Text;
Properties.Settings.Default.Save(); //Сохранение настроек
try  //Проверка на возможность работы с БД
{
   using (var context = new BoardGameContext())
   {
       string qwery = "SELECT BoardGame.IdBoardGame,
           BoardGame.Name, BoardGame.Price, BoardGame.MinAge,
           BoardGame.MinCountPlayers,BoardGame.MaxCountPlayer,
           BoardGame.Description, BoardGame.AverageTime,
           BoardGame.Picture, BoardGame.Amount FROM BoardGame
           RIGHT JOIN Top10BoardGame ON BoardGame.Name =
```

```
            Top10BoardGame.Name";
        var TopBoardGame =
            context.BoardGames.FromSqlRaw(qwery).ToList();
    }
    this.DialogResult = true;//Изменение диалогового результата
    Close(); //Закрытие диалогового окна
}
//Если возникли проблемы с работой БД
catch (Exception ex)
{
    //Вывод сообщение о неудачном подключение
    MessageBox.Show($"Подключение не удалось\n{ex.Message}",
        "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);
}
```

Получение десяти самых популярных игр на главной странице осуществляется с помощью представления Top10BoardGame, вызываемого с помощью метода FromSqlRaw.

Фильтрация игр на странице происходит с использованием LINQ-запросов, которые показаны листингом 7.

Листинг 7 – Код LINQ-запросов фильтрации на главной странице

```
//Получение введённого пользователем части имени игры
string name = NameTextBox.Text.Trim();
//Получение игр с именем, где есть введённая часть
var games = context.BoardGames.Where(bg =>
    bg.Name.Contains(name));
if(age != -1) //Проверка указан ли пользователем возраст
{
  //Если возраст указан то происходит сортировка игр по
  //минимальному возрасту
  games = games.Where(g=>g.MinAge>=age);
}
if (count != 0) //Проверка указано ли количество
{
  //Если количество указано, то происходит сортировка по
  //значению так, чтобы количество игроков попадало в
  //диапазон возможного количества для игры
  games = games.Where(g => g.MinCountPlayers <= count &&
    g.MaxCountPlayer>=count);
}
if(categories.Count != 0) //Проверка указана ли категория
{
  int currect = 0;//Кол-во совпавших категорий
  //Получение списка игр с категориями
```

```
  var catgames = games.Include(c => c.IdCategories).ToList();
  //Создание списка для будущих добавленных игр на главную
  //страницу
  List<BoardGame> bg = new List<BoardGame>();
  foreach (var g in catgames) //Перебор всех игр
  {
     //Перебор всех категорий игры
     foreach (var c in g.IdCategories)
     {
        //Перебор списка указанных пользователем категорий
        foreach(var item in categories)
        {
           //Если категория из списка указанных и категория
           //игры совпадают то число совпавших категорий
           //увеличивается на единицу
           if (c == context.Categories.FirstOrDefault(ca =>
               ca.IdCategory == (int)item))
           {
               currect++;
           }
        }
     }
     //Если число совпавших категорий равно числу указанных то
     //игра добавляеться в список
     if(currect == categories.Count())
     {
        bg.Add(g);
     }
     currect = 0; //Число совпавших категорий отчищается
  }
  AddGame(bg); //Добавление списка на главную страницу
  return;//Выход из метода, чтобы не повторять игры
}
//Если категории не указаны, то производится добавление игр
AddGame(games.ToList());
```

3.2 Разграничение прав доступа пользователей

Для разграничение прав доступа созданы таблицы User и RoleUser. В таблице User три столбца: Login (PK), Password и IdRoleUser (FK). Таблица RoleUser имеет два столбца: IdRoleUser (PK) и NameRoleUser.

Роль пользователя, успешно авторизованного в приложении, записывается в свойство UserRole класса CurrentUser. При загрузке страницы для управления объектами БД происходит проверка имени роли. Листингом 8 представлен код настройки интерфейса для роли «Контент-менеджер».

Листинг 8 – Код настройки интерфейса для роли «Контент-менеджер»

```
//Проверка является роль - "Контент-менеджер"
if (CurrentUser.UserRole == "Контент-менеджер")
{
   //Если роль — "Контент-менеджер" то TabItem с
   //редактированием настольных игр виден
   ProductPage.Visibility = Visibility.Visible;
   //TabItem с редактированием дополнений виден
   AdditionPage.Visibility = Visibility.Visible;
   //TabItem с настройками не виден
   SetterPage.Visibility = Visibility.Hidden;
   //TabItem со статистикай не виден
   StatisticPage.Visibility = Visibility.Hidden;
   //TabItem с редактированием категорий не виден
   CategoryPage.Visibility = Visibility.Hidden;
}
```

3.3 Разработка и экспорт отчетов

В клиентском приложение возможно создавать два типа документа: чек заказа (PDF-документ) и статистику заказов по дням текущего месяца (HTML-документ).

