|  |  |  |
| --- | --- | --- |
| Защищено:  Гапанюк Ю.Е.    "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г. |  | Демонстрация:  Гапанюк Ю.Е.  "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г. |

**Отчет по лабораторной работе № 8 по курсу**

**Разработка интернет-приложений**

**ГУИМЦ**

#### Тема работы: " Разработка пользовательского интерфейса с использованием библиотеки React "

14

(количество листов)

Вариант № **1**

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛЬ: |  |
| студент группы ИУ5Ц-74Б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | (подпись) |
| Карпов И.Е. | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г. |

Москва, МГТУ - 2021

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**СОДЕРЖАНИЕ**

[1. Описание задания 3](#_Toc90926229)

[2. Текст программы 4](#_Toc90926230)

[3. Результаты выполнения программы 12](#_Toc90926231)

# Описание задания

На основе [методических указаний](https://github.com/iu5team/iu5web-fall-2021/blob/main/tutorials/lab8/lab8_tutorial.md) разработайте React-приложение. Для создания приложения необходимо решить следующие задачи:

1. Создать стартовый React-проект. Удалить неиспользуемый код. Организовать директории для страниц, компонентов, утилит и работы с сетью.
2. Организовать роутниг в веб-приложении.
3. Разработать базовые страницы, на которых будут отображаться сущности из выбранной вами предметной области:
   1. Стартовая страница.
   2. Страница просмотра списка объектов.
   3. Страница просмотра конкретного объекта.
4. Вынести переиспользуемые компоненты в отдельные файлы:
   1. Для навигации по приложению можно добавить header.
   2. Для отображения дополнительной информации (данные о студенте и предметной области) можно использовать footer.
   3. Источники ввода-вывода (поля ввода (inputs)/формы/текстовые блоки).
   4. Переиспользуемые таблицы/гриды.
5. Добавить асинхронные запросы в разработанный API, чтобы страница получала данные с сервера.
6. Если в Вашем проекте реализована сложная логика работы с состоянием приложения, то рекомендуется добавить пользовательские хуки.
7. Страницы приложения должны хорошо отображаться как на больших, так и на маленьких экранах.

# Текст программы

**package.json**

{  
 "name": "lab8",  
 "version": "0.1.0",  
 "private": true,  
 "dependencies": {  
 "@testing-library/jest-dom": "^5.11.4",  
 "@testing-library/react": "^11.1.0",  
 "@testing-library/user-event": "^12.1.10",  
 "axios": "^0.24.0",  
 "react": "^17.0.2",  
 "react-bootstrap": "^2.0.3",  
 "react-dom": "^17.0.2",  
 "react-router": "^6.0.2",  
 "react-router-dom": "^6.0.2",  
 "react-scripts": "4.0.3",  
 "web-vitals": "^1.0.1"  
 },  
 "scripts": {  
 "start": "react-scripts start",  
 "build": "react-scripts build",  
 "test": "react-scripts test",  
 "eject": "react-scripts eject"  
 },  
 "eslintConfig": {  
 "extends": [  
 "react-app",  
 "react-app/jest"  
 ]  
 },  
 "browserslist": {  
 "production": [  
 ">0.2%",  
 "not dead",  
 "not op\_mini all"  
 ],  
 "development": [  
 "last 1 chrome version",  
 "last 1 firefox version",  
 "last 1 safari version"  
 ]  
 }  
}

**public/index.html**

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="utf-8"/>  
 <link rel="icon" href="%PUBLIC\_URL%/favicon.ico"/>  
 <meta name="viewport" content="width=device-width, initial-scale=1"/>  
 <meta name="theme-color" content="#000000"/>  
 <meta  
 name="description"  
 content="Web site created using create-react-app"  
 />  
 *<!--  
 manifest.json provides metadata used when your web app is installed on a  
 user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/  
 -->* <link rel="manifest" href="%PUBLIC\_URL%/manifest.json"/>  
 *<!--  
 Notice the use of %PUBLIC\_URL% in the tags above.  
 It will be replaced with the URL of the `public` folder during the build.  
 Only files inside the `public` folder can be referenced from the HTML.  
  
 Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC\_URL%/favicon.ico" will  
 work correctly both with client-side routing and a non-root public URL.  
 Learn how to configure a non-root public URL by running `npm run build`.  
 -->* <title>React App</title>  
 *<!-- Bootstrap CSS -->* <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet"  
 integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">  
</head>  
<body>  
<noscript>You need to enable JavaScript to run this app.</noscript>  
<div id="root"></div>  
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"  
 integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p"  
 crossorigin="anonymous"></script>  
</body>  
</html>

**public/robots.txt**

# https://www.robotstxt.org/robotstxt.html  
User-agent: \*  
Disallow:

**public/pic.jpg**



**src/App.js**

import React from 'react';  
import *Header* from "./components/header/Header";  
import *Main* from "./components/main/Main";  
import *Footer* from "./components/footer/Footer";  
  
const *App* = () => {  
 return (  
 <div>  
 <Header/>  
 <Main/>  
 <Footer/>  
 </div>  
 );  
};  
  
export default *App*;

**src/index.js**

import React from 'react';  
import ReactDOM from 'react-dom';  
import *App* from './App';  
import {BrowserRouter} from "react-router-dom";  
  
ReactDOM.render((  
 <BrowserRouter>  
 <App/>  
 </BrowserRouter>  
), document.getElementById('root'));

**src/utils/getFromServer.js**

import axios from "axios";  
  
export const *getFromServer* = (url) => {  
 return axios.get(url).then((res) => res.data).catch((msg) => *alert*(msg));  
};

**src/utils/useWindowSize.js**

import {*useEffect*, *useState*} from 'react';  
  
const *useWindowSize* = () => {  
 *// в данном пользовательском хуке мы используем хук состояния и хук эффекта* const [windowSize, setWindowSize] = *useState*({  
 width: undefined,  
 height: undefined,  
 });  
 *useEffect*(() => {  
 *// при вызове этой функции, мы будем "класть" в состояние актуальную высоту и ширирну экрана* function handleResize() {  
 setWindowSize({  
 width: window.innerWidth,  
 height: window.innerHeight,  
 });  
 }  
  
 *// В данном примере мы будет подписываться на изменение размеров экрана, чтобы всегда иметь актуальные данные* window.addEventListener("resize", handleResize);  
 handleResize();  
 *// После того, как компонент "уничтожается", желательно избавиться от всех "слушателей", чтобы не тратить ресурсы браузера* return () => window.removeEventListener("resize", handleResize);  
 }, []);  
  
 return windowSize;  
}  
  
export default *useWindowSize*;

**src/components/card/AnimalCard.jsx**

import React from 'react';  
import {Card} from 'react-bootstrap'  
import {*Link*} from "react-router-dom";  
  
const *AnimalCard* = ({id, animal\_name, animal\_type}) => {  
  
 return (  
 <Card className="card">  
 <Card.Body>  
 <div className="textStyle">  
 <Card.Title>{animal\_name}</Card.Title>  
 </div>  
 <div className="textStyle">  
 <Card.Text>{animal\_type}</Card.Text>  
 </div>  
 <Link to={'/animal/'+id.toString()}>Подробнее</Link>  
 </Card.Body>  
 </Card>  
 );  
};  
  
export default *AnimalCard*;

**src/components/footer/Footer.jsx**

import React from 'react';  
  
const *Footer* = () => {  
 return (  
 <footer className='d-flex justify-content-center'>  
 <div style={{'text-align':'center'}}>  
 <h4>Сайт разработал</h4>  
 <h4>студент группы ИУ5Ц-74Б</h4>  
 <h5>Карпов И.Е.</h5>  
 <p>Предметная область: животные</p>  
 </div>  
 </footer>  
 );  
};  
  
export default *Footer*;

