МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

Кафедра математического и компьютерного моделирования

Кулахсзян Сергей Грайрович, Талмазан Анастасия Александровна

ПРОЕКТ ПО WEB-ПРОГРАММИРОВАНИЮ

КУРСОВОЙ ПРОЕКТ

Направление подготовки 02.03.01сцт Сквозные цифровые технологии,
бакалавриатская программа «Математика и компьютерные науки»
Очной (заочной) формы обучения

Студенты группы Б9123-02.03.01сцт
_____ С. Г. Кулахсзян
_____ А. А. Талмазан

Руководитель проекта
_____ И. С. Дроздова

Владивосток
2025

Содержание

1 Введение

Данный проект представляет собой веб-приложение "Магазин видеоигр разработанное с использованием серверного фреймворка Django на языке программирования Python и клиентской библиотеки React на языке программирования JavaScript.

Основная цель проекта — создание удобного онлайн-магазина, где пользователи могут просматривать, выбирать и приобретать видеоигры.

Основные задачи проекта:

- Реализовать серверную часть на Django для обработки запросов, работы с базой данных и предоставления API;
- Разработать клиентскую часть на React для создания удобного пользовательского интерфейса;
- Настроить взаимодействие фронтенда и бэкенда через REST API;
- Реализовать авторизацию, аутентификацию пользователя, а также возможность выхода из аккаунта и его удаление;
- Получить обширную информацию о видеоиграх, странах и основных валютах, соответсвующие стране, и курсе валют для формирования базы данных и дальнейшего применения этой информации;
- Добавить платёжную систему для совершения сделки купли-продажи поользователем продукта;
- Добавить возможность создания заявлений пользователем;

2 Frontend

2.1 Математическое приложение

2.1.1 Конвертирование валюты

В базе данных цены на игры хранятся в центах, это позволяет избегать хранения информации в дробных числах, из-за чего расчёты будут более точными.

У каждого пользователя цены на игры будут отображаться в той валюте, которая будет основной в стране, которую он выбрал при регистрации. Для корректного отображения необходимо конвертировать центы в корректную валюту. Это делается по формуле:

$$S_{\text{currency}} = S_{\text{USD}} \times K_{\text{currency}},$$
$$S_{\text{USD}} = \frac{S_{\text{cents}}}{100},$$

где S_{cents} – значение цены в центах, S_{USD} – значение в долларах, K_{currency} – соотношение курса избранной валюты к доллару, S_{currency} – значение цены в избранной валюте.

2.1.2 Расчёт возраста

У всех игр есть возрастной рейтинг, благодаря которому определяется минимальный возраст, допустимый для приобретения продукта, согласно законодательству. Для того, чтобы знать при каких случаях стоит предотвращать доступ к пользователя к определённому товару, необходимо рассчитать возраст пользователя.

Возраст рассчитывается в полных годах и зависит от текущей даты. Для этого используется следующая формула:

$$A = Y_{\text{today}} - Y_{\text{birth}},$$

где Y_{today} – текущий год, Y_{birth} – год рождения пользователя, A – возраст пользователя в полных годах.

Однако необходимо учитывать, что если день и месяц текущей даты ещё не достигли дня и месяца рождения пользователя в текущем году, возраст должен быть уменьшен на единицу. Условие корректировки записывается следующим образом:

$$A = A - 1, \quad \text{если } (M_{\text{today}} < M_{\text{birth}}) \text{ или } (M_{\text{today}} = M_{\text{birth}} \text{ и } D_{\text{today}} < D_{\text{birth}}),$$

где M_{today} – текущий месяц, M_{birth} – месяц рождения пользователя, D_{today} – текущее число, D_{birth} – число рождения пользователя.

2.1.3 Анимация загрузки

Часто на сайте изображения долго грузятся, дабы не было никакого пустого пространиства на сайте, было принято решение добавить анимацию загрузки, которая представляет из себя движение 8 кругов по окружности друг за другом.

Создан компонент, представляющий собой анимацию, в которой 8 кругов вращаются вокруг центра по окружности, начиная с разных позиций и плавно появляется. Рассмотрим работу этого компонента более детально.

Для создания 8 элементов (кругов) используется следующий цикл:

$$Array.from(\ length : 8).map((\_, \text{index}) \Rightarrow \ldots)$$

Этот массив из 8 элементов проходит цикл, и для каждого индекса генерируется элемент типа `div` с уникальными стилями. Основная цель заключается в том, чтобы отобразить 8 кругов, которые будут расположены равномерно по окружности.

Каждому кругу присваивается угловая трансформация, которая зависит от его индекса в массиве. Стиль для каждого круга выглядит так:

$$transform : rotate\left(\frac{360}{8} \times \text{index}\right) deg, \text{translateX}(150\%), \text{rotate}\left(-\frac{360}{8} \times \text{index}\right) deg$$

Где:

- $rotate\left(\frac{360}{8} \times \text{index}\right)$ — этот угол позволяет расположить каждый круг на равном расстоянии друг от друга по окружности (360 градусов делятся на 8 частей).
- translateX(150%) — этот параметр сдвигает круг от его исходной позиции на 150% от его радиуса, таким образом круги будут располагаться по внешней окружности.
- $rotate\left(-\frac{360}{8} \times \text{index}\right)$ — поворачивает каждый круг на противоположный угол, чтобы сохранить правильную ориентацию элементов.