Чек заказа создаётся при нажатии на кнопку с подписью «Создать PDF-документ чека» на странице, появляющейся после оформления заказа. Для создания документа использована .NET библиотека PDFsharp.

Статистику заказов по дням текущего месяца можно получить во вкладке «Статистика», которая является доступной для пользователя с ролью «Администратор». Для создания HTML-документа с таблицей требуется нажать на кнопку в правом нижнем углу вкладки. Код обработчика события нажатия представлен в листинге 9 [4].

Листинг 9 – Код обработчика события HTMLButton_Click

```
private void HTMLButton_Click(object sender, RoutedEventArgs e)
{
    SaveFileDialog dialog = new SaveFileDialog();
    //Установка фильтра для диалогового окна
```

```
    dialog.Filter = "Документ HTML|*.html";
    //Проверка выбран ли файн
    if (dialog.ShowDialog() != true)
    {
        return;
    }
    using (var context = new OrderContext())
    {
        //Получение заказов за текущий месяц
        var orders = context.MonthOrders;
        //Добавление в файл строки с заголовком, в которой
        //указывается текущий месяц и год
        File.AppendAllText(dialog.FileName, $"<h2>Месяц:
             {DateTime.Now:Y}</h2>");
        //Добавление в файл открывающего тега таблицы и
        //заголовка
        File.AppendAllText(dialog.FileName, "<Table border='1'
             cellspacing='0'><tr bgcolor='#7B2525'
             align='center' style='color: white;'>
             <th>День</th><th>Кол-возаказов</th> <th> Итог по
             настольным играм</th><th>Итог по дополнениям</th>
             <th>Общий итог</th></tr>");
        //Получение самой поздней даты среди заказов
        var lastDay = context.MonthOrders.Max(o => o.DateTime);
        string text; //Объявление строковой переменной
        //Цикл для перечисления всех дней до последнего
        for (int i = 1; i <= lastDay.Day; i++)
        {
            //Получение списка заказов сделанных в указанный
            //день
            var ordersDay = context.MonthOrders.Where(o =>
                o.DateTime.Day == i);
            //Проверка есть ли в списке значения
            if (ordersDay != null)
            {
                //Если хотя бы один заказ был сделан в
                //указанный день, то в переменную записываются
                //теги для строки таблицы со значениями
                text = $"<tr><td>{i}</td><td>
                    {ordersDay.Count()} </td><td>
                    {ordersDay.Sum(o => o.BoardGamesPrice)}
                    </td><td>{ordersDay.Sum(o =>
                    o.AdditionGamesPrice)}</td><td>
                    {ordersDay.Sum(o => o.OrderPrice)}</td>
                    </tr>";
            }
            else
            {
                //Если заказов в указанный день не найдено, то
                //в переменную записывается строка с пустыми
                //значениями
                text = $"<tr><td>{i}</td><td>0</td><td>0,00
                    </td><td>0,00</td><td>0,00</td></tr>";
```

```
            }
            //Запись в файл переменной
            File.AppendAllText(dialog.FileName, text);
        }
        //Запись в файл закрывающего тега таблицы
        File.AppendAllText(dialog.FileName, "</Table>");
        //Вывод сообщения о успешном создание файла
        MessageBox.Show("Файл статистики успешно создан",
            "Оповещение", MessageBoxButton.OK,
            MessageBoxImage.Information);
    }
}
```

3.4 Тестирование разработанной системы

Для тестирования выбран метод чёрного ящика. Тестирование поделено на тестирование функциональности незарегистрированного пользователя и зарегистрированного. В таблице 2 представлен набор тестов для фильтрации, поиска и отображения данных для незарегистрированного пользователя.

Таблица 2 – Набор тестов функциональности незарегистрированного пользователя

Действие	Ожидаемый результат	Полученный результат
Ввод в поле поиск «ко», нажатие кнопки «Найти >»	Вывод настольных игр с «ко» в имени без учёта регистра	Вывод настольных игр с «ко» с учётом регистра
Выбор категории «Карточные игры», указание возраста игроков — 12, указание количества игроков — 4	Вывод настольных игр, удовлетворяющих выбранные настройки	Совпадает с ожидаемым
Нажатие у настольной игры «Битва за Рокуган» кнопки «Узнать больше»	Переход на страницу с информацией о настольной игре «Битва за Рокуган»	Совпадает с ожидаемым
Нажатие у настольной игры «Бэнг!» кнопки «Узнать больше», нажатие у дополнения «Бэнг! Долина теней» кнопки «Узнать больше»	Переход на страницу с информацией о дополнении «Бэнг! Долина теней»	Совпадает с ожидаемым

В таблице 3 представлен набор тестов для страницы входа с разграничением прав доступа.