**src/components/main/Main.jsx**

import React from 'react';  
import {Routes, Route} from "react-router-dom";  
import *MainPage* from "../../pages/main/MainPage";  
import *Animals* from "../../pages/animals/Animals";  
import *API* from "../../pages/api/API";  
import *AnimalDetail* from "../../pages/animals/AnimalDetail";  
  
  
const *Main* = () => {  
 return (  
 <div className="flex wrapper" style={{'textAlign':'center'}}>  
 <Routes>  
 <Route exact path='/' element={<MainPage/>}/>  
 <Route path='/animals' element={<Animals/>}/>  
 <Route path='/animal/:id' element={<AnimalDetail/>}/>  
 <Route path='/api' element={<API/>}/>  
 </Routes>  
 </div>  
 );  
};  
  
export default *Main*;

**src/pages/animals/AnimalDetail.jsx**

import React, {*useEffect*, *useState*} from 'react';  
import {*useParams*} from "react-router";  
import {*getFromServer*} from "../../utils/getFromServer";  
  
const *AnimalDetail* = () => {  
 const id = *useParams*().id;  
  
 const [oper, setOper] = *useState*();  
 const [isLoaded, setIsLoaded] = *useState*(false);  
 *useEffect*(() => {  
 *getFromServer*('http://127.0.0.1:8000/animals/' + id.toString()).then((data) => {  
 setOper(data);  
 setIsLoaded(true);  
 });  
 }, []);  
  
 return (  
 <div className="d-flex justify-content-center m-5">  
 <table style={{'border': '2px solid black'}}>  
 <tr>  
 <td>Вид:</td>  
 <td>{isLoaded ? oper.animal\_type : 'загружается...'}</td>  
 </tr>  
 <tr>  
 <td style={{'padding':'10px'}}>Кличка:</td>  
 <td>{isLoaded ? oper.animal\_name : 'загружается...'}</td>  
 </tr>  
 <tr>  
 <td>Фото:</td>  
 <td>{isLoaded ? <img src={oper.animal\_photo} alt="Ошибка" width="99%"/> : 'загружается...'}</td>  
 </tr>  
 </table>  
 </div>  
 );  
};  
  
export default *AnimalDetail*;

**src/pages/animals/Animals.jsx**

import React, {*useState*} from 'react';  
import {Button, Row, Col} from "react-bootstrap";  
import *AnimalCard* from "../../components/card/AnimalCard";  
import *useWindowSize* from "../../utils/useWindowSize";  
import {*getFromServer*} from "../../utils/getFromServer";  
  
const *Animals* = () => {  
  
 const [opers, setOpers] = *useState*([]);  
  
 const loadOpers = async () => {  
 const results = await *getFromServer*('http://127.0.0.1:8000/animals/');  
 await setOpers(results);  
 document.getElementById("loadButton").hidden = true;  
 }  
  
 const {width} = *useWindowSize*();  
 const isMobile = width && width <= 600;  
  
 return (  
 <div className="d-flex flex-column container justify-content-center">  
 <div className="m-5" id="loadButton">  
 <Button onClick={loadOpers}>Загрузить список животных</Button>  
 </div>  
 <div className="mb-5">  
 <Row xs={1} md={isMobile ? 1 : 3} className="g-3">  
 {opers.map((item, index) => {  
 return <Col>  
 <AnimalCard {...item} key={index}/>  
 </Col>  
 })}  
 </Row>  
 </div>  
 </div>  
 );  
};  
  
export default *Animals*;

