

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: Маркетплейс по продаже видеоигр.

Студент гр. 0304

Асташёнок М.С

Студент гр. 0304

Шквиря Е.В.

Студент гр. 0304

Нагибин И.С.

Преподаватель

Заславский М.М.

Санкт-Петербург

2023

ЗАДАНИЕ

Студенты

Асташёнок М.С.

Шквиря Е.В.

Нагибин И.С.

Группа 03804

Тема проекта: Маркетплейс по продаже игр.

Исходные данные: Необходимо реализовать маркетплейс по продаже игр, где будут собраны предложения от различных продавцов игр, которыми пользователь сможет воспользоваться для покупки игры.

Содержание пояснительной записи:

«Содержание»

«Введение»

«Сценарии использования»

«Модель данных»

«Разработанное приложение»

«Заключение»

«Приложения»

«Литература»

Предполагаемый объем пояснительной записи:

Не менее 10 страниц.

Дата выдачи задания: 20.09.2023

Дата сдачи реферата: 27.12.2023

Дата защиты реферата: 27.12.2023

Студент гр. 0304

Студент гр. 0304

Студент гр. 0304

Преподаватель

Асташёнок М.С.

Шквиря Е.В.

Нагибин И.С.

Заславский М.М.

АННОТАЦИЯ

В рамках данного курса была поставлена задача разработать веб-приложение, реализующее функционал для заведения новых пользователей в систему и предоставления каталога игр с предложениями от различных продавцов цифровых версий игр (напр. ключей активации). Веб-приложение имеет следующий функционал: работа с базой данных пользователей и игр, сервис с авторизацией, личный кабинет, каталог игр и инструмент для заведения новых игр в систему.

SUMMARY

As part of this course, the task was set to develop a web application that implements functionality for adding new users to the system and providing a catalog of games with offers from various sellers of digital versions of games (for example, activation keys). The web application has the following functionality: working with a database of users and games, a service with authorization, a personal account, a catalog of games, and a tool for adding new games to the system.

СОДЕРЖАНИЕ

Задание	2
Аннотация	4
1. Введение	7
2. Сценарии использования	8
2.1. Макет UI	8
2.2. Описание сценариев использования	8
2.2.1. Покупка игры	8
2.2.2. Удаление комментария по жалобе	9
2.2.3. Просмотр статистики продаж на сайте	9
2.2.4. Обращение в поддержку	9
2.2.5. Ответ на обращение в поддержку	9
3. Модель данных	11
3.1. Нереляционная модель данных	11
3.1.1. Графическое представление	11
3.1.2. Описание назначений коллекций, типов данных и сущностей	11
3.1.3. Оценка объема информации, хранимой в модели	14
3.1.4. Примеры запросов	15
3.2. Реляционная модель данных	18
3.2.1. Графическое представление	18
3.2.2. Описание назначений типов данных и сущностей	19
3.2.3. Оценка объема информации, хранимой в модели	22
3.2.4. Примеры запросов	22
3.3. Сравнение моделей	27
3.3.1. Удельный объем информации	27
3.3.2. Запросы по отдельным юзкейсам	28
3.3.3. Вывод	29

4.	Разработанное приложение	30
4.1.	Описание приложения	30
4.2.	Использованные технологии	31
4.3.	Снимки экрана приложения	31
5.	Выводы	38
5.1.	Достигнутые результаты	38
5.2.	Недостатки и пути для улучшения полученного решения	38
5.3.	Будущее развитие решения	39
	Литература	40
	Приложения	41

1. ВВЕДЕНИЕ

Цель работы - разработка онлайн-сервиса цифрового распространения компьютерных игр. Сервис должен продавать ключи к играм и подбирать для пользователя предложения других сайтов по продаже игр.

Благодаря подбору предложений с других сайтов наш сервис помогает найти желаемые игры по низкой цене.

Для достижения поставленной цели необходимо, чтобы у пользователя была возможность искать игры и просматривать для них предложения на сайте.

Качественные требования к решению: требуется разработать приложение с использованием СУБД MongoDB и реализовать развертывание приложения через docker-compose.

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

2.1. Макет UI

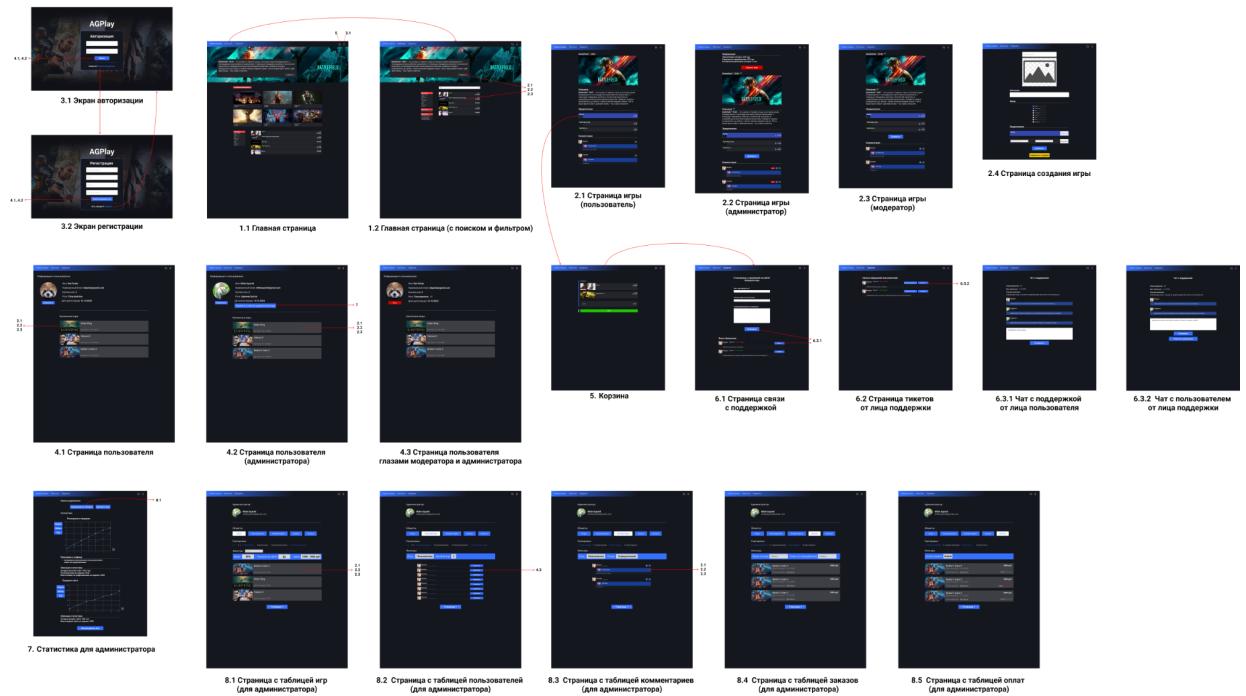


Рисунок 1 - Макет UI.

