Оглавление

[ВВЕДЕНИЕ 2](#_Toc154260136)

[ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 5](#_Toc154260137)

[1.1 ПРЕДМЕТНАЯ ОБЛАСТЬ 5](#_Toc154260138)

[1.2. ТЕХНИЧЕСКАЯ ПОСТАНОВКА 6](#_Toc154260139)

[1.3 ВЫБОР ИНСТРУМЕНТОВ РЕАЛИЗАЦИИ 7](#_Toc154260140)

[2 ТЕХНИЧЕСКАЯ ЧАСТЬ 8](#_Toc154260141)

[2.1 РАЗРАБОТКА МОДЕЛИ БАЗЫ ДАННЫХ 8](#_Toc154260142)

[2.2 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА УПРАВЛЕНИЯ БАЗОЙ ДАННЫХ 12](#_Toc154260143)

[ЗАКЛЮЧЕНИЕ 17](#_Toc154260144)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 18](#_Toc154260145)

# ВВЕДЕНИЕ

Таблицы являются неотъемлемой частью нашей повседневной жизни и значительно ее упрощают. Простейшее определение: таблица — это совокупность связанных данных, хранящихся в структурированном виде, состоящая из столбцов и строк [1]. Именно они позволяют нам представлять различные данные в удобной форме, например, меню ресторана или квитанции для оплаты. Однако, когда мы имеем дело с огромными электронными таблицами, количество строк и столбцов в которых достигают сотни тысяч, обработка данных становится сложной даже при использовании специальных программ типа MS Excel.

Для решения этой задачи часто применяется разбивка большой таблицы на несколько более мелких и установление связей между их значениями. Такие структуры данных называют базами данных, и они представляют собой упорядоченную информацию, сохраненную в цифровом формате.

Базы данных - это структурированные наборы данных, которые организованы и хранятся для эффективного доступа, управления и обработки информации [2]. Базы данных широко используются в различных областях, включая бизнес, науку, образование и технологии и культуру.

Существует множество различных систем управления базами данных (СУБД). По сути это комплекс программ, позволяющих создать базу данных и манипулировать данными (вставлять, обновлять, удалять и выбирать) [1]. Система обеспечивает безопасность, надёжность хранения и целостность данных, а также предоставляет средства для администрирования БД. Конкретный выбор СУБД зависит от типа и сложности поставленной задачи. Ниже приведен список наиболее используемых типов СУБД:

1. Реляционные СУБД (SQL): Реляционные СУБД основаны на структурированном языке запросов SQL (Structured Query Language) Они используют таблицы с реляционными связями для хранения и организации данных.
2. NoSQL СУБД: NoSQL (Not Only SQL) СУБД предоставляют гибкую модель хранения данных, отличную от реляционных баз данных. Они позволяют хранить и обрабатывать неструктурированные данные, такие как документы, графы, ключ-значение и временные ряды.
3. Графовые СУБД: Графовые СУБД предназначены для работы с графовыми структурами данных, где узлы представляют объекты, а ребра - связи между ними [3].
4. Колоночные СУБД: Колоночные СУБД организуют данные по столбцам, что позволяет эффективно работать с большими объемами данных и выполнением аналитических запросов.
5. Документоориентированные СУБД: Документоориентированные СУБД хранят данные в формате документов, таких как JSON или XML.
6. Временные СУБД: Временные СУБД предназначены для хранения и обработки временных данных, таких как временные ряды или события.
7. Распределенные СУБД: Распределенные СУБД позволяют хранить данные на нескольких узлах или серверах, обеспечивая масштабируемость и отказоустойчивость.
8. Объектно-ориентированные СУБД: Объектно-ориентированные СУБД предоставляют возможность хранить и обрабатывать объекты, сохраняя их структуру и связи.
9. XML-ориентированные СУБД: XML-ориентированные СУБД предназначены для работы с данными в формате XML.

Применение баз данных является неотъемлемой частью эффективного управления и обработки данных в различных областях [4, 5]. Вот несколько примеров, иллюстрирующих необходимость использования баз данных:

* **Аналитика данных:** Базы данных применяются для хранения и анализа данных, собранных из различных источников. Они позволяют проводить сложные аналитические запросы, выявлять взаимосвязи и тенденции в данных. Это обеспечивает организациям возможность принимать обоснованные решения на основе подробного анализа данных.
* **Интернет-приложения:** В сфере интернет-приложений базы данных используются для хранения информации о пользователях, контенте, транзакциях и других параметрах. Они обеспечивают быстрый доступ к данным и поддерживают функциональность таких приложений, как социальные сети, электронная коммерция и онлайн-банкинг.
* **Бизнес-приложения:** Базы данных играют ключевую роль в хранении информации о клиентах, продуктах, заказах и других бизнес-данных. Они обеспечивают эффективное управление и быстрый доступ к этой информации, что помогает компаниям принимать обоснованные решения на основе обработки данных.
* **Научные исследования:** В области научных исследований базы данных применяются для хранения и анализа обширных объемов научных данных, таких как геномные, климатические, информация о популяциях и другие. Это позволяет исследователям эффективно организовывать и анализировать данные, способствуя новым открытиям и расширению знаний.

Целью данной курсовой работы является создание информационной системы (ИС) в виде программного продукта с использованием современных средств разработки. Для выполнения работы сформированы следующие задачи:

* Проанализировать предметную область для выявления предпочтительных путей реализации.
* Разработать логическую и физическую ER-модели данных.
* Создать базы данных на основе физической модели с использованием СУБД MySQL.
* Разработать интерфейс ИС с использованием языка программирования C#.
* Реализовать систему запросов и отчётов при работе пользователя через интерфейс с базами данных.
* Составить документацию описывающую проект, включая UML-диаграммы реализованные с помощью StarUML.

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## 1.1 ПРЕДМЕТНАЯ ОБЛАСТЬ

Информационная система, разрабатываемая в рамках данной курсовой работы, является агрегатором предназначенным, в первую очередь, для сравнения цен видео игр на разных онлайн торговых площадках.

Агрегаторы представляют собой каталоги, где можно найти нужные позиции по определённым характеристикам и сравнить их между собой. Они играют значительную роль в современном цифровом мире [6].

