

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ
по дисциплине «Введение в нереляционные базы данных»
Тема: Агрегатор отзывов на игры

Студенты гр. 2384

Вовченко С.Е.
Валеева А.А.
Поглазов Н.В.
Кузьминых Е.М.

Студент гр. 2383

Борисов И.П.

Преподаватель

Заславский М.М.

Санкт-Петербург

2025

ЗАДАНИЕ

Студенты

Вовченко С.Е. 2384

Валеева А.А. 2384

Поглазов Н.В. 2384

Кузьминых Е.М. 2384

Борисов И.П. 2383

Тема проекта: Разработка приложения для просмотра и создания отзывов на игры.

Исходные данные:

Приложение необходимо реализовать с использованием нереляционной базы данных, нами была выбрана MongoDB.

Содержание пояснительной записи:

«Содержание»

«Введение»

«Сценарий использования»

«Модель данных»

«Разработанное приложение»

«Выводы»

«Приложение»

Предполагаемый объем пояснительной записи:

Не менее 10 страниц

Дата выдачи задания: 11.02.2025

Дата сдачи реферата: 21.05.2025

Студенты гр. 2384

Студент гр. 2383

Преподаватель

Вовченко С.Е.
Валеева А.А.
Поглазов Н.В.
Кузьминых Е.М.

Борисов И.П.

Заславский М.М.

АННОТАЦИЯ

В рамках данного курса предполагалось разработать веб-приложение с использованием нереляционной базы данных. Была выбрана своя тема, так как она показалась нам актуальной и имеет уже некоторое количество референсов (популярных агрегаторов отзывов).

Во внимание будут приниматься такие аспекты как user friendly интерфейс и понятная архитектура.

Исходный код и дополнительную информацию можно найти по ссылке: <https://github.com/moevm/nosql1h25-review>

Оглавление

1. Введение.....	6
2. Сценарии использования.....	8
2.1 Макет UI.....	8
2.2 Сценарии использования для задачи импорта данных.....	8
2.3 Сценарии использования для задачи представления данных.....	9
2.4 Сценарии использования для задачи анализа данных.....	11
2.5 Сценарии использования для задачи экспорта данных.....	11
2.6 Вывод о преобладающих операциях.....	12
3. Модель данных.....	12
3.1 Нереляционная модель данных MongoDB.....	12
3.1.1 Описание назначений коллекций, типов данных и сущностей.....	13
3.1.2 Размеры документов.....	14
3.1.3 Формула для оценки объема данных.....	15
3.1.4 Примеры запросов.....	16
3.2 Реляционная модель данных.....	25
3.2.1 Описание назначений коллекций, типов данных и сущностей.....	25
3.2.2 Размеры полей (в байтах).....	26
3.2.3 Формула для оценки объема данных.....	28
3.2.4 Примеры запросов.....	28
3.3 Сравнение моделей.....	33
3.3.1 Удельный объём информации.....	33
3.3.2 Запросы по отдельным юзкейсам.....	34
3.3.3 Выводы.....	36
4. Разработанное приложение.....	36
5. Выводы.....	46
6. Приложения.....	48
7. Литература.....	49

1. Введение

Актуальность решаемой проблемы

Индустрія видеоигр развивается стремительными темпами, рассмотрим такой популярный агрегатор, как *Metacritic*. Согласно открытой статистике, посещаемость составляет около 11,7 миллионов визитов в месяц, большая часть пользователь использует десктопное приложение (см. Рис. 1).



Рис. 1 - Статистика посещаемости *Metacritic*

Но агрегаторы такие как *Metacritic*, *Rotten Tomatoes* и другие зачастую ориентированы на западную аудиторию, и их пользовательская база в странах СНГ, включая Россию, очень мала, что можно видеть по географии пользователей:

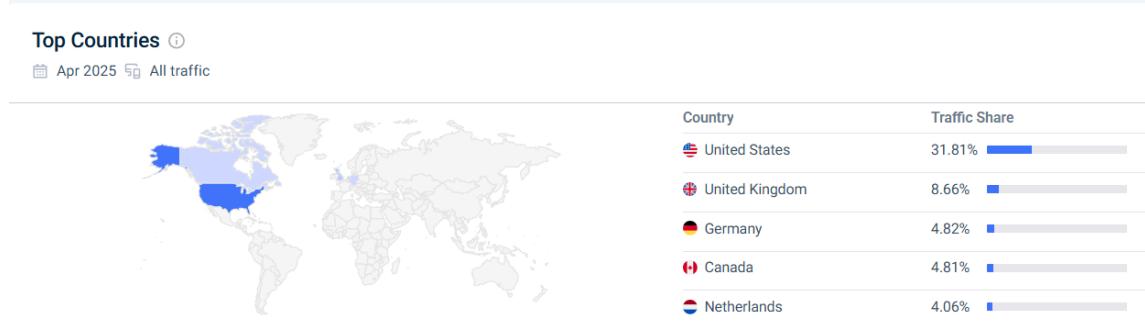


Рис. 3 - Статистика посещаемости *Metacritic* по странам

Это создаёт потребность в создании локализованной платформы, предоставляющей подобный функционал, но ориентированной в том числе и на русскоязычных пользователей.

Постановка задачи

Целью проекта является разработка веб-приложения, аналогичного *Metacritic*, с возможностью просмотра и оценки видеоигр. Приложение должно позволять:

- Пользователям авторизоваться;
- Оставлять отзывы на игры на разных платформах;
- Смотреть средние оценки по играм и отзывам;
- Искать и фильтровать игры по платформам;
- Получать информацию о рецензиях критиков;
- Просматривать статистику своих отзывов;
- Администратору управлять контентом (добавлять отзывы, модерировать данные).

Проект должен обеспечивать удобный пользовательский интерфейс, масштабируемость и возможность дальнейшего расширения функционала.

Предлагаемое решение

Для реализации проекта используется фреймворк *Django*, обеспечивающий гибкую архитектуру и удобную работу с шаблонами, а также база данных *MongoDB*, оптимизированная для хранения неструктурированных и полуструктурных данных, таких как отзывы и комментарии.

Качественные требования к решению

- Надёжность — корректная работа всех функций при разных сценариях использования.
- Юзабилити — простой и интуитивно понятный интерфейс, адаптированный под русскоязычную аудиторию.
- Масштабируемость — возможность добавления новых функций (например, рейтинги пользователей, система рекомендаций)

2. Сценарии использования

2.1 Макет UI

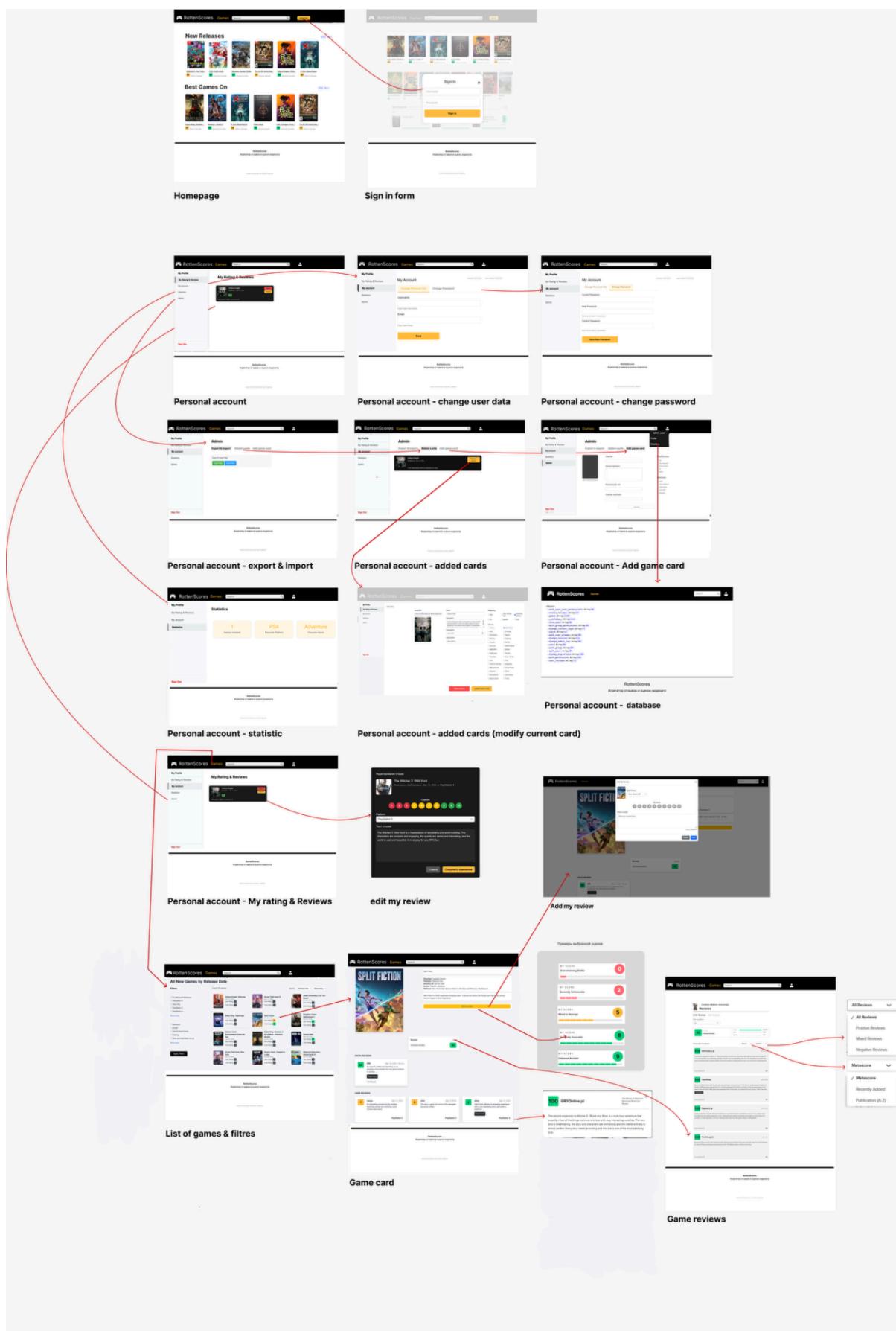


Рис. 4 - Макет приложения

2.2 Сценарии использования для задачи импорта данных

Добавление собственного отзыва

- Пользователь нажал на карточку игры
- Пользователь нажал на add my review
- Открылся экран add my review
- Пользователь нажал на post
- Отзыв пользователя появился у игры

