



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

*«Разработка приложения для просмотра
статистики о времени прохождения игр»*

Студент ИУ7-64Б
(Группа)

(Подпись, дата)

А. В. Золотухин
(И.О.Фамилия)

Руководитель курсовой работы

(Подпись, дата)

А. А. Павельев
(И.О.Фамилия)

2023 г.

РЕФЕРАТ

Курсовая работа представляет собой реализацию базы данных, позволяющую собирать у пользователей информацию об играх:

- время прохождения игры(время прохождения сюжета, время 100% прохождения);
- отзывы и оценки;

а также приложение, предоставляющее интерфейс для доступа к базе данных. Готовый продукт позволяет просматривать, изменять, добавлять и удалять данные из базы.

В качестве системы управления базой данных используется Sql Server, которая подключается к приложению, реализованному на языке программирования C#.

Ключевые слова: база данных, Sql Server, система управления базами данных, интерфейс.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	6
1 АНАЛИТИЧЕСКАЯ ЧАСТЬ	7
1.1 Формализация задачи	7
1.2 Формализация данных	7
1.3 Типы пользователей	9
1.4 Анализ существующих решений	12
1.4.1 Критерии	12
1.4.2 Steam	13
1.4.3 Сайт HowLongToBeat	13
1.4.4 Сравнение существующих решений	14
1.5 Анализ баз данных	14
1.5.1 Кэширование данных	15
2 КОНСТРУКТОРСКАЯ ЧАСТЬ	17
2.1 Проектирование базы данных	17
2.2 UML диаграмма классов приложения	20
3 ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ	22
3.1 Выбор СУБД для решения задачи	22
3.2 Выбор инструментов разработки	22
3.3 Детали реализации	23
3.3.1 Ролевая модель	23
3.3.2 Реализация процедуры преобразования строки в xml формат и вставки в таблицу	24
3.3.3 Модели хранения данных	26
3.4 Интерфейс приложения	29
4 ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ	37
4.1 Цель эксперимента	37
4.2 Технические характеристики	37
4.3 Описание эксперимента	37
4.3.1 Используемые данные	38
4.4 Результаты эксперимента	38

4.5 Вывод	40
ЗАКЛЮЧЕНИЕ	41
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	42

ВВЕДЕНИЕ

Легко проверить, сколько времени уходит на просмотр фильма или телепередачи, так почему бы не сделать то же самое и с видеоиграми? Никто не хочет покупать за немаленькие деньги слишком короткую игру или 200-часовую RPG, на которую нет столько времени.

Было бы неплохо, если бы существовало такое приложение или сайт, на котором люди могли бы смотреть примерное время прохождения игры.

Цель курсовой работы – разработка базы данных для просмотра информации о времени прохождения игр.

Чтобы достигнуть поставленной цели, требуется решить следующие задачи:

- проанализировать существующие решения;
- формализовать задачу и данные;
- проанализировать способы хранения данных и системы управления базами данных, выбрать наиболее подходящие решения для поставленной задачи;
- спроектировать и разработать базу данных;
- реализовать программное обеспечение, которое позволит получить доступ к данным;
- провести исследование зависимости времени обработки данных от их объема и от распределения вычислений между базой данных и приложением.

1 АНАЛИТИЧЕСКАЯ ЧАСТЬ

В данном разделе приведена формализация задачи и данных, проанализированы существующие решения, рассмотрены типы пользователей и требуемый функционал. Представлен анализ способов хранения данных, а также произведен выбор оптимального для решения поставленной задачи подхода к хранению данных.

1.1 Формализация задачи

Необходимо спроектировать и реализовать базу данных для просмотра информации о времени прохождения игр. Также необходимо разработать интерфейс, позволяющий работать с данной базой для получения и изменения хранящейся в ней информации. Требуется реализовать, как минимум, три вида ролей – авторизованный пользователь, гость и администратор.

1.2 Формализация данных

База данных должна хранить информацию об:

- играх;
- отзывах об играх;
- временных отметках о времени прохождения игры;
- данные авторизации пользователей;
- платформах запуска игр;
- связи между игрой и платформами, на которых запускается игра.

В таблице 1.1 приведены категории и сведения о данных.

Таблица 1.1 – Категории и сведения о данных

Категория	Сведения
Игры	ID игры, название игры, дата выхода, разработчик, издатель, жанры
Временная отметка	ID игры, имя пользователя, количество часов, количество минут, тип прохождения.
Отзыв	ID игры, имя пользователя, текст отзыва, оценка, дата публикации.
Пользователь	Имя пользователя, пароль.
Платформа	ID платформы, название, средняя стоимость
Связь платформ и игр	ID платформы, ID игры.

На рисунке 1.1 отображена ER-диаграмма системы, основанная на приведенной выше таблице.

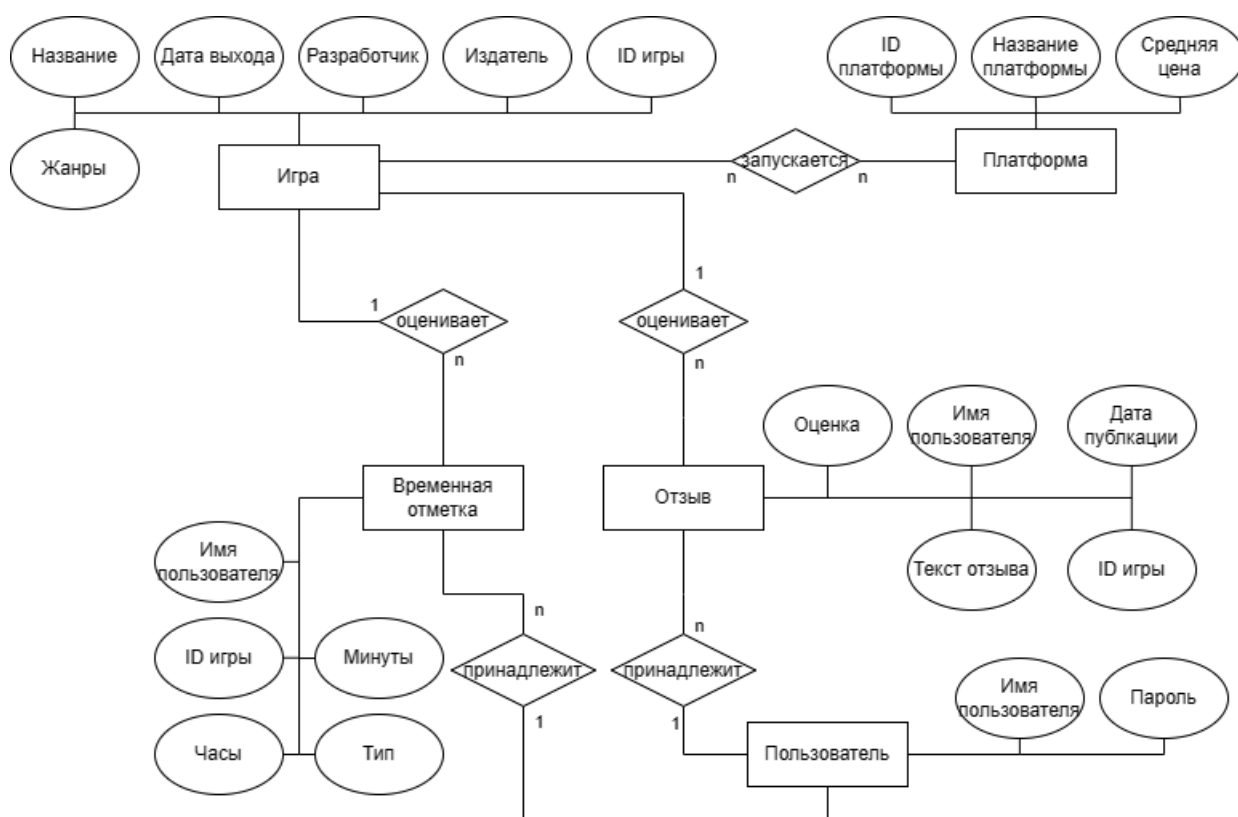


Рисунок 1.1 – ER-диаграмма

1.3 Типы пользователей

В соответствии с поставленной задачей необходимо разработать приложение с возможностью аутентификации пользователей, что делит их, прежде всего, на авторизованных и неавторизованных. Для управления приложением необходима ролевая модель: авторизованный (обычный) пользователь, неавторизованный пользователь (гость) и администратор.

Для каждого типа пользователя предусмотрен свой набор функций, который можно описать с помощью текста и Use Case Diagram (диаграммы прецедентов).

Набор функций неавторизованного пользователя:

- регистрация,
- аутентификация,
- просмотр различной статистики о времени прохождения игры по разным категориям,
- просмотр отзывов и оценок,
- просмотр информации об игре.

На рисунке 1.2 представлена диаграмма прецедентов для неавторизованного пользователя.