Вот несколько аспектов, подчеркивающих актуальность цифровых агрегаторов:

* **Сбор и организация информации.** Цифровые агрегаторы предоставляют возможность собирать информацию из различных источников и сохранять в одном месте. Люди могут получать доступ к разнообразным данным без необходимости посещать множество различных платформ.
* **Экономия времени и удобство пользователя.** Агрегаторы упрощают жизнь пользователей, предоставляя одно окно доступа ко множеству сервисов и источников информации. Пользователи экономят время, которое раньше тратили на поиск и переключение между приложениями или сайтами.
* **Персонализированный контент.** Многие цифровые агрегаторы используют алгоритмы машинного обучения для адаптации контента под конкретные интересы пользователя. Это создает персонализированный опыт и улучшает качество предоставляемой информации.
* **Сравнение цен и услуг.** В сфере электронной коммерции цифровые агрегаторы позволяют пользователям сравнивать цены на товары и услуги с различных платформ. Это помогает потребителям принимать более информированные решения о покупках.
* **Расширение рынков и предоставление платформ для бизнеса.** Для предпринимателей цифровые агрегаторы могут служить платформой для привлечения новых клиентов, объединения услуг и увеличения видимости бизнеса в цифровом пространстве.

## 1.2. ТЕХНИЧЕСКАЯ ПОСТАНОВКА

Обозначим набор основных требований к создаваемой информационной системе для сравнения цен видео игр на разных онлайн торговых площадках, назовем ее ИС Агрегатор.

Интерфейс ИС Агрегатор должен реализовывать следующие возможности:

* Поиск и демонстрация продуктов соответствующих заданным пользователем параметрам.
* Возможность добавления и удаления продуктов в корзину покупок.
* Подсчет общей стоимости продуктов в корзине покупок.

Поиск производиться посредством выбора параметра продукта, по которому производится поиск, и ввода значения данного параметра. После чего отображаются либо список найденных продуктов, либо сообщение об отсутствии товаров, соответствующих запросу.

В число параметров, по которым может производиться поиск, должны входить:

* Название игры.
* Разработчик.
* Издатель игры.
* Интернет магазин, в котором продается игра.

Информация о продукте, получаемая в результате запроса должна включать:

* Интернет магазин, в котором продается игра.
* Название игры.
* Описание игры.
* Цена.
* Оценка отзывов пользователей.
* Разработчик, издатель.
* Год выпуска.

Корзина покупок должна отображать следующие характеристики добавленного в неё продукта:

* Интернет магазин, в котором продается игра.
* Название игры.
* Цена

## 1.3 КОНКРЕТИЗАЦИЯ ВОЗМОЖНОСТЕЙ И РОЛЕЙ

Для реализации поставленной задачи необходимо подробно рассмотреть сведения и взаимодействия отдельных элементов для создаваемой программы. Схематичное изображение помогает проиллюстрировать представленную информацию. Подробнее рассмотрим возможности предоставляемые пользователю и распределение процессов по частям системы.

Пользователь системы должен иметь возможность просмотра списка товаров, ограниченного заданными им параметрами поиска. Также программа должна предоставлять возможность просмотра и изменения содержимого корзины покупок. В список опций по изменению корзины входят добавление и удаление товаров. Это отражено на Диаграмме прецедентов (Рис 1).

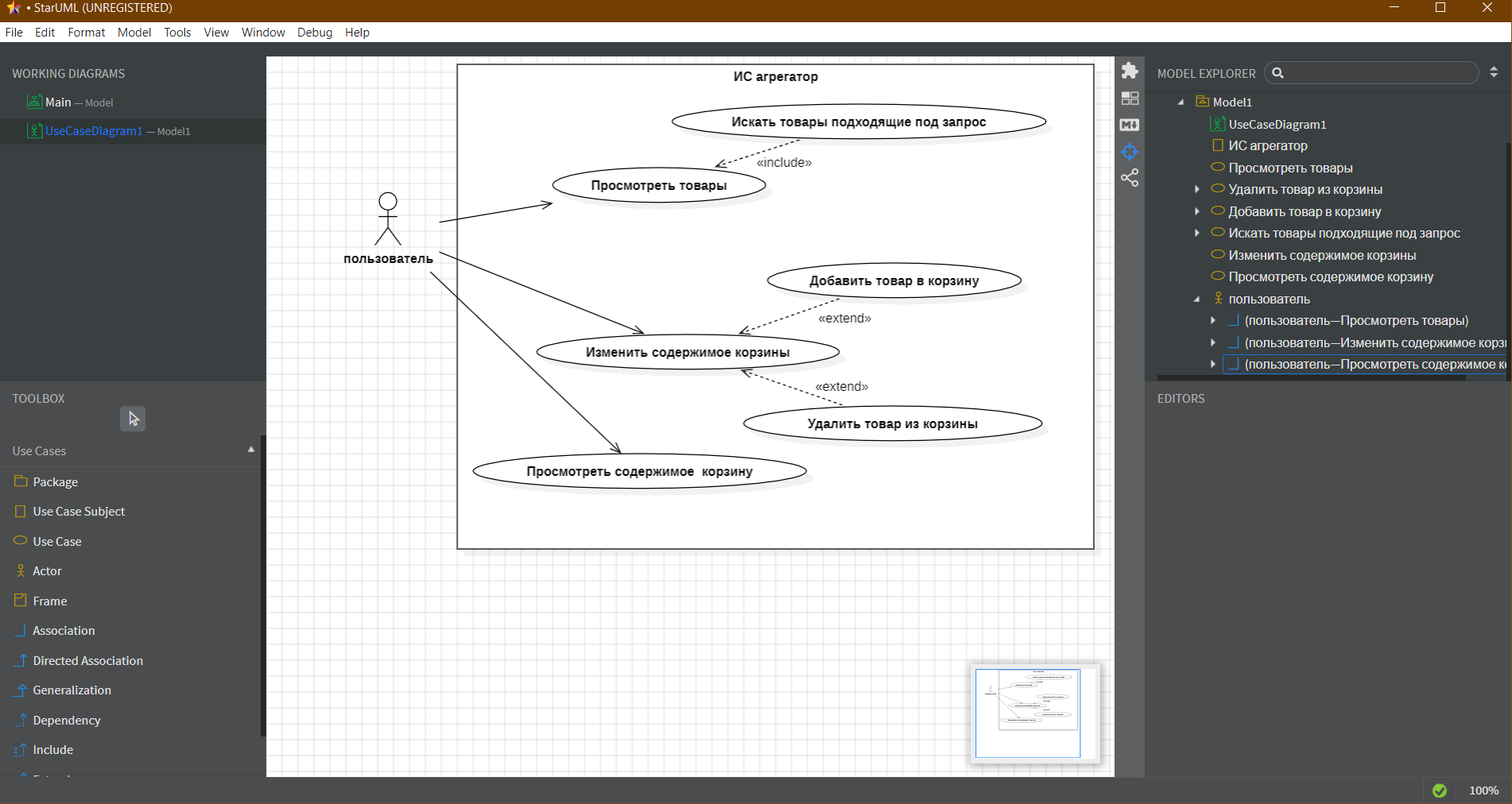


Рис. 1. Диаграмма прецедентов

На рисунке 2 представлена Диаграмма последовательности, на которой показаны последовательность взаимодействия пользователя и подсистем ИС Агрегатор.

