|  |  |
| --- | --- |
|  | МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)» |

Институт № 3 «Системы управления, информатика и электроэнергетика»

Кафедра 304 «Вычислительные машины, системы и сети»

Курсовая работа  
по дисциплине «Web-технологии»  
на тему Разработка web-приложения “Информационный сайт о фильме “Пираты Карибского моря””

Выполнил  
студент группы М3О-107БВ-24  
Коркина К.В.

Приняла  
Масленикова Т.Ю.

Москва  
2025

# Задание

*Используя концепцию CRUD (Create, Read, Update, Delete), разработать single-page application, представляющее собой список / витрину уникальных сущностей.*

*В приложении должен быть реализован функционал:*

*• добавления и удаления сущностей,*

*• просмотра сущностей,*

*• редактирования сущностей.*

*Под «сущностями» могут пониматься, например, видеоигровые персонажи, преподаватели, музеи, книги, памятные фотографии и так далее.*

*В процессе разработки необходимо использовать стек технологий PERN:*

*• PostgreSQL,*

*• Express,*

*• React,*

*• Node.js.*

## Сервер

**Описание функций**

1. **pool**

**Назначение:** получение доступа к базе данных

**Обращение:**

*const pool = new Pool({*

*user: 'postgres',*

*host: 'db',*

*database: 'moviecharacters',*

*password: 'password',*

*port: 5432,*

*retryDelay: 5000,*

*retryTimeout: 20000*

*});*

1. **get/characters**

**Назначение:** получение списка всех персонажей

**Обращение:**

*app.get('/characters', async (req, res) => {*

*try {*

*const { rows } = await pool.query('SELECT \* FROM characters ORDER BY id');*

*res.json(rows);*

*} catch (err) {*

*console.error(err);*

*res.status(500).json({ error: 'Internal server error' });*

*}*

*});*

1. **get/characters/:id**

**Назначение:** получить персонажа по id

**Обращение:**

*app.get('/characters/:id', async (req, res) => {*

*const { id } = req.params;*

*try {*

*const { rows } = await pool.query('SELECT \* FROM characters WHERE id = $1', [id]);*

*if (rows.length === 0) {*

*return res.status(404).json({ error: 'Character not found' });*

*}*

*res.json(rows[0]);*

*} catch (err) {*

*console.error(err);*

*res.status(500).json({ error: 'Internal server error' });*

*}*

*});*

1. **post/characters**

**Назначение**:Добавление нового персонажа

**Обращение**:

*app.post('/characters', async (req, res) => {*

*const { name } = req.body;*

*try {*

*const { rows } = await pool.query(*

*'INSERT INTO characters (name) VALUES ($1) RETURNING \*',*

*[name]*

*);*

*res.status(201).json(rows[0]);*

*} catch (err) {*

*console.error(err);*

*res.status(500).json({ error: 'Internal server error' });*

*}*

*});*

1. **Put/characters/:id**

**Назначение**: обновление данных персонажа по id

**Обращение**:

*app.put('/characters/:id', async (req, res) => {*

*const { id } = req.params;*

*const { name } = req.body;*

*try {*

*const { rows } = await pool.query(*

*'UPDATE characters SET name = $1 WHERE id = $2 RETURNING \*',*

*[name, id]*

*);*

*if (rows.length === 0) {*

*return res.status(404).json({ error: 'Character not found' });*

*}*

*res.json(rows[0]);*

*} catch (err) {*

*console.error(err);*

*res.status(500).json({ error: 'Internal server error' });*

*}*

*});*

#### **delete /characters/:id**

**Назначение:** Удаление персонажа по ID.

**Обращение:**

*app.delete('/characters/:id', async (req, res) => {*

*const { id } = req.params;*

*try {*

*const { rowCount } = await pool.query('DELETE FROM characters WHERE id = $1', [id]);*