[Admin] Массовый импорт

- Пользователь со статусом “админ” нажал на иконку пользователя
- Пользователь нажал на Admin
- Пользователь нажал на Export & import
- Открылось окно personal account (export & import)
- Пользователь нажал на Import Data, открылась файловая система компьютера
- Пользователь выбрал json файл, данные загрузились в веб приложение, произошел редирект на главную страницу

[Admin] Загрузка игры в систему

- Пользователь со статусом “админ” нажал на иконку пользователя
- Пользователь нажал на Admin
- Пользователь нажал на Add game card
- Открылся экран personal account (add game card)
- Пользователь заполнил все необходимые поля и нажал на Save game card
- Игра сохранилась в разделе Added cards

2.3 Сценарии использования для задачи представления данных

Первый вход на сайт

- Пользователь открыл веб-приложение
- Открылся экран homepage

Нажатие на название приложения

- Пользователь нажал на название приложения в левом верхнем углу
- Открылся экран homepage

Поиск игры

- Пользователь начинает вводить в поле поиска название игры
- Появляется список подходящих по введенному тексту игр
- Пользователь нажимает на желаемую игру
- Открылся экран game card

Переход в список игр

- Пользователь нажал на раздел Games
- Открылся экран list of games & filters

Нажатие на карточку игры

- Пользователь нажал на карточку игры
- Открылся экран game card

[Admin] Просмотр добавленных карточек игр

- Пользователь со статусом “админ” нажал на иконку пользователя
- Пользователь нажал на Admin
- Пользователь нажал на Added cards
- Открылся экран personal account (added cards)

Просмотр статистики

- Пользователь нажал на иконку пользователя
- Пользователь нажал на Profile в выпадающем окне
- Пользователь нажал на Statistics
- Открылся экран personal account (statistics)
- Показывается общее количество написанных отзывов, любимая платформа и любимый жанр

Просмотр всех отзывов на игру

- Пользователь нажал на карточку игры
- Пользователь пролистал вниз к разделу последних отзывов
- На экране отображается последние 3 отзыва критика и 3 пользовательских отзыва
- Пользователь нажал see all, по умолчанию открывается раздел critic reviews
- Открылся экран game reviews

Просмотр полного отзыва критика на игру

- Пользователь нажал на карточку игры
- Пользователь пролистал вниз к разделу последних отзывов
- Пользователь выбрал интересующий отзыв, нажал на Full review
- Пользователь был переадресован на сторонний сайт с полным отзывом

[Admin] Просмотр базы данных в виде интерактивного списка

- Пользователь нажал на иконку пользователя
- Пользователь нажал на Database в выпадающем окне
- Открылся экран View database
- При нажатии на коллекцию открывается список ее объектов

Просмотр всех игр

- Пользователь зашел на экран list of games & filters
- Есть возможность отфильтровать игры по жанру и платформе
- Есть возможность отсортировать игры по user score, critic score и дате выхода по убыванию и возрастанию
- Пользователь нажал на карточку игры
- Открылся экран game card

2.4 Сценарии использования для задачи анализа данных

Фильтрация всех игр

- Пользователь зашел на экран list of games & filters
- Есть возможность отфильтровать игры по жанру и платформе

- Пользователь выбрал интересующие платформы и нажал Add filter
- Отобразились игры, выпущенные на данной платформе

Фильтрация всех отзывов

- Пользователь нажал на карточку игры
- Пользователь пролистал вниз к разделу последних отзывов
- На экране отображается последние 3 отзыва критика и 3 пользовательских отзыва
- Пользователь нажал see all, по умолчанию открывается раздел critic reviews
- Пользователь выбрал сортировку от новых отзывов к старым
- Отзывы отсортированы от новых к старым

2.5 Сценарии использования для задачи экспорта данных

[Admin] Массовый экспорт

- Пользователь со статусом “админ” нажал на иконку пользователя
- Пользователь нажал на Admin
- Пользователь нажал на Export & import
- Открылось окно personal account (export & import)
- Пользователь нажал на Export Data
- Автоматически сохранился файл в формате .json в файловую систему пользователя

2.6 Вывод о преобладающих операциях

По сценариям видно, что преобладает именно чтение, так как почти все страницы нацелены именно на чтение отзывов и игр. Для записи доступно только создание и изменение своего отзыва, а в случае статуса админа еще и создание и изменение карточки игры.

3. Модель данных

3.1 Нереляционная модель данных MongoDB

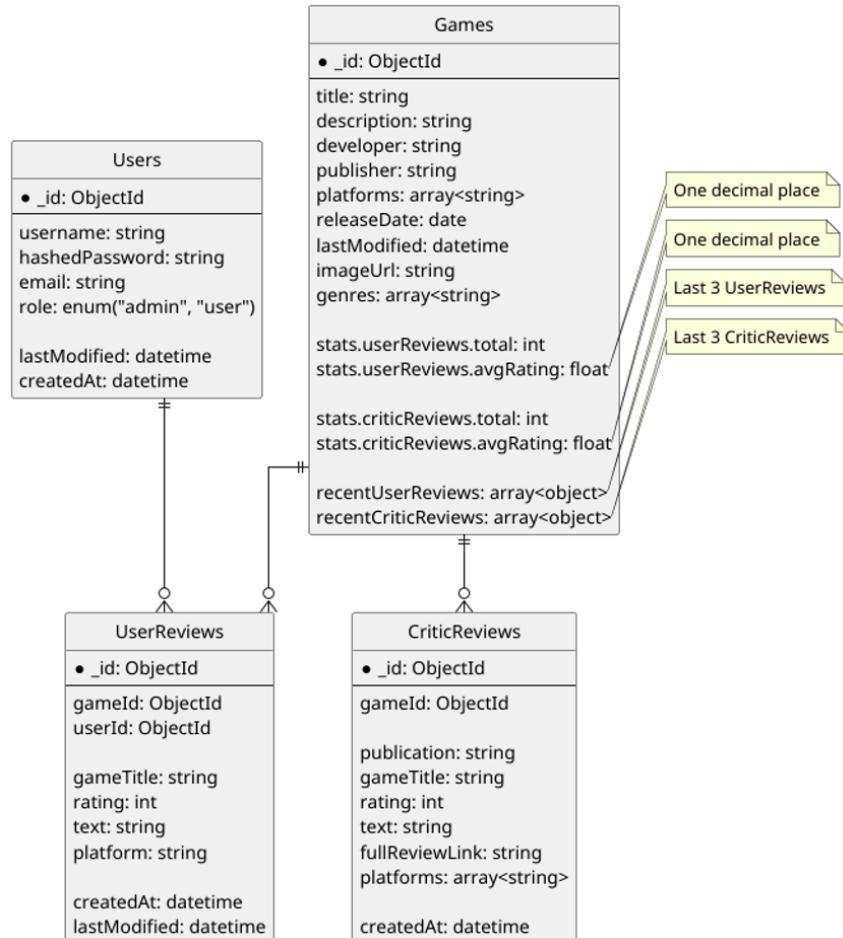


Рис. 5 - Графическое представление нереляционной модели данных

3.1.1 Описание назначений коллекций, типов данных и сущностей

1) games

Информация об играх.

Основные поля: title, description, developer, publisher, platforms, releaseDate, imageUrl, genres

Статистика: stats.userReviews, stats.criticReviews (количество и средний рейтинг для быстрого отображения без агрегирования и пересчета)
Оптимизация: recentUserReviews, recentCriticReviews(последние 3 отзыва для быстрого отображения на карточке игры)

Метаданные: lastModified

2) users

Данные о пользователях.

Основные поля: username, hashedPassword, email, role

Метаданные: lastModified, createdAt

3) userReviews

Пользовательские обзоры на игры.

Связи: gameId, userId

Основные поля: rating, text, platform, createdAt

Метаданные: lastModified

4) criticReviews

Рецензии критиков.

Связи: gameId

Основные поля: publication, rating, text, createdAt, fullReviewLink, platforms

3.1.2 Размеры документов

Каждый объект хранит id (UUID) и (если есть) title, description (строка). Примем средние размеры: UUID: 16 байт, title: 150 байт (но для заголовка игры было взято значение самого длинного существующего названия игры), description: 1000 байт (если есть). Оценим размеры объектов в базе данных MongoDB:

Игра (games):

- id: 16 байт

- title: 336 байт
- description: 1000 байт
- developer: 100 байт
- publisher: 100 байт
- platforms: 50 байт
- releaseDate: 8 байт
- lastModified: 8 байт
- genres: 50 байт
- imageUrl: 200 байт
- stats.userReviews.total: 4 байт
- stats.userReviews.avgRating: 4 байт
- stats.criticReviews.total: 4 байт
- stats.criticReviews.avgRating: 4 байт
- recentUserReviews: $3 * 738 = 2214$ байт
- recentCriticReviews: $3 * 2230 = 6690$ байт
- **Итого на игру:** ~10688 байт

Пользователь (users):

- id: 16 байт
- username: 50 байт
- hashedPassword: 60 байт
- email: 50 байт
- role: 4 байт (enum)
- lastModified: 8 байт
- createdAt: 8 байт
- **Итого на пользователя:** ~196 байт

Пользовательский отзыв (userReviews):

- id: 16 байт
- gameId: 16 байт
- userId: 16 байт
- gameTitle: 336 байт
- rating: 4 байт
- text: 500 байт (ограничение)

- platform: 20 байт
- createdAt: 8 байт
- lastModified: 8 байт
- **Итого на отзыв:** ~738 байт

Рецензия критика (criticReviews):

- id: 16 байт
- gameId: 16 байт
- publication: 100 байт
- gameTitle: 336 байт
- rating: 4 байт
- text: 1500 байт (ограничение)
- fullReviewLink: 200 байт
- platforms: 50 байт
- createdAt: 8 байт
- **Итого на рецензию критика:** ~2230 байт

3.1.3 Формула для оценки объема данных

Пусть:

G — количество игр

U — количество пользователей

среднее количество пользовательских отзывов на игру R_u

среднее количество отзывов критиков на игру R_c

Количество пользователей U обычно превышает количество игр (предположим $U = k \times G$, где k - коэффициент)

Тогда общий объём данных V в байтах:

$$V(G) = G \times \text{Размер}_{\text{Games}} + U \times \text{Размер}_{\text{Users}} + (G \times R_u) \times \text{Размер}_{\text{UserReviews}} + (G \times R_c) \times \text{Размер}_{\text{CriticReviews}}$$

Подставляем значения:

$$V(G) = G \times 10688 + (k \times G) \times 196 + (G \times R_u) \times 738 + (G \times R_c) \times 2230$$

Пусть: ($k = 5$), ($R_u = 20$), ($R_c = 5$)

$$V(G) = G \times 10688 + G \times 980 + G \times 14760 + G \times 11150$$

$$V(G) = G \times 37578 \text{ байт}$$

Формула показывает, что объем данных линейно зависит от количества игр (G) в базе данных. Наибольшее влияние на объем данных оказывают пользовательские отзывы, несмотря на то, что каждый отзыв критика занимает больше места (2230 байт против 738 байт). Это объясняется их большим количеством (в среднем 20 на игру против 5 отзывов критиков). Размер самих документов игр также существенен (28.4%), особенно из-за хранения недавних отзывов непосредственно в документе игры.

