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

**Итоговый отчет**

по курсу   
Технологии сетевого программирования

Группа 6302-010302D

Студент О.В. Удалова

(*подпись*)

Самара 2025

## Описание приложения

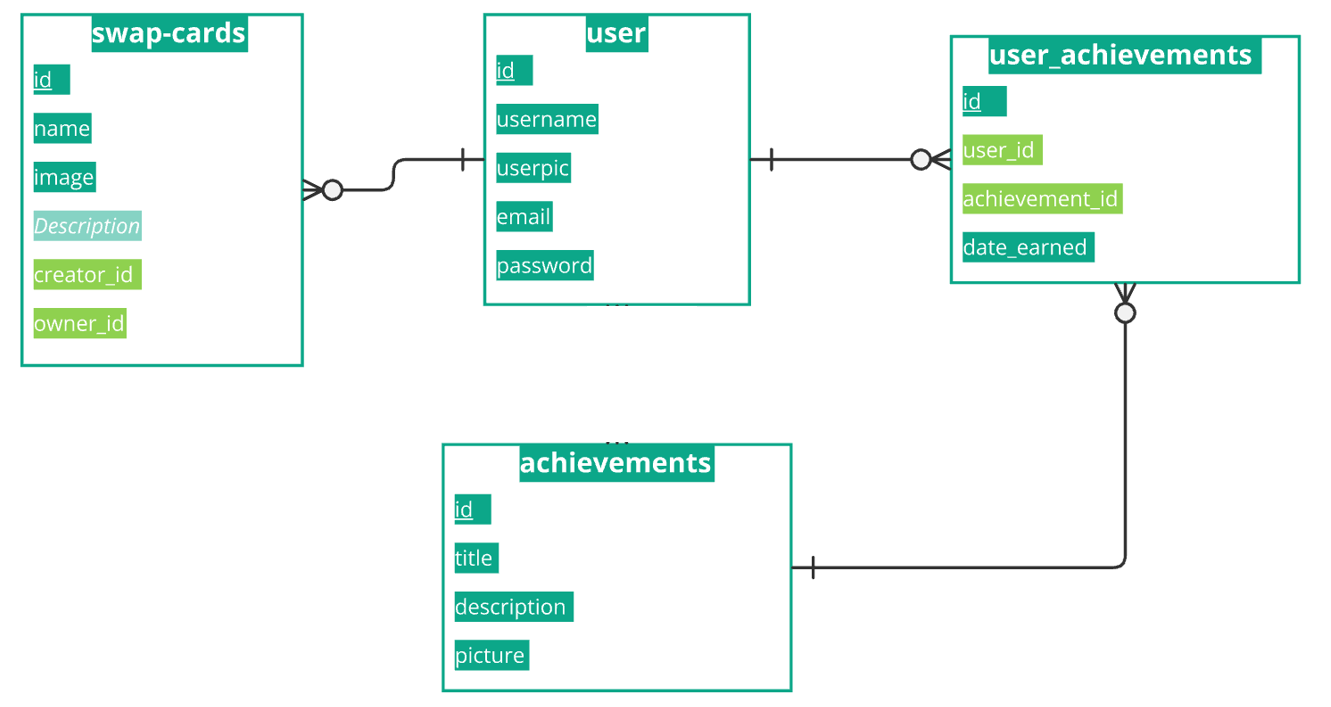
**Swapka** — это веб-приложение, которое переносит недавно появившееся хобби у российских художников по обмену авторских карточек в онлайн-пространство. Приложение предоставляет платформу для создания и обмена авторскими коллекционными карточками. Каждый пользователь может создать свою уникальную карточку, собрать коллекцию из карточек других авторов, а также получить достижения за активность на платформе.

Приложение является многопользовательским, все пользователи обладают равными правами и могут свободно взаимодействовать друг с другом через обмен карточками.

## Архитектура разработанного приложения

**Архитектура:**

* + **Backend:** Java 17, Spring Boot, Spring Security, JWT, JPA (Hibernate)
  + **Frontend:** HTML, CSS, Thymeleaf
  + **База данных:** PostgreSQL 13.3
  + **Документация:** OpenAPI/Swagger
  + **Контейнеризация:** Docker

1. **Схема базы данных**

Сущности и их связи:

1. **users** — хранит информацию о пользователях.
   * user\_id (INTEGER, PRIMARY KEY, AUTO\_INCREMENT) — уникальный идентификатор пользователя.
   * username (VARCHAR, NOT NULL) — имя пользователя.
   * userpic (VARCHAR) — ссылка на изображение профиля.
   * email (VARCHAR, UNIQUE, NOT NULL) — адрес электронной почты.
   * password (VARCHAR, NOT NULL) — хешированный пароль.
2. **swap\_cards** — содержит информацию о картах, созданных пользователями.
   * card\_id (INTEGER, PRIMARY KEY, AUTO\_INCREMENT) — уникальный идентификатор карты.
   * name (VARCHAR, NOT NULL) — название карты.
   * image (VARCHAR, NOT NULL) — ссылка на изображение карты.
   * description (TEXT) — описание карты.
   * creator\_id (INTEGER, FOREIGN KEY -> users.user\_id, NOT NULL) — идентификатор создателя карты.
   * owner\_id (INTEGER, FOREIGN KEY -> users.user\_id, NOT NULL) — идентификатор текущего владельца карты.
3. **achievements** — содержит информацию о достижениях.
   * achievement\_id (INTEGER, PRIMARY KEY, AUTO\_INCREMENT) — уникальный идентификатор достижения.
   * title (VARCHAR, NOT NULL) — название достижения.
   * icon (VARCHAR, NOT NULL) — ссылка на изображение.
   * description (TEXT, NOT NULL) — описание достижения.
   * condition (VARCHAR, NOT NULL) — условие получения достижения.
4. **user\_achievements** — таблица для связывания пользователей и их достижений.
   * user\_id (INTEGER, FOREIGN KEY -> users.user\_id, PRIMARY KEY) — идентификатор пользователя.
   * achievement\_id (INTEGER, FOREIGN KEY -> achievements.achievement\_id, PRIMARY KEY) — идентификатор достижения.
   * date\_earned (DATE, NOT NULL) — дата получения достижения.

## 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.
* **Авторизация (/api/auth)** – регистрация, вход, смена пароля, выход.

## 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.

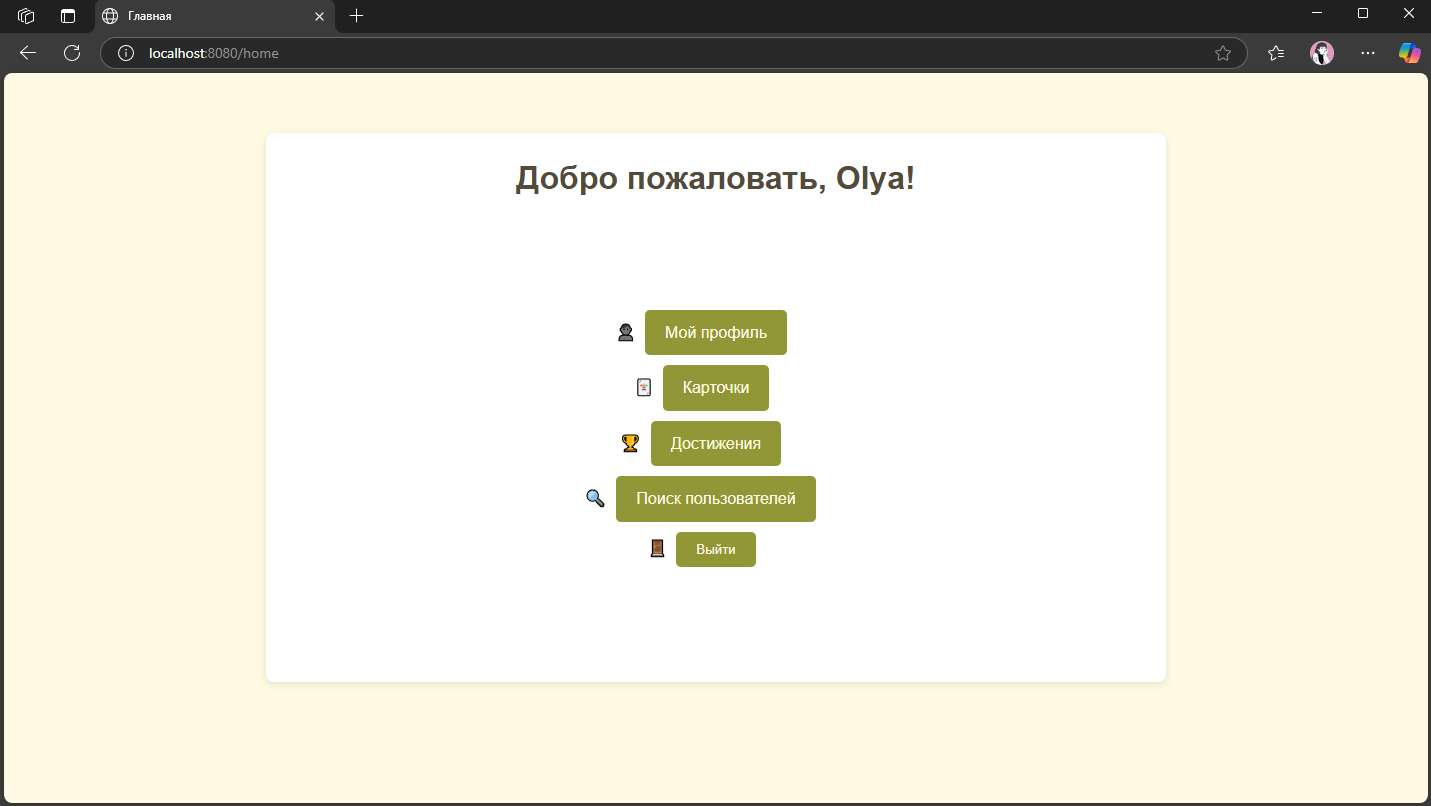
**2.3. Методы для достижений (/api/achievements)**