Пользователь выполняет поиск по заданным параметрам. Исходя из характеристики и заданного запроса, ИС выбирает SQL запрос. СУБД выполняет этот запрос и возвращает значение в виде таблицы. За тем программа преобразует таблицу в список и выводит его на экран для просмотра пользователем.

Схожим образом выполняется изменений содержимого корзины покупок.

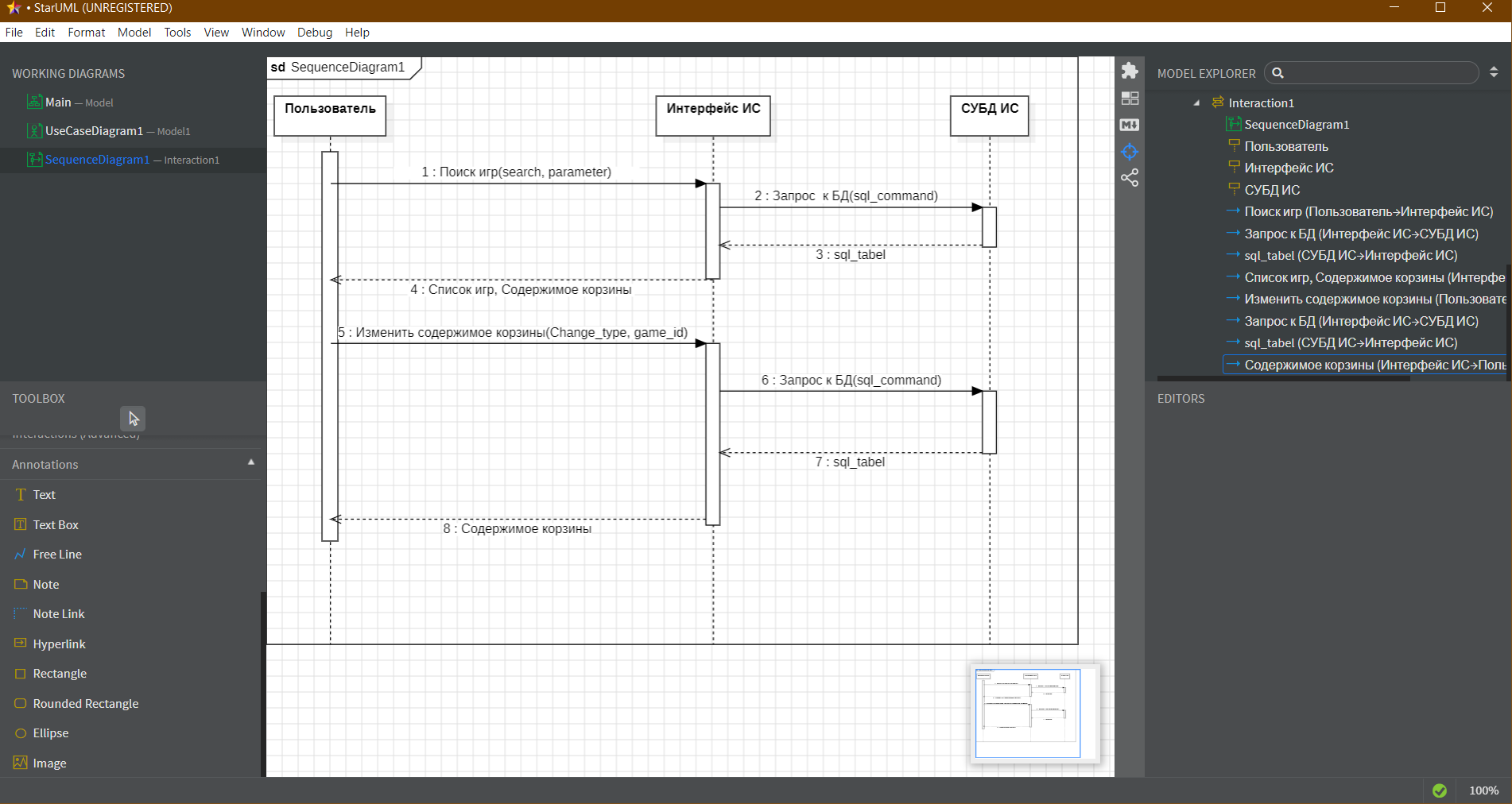


Рис. 2. Диаграмма последовательностей

## 1.4 ВЫБОР ИНСТРУМЕНТОВ РЕАЛИЗАЦИИ

Выполнение поставленной задачи требует два основных средства реализации: система управления базами данных и язык программирования для написания программного средства работы с этой базой данных.

В ходе проведенного исследования, предпочтение отдано языку программирования C#. Для реализации пользовательского интерфейса информационной системы он имеет несколько преимуществ перед аналогами, такими как Python и Java, в частности:

* Использование платформы .NET: C# разработан для работы на платформе .NET, которая предоставляет широкий набор инструментов и библиотек для разработки приложений.
* Библиотеки, такие как Windows Forms или Universal Windows Platform, обеспечивают удобные инструменты для создания современных и красивых интерфейсов.
* C# обладает возможностью легкой интеграции с базами данных и веб-службами, что позволяет облегчить взаимодействие частей приложения.

В качестве инструмента реализации было решено выбрать систему управления базами данных реляционного типа MySQL. Данный выбор обусловлен рядом имеющихся преимуществ среди которых:

1. Простота в использовании: MySQL предлагает простой и интуитивно понятный синтаксис, что делает его легким для изучения и использования.
2. Высокая производительность и масштабируемость: MySQL обеспечивает быструю обработку запросов и хорошую производительность при работе с большими объемами данных.
3. Хорошая совместимость с другими технологиями: MySQL интегрируется легко с другими технологиями, такими как PHP, C#, Java и другими. Это позволяет создавать более сложные информационные системы, которые могут взаимодействовать с другими компонентами создаваемого приложения.