Таблица 3 – Набор тестов для страницы входа

Действие	Ожидаемый результат	Полученный результат
Нажатие кнопки «Вход», нажатие кнопки «Войти»	Вывод сообщения «Логин или пароль неверен»	Совпадает с ожидаемым
Нажатие кнопки «Вход», ввод в поле логин «FVST», в поле пароль «Dtkbrfz7Djqyf0»,нажатие кнопки «Войти»	Вывод сообщения «Успешный вход», переход на страницу с вкладками: «Изменение настольных игр», «Изменение дополнений», «Изменение категорий», «Статистика», «Настройки»	Совпадает с ожидаемым
Нажатие кнопки «Вход», ввод в поле логин «Teaca», в поле пароль «q2m1t», нажатие кнопки «Войти»	Вывод сообщения «Успешный вход», переход на страницу с вкладками: «Изменение настольных игр», «Изменение дополнений»,	Совпадает с ожидаемым

В таблице 4 представлен набор тестов для модификации настроек подключения и экспорта HTML-отчётов для зарегистрированного пользователя с ролью «Администратор».

Таблица 4 – Набор тестов функциональности зарегистрированного пользователя

Действие	Ожидаемый результат	Полученный результат
Нажатие кнопки «Вход», ввод в поле логин «FVST», в поле пароль «Dtkbrfz7Djqyf0», нажатие кнопки «Войти», переход на вкладку «Настройки», нажатие на кнопку «Открыть окно настроек», изменить «Имя пользователя БД» на «FVST»	Вывод сообщения «Подключение установлено», закрытие окна настроек	Совпадает с ожидаемым
Нажатие кнопки «Вход», ввод в поле логин «FVST», в поле пароль «Dtkbrfz7Djqyf0», нажатие кнопки «Войти», переход на вкладку «Настройки», нажатие на кнопку «Открыть окно настроек», изменить «Имя пользователя БД» на «FVSTв»	Вывод сообщения «Подключение не удалось (ошибка)»	Совпадает с ожидаемым
Нажатие кнопки «Вход», ввод в поле логин «FVST», в поле пароль «Dtkbrfz7Djqyf0», нажатие кнопки «Войти», переход на вкладку «Статистика», нажатие на кнопку «Скачать краткую статистику за месяц в HTML», указать имя файла «1»	Вывод сообщения «Файл статистики успешно создан», создание файла 1.html со статистикой	Совпадает с ожидаемым

При тестировании функциональности незарегистрированного пользователя выявлена ошибка с поиском настольных игр по введённой пользователем части: происходит регистрозависимый поиск по именам настольных игр вместо нерегистрозависимого. Листингом 10 показан код, вызвавший ошибочное срабатывание приложения.

Листинг 10 – Первоначальный код

```
//Получение введённого пользователем части имени игры
string name = NameTextBox.Text.Trim();
//Получение игр с именем, где есть введённая часть
var games = context.BoardGames.Where(bg =>
    bg.Name.Contains(NameTextBox.Text));
```

Для того, чтобы программа работала корректно, требуется привести и введённую пользователем часть имени и имя настольной игры к одному регистру. В данном случае решено использовать нижний регистр. Исправленный код представлен листингом 11.

Листинг 11 – Исправленный код

```
//Получение введённого пользователем части имени игры в
//нижнем регистре
string name = NameTextBox.Text.Trim().ToLower();
//Получение игр с именем в нижнем регистре, где есть введённая
//часть
var games = context.BoardGames.Where(bg =>
    bg.Name.ToLower().Contains(name));
```

После тестирования выявлено, что разработанное приложение удовлетворяет следующие критерии качества ПО:

- эргономичность.
- эффективность,
- функциональность,
- надёжность.

4 Руководство пользователя

4.1 Установка приложения

Для установки на стороне сервера требуется Microsoft SQL Server версии не ниже 2019 года и программное обеспечение SQL Server Management Studio Management Studio 19. Для корректной установки требуется:

- более 6 ГБ дискового пространства,
- 1 ГБ оперативной памяти,
- 64-разрядный процессор с тактовой частотой 1.4 ГГц.