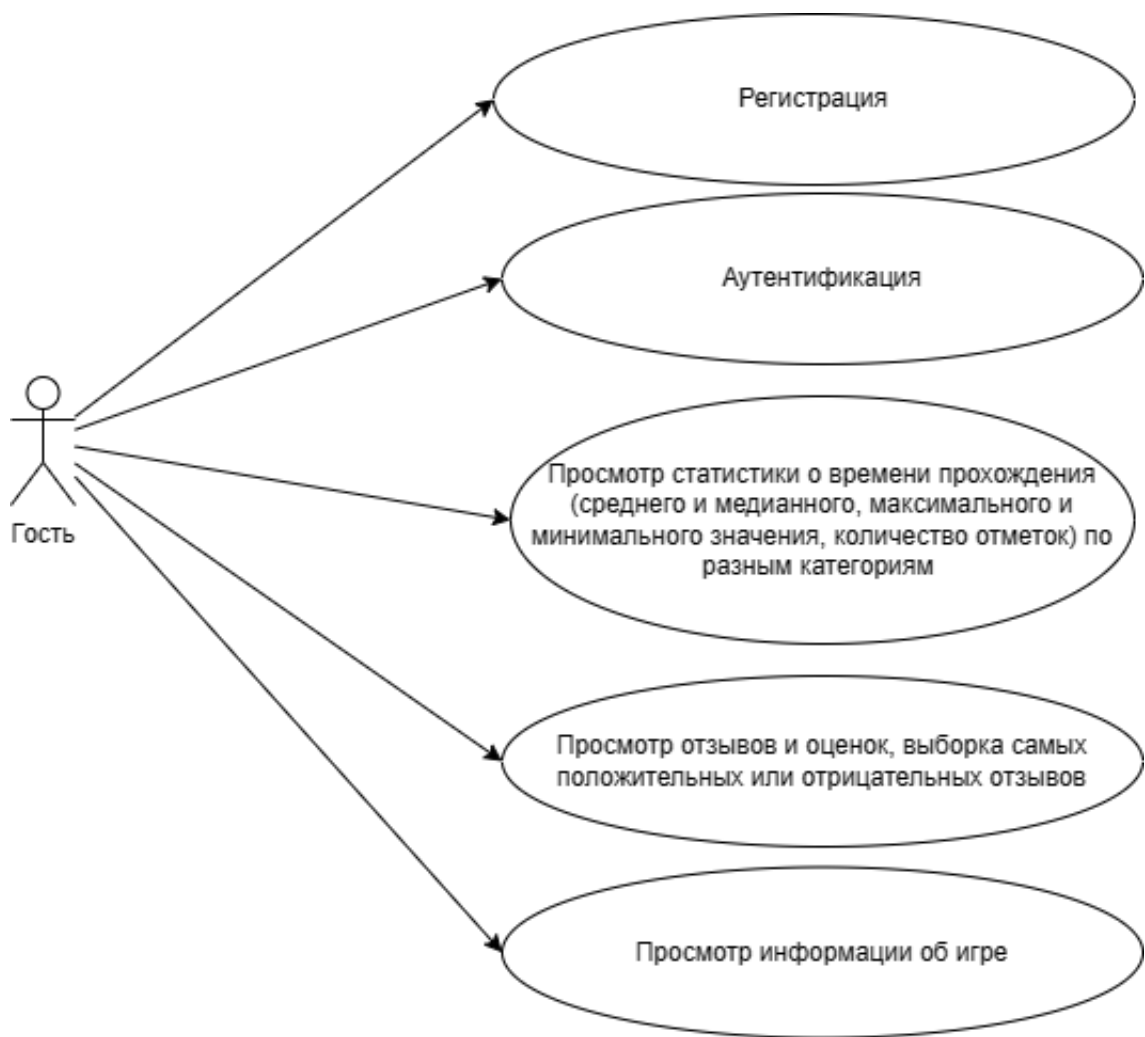


Рисунок 1.2 – Диаграмма прецедентов для неавторизованного пользователя

Набор функций авторизованного пользователя:

- выход,
- просмотр различной статистики о времени прохождения игры по разным категориям,
- добавление отзывов и оценок,
- добавление информации о времени прохождения игры,
- просмотр отзывов и оценок,
- просмотр информации об игре.

На рисунке 1.3 представлена диаграмма прецедентов для неавторизованного пользователя.



Рисунок 1.3 – Диаграмма прецедентов для авторизованного пользователя

Набор функций администратора:

- выход,
- просмотр и изменение всей информации, доступной в базе данных.

На рисунке 1.4 представлена диаграмма прецедентов для администратора.

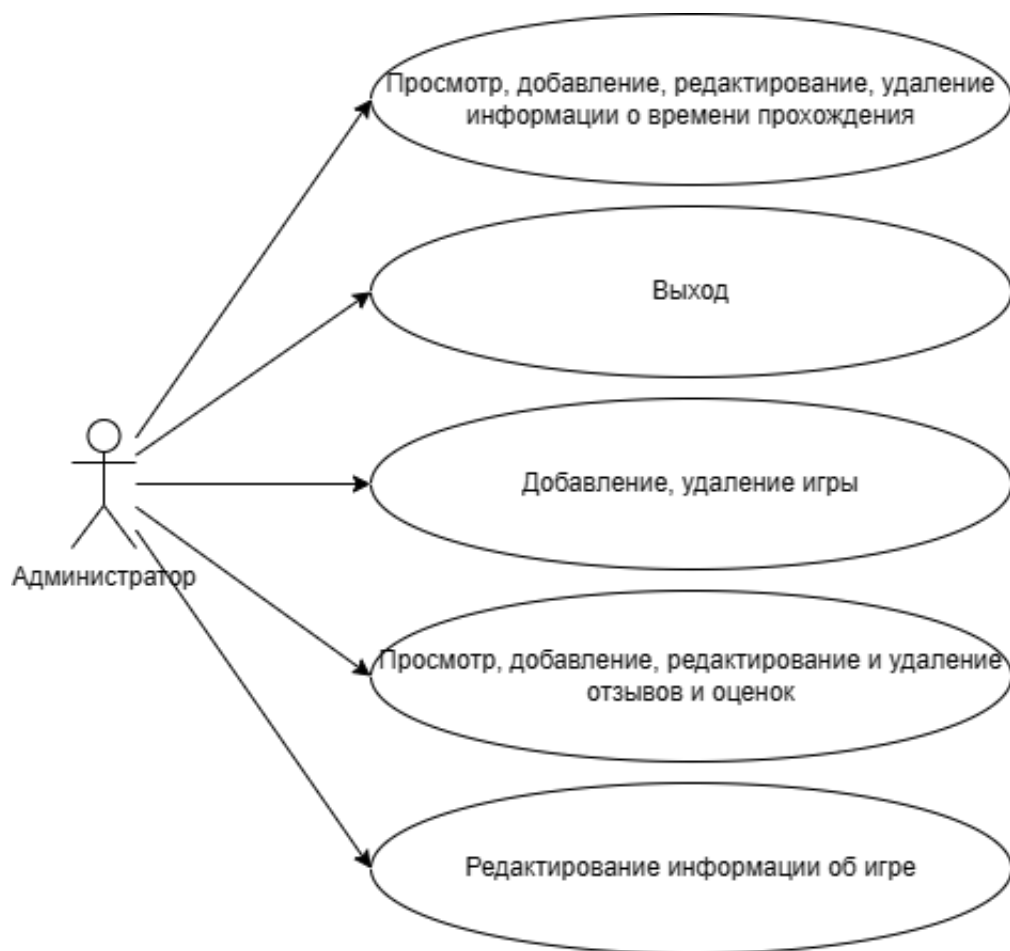


Рисунок 1.4 – Диаграмма прецедентов для администратора

1.4 Анализ существующих решений

1.4.1 Критерии

Проведем анализ аналогичных решений по следующим критериям:

1. Возможность указать время прохождения игры;
2. Возможность оставлять оценки и отзывы;
3. Наличие интерфейса на русском языке.

1.4.2 Steam

Steam — один из самых популярных магазинов видео игр. Он Позволяет не только покупать игры, но и оставлять обзоры на них. Также Steam считает время проведенное в игре, но не может собирать информацию времени прохождения (рисунок 1.5). Имеет русский интерфейс.

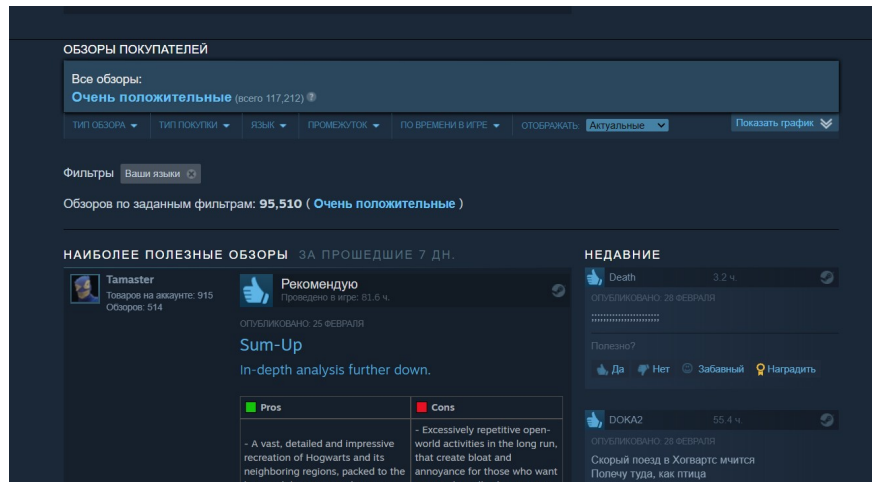


Рисунок 1.5 – Приложение Steam

1.4.3 Сайт HowLongToBeat

Сайт, на котором можно оставлять обзоры и оценки видеоигр, а также оставлять время прохождения игры по 2 категориям: основной сюжет, 100% прохождение (рисунок 1.6). Данный сайт имеет серьезный недостаток, он полностью на английском языке.