Для настройки взаимодействия базы данных и интерфейса был использован MySQL Connector/NET.

В итоге проведенного сравнения, для реализации данной информационной системы использовались следующие инструменты: язык программирования C#, система управления базами данных MySQL, драйвер MySQL Connector/NET. В качестве сред разработки выступили Microsoft Visual Studio и MySQL Workbench.

# 2 ТЕХНИЧЕСКАЯ ЧАСТЬ

## 2.1 РАЗРАБОТКА МОДЕЛИ БАЗЫ ДАННЫХ

Реализация поставленной задачи требует наличие в базе данных следующих таблиц:

1)game содержит данные непосредственно об игре:

* game\_id – первичный ключ,
* title – название игры,
* description – описание игры,
* release\_year – год выхода (или появления в магазине),
* cost – стоимость игры,
* reviews – обобщенная оценка от пользователей,
* inventory\_inventory\_id – связь с таблицей inventory,
* publisher\_publisher\_id – связь с таблицей publisher,
* developer\_developer\_id – связь с таблицей developer.

2) inventory содержит данные о связи игры и магазина:

* inventory\_id – первичный ключ,
* web\_store\_web\_store\_id – связь с таблицей web\_store.

3) web\_store содержит данные о интернет магазинах:

* web\_store\_id – первичный ключ,
* web\_store – название магазина,
* web\_addres – web адрес магазина.

4) shopping\_cart содержит данные о содержимом корзины покупок:

* product\_id – первичный ключ,
* game\_game\_id – связь с таблицей game,
* bill\_bill\_id – связь с таблицей bill.

5) bill содержит данные об общей стоимости содержимого корзины покупок:

* bill\_id – первичный ключ,
* bill\_sum – общая стоимость содержимого корзины покупок.

6) developer содержит данные о разработчиках:

* developer\_id – первичный ключ,
* developer\_name – название компании разработчика,

7) publisher содержит данные о издателях:

* publisher\_id – первичный ключ,
* publisher\_name – название компании издателя.

8) publisher\_city содержит данные о связи издателя и города, в котором он базируется:

* city\_id – связь с таблицей city,
* publisher\_publisher\_id – связь с таблицей publisher.

9) city содержит данные о городах, в которых базируются издатели:

* city\_id – первичный ключ,
* city – название города,
* country\_country\_id – связь с таблицей country.

10) country содержит данные о странах, в которых базируются издатели:

* country\_id – первичный ключ,
* country – название города.

11) game\_tag содержит данные о связи тегов и игры

* tag\_tag\_id – связь с таблицей tag
* game\_id – связь с таблицей game

12) tag содержит данные о странах:

* tag\_id – первичный ключ,
* tag\_name – название тега,
* description – описание тега.

13) user содержит данные о пользователях:

* user\_id – первичный ключ,
* user\_name – имя пользователя,
* user\_password – пароль пользователя,
* user\_role – роль пользователя, определяет является он администратором или обычным пользователем.

Данная модель была составлена, исходя из требований, выставленных к системе, EER диаграмма базы данных выглядит следующим образом:

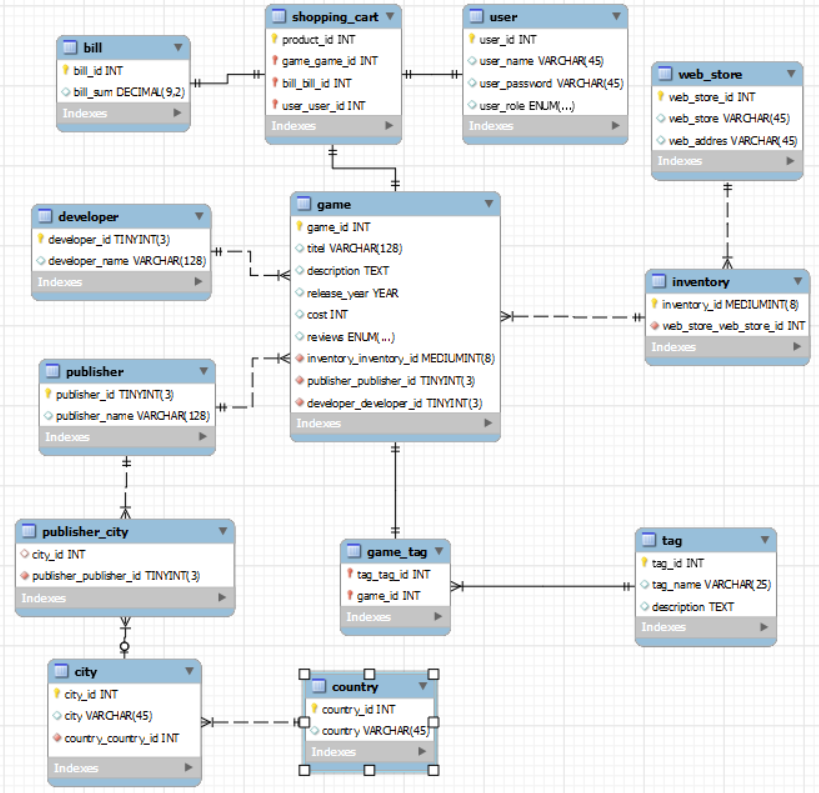


Рис 3. EER диаграмма базы данных

Помимо всего вышеперечисленного так же необходимо создать ряд представлений.

1) shopping\_cart\_summary должно отображать:

* product\_id – идентификатор продукта,
* titel – название игры,
* web\_store – название интернет магазина,
* description – описание игры,
* cost – стоимость игры,
* bill\_sum – общая стоимость продуктов в корзине покупок.

1) game\_summary должно отображать:

* game\_id – идентификатор игры,
* title – название игры,
* description – описание игры,
* release\_year – год выпуска игры,
* cost – стоимость игры,
* reviews – обобщенная оценка от пользователей,
* deviloper\_name – название компании разработчика,
* publisher\_name – название компании издателя,
* web\_store – название интернет магазина.

Для корректного функционирования корзины покупок написаны необходимые триггеры, которые будут обновлять общую стоимость товаров в корзине покупок.