2.2. Описание сценариев использования

2.2.1 Покупка игры

Действующее лицо: **Пользователь**

- Пользователь заходит на сайт.
- Пользователь выбирает игру с главной страницы.
- Пользователь выбирает предложение AGPlay (нашего сайта).
- Пользователь входит в свой аккаунт.
- Пользователь переходит в корзину.
- Пользователь оплачивает игру.

2.2.2 Удаление комментария по жалобе

Действующее лицо: **Модератор**

- Модератор получает письмо на почту о плохом комментарии.

- Модератор входит на сайт через свой аккаунт.
- Модератор переходит к игре и находит комментарий, на который пожаловались.
- Модератор удаляет комментарий аморального пользователя.
- Модератор отвечает на жалобное письмо.

2.2.3 Просмотр статистики продаж на сайте

Действующее лицо: **Администратор**

- Администратор переходит на сайт через свой профиль.
- Администратор переходит на страницу своего профиля.
- Администратор нажимает кнопку “Перейти в панель администратора”.
- Администратор отправляет скриншот статистики.

2.2.4 Обращение в поддержку

Действующее лицо: **Пользователь**

- Пользователь видит неработающий элемент сайта, неработающую ссылку или контент, нарушающий правила сайта.
- Пользователь нажимает кнопку “Поддержка” в шапке сайта.
- Пользователь заполняет форму и в ней описывает проблему.
- Пользователь получает сообщение о решении проблемы администраторами.

2.2.5 Ответ на обращение в поддержку

Действующее лицо: **Работник поддержки**

(Модератор/Администратор)

- Работник поддержки видит обращение от пользователя о проблеме на сайте.

- Работник поддержки проверяет, действительно ли данная проблема существует.
- Работник поддержки отвечает, что проблема будет исправлена в ближайшее время.
- Работник поддержки пишет пользователю о решении проблемы, когда она будет исправлена.
- Работник поддержки закрывает задачу.

3. МОДЕЛЬ ДАННЫХ

3.1. Нереляционная модель данных

3.1.1. Графическое представление

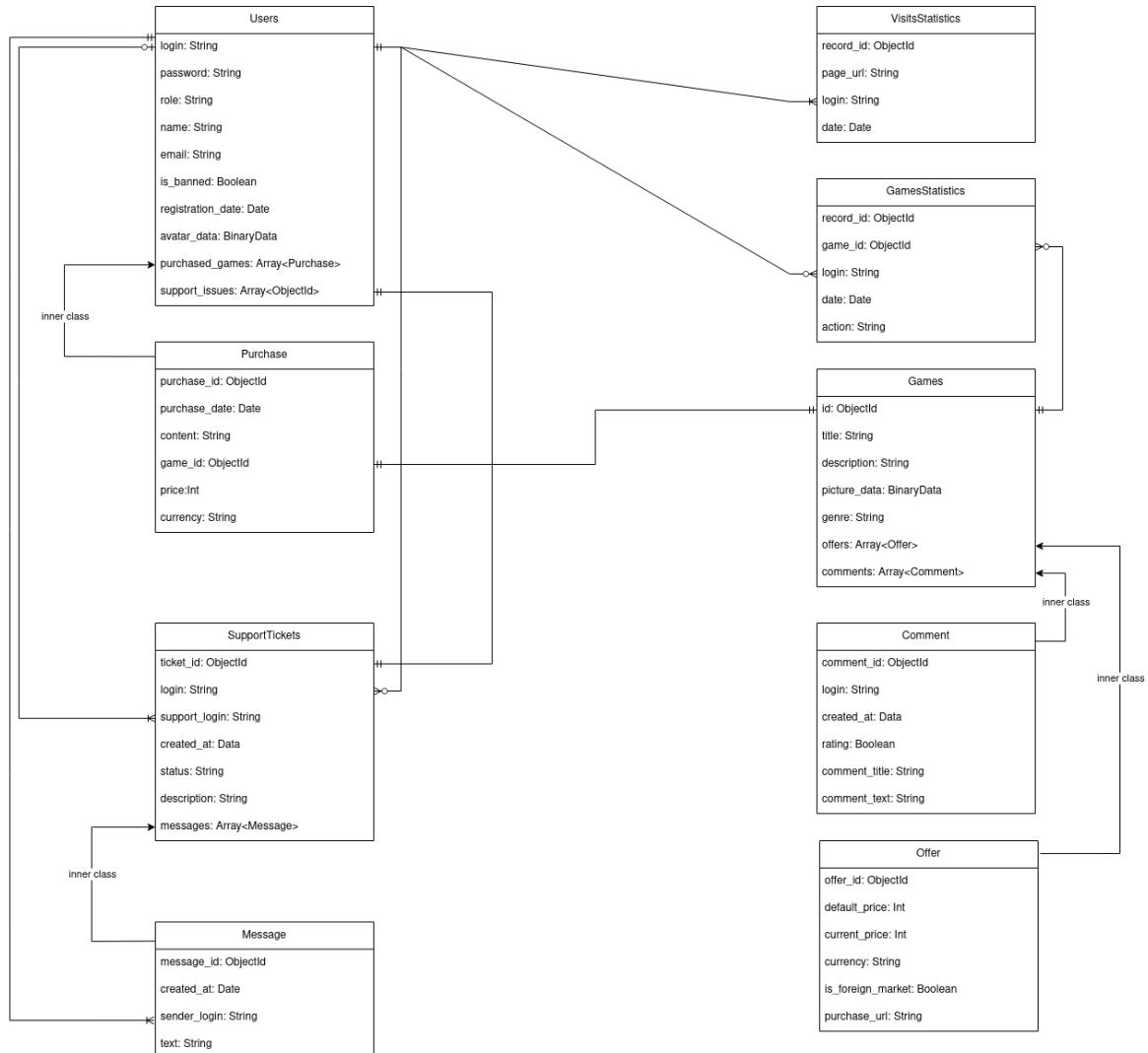


Рисунок 2 - Модель нереляционной БД

3.1.2. Описание модели

Коллекция "Users":

- login: String - логин пользователя.
- password: String - пароль пользователя.

- role: String - роль пользователя (Пользователь, Администратор, Модератор).
- name: String - никнейм пользователя.
- email: String - привязанная почта пользователя.
- is_banned: Boolean - заблокирован ли пользователь.
- registration_date: Date - дата регистрации пользователя.
- avatar_data: String - данные аватарки.
- purchased_games: Array - список покупок.
 - purchase_id: ObjectId - идентификатор покупки.
 - purchase_date: Date - дата получения товара.
 - content: String - содержимое покупки (в большинстве - ключ активации).
 - game_id: ObjectId - идентификатор игры.
 - price: Int - стоимость.
 - currency: String - валюта.
- support_issues: Array<ObjectId> - идентификаторы обращений в поддержку (тиков).

Коллекция "Games":

- id: ObjectId - идентификатор игры.
- title: String - название игры.
- description: String - описание игры.
- picture_data: BinaryData - данные изображения.
- genre: String - жанр игры.
- offers: Array - структура предложений для покупки игры.
 - offer_id: ObjectId - идентификатор предложения.
 - default_price: Int - стандартная цена игры.
 - current_price: Int - текущая цена игры.