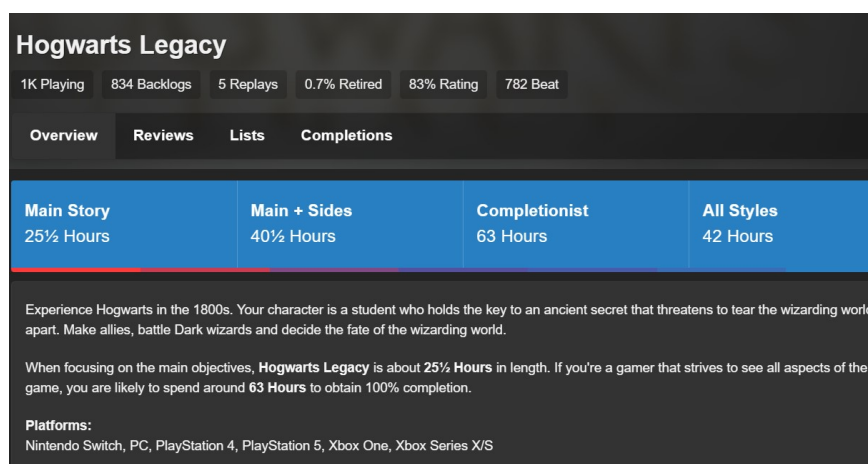


Рисунок 1.6 – Сайт HowLongToBeat

1.4.4 Сравнение существующих решений

В таблице 1.2 приведено сравнение решений по наличию в них информации, перечисленной в пункте 1.4.1. В таблице приняты следующие обозначения: время прохождения — возможность указать время прохождения игры, отзывы и оценки — возможность оставлять оценки и отзывы, русский язык — наличие интерфейса на русском языке.

Таблица 1.2 – Сравнение решений

Сервис	Время прохождения	Отзывы и оценки	Русский язык
Steam	-	+	+
HowLongToBeat	+	+	-

Таким образом, ни одно из существующих решений не отображает всю информацию, которую предполагается предоставлять в разрабатываемом приложении. В частности, в Steam не возможно посмотреть среднее время прохождения игры, а HowLongToBeat не имеет интерфейса на русском языке.

1.5 Анализ баз данных

Для решения поставленной цели необходимо разработать базу данных (БД). БД — это упорядоченный набор структурированной информа-

ции или данных, которые обычно хранятся в электронном виде в компьютерной системе [1].

Классификация реляционных баз данных по способу хранения

Реляционные базы данных по способу хранения делятся на две группы: строковые и колоночные.

Строковые базы данных

Строковыми базами данных называются такие базы данных, записи которых хранятся построчно, в основном такой тип используется в транзакционных системах (англ. OLTP [2]), для которых характерно большое количество коротких транзакций с операциями вставки, обновления и удаления данных. Системы OLTP наиболее подходят для быстрой обработки запросов и поддержания целостности данных в средах с множественным доступом.

Колоночные базы данных

Колоночными называются базы данных, записи которых хранятся по столбцам, в основном такой тип используется в аналитических системах (англ. OLAP [3]), которые характеризуются низким объёмом транзакций, а запросы часто являются составными и включают в себя агрегацию, мерой эффективности таких систем является время отклика. Наличие агрегированных и исторических данных позволяет использовать OLAP-системы для реализации методов интеллектуального анализа.

1.5.1 Кэширование данных

Чтобы уменьшить время отклика, можно использовать механизм кэширования данных, для реализации которого подходят NoSQL [4] in-memory базы данных, хранящие данные в оперативной памяти, что обеспечивает более быстрый доступ, по сравнению с традиционными реализациями.

Проблема синхронизации данных

Приложение пишет в кэш и базу данных, которые между собой никак не синхронизируются, таким образом возникает несогласованность данных. Например, возможна ситуация, когда данные удаляются из хранилища и их нужно удалить из кэша. Эту проблему можно решить установкой триггеров, которые будут срабатывать на изменение и удаление записей, чтобы синхронизировать данные в кэше.

Проблема «холодного старта»

Когда кэш только создаётся, он пуст и не имеет никаких данных – все запросы идут напрямую в БД, и только через некоторое время кэш будет содержать необходимую информацию. Эту проблему можно решить, выбрав СУБД с журналированием всех операций – при перезагрузке можно восстановить предыдущее состояние кэша с помощью журнала событий, который хранится на диске. При перезагрузке кэша нужно синхронизировать данные с хранилищем – возможно, какие-то данные, находящиеся в кэше, перестали быть актуальными за время его перезагрузки.

2 КОНСТРУКТОРСКАЯ ЧАСТЬ

В данном разделе представлены этапы проектирования системы: функциональная модель приложения, диаграмма базы данных.

2.1 Проектирование базы данных

В соответствии с ER-диаграммой системы, изображенной на рисунке 1.1, база данных должна хранить следующие таблицы:

- таблица с играми Games;
- таблица с временными отметками TimeRecords;
- таблица с отзывами Reviews;
- таблица с пользователями Users;
- таблица с платформами Platforms;
- таблица связка платформ и игр GamePlatform;

На рисунке 2.1 представлена диаграмма базы данных.

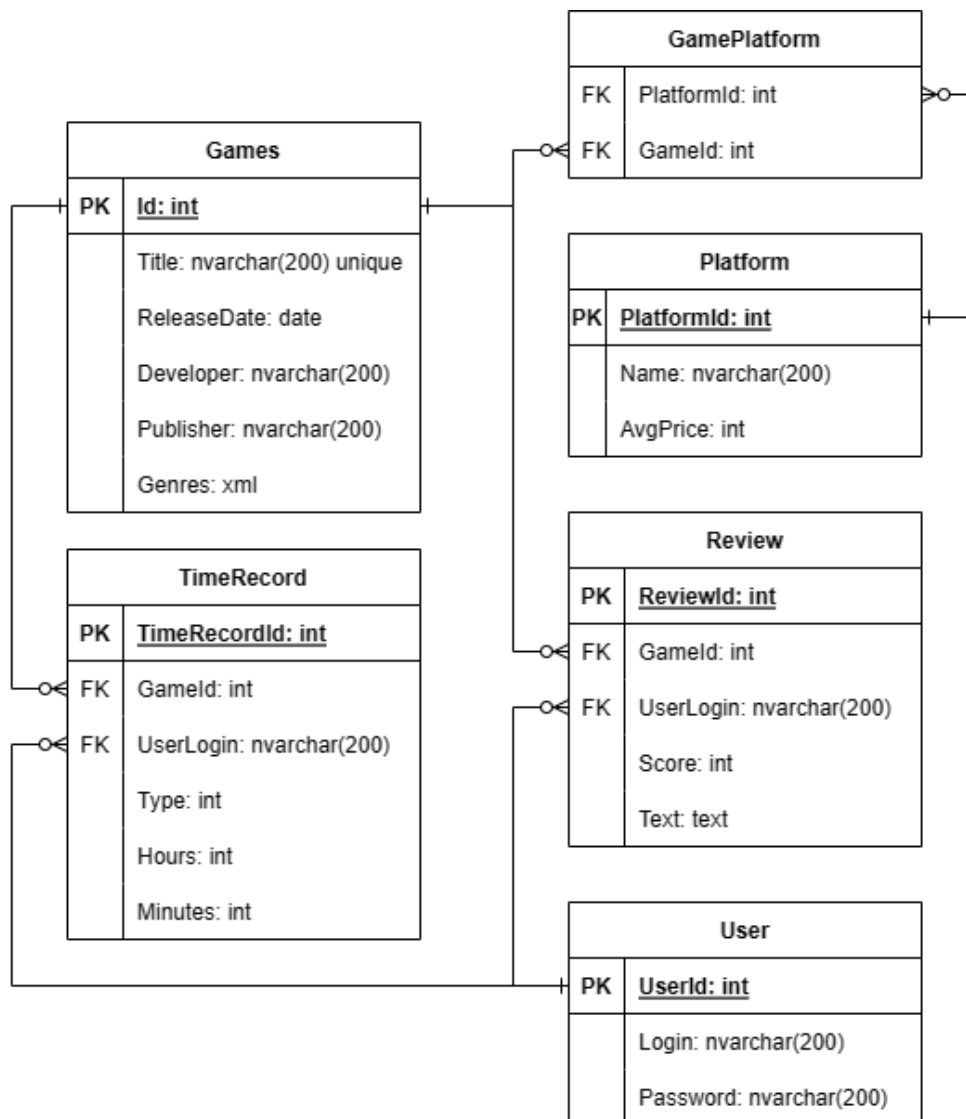


Рисунок 2.1 – Диаграмма сущность-связь базы данных

Таблица Games хранит информацию о играх и содержит следующие поля:

- Id — идентификатор игры, первичный ключ, int;
- Title — уникальное название игры, string;
- ReleaseDate — дата выхода игры, date;
- Developer — разработчик игры, string;
- Publisher — издатель, string;
- Genres — жанры, xml.