1. Добавление товара в корзину:

DELIMITER //  
CREATE TRIGGER update\_bill\_sum  
AFTER INSERT ON shopping\_cart  
FOR EACH ROW  
BEGIN  
 UPDATE bill  
 SET bill\_sum = bill\_sum + (  
 SELECT cost  
 FROM game  
 WHERE game\_id = NEW.game\_game\_id)  
 WHERE bill\_id = NEW.bill\_bill\_id;  
END //  
DELIMITER ;

1. Удаление товара из корзины:

DELIMITER //  
CREATE TRIGGER update\_bill\_sum\_on\_delete  
AFTER DELETE ON shopping\_cart  
FOR EACH ROW  
BEGIN  
 UPDATE bill  
 SET bill\_sum = bill\_sum - (  
 SELECT cost  
 FROM game  
 WHERE game\_id = OLD.game\_game\_id  
 )  
 WHERE bill\_id = OLD.bill\_bill\_id;  
END //  
DELIMITER ;

Для корректного отображения списка продуктов после добавления или удаления элемента, написаны необходимые вложенные процедуры, которые будут обновлять представления.

DELIMITER //

CREATE PROCEDURE update\_game\_summary()

BEGIN

DROP VIEW IF EXISTS game\_summary;

CREATE VIEW game\_summary AS

SELECT

g.game\_id,

g.titel,

g.description,

g.release\_year,

g.cost,

g.reviews,

d.developer\_name AS developer\_name,

p.publisher\_name AS publisher\_name,

ws.web\_store AS web\_store

FROM

game g

JOIN

developer d ON g.developer\_developer\_id = d.developer\_id

JOIN

publisher p ON g.publisher\_publisher\_id = p.publisher\_id

JOIN

inventory i ON g.inventory\_inventory\_id = i.inventory\_id

JOIN

web\_store ws ON i.web\_store\_web\_store\_id = ws.web\_store\_id;

END//

DELIMITER ;

## 2.2 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА УПРАВЛЕНИЯ БАЗОЙ ДАННЫХ

Для реализации функций агрегатора было разработано программное средство, включающее в себя следующие функции:

1) **Подключение к базе данных**. Для подключения к базе данных необходимо задать параметры подключения, а также задать функции для подключения и отключения.

MySqlConnection connection = new MySqlConnection("server=localhost;user=root;password=root;database=kyrsovaya\_v0;");

public void openConnection()

{

if (connection.State == System.Data.ConnectionState.Closed)

{

connection.Open();

}

}

public void closeConnection()

{

if (connection.State == System.Data.ConnectionState.Open)

{

connection.Close();

}

}

public MySqlConnection getConnection()

{

return connection;

}

2) **Авторизация.** Для авторизации необходимо логин и пароль, нажать кнопку «Авторизоваться» (Рис. 4). В случае нахождения в базе данных совпадений, в зависимости от роли пользователя он перейдет или на форму пользователя либо на форму администратора. Если совпадений найдено не было выводится сообщение, информирующее пользователя об этом.

private void buttonLogIn\_Click(object sender, EventArgs e)

{

String loginUser = textBoxUsername.Text;

String passwordUser = textBoxPassword.Text;

DataBase db = new DataBase();

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command = new MySqlCommand("SELECT \* FROM `user` WHERE `user\_name` = @username AND `user\_password` = @user\_password", db.getConnection());

command.Parameters.Add("@username", MySqlDbType.VarChar).Value = loginUser;

command.Parameters.Add("@user\_password", MySqlDbType.VarChar).Value = passwordUser;

adapter.SelectCommand = command;

adapter.Fill(table);

if (table.Rows.Count > 0)

{

DataRow[] rows = table.Select();

User user = new User(

Convert.ToString(rows[0]["user\_id"]),

Convert.ToString(rows[0]["user\_name"]),

Convert.ToString(rows[0]["user\_password"]),

Convert.ToString(rows[0]["user\_role"]));

this.Hide();

if (user.Role == "User")

{

Form1 form1 = new Form1(user);

form1.Show();

}

else if (user.Role == "Admin")

{

Form3 form3 = new Form3(user);

form3.Show();

}

}

else

{ MessageBox.Show("Не верный логин или пароль. Проверьте введеные данные."); }

}

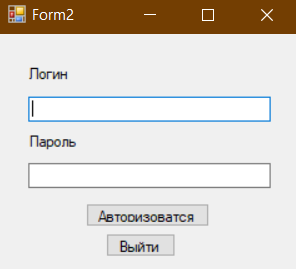


Рис 4. Форма Авторизации.

3) **Поиск**. Для выполнения поиска необходимо выбрать характеристику, по которой будет производиться поиск, ввести поисковой запрос и нажать кнопку «Найти». В случае нахождения в базе данных совпадений, игры должны выводиться по одной на страницу с возможностью переключения страниц (Рис. 5). Если совпадений найдено не было выводится сообщение, информирующее пользователя об этом.

private void FindButton\_Click(object sender, EventArgs e)

{

string search = textBoxSearch.Text;

DataBase db = new DataBase();

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command;

if (radioButtonTitel.Checked) {

command = new MySqlCommand("SELECT \* FROM `game\_summary` WHERE `titel` = @Titel", db.getConnection());

command.Parameters.Add("@Titel", MySqlDbType.VarChar).Value = search;

adapter.SelectCommand = command;

adapter.Fill(table);

if (!(table.Rows.Count > 0))

{

MessageBox.Show("По вашему запросу ничего не найдено");

FindedGames = new List<Game>();

UpdateGameInterfece();

}

else {

FromTableToListGame(table);

UpdateGameInterfece();

}

}

else if (radioButtonDeveloper.Checked) {

command = new MySqlCommand("SELECT \* FROM `game\_summary` WHERE `developer\_name` = @Developer;", db.getConnection());

command.Parameters.Add("@Developer", MySqlDbType.VarChar).Value = search;

adapter.SelectCommand = command;

adapter.Fill(table);

if (!(table.Rows.Count > 0))

{

MessageBox.Show("По вашему запросу ничего не найдено");

FindedGames = new List<Game>();

UpdateGameInterfece();

}

else {

FromTableToListGame(table);

UpdateGameInterfece();

}

}

else if (radioButtonPublisher.Checked)

{

command = new MySqlCommand("SELECT \* FROM `game\_summary` WHERE `publisher\_name` = @Publisher;", db.getConnection());

command.Parameters.Add("@Publisher", MySqlDbType.VarChar).Value = search;

adapter.SelectCommand = command;

adapter.Fill(table);

if (!(table.Rows.Count > 0))