Таким образом, каждый круг будет располагаться на окружности, и их движение будет синхронизировано.

Для анимации вращения каждого круга используется следующий CSS-код:

```
@keyframes orbit {
    0% {
        transform: rotate(0deg) translateX(150%)
          rotate(0deg);
    }
    100% {
    transform: rotate(360deg) translateX(150%)
      rotate(-360deg);
    }
}
```

В начале анимации (0%) каждый круг находится в своей начальной позиции. В конце анимации (100%) круг проходит полный оборот (360 градусов) вокруг центра и возвращается на своё начальное место.

Для плавного появления каждого круга используется анимация изменения прозрачности:

```
@keyframes fade-in {
    0% {
        opacity: 0;
    }
    100% {
        opacity: 1;
    }
}
```

Каждому кругу присваивается задержка анимации, которая зависит от его индекса:

```
animation: orbit 1.5s linear infinite, fade-in 0.19s
  {index} * 0.19s forwards;
```

Анимация вращения (`orbit`) длится 1.5 секунды и выполняется бесконечно. Анимация появления (`fade-in`) имеет задержку, которая зависит от индекса элемента, создавая эффект последовательного появления элементов.

2.2 Хранение пользовательских данных

Полсле регистрации или авторизации для корректной работы сервиса необходимо в каком-то виде хранить информацию о пользователе. Для этого я применяю токены.

2.2.1 Определение

JSON Web Token (JWT) представляет собой компактный и безопасный способ передачи данных между сторонами в виде JSON-объекта. Он широко используется для аутентификации и авторизации в веб-приложениях.

2.2.2 Структура JWT

JWT состоит из трёх частей, разделённых точками:

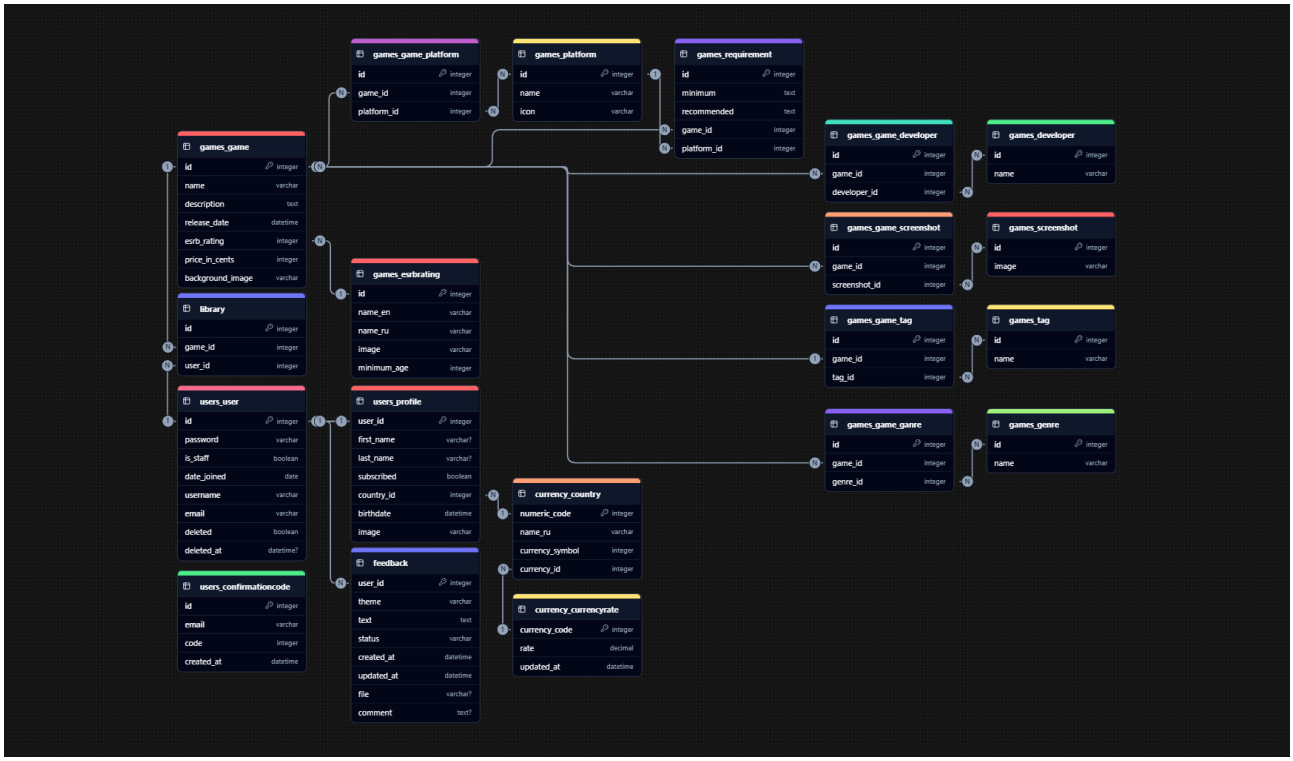
- **Header (заголовок)**
- **Payload (полезная нагрузка)**
- **Signature (подпись)**