В установленном сервере создаётся БД и пользователь, который может подключится к ней. Для создания объектов требуется использовать скрипты. Сначала запускается скрипт для создания таблиц и записей в этих таблицах (ScriptTables.sql), а затем скрипт для представлений (Views.sql). Далее запускаются скрипты для функций (Functions.sql) и процедур (ProcedureScript.sql).

На стороне клиента требуется скачать папку приложения, в которой находится GameStarts.exe. Для запуска нужно дважды нажать на указанный ранее файл.

По умолчанию в БД находится пользователь с ролью администратор. Логин пользователя — FVST, а пароль — Dtkbrfz7Djqyf0.

4.2 Инструкция по работе

При запуске приложения незарегистрированному пользователю показывается главная страница приложения, где возможны поиск и сортировка настольных игр. Вид стартовой страницы показан на рисунке 4.

При нажатии на кнопку «Узнать больше» пользователь попадает на страницу с информацией о настольной игре (представлена на рисунке 5) с которой можно перейти на страницу дополнения или добавить игру в корзину.

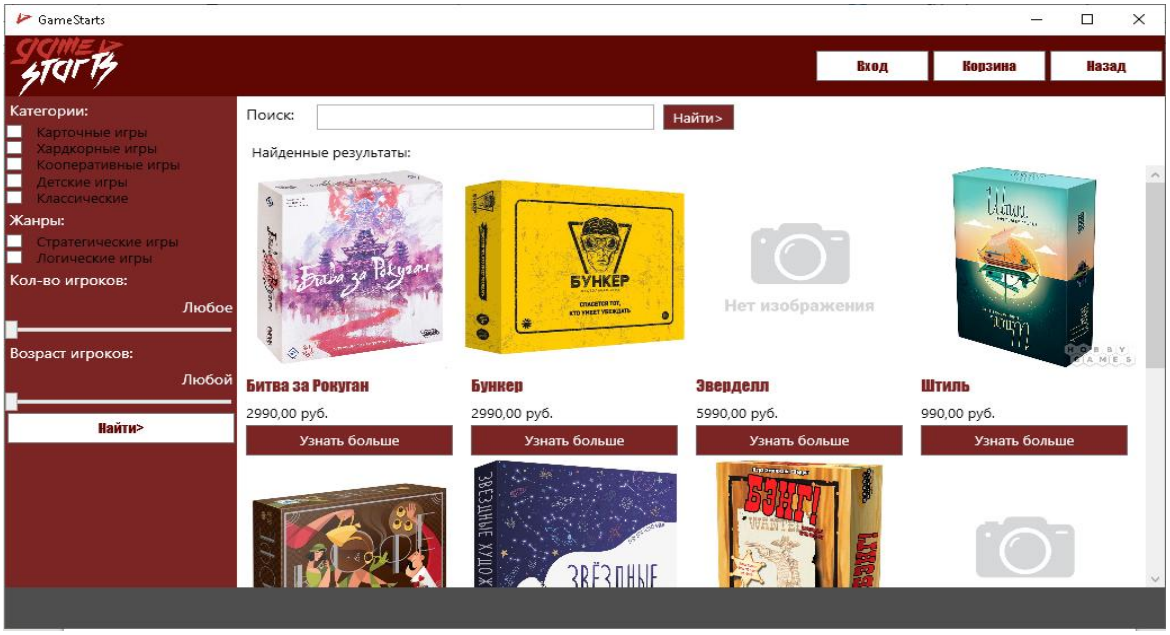

Рисунок 4 – GameStarts. Вид стартовой страница

Рисунок 5 – GameStarts. Вид страницы с информацией о настольной игре

При нажатии на кнопку «В корзину» выводится сообщение, что игра добавлена в корзину, или же, если настольная игра уже есть в корзине, сообщение о ошибке (рисунок 6).

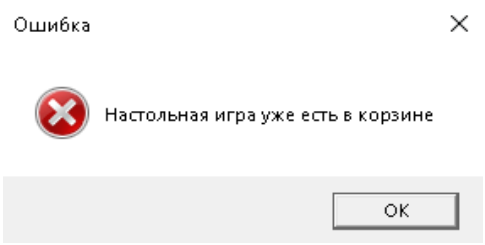

Рисунок 6 – GameStarts. Вид сообщения об ошибке

При нажатии на кнопку «Корзина» пользователь перенаправляется на страницу корзины, где можно изменить количество приобретаемых экземпляров или удалить игру из корзины. При удалении всплывает диалоговое окно с вопросом, действительно ли пользователь хочет удалить настольную игру из корзины. Диалоговое окно показано на рисунке 7. При нажатии на кнопку «Купить» пользователя переносит на страницу для указания информации о покупателе (рисунок 8). Если пользователь вводит некорректную информацию или не заполняет обязательные поля, то выводится сообщение с перечислением всех допущенных ошибок. На рисунке 9 показано сообщение с перечислением ошибок.