{

MessageBox.Show("По вашему запросу ничего не найдено");

FindedGames = new List<Game>();

UpdateGameInterfece();

}

else {

FromTableToListGame(table);

UpdateGameInterfece();

}

}

else if (radioButtonWebStore.Checked) {

command = new MySqlCommand("SELECT \* FROM `game\_summary` WHERE `web\_store` = @Web\_Store;", db.getConnection());

command.Parameters.Add("@Web\_Store", MySqlDbType.VarChar).Value = search;

adapter.SelectCommand = command;

adapter.Fill(table);

if (!(table.Rows.Count > 0))

{

MessageBox.Show("По вашему запросу ничего не найдено");

FindedGames = new List<Game>();

UpdateGameInterfece();

}

else {

FromTableToListGame(table);

UpdateGameInterfece();

}

}

else if (radioButtonCost.Checked)

{

try { Convert.ToInt32(search); }

catch

{

MessageBox.Show("Цена должна быть числом");

return;

}

command = new MySqlCommand("SELECT \* FROM `game\_summary` WHERE `cost` < @Cost;", db.getConnection());

command.Parameters.Add("@Cost", MySqlDbType.Int32).Value = search;

adapter.SelectCommand = command;

adapter.Fill(table);

if (!(table.Rows.Count > 0))

{

MessageBox.Show("По вашему запросу ничего не найдено");

FindedGames = new List<Game>();

UpdateGameInterfece();

}

else

{

FromTableToListGame(table);

UpdateGameInterfece();

}

}

else {

MessageBox.Show("Параметры поcика не выбраны!");

}

}

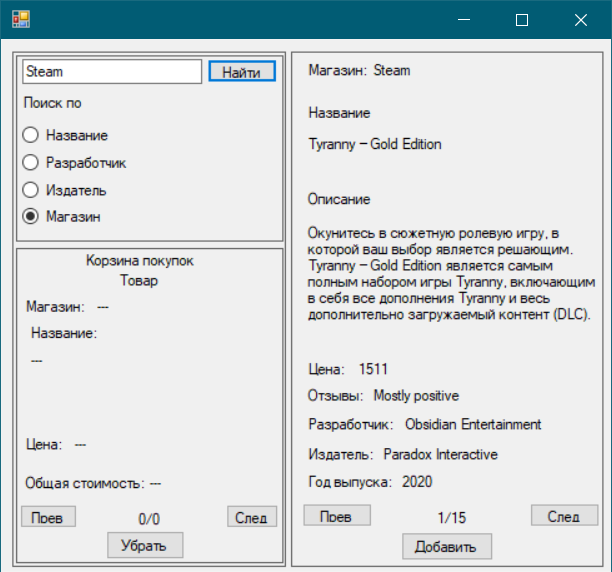


Рис. 5. Выполненный поиск по названию магазина

4) **Добавление товара в корзину**. Для добавления товара в корзину нужно выбрать игру и нажать на кнопку «Добавить», после этого должно появиться сообщение об успешном добавлении товара и в разделе «Корзина покупок» должна появиться выбранная игра (Рис. 6). В случае если игра не выбрана, выводится сообщение об ошибке.

private void buttonAddProduct\_Click(object sender, EventArgs e)

{

int game\_id = Convert.ToInt32(FindedGames[GameMarker].Game\_Id);

DataBase db = new DataBase();

MySqlCommand command = new MySqlCommand("INSERT INTO `kyrsovaya\_v0`.`shopping\_cart` (`game\_game\_id`, `bill\_bill\_id`) VALUES (@Game\_Id, '1')", db.getConnection());

command.Parameters.Add("@Game\_Id", MySqlDbType.Int32).Value = game\_id;

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{ MessageBox.Show("Игра успешно добавленна в корзину"); }

else

{ MessageBox.Show("Упс! При добавлении иры в корзину возникла ошибка"); }

db.closeConnection();

getShoppingCartSummary();

}

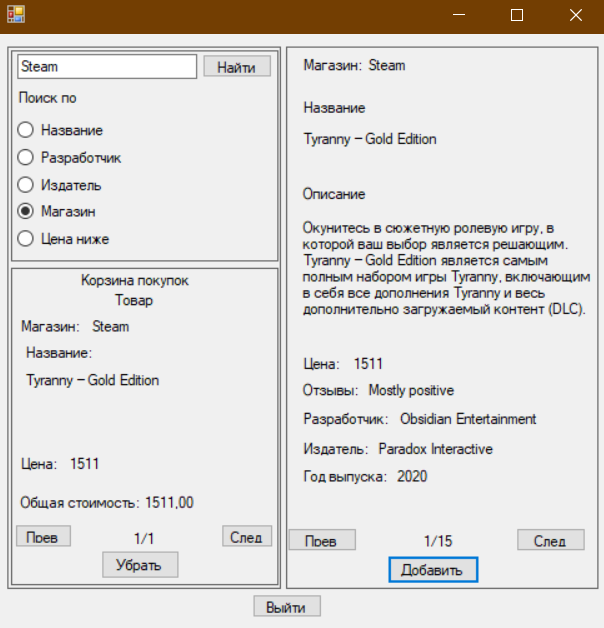


Рис. 6. Добавление товара в корзину

5) **Удаление товара из корзины**. Для удаления товара из корзины нужно выбрать игру в разделе «Корзина покупок» и нажать на кнопку «Убрать» (Рис. 7). После этого должно появиться сообщение об успешном удалении товара. В разделе «Корзина покупок» выбранная игра должна отсутствовать. В случае если игра не выбрана, выводится сообщение об ошибке.

private void buttonRemoveProduct\_Click(object sender, EventArgs e)

{

int product\_id = Convert.ToInt32(ShoppingCart[ProductMarker].Product\_ID);

DataBase db = new DataBase();

MySqlCommand command = new MySqlCommand("DELETE FROM `kyrsovaya\_v0`.`shopping\_cart` WHERE `product\_id` = @Product\_ID", db.getConnection());

command.Parameters.Add("@Product\_ID", MySqlDbType.Int32).Value = product\_id;

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{ MessageBox.Show("Игра успешно удалена в корзину"); }

else

{ MessageBox.Show("Упс! При удалении из корзину возникла ошибка"); }

db.closeConnection();

getShoppingCartSummary();

UpdateProductInterfece();

