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 №4

Тема: «Разработка пользовательского интерфейса и взаимодействие с API»

Дисциплина: «Технологии сетевого программирования»

Выполнил: Якухин И.В., гр.6303

Проверил: Кашапов А.И.

Самара 2025

СОДЕРЖАНИЕ

[Введение 3](#_Toc199551638)

[1 Разработка пользовательского интерфейса 4](#_Toc199551639)

[2 Настройка взаимодействия с сервером 6](#_Toc199551640)

[3 Работа с аутентификацией 7](#_Toc199551641)

[Приложение А 8](#_Toc199551642)

[Ссылка на GitHub с проектом 8](#_Toc199551643)

[Приложение Б 9](#_Toc199551644)

[Страница обучения card-learn.html 9](#_Toc199551645)

[React компонент Learn.jsx 10](#_Toc199551646)

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. Разработка пользовательского интерфейса

Для создания пользовательского интерфейса в программе Photoshop были выполнены макеты главной страницы, страниц профиля, колод, карточек пользователя, выбора карточки и обучения. На рисунке 1 приведен скриншот макета главной страницы.

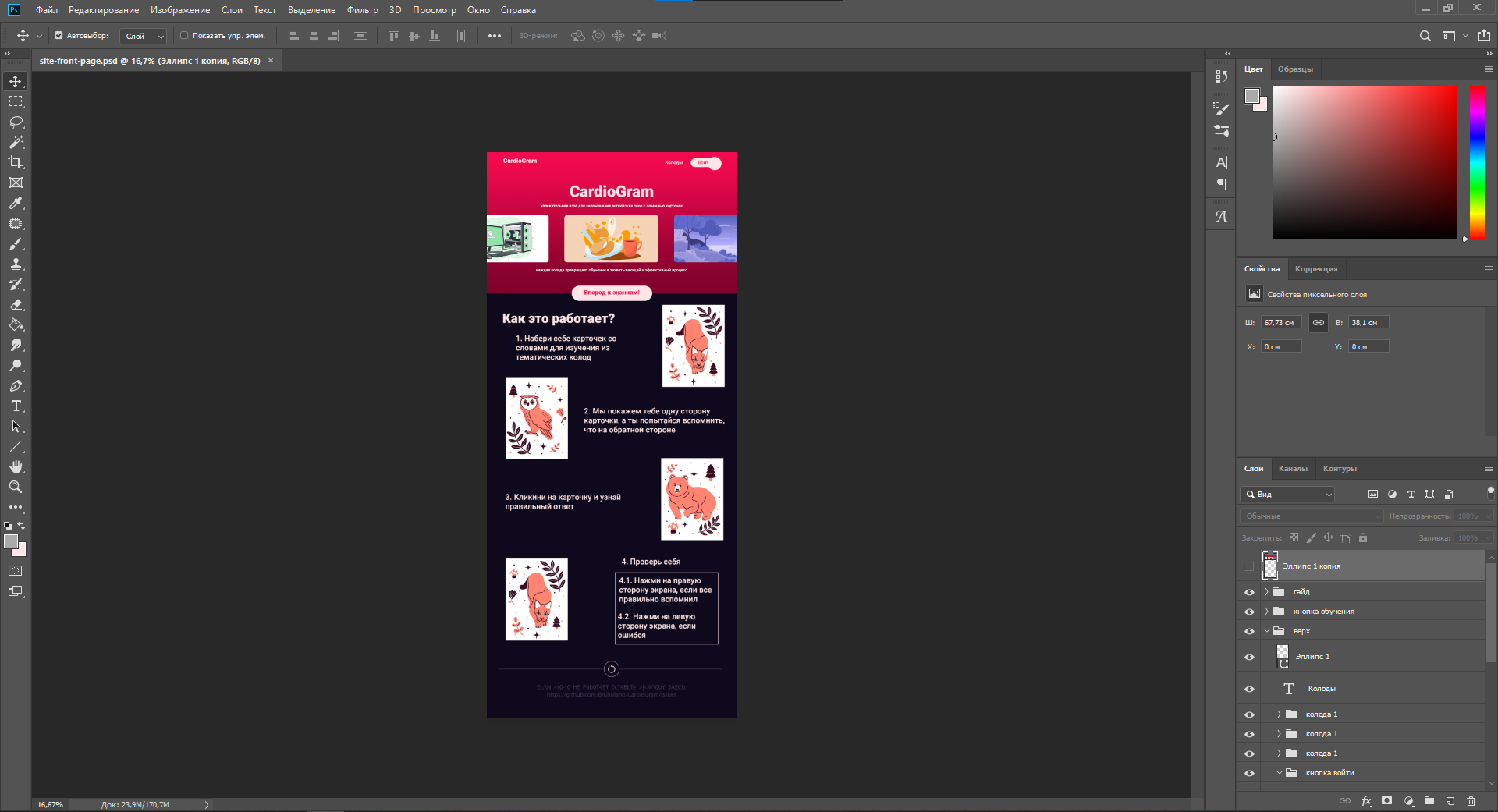


Рисунок 1 – макет главной страницы приложения в Photoshop

После создания макетов они были перенесены в формат html/css для последующей интеграции в React. Пример страницы обучения в формате html приведен в приложении Б.

Также помимо вышеперечисленных шаблонов в html были созданы страницы с формами для обработки данных, а именно страницы регистрации, входа, смены пароля, редактирования данных пользователя. Все эти страницы создавались по одному шаблону с изменением полей формы. Пример страницы с формой представлен на рисунке 2.