Таблица Platforms хранит информацию о платформах и содержит следующие поля:

- Id — идентификатор платформы, первичный ключ, int;
- Name — уникальное название платформы, string;
- AvgPrice — средняя цена платформы, int.

Таблица GamePlatform хранит информацию о связи игр и платформ и содержит следующие поля:

- GameId — идентификатор игры, int;
- PlatformId — идентификатор платформы, int.

Таблица TimeRecords хранит информацию о временных отметках на игре и содержит следующие поля:

- Id — идентификатор, int;
- GameId — идентификатор игры, int;
- UserLogin — имя пользователя который оставил эту отметку, string;
- Hours — количество полных часов потраченных на прохождение игры, int;
- Minutes — количество полных минут потраченных на прохождение игры, int;
- Type — тип прохождения, int.

Таблица Reviews хранит информацию об отзывах и содержит следующие поля:

- Id — идентификатор, int;
- GameId — идентификатор игры на которую оставлен отзыв;
- UserLogin — имя пользователя оставившего отзыв, string;

- Text — текст отзыва, string;
- PublicationDate — дата публикации отзыва, date.

Таблица Users хранит информацию о пользователях и содержит следующие поля:

- Login — имя пользователя, string;
- Password — пароль, string.

2.2 UML диаграмма классов приложения

На рисунке 2.2 изображена UML диаграмма классов компонента бизнес-логики и компонента доступа к базе данных.

3 ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

В этом разделе будет проведен выбор средств реализации ПО. Будет выбран язык программирования, система управления базой данных, ПО для графического интерфейса. Будут приведены листинги кода для реализованной базы данных и листинги реализации взаимодействия приложения и базы данных. Будет проведена демонстрация работы программы с реализованным функционалом для каждой роли.

3.1 Выбор СУБД для решения задачи

Система управления базами данных (СУБД) – это совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных [1].

В данной работе предполагается хранение в БД большого количества информации.

Рассмотрим несколько требований, которым должна удовлетворять СУБД для решения поставленной задачи.

1. Для хранения данных СУБД должна предоставлять такие типы данных как строки, целые числа, даты и xml.
2. Для загрузки xml файла из приложения в СУБД должна быть реализована возможность создания хранимых процедур.
3. Необходимо надежное хранение данных, без риска их потери даже в случае сбоя в системе.

В качестве СУБД был выбран SQL Server, так как он удовлетворяет перечисленным требованиям.

3.2 Выбор инструментов разработки

В качестве языка программирования выбран язык C# [5]. Во-первых, он объектно-ориентирован, что позволит использовать наследование, ин-

терфейсы, абстракции и так далее. Во-вторых, в стандартной библиотеке C# имеется пространство имен `System.Data.Linq.Mapping` [6], которое содержит классы, используемые для формирования объектной модели LINQ to SQL, которая представляет структуру и содержимое реляционной базы данных.

Для упаковки приложения в готовый продукт была выбрана система контейнеризации Docker [7]. С его помощью можно создать изолированную среду для программного обеспечения, которое можно будет развернуть на различных устройствах без дополнительных настроек.

3.3 Детали реализации

3.3.1 Ролевая модель

Необходимо реализовать ролевую модель в соответствии с пунктом 1.3. Роль — это разрешение, предоставляемое группе пользователей для доступа к данным.

Ролевая модель реализована на нескольких уровнях, в том числе, на уровне базы данных, что продемонстрировано в листинге 3.1. Последовательно создаются и наделяются правами роли неавторизованного пользователя, авторизованного пользователя, сотрудника лыжного патруля и администратора, каждый наследует и расширяет права предыдущего.

Листинг 3.1 – Ролевая модель на уровне БД

```
1 create login [AuthorizedUser] with password = 'UserPassword1'
2 create user AU for login [AuthorizedUser]
3 create role [AuthorizedUserRole]
4 alter role [AuthorizedUserRole] add member AU
5 deny delete on Reviews to [AuthorizedUserRole];
6 deny update on Reviews to [AuthorizedUserRole];
7 deny delete on TimeRecords to [AuthorizedUserRole];
8 deny update on TimeRecords to [AuthorizedUserRole];
9 deny insert on Games to [AuthorizedUserRole];
10 deny delete on Games to [AuthorizedUserRole];
11 deny update on Games to [AuthorizedUserRole];
12 deny delete on Users to [AuthorizedUserRole];
```

```

13 deny update on Users to [AuthorizedUserRole];
14 deny alter to [AuthorizedUserRole];
15 grant insert on Reviews to [AuthorizedUserRole];
16 grant insert on TimeRecords to [AuthorizedUserRole];
17 grant insert on Users to [AuthorizedUserRole];
18 grant select on Users to [AuthorizedUserRole];
19 grant select on Games to [AuthorizedUserRole];
20 grant select on Reviews to [AuthorizedUserRole];
21 grant select on TimeRecords to [AuthorizedUserRole];
22
23 create login [GuestUser] with password = 'GuestPassword1'
24 create user GU for login [GuestUser]
25 create role [GuestUserRole]
26 alter role [GuestUserRole] add member GU
27 deny delete on Reviews to [GuestUserRole];
28 deny update on Reviews to [GuestUserRole];
29 deny delete on TimeRecords to [GuestUserRole];
30 deny update on TimeRecords to [GuestUserRole];
31 deny insert on Games to [GuestUserRole];
32 deny delete on Games to [GuestUserRole];
33 deny update on Games to [GuestUserRole];
34 deny delete on Users to [GuestUserRole];
35 deny update on Users to [GuestUserRole];
36 deny alter to [GuestUserRole];
37 deny insert on Reviews to [GuestUserRole];
38 deny insert on TimeRecords to [GuestUserRole];
39 grant insert on Users to [GuestUserRole];
40 grant select on Users to [GuestUserRole];
41 grant select on Games to [GuestUserRole];
42 grant select on Reviews to [GuestUserRole];
43 grant select on TimeRecords to [GuestUserRole];

```

3.3.2 Реализация процедуры преобразования строки в xml формат и вставки в таблицу

В листинге 3.2 представлена реализация процедуры преобразования строки с перечислением жанров в xml формат и вставки в таблицу Games на стороне базы данных.

Листинг 3.2 – Процедура преобразования строки в xml формат и вставки в таблицу на стороне базы данных

```
1 create or alter procedure InsertGameProcedure
2 (
3     @title nvarchar(200),
4     @releaseDate date,
5     @developer nvarchar(200),
6     @publisher nvarchar(200),
7     @genres nvarchar(200)
8 )
9 as
10 begin
11     delete temp where 0 = 0
12     insert into temp
13     select @title, genre_split.value
14     from string_split(@genres, '␣') as genre_split
15
16     insert into Games
17     values(@title, @releaseDate, @developer, @publisher,
18     (
19         select temp.genre
20         from temp
21         for xml path(''), type, root('Genre')
22     ))
23     delete temp where 0 = 0
24 end
```

В листинге 3.3 приведена реализация процедуры преобразования строки с перечислением жанров в xml формат и вставки в таблицу Games на стороне приложения.

Листинг 3.3 – Метод преобразования строки в xml формат и вставки в таблицу на стороне приложения

```
1 public void AddGameXMLApp(Game game)
2 {
3     DataContext db = new DataContext(adminConnection);
4     Table<GamesXML> gameTable = db.GetTable<GamesXML>();
5     XmlDocument xml = GetXml(game.Genres);
6     GamesXML g = new GamesXML()
7     {
8         Title = game.Title,
```

```

9         ReleaseDate = game.ReleaseDate ,
10         Developer = game.Developer ,
11         Publisher = game.Publisher ,
12         Genres = xml.InnerXml ,
13     };
14     gameTable.InsertOnSubmit(g);
15     try
16     {
17         db.SubmitChanges();
18     }
19     catch (DuplicateKeyException e)
20     {
21         throw new GameAlreadyExistsException(e.Message);
22     }
23 }
24
25 XmlDocument GetXml(string genres)
26 {
27     string[] genreSplit = genres.Split(' ');
28     XmlDocument xml = new XmlDocument();
29     XmlElement root = xml.CreateElement("Genres");
30     xml.AppendChild(root);
31     foreach (var genre in genreSplit)
32         xml["Genres"].AppendChild(xml.CreateElement("genre"))
33             .InnerText = genre;
34     return xml;
35 }

```

3.3.3 Модели хранения данных

В листинге 3.4 продемонстрировано создание таблицы Reviews в базе данных.