**src/pages/api/API.jsx**

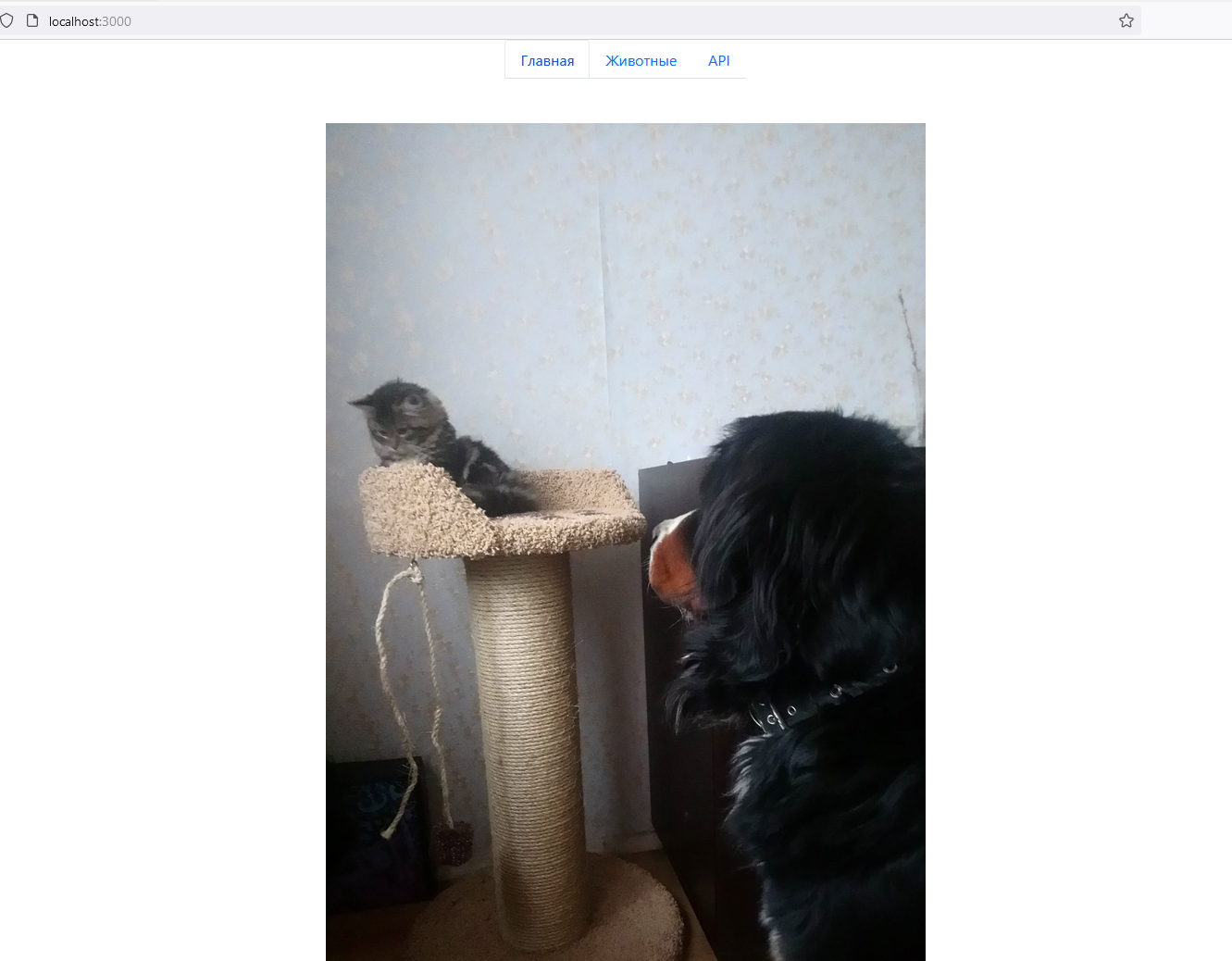
import React from 'react';  
  
const *API* = () => {  
  
 return (  
 <div className="d-flex flex-column container justify-content-center">  
 Для внесения изменений в данные через браузер нажмите на ссылку ниже<br/>  
 <h1><a href="http://localhost:8000/">Открыть API</a></h1>  
 </div>  
 );  
};  
  
export default *API*;

**src/pages/main/MainPage.jsx**

import React from 'react';  
  
const *MainPage* = () => {  
 return (  
 <div>  
 <div className='flex container mt-5'>  
 <img  
 src='pic.jpg'  
 style={{'height':'100%','max-height':'100%', 'width':'100%', 'max-width':'50%'}}  
 alt='Фото 2-х животных'/>  
 </div>  
 </div>  
 );  
};  
  
export default *MainPage*;