3.1.4 Примеры запросов

Юзкейс №1/2: Главная страница (новые и лучшие игры)

Параметры:

Коллекции: 1 - games

Количество запросов: 2

Сложность: - Без индекса: $O(n)$ full collection scan + $O(n \log n)$ сортировка - С индексом: $O(\log n)$ поиск по индексу + $O(k)$ лимит результатов ($k=8$)

```
// Новые релизы
db.games.aggregate([
  { $sort: { releaseDate: -1 } },
  { $limit: 8 },
  { $project: { title: 1, image_ref: 1, stats: 1 } }
])

// Лучшие игры для PS5
db.games.aggregate([
  { $match: { platforms: "PS5" } },
  { $sort: { stats: -1 } },
  { $limit: 8 }
])
```

```
{ $sort: { "stats.criticReviews.avgRating": -1 } },
{ $limit: 8 },
{ $project: { title: 1, image_ref: 1, stats: 1 } }
])
```

Юзкейс №3: Поиск игры

Параметры:

Коллекции: 1 - games

Количество запросов: 1

Сложность: - Без индекса: $O(n)$ full collection scan - С индексом: $O(\log n)$ поиск по B-дереву

```
db.games.findOne(
  { title: "Cyberpunk 2077" },
  { title: 1, description: 1, developer: 1, releaseDate: 1 }
)
```

Юзкейс №4: Список игр

Параметры:

Коллекции: 1 - games

Количество запросов: 1

Сложность: - Всегда $O(n)$ - требует полного сканирования коллекции

```
db.games.find(
  {},
  { title: 1, releaseDate: 1, image_ref: 1, description: 1 }
)
```

Юзкейс №5: Карточка игры

Параметры:

Коллекции: 1 - games

Количество запросов: 1

Сложность: - С индексом по `_id`: $O(1)$ - прямое обращение по ObjectId - Без индекса: $O(n)$

```
db.games.findOne (
```

```

{ _id: ObjectId("507f1f77bcf86cd799439011") } ,
{
  title: 1,
  releaseDate: 1,
  image_ref: 1,
  description: 1,
  platforms: 1,
  developer: 1,
  publisher: 1,
  stats: 1,
  genres: 1
}
)

```

Юзейс №6: Фильтрация отзывов

Параметры:

Коллекции: 1 - `usersReview`

Количество запросов: 1

Сложность:

- Без индекса: $O(n)$ полное сканирование
- С составным индексом { gameId: 1, rating: -1 }: $O(\log n)$ поиск + $O(k)$ фильтрация

```

db.userReviews.aggregate([
  { $match: {
    gameId: ObjectId("507f1f77bcf86cd799439011"),
    rating: { $gte: 7 }
  }},
  { $sort: { rating: -1, createdAt: -1 } },
  { $project: { text: 1, rating: 1, createdAt: 1 } }
])

```

Юзейс №9: Смена username

Параметры:

Коллекции: 1 - users

Количество запросов: 1

Сложность: - Без индекса: $O(n)$ - С индексом: $O(\log n)$ поиск + $O(1)$ обновление

```
db.users.updateOne(  
  { username: "gamer1" },  
  { $set: { username: "new_username", lastModified: new Date() } }  
)
```

Юзейс №10: Смена пароля

Параметры:

Коллекции: 1 - users

Количество запросов: 1

Сложность: - Без индекса: $O(n)$ полное сканирование коллекции.

С индексом по username: $O(\log n)$ поиск + $O(1)$ обновление.

```
db.users.updateOne(  
  { username: "gamer1" },  
  { $set: { hashedPassword: "new_hash", lastModified: new Date() } }  
)
```

Юзейс №11: Добавление отзыва

Параметры:

Коллекции: 2 - usersReviews, games

Количество запросов: 2

Сложность: - Вставка отзыва: $O(1)$ - Обновление статистики: - С индексом: $O(\log n)$ поиск игры + $O(1)$ обновление - Без индекса: $O(n)$

```
const review = {  
  gameId: ObjectId("507f1f77bcf86cd799439011"),  
  userId: ObjectId("507f191e810c19729de860eb"),  
  rating: 9,  
  text: "Отличная игра!",  
  platform: "PC",  
  createdAt: new Date(),
```

```

    lastModified: new Date()
}

db.userReviews.insertOne(review)

db.games.updateOne(
  { _id: review gameId },
  {
    $inc: { "stats.userReviews.total": 1 },
    $set: {
      "stats.userReviews.avgRating": {
        $divide: [
          { $add: [
              { $multiply: ["$stats.userReviews.avgRating",
                "$stats.userReviews.total"] },
              review.rating
            ] },
          { $add: ["$stats.userReviews.total", 1] }
        ]
      },
      lastModified: new Date()
    },
    $push: {
      recentUserReviews: {
        $each: [
          rating: review.rating,
          text: review.text,
          userId: review.userId
        ],
        $sort: { createdAt: -1 },
        $slice: 3
      }
    }
  }
)

```

Юзкейс №15: Просмотр игр (админ)

Параметры:

Коллекции: 1 - games

Количество запросов: 1

Сложность: - $O(n \log n)$

```
db.games.find({},
  { title: 1, releaseDate: 1, lastModified: 1 }
).sort({ lastModified: -1 })
```

Юзкейс №16: Изменение игры (админ)

Параметры:

Коллекции: 1 - games

Количество запросов: 1

Сложность: - По title без индекса: $O(n)$

С индексом по title: $O(\log n)$ поиск + $O(1)$ обновление

По _id: $O(1)$ (встроенный индекс)

```
db.games.updateOne(
  { title: "Cyberpunk 2077" },
  {
    $set: {
      description: "Updated description",
      platforms: ["PC", "PS5", "Xbox Series X"],
      lastModified: new Date()
    }
  }
)
```

Юзкейс №17: Удаление игры (админ)

Параметры:

Коллекции: 3 - userReviews, criticReviews, games

Количество запросов: 3

Сложность: - Удаление игры: $O(1)$ с индексом по _id - Удаление отзывов: - Без индекса: $O(n)$ полное сканирование - С индексом: $O(\log n)$ поиск + $O(m)$ удаление (m - кол-во отзывов)

```
const gameId = ObjectId("507f1f77bcf86cd799439011")
```

```
db.userReviews.deleteMany({ gameId: gameId })
db.criticReviews.deleteMany({ gameId: gameId })
```

```
db.games.deleteOne({ _id: gameId })
```

Юзкейс №18: Добавление игры (админ)

Параметры:

Коллекции: 1 - games

Количество запросов: 1

Сложность: - O(1) - вставка в конец коллекции

```
db.games.insertOne({
  title: "Новая игра",
  description: "Описание...",
  platforms: ["PC", "PS5"],
  releaseDate: new Date("2025-12-10"),
  stats: {
    userReviews: { total: 0, avgRating: 0 },
    criticReviews: { total: 0, avgRating: 0 }
  },
  createdAt: new Date(),
  lastModified: new Date()
})
```

Юзкейс №20: Все отзывы на игру

Параметры:

Коллекции: 2 - userReviews, users

Количество запросов: 1 запрос с \$lookup

Сложность: - Без индексов: O(n + m) полное сканирование обеих коллекций - С индексами: - O(log n) поиск отзывов по gameId - O(k log m) lookup пользователей (k - кол-во отзывов)

```
db.userReviews.aggregate([
  { $match: { gameId: ObjectId("game_id") } },
  { $sort: { createdAt: -1 } },
  { $limit: 50 },
  {
    $lookup: {
      from: "users",
      localField: "userId",
      foreignField: "id"
    }
  }
])
```

```

        foreignField: "_id",
        as: "user"
    }
},
{ $unwind: "$user" },
{ $project: { text: 1, rating: 1, createdAt: 1, "user.username": 1 } }
])

```

Юзкейс №22: Редактирование отзыва

Параметры:

Коллекции: 1 - usersReview

Количество запросов: 1

Сложность: - По _id: O(1) (встроенный индекс)

По составному условию (_id + userId):

Без индекса: O(n)

С составным индексом { _id: 1, userId: 1 }: O(log n)

```

db.userReviews.updateOne (
  {
    _id: ObjectId("review_id"),
    userId: ObjectId("user_id")
  },
  {
    $set: {
      rating: 8,
      text: "Обновленный текст",
      lastModified: new Date()
    }
  }
)

```

Юзкейс №23: Удаление отзыва

Параметры:

Коллекции: 1 - usersReview

Количество запросов: 1

Сложность: - По `_id`: O(1) (встроенный индекс) - По составному условию (`_id + userId`): Без индекса: O(n), С составным индексом { `_id: 1, userId: 1` }: O(log n)

```
db.userReviews.deleteOne ({
  _id: ObjectId("review_id"),
  userId: ObjectId("user_id")
})
```

3.2 Реляционная модель данных

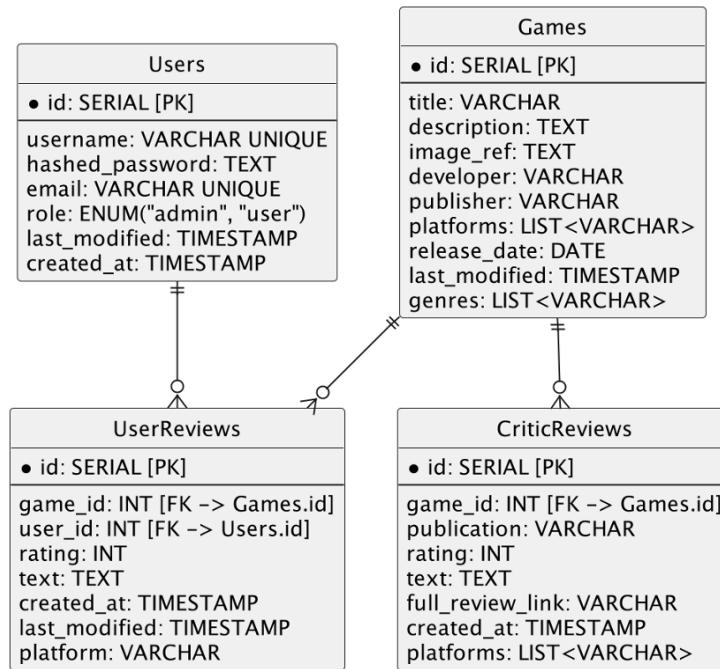


Рис. 6 - Графическое представление реляционной модели данных

3.2.1 Описание назначений коллекций, типов данных и сущностей

Games

Информация об играх.