Листинг 3.4 – Создание таблицы Reviews

```

1 create table Reviews
2 (
3     Gameld int ,
4     UserLogin nvarchar(200) ,
5     [Text] nvarchar(1000) ,

```

```

6     Score int ,
7     PublicationDate date ,
8     primary key(Gameld , UserLogin)
9 );

```

В листинге 3.5 приведена реализация модели Review, используемой в компоненте бизнес-логики.

Листинг 3.5 – Модель Review

```

1 public class Review
2 {
3     public int Gameld { get; }
4     public string UserLogin { get; }
5     public uint Score { get; }
6     public string Text { get; }
7     public DateTime PublicationDate { get; }
8     public Review(int gameld, string userLogin, uint score,
9         string text, DateTime publicationDate)
10    {
11        Gameld = gameld;
12        UserLogin = userLogin;
13        Score = score;
14        Text = text;
15        PublicationDate = publicationDate;
16    }
17 }

```

В листинге 3.6 приведена реализация репозитория, который обеспечивает доступ к данным таблицы Reviews из базы данных.

Листинг 3.6 – Реализация репозитория для доступа к данным таблицы Reviews из базы данных

```

1 [Table]
2 public class Reviews
3 {
4     [Column(IsPrimaryKey = true)]
5     public int Gameld { get; set; }
6     [Column(IsPrimaryKey = true)]
7     public string UserLogin { get; set; }
8     [Column]
9     public string Text { get; set; }
10    [Column]

```

```

11     public int Score { get; set; }
12     [Column]
13     public DateTime PublicationDate { get; set; }
14 }
15
16 public class SqlServerReviewRepository : SqlServerRepository ,
    IRepository
17 {
18     public SqlServerReviewRepository() : base() { }
19
20     public void AddReview(Review review)
21     {
22         DataContext db = new DataContext(userConnection);
23         Table<Reviews> reviewTable = db.GetTable<Reviews>();
24         Reviews r = new Reviews()
25         {
26             Gameld = review.Gameld,
27             UserLogin = review.UserLogin,
28             Score = (int)review.Score,
29             Text = review.Text,
30             PublicationDate = review.PublicationDate
31         };
32         reviewTable.InsertOnSubmit(r);
33         db.SubmitChanges();
34     }
35
36     public void DeleteReview(Review review)
37     {
38         DataContext db = new DataContext(adminConnection);
39         Reviews rev = db.GetTable<Reviews>().Where(r => r.Gameld
            == review.Gameld && r.UserLogin ==
            review.UserLogin).First();
40         db.GetTable<Reviews>().DeleteOnSubmit(rev);
41         db.SubmitChanges();
42     }
43
44     public List<Review> GetReviews(int id)
45     {
46         DataContext db = new DataContext(guestConnection);
47         IQueryable<Reviews> gameReviews = from r in
            db.GetTable<Reviews>()

```

```
48     where r.GameId == id
49     select r;
50     List<Review> reviews = new List<Review>();
51     foreach (var rev in gameReviews)
52         reviews.Add(new Review(rev.GameId, rev.UserLogin,
53                                 (uint)rev.Score, rev.Text, rev.PublicationDate));
53     return reviews;
54 }
55 }
```

3.4 Интерфейс приложения

На рисунке 3.1 показан интерфейс приложения при запуске, библиотека видеоигр. Пользователь не авторизован. Пользователь может только авторизоваться, зарегистрироваться и посмотреть информацию и статистику об игре.

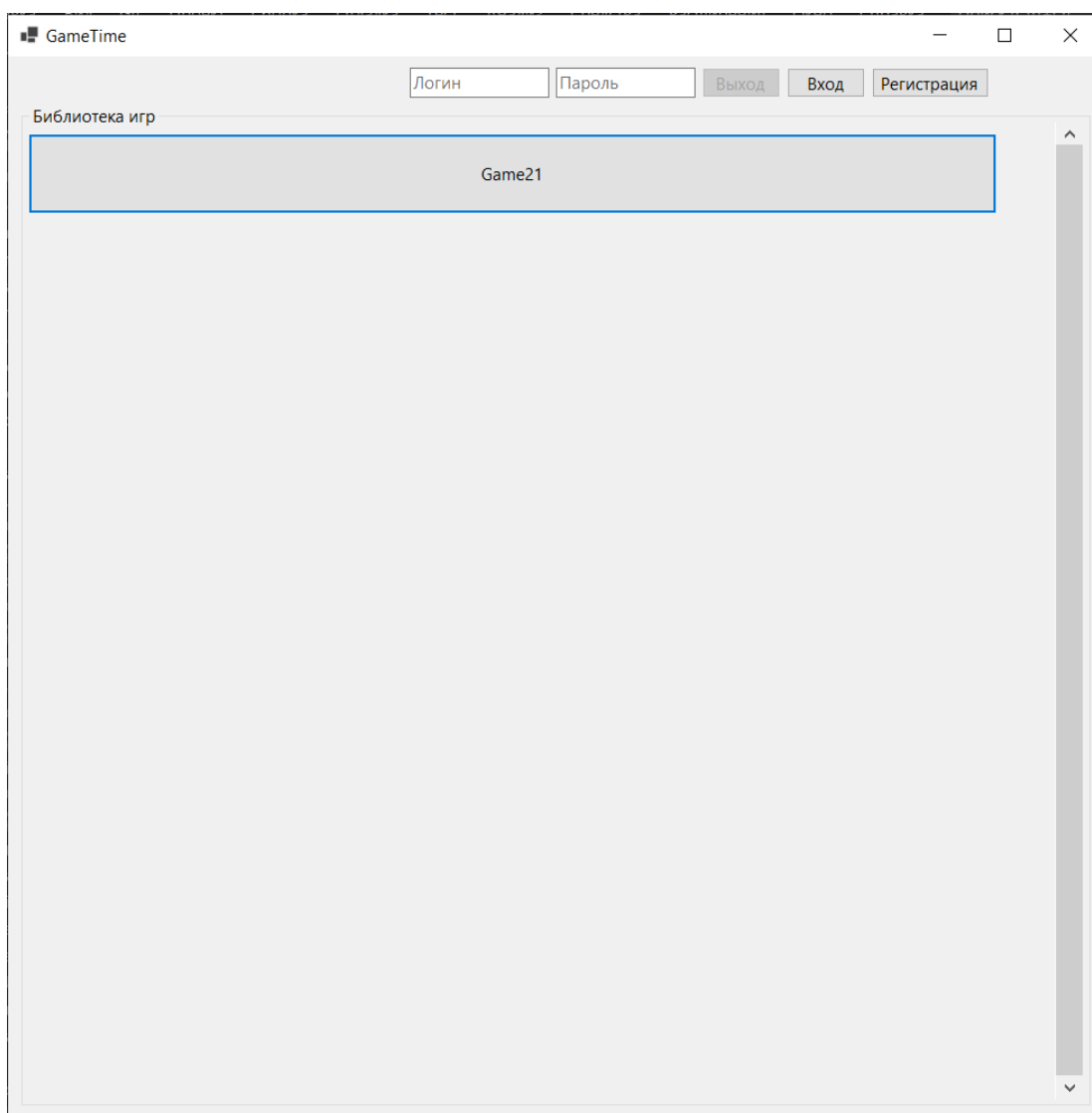


Рисунок 3.1 – Библиотека видеоигр

На рисунках 3.2, 3.3, 3.4 показана страница игры, которая доступна неавторизованному, авторизованному пользователю и администратору соответственно.