Пример структуры JWT:
`eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiIxMjM0NTY3OD`
`kwIiwibmFtZSI6IkpvaG4gRG9lIiwicm9sZXMiOlsiYWRtaW4iLCJ1c2VyIl19.S`
`flKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c`
Каждая часть кодируется в формате Base64 URL:

- **Header**: содержит информацию о типе токена (`JWT`) и алгоритме подписи (например, `HS256`).
- **Payload**: включает данные о пользователе и дополнительные метаданные (например, роли, ID пользователя, срок действия токена).
- **Signature**: создаётся с использованием алгоритма подписи и секретного ключа для проверки подлинности токена.

3 Backend

3.1 Структура базы данных



3.2 Подключенные инструменты

3.2.1 Simple JWT

Для аутентификации пользователей нак Django мне необходим инструмент для работы с токенами. Одним из популярных инструментов для этого является пакет django-rest-framework-simple-jwt, который реализует аутентификацию с использованием JSON Web Token (JWT). Simple JWT предоставляет эндпоинты для получения и обновления токенов.

В данном проекте Simple JWT имеет следующие настройки:

```
SIMPLE_JWT = {
    'ACCESS_TOKEN_LIFETIME': timedelta(minutes=5),
    'REFRESH_TOKEN_LIFETIME': timedelta(days=50),
    'ROTATE_REFRESH_TOKENS': True,
    'BLACKLIST_AFTER_ROTATION': True,
    'UPDATE_LAST_LOGIN': False,

    'ALGORITHM': 'HS256',
    'VERIFYING_KEY': None,
    'AUDIENCE': None,
    'ISSUER': None,
```

```
    'JWK_URL': None,
    'LEEWAY': 0,

    'AUTH_HEADER_TYPES': ('Bearer',),
    'AUTH_HEADER_NAME': 'HTTP_AUTHORIZATION',
    'USER_ID_FIELD': 'id',
    'USER_ID_CLAIM': 'user_id',
    'USER_AUTHENTICATION_RULE':
      'rest_framework_simplejwt.authentication.default_
user_authentication_rule',

    'AUTH_TOKEN_CLASSES':
      ('rest_framework_simplejwt.tokens.AccessToken',),
    'TOKEN_TYPE_CLAIM': 'token_type',
    'TOKEN_USER_CLASS':
      'rest_framework_simplejwt.models.TokenUser',
    'JTI_CLAIM': 'jti',

    'SLIDING_TOKEN_REFRESH_EXP_CLAIM': 'refresh_exp',
    'SLIDING_TOKEN_LIFETIME': timedelta(minutes=5),
    'SLIDING_TOKEN_REFRESH_LIFETIME':
      timedelta(days=1),
}
```

- **Жизненный цикл токенов:**
  - **ACCESS_TOKEN_LIFETIME** – время жизни access-токена 5 минут;
  - **REFRESH_TOKEN_LIFETIME** – время жизни refresh-токена 50 дней;
  - **ROTATE_REFRESH_TOKENS** – при обновлении access-токена создаётся новый refresh-токен;
  - **BLACKLIST_AFTER_ROTATION** – старые refresh-токены попадают в чёрный список после обновления, это сделано для большей безопасности;
  - **SLIDING_TOKEN_LIFETIME** – время жизни sliding-токена 5 минут;
  - **SLIDING_TOKEN_REFRESH_LIFETIME** – время жизни sliding refresh-токена 1 день.
- **Криптографические параметры:**

- **ALGORITHM** – алгоритм шифрования токена (HMAC с SHA-256). HMAC с SHA-256 – это алгоритм для создания криптографических подписей. Он берёт данные токена, добавляет к нему секретный ключ, применяет хэш-функцию SHA-256, в итоге на выходе получается HMAC (Hash-based Message Authentication Code) — цифровая подпись, защищающая данные от подделки;
- **VERIFYING_KEY** – ключ для верификации токена;
- **JWK_URL** – URL для загрузки JSON Web Key.

- **Аутентификация и авторизация:**
  - **AUTH_HEADER_TYPES** – тип заголовка авторизации;
  - **AUTH_HEADER_NAME** – имя заголовка с токеном;
  - **USER_ID_FIELD** – поле, идентифицирующее пользователя;
  - **USER_ID_CLAIM** – claim (ключ в payload токена), в котором передаётся user_id;
  - **USER_AUTHENTICATION_RULE** – правило аутентификации пользователей;
  - **TOKEN_TYPE_CLAIM** – claim, указывающий тип токена;
  - **TOKEN_USER_CLASS** – класс пользователя, связанного с токеном;
  - **JTI_CLAIM** – уникальный идентификатор токена (используется для blacklist).

- **Дополнительные параметры:**
  - **UPDATE_LAST_LOGIN** – не обновлять поле last_login при использовании токена;
  - **LEEWAY** – дополнительное время для валидации токена (учитывает возможную разницу времени между серверами).