}

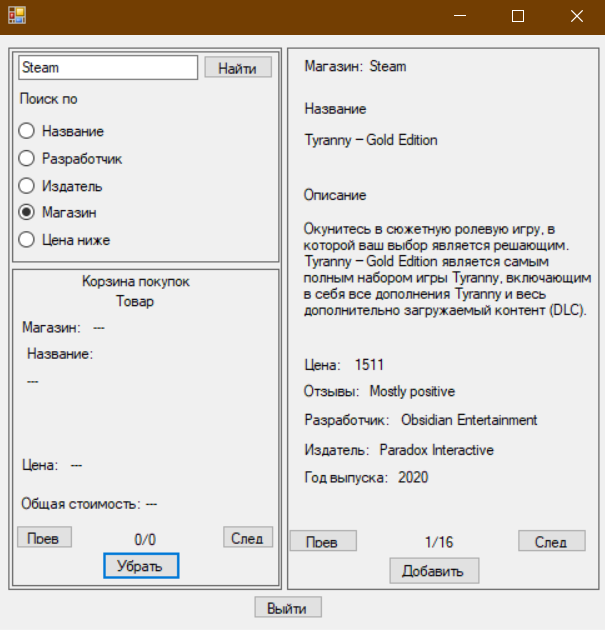


Рис. 7. Удаление игры из корзины.

6) **Добавление продукта**. Для добавления товара в корзину нужно заполнить все поля в разделе игру и нажать на кнопку «Добавить», после этого должно появиться сообщение об успешном добавлении товара и в разделе «Корзина покупок» должна появиться выбранная игра (Рис. 6). В случае если игра не выбрана, выводится сообщение об ошибке.

private List<Developer> getDevelopers()

{

List<Developer> developers = new List<Developer>();

DataBase db = new DataBase();

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command;

command = new MySqlCommand("SELECT \* FROM `developer`", db.getConnection());

adapter.SelectCommand = command;

adapter.Fill(table);

DataRow[] rows = table.Select();

for (int i = 0; i < rows.Length; i++)

{

Developer p = new Developer(Convert.ToInt32(rows[i]["developer\_id"]), Convert.ToString(rows[i]["developer\_name"]));

developers.Add(p);

}

return developers;

}

private List<Publisher> getPublishers()

{

List<Publisher> publishers = new List<Publisher>();

DataBase db = new DataBase();

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command;

command = new MySqlCommand("SELECT \* FROM `publisher`", db.getConnection());

adapter.SelectCommand = command;

adapter.Fill(table);

DataRow[] rows = table.Select();

for (int i = 0; i < rows.Length; i++)

{

Publisher p = new Publisher(Convert.ToInt32(rows[i]["publisher\_id"]), Convert.ToString(rows[i]["publisher\_name"]));

publishers.Add(p);

}

return publishers;

}

private void createDeveloper()

{

string developer\_name = textBoxDeveloperName.Text;

DataBase db = new DataBase();

MySqlCommand command = new MySqlCommand("INSERT INTO `kyrsovaya\_kis`.`developer` (`developer\_name`) VALUES (@Developer\_Name);", db.getConnection());

command.Parameters.Add("@Developer\_Name", MySqlDbType.String).Value = developer\_name;

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{ MessageBox.Show("Разработчик успешно добавленн"); }

else

{ MessageBox.Show("Упс! При добавлении Разработчика возникла ошибка"); }

db.closeConnection();

}

private void createPublisher()

{

string publisher\_name = textBoxPublisherName.Text;

DataBase db = new DataBase();

MySqlCommand command = new MySqlCommand("INSERT INTO `kyrsovaya\_kis`.`publisher` (`publisher\_name`) VALUES (@Publisher\_Name);", db.getConnection());

command.Parameters.Add("@Publisher\_Name", MySqlDbType.String).Value = publisher\_name;

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{ MessageBox.Show("Издатель успешно добавленн"); }

else

{ MessageBox.Show("Упс! При добавлении Издателя возникла ошибка"); }

db.closeConnection();

}

private void buttonAddProduct\_Click(object sender, EventArgs e)

{

List<Developer> developers = getDevelopers();

List<Publisher> publishers = getPublishers();

Developer curent\_developer = null;

Publisher curent\_publisher = null;

int web\_store\_id=0;

string web\_store = textBoxWebStore.Text;

string titel = textBoxTitel.Text;

string description = textBoxDescription.Text;

string cost = textBoxCost.Text;

string reviews = comboBox1.Text;

string developer\_name = textBoxDeveloperName.Text;

string publisher\_name = textBoxPublisherName.Text;

string release\_year = textBoxReleaseYear.Text;

// Проверки полей.

if (web\_store == "")

{

MessageBox.Show("Не введено название магазаина");

return;

}

else if (web\_store == "Steam") { web\_store\_id = 1; }

else if (web\_store == "Epic Games") { web\_store\_id = 2; }

else

{

MessageBox.Show("Такого магазина не существует, проверьде корректность введенных данных.") ;

return;

}

if (titel == "")

{

MessageBox.Show("Не введено название игры");

return;

}

if(description == "")

{

MessageBox.Show("Не введено название описание игры");

return;

}

if(cost == "")

{

MessageBox.Show("Не введена стоимость игры");

return;

}

if(reviews == "")

{

MessageBox.Show("Не введены отзывы");

return;

}

if (developer\_name == "")

{

MessageBox.Show("Не введено название разработчика");

return;

}

if(publisher\_name == "")

{

MessageBox.Show("Не введено название издателя");

return;

}

if(release\_year == "")

{

MessageBox.Show("Не введен год выхода игры");

return;

}

try

{ int new\_cost = Convert.ToInt32(cost); }

catch

{

MessageBox.Show("Цена должна быть числом");

return;

}

try

{ int new\_release\_year = Convert.ToInt32(release\_year);

}

catch

{

MessageBox.Show("Год выхода должен быть числом");

return;

}

for (int i = 0; i < developers.Count; i++)

{

if (developers[i].Developer\_Name == developer\_name)

{

curent\_developer = developers[i];

}

}

if (curent\_developer == null)

{

createDeveloper();

developers = getDevelopers();

for (int i = 0; i < developers.Count; i++)

{

if (developers[i].Developer\_Name == developer\_name)

{

curent\_developer = developers[i];

}

}

}

for (int i = 0; i < publishers.Count; i++)

{

if (publishers[i].Publisher\_Name == publisher\_name)