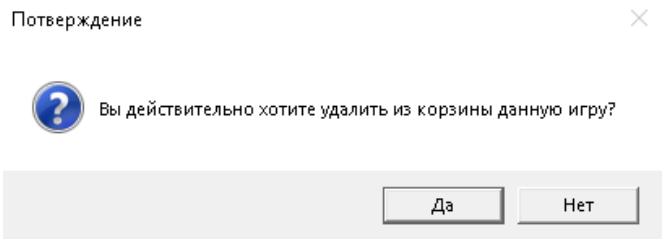

Рисунок 7 – GameStarts. Вид диалоговое окно подтверждения

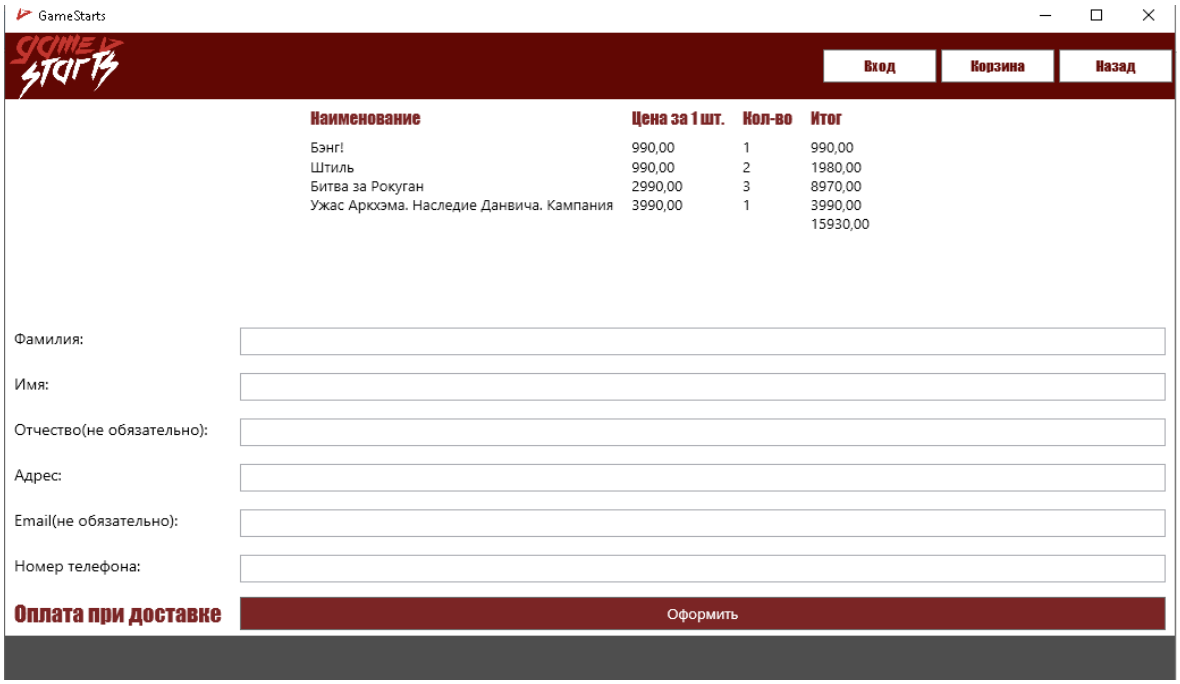

Рисунок 8 – GameStarts. Вид страницы для указания информации покупателя

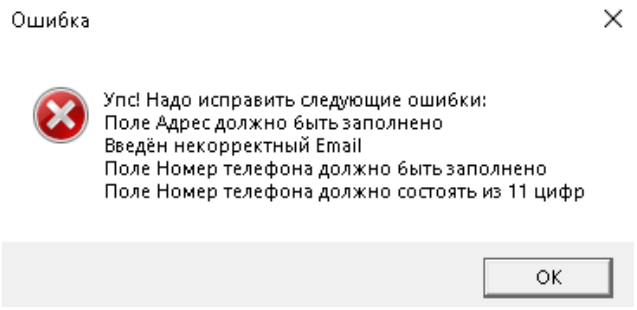

Рисунок 9 – GameStarts. Вид диалогового окна подтверждения

После исправления ошибок и ввода корректной информации пользователь может оформить заказ. Перед оформлением выводится диалоговое окно, показанное на рисунке 10.