- currency: String - валюта.
 - is_foreign_market: Boolean - предложение с стороннего сайта или внутренний товар AGPlay.
 - purchase_url: String - ссылка на банковскую транзакцию или ссылка на игру на сайте стороннего продавца.
- comments: Array - список комментариев к игре.
 - comment_id: ObjectId - идентификатор комментария.
 - login: String - логин пользователя.
 - created_at: Date - дата написания комментария.
 - rating: Boolean - оценка игры.
 - comment_title: String - заголовок отзыва.
 - comment_text: String - текст комментария к игре (может быть свободно составлен рецензентом).

Коллекция "SupportTickets" :

- ticket_id: ObjectId - идентификатор обращения.
- login: String - логин пользователя.
- support_login: String - логин сотрудника поддержки.
- created_at: Date - дата создания обращения.
- status: String - статус обращения ("Не в работе", "В работе", "Решен").
- description: String - описание проблемы.
- messages: Array - список сообщений в чате.
 - message_id: ObjectId - идентификатор сообщения.
 - created_at: Date - дата написания сообщения.
 - sender_login: String - идентификатор отправителя.
 - text: String - текст сообщения.

Коллекция "VisitsStatistics":

- record_id: ObjectId - идентификатор записи.
- page_url: String - url посещенной страницы.
- login: String - логин пользователя.
- date: Date - дата создания записи.

Коллекция "GamesStatistics" :

- record_id: ObjectId - идентификатор записи.
- game_id: ObjectId - идентификатор посещённой игры.
- login: String - логин пользователя.
- date: Date - дата создания записи.
- action: String - тип действия. Список возможных действий:
"offer_click" - клик по предложению, "game_view" - просмотр,
"game_purchase" - покупка.

3.1.3. Оценка объема информации, хранимой в модели

Рассчитаем объем каждой сущности:

- Users: $N_6 * (1048740 + 72 * N_1 + 12 * N_2 \text{ байт})$ [N_6 - Количество пользователей; N_2 - количество обращений; N_1 - количество покупок]
- Games: $N_7 * (1049804 + 281 * N_3 + 595 * N_4 \text{ байт})$ [N_7 - Количество игр; N_3 - количество предложений; N_4 - количество комментариев]
- SupportTickets: $N_8 * (608 + 562 * N_5 \text{ байт})$ [N_8 - Количество обращений; N_5 - количество сообщений]
- VisitsStatistics: $N_9 * 306 \text{ байт}$ [N_9 - Количество записей посещения]

- GamesStatistics: N10 * 76 байт [N10 - Количество записей действий с играми]

Рассчитаем избыточность модели. Будем называть поля "избыточными", если они содержат дубликаты ключей других коллекций. Таким образом, полученная избыточность равна 1,0056.

Модель больше всего набирает в объеме занимаемой памяти при увеличении количества пользователей Users, т.к. при регистрации нового пользователя необходимо выделить минимум 1048740 байтов памяти и экземпляры данной коллекции появляются с высокой частотой.

3.1.4. Примеры запросов

Получение информации о пользователе::

```
db.users.find({  
    login: "pro100Kot"  
})
```

Получение игры из коллекции games по идентификатору:

```
db.games.find({  
    id: 123  
})
```

Получение игры из коллекции games по его названию:

```
db.games.find({  
    title: "Battlefield"  
})
```

Покупка желаемой игры:

```
db.users.updateOne(  
    {login: "ogyrec"},  
    {$push:  
        {  
            purchased_games:
        }}
```

```

        {
            purchase_date: Date(),
            content:
        "ASD-JFS-KV3-KKK-374",
            game_id: 3,
            price: Int: 300,
            currency: "RUB"
        }
    )
)

```

Добавление новой игры:

```

db.games.insertOne({
    title: "Counter Strike 3.1",
    description: "Гейб Ньюэлл продолжает легендарную франшизу Counter Strike, выпуская очередную версию нашумевшего шутера.",
    picture_data: <...>,
    genre: "Шутер",
    offers: [],
    comments: []
})

```

Добавление предложений к игре:

```

db.games.updateOne(
    {id: 1},
    {
        $push{
            offers:
            {
                default_price: 2048,
                current_price: 1024,
                currency: "RUB",
                is_foreign_market: 0,
                purchase_url: "ссылка_на_покупку"
            }
        }
    }
)

```

Удаление игры:

```
db.games.removeOne({  
    id: 123  
})
```

Добавление комментария в коллекции games:

```
db.games.insertOne({  
    login:"Krytoi{P}erec",  
    created_at: Date(),  
    rating: true,  
    comment_title: "Игра года",  
    comment_text: "Крутая игра, куплю своей  
сестре"  
})
```

Удаление комментария в коллекции games:

```
db.games.remove({  
    comments:{  
        comment_id: 3  
    }  
,  
{  
    justOne: true  
})
```

Создание нового обращения в поддержку:

```
db.supportTickets.insertOne({  
    login: "ogyrec",  
    support_login: "",  
    created_at: Date(),  
    status: "Не решен",  
    description: "Не купилась игра, верните  
деньги",  
    messages: []  
})
```

Смена статуса обращения в поддержку:

```
db.supportTickets.updateOne({  
    ticket_id: 2  
,  
})
```

```
{  
    status: "Решен"  
} )
```

Отправка сообщения:

```
db.supportTickets.updateOne(  
    {  
        ticket_id: 2  
    },  
    {  
        $push:  
            comments:  
                created_at: Date(),  
                sender_login:  
"support_Artem_Mal'kov",  
                text: "Добрый день, мы сделаем  
всё, что в наших силах, чтобы помочь"  
    }  
}
```

Бан пользователя:

```
db.users.updateOne(  
    {login: "BAD_BOY777"},  
    {  
        is_banned: 1  
    })
```