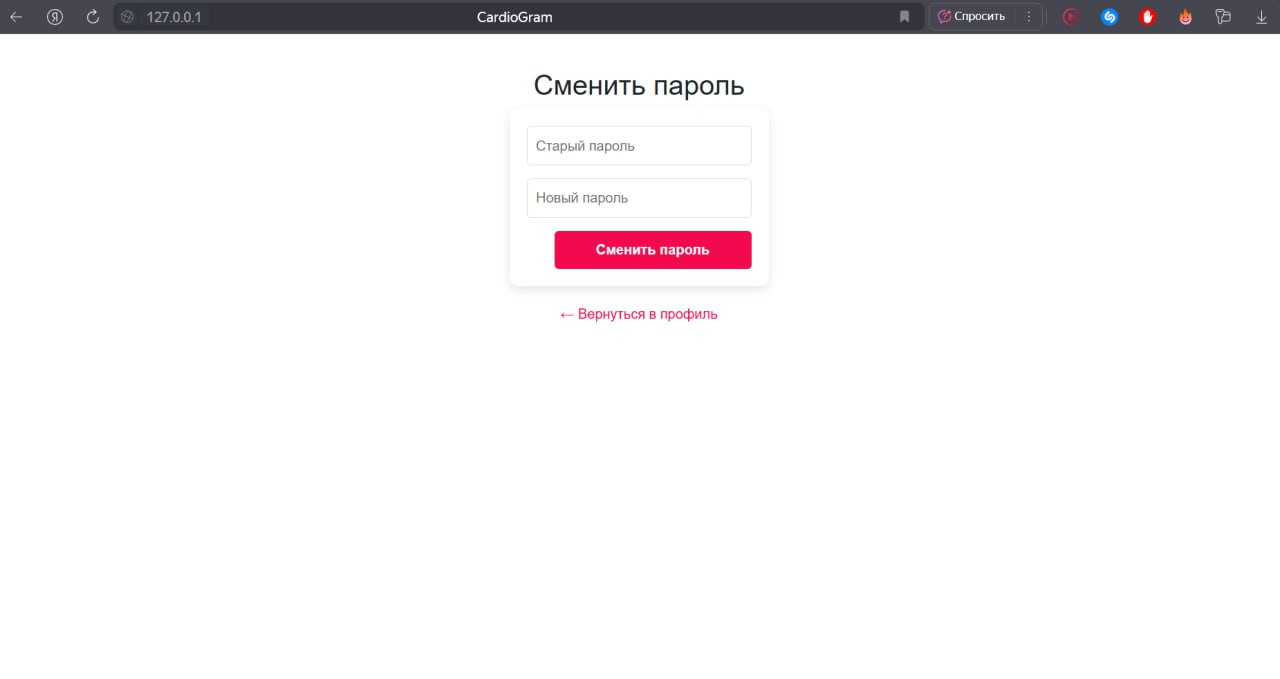


Рисунок 2 – Скриншот страницы смены пароля

1. Настройка взаимодействия с сервером

После создания всех html шаблонов они были преобразованы в React компоненты. Стили из .css файлов были перенесены в .module.css файлы и правильно подключены в React. Следующим шагом было налажено взаимодействие клиентского приложения с серверным. В React компонентах запросы к серверу посылаются с помощью библиотеки axios. Данные с сервера попадают в клиентское приложение и динамически отрисовываются на страницах. Пример React компонента с запросом к серверу и динамическим обновлением страницы приведен в приложении Б. На рисунке 3 представлена страница с изображением и текстом, полученными от сервера.