Основные поля: `title`, `description`, `image_ref`, `developer`, `publisher`, `platforms`, `release_date`, `genres`

Метаданные: `last_modified`

Users

Данные о пользователях.

Основные поля: username, hashed_password, email, role

Метаданные: last_modified, created_at

UserReviews

Пользовательские обзоры на игры.

Внешние ключи: game_id, user_id

Основные поля: rating, text, platform, created_at

Метаданные: last_modified

CriticReviews

Рецензии критиков.

Внешние ключи: game_id

Основные поля: publication, rating, text, created_at, full_review_link, platforms

3.2.2 Размеры полей (в байтах)

Таблица Games

- id: 4 байта (*INT*)
- title: 336 байт (*336 символов* 1 байт/символ в UTF-8)*
- description: 1000 байт (*1000 символов*)
- image_ref: 100 байт (*URL*)
- developer: 50 байт
- publisher: 50 байт
- platforms: 100 байт (*5 элементов* 20 байт)*
- release_date: 4 байта (*DATE*)
- last_modified: 8 байт (*TIMESTAMP*)
- genres: 60 байт (*3 элемента* 20 байт)*

Итого на одну игру: Размер_{Games} = 4 + 336 + 1000 + 100 + 50 + 50 + 100 + 4 + 8 + 60 = 1712 байт

Таблица Users

- id: 4 байта
- username: 15 байт
- hashed_password: 64 байта (*hex*)
- email: 50 байт
- role: 1 байт (*ENUM*)
- last_modified: 8 байт
- created_at: 8 байт

Итого на одного пользователя: Размер_{Users} = 4 + 15 + 64 + 50 + 1 + 8 + 8 = 150 байт

Таблица UserReviews

- id: 4 байта
- game_id: 4 байта
- user_id: 4 байта
- rating: 4 байта (*INT*)
- text: 500 байт
- created_at: 8 байт
- last_modified: 8 байт
- platform: 20 байт

Итого на один отзыв: Размер_{UserReviews} = 4 + 4 + 4 + 4 + 500 + 8 + 8 + 20 = 552 байт

Таблица CriticReviews

- id: 4 байта
- game_id: 4 байта
- publication: 50 байт
- rating: 4 байта
- text: 1500 байт
- full_review_link: 100 байт (*URL*)
- created_at: 8 байт

- platforms: 60 байт (3 элемента 20 байт)*

Итого на один отзыв: $\text{Размер}_{\text{CriticReviews}} = 4 + 4 + 50 + 4 + 1500 + 100 + 8 + 60 = 1730$ байт

3.2.3 Формула для оценки объема данных

Пусть:

G — количество игр

U — количество пользователей

R_u — среднее количество пользовательских отзывов на игру

R_c — среднее количество критических отзывов на игру

Тогда общий объём данных (V) в байтах:

$$V(G, U, R_u, R_c) = G \times \text{Размер}_{\text{Games}} + U \times \text{Размер}_{\text{Users}} + (G \times R_u) \times \text{Размер}_{\text{UserReviews}} + (G \times R_c) \times \text{Размер}_{\text{CriticReviews}}$$

Подставляя рассчитанные значения:

$$V(G, U, R_u, R_c) = G \times 1712 + U \times 150 + (G \times R_u) \times 552 + (G \times R_c) \times 1730$$

Упрощенная форма:

$$V(G, U, R_u, R_c) = G (1712 + 552R_u + 1730R_c) + 150U$$

3.2.4 Примеры запросов

Юзкейс №1 Первый вход на сайт:

```
-- New releases
SELECT Games.title, Games.image_ref, AVG(CR.rating)
FROM Games
INNER JOIN CriticReviews CR on Games.id = CR.game_id
GROUP BY Games.title, Games.image_ref, CR.rating,
```

```

Games.release_date
ORDER BY Games.release_date DESC
LIMIT 8;

-- Best games on
SELECT Games.title, Games.image_ref, AVG(CR.rating)
FROM Games
INNER JOIN CriticReviews CR on Games.id = CR.game_id
GROUP      BY      Games.title,      Games.image_ref,      CR.rating,
Games.release_date, Games.platforms
HAVING AVG(CR.rating) >= 90 AND Games.platforms LIKE '%PS5%'
ORDER BY AVG(CR.rating) DESC
LIMIT 8;

```

Юзкейс №2 Нажатие на иконку приложения: Запрос, аналогичный юзкейсу №1.

Юзкейс №3 Поиск игры:

```

SELECT      Games.title,      Games.description,      Games.developer,
Games.release_date
FROM Games
WHERE Games.title = 'Cyberpunk 2077';

```

Юзкейс №4 Переход в список игр:

```

SELECT      Games.title,      Games.release_date,      Games.image_ref,
Games.description
FROM Games;

```

Юзкейс №5 Нажатие на карточку игры:

```

-- Game card info
SELECT Games.title,
       Games.release_date,
       Games.image_ref,
       Games.description,
       Games.platforms,
       Games.developer,
       Games.publisher,
       AVG(CR.rating),
       Games.genres

FROM Games
INNER JOIN CriticReviews CR on Games.id = CR.game_id
WHERE Games.id = 1;

```

```
-- Critic Reviews
SELECT CriticReviews.rating,
       CriticReviews.created_at,
       SUBSTR(CriticReviews.text, 1, 100),
       CriticReviews.publication,
       CriticReviews.full_review_link,
       CriticReviews.platforms
FROM CriticReviews
INNER JOIN Games G on G.id = CriticReviews.game_id
WHERE G.id = 1
ORDER BY CriticReviews.created_at DESC
LIMIT 3;
```

```
-- User Reviews
SELECT UserReviews.rating,
       UserReviews.created_at,
       SUBSTR(UserReviews.text, 1, 100),
       UserReviews.platform
FROM UserReviews
INNER JOIN Games G on G.id = UserReviews.game_id
WHERE G.id = 1
ORDER BY UserReviews.created_at DESC
LIMIT 3;
```

Юзкейс №6 Фильтрация всех пользовательских отзывов на игру по положительной оценке(>=7):

```
SELECT           UserReviews.text,           UserReviews.rating,
UserReviews.created_at
FROM UserReviews
WHERE UserReviews.game_id = 1 AND UserReviews.rating >=7
ORDER BY UserReviews.rating DESC;
```

Юзкейс №7 Вход в аккаунт и Юзкейс №8 Выход из аккаунта не взаимодействуют с базой данных.

Юзкейс №9 Смена личных данных(например, смена username):

```
UPDATE Users
SET username = 'new_username'
WHERE username = 'gamer1';
```

Юзкейс №10 Смена пароля:

```
UPDATE Users
SET hashed_password = 'new_hashed_password', last_modified =
CURRENT_TIMESTAMP
WHERE username = 'gamer1';
```

Юзейс №11 Добавление собственного отзыва:

```
INSERT INTO UserReviews (game_id, user_id, rating, text,
created_at, last_modified, platform)
VALUES (1, 2, 9, 'Отличная игра! Очень понравился сюжет.', CURRENT_TIMESTAMP, CURRENT_TIMESTAMP, 'PC');
```

Юзейс №12 [Admin] Просмотр личного аккаунта нет взаимодействия с базой данных.

Юзейс №13, 14 [Admin] Массовый импорт и экспорт - данный сценарий только для нереляционной СУБД.

Юзейс №15 [Admin] Просмотр добавленных карточек игр:

```
SELECT Games.title,
       Games.release_date,
       Games.last_modified,
       SUBSTR(Games.description, 1, 30),
       Games.image_ref
FROM Games;
```

Юзейс №16 [Admin] Изменение карточки игры:

```
UPDATE Games
SET
    description = 'Updated description for the new edition of the game.',
    platforms = 'PC, PS5, Xbox Series X',
    last_modified = CURRENT_TIMESTAMP
WHERE title = 'Cyberpunk 2077';
```

Юзейс №17 [Admin] Удаление карточки игры:

```
DELETE FROM Games
WHERE id = 1;
```

Подразумевается использование ON DELETE CASCADE

Юзейс №18 [Admin] Загрузка игры в систему:

```
INSERT INTO Games (title, description, image_ref, developer,
publisher, platforms, release_date, last_modified, genres)
```

```

VALUES (
    'Elden Ring',
    'Open-world action RPG from FromSoftware.',

'https://assets.xboxservices.com/assets/7b/54/7b54f5e4-0857-4ce3-8
a18-2b8c431e8a9e.jpg?n=Elden-Ring_GLP-Page-Hero-0_1083x1222_01.jpg
',
    'FromSoftware',
    'Bandai Namco',
    '["PC", "PS5", "Xbox"]',
    '2022-02-25',
    CURRENT_TIMESTAMP,
    '["RPG", "Action", "Adventure"]'
);

```

Юзкейс №19 [Admin] Просмотр статистики:

```

WITH UserReviewStats AS (
    SELECT
        user_id,
        COUNT(*) AS total_reviews,
        AVG(rating) AS avg_rating
    FROM UserReviews
    WHERE user_id = 4
    GROUP BY user_id
),
MostFrequentPlatform AS (
    SELECT platform, COUNT(*) as count
    FROM UserReviews
    WHERE user_id = 4
    GROUP BY platform
    ORDER BY count DESC
    LIMIT 1
),
MostFrequentGenre AS (
    SELECT g.genre, COUNT(*) as count
    FROM UserReviews ur
    JOIN Games g ON ur.game_id = g.id,
    json_each(g.genres) as g
    WHERE ur.user_id = 4
    GROUP BY g.genre
    ORDER BY count DESC
    LIMIT 1
)

```

```
SELECT
    u.total_reviews,
    ROUND(u.avg_rating, 1) AS avg_rating,
    p.platform AS most_frequent_platform,
    g.genre AS most_frequent_genre
FROM UserReviewStats u
LEFT JOIN MostFrequentPlatform p ON 1=1
LEFT JOIN MostFrequentGenre g ON 1=1;
```

Юзейс №20 Просмотр всех отзывов на игру:

```
(SELECT 'user' as review_type, id, user_id as author_id, rating,
text, created_at
FROM UserReviews
WHERE game_id = 7
ORDER BY created_at DESC
LIMIT 3)
UNION ALL
(SELECT 'critic' as review_type, id, publication as author_id,
rating, text, created_at
FROM CriticReviews
WHERE game_id = 7
ORDER BY created_at DESC
LIMIT 3);
```

Юзейс №21 Просмотр полного отзыва критика: не подразумевает взаимодействие с БД.