3.2.2 Simple Mail Transfer Protocol

SMTP (Simple Mail Transfer Protocol) — это протокол для отправки электронной почты. Данный протокол применяется в проекте для отправки электронных писем пользователям о коде подтверждения или для почтовой рассылки.

В качестве почтового хоста в проекте применяется smtp.gmail.com, также для данного проекта была создана специальная электронная почта gamestore.django.app от которого совершаются в отправки писем.

3.2.3 Celery

Celery — это асинхронный менеджер задач для Python. Он позволяет выполнять фоновые задачи (например, обработку данных, отправку email) без блокировки основного приложения. Для работы с Django применяется пакет django-celery-beat. Он создаёт новые таблицы в базе данных для создания с разными временными периодами выполнения.

Celery необходим брокер для хранении информации о задаче, в проекте в роли брокера используется Reddis. Для работы celery необходимо запустить 2 модуля: beat и worker. Celery beat просматривает информацию в специальной таблице базы данных для поиска задач, которые необходимо выполнить, после нахождения задачи, он передаёт в Reddis эту задачу. Celery worker в свою очередь выполняет все задачи, которые были переданы в Reddis.

Всего в проекте 2 задачи. Первая – send_subscription_emails. Эта задача имеет периодичность выполнения раз в день. Она находит всех пользователей, которые имеют статус "подписан после чего отправляет всем найденным пользователям почтовую россылку. Вторая – delete_old_users. Данная задача каждый день ищет пользователей со статусом "удалённый"и проверяет дату приобретения этого статуса, пользователь получил статус "удалённый"более 30 дней назад, то информация о пользователе стирается из базы данных.

3.2.4 Stripe API

Stripe API – это мощный API для работы с платежами, подписками и финансами в веб-приложениях. В проекте он применяется, как со стороны фронтэнда, так и бекэнда.

В клиентской части создаются специальные компоненты для ввода данных о банковской карте пользователя. После эти данные специальным образом конвертируются для безопасности сохранности данных.

Таким образом, в серверную часть уже передаются данные не самой карты, а некоторая информация, которую Stripe расшифровывает для последующей работы с ним. После совершается транзакция, а приобретённый продукт передаётся пользователю.

3.3 Роуты для обращения к API

Серверная часть сайта разделён на 6 приложений, каждая из которых выполняет свою задачу: currency, feedback, games, library, payments, users.

3.3.1 currency

GET: /currency/currency_rates/

Описание: Возвращает все объекты модели CurrencyRate.

Тело ответа:

```
[
    {
        "currency_code": "USD",
        "rate": "1.000000",
        "updated_at": "2024-12-02T12:55:54.013000Z"
    },
    ...
]
```

GET: /currency/currency_rates/get/{currency_code}/

Описание: Возвращает определённый объект модели CurrencyRate в соответсвии с currency_code.

Тело ответа:

```
{
        "currency_code": "USD",
        "rate": "1.000000",
        "updated_at": "2024-12-02T12:55:54.013000Z"
}
```

GET: /currency/country/

Описание: Возвращает все объекты модели Country.

Тело ответа:

```
[
    {
            "numeric_code": 520,
            "name_ru": "Наруру",
            "currency_symbol": "$",
            "currency": "AUD"
    },
    ...
]
```

GET: /currency/country/get/{numeric_code}/

Описание: Возвращает определённый объект модели Country в соответсвии с numeric_code.

Тело ответа:

```
{
        "numeric_code": 520,
        "name_ru": "Наypy",
        "currency_symbol": "$",
        "currency": "AUD"
}
```

3.3.2 feedback

GET: /feedback/feedback/

Описание: Возвращает все объекты модели Feedback, доступно только администраторам.

Тело ответа:

```
[
    {
        "id": 29,
        "theme": "Some theme",
        "text": "Some text",
        "file":
          "http://127.0.0.1:8000/static/files/file.
file",
        "status": "Some status",
        "created_at":
          "2025-01-26T10:52:30.333019Z",
        "updated_at":
          "2025-01-26T16:04:07.857400Z",
        "user": 1,
        "comment": "Some comment"
    },
    ...
]
```

GET: /feedback/feedback/get/{user_id}/

Описание: Возвращает все объекты модели Feedback отправленные определённым пользователем в соответсвии с указанным ID пользователя user_id.

Тело ответа:

```
[
    {
        "id": 29,
        "theme": "Some theme",
        "text": "Some text",
        "file":
          "http://127.0.0.1:8000/static/files/file.
file",
        "status": "Some status",
        "created_at":
          "2025-01-26T10:52:30.333019Z",
        "updated_at":
          "2025-01-26T16:04:07.857400Z",
        "user": 1,
        "comment": "Some comment"
    },
    ...
]
```

GET: /feedback/feedback/get_by_id/{feedback_id}/

Описание: Возвращает определённый объект модели Feedback в соответсвии с feedback_id.