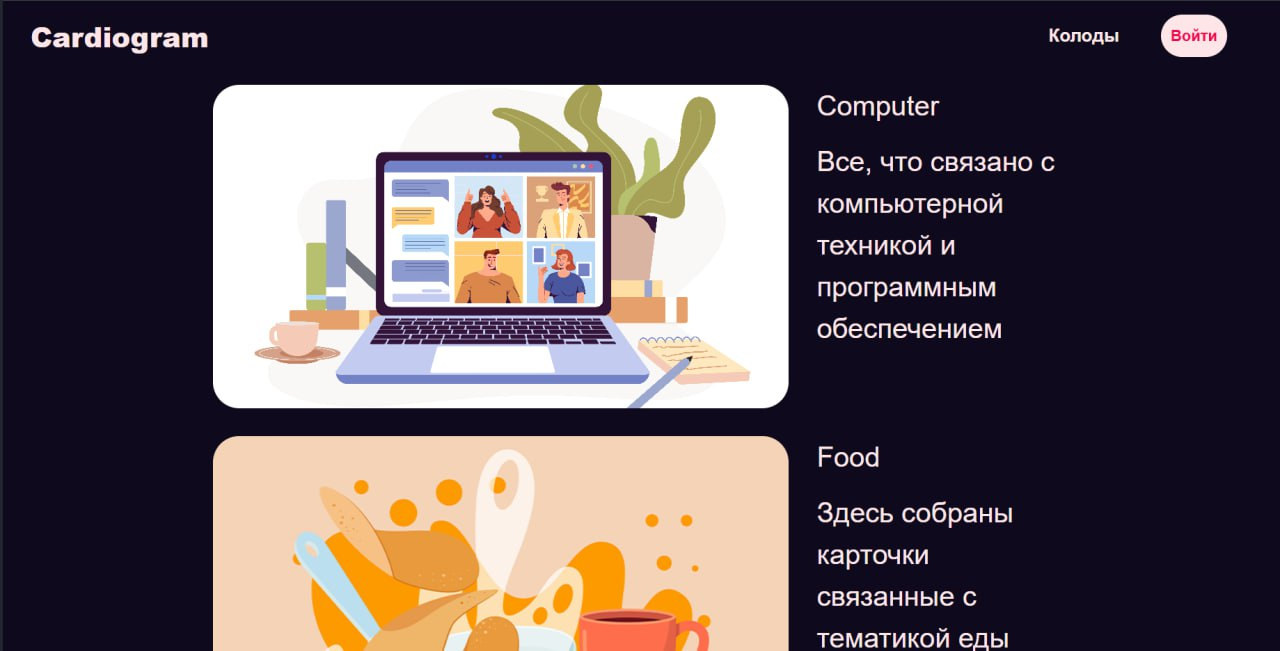


Рисунок 3 – Скриншот страницы с колодами, на которой данные загружаются с сервера

1. Работа с аутентификацией

Когда пользователь регистрируется в приложении или входит в свой аккаунт, сервер создает для него JWT токены и сохраняет их в Cookie файлы

Axios запросы используют Cookie файлы созданные сервером для получения доступа к защищенным эндпоинтам. Так как клиентская и серверная части приложения находятся на разных доменах, необходимо провести дополнительную настройку сервера, чтобы сделать такие запросы возможными.

