МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ имени академика С.П.КОРОЛЕВА»

ИНСТИТУТ ИНФОРМАТИКИ И КИБЕРНЕТИКИ

КАФЕДРА ТЕХНИЧЕСКОЙ КИБЕРНЕТИКИ

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №0

Тема: «Проектирование приложения»

Дисциплина: «Технологии сетевого программирования»

Выполнил: Якухин И.В., гр.6303

Проверил: Кашапов А.И.

Самара 2025

СОДЕРЖАНИЕ

[Введение 3](#_Toc199539454)

[1 Схема взаимодействия компонентов приложения 4](#_Toc199539455)

[2 Логическая схема базы данных 5](#_Toc199539456)

[3 Структура API 6](#_Toc199539457)

[4 Стек технологий 9](#_Toc199539458)

[Приложение А 10](#_Toc199539459)

[Ссылка на GitHub с проектом 10](#_Toc199539460)

Введение

Cardiogram – приложение с увлекательной игрой для запоминания английских слов с помощью карточек. Стек технологий приложения: Python, Django, React, PostgreSQL. Все компоненты конечной системы контенеризованы с помощью Docker для обеспечения изоляции окружения, удобства развертывания и масштабирования.

Учебный проект разработан с целью демонстрации навыков работы с базой данных и инструментами серверной и клиентской разработки, проектирования структуры приложений и API, контейнеризации.

Ссылка на GitHub страницу со всеми файлами проекта можно найти в приложении А.

1. Схема взаимодействия компонентов приложения

Приложение разделено на 3 компонента: база данных, серверное приложение (backend), клиентское приложение (frontend).

1. **База данных**
   * Хранит все данные приложения: пользователей, колоды, карточки, прогресс.
2. **Серверное приложение**
   * Обрабатывает запросы клиента;
   * Выполняет внутреннюю логику приложения (регистрация, аутентификация, сохранение прогресса);
   * Взаимодействует с базой данных;
   * Возвращает данные клиенту.
3. **Клиентское приложение**
   * Отображает интерфейс, с которым взаимодействует пользователь;
   * Отправляет запросы на сервер для получения или записи данных.
4. Логическая схема базы данных

Основу базы данных составляют 4 таблицы: Users, Cards, Decks, User\_progress. Таблица Users хранит информацию о пользователе и связана с таблицей, хранящей прогресс пользователя – User\_progress. Таблица Cards хранит информацию о карточках и связана с Decks – таблицей колод через поле идентификатора колоды. Таблица User\_progress связывает между собой таблицы пользователей и карточек, позволяя отслеживать прогресс каждого пользователя по каждой отдельной карточке. Так же для корректной работы Django создает несколько вспомогательных таблиц. Схема взаимодействия основных таблиц базы данных представлена на рисунке 1.