Тело ответа:

```
{
    "id": 29,
    "theme": "Some theme",
    "text": "Some text",
    "file":
      "http://127.0.0.1:8000/static/files/file.file",
    "status": "Some status",
    "created_at": "2025-01-26T10:52:30.333019Z",
    "updated_at": "2025-01-26T16:04:07.857400Z",
    "user": 1,
    "comment": "Some comment"
}
```

POST: /feedback/create/{user_id}/

Описание: Создаёт новый объект модели Feedback от пользователя в соответсвии с user_id.

Тело запроса:

```
{
    "theme": "Some theme",
    "text": "Some text",
    "file": "file.file"
}
```

Тело ответа в случае успеха:

```
{
    "message": "Feedback created"
}
```

Тело ответа в случае провала:

```
{
    "details": "Some error"
}
```

PUT: /feedback/feedback/update/{feedback_id}/

Описание: Обновляет данные объекта модели Feedback в соответствии с feedback_id.

Тело запроса:

```
{
    "theme": "Some theme",
    "text": "Some text",
    "file": "file.file"
}
```

Тело ответа в случае успеха:

```
{
    "message": "Feedback updated"
}
```

Тело ответа в случае провала:

```
{
    "details": "Some error"
}
```

DELETE: /feedback/feedback/delete/{feedback_id}/

Описание: Удаляет данные объекта модели Feedback в соответствии с feedback_id.

Тело ответа в случае успеха:

```
{
    "message": "Feedback deleted"
}
```

Тело ответа в случае провала из-за статуса, не равному "Отправлено":

```
{
    "message": "Feedback can't be deleted"
}
```

Тело ответа в случае других причин провала:

```
{
    "details": "Some error"
}
```

3.3.3 games

GET: /games/platform/

Описание: Возвращает все объекты модели Platform.

Тело ответа:

```
[
    {
        "id": 4,
        "name": "Windows",
        "icon":
          "http://127.0.0.1:8000/static/icons/windo
ws-icon.png"
    },
    ...
]
```

GET: /games/platform/get/{platform_id}/

Описание: Возвращает определённый объект модели Platform в соответствии с platform_id.

Тело ответа:

```
{
    "id": 4,
    "name": "Windows",
    "icon":
      "http://127.0.0.1:8000/static/icons/windows-i
con.png"
}
```

GET: /games/esrb_rating/

Описание: Возвращает все объекты модели ESRBRating.

Тело ответа:

```
[
    {
        "id": 1,
        "name_en": "Everyone 10 and older",
        "name_ru": "С 10 лет",
        "image":
          "http://127.0.0.1:8000/static/icons/ESRB-
Everyone-10.png",
        "minimum_age": 10
    },
    ...
]
```

GET: /games/esrb_rating/get/{esrb_rating_id}/

Описание: Возвращает определённый объект модели ESRBRating в соответствии с esrb_rating_id.

Тело ответа:

```
{
    "id": 1,
    "name_en": "Everyone 10 and older",
    "name_ru": "С 10 лет",
    "image":
      "http://127.0.0.1:8000/static/icons/ESRB-Every
one-10.png",
    "minimum_age": 10
}
```

GET: /games/genre/

Описание: Возвращает все объекты модели Genre.

Тело ответа:

```
[
    {
        "id": 1,
        "name": "Racing"
    },
    ...
]
```

GET: /games/genre/get/{genre_id}/

Описание: Возвращает определённый объект модели Genre в соответствии с genre_id.

Тело ответа:

```
{
    "id": 1,
    "name": "Racing"
}
```

GET: /games/tag/

Описание: Возвращает все объекты модели Tag.

Тело ответа:

```
[
    {
        "id": 1,
        "name": "Survival"
    },
    ...
]
```

GET: /games/tag/get/{tag_id}/

Описание: Возвращает определённый объект модели Tag в соответствии с tag_id.

Тело ответа:

```
{
    "id": 1,
    "name": "Survival"
}
```

GET: /games/developer/

Описание: Возвращает все объекты модели Developer.

Тело ответа:

```
[
    {
        "id": 10,
        "name": "Rockstar Games"
    },
    ...
]
```

GET: /games/developer/get/{developer_id}/

Описание: Возвращает определённый объект модели Developer в соответствии с developer_id.

Тело ответа:

```
{
    "id": 10,
    "name": "Rockstar Games"
}
```

GET: /games/screenshot/

Описание: Возвращает все объекты модели Screenshot.

Тело ответа:

```
[
    {
        "id": 1827221,
        "image":
          "https://media.rawg.io/media/screenshots/
a7c/a7c43871a54bed6573a6a429451564ef.jpg"
    },
    ...
]
```

GET: /games/screenshot/get/{screenshot_id}/

Описание: Возвращает определённый объект модели Screenshot в соответствии с screenshot_id.

Тело ответа:

```
{
    "id": 1827221,
    "image":
      "https://media.rawg.io/media/screenshots/a7c/
a7c43871a54bed6573a6a429451564ef.jpg"
}
```

GET: /games/game/?page={page_number}

Описание: Возвращает все объекты модели Game с пагинацией на странице page_number, на каждой странице 39 объектов.