*if (rowCount === 0) {*

*return res.status(404).json({ error: 'Character not found' });*

*}*

*res.status(204).send();*

*} catch (err) {*

*console.error(err);*

*res.status(500).json({ error: 'Internal server error' });*

*}*

*});*

**Код:**

* index.js

*const express = require('express');*

*const bodyParser = require('body-parser');*

*const cors = require('cors');*

*const { Pool } = require('pg');*

*const app = express();*

*app.use(cors());*

*app.use(bodyParser.json());*

*const pool = new Pool({*

*user: 'postgres',*

*host: 'db',*

*database: 'moviecharacters',*

*password: 'password',*

*port: 5432,*

*retryDelay: 5000,*

*retryTimeout: 20000*

*});*

*pool.query('SELECT NOW()', (err, res) => {*

*if (err) {*

*console.error('Error connecting to the database', err.stack);*

*} else {*

*console.log('Connected to the database at', res.rows[0].now);*

*}*

*});*

*const createTableQuery = `*

*CREATE TABLE IF NOT EXISTS characters (*

*id SERIAL PRIMARY KEY,*

*name VARCHAR(255) NOT NULL,*

*created\_at TIMESTAMP DEFAULT NOW()*

*);*

*`;*

*pool.query(createTableQuery, (err, res) => {*

*if (err) {*

*console.error('Error creating table', err.stack);*

*} else {*

*console.log('Table "characters" is ready');*

*}*

*});*

*// Получить всех персонажей*

*app.get('/characters', async (req, res) => {*

*try {*

*const { rows } = await pool.query('SELECT \* FROM characters ORDER BY id');*

*res.json(rows);*

*} catch (err) {*

*console.error(err);*

*res.status(500).json({ error: 'Internal server error' });*

*}*

*});*

*// Получить одного персонажа по ID*

*app.get('/characters/:id', async (req, res) => {*

*const { id } = req.params;*

*try {*

*const { rows } = await pool.query('SELECT \* FROM characters WHERE id = $1', [id]);*

*if (rows.length === 0) {*

*return res.status(404).json({ error: 'Character not found' });*

*}*

*res.json(rows[0]);*

*} catch (err) {*

*console.error(err);*

*res.status(500).json({ error: 'Internal server error' });*

*}*

*});*

*// Добавить нового персонажа*

*app.post('/characters', async (req, res) => {*

*const { name } = req.body;*

*try {*

*const { rows } = await pool.query(*

*'INSERT INTO characters (name) VALUES ($1) RETURNING \*',*

*[name]*

*);*

*res.status(201).json(rows[0]);*

*} catch (err) {*

*console.error(err);*

*res.status(500).json({ error: 'Internal server error' });*

*}*

*});*

*// Обновить персонажа*

*app.put('/characters/:id', async (req, res) => {*

*const { id } = req.params;*

*const { name } = req.body;*

*try {*

*const { rows } = await pool.query(*

*'UPDATE characters SET name = $1 WHERE id = $2 RETURNING \*',*

*[name, id]*

*);*

*if (rows.length === 0) {*

*return res.status(404).json({ error: 'Character not found' });*

*}*

*res.json(rows[0]);*

*} catch (err) {*

*console.error(err);*

*res.status(500).json({ error: 'Internal server error' });*

*}*

*});*

*// Удалить персонажа*

*app.delete('/characters/:id', async (req, res) => {*

*const { id } = req.params;*

*try {*

*const { rowCount } = await pool.query('DELETE FROM characters WHERE id = $1', [id]);*