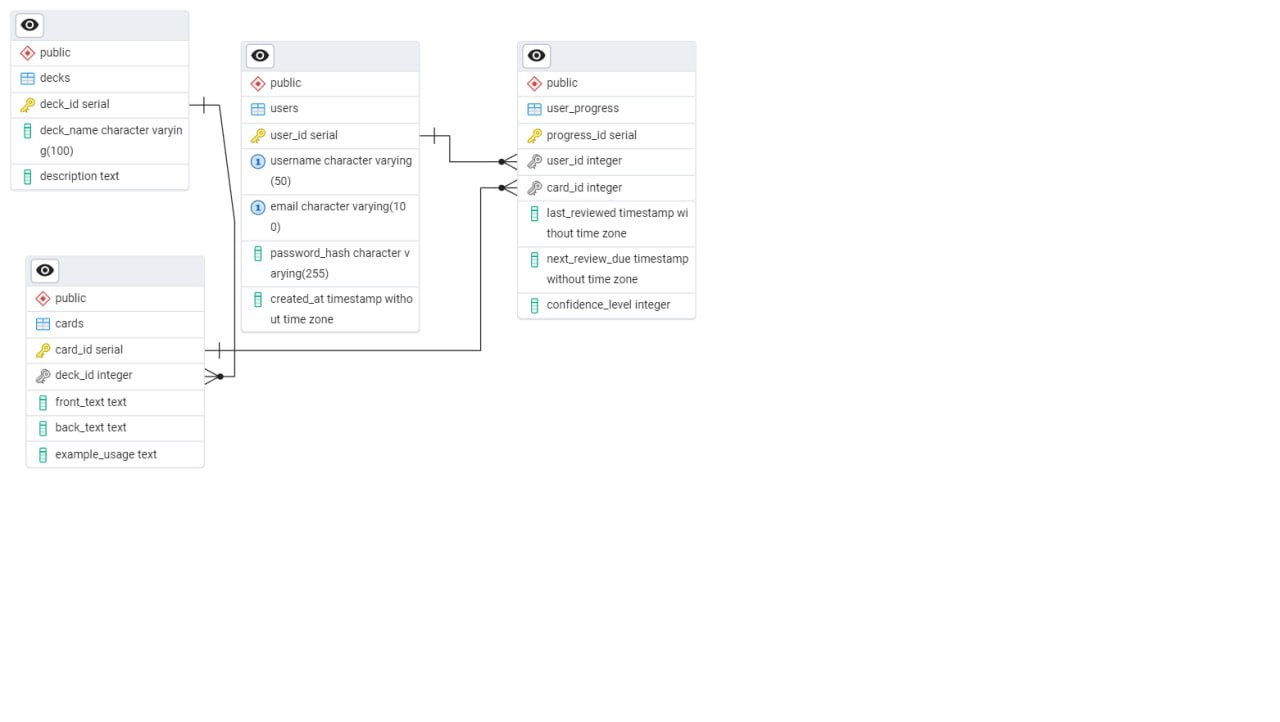


Рисунок 1 –Логическая схема базы данных

1. Структура API

Структура API, включая методы, URL, параметры, форматы запросов и ответов, описана в таблице 1.

Таблица 1 – Описание API

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Действие | Метод | URL | Параметры | Формат запроса | Формат ответа |
| Получение информации о карточке | GET | card/<id> | Id – номер карточки в таблице базы данных |  | {  “id”: 23,  “front\_text”: “forest”,  “back\_text”: “Лес”,  “example\_usage”: “Forest is beautiful! (Лес прекрасен!)”  } |
| Получение информации о колодах | GET | deck/ |  |  | [{  “id”: 4,  “name”: “Nature”,  “description”: “Колода содержит слова, связанные с темой природы.”,  “cover”: “http://127.0.0.1/deck/forest.png”  },  {  …  }] |
| Получение из колоды всех карточек, которых нет у пользователя | GET | deck/<id> | Id – номер колоды в таблице базы данных |  | [{  “id”: 23,  “front\_text”: “forest”,  “back\_text”: “Лес”,  “example\_usage”: “Forest is beautiful! (Лес прекрасен!)”  },  {  “id”: 23,  “front\_text”: “nature”,  “back\_text”: “природа”,  “example\_usage”: “Nature is beautiful! (Природа прекрасна!)”  },…] |
| Регистрация | POST | auth/register | username,  fisrt\_name,  last\_name,  email,  password | {“username”: “Werd”,  “fisrt\_name”: “Иван”,  “last\_name”: “Якухин”,  “email": “werb32@mail.ru”,  “password”: “1234”  } | {“message”: “registration successful”} |
| Авторизация | POST | auth/login | username,  password | {“username”: “Werd”,  “password”: “1234”  } | {“message”: “login successful”} |
| Смена пароля | POST | auth/change-password | old\_password,  new-password | {  “old\_password”: “1234”,  “new-password”: “1111”  } | {“message”: “password changed successful”} |
| Выход из системы | POST | auth/logout |  |  | {“message”: “logout successful”} |
| Изменение личных данных | POST | auth/edit | username,  fisrt\_name,  last\_name,  email | {“username” : “Werd”,  “fisrt\_name”: “Иван”,  “last\_name”: “Якухин”,  “email": “werb32@mail.ru”  } | {“message”: “Profile editing completed successfully!”} |
| Получение данных о прогрессе пользователя | GET | progress/ |  |  | [{  “user”: 23,  “card”: 34,  “attempts”: 23,  “successful\_attempts”: 10  },  {  “user”: 23,  “card”: 32,  “attempts”: 21,  “successful\_attempts”: 5  },…] |
| Добавление карточки пользователю | POST | progress/ | card | {“card”: 23} | {“message”: “Card added to your deck successfully!  ”} |
| Получение профиля пользователя | GET | profile/ |  |  | {“username” : “Werd”,  “fisrt\_name”: “Иван”,  “last\_name”: “Якухин”,  “email": “werb32@mail.ru”  } |
| Пользователь правильно выбрал карточку | PATCH | profile/right | card | {“card”: 23} | {“message”: “the card information has been updated”} |
| Пользователь неправильно выбрал карточку | PATCH | profile/wrong | card | {“card”: 23} | {“message”: “the card information has been updated”} |
| Удаление аккаунта пользователя | DELETE | delete/ |  |  | {'message': 'Account deleted successfully!'} |

1. Стек технологий

Серверное приложение написано на языке Python с использованием библиотеки Django. Для разработки API используется дополнительная библиотека Django-rest-framework. Система JWT токенов реализована с использованием библиотеки SimpleJWT.

В качестве СУБД для проекта был выбран PostgreSQL.

Клиентское приложение реализовано на языке JavaScript с использованием библиотеки React. Так же была использована библиотека axios для выполнения запросов к серверу, библиотека react-router-dom для маршрутизации по одностраничному frontend приложению, библиотека swiper для создания каруселей на странице с карточками пользователя.

Приложение А

Ссылка на GitHub с проектом

https://github.com/BruhMano/CardioGram.git