Тело ответа:

```
{
    "total_count": 9999,
    "total_pages": 257,
    "page": 2,
    "next":
      "http://127.0.0.1:8000/games/game/?page=3",
    "previous":
      "http://127.0.0.1:8000/games/game/",
    "results": [
    {
        "id": 19710,
        "name": "Half-Life 2: Episode One",
        "background_image":
          "https://media.rawg.io/media/games/7a2/7a
2500ee8b2c0e1ff268bb4479463dea.jpg",
        "description": "<p>Description...</p>",
        "esrb_rating": 6,
        "release_date": "2006-06-01",
        "price_in_cents": 0,
        "platforms": [4, 5, 6],
        "genres": [2, 4],
        "tags": [8, 13, 30, 31, 32, 42, 62, 63,
          110, 111, 118, 119, 172, 193, 232, 319,
          11669, 40833, 40834, 40839, 40845,
          40847, 40849],
        "screenshots": [185831, 185832, 185833,
          185834, 185835],
        "developers": [1612, 23342]
    },
    ...
}
```

GET: `/games/games/get/{user_id}/?page={page_number}`

Описание: Возвращает определённый объект модели Game, которые отсутствуют в библиотеке у пользователя с ID user_id, с пагинацией на странице page_number, на каждой странице 39 объектов.

Необязательный параметры:

- `[&platforms={platform_id1}[&platforms={platform_id2}[...]]]` – параметры для фильтрации игр по платформам, возвращает только те игры, где в качесте платформы указан хотя бы один из перечисленных `platform_id1[, platform_id2[, ...]]`;
- `[&genres={genre_id1}[&genres={genre_id2}[...]]]` – параметры для фильтрации игр по жанрам, возвращает только те игры, где в качесте жанра указан хотя бы один из перечисленных `genre_id1[, genre_id2[, ...]]`;
- `[&tags={tag_id1}[&tags={tag_id2}[...]]]` – параметры для фильтрации игр по тэгам, возвращает только те игры, где в качесте тэга указан хотя бы один из перечисленных `tag_id1[, tag_id2[, ...]]`;
- `[&developers={developer_id1}[&developers={developer_id2}[...]]]` – параметры для фильтрации игр по разработчикам, возвращает только те игры, где в качесте разработчиков указан хотя бы один из перечисленных `developer_id1[, developer_id2[, ...]]`;
- `&name={game_title}` – параметр для фильтрации игр по названию, возвращает только те игры, где название включает в себя game_title.

Тело ответа:

```
{
    "total_count": 9999,
    "total_pages": 257,
    "page": 2,
    "next":
      "http://127.0.0.1:8000/games/game/?page=3",
    "previous":
      "http://127.0.0.1:8000/games/game/",
    "results": [
    {...
```

GET: /games/games/get/{user_id}/?page={page_number}

Продолжение...

```
...
    {
        "id": 19710,
        "name": "Half-Life 2: Episode One",
        "background_image":
          "https://media.rawg.io/media/games/7a2/7a
2500ee8b2c0e1ff268bb4479463dea.jpg",
        "description": "<p>Description...</p>",
        "esrb_rating": 6,
        "release_date": "2006-06-01",
        "price_in_cents": 0,
        "platforms": [4, 5, 6],
        "genres": [2, 4],
        "tags": [8, 13, 30, 31, 32, 42, 62, 63,
          110, 111, 118, 119, 172, 193, 232, 319,
          11669, 40833, 40834, 40839, 40845,
          40847, 40849],
        "screenshots": [185831, 185832, 185833,
          185834, 185835],
        "developers": [1612, 23342]
    },
    ...
}
```

GET: /games/game/get/{game_id}

Описание: Возвращает определённый объект модели Game в соответствии с game_id.

Тело ответа:

```
{
    "id": 19710,
    "name": "Half-Life 2: Episode One",
    "background_image":
      "https://media.rawg.io/media/games/7a2/7a2500
ee8b2c0e1ff268bb4479463dea.jpg",
    "description": "<p>Description...</p>",
    "esrb_rating": 6,
    "release_date": "2006-06-01",
    "price_in_cents": 0,
    "platforms": [4, 5, 6],
    "genres": [2, 4],
    "tags": [8, 13, 30, 31, 32, 42, 62, 63, 110,
      111, 118, 119, 172, 193, 232, 319, 11669,
      40833, 40834, 40839, 40845, 40847, 40849],
    "screenshots": [185831, 185832, 185833,
      185834, 185835],
    "developers": [1612, 23342]
}
```

GET: /games/random_games/{user_id}

Описание: Возвращает 6 случайных объекта модели Game, которые отстутствуют в библиотеке пользователься с ID user_id.

Тело ответа:

```
[
    {
        "id": 19710,
        "name": "Half-Life 2: Episode One",
        "background_image":
          "https://media.rawg.io/media/games/7a2/7a
2500ee8b2c0e1ff268bb4479463dea.jpg",
        "description": "<p>Description...</p>",
        "esrb_rating": 6,
        "release_date": "2006-06-01",
        "price_in_cents": 0,
        "platforms": [4, 5, 6],
        "genres": [2, 4],
        "tags": [8, 13, 30, 31, 32, 42, 62, 63,
          110, 111, 118, 119, 172, 193, 232, 319,
          11669, 40833, 40834, 40839, 40845,
          40847, 40849],
        "screenshots": [185831, 185832, 185833,
          185834, 185835],
        "developers": [1612, 23342]
    },
    ...
]
```

GET: /games/check_game_in_library/{user_id}/{game_id}

Описание: Возвращает true или false в зависимости от того, есть ли у пользователя с ID user_id в библиотеке игра с ID game_id.