Юзейс №22 Редактирование собственных отзывов:

```
UPDATE UserReviews
SET rating = 9, text = 'Улучшенная боевка, отличная графика!',
last_modified = CURRENT_TIMESTAMP
WHERE id = 9 AND user_id = 4;
```

Юзейс №23 Удаление собственных отзывов:

```
DELETE FROM UserReviews
WHERE id = 15 AND user_id = 8;
```

Юзейс №24 Просмотр базы данных в виде интерактивного списка-данный сценарий только для нереляционной СУБД.

Юзейс №25 Просмотр всех игр:

```
SELECT * FROM Games;
```

3. 3 Сравнение моделей

3.3.1 Удельный объём информации

Таблица 1 - Удельный объем информации

Параметр	NoSQL	SQL
Формула объема	$V(G) = G \times 37578$ байт	$V(G) = G \times 22152$ байт
Объем при $G=10$	375.78 КБ	221.52КБ
Зависимость от G	Быстрый рост (множитель 37578)	Медленный рост (множитель 22152)
Избыточность	1.82 (82% метаданных)	1.09 (9% метаданных)

3.3.2 Запросы по отдельным юзкейсам

Таблица 2 - Запросы по отдельным юзкейсам

Юзке йс	Mongo DB (кол-во запрос ов)	Сложность MongoDB	SQL (кол-во запрос ов)	Сложность SQL	Преимущест во MongoDB	Преимуществ о SQL
1/2	2	$O(\log n) + O(k)$ с индексом $O(n) + O(n \log n)$ без	2	$O(n \log n) + O(m)$	Все данные в одном документе	Более точная агрегация
3	1	$O(\log n)$ с индексом $O(n)$ без	1	$O(\log n)$ с индексом $O(n)$ без	Полнотекстовый поиск	Простые WHERE-условия
4	1	$O(n)$	1	$O(n)$	Простой find	SELECT простой
5	1	$O(1)$ с индексом $O(n)$ без	3	$O(n)$ для каждого	Нет JOIN'ов	Нормализованные данные

6	1	$O(\log n)$ с индексом $O(n)$ без	1	$O(\log n)$ с индексом $O(n)$ без	Гибкая фильтрация	WHERE+ORDER BY
9	1	$O(\log n)$ с индексом $O(n)$ без	1	$O(\log n)$ с индексом $O(n)$ без	Быстрый update	UPDATE с WHERE
10	1	$O(\log n)$ с индексом $O(n)$ без	1	$O(\log n)$ с индексом $O(n)$ без	Обновление по фильтру	WHERE по username
11	2	$O(1)$ вставка $O(\log n)$ обновление с индексом $O(n)$ без	3	$O(1)$ вставка $O(n)$ обновление	Атомарное обновление	ACID-гарантии
14	1	$O(n \log n)$	1	$O(n \log n)$	Быстрая сортировка	ORDER BY
15	1	$O(1)$ по <code>_id</code> $O(\log n)$ по title с индексом $O(n)$ без	1	$O(1)$ с индексом $O(n)$ без	Частичное обновление	Четкий контроль
16	3	$O(1)$ игра $O(\log n) + O(m)$ отзывы с индексом $O(n)$ без	3+	$O(n)$ все операции	Быстрое удаление	Каскадное удаление
17	1	$O(1)$	1	$O(1)$	Простая вставка	Простая вставка
18	1	$O(n)$ без индекса $O(\log n) + O(m)$ с индексом	1	$O(n)$	Гибкая агрегация	CTE и JOIN
20	1	$O(\log n) + O(k \log m)$ с индексами $O(n + m)$ без	1-2	$O(n \log n)$ подзапросы	\$lookup вместо JOIN	UNION ALL проще

20	1	$O(\log n)$ с индексом $O(n)$ без	1	$O(\log n)$ с индексом $O(n)$ без	Гибкая фильтрация	WHERE+ORDER BY
22	1	$O(1)$ по <code>_id</code> $O(\log n)$ составной индекс	1	$O(1)$	Быстрый доступ	Простая фильтрация
23	1	$O(1)$ по <code>_id</code> $O(\log n)$ составной индекс	1	$O(1)$	Удаление по фильтру	Удаление по WHERE

3.3.3 Выводы

Из проведенного сравнения видно, что использование MongoDB потребует больше памяти, но предоставит возможности для эффективной обработки данных, в том числе благодаря денормализации и хранении агрегированных полей статистики, что делалось бы в SQL при помощи составных запросов и соединений таблиц. Таким образом, решение с MongoDB является более предпочтительным для нашего проекта.

4. Разработанное приложение

Краткое описание

Backend представляет из себя *python*-приложение на фреймворке *Django*. *Frontend* написан на *HTML*, *JavaScript* и *CSS*. Коллекции данных хранятся в нереляционной базе данных *MongoDB*. Приложение собирается через *Docker*-файл.

Приложение имеет монолитную архитектуру. Проект состоит из приложений *games*, *reviews*, *user_profile*, *core*, *admin_panel*.

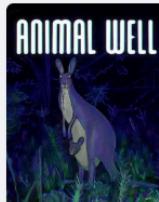
Использованные технологии

Django, *MongoDB*, *Docker*; для версионирования использован *Git*.

Снимки всех экранов приложения

 RottenScores Games Search Authorization

New Releases SEE ALL

 Split Fiction Universal... 90	 Kingdom Co... Not Rated	 Atomic Heart:... Not Rated	 Elden Ring: S... Not Rated	 Animal Well Not Rated	 Grand Theft ... Not Rated
---	---	--	--	--	---

Best Games on SEE ALL

 The Witcher ... Universal... 97	 Split Fiction Universal... 90	 Hollow Knight... Not Rated	 Elden Ring: ... Not Rated	 It Takes Two Not Rated	 Elden Ring: S... Not Rated
--	--	---	--	--	---

RottenScores
Агрегатор отзывов и оценок видеоигр

© 2025 ROTTENSCORES. ALL RIGHTS RESERVED.