3.2. Реляционная модель данных

3.2.1. Графическое представление

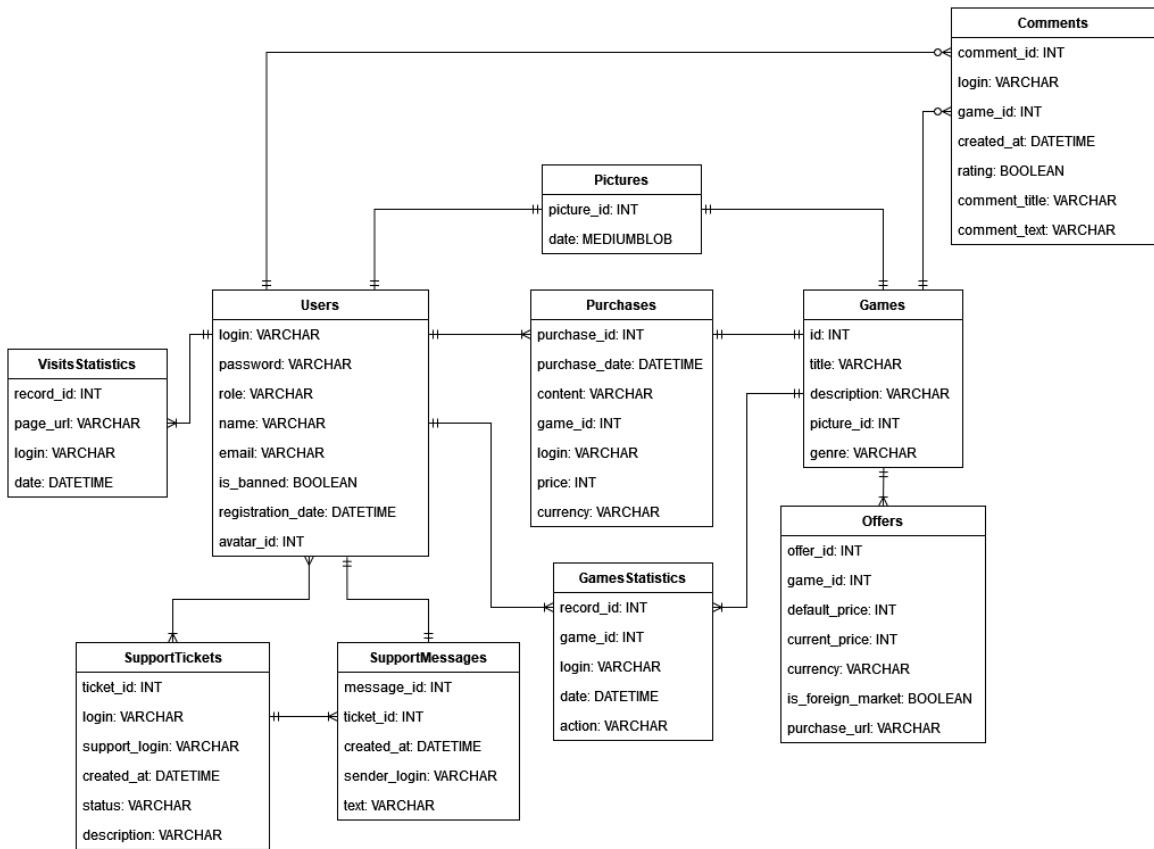


Рисунок 3 - Модель реляционной БД

3.2.2. Описание назначений типов данных и сущностей

Таблица "Users":

- login: Varchar - логин пользователя.
- password: Varchar - пароль пользователя.
- role: Varchar - роль пользователя (Пользователь, Администратор, Модератор).
- name: Varchar - никнейм пользователя.
- email: Varchar - привязанная почта пользователя.
- is_banned: Boolean - заблокирован ли пользователь.
- registration_date: Datetime - дата регистрации пользователя.
- avatar_id: Int - идентификатор аватарки.

Таблица "Games":

- logId - уникальный идентификатор
- changeDate - время лога
- action - содержание лога
- userId - каким админом были произведены изменения (в зависимости от лога , может быть NULL, связь один ко многим)
- tableId - в какой таблице произошли изменения (в зависимости от лога , может быть NULL, связь один ко многим)

Таблица "Purchases":

- purchase_id: Int - идентификатор покупки.
- purchase_date: Datetime - дата получения товара.
- content: Varchar - содержимое покупки (в большинстве - ключ активации).
- game_id: Int - идентификатор игры.
- login: Varchar - идентификатор пользователя.
- price: Int - стоимость.
- currency: String - валюта.

Таблица "SupportTickets":

- ticket_id: Int - идентификатор обращения.
- login: Varchar - идентификатор пользователя.
- support_login: Varchar - идентификатор сотрудника поддержки.
- created_at: Datetime - дата создания обращения.
- status: Varchar - статус обращения ("Не в работе", "В работе", "Решен").
- description: Varchar - описание проблемы.

Таблица "SupportMessages":

- message_id: Int - идентификатор сообщения.
- ticket_id: Int - идентификатор обращения.
- created_at: Datetime - дата написания сообщения.
- sender_login: Varchar - идентификатор отправителя.
- text: Varchar - текст сообщения.

Таблица "Comments":

- comment_id: Int - идентификатор комментария.

- login: Varchar - идентификатор пользователя.
- game_id: Int - идентификатор игры.
- created_at: Datetime - дата написания комментария.
- rating: Boolean - оценка игры.
- comment_title: Varchar - заголовок отзыва.
- comment_text: Varchar - текст комментария к игре (может быть свободно составлен рецензентом).

Таблица "Pictures":

- picture_id: Int - идентификатор картинки.
- data: MediumBlob - файл картинки.

Таблица "VisitsStatistics":

- record_id: Int - идентификатор записи.
- page_url: Varchar - url посещенной страницы.
- login: Varchar - идентификатор пользователя.
- date: Datetime - дата создания записи.

Таблица "GamesStatistics":

- record_id: Int - идентификатор записи.
- game_id: Int - идентификатор посещённой игры.
- login: Varchar - идентификатор пользователя.
- date: Datetime - дата создания записи.
- action: Varchar - тип действия. Список возможных действий: "offer_click" - клик по предложению, "game_view" - просмотр, "game_purchase" - покупка.

Таблица "Offers":

- offer_id: Int - идентификатор предложения.
- game_id: Int - идентификатор игры.
- default_price: Int - стандартная цена игры.
- current_price: Int - текущая цена игры.
- currency: Varchar - валюта.
- is_foreign_market: Boolean - предложение с стороннего сайта или внутренний товар AGPlay.

- purchase_url: Varchar - ссылка на банковскую транзакцию или ссылка на игру на сайте стороннего продавца.

3.2.3. Оценка объема информации, хранимой в модели

Рассчитаем объем каждой сущности:

- Users: N6
- Games: N7
- Purchases: N1 * N6
- SupportTickets: N8
- SupportMessages: N5 * N8
- Comments: N7 * N4
- Pictures: N6 + N7
- VisitsStatistics: N9
- GamesStatistics: N10

Тогда объем данных в БД: $V=9 * N1 * N6 + 78 * N10 + 277 * N3 * N7 + 609 * N4 * N7 + 558 * N5 * N8 + 1048748 * N6 + 1049804 * N7 + 600 * N8 + 298 * N9.$

А избыточность модели равна 1,0159.