На странице, на которую перенаправляется пользователь после оформления заказа, возможно сохранение чека в виде PDF-документа. Пример созданного PDF-документа представлен на рисунке 11.

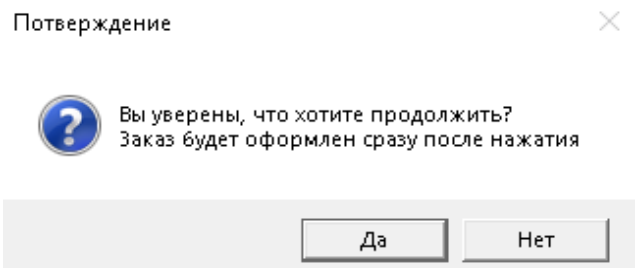

Рисунок 10 – GameStarts. Вид диалогового окна подтверждения

Рисунок 11 – GameStarts. Вид PDF-документ чека

Для получения функциональности зарегистрированного пользователя, нужно авторизоваться.

Пользователям с ролью «Администратор» открыты все вкладки на странице действий. Страница показана на рисунке 12.

Пользователям с ролью «Контент-менеджер» доступны только две вкладки на странице. Страница с открытыми вкладками показана на рисунке 13.

Во вкладках «Изменение настольных игр», «Изменение дополнений» и «Изменение категорий» возможно добавление или изменение настольных игр, дополнений и категорий соответственно. Во вкладке «Статистика»,

показанной на рисунке 14, можно создать HTML-документ статистики по дням текущего месяца.

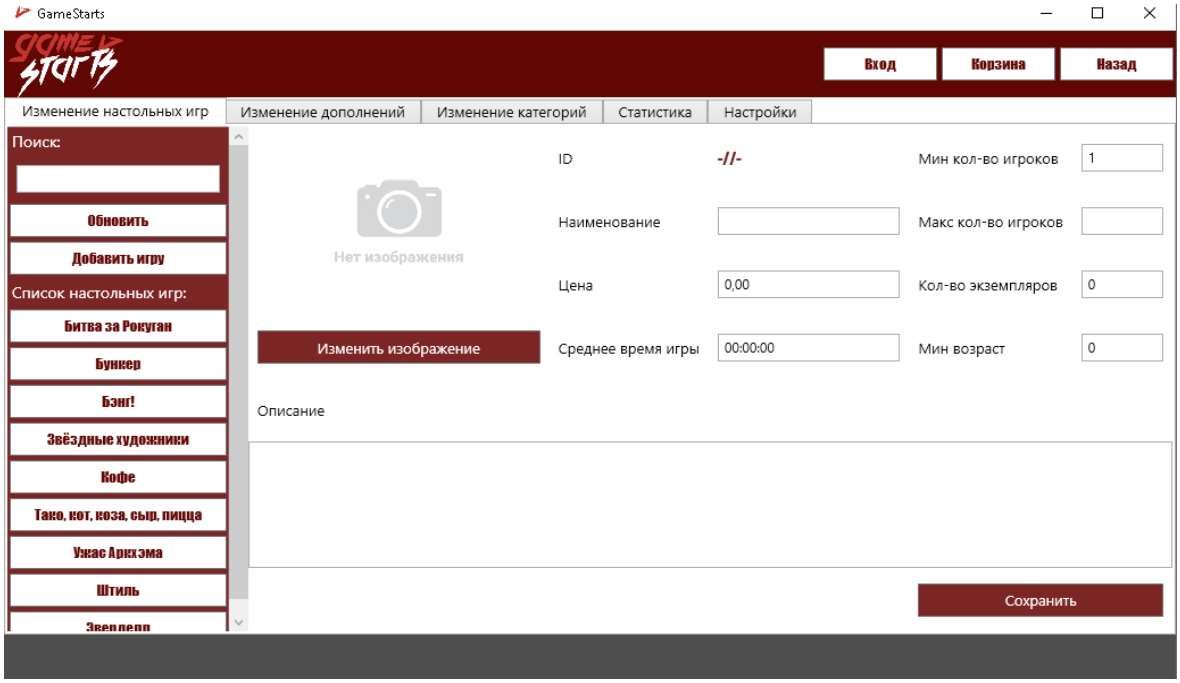

Рисунок 12 – GameStarts. Вид страницы для роли «Администратор»

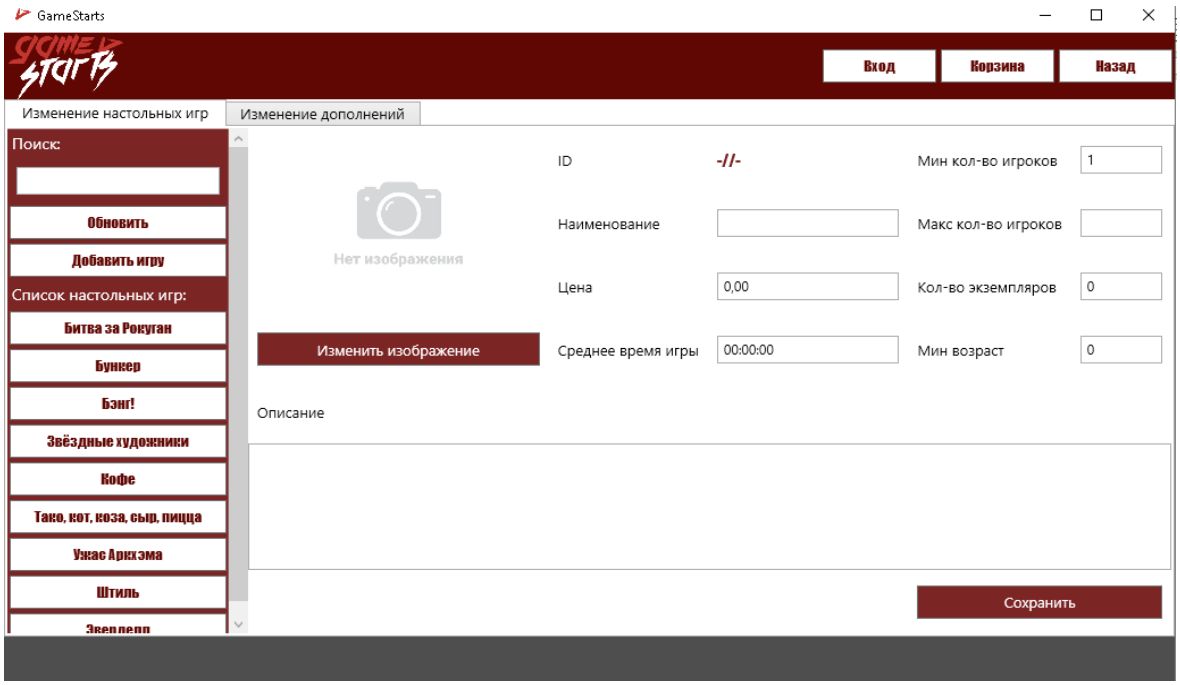

Рисунок 13 – GameStarts. Вид страницы для роли «Контент-менеджер»

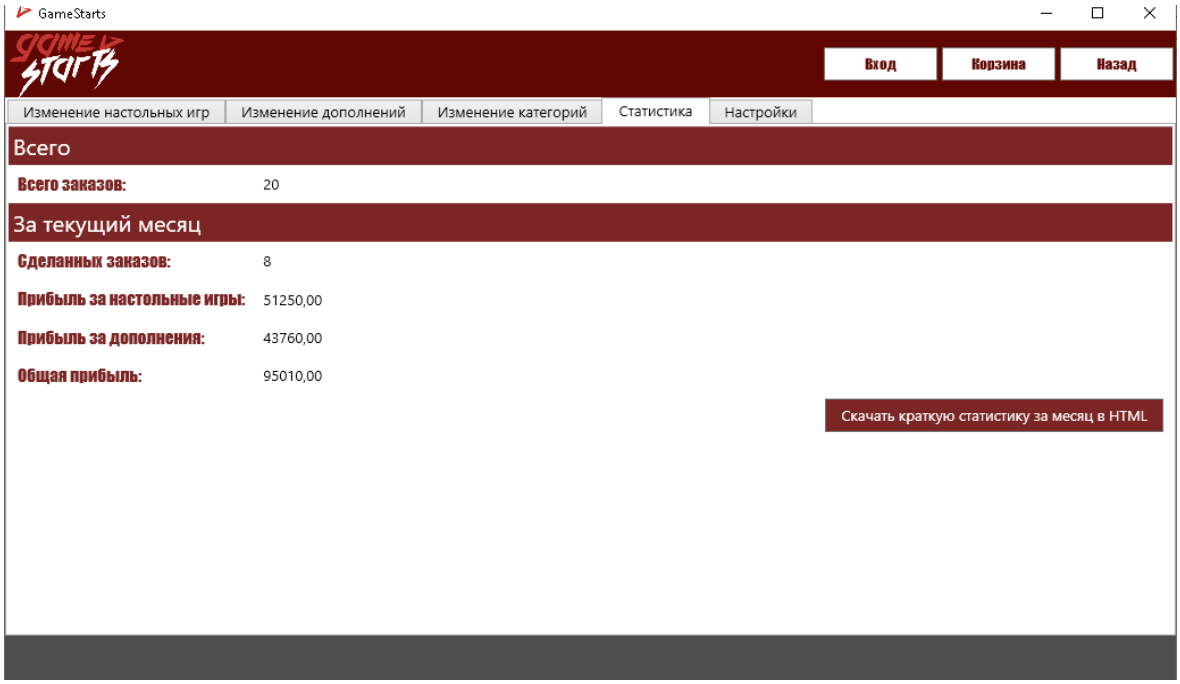

Рисунок 14 – GameStarts. Вид вкладки «Статистика»

Во вкладке «Настройки» можно вызвать диалоговое окно настроек, показанное на рисунке 15.