Рис. 7 - Главная страница

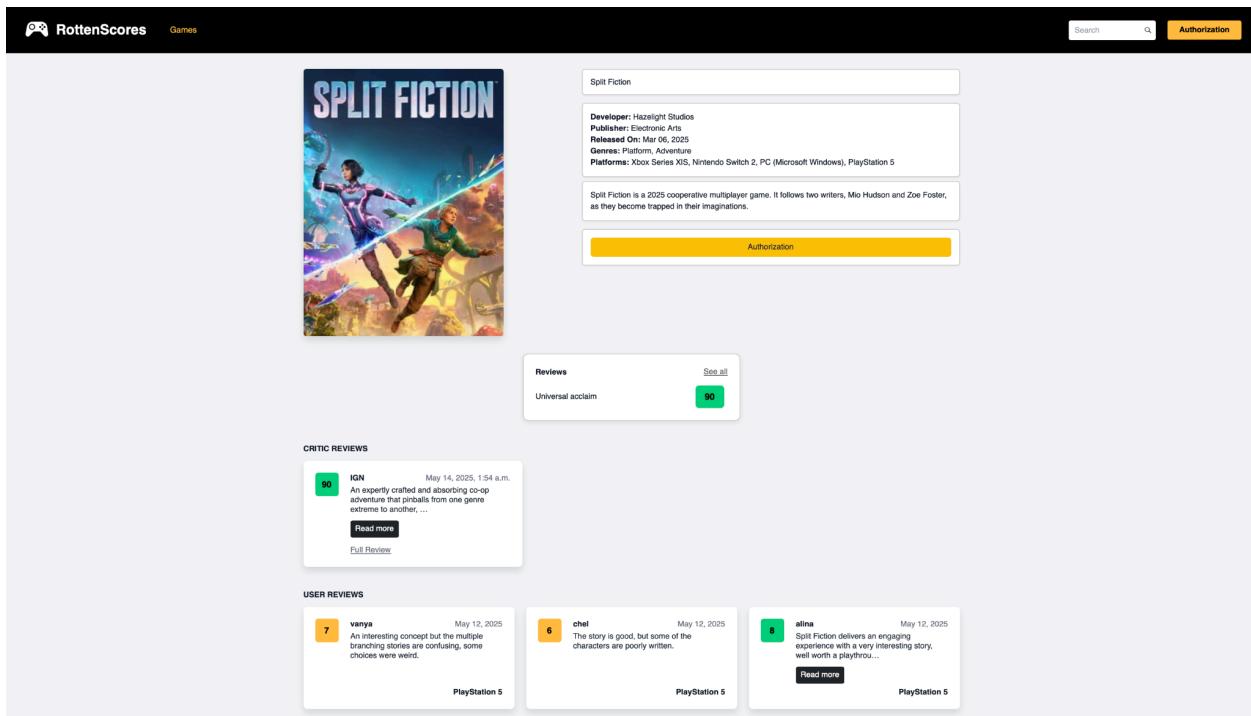


Рис. 8 - Карточка игры

Рис. 9 - Список отзывов на игру

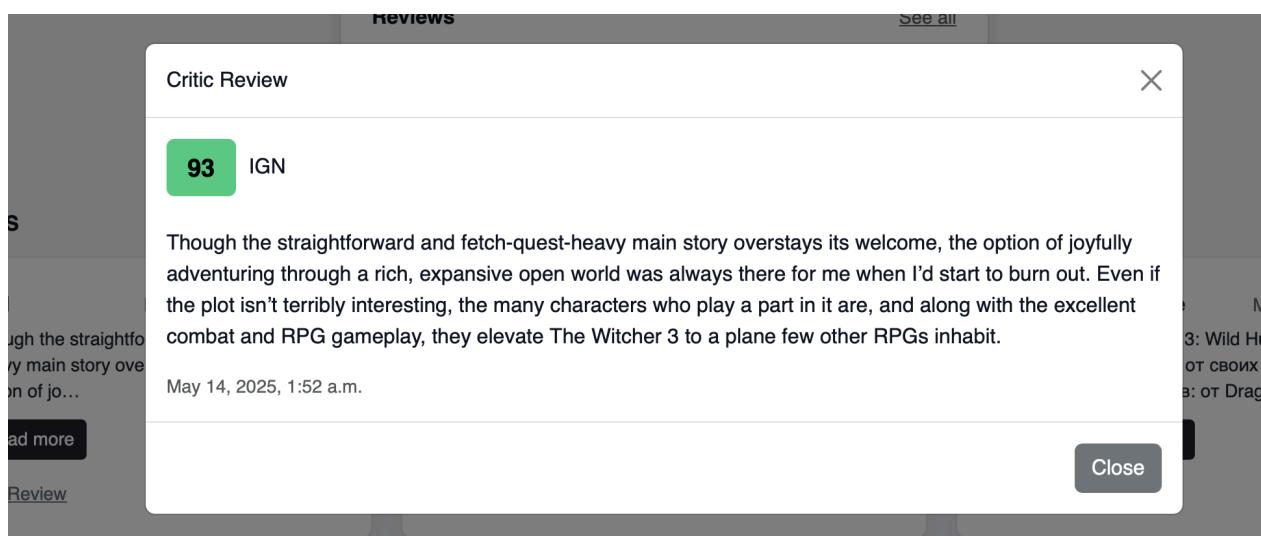


Рис. 10 - Плашка полного отзыва

All New Games by Release Date

Filters

- PC (Microsoft Windows)
- PlayStation 4
- Xbox One
- PlayStation 3
- PlayStation 5

Show more

- Adventure
- Arcade
- Card & Board Game
- Fighting
- Hack and slash/Beat 'em up

Show more

Found 203 games

Sort by: Release Date Descending

	Hollow Knight: Silksong Dec 31, 2025 User Rating N/A Critic Rating N/A		Grand Theft Auto VI Dec 31, 2025 User Rating N/A Critic Rating N/A		Death Stranding 2: On The Beach Jun 26, 2025 User Rating N/A Critic Rating N/A
	Elden Ring: Nightrregn May 30, 2025 User Rating N/A Critic Rating N/A		Split Fiction Mar 06, 2025 User Rating 7.8 Critic Rating 90		Kingdom Come: Deliverance II Feb 04, 2025 User Rating 8.2 Critic Rating N/A
	Atomic Heart: Enchantment Under the Sea Jan 28, 2025 User Rating N/A Critic Rating N/A		Elden Ring: Shadow of the Erdtree - Premium Bundle Jun 21, 2024 User Rating N/A Critic Rating N/A		Animal Well May 09, 2024 User Rating 8.7 Critic Rating N/A
	Grand Theft Auto: New York City		Atomic Heart: Trapped		Minecraft Education: Planets Earth

Apply Filters

Рис. 11 - Список игр

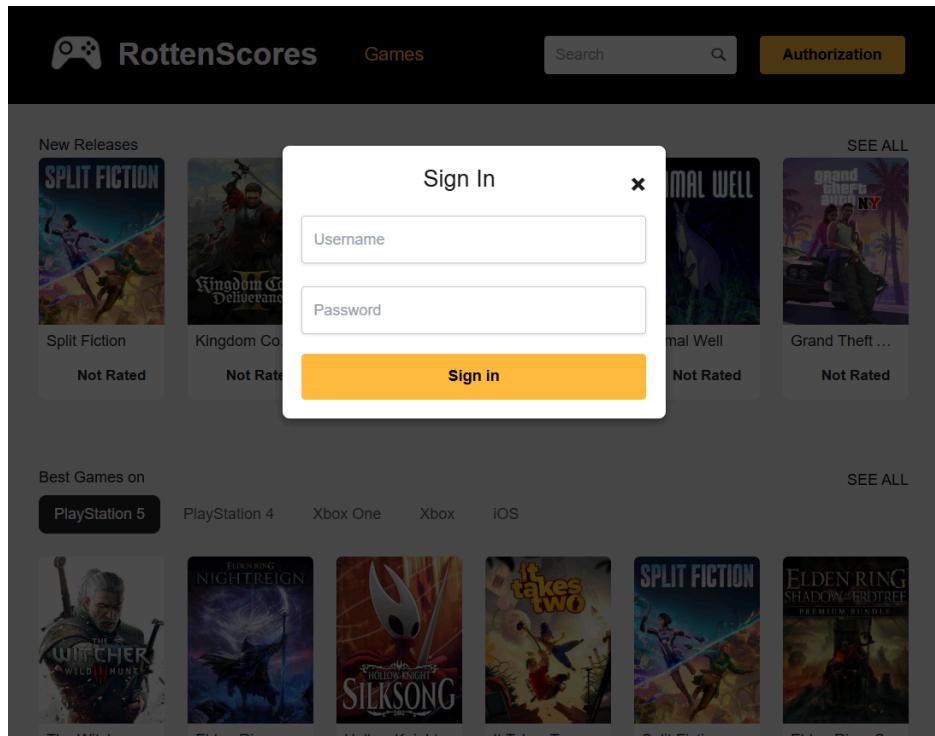


Рис. 12 - Страница авторизации

A screenshot of the RottenScores "My Profile" page. On the left, there is a sidebar with "My Profile" and "My Rating & Reviews" (which is currently selected). Below these are links for "My account" and "Statistics". At the bottom of the sidebar is a red "Sign Out" button. The main content area is titled "My Rating & Reviews" and shows three reviews. Each review includes a game thumbnail, the game's title, the date it was reviewed ("Reviewed – May 12, 2025"), the platform ("on PlayStation 5" or "on PlayStation 4"), a green rating box (containing the score), and two buttons: "Изменить" (Change) and "Удалить" (Delete). The reviews are for "The Witcher 3: Wild Hunt" (score 10), "Ghost of Tsushima" (score 7), and "The Last of Us Part II".