*if (rowCount === 0) {*

*return res.status(404).json({ error: 'Character not found' });*

*}*

*res.status(204).send();*

*} catch (err) {*

*console.error(err);*

*res.status(500).json({ error: 'Internal server error' });*

*}*

*});*

*const PORT = process.env.PORT || 5000;*

*const startServer = async () => {*

*try {*

*await pool.query('SELECT NOW()');*

*console.log('Database connected!');*

*app.listen(PORT, () => {*

*console.log(`Server running on port ${PORT}`);*

*});*

*} catch (err) {*

*console.error('Database connection failed:', err.message);*

*setTimeout(startServer, 5000);*

*}*

*};*

*startServer();*

**Клиент**

**Описание функций**

1. **App**

**Назначение:** настраивает маршрутизацию и отображает основные элементы интерфейса.

**Обращение:**

*function App() {*

*return (*

*<Router>*

*<div className="container">*

*<div*

*className="pirate-header"*

*style={{ backgroundImage: `url(${pirateBg})` }}*

*>*

*<h1>Pirates of the Caribbean Sea</h1>*

*</div>*

*<Routes>*

*<Route path="/" element={<CharacterList />} />*

*<Route path="/add" element={<CharacterForm />} />*

*<Route path="/edit/:id" element={<CharacterForm editMode={true} />} />*

*</Routes>*

*</div>*

*</Router>*

*);*

*}*

#### **CharacterList**

**Назначение:** Отображает список персонажей с возможностью удаления и редактирования. (**fetchCharacters** - получает список персонажей с сервера, **deleteCharacter** - удаляет персонажа по ID)

**Обращение:**

*function CharacterList() {*

*const [characters, setCharacters] = useState([]);*

*const fetchCharacters = async () => {*

*try {*

*const response = await axios.get('http://localhost:5000/characters');*

*setCharacters(response.data);*

*} catch (error) {*

*console.error('Error fetching characters:', error);*

*}*

*};*

*const deleteCharacter = async (id) => {*

*try {*

*await axios.delete(`http://localhost:5000/characters/${id}`);*

*fetchCharacters();*

*} catch (error) {*

*console.error('Error deleting character:', error);*

*}*

*};*

*return (*

*<div>*

*<Link to="/add">*

*<button className="add">Add New Character</button>*

*</Link>*

*<table>*

*<thead>*

*<tr>*

*<th>Character Name</th>*

*<th>Actions</th>*

*</tr>*

*</thead>*

*<tbody>*

*{characters.map((character) => (*

*<tr key={character.id}>*

*<td>{character.name}</td>*

*<td>*

*<Link to={`/edit/${character.id}`}>*

*<button className="edit">Edit</button>*

*</Link>*

*<button className="delete" onClick={() => deleteCharacter(character.id)}>*

*Delete*

*</button>*

*</td>*

*</tr>*

*))}*

*</tbody>*

*</table>*

*</div>*

*);*

*}*

#### **CharacterForm**

**Назначение:** Форма для добавления или редактирования персонажа (**handleChange** - обрабатывает изменения в полях формы, **handleSubmit** - отправляет данные на сервер для создания или обновления персонажа.)

**Обращение:**

*function CharacterForm({ editMode }) {*

*const [character, setCharacter] = useState({ name: '' });*

*const handleChange = (e) => {*

*const { name, value } = e.target;*

*setCharacter(prevState => ({ ...prevState, [name]: value }));*

*};*

*const handleSubmit = async (e) => {*

*e.preventDefault();*

*try {*

*if (editMode) {*

*const id = window.location.pathname.split('/')[2];*

*await axios.put(`http://localhost:5000/characters/${id}`, character);*

*} else {*

*await axios.post('http://localhost:5000/characters', character);*

*}*

*window.location.href = '/';*

*} catch (error) {*

*console.error('Error saving character:', error);*

*}*

*};*

*return (*

*<form onSubmit={handleSubmit}>*

*<div className="form-group">*

*<label>Character Name:</label>*

*<input*

*type="text"*

*name="name"*

*value={character.name}*

*onChange={handleChange}*

*required*

*/>*

*</div>*

*<div className="form-actions">*

*<Link to="/">*

*<button type="button">Cancel</button>*

*</Link>*

*<button type="submit" className={editMode ? 'edit' : 'add'}>*

*{editMode ? 'Update Character' : 'Add Character'}*

*</button>*

*</div>*

*</form>*

*);*

*}*

**Код**

* index.js

*import React from 'react';*

*import ReactDOM from 'react-dom/client';*

*import './index.css';*

*import App from './App';*

*import reportWebVitals from './reportWebVitals';*

*const root = ReactDOM.createRoot(document.getElementById('root'));*

*root.render(*

*<React.StrictMode>*

*<App />*

*</React.StrictMode>*

*);*

*reportWebVitals();*

* App.js

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

*import axios from 'axios';*

*import { BrowserRouter as Router, Route, Routes, Link } from 'react-router-dom';*