{

curent\_publisher = publishers[i];

}

}

if (curent\_publisher == null)

{

createPublisher();

publishers = getPublishers();

for (int i = 0; i < publishers.Count; i++)

{

if (publishers[i].Publisher\_Name == publisher\_name)

{

curent\_publisher = publishers[i];

}

}

}

DataBase db = new DataBase();

MySqlCommand command = new MySqlCommand("INSERT INTO `kyrsovaya\_kis`.`game` (`titel`, `description`, `release\_year`, `cost`, `reviews`, `inventory\_inventory\_id`, `publisher\_publisher\_id`, `developer\_developer\_id`) VALUES (@Titel, @Description, @Release\_Year, @Cost, @Reviews, @Inventory\_Inventory\_Id, @Publisher\_Publisher\_Id, @Developer\_Developer\_Id);", db.getConnection());

command.Parameters.Add("@Titel", MySqlDbType.VarChar).Value = titel;

command.Parameters.Add("@Description", MySqlDbType.Text).Value = description;

command.Parameters.Add("@Release\_Year", MySqlDbType.Year).Value = release\_year;

command.Parameters.Add("@Cost", MySqlDbType.Int32).Value = cost;

command.Parameters.Add("@Reviews", MySqlDbType.Enum).Value = reviews;

command.Parameters.Add("@Inventory\_Inventory\_Id", MySqlDbType.Int32).Value = web\_store\_id;

command.Parameters.Add("@Publisher\_Publisher\_Id", MySqlDbType.Int32).Value = curent\_publisher.Publisher\_Id;

command.Parameters.Add("@Developer\_Developer\_Id", MySqlDbType.Int32).Value = curent\_developer.Developer\_Id;

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{ MessageBox.Show("Продукт успешно добавленн"); }

else

{ MessageBox.Show("Упс! При добавлении Продукта возникла ошибка"); }

db.closeConnection();

}

7) **Удаление продукта**. Для удаления товара из корзины нужно выбрать игру в разделе «Результаты поиска» и нажать на кнопку «Удалить». После этого должно появиться сообщение об успешном удалении товара. В разделе «Результаты поиска» выбранная игра должна отсутствовать. В случае если игра не выбрана, выводится сообщение об ошибке.

private void buttonDeleteProduct\_Click(object sender, EventArgs e)

{

int game\_id = Convert.ToInt32(FindedGames[GameMarker].Game\_Id);

DataBase db = new DataBase();

MySqlCommand command = new MySqlCommand("DELETE FROM `kyrsovaya\_kis`.`game` WHERE `game\_id` = @Game\_ID", db.getConnection());

command.Parameters.Add("@Game\_ID", MySqlDbType.Int32).Value = game\_id;

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{ MessageBox.Show("Игра успешно удалена "); }

else

{ MessageBox.Show("Упс! При удалении возникла ошибка"); }

db.closeConnection();

FindButton\_Click(sender, e);

}

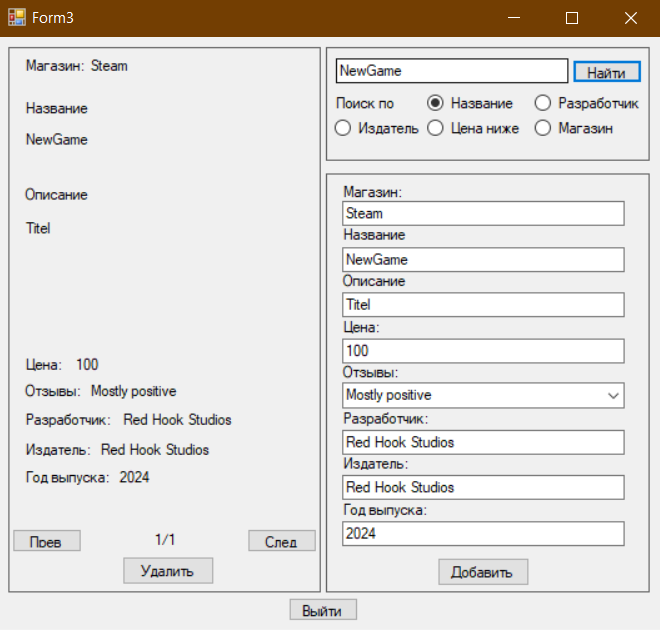


Рис. 8. Форма администратора.

# ЗАКЛЮЧЕНИЕ

В результате выполнении курсовой работы была создана Информационная система, которая является агрегатором, предназначенным для сравнения цен видео игр на разных онлайн торговых площадках.

Для выполнения поставленной цели была проанализирована предметная область для выявления предпочтительных путей реализации. Среди инструментов, позволяющих реализовать обозначенные функции, выбраны:

* язык программирования C#,
* система управления базами данных MySQL,
* драйвер MySQL Connector/NET.

В качестве сред разработки выступили Microsoft Visual Studio и MySQL Workbench.

На следующем этапе разработана логическая и физическая EER-модели данных и создана база данных на основе физической модели с использованием СУБД MySQL. Далее создан интерфейс ИС с использованием языка программирования C#. И на конечном этапе реализована система запросов и отчётов при работе пользователя через интерфейс с базами данных.

Все этапы описывающие реализацию проекта, подробно представлены в разделах данной курсовой работы, дополнены рисунками и UML-диаграммами, реализованными с помощью StarUML.

В заключении можно сказать, что все поставленные задачи успешно выполнены, поставленная цель достигнута: создана Информационная система, которая является агрегатором, предназначенным для сравнения цен видео игр на разных онлайн торговых площадках. В ходе тестирования ИС показала свою работоспособность и удобство использования. Дополнительно предусмотрена возможность доработки функционала под расширенные требования и может быть использована для конкретного заказчика.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. <https://ru.wikipedia.org/wiki>
2. <https://www.oracle.com/cis/database/what-is-database/>
3. https://habr.com/ru/articles/579248/
4. Рогачева Ю.И. Киржаев Д.А. Анализ популярных СУБД и необходимость их применения в современных информационных системах // Экономика и социум, 2020. – С. 542-545
5. Матвеева М.В. Применение баз данных в сфере образования // Матвеева М.В. Малинова О.Е. Новожеева А.А. Орлова М.О. - Приоритетные научные направления: от теории к практике, 2016. – С. 71-75
6. https://secretmag.ru/enciklopediya/chto-takoe-agregator-obyasnyaem-prostymi-slovami.htm