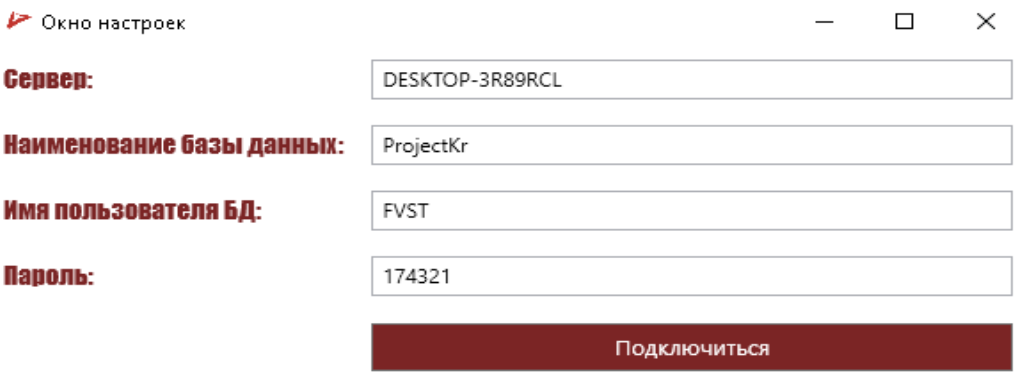

Рисунок 15 – GameStarts. Вид диалогового окна настроек

ЗАКЛЮЧЕНИЕ

В результате курсового проектирования разработана многопользовательская клиент-серверная информационная система для хранения, использования и редактирования информации о настольных играх.

В процессе достижения цели выполнены задачи:

- проанализировать особенности применения предметной области, относящейся к теме курсового проектирования,
- спроектировать концептуальную модель,
- спроектировать логическую модель,
- спроектировать физическую модель,
- разработать таблицы БД,
- разработать связи БД,
- заполнить таблицы БД,
- разработать представления,
- разработать функции,
- разработать процедуры,
- разработать триггеры,
- разработать приложение для доступа к БД,
- реализовать разграничение прав доступа пользователей,
- реализовать формирование отчётов,
- выполнить тестирование и отладку приложения,
- разработать руководство пользователя.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Голицына, О. Л. Базы данных : учебное пособие / О. Л. Голицына, Н. В. Максимов, И. И. Попов. – 4-е изд., перераб. и доп. – Москва : ФОРУМ : ИНФРА-М, 2020. – 400 с. – URL: https://znanium.com/catalog/document?id=362825. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.

2. Голицына, О. Л. Основы проектирования баз данных : учебное пособие / О. Л. Голицына, Т. Л. Партыка, И. И. Попов. – 2-е изд., перераб. и доп. Москва : ФОРУМ : ИНФРА-М, 2021. – 416 с. – URL: https://znanium.com/catalog/document?id=364900. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.

3. Дадян, Э. Г. Данные: хранение и обработка : учебник / Э. Г. Дадян. – Москва : ИНФРА-М, 2020. – 205 с. – URL: https://znanium.com/catalog/document?id=346013. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.

4. Павловская, Т. А. C#. Программирование на языке высокого уровня: Учебник для вузов / Т. А. Павловская. – Санкт-Петербург : Питер, 2021. – 432 с. – URL: https://ibooks.ru/bookshelf/377952/reading. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.

5. Фленов, М. Е. Библия C# / М. Е. Фленов. – 4-е изд., перераб. и доп. – Санкт-Петербург : БХВ-Петербург, 2019. – 512 с. – URL: https://ibooks.ru/bookshelf/366634/reading. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.