Рис. 13 - Личный кабинет с отзывами

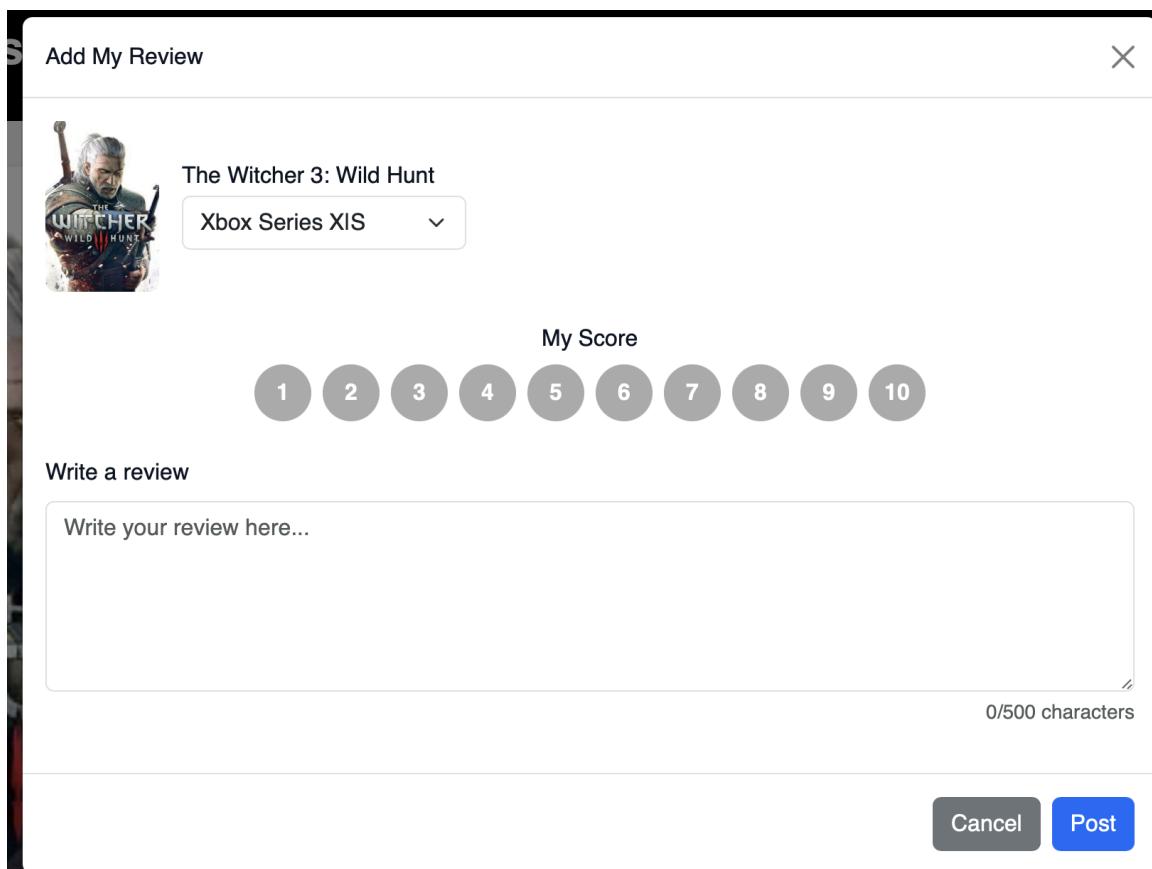


Рис. 14 - Добавление отзыва

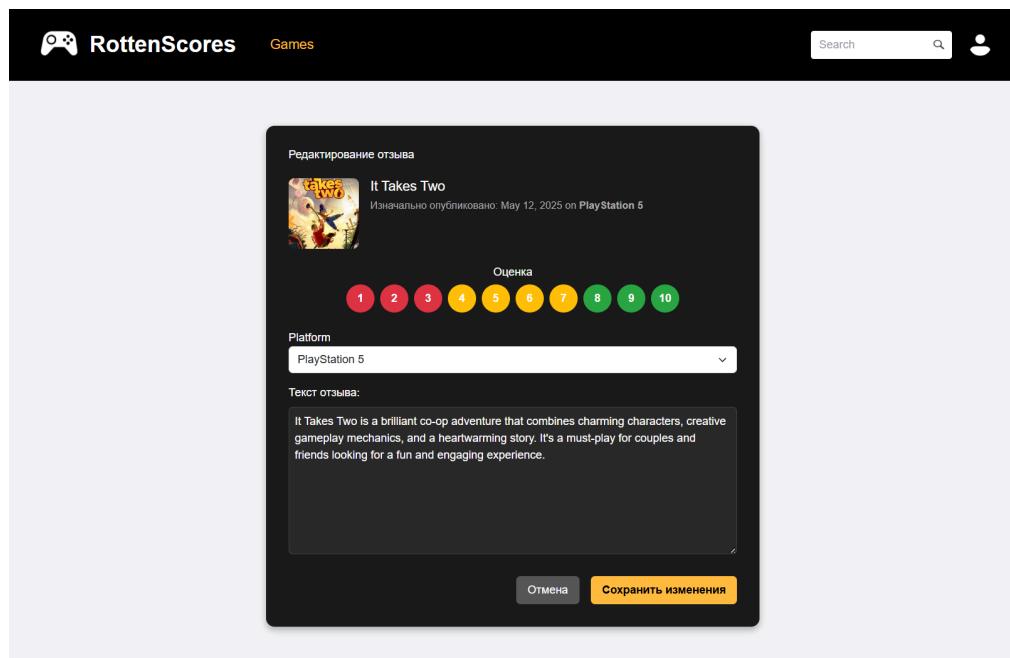


Рис. 15 - Редактирование отзыва

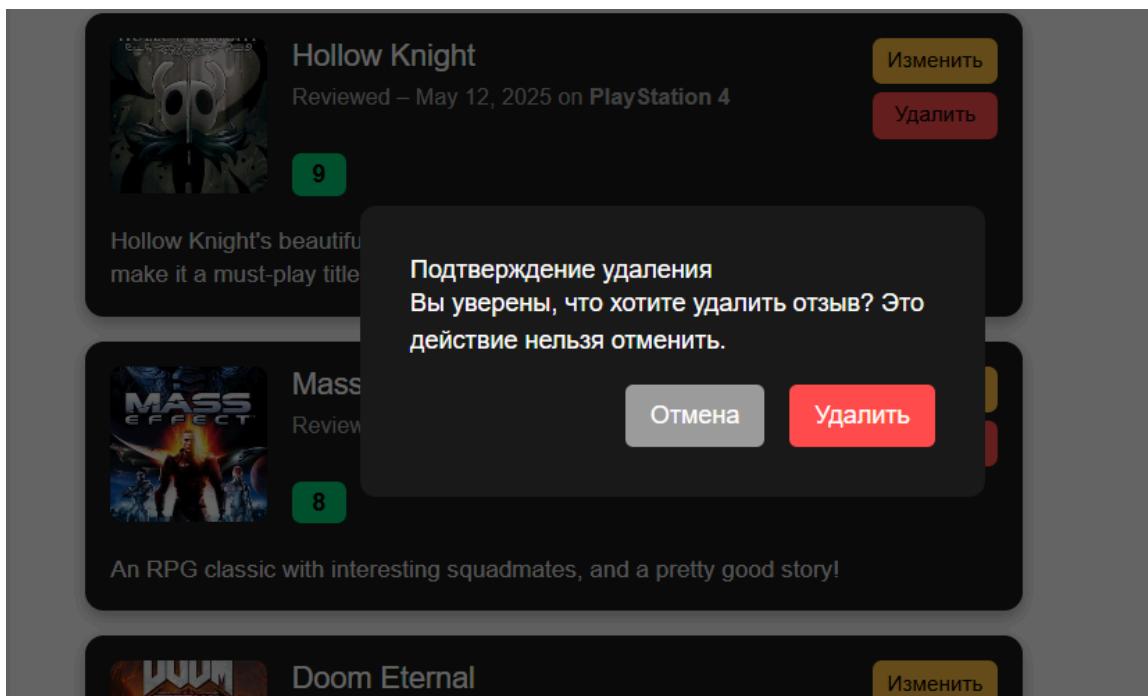


Рис. 16 - Удаление отзыва

A screenshot of the RottenScores website's user profile page. The left sidebar has tabs for 'My Profile', 'My Rating & Reviews', 'My account' (which is selected and highlighted in grey), and 'Statistics'. The main content area shows 'My Account' details: 'Created at: April 12, 2025, 5:47 p.m. | Last modified: April 12, 2025, 5:47 p.m.'. Below this are two tabs: 'Change Personal Info' (selected) and 'Change Password'. Under 'Change Personal Info', there are fields for 'Username' and 'Email', each with a placeholder and a note below it. A large orange 'Save' button is at the bottom. The top navigation bar includes a logo, a 'Games' link, a search bar, and a user icon.

