**Отчет о выполнении лабораторной работы №0**

Якухин Иван, 6303-010302D

*Идея: Приложение с карточками для запоминания английских слов*

1. **Схема взаимодействия клиента и приложения**

**Компоненты системы:**

1. **Клиент (Frontend):**
   * Веб-приложение, с которым взаимодействует пользователь.
   * Отображает интерфейс для регистрации, входа, изучения карточек, просмотра прогресса и т.д.
   * Отправляет запросы на сервер для получения данных или выполнения действий.
2. **Сервер (Backend):**
   * Обрабатывает запросы от клиента.
   * Выполняет бизнес-логику (регистрация, аутентификация, управление карточками, сохранение прогресса).
   * Взаимодействует с базой данных для чтения и записи данных.
   * Возвращает результаты клиенту.
3. **База данных (Database):**
   * Хранит все данные приложения: пользователей, колоды, карточки, прогресс и т.д.

**Схема взаимодействия:**

1. **Клиент → Сервер:**
   * Клиент отправляет HTTP-запросы на сервер через REST API.
   * Примеры запросов:
     + Регистрация пользователя (POST /api/register).
     + Вход в систему (POST /api/login).
     + Получение списка колод (GET /api/decks).
2. **Сервер → База данных:**
   * Сервер обрабатывает запросы клиента и выполняет соответствующие запросы к базе данных.
   * Примеры запросов:
     + Добавление нового пользователя (INSERT INTO users ...).
     + Получение карточек из колоды (SELECT \* FROM cards WHERE deck\_id = ?).
     + Обновление прогресса пользователя (UPDATE user\_progress SET ...).
3. **База данных → Сервер:**
   * База данных возвращает результаты запросов серверу.
   * Примеры ответов:
     + Данные пользователя после регистрации.
     + Список карточек из колоды.
     + Текущий прогресс пользователя.
4. **Сервер → Клиент:**
   * Сервер обрабатывает данные, полученные из базы данных, и возвращает их клиенту.
   * Примеры ответов:
     + Успешная регистрация ({ "status": "success", "user\_id": 1 }).
     + Список колод ({ "decks": [ { "deck\_id": 1, "deck\_name": "Основные слова" } ] }).
     + Прогресс пользователя ({ "progress": [ { "card\_id": 1, "confidence\_level": 3 } ] }).
5. **Логическая схема БД**

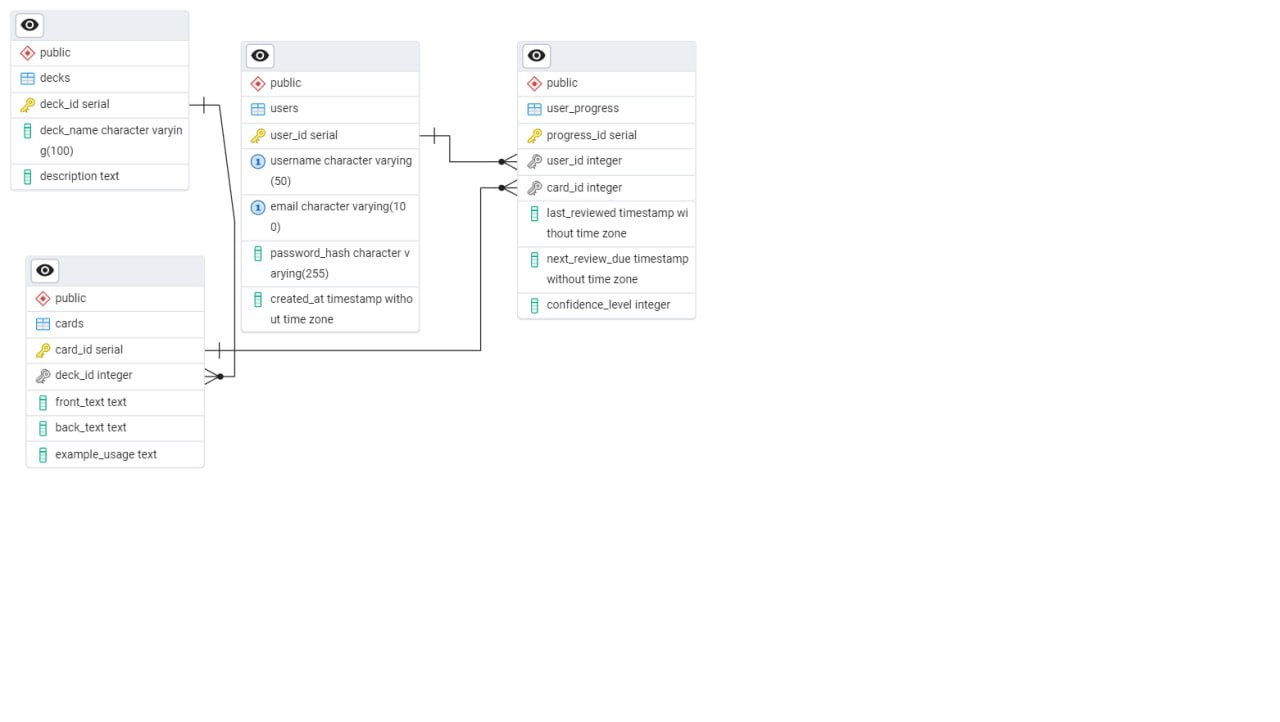


Таблица *Users* хранит информацию о пользователе и связана с таблицей, хранящей прогресс пользователя – *User\_progress*.