3.2.4. Примеры запросов

Вход пользователя в систему

```
SELECT * FROM users
JOIN purchases USING (login)
WHERE users.login = "логин_пользователя";
```

Информация о пользователях

```
INSERT INTO pictures (data)
VALUES ("бинарные_данные_изображения");
```

```
UPDATE users

SET users.avatar_id = (
    SELECT picture_id
    FROM pictures
    WHERE picture_id = LAST_INSERT_ID();
) AS picture_id

WHERE users.login = "логин_пользователя"
```

Бан пользователя

```
UPDATE users

SET users.is_banned = True

WHERE users.login = "логин пользователя"
```

Взаимодействие с играми

Получение игры из коллекции games по идентификатору:

```
SELECT * FROM games

JOIN offers USING (game_id)

JOIN comments USING (game_id)

WHERE games.game_id = "идентификатор_игры";
```

Получение игры из коллекции games по его названию:

```
SELECT * FROM games

JOIN offers USING (game_id)

JOIN comments USING (game_id)
```

```
WHERE games.title = "название_игры";
```

Покупка желаемой игры:

```
INSERT INTO purchases (login, game_id, content,
price, currency)

VALUES (
    "логин_пользователя",
    "идентификатор_игры",
    "содержимое_покупки",
    "цена_игры",
    "валюта"

);
```

Добавление новой игры:

```
INSERT INTO pictures (data)

VALUES ("бинарные_данные_изображения");

INSERT INTO games (title, description, genre,
picture)

VALUES (
    "название_игры",
    "описание_игры",
    "жанр_игры",
    (

        SELECT picture_id
        FROM pictures
        WHERE picture_id = LAST_INSERT_ID();
```

```
) AS picture_id  
);
```

Добавление предложений к игре:

```
INSERT INTO offers (  
    game_id,  
    default_price,  
    current_price,  
    currency,  
    is_foreign_market,  
    purchase_url  
)  
VALUES (  
    "идентификатор_игры",  
    "стандартная_цена",  
    "текущая_цена",  
    "валюта",  
    "является_внешним_магазином",  
    "ссылка_на_покупку"  
)
```

Удаление игры:

```
DELETE FROM games  
WHERE games.game_id = "идентификатор игры";
```

```
DELETE FROM offers  
WHERE offers.game_id = "идентификатор игры";
```

Добавление комментария к игре:

```
INSERT INTO comments (login, game_id, rating,  
comment_title, comment_text)  
VALUES (  
    "логин_пользователя",  
    "идентификатор_игры",  
    "оценка_игры",  
    "заголовок_комментария",  
    "текст_комментари"  
) ;
```

Удаление комментария:

```
DELETE FROM comments  
WHERE comments.comment_id =  
"идентификатор_комментария";
```

Создание нового обращения в поддержку:

```
INSERT INTO support_tickets(login, description)  
VALUES ("идентификатор_отправителя",  
"текст_обращения");
```

Добавление нового сообщения в диалог обращения с поддержкой:

```

INSERT INTO support_messages(sender_login, text)
VALUES ("идентификатор_отправителя",
"текст_сообщения")

WHERE support_messages.ticket_id =
"идентификатор_обращения";

```

Смена статуса обращения в поддержку:

```

UPDATE support_tickets
SET support_tickets.status = "статус_обращения"
WHERE support_tickets.ticket_id =
"идентификатор_обращения";

```

3.3. Сравнение моделей

3.3.1. Удельный объем информации

Если сравнивать MongoDB и MySQL, то можно заметить, что для реляционной версии базы данных требуется сильно меньше памяти.

- **Нереляционная модель:** $72 * N1 * N6 + 76 * N10 + 12 * N2 * N6 + 281 * N3 * N7 + 595 * N4 * N7 + 562 * N5 * N8 + 1048740 * N6 + 1049804 * N7 + 608 * N8 + 306 * N9$ байт.
- **Реляционная модель:** $79 * N1 * N6 + 78 * N10 + 277 * N3 * N7 + 609 * N4 * N7 + 558 * N5 * N8 + 1048748 * N6 + 1049804 * N7 + 600 * N8 + 298 * N9$ байт.

Если брать в расчёт подстановку верхних значений для всех переменных, кроме количества пользователей, то можно получить разницу занимаемой памяти между двумя моделями:

- Для нереляционной модели: $N6 * 1081948 + 37842520000$.
- Для реляционной модели: $N6 * 1088248 + 37958380000$.

- Разница (MongoDB - MySQL): $N6 * 1081948 + 37842520000 - (N6 * 1088248 + 37958380000) = -6300 * N6 - 115860000$.

Как видно из выражения разницы занимаемой памяти для MongoDB и MySQL выше, объем памяти, занимаемый реляционной моделью больше, чем занимаемый нереляционной моделью.

3.3.2. Запросы по отдельным юзкейсам

Таблица №1 - Количество запросов для совершения Use Case

Запросы к БД	MySQL	MongoDB
Получение информации о пользователе	1	1
Смена картинки пользователя	3	1
Блокировка пользователя	1	1
Получение игры из коллекции <i>games</i> по идентификатору	1	1
Получение игры из коллекции <i>games</i> по названию	1	1
Покупка желаемой игры	1	1
Добавление новой игры	3	1
Добавление предложений к игре	1	1
Удаление игры	2	1
Создание нового обращения в поддержку	1	1
Добавление комментария к игре	1	1
Добавление нового сообщения в диалог с поддержкой	1	1
Смена статуса обращения в поддержку	1	1
Сумма	18	13

Как видно из таблицы 1, количество запросов, требуемых для совершения операций в MySQL больше, чем для MongoDB.

3.3.3. Вывод

Как у реляционной, так и у нереляционной реализации разрабатываемой базы данных есть свои преимущества и недостатки:

- Реляционная база данных на MySQL требует больше памяти, например для хранения идентификаторов сущностей, а также имеет низкую скорость работы и сложную структуру.
- Нереляционная база данных на MongoDB наоборот - быстрее в обращении и требует меньше памяти.
- Для выполнения одного use case необходимое количество запросов к реляционной модели либо совпадает с количеством запросов к нереляционной, либо превышает.

Подводя итоги, можно сказать, что MongoDB для данной задачи будет более удачным выбором.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

4.1. Описание приложения

Веб-интерфейс представляет собой приложение, состоящее из двух частей: back-end и front-end.

Front-end реализован на фреймворке React. В качестве контроля состояния использовались React Hooks. В качестве сборщика выступает сборщик по умолчанию для React - Babel.

Back-end написан на NodeJS, веб-сервер реализован с помощью ExpressJS, взаимодействие с базой данных реализовано с помощью библиотеки Mongoose

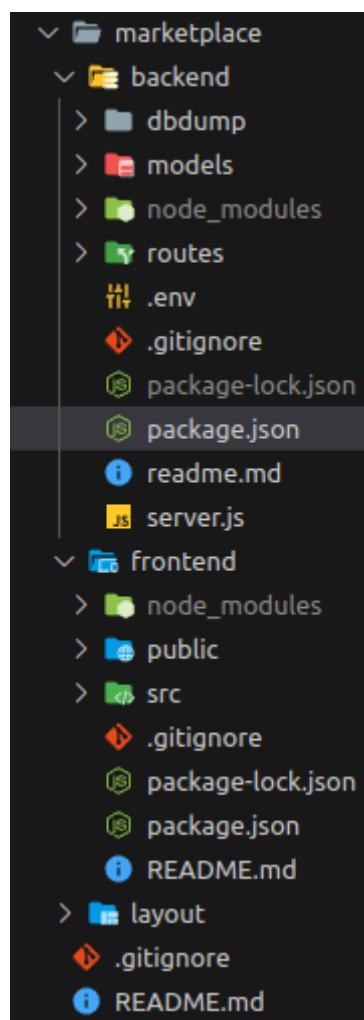


Рисунок 4 - Структура веб-приложения.