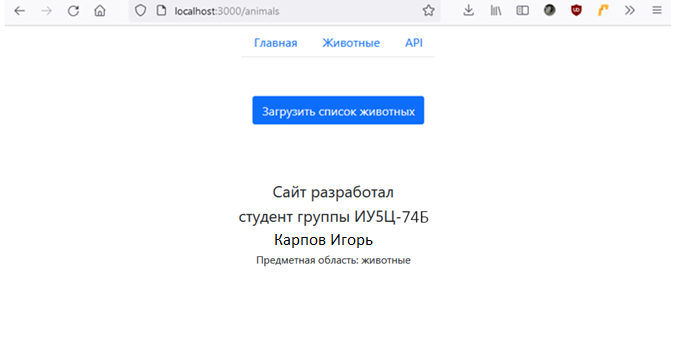
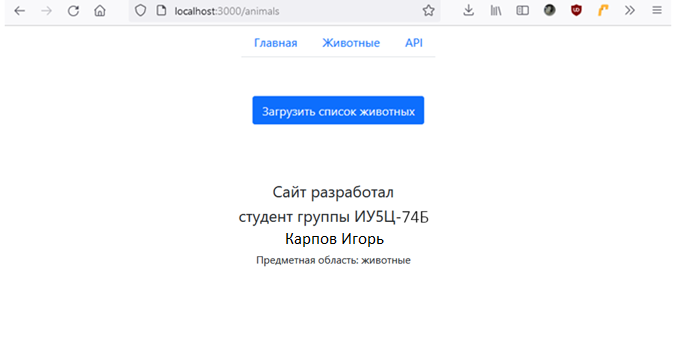
# Результаты выполнения программы

**Главная страница**



Содержит одно фото. Кнопки перехода на другие страницы и информация об ЛР имеется на всех страницах.

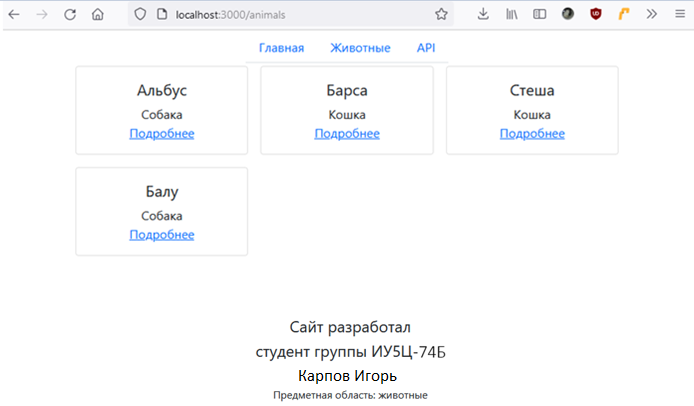
**Животные (до загрузки)**





Содержит кнопку «Загрузить список животных», после нажатия на которую отобразится список из карточек (см. ниже).

**Животные (после загрузки)**

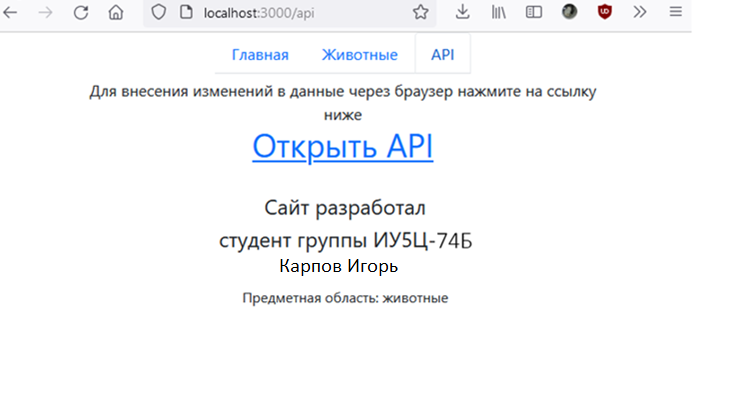




При нажатии ссылки «Подробнее» открывается страница с информацией об выбранном животном



**Переход к API**





После нажатия на ссылку «Открыть API» откроется WEB-интерфейс API, созданный в ЛР №6.