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.

***Код создания таблиц:***

CREATE DATABASE CardioGram;

CREATE TABLE users (

user\_id SERIAL PRIMARY KEY, -- Идентификатор пользователя

username VARCHAR(50) UNIQUE NOT NULL, -- Логин пользователя

email VARCHAR(100) UNIQUE NOT NULL, -- Электронная почта

password\_hash VARCHAR(255) NOT NULL, -- Хэш пароля

created\_at TIMESTAMP DEFAULT NOW() -- Дата регистрации

);

CREATE TABLE decks (

deck\_id SERIAL PRIMARY KEY, -- Идентификатор колоды

deck\_name VARCHAR(100) NOT NULL, -- Название колоды

description TEXT, -- Описание колоды

);

CREATE TABLE cards (

card\_id SERIAL PRIMARY KEY, -- Идентификатор карточки

deck\_id INT REFERENCES decks(deck\_id) ON DELETE CASCADE, -- Связь с колодой

front\_text TEXT NOT NULL, -- Текст на лицевой стороне

back\_text TEXT NOT NULL, -- Текст на обратной стороне

example\_usage TEXT -- Пример использования

);

CREATE TABLE user\_progress (

progress\_id SERIAL PRIMARY KEY, -- Идентификатор записи прогресса

user\_id INT REFERENCES users(user\_id) ON DELETE CASCADE, -- Связь с пользователем

card\_id INT REFERENCES cards(card\_id) ON DELETE CASCADE, -- Связь с карточкой

last\_reviewed TIMESTAMP, -- Дата последнего повторения

next\_review\_due TIMESTAMP, -- Дата следующего повторения

confidence\_level INT DEFAULT 0 -- Уровень уверенности (от 0 до 5)

);

1. **Структура API**
2. **Регистрация пользователя**

* **Метод:** POST
* **URL:** /auth/register
* **Параметры запроса:** {

"username": "john\_doe",

"email": "john@example.com",

"password": "securepassword123"

}

* **Ответ:** {

"status": "success",

"user\_id": 1,

"message": "User registered successfully"

}

1. **Вход в систему**

* **Метод:** POST
* **URL:** /auth/login
* **Параметры запроса:** {

"email": "john@example.com",

"password": "securepassword123"

}

* **Ответ:** {

"status": "success",

"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",

"user\_id": 1

}

1. **Получить все колоды**

* **Метод:** GET
* **URL:** /decks
* **Ответ:** {

"decks": [

{

"deck\_id": 1,

"deck\_name": "Основные слова",

"description": "Базовые слова для начинающих",

"created\_by": 1

}

]

}

1. **Получить карточки из колоды**

* **Метод:** GET
* **URL:** /decks/{deck\_id}/cards
* **Ответ**: {

"cards": [

{

"card\_id": 1,

"front\_text": "Hello",

"back\_text": "Привет",

"example\_usage": "Hello, how are you?"

}

]

}

1. **Добавить карточку в список для изучения**

* **Метод:** POST
* **URL:** /users/{user\_id}/selected\_cards
* **Параметры запроса (JSON):** {

"card\_id": 1

}

* **Ответ (JSON):** {

"status": "success",

"message": "Card added to selected cards successfully"

}

1. **Удалить карточку из списка для изучения**

* **Метод:** DELETE
* **URL:** /users/{user\_id}/selected\_cards/{card\_id}
* **Параметры запроса:** Нет
* **Ответ (JSON):** {

"status": "success",

"message": "Card removed from selected cards successfully"

}

1. **Получить выбранные карточки пользователя**

* **Метод:** GET
* **URL:** /users/{user\_id}/selected\_cards
* **Параметры запроса:** Нет
* **Ответ (JSON):** {

"selected\_cards": [

{

"card\_id": 1,

"front\_text": "Hello",

"back\_text": "Привет",

"example\_usage": "Hello, how are you?

}

]

}

1. **Стек технологий**

**Backend:** python, Django

**Frontend:** React

**СУБД**: PostgreSQL

1. **Подготовка окружения**

python -m venv venv

venv/Scripts/activate

pip install django

pip install dot-env

1. **GitHub**

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