Рис. 17 - Изменение данных пользователя

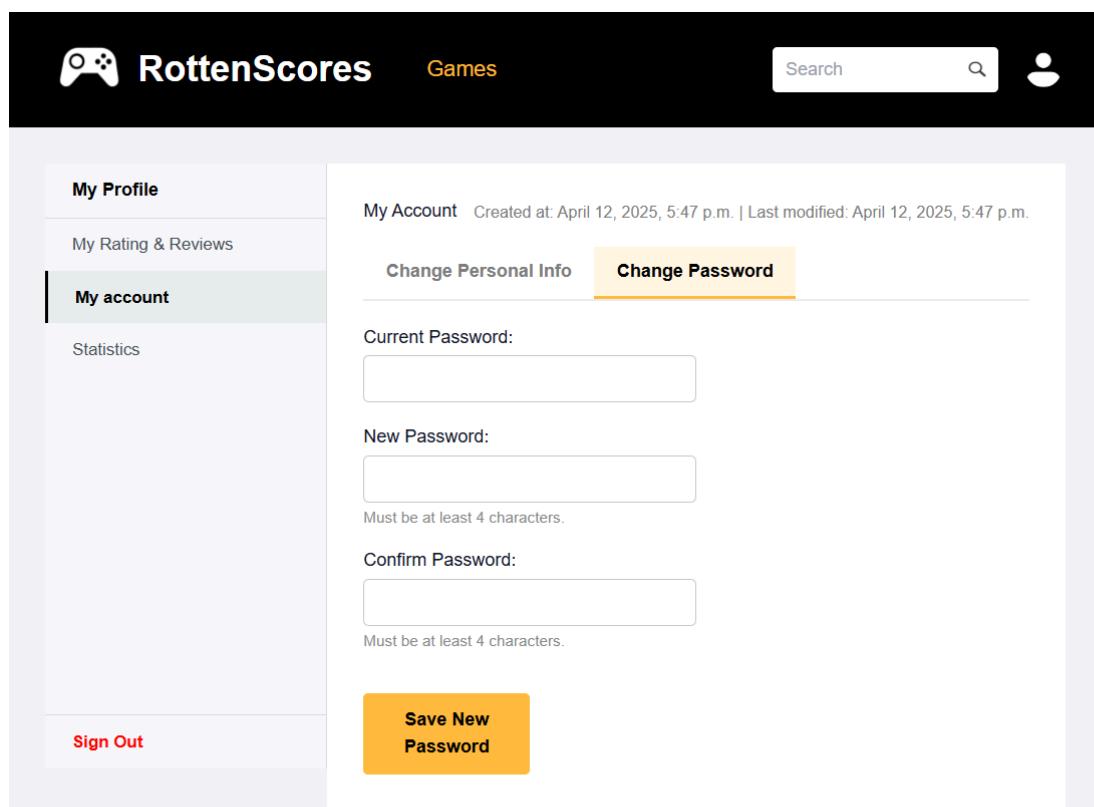


Рис. 18 - Изменение пароля аккаунта

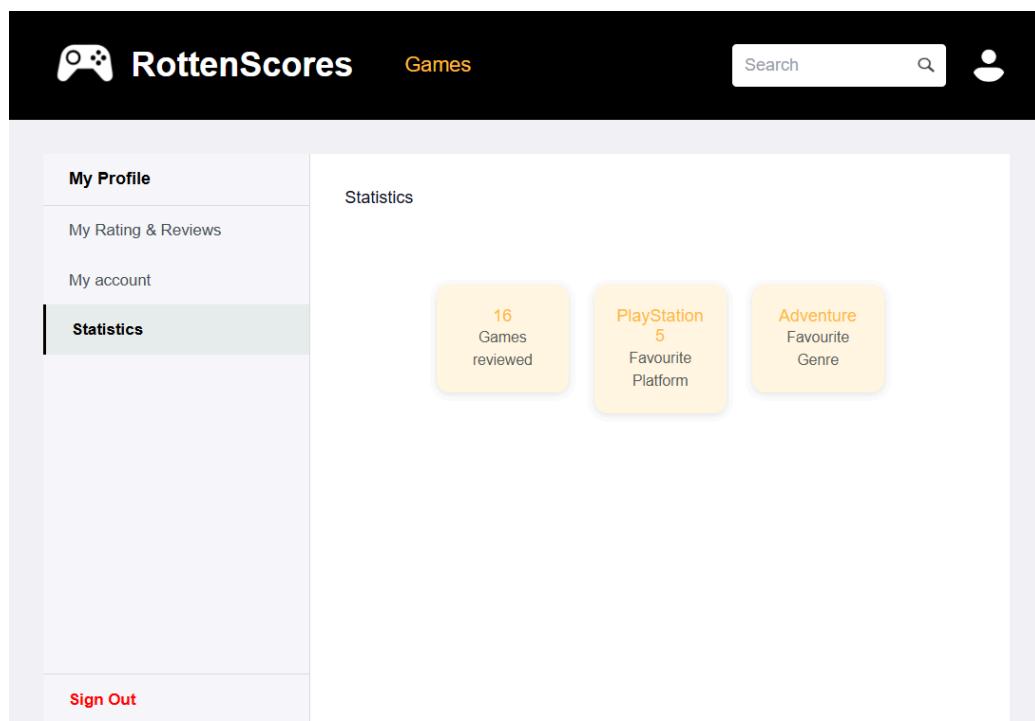


Рис. 19 - Статистика по отзывам пользователя

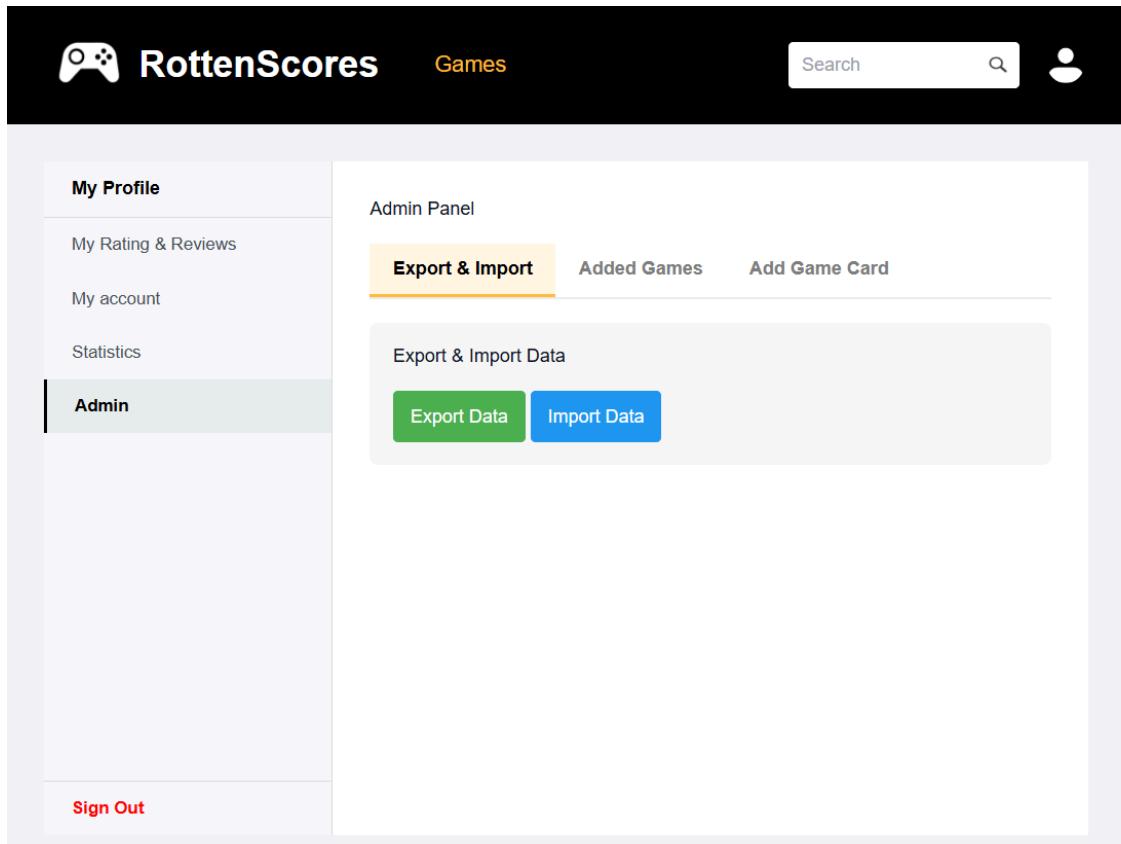


Рис. 20 - Экран экспорта и импорта

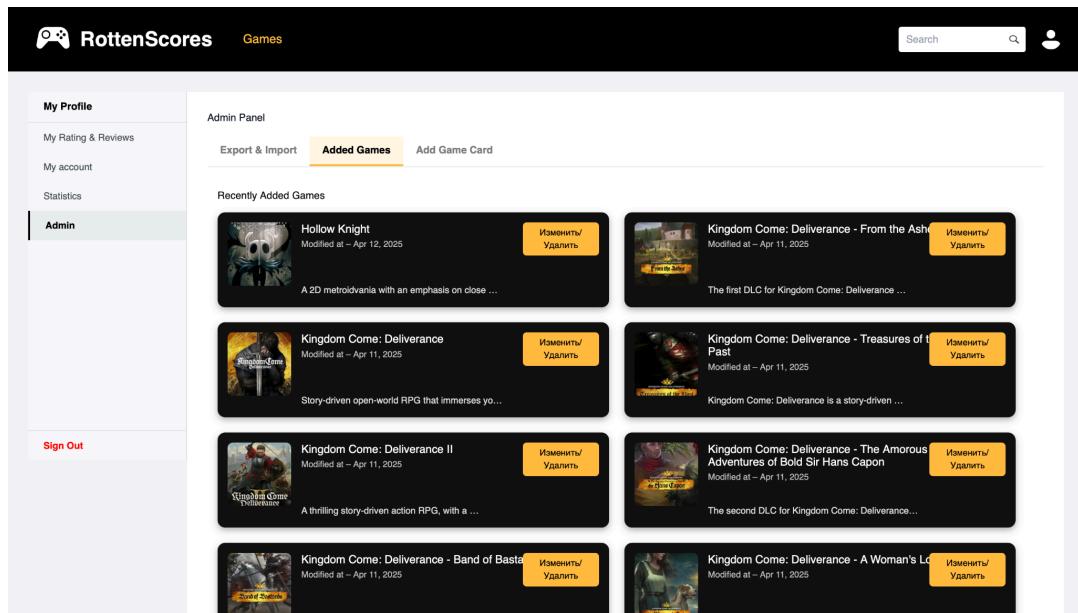


Рис. 21 - Экран добавленных игр

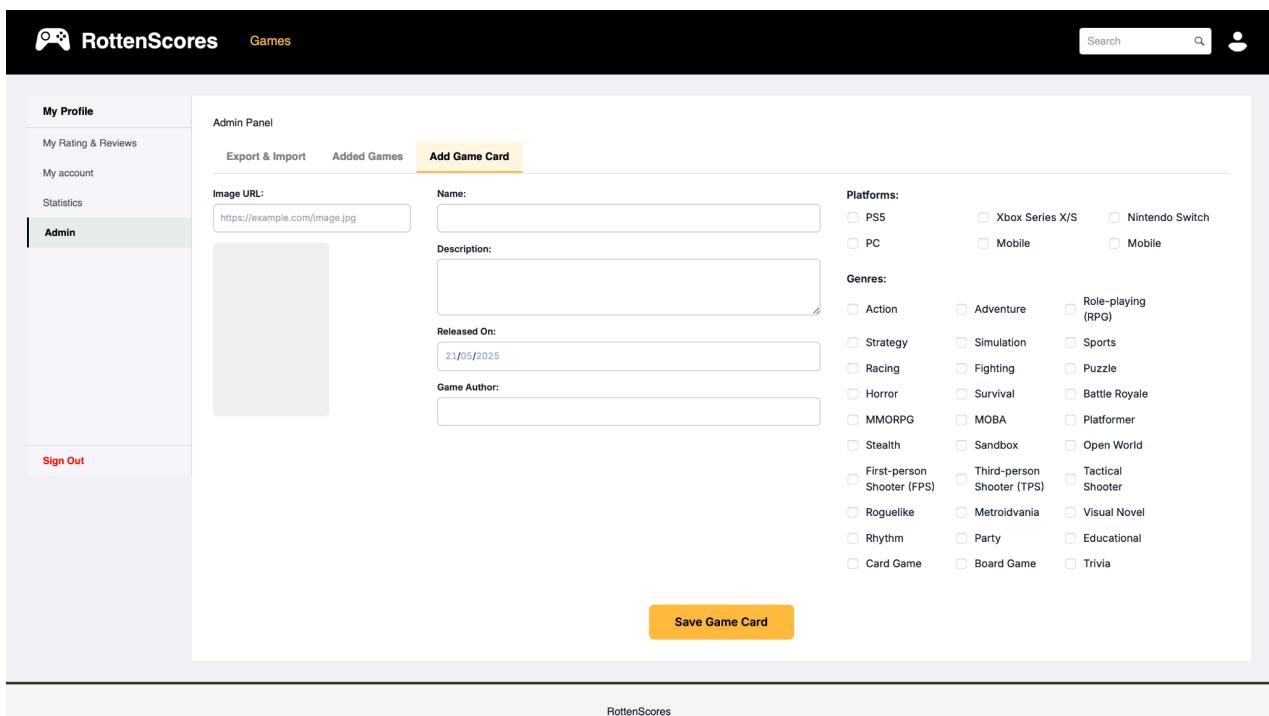


Рис. 22 - Экран добавления игры

5. Выводы

Достигнутые результаты

В рамках проекта был реализован веб-сервис для оценки и просмотра информации об играх, аналогичный *Metacritic*, с ориентацией на русскоязычную аудиторию. На текущем этапе достигнуты следующие результаты:

- Разработан функциональный веб-интерфейс с разделением прав доступа (пользователь, администратор);
- Реализована авторизации пользователей *Session-based*;
- Добавлены карточки игр с информацией о жанре, платформе, описании и оценках;

- Реализована система пользовательских и критических отзывов, отображение средней оценки и статистики;
- Разработан кабинет администратора, позволяющий управлять играми;
- Использована база данных *MongoDB*, обеспечивающая гибкость хранения отзывов, профилей и информации об играх.

Недостатки и пути для улучшения полученного решения

1. Нет регистрации пользователей;
2. Отсутствие адаптации под мобильные устройства. На рис. 1 видим, что треть пользователей используют мобильное приложение;
3. Отсутствие системы модерации отзывов;
4. Простая система поиска и фильтрации.

Будущее развитие решения

1. Создание рекомендательной системы на основе поведения и интересов пользователя и механизм похожих игр.
2. Социальная составляющая:
 - Возможность подписки на пользователей и критиков;
 - Комментарии и обсуждения под отзывами;
 - Система достижений и активностей.
3. Создание мобильного приложения на основе текущей версии;
4. Монетизация и продвижение:
 - Размещение рекламы, партнёрские программы;
 - Интеграция с игровыми магазинами;
 - SEO-оптимизация и таргетинг на русскоязычный рынок.

6. Приложения

Для запуска проекта используйте

```
docker compose build --no-cache && docker compose up
```

Как только запуск закончится (вывод в консоли остановится) переходите по адресу 127.0.0.1:8080.

7. Литература

1. Ссылка на репозиторий - <https://github.com/moevm/nosql1h25-review>
2. Документация MongoDB: <https://docs.mongodb.com/manual/>
3. Статистика *Metacritic*:

https://pro.similarweb.com/#/digitalsuite/websiteanalysis/overview/website-performance/*/999/1m?webSource=Total&key=metacritic.com