Как видно из рис.4, структура приложения состоит из 3 директорий:

1. marketplace/backend - содержит в себе реализацию серверной стороны приложения - запросы, MongoDB-модели и dump-база данных.
2. marketplace/frontend - содержит в себе реализацию front-end стороны приложения - приложение на React и скрипты с запросами на сервер.
3. marketplace/layout - содержит в себе верстку страниц сайта - .html страницы и .css стили для этих страниц.

4.2. Использованные технологии

БД: MongoDB, Mongoose.

Back-end: NodeJS, ExpressJS.

Front-end: HTML, CSS, ReactJS.

4.3. Снимки экрана приложения

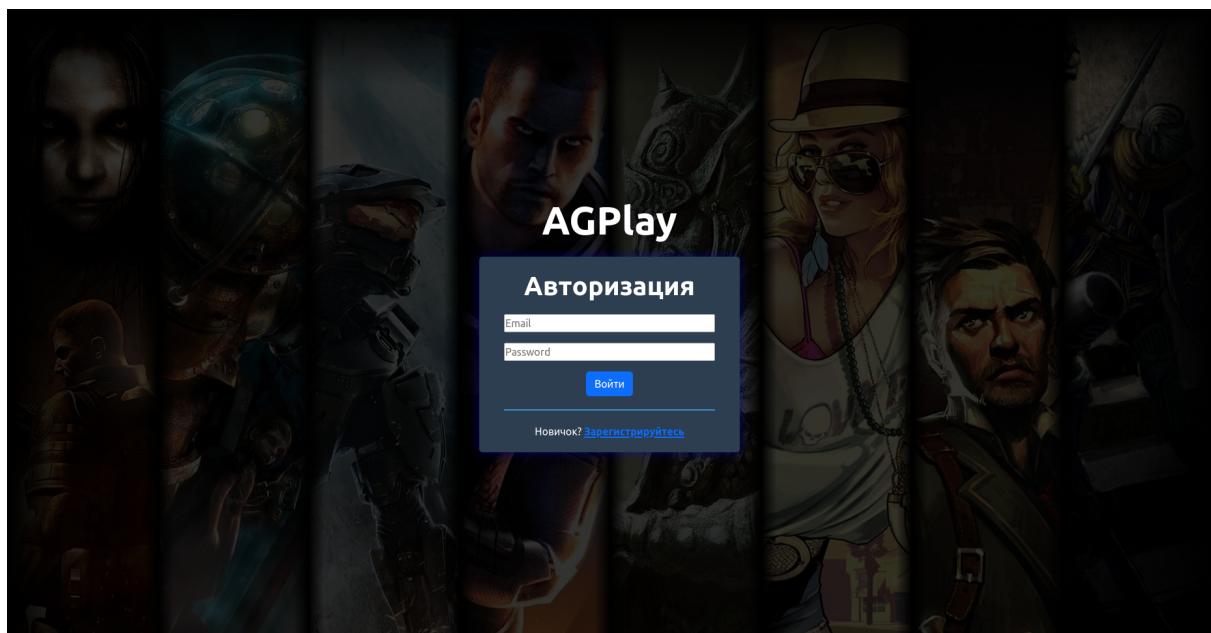


Рисунок 5 - Экран авторизации.

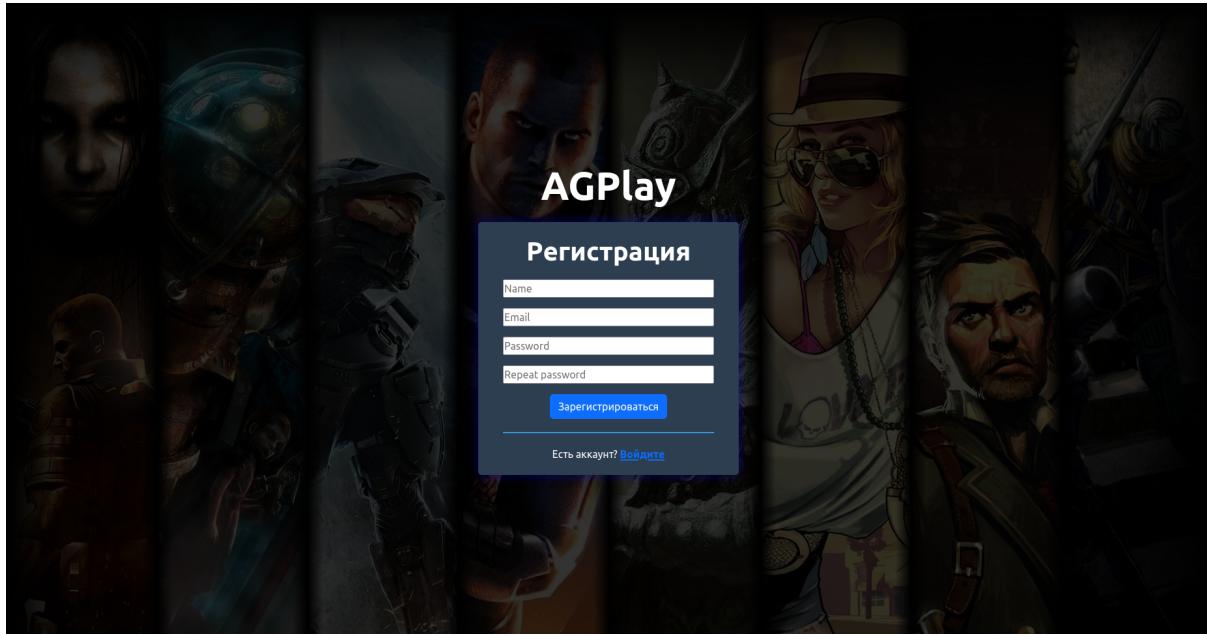


Рисунок 6 - Экран регистрации

A screenshot of the user profile page. At the top, there are navigation links: "Главная страница", "Профиль", and "Создать игру". The main section is titled "Информация о пользователе". It shows a placeholder user icon (a white person silhouette in a blue circle). To the right, the user's details are listed: Имя: Михаил Асташёнов, Электронная почта: mikeastademo@gmail.com, Куплено игр: 0, Роль: user, and Дата регистрации: 2023-12-27T13:02:35.261Z.