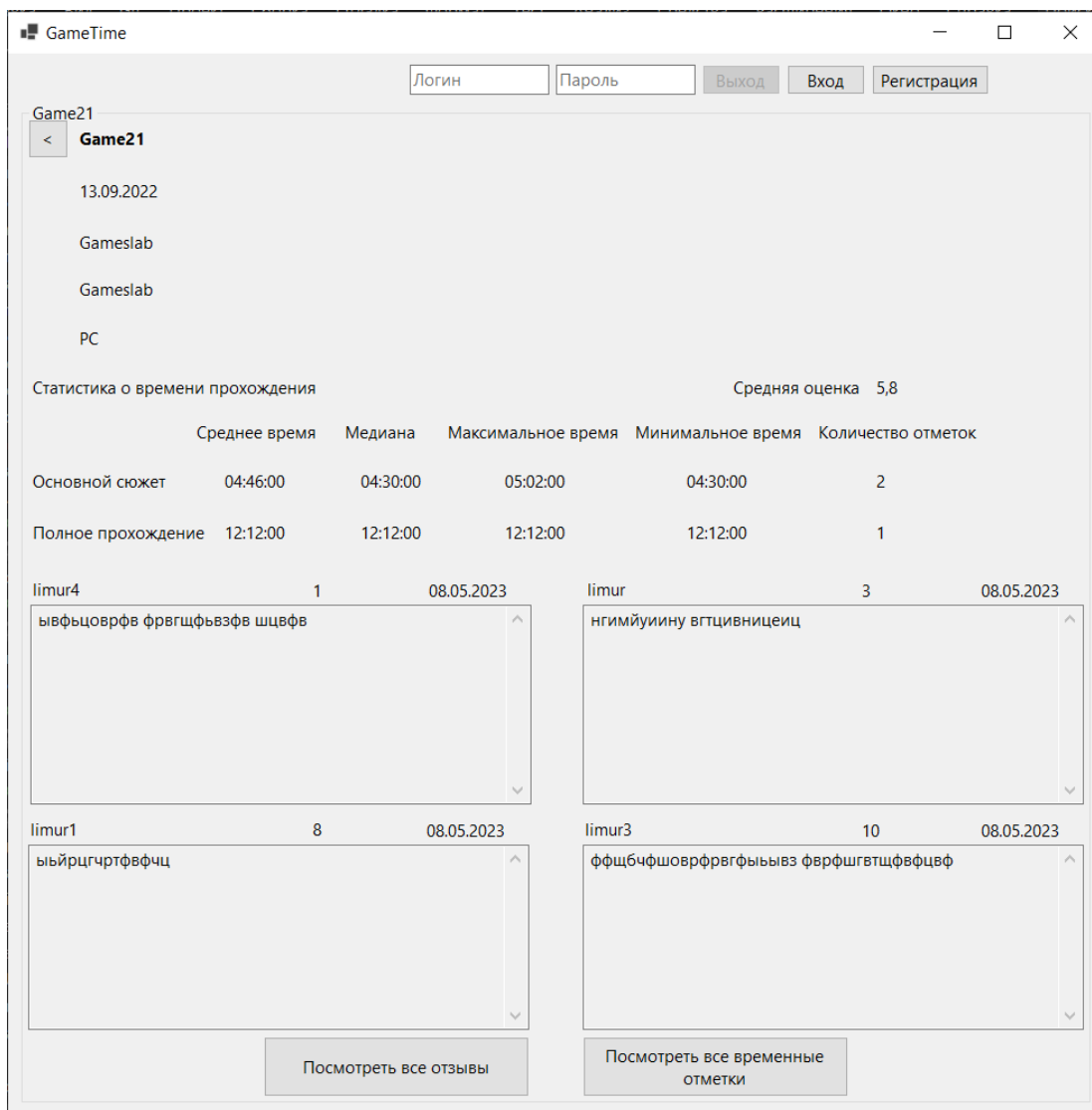


Рисунок 3.2 – Страница игры (пользователь не авторизован)

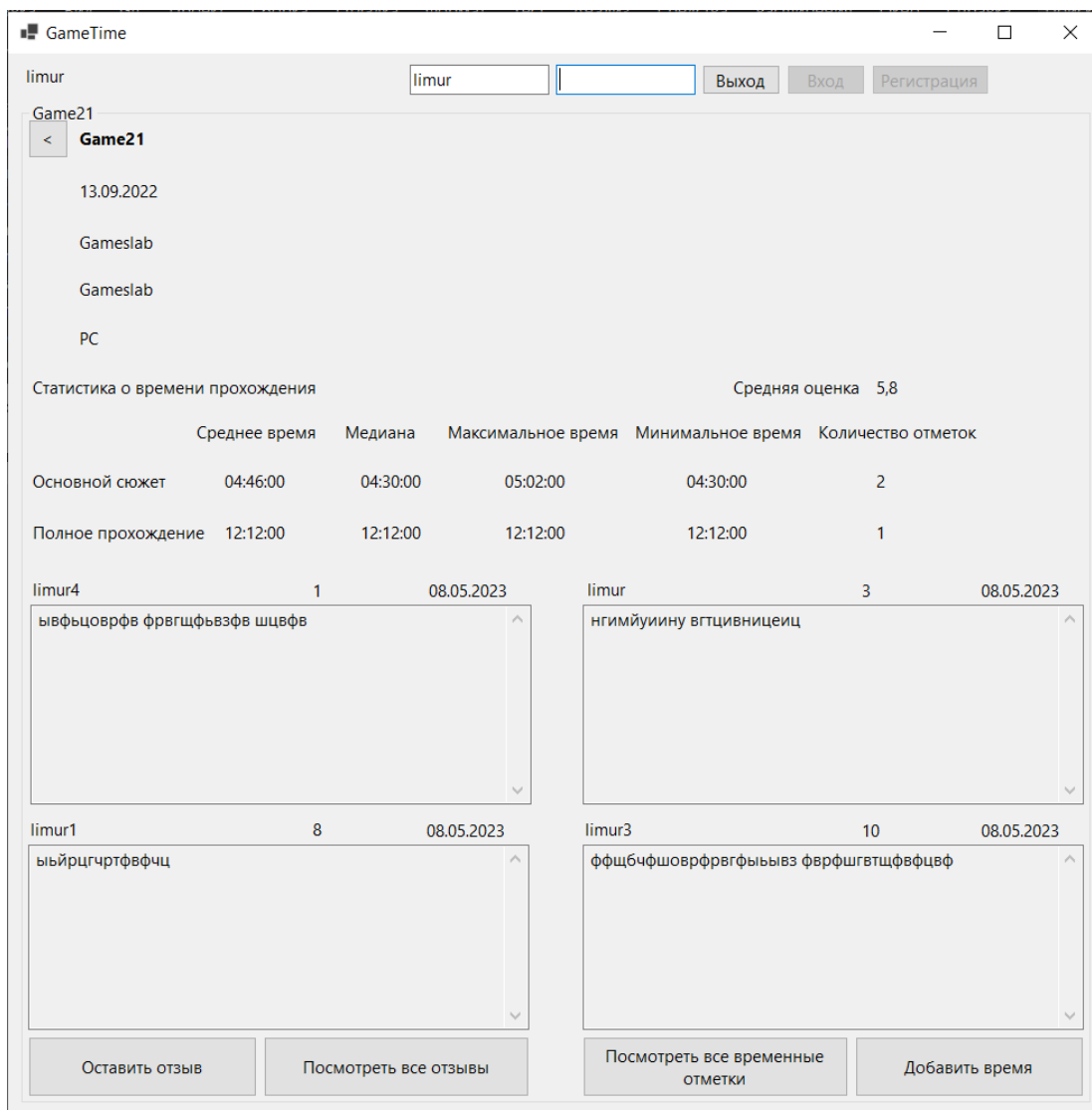


Рисунок 3.3 – Страница игры (пользователь авторизован)

GameTime

Логин

Пароль

Выход

Вход

Регистрация

Добавить игру

Game21

<

Game21

Удалить игру

Изменить игру

13.09.2022
Gameslab
Gameslab
PC

Статистика о времени прохождения

Средняя оценка 5,8

	Среднее время	Медиана	Максимальное время	Минимальное время	Количество отметок
Основной сюжет	04:46:00	04:30:00	05:02:00	04:30:00	2
Полное прохождение	12:12:00	12:12:00	12:12:00	12:12:00	1

limur4

1

08.05.2023

ывфьцоврфв фвгщфьвзфв шцвфв

limur

3

08.05.2023

нгимйуину вгтцивницеиц

limur1

8

08.05.2023

ыййрцгчртфвфцц

limur3

10

08.05.2023

ффщбчфшоврфвгфгыывз фврфшгвтщфвфцвф

Оставить отзыв

Посмотреть все отзывы

Посмотреть все временные отметки

Добавить время

Рисунок 3.4 – Страница игры (администратор)

На рисунке 3.5 показан вид окна для добавления временной отметки игре.

Добавить время

1

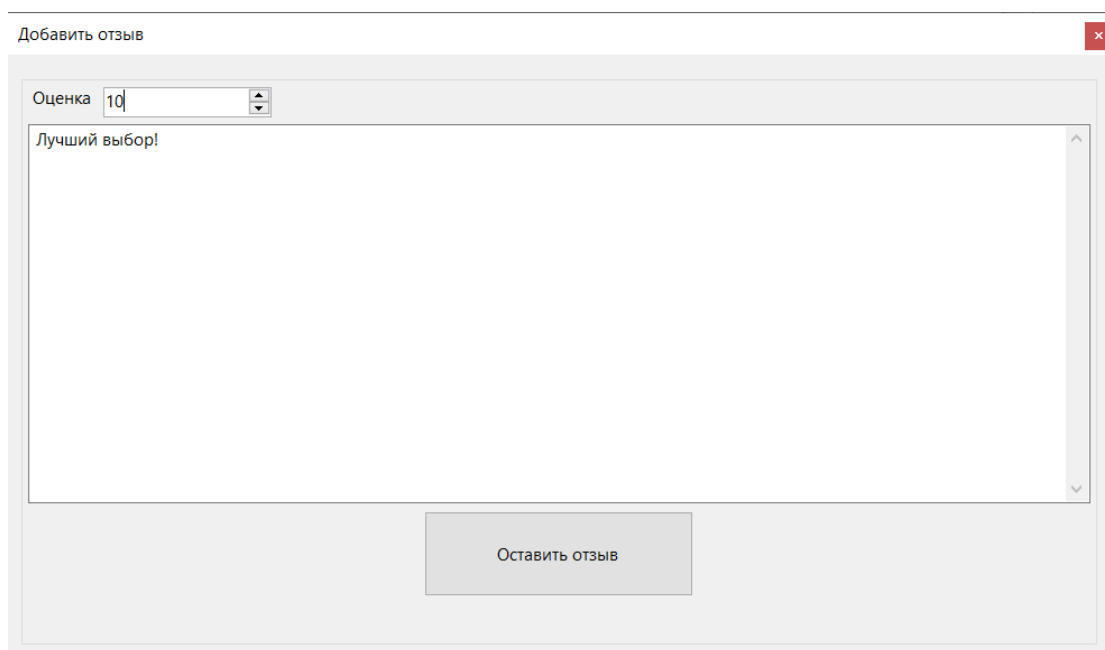
30

Обычное прохождение

Добавить игре временную отметку

Рисунок 3.5 – Окно для добавления временной отметки игре

На рисунке 3.6 показан вид окна для добавления отзыва об игре.