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

**2.4. Методы для авторизации (/api/auth)**

* POST /api/auth/register – Регистрация нового пользователя, возвращает JWT-токен.
* POST /api/auth/login – Аутентификация по email и паролю, возвращает JWT-токен.
* POST /api/auth/change-password – Смена пароля текущего пользователя.
* POST /api/auth/logout – Выход пользователя с внесением токена в blacklist.

## Итоги работы

  
Рисунок 1 – «домашняя» страница веб-приложения

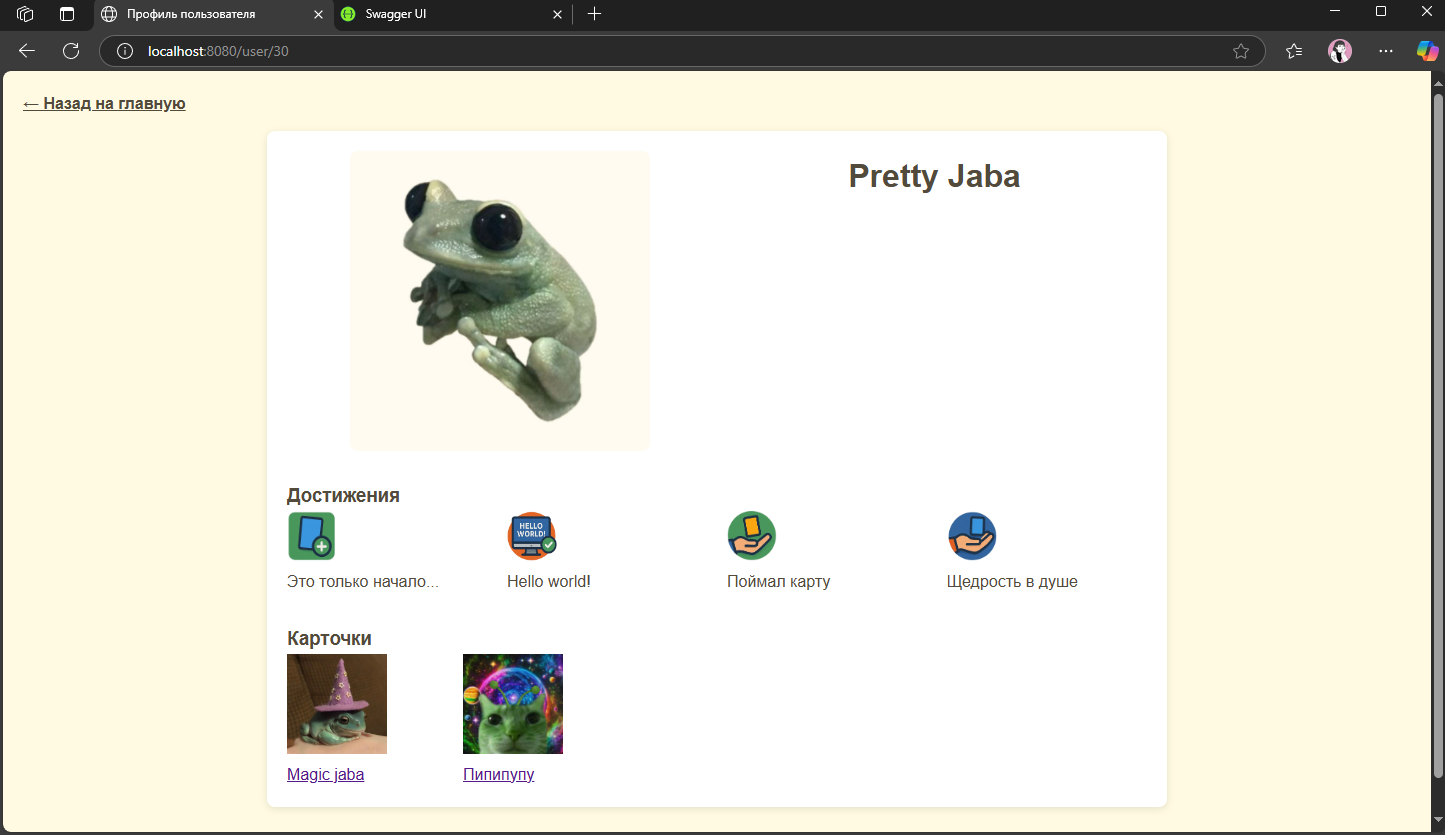
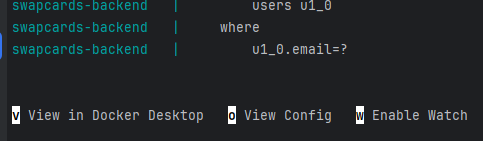
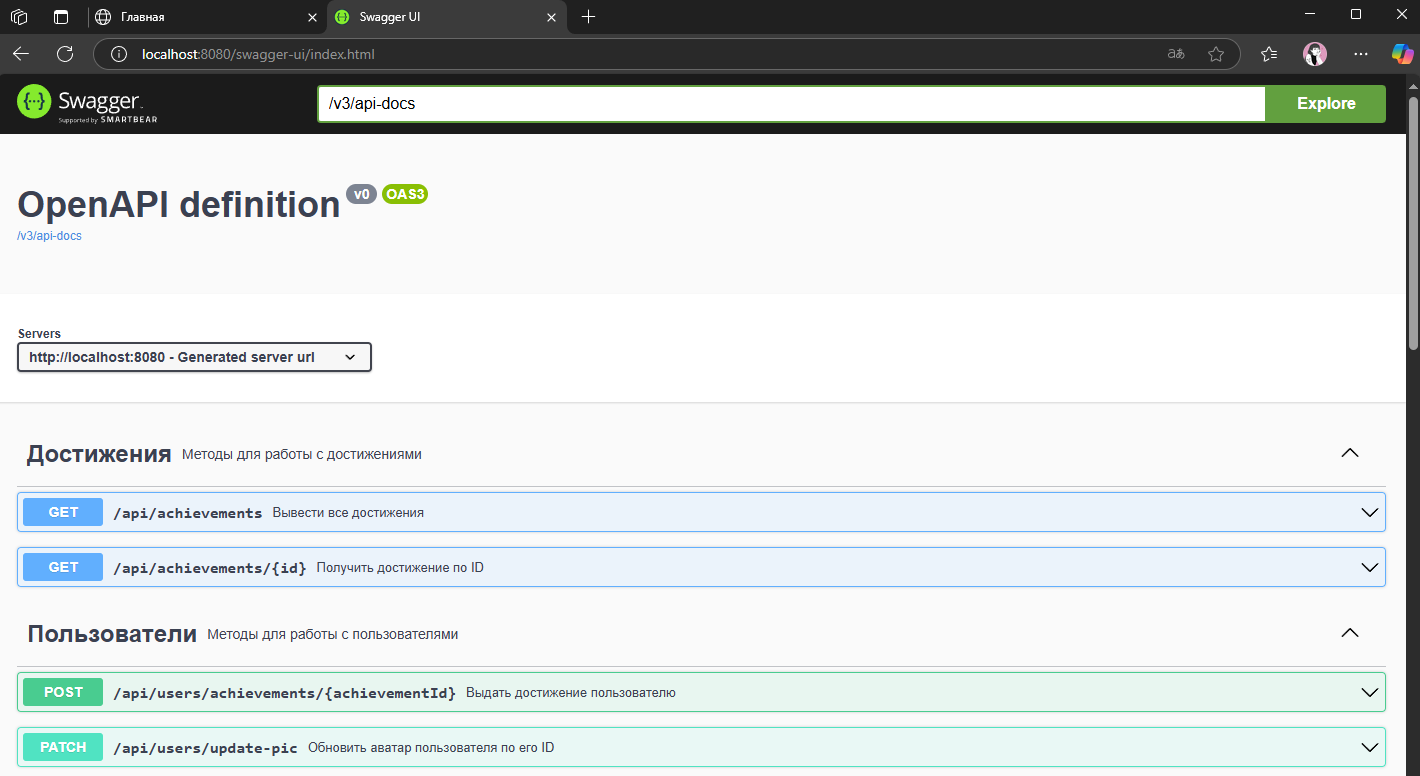
  
Рисунок 2 – профиль пользователя

  
Рисунок 3 – запущенные контейнеры в docker

  
Рисунок 4 – запущенное приложение в терминале

  
Рисунок 5 – Swagger UI