Рисунок 7 - Страница пользователя

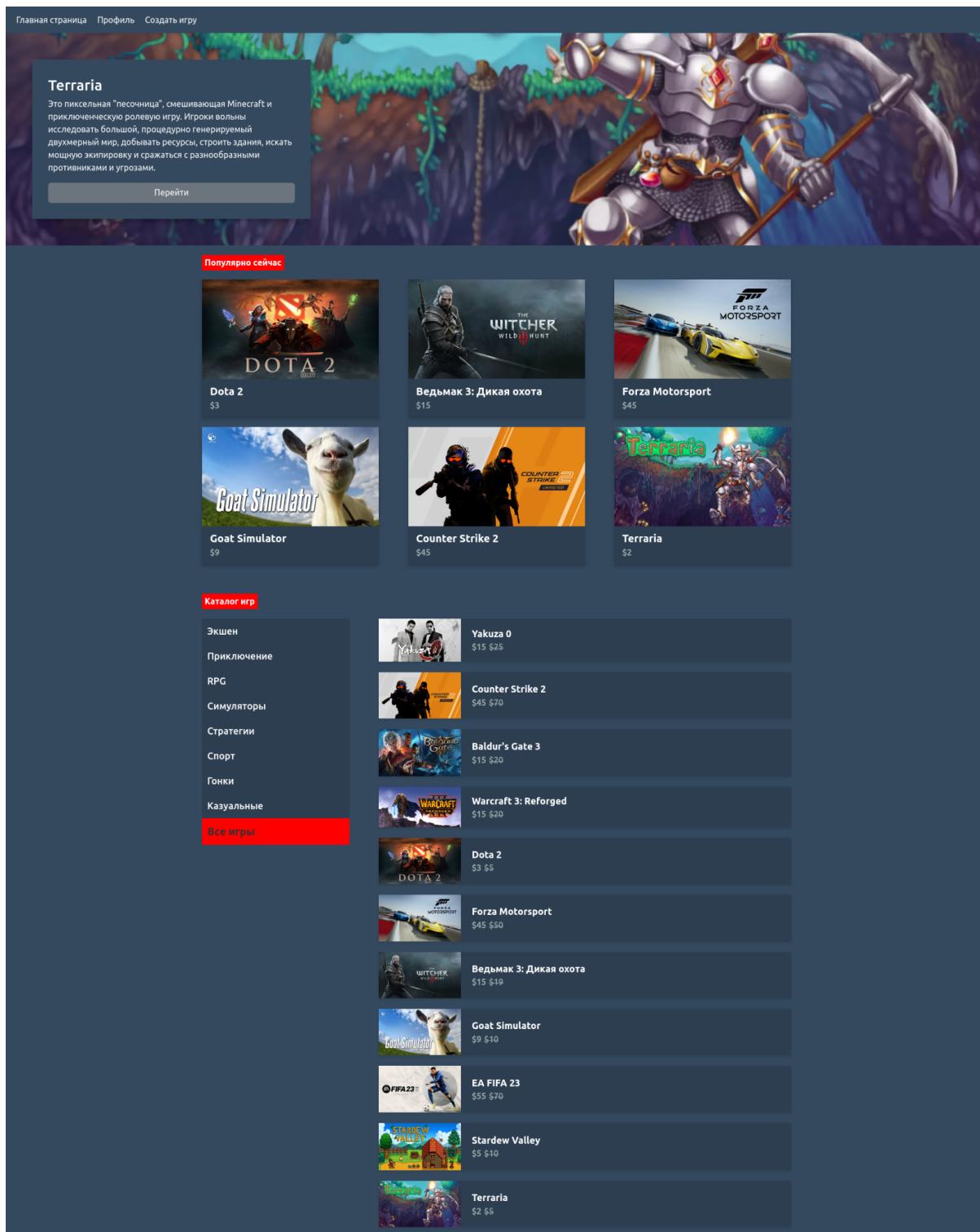


Рисунок 8 - Главная страница сайта

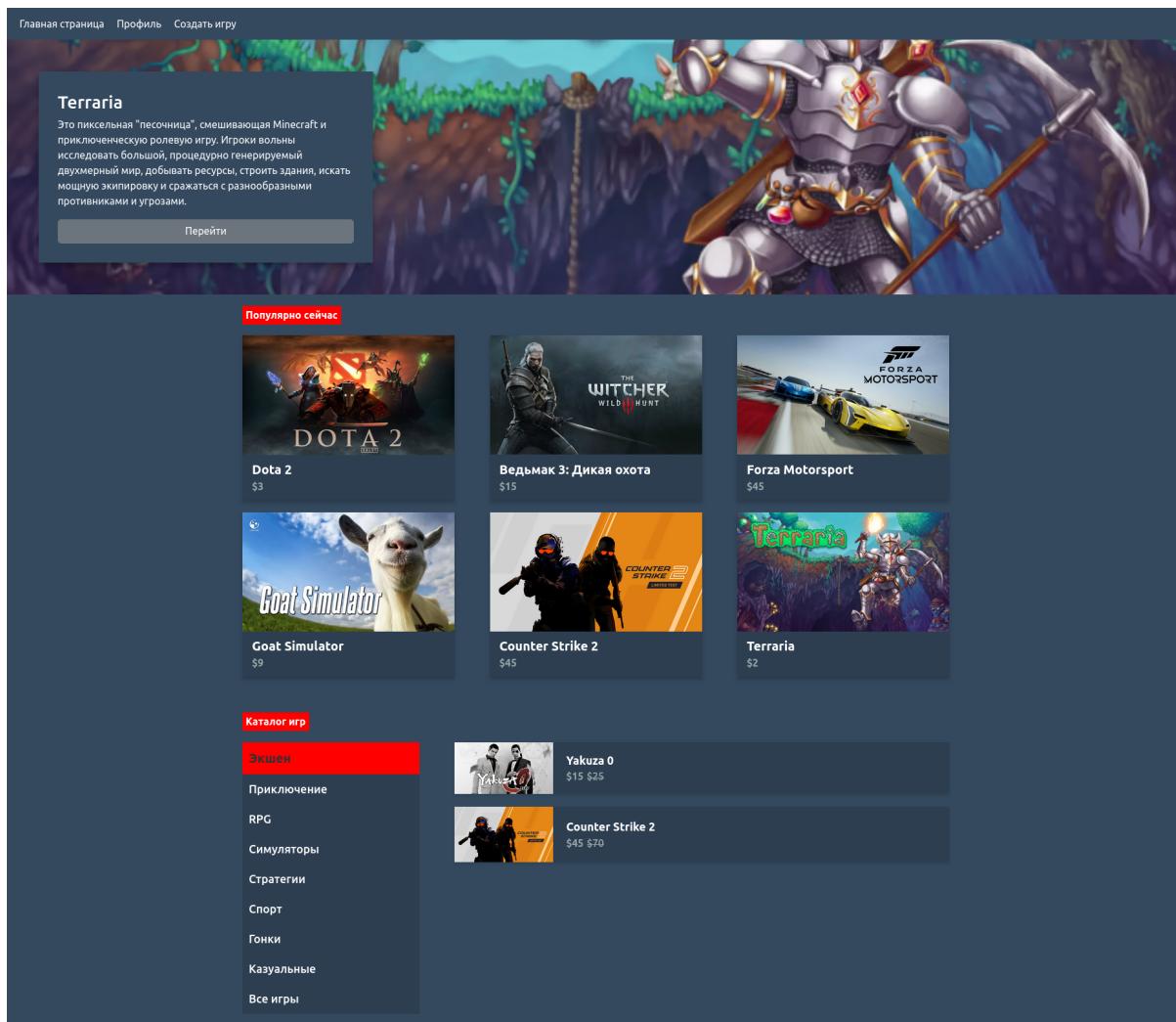


Рисунок 9 - Главная страница сайта с примененным к каталогу фильтром

Главная страница Профиль Создать игру

Ведьмак 3: Дикая охота



Описание

Ведьмак 3: Дикая охота - Третья игра одноимённой серии, а также заключительная часть трилогии, разработанная польской компанией CD Projekt RED по мотивам серии романов «Ведьмак» польского писателя Анджея Сапковского, продолжение игр Ведьмак и Ведьмак 2: Убийцы Королей.