Добавить отзыв

Оценка 10

Лучший выбор!

Оставить отзыв

Рисунок 3.6 – Окно для добавления отзыва об игре

На рисунке 3.7 показан вид окна, где можно посмотреть все временные отметки игры. Администратору доступно удаление конкретной (рисунок 3.8).

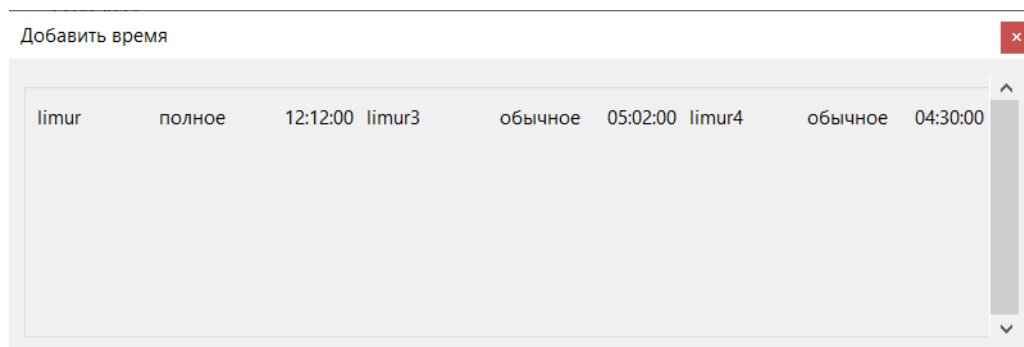


Рисунок 3.7 – Все временные отметки игры

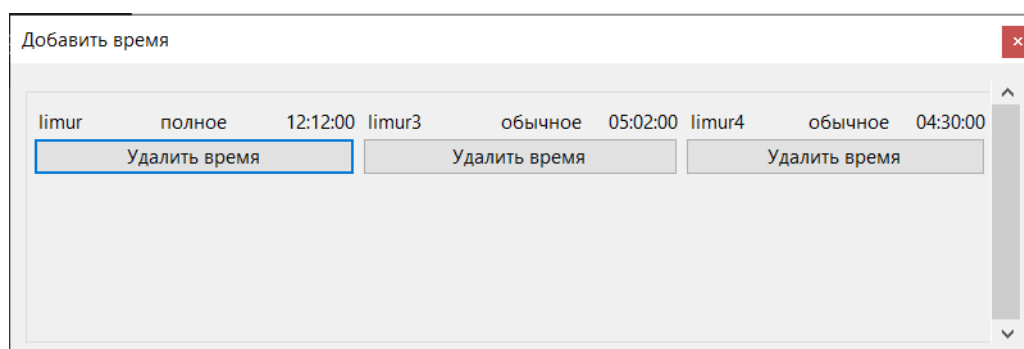


Рисунок 3.8 – Все временные отметки игры (администратор)

На рисунке 3.9 показан вид окна, где можно посмотреть все отзывы игры. Администратору доступно удаление конкретного (рисунок 3.10).

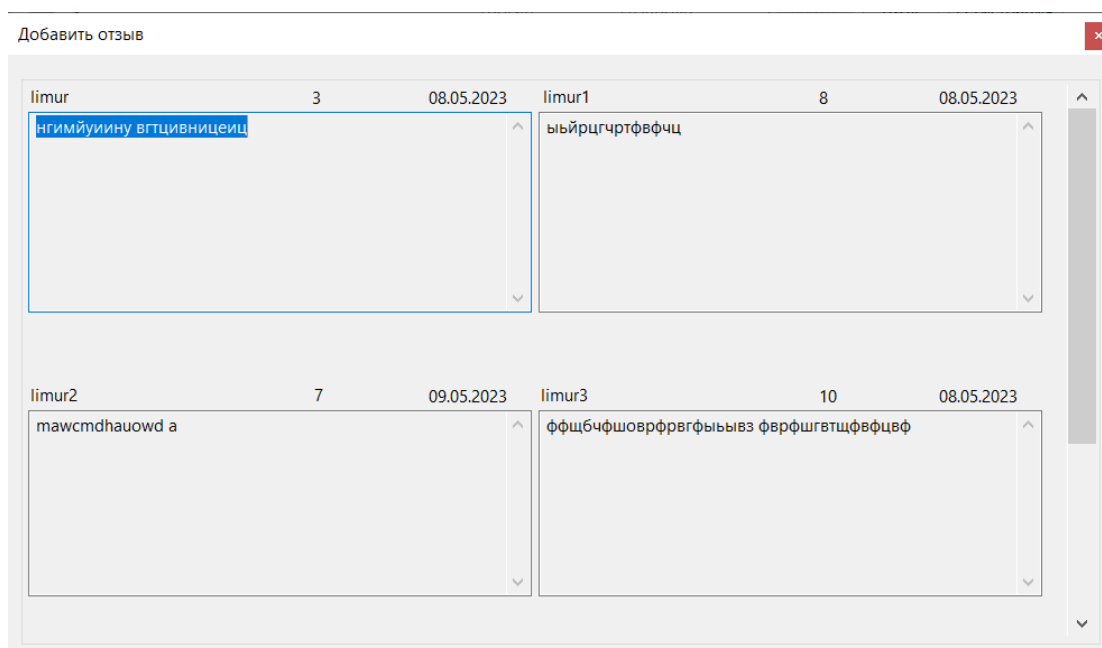


Рисунок 3.9 – Все отзывы игры

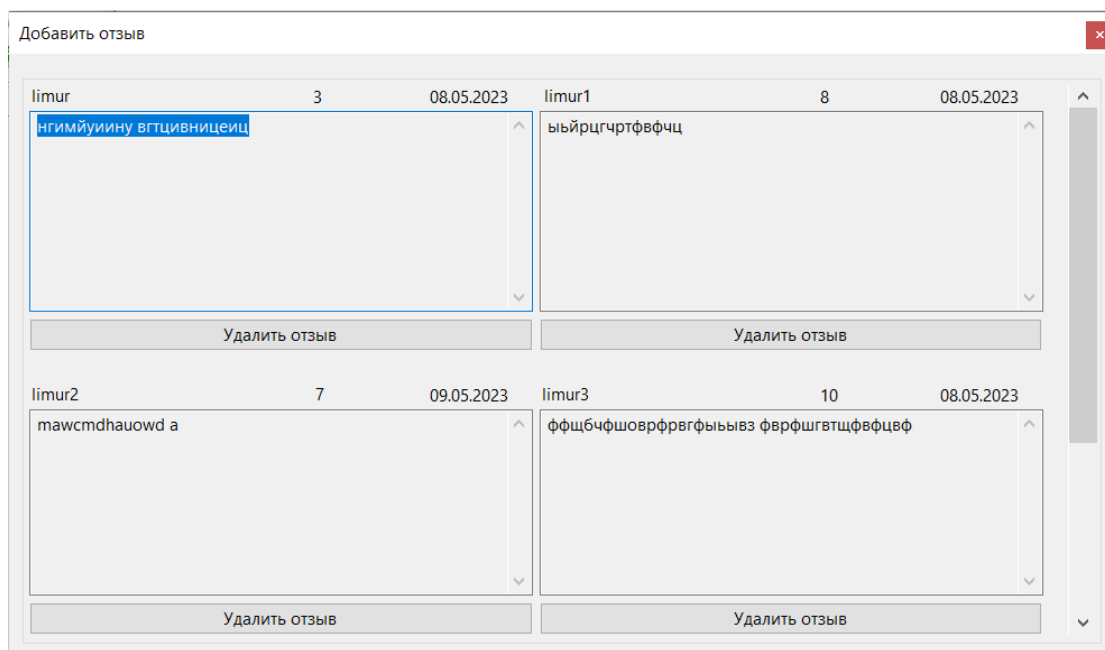


Рисунок 3.10 – Все отзывы игры (администратор)

На рисунке 3.11 изображено окно для добавления или изменения игры.

