МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное   
образовательное учреждение высшего образования  
«Самарский национальный исследовательский университет   
имени академика С.П. Королева»

(Самарский университет)

Институт информатики и кибернетики  
Кафедра технической кибернетики

ЛАБОРАТОРНАЯ РАБОТА № 2

**Разработка API**

по курсу   
Технологии сетевого программирования

Группа 6302-010302D

Студент О.В. Удалова

(*подпись*)

Самара 2025

## Структура API

API состоит из трех основных категорий:

* **Пользователи (/api/users)** – управление пользователями и их данными.
* **Карточки (/api/cards)** – создание и обмен карточками.
* **Достижения (/api/achievements)** – получение и назначение достижений пользователям.

## Описание основных методов

**2.1. Методы для пользователей (/api/users)**

* GET /api/users – Получить всех пользователей.
* POST /api/users – Создать нового пользователя.
* GET /api/users/{id} – Получить пользователя по ID.
* GET /api/users/search?name={name} – Найти пользователя по никнейму.
* PATCH /api/users/{id}/update-pic?newPic={url} – Обновить аватар пользователя.
* DELETE /api/users/{id}/delete-pic – Удалить аватар пользователя.
* GET /api/users/{id}/cards – Получить карточки пользователя.
* GET /api/users/{id}/achievements – Получить достижения пользователя.
* POST /api/users/{id}/achievements/{achievementId} – Назначить достижение пользователю.

**2.2. Методы для карточек (/api/cards)**

* GET /api/cards – Получить все карточки.
* POST /api/cards – Создать новую карточку.
* GET /api/cards/search?nameQuery={name} – Найти карточку по названию.
* PATCH /api/cards/{cardId}/transfer?currentOwnerId={id}&newOwnerId={id} – Передать карточку другому пользователю.

**2.3. Методы для достижений (/api/achievements)**

* GET /api/achievements – Получить список всех достижений.
* GET /api/achievements/{id} – Получить информацию о конкретном достижении.

## Примеры запросов и ответов

**3.1. Создание пользователя**

**Запрос:**

POST /api/users

Content-Type: application/json

{

"userName": "user ",

"email": "user@mail.com",

"password": "12345"

}

**Ответ:**

{

"id": 12,

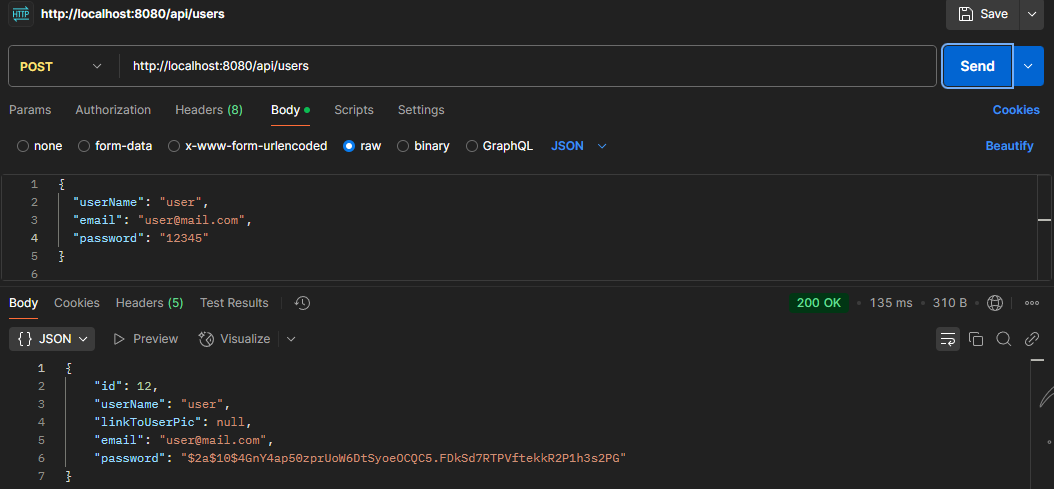
"userName": "user",

"linkToUserPic": null,

"email": "user@mail.com",

"password": "$2a$10$4GnY4ap50zprUoW6DtSyoeOCQC5.FDkSd7RTPVftekkR2P1h3s2PG"

}



**3.2. Передача карточки**

**Запрос:**

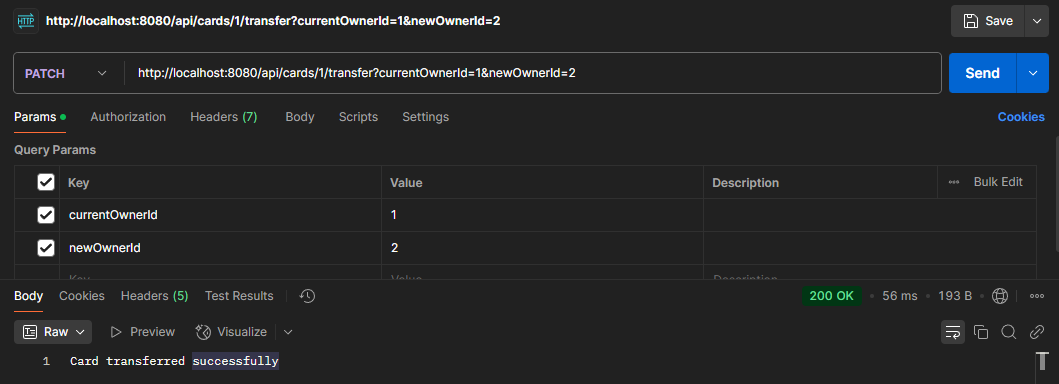
PATCH /api/cards/1/transfer?currentOwnerId=1&newOwnerId=2

**Ответ:**

{

"message": "Card transferred successfully"

}



## Проделанная работа

1. Были написаны класс UserController.java, SwapCardController.java, AchievementController.java, в которых были настроены маршруты и обработки запросов.
2. Был установлен и настроен Postman для проверки работоспособности всех маршрутов.
   1. Каждый метод API был протестирован вручную с использованием реальных запросов.
   2. Дополнительно были созданы тесты внутри Postman с помощью встроенного ИИ для автоматической проверки корректности работы API.
   3. В папке "screenshots" были сохранены скриншоты всех тестов для документирования процесса.
3. Для автоматической генерации документации были добавлены аннотации @Tag, @Operation, @ApiResponses и @Parameter к каждому методу и классу в Controller, чтобы Swagger (OpenAPI) мог корректно сформировать спецификацию API.
4. После генерации OpenAPI-спецификации по адресу http://localhost:8080/v3/api-docs, JSON-файл был скачан и преобразован в Markdown-документацию (api-documentation.md) через ИИ.
5. В ходе тестирования API были выявлены и исправлены ошибки, такие как некорректная обработка несуществующих пользователей и карточек, а также отсутствие обработки ошибок для некоторых методов.