Тело ответа в случае, если есть:

```
{
    "in_library": true
}
```

Тело ответа в случае, если нет:

```
{
    "in_library": false
}
```

GET: /games/requirement/

Описание: Возвращает все объекты модели Requirement.

Тело ответа:

```
[
    {
        "id": 1,
        "minimum": "Minimum:OS: Windows 10 64 Bit, Windows 8.1 64 Bit, Windows 8 64...",
        "recommended": "Recommended:OS: Windows 10 64 Bit, Windows 8.1 64 Bit, Windows...",
        "platform": 4,
        "game": 3498
    },
    ...
]
```

GET: /games/requirement/get/{game_id}/

Описание: Возвращает все объекты модели Requirement игры с ID game_id.

Тело ответа:

```
[
    {
        "id": 1,
        "minimum": "Minimum:OS: Windows 10 64 Bit, Windows 8.1 64 Bit, Windows 8 64...",
        "recommended": "Recommended:OS: Windows 10 64 Bit, Windows 8.1 64 Bit, Windows...",
        "platform": 4,
        "game": 3498
    },
    ...
]
```

3.3.4 library

GET: /library/library/

Описание: Возвращает все объекты модели Library.

Тело ответа:

```
[
    {
        "user": 1,
        "game": 50677
    },
    ...
]
```

GET: /library/library/get/{user_id}/

Описание: Возвращает все объекты модели Library пользователя с ID user_id.

Тело ответа:

```
[
    {
        "user": 1,
        "game": 50677
    },
    ...
]
```

POST: /library/library/add/{user_id}/{game_id}/

Описание: Создаёт новый объект модели Library с игрой ID game_id для пользователя с ID user_id.

Тело ответа в случае успеха:

```
{
    "message": "Game added to library"
}
```

Тело ответа в случае провала из-за наличия игры у пользователя:

```
{
    "error": "This game is already in the library"
}
```

3.3.5 payments

POST: /payments/save_stripe_info/

Описание: Совершает запрос в приложение stripe для совершения транзакции.

Тело запроса:

```
{
    "email": "some_email@mail.com",
    "payment_method_id": 999,
    "price": 999
}
```

Тело ответа в случае, если пользователь с указанной электронной почтной впервые совершает транзакцию:

```
{
    "message": "Success",
    "data": {
        "customer_id": 999,
        "extra_msg": ""
    }
}
```

Тело ответа в случае, если пользователь с указанной электронной почтной уже совершал транзакцию:

```
{
    "message": "Success",
    "data": {
        "customer_id": 999,
        "extra_msg": "Customer already existed."
    }
}
```

3.3.6 users

POST: /users/token/

Описание: Возвращает JSON Web Token с информацией о зарегистрированном пользователе.

Тело запроса:

```
{
    "email": "examole@mail.com"
    "password": "example_password"
}
```

Тело ответа в случае успеха:

```
{
    "refresh":
      "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "access":
      "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

Тело ответа в случае провала из-за неправильной почты или неправильного пароля:

```
{
    "detail": "No active account found with the
      given credentials"
}
```

POST: /users/token/refresh/

Описание: Обновляет JSON Web Token пользователя, это необходимо из-за ограниченного срока действия токена.

Тело запроса:

```
{
    "refresh":
      "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

Тело ответа в случае успеха:

```
{
    "refresh":
      "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "access":
      "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

Тело ответа в случае провала из-за некорректного refresh токена:

```
{
    "detail": "Token is invalid or expired",
    "code": "token_not_valid"
}
```

Тело ответа в случае провала из-за заблокированного refresh токена:

```
{
    "detail": "Token is blacklisted",
    "code": "token_not_valid"
}
```

POST: /users/register/

Описание: Создаёт новые объекты User и соответствующий Profile.

Тело запроса:

```
{
    "username": "user",
    "email": "exaple@mail.com",
    "password": "example_password",
    "password2": "example_password",
    "first_name": "Name",
    "last_name": "Surname",
    "birthdate": 30.12.2000,
    "country": 643
}
```

Тело ответа в случае успеха:

```
{
    "username": "user",
    "email": "exaple@mail.com"
}
```

Тело ответа в случае провала из-за занятых электронной почты и/или имени пользователя:

```
{
    "username": [
        "user with this username already exists."
    ],
    "email": [
        "user with this email already exists."
    ]
}
```

Тело ответа в случае провала из-за пустого значения одного из полей:

```
{
    "username": [
        "This field may not be blank."
    ],
    ...
}
```

POST: /users/check_password/{user_id}/

Описание: Проверяет правильность пароля, соответсвующий пользователю с ID user_id.

Тело запроса:

```
{
    "password": "example_password"
}
```

Тело ответа в случае успеха:

```
{
    "details": "Password is valid"
}
```

Тело ответа в случае провала из-за невалидного пароля:

```
{
    "details": "Password is invalid"
}
```

Тело ответа в случае провала из-за отсутсвия пользователя с ID user_id:

```
{
    "details": "User not found"
}
```

POST: /users/check_email/

Описание: Проверяет, что переданная электронная почта не занята другим пользователем.

Тело запроса:

```
{
    "email": "example@mail.com"
}
```

Тело ответа в случае успеха:

```
{
    "details": "Email does not exist"
}
```

Тело ответа в случае провала из-за существования пользователя с данной электронной почтой:

```
{
    "details": "Email already exists"
}
```

Тело ответа в случае провала из-за существования пользователя с данной электронной почтой, имеющий статус "удалённый":

```
{
    "details": "User is deleted"
}
```

POST: /users/check_username/

Описание: Проверяет, что переданное имя пользователя не занята другим пользователем.

Тело запроса:

```
{
    "username": "username"
}
```

Тело ответа в случае успеха:

```
{
    "details": "Username does not exist"
}
```

Тело ответа в случае провала из-за существования пользователя с данным именем:

```
{
    "details": "Username already exists"
}
```

Тело ответа в случае провала из-за существования пользователя с данным именем, имеющий статус "удалённый":

```
{
    "details": "User is deleted"
}
```

PUT: /users/update/{user_id}/

Описание: Обновляет данные пользователя, кроме электронной почты и пароля.

Тело запроса:

```
{
    "username": "username",
    "password": "password",
    "first_name": "Name",
    "last_name": "Surname",
    "birthdate": 30.12.2000,
    "country": 643,
    "image": "file.file"
}
```

Тело ответа в случае успеха:

```
{
    "details": "Data updated"
}
```

Тело ответа в случае провала из-за неверного пароля:

```
{
    "details": "Invalid password"
}
```

Тело ответа в случае провала из-за существования иного пользователя с указанным именем:

```
{
    "details": "User with this username already
      exists"
}
```

PUT: /users/subscribe/{user_id}/

Описание: Изменяет статус подписки пользователя с ID user_id на противоположный.

Тело ответа в случае подписки:

```
{
    "details": "User subscribed true"
}
```

Тело ответа в случае отписки:

```
{
    "details": "User subscribed false"
}
```

DELETE: /users/delete/{user_id}/

Описание: Делает статус пользователя с ID user_id удалённым.

Тело ответа в случае успеха:

```
{
    "detail": "User deleted"
}
```

Тело ответа в случае провала из-за отсутсвия пользователя с указанным user_id:

```
{
    "detail": "User not found."
}
```

Тело ответа в случае провала из-за запроса от иного пользователя:

```
{
    "detail": "You can't delete stranger account."
}
```

PUT: /users/recover/

Описание: Убирает статус "удалённый"у пользователя.

Тело запроса:

```
{
    "recover": "email or username"
}
```

Тело ответа в случае успеха:

```
{
    "User recovered"
}
```

Тело ответа в случае провала из-за отсутствия пользователя с таким данным:

```
{
    "details": "User not found"
}
```

Тело ответа в случае провала из-за иных причин:

```
{
    "details": "error: Some error"
}
```

POST: /users/create_confirmation_code/

Описание: Создаёт или обновляет объект модели ConfirmationCode, соответствующий указанной электронной почте, и отправляет эллектронное письмо на эту почту с кодом подтверждения.

Тело запроса:

```
{
    "email": "example@mail.com"
}
```

Тело ответа в случае успеха:

```
{
    "details": "Reset password code sent"
}
```

Тело ответа в случае провала:

```
{
    "details": "error: Some error"
}
```

POST: /users/check_confirmation_code/

Описание: Проверяет код подтверждения, соответствующий электронной почте.

Тело запроса:

```
{
    "email": "example@mail.com",
    "code": "1234"
}
```

Тело ответа в случае успеха:

```
{
    "details": "Confirmation code is valid"
}
```

Тело ответа в случае провала из-за невалидного кода:

```
{
    "details": "Confirmation code is invalid"
}
```

Тело ответа в случае провала из-за отсутствия кода подтверждения для указанной электронной почты:

```
{
    "details": "Confirmation code not found"
}
```

Тело ответа в случае провала из-за иных причин:

```
{
    "details": "error: Some error"
}
```

PUT: /users/reset_password/

Описание: Обновляет пароль пользователя.

Тело запроса:

```
{
    "email": "exaple@mail.com",
    "password": "new_password"
}
```

Тело ответа в случае усаеха:

```
{
    "details": "Password changed"
}
```

Тело ответа в случае провала:

```
{
    "details": "error: Some error"
}
```

PUT: /users/change_email/{user_id}/

Описание: Обновляет электронную почту пользователя в соответсвии с user_id.

Тело запроса:

```
{
    "email": "example@mail.com"
}
```

Тело ответа в случае успеха:

```
{
    "details": "Email changed"
}
```

тело ответа в случае провала:

```
{
    "details": f"error: Some error"
}
```