The screenshot shows a window titled 'Добавить игру' (Add Game) with a close button in the top right corner. It contains several input fields for game information and a large 'Добавить игру' (Add Game) button at the bottom.

Field	Value
Название	
Разработчик	
Издатель	
Жанры	
Дата	9 мая 2023 г.
Button	Добавить игру

Рисунок 3.11 – Окно добавления игры

4 ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ

В данном разделе представлены характеристики компьютера, на котором проводилось исследование. Также в данном разделе приведены результаты сравнения времени вставки игры в таблицу в зависимости от количества игр и от того, где происходит преобразование строки, содержащей информацию о жанрах, в xml-формат на стороне приложения или на стороне базы данных.

4.1 Цель эксперимента

Целью эксперимента является исследование зависимости вставки игры в таблицу в зависимости от количества игр и от того, где происходит преобразование строки, содержащей информацию о жанрах, в xml-формат на стороне приложения или на стороне базы данных.

4.2 Технические характеристики

Технические характеристики устройства, на котором выполнялись измерения:

- операционная система — Windows 10;
- оперативная память — 24 Гб;
- процессор — AMD Ryzen 5 4600H с Radeon Graphics 3.00 ГГц [8].

Во время тестирования ноутбук был включен в сеть питания и нагружен только приложениями встроенными в операционную систему.

4.3 Описание эксперимента

В эксперименте сравнивается время вставки игры в таблицу в зависимости от количества игр и от того, где происходит преобразование строки, содержащей информацию о жанрах, в xml-формат. Для каждого

количества игр и каждой реализации замеры производятся 50 раз, а затем вычисляется среднее арифметическое.

4.3.1 Используемые данные

В поставленном эксперименте для реализации каждого алгоритма измеряется время вставки игр в таблицу при их различном количестве: 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000. Каждой игре при этом добавляется 20 жанров.

4.4 Результаты эксперимента

В таблице 4.1 представлены результаты поставленного эксперимента.

Таблица 4.1 – Результаты сравнения времени (мс), необходимого для вставки игры в таблицу в зависимости от их количества и используемого алгоритма

Количество игр	Реализация алгоритма	
	на стороне базы данных	на стороне приложения
100	1116,7	678,8
200	1347,0	1358,1
300	2045,2	2585,2
400	4576,3	4851,6
500	9162,1	8373,2
600	10861,2	13313,7
700	14750,6	18671,4
800	15918,4	21309,9
900	23513,6	31194,5
1000	39439,7	45007,0

На рисунке 4.1 представлены графики зависимости времени вставки игр в таблицу от их количества и от используемого алгоритма.

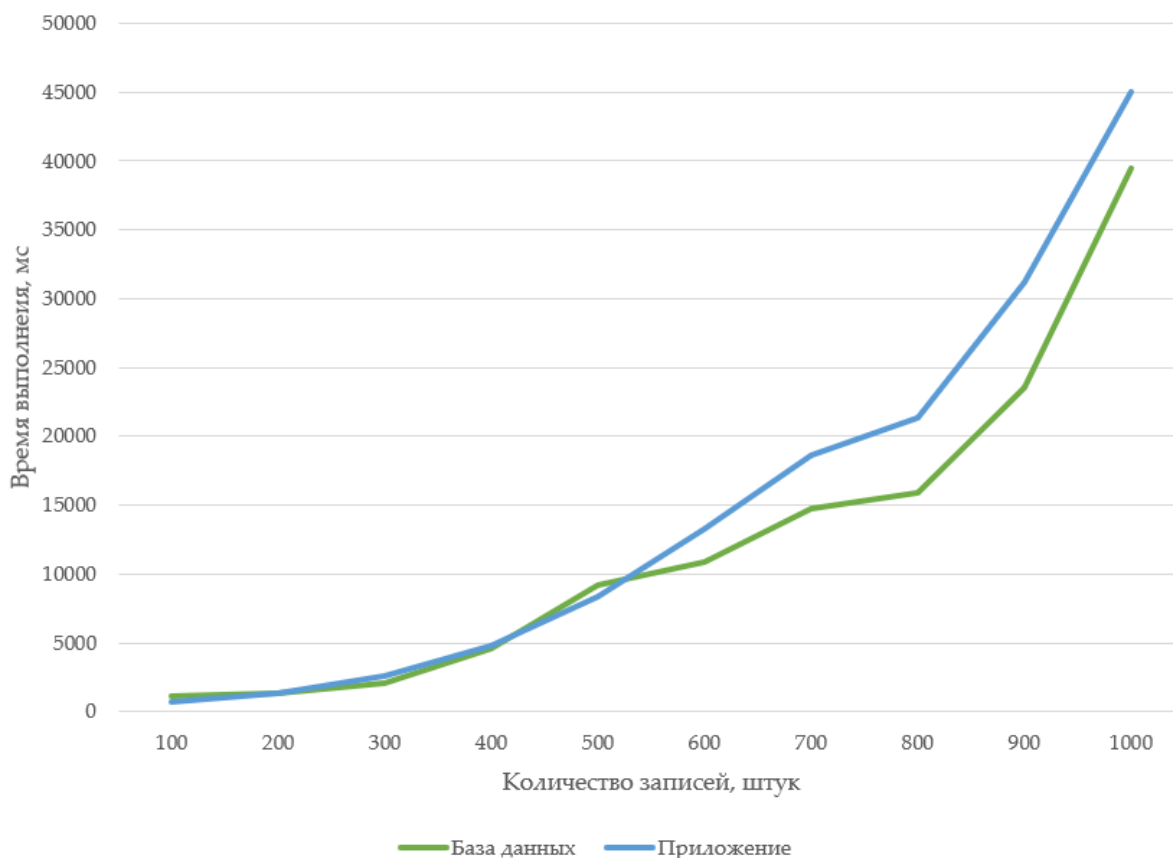


Рисунок 4.1 – Зависимость времени вставки игр в таблицу от их количества и от используемого алгоритма

По результатам эксперимента видно, что вставка игр в таблицу с алгоритмом преобразования строки в xml формат, реализованным на стороне базы данных, эффективнее по времени, чем вставка игр с алгоритмом реализованном на стороне приложения. При вставке 800 игр реализация на стороне базы данных быстрее реализации на стороне приложения примерно на 25%.

При реализации вставки на стороне базы данных в процедуру передается одна запись и все вычисления производит СУБД, в то время как при реализации на стороне приложения необходимо сначала получить таблицу в которую будет произведена вставка, затем преобразовать строку в xml формат и в конце вставить запись в таблицу.

4.5 Вывод

Исследование показало, что при фиксированном игровом времени вставки их в таблицу больше при реализации её на стороне приложения. Большое влияние на время оказывает не то, где производятся основные шаги вычислений (на стороне базы данных или на стороне приложения), а объем данных, передаваемых между приложением и базой данных.

ЗАКЛЮЧЕНИЕ

Цель курсовой работы достигнута, все поставленные задачи решены. Была разработана база данных для просмотра информации о времени прохождения игр

В ходе выполнения работы были изучены способы хранения данных, рассмотрены существующие СУБД, получены знания в области проектирования базы данных и приложений. Также было реализовано программное обеспечение, позволяющее получать доступ к данным.

В ходе выполнения экспериментально-исследовательской части было установлено, что на время выполнения операций, требующих обращения к БД большее влияние оказывает не то, где производятся основные шаги вычислений (на стороне БД или на стороне приложения), а объем данных, передаваемых между приложением и БД.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Что такое база данных | Oracle Россия и СНГ [Электронный ресурс]. Режим доступа: <https://www.oracle.com/ru/database/what-is-database/> (дата обращения: 25.03.2023).
2. What is OLTP? | IBM [Электронный ресурс]. Режим доступа: <https://www.ibm.com/cloud/learn/oltp> (дата обращения: 25.03.2023).
3. What is OLAP? | IBM [Электронный ресурс]. Режим доступа: <https://www.ibm.com/cloud/learn/olap> (дата обращения: 25.03.2023).
4. Что такое NoSQL? | Amazon AWS [Электронный ресурс]. Режим доступа: <https://aws.amazon.com/ru/nosql/> (дата обращения: 25.03.2023).
5. Документация C# [Электронный ресурс]. Режим доступа: <https://docs.microsoft.com/ru-ru/dotnet/csharp/> (дата обращения: 03.04.2023).
6. System.Data.Linq.Mapping Пространство имен [Электронный ресурс]. Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/api/system.data.linq.mapping?view=netframework-4.8> (дата обращения: 25.03.2023).
7. Docker: Empowering App Development for Developers [Электронный ресурс]. Режим доступа: <https://www.docker.com/> (дата обращения: 03.04.2023).
8. AMD Ryzen™ 5 4600H [Электронный ресурс]. Режим доступа: <https://www.amd.com/ru/products/apu/amd-ryzen-5-4600h> (дата обращения: 25.03.2023).