Жанр игры: Приключение

Предложения

Steam	\$15 \$19
AGPlay	\$20 \$25

Рисунок 10 - Страница игры

Главная страница Профиль Создать игру

Выберите название для игры

Название игры

Выберите изображение для обложки игры

URL картинки

Введите описание для игры

Описание

Введите жанр игры

Жанр

Предложения

Название сайта
Название сайта

Ссылка на предложение
URL предложения

Текущая цена
Текущая цена

Стандартная цена
Стандартная цена

Добавить предложение

Добавить игру

The screenshot shows a dark-themed web application for creating a new game. At the top, there are navigation links: 'Главная страница', 'Профиль', and 'Создать игру'. Below this, the main form is titled 'Выберите название для игры' (Select a name for the game) with a placeholder 'Название игры'. The next section is 'Выберите изображение для обложки игры' (Select an image for the cover) with a placeholder 'URL картинки'. A large, empty white box is provided for uploading the image. The third section is 'Введите описание для игры' (Enter a description for the game) with a placeholder 'Описание'. The fourth section is 'Введите жанр игры' (Enter the genre of the game) with a placeholder 'Жанр'. Below these four sections is a heading 'Предложения' (Offerings). It contains several input fields for adding offers: 'Название сайта' (Site name) with a placeholder 'Название сайта', 'Ссылка на предложение' (Offer link) with a placeholder 'URL предложения', 'Текущая цена' (Current price) with a placeholder 'Текущая цена', and 'Стандартная цена' (Standard price) with a placeholder 'Стандартная цена'. There is also a button 'Добавить предложение' (Add offer). At the bottom of the form is a large green button labeled 'Добавить игру' (Add game).

Рисунок 11 - Страница добавления новой игры

Главная страница Профиль Создать игру

Выберите название для игры

Metal Gear Rising: Revengeance

Выберите изображение для обложки игры

/Images/ImageFeed/This%20is%20what%20Metal%20Gear%20Rising%3A%20Revengeance%27s%20Box%20Art%20Looks%20Like/mgsrrnaba_610.jpg

Введите описание для игры

Эта игра представляет собой спин-офф серии Metal Gear; её действие происходит спустя четыре года после событий Metal Gear Solid 4: Guns of the Patriots. Главный герой игры, вооруженный высокочастотным мечом-катаной киборг Райден, противостоит частной военной компании Desperado Enforcement LLC в разных странах мира.

Введите жанр игры

Экшен

Предложения

Steam
https://store.steampowered.com/app/235460/METAL_GEAR_RISING_REVENGEANCE/ 15\$ 20\$

Название сайта	Название сайта
Ссылка на предложение	URL предложения
Текущая цена	Текущая цена
Стандартная цена	Стандартная цена

Добавить предложение

Добавить игру

Рисунок 12 - Заполненная страница добавления новой игры

5. ВЫВОДЫ

5.1. Достигнутые результаты

В ходе работы над проектом был разработан веб-сайт маркетплейса по продаже игр. Был реализован следующий функционал:

- Регистрация и авторизация пользователей
- Просмотр информации о пользователе
- Каталог с имеющимися играми, включающий в себя 3 секции: трендовая игра, популярные игры и каталог игр с фильтром.
- Инструмент для заведения новых игр и предложений сайтов-продавцов к этим играм в разрабатываемую систему.

Для реализации данного функционала были написаны веб-приложение на ReactJS и сервер на NodeJS/ExpressJS с применением Mongoose для работы с базой данных MongoDB.

5.2. Недостатки и пути для улучшения полученного решения

Полученное решение имеет ряд недостатков.

Первое направление для улучшения проекта это оптимизация базы данных: увеличение скорости доступа к данным и оптимизация текущей структуры данных. Необходима индексация базы данных для улучшения скорости доступа к данным. Также для уменьшения занимаемого пространства необходимо пересмотреть структуру коллекций users и games.

Вторым направлением для улучшения проекта может стать оптимизация front-end составляющей: разработка мобильной версии сайта, применение более эффективных методов и библиотек для изменения и хранения состояний, более эргономичная организация UI и UX, добавление обработчика ошибок, изменение интерфейса обратной связи (например, при успешной или провальной авторизации, добавлении игры и т.д.), а также применение более совершенных технологий хранения информации об авторизованном пользователе (сейчас информация о

пользователе хранится с использованием LocalStorage браузера в незашифрованном виде, однако лучшим решением будет применение Json Web токенов).

5.3. Будущее развитие решения

Следующим этапом в развитии проекта кроме исправления недостатков и внесения улучшений, представленных в предыдущем пункте является расширение функционала, а именно:

- Добавление функционала для корзины пользователя
- Добавление функционала для связи с поддержкой
- Добавление функционала для редактирования игры

ЛИТЕРАТУРА

1. Ссылка на github проекта:

<https://github.com/moevm/nosql2h23-marketplace>

2. Документация MongoDB: <https://www.mongodb.com/docs/>

3. Документация по работе с MongoDB на NodeJS:

<https://www.mongodb.com/developer/languages/javascript/node-crud-tutorial/>

4. Документация NodeJS: <https://nodejs.org/docs/latest/api/>

5. Документация React: <https://react.dev/reference/react>

ПРИЛОЖЕНИЯ

ДОКУМЕНТАЦИЯ ПО СБОРКЕ И РАЗВЕРТЫВАНИЮ

ПРИЛОЖЕНИЯ

Скринкаст работы с приложением

Ссылка на Яндекс.Диск: <https://disk.yandex.ru/i/e47OiqEt1jYFng>

Запуск

Для установки всех необходимых зависимостей необходим NodeJS major версии ≥ 20

1. Открываем один терминал в корне проекта, выполняем следующие команды:

```
cd marketplace/frontend  
npm install  
npm start
```

Производится переход в директорию React-приложения, установка всех необходимых зависимостей и запуск приложения.

Откроется страница сайта, но без рабочего бекенда.

2. Открываем другой терминал в корне проекта, выполняем следующие команды:

```
sudo systemctl start mongod  
cd marketplace/backend  
npm install  
npm run devStart
```

Производится запуск MongoDB, переход в директорию бекенда проекта, установка всех необходимых зависимостей и запуск сервера.

3. Веб-приложение готово к эксплуатации.