Для разрешения запросов с других доменов на сервере была установлена библиотека corsheaders и добавлены в settings.py параметры CORS\_ALLOWED\_ORIGINS и CORS\_ALLOW\_CREDENTIALS. Теперь axios запросы с клиента с параметром withCredentials: true будут корректно обработаны вместе с Cookie файлами.

Приложение А

Ссылка на GitHub с проектом

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

Приложение Б

Страница обучения card-learn.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>CardioGram</title>

<link rel="stylesheet" href="static/cards-style.css" />

<style>

.screen-left,

.screen-right {

position: absolute;

top: 0;

bottom: 0;

width: 50%;

z-index: 3;

cursor: pointer;

}

.screen-right {

right: 0;

}

</style>

</head>

<body style="background-image: url(static/cards-background.png)">

<header class="header">

<div class="logo">CardioGram</div>

<nav class="nav">

<a href="#">Колоды</a>

<!-- <button class="login-button">Войти</button> -->

<button class="button">

<img src="static/profile.png" alt="" class="profile-img" />

</button>

</nav>

</header>

<div class="card-space">

<!-- <div class="card-front">Forest</div> -->

<div class="card-back">

<p class="title">Лес</p>

<div class="image"></div>

<div>

<p class="desc">The forest is home to many animals</p>

<p class="translation">(Лес является домом для многих животных.)</p>

</div>

</div>

</div>

</body>

</html>

React компонент Learn.jsx

import React, { useState, useEffect } from 'react';

import Header from '../components/Header';

import CardBack from '../components/CardBack';

import LeftRightButtons from '../components/LeftRightButtons';

import bg from '../assets/static/cards-background.png';

import axios from 'axios';

import styles from '../assets/static/card.module.css';

const Learn = () => {

const [cards, setCards] = useState([]);

const [card, setCard] = useState(null);

const [currentIndex, setCurrentIndex] = useState(0);

const [showBack, setShowBack] = useState(false);

useEffect(() => {

axios

.get('http://127.0.0.1:8000/api/progress/', { withCredentials: true })

.then((res) => {

setCards(res.data);

})

.catch((err) => {

console.error(err);

});

}, []);

useEffect(() => {

if (cards.length && currentIndex !== -1) {

axios

.get(`http://127.0.0.1:8000/api/card/${cards[currentIndex].card}/`, { withCredentials: true })

.then((res) => {

setCard(res.data);

setShowBack(false);

})

.catch((err) => {

console.error(err);

});

}

}, [cards, currentIndex]);

const handleLeft = () => {

if (currentIndex !== -1) {

axios

.patch('http://127.0.0.1:8000/api/progress/right/', { card\_id: cards[currentIndex].card }, { withCredentials: true })

// .then(() => {})

.catch((err) => {

console.error(err);

});

}

if (currentIndex < cards.length - 1 && currentIndex !== -1) {

setCurrentIndex(currentIndex + 1);

} else {

setCurrentIndex(-1);

}

};

const handleRight = () => {

if (currentIndex !== -1) {

axios

.patch('http://127.0.0.1:8000/api/progress/wrong/', { card\_id: cards[currentIndex].card }, { withCredentials: true })

.then(() => {})

.catch((err) => {

console.error(err);

});

}

if (currentIndex < cards.length - 1 && currentIndex !== -1) {

setCurrentIndex(currentIndex + 1);

} else {

setCurrentIndex(-1);

}

};

return (

<div className={styles.cardPage} style={{ backgroundImage: `url(${bg})` }}>

<Header />

<LeftRightButtons

leftFunc={showBack ? handleLeft: () => {setShowBack(true)}}

rightFunc={showBack ? handleRight: () => {setShowBack(true)}}

/>

{currentIndex !== -1 ? (

<CardBack

title={card?.front\_text}

translation={card?.back\_text}

description={card?.example\_usage}

showBack={showBack}

isSmall = {false}

deckId = {card?.deck}

/>

) : (

<p>Карточки закончились! Можете набрать новых или отдохнуть)</p>

)}

</div>

);

};

export default Learn;