*import './App.css';*

*import pirateBg from './assets/1.jpg'; // Импорт изображения*

*function App() {*

*return (*

*<Router>*

*<div className="container">*

*<div*

*className="pirate-header"*

*style={{ backgroundImage: `url(${pirateBg})` }}*

*>*

*<h1>Pirates of the Caribbean Sea</h1>*

*</div>*

*<Routes>*

*<Route path="/" element={<CharacterList />} />*

*<Route path="/add" element={<CharacterForm />} />*

*<Route path="/edit/:id" element={<CharacterForm editMode={true} />} />*

*</Routes>*

*</div>*

*</Router>*

*);*

*}*

*function CharacterList() {*

*const [characters, setCharacters] = useState([]);*

*useEffect(() => {*

*fetchCharacters();*

*}, []);*

*const fetchCharacters = async () => {*

*try {*

*const response = await axios.get('http://localhost:5000/characters');*

*setCharacters(response.data);*

*} catch (error) {*

*console.error('Error fetching characters:', error);*

*}*

*};*

*const deleteCharacter = async (id) => {*

*try {*

*await axios.delete(`http://localhost:5000/characters/${id}`);*

*fetchCharacters();*

*} catch (error) {*

*console.error('Error deleting character:', error);*

*}*

*};*

*return (*

*<div>*

*<Link to="/add">*

*<button className="add">Add New Character</button>*

*</Link>*

*<table>*

*<thead>*

*<tr>*

*<th>Character Name</th>*

*<th>Actions</th>*

*</tr>*

*</thead>*

*<tbody>*

*{characters.map((character) => (*

*<tr key={character.id}>*

*<td>{character.name}</td>*

*<td>*

*<Link to={`/edit/${character.id}`}>*

*<button className="edit">Edit</button>*

*</Link>*

*<button className="delete" onClick={() => deleteCharacter(character.id)}>*

*Delete*

*</button>*

*</td>*

*</tr>*

*))}*

*</tbody>*

*</table>*

*</div>*

*);*

*}*

*function CharacterForm({ editMode }) {*

*const [character, setCharacter] = useState({*

*name: ''*

*});*

*useEffect(() => {*

*if (editMode) {*

*const fetchCharacter = async () => {*

*const id = window.location.pathname.split('/')[2];*

*try {*

*const response = await axios.get(`http://localhost:5000/characters/${id}`);*

*setCharacter(response.data);*

*} catch (error) {*

*console.error('Error fetching character:', error);*

*}*

*};*

*fetchCharacter();*

*}*

*}, [editMode]);*

*const handleChange = (e) => {*

*const { name, value } = e.target;*

*setCharacter(prevState => ({*

*...prevState,*

*[name]: value*

*}));*

*};*

*const handleSubmit = async (e) => {*

*e.preventDefault();*

*try {*

*if (editMode) {*

*const id = window.location.pathname.split('/')[2];*

*await axios.put(`http://localhost:5000/characters/${id}`, character);*

*} else {*

*await axios.post('http://localhost:5000/characters', character);*

*}*

*window.location.href = '/';*

*} catch (error) {*

*console.error('Error saving character:', error);*

*}*

*};*

*return (*

*<form onSubmit={handleSubmit}>*

*<div className="form-group">*

*<label>Character Name:</label>*

*<input*

*type="text"*

*name="name"*

*value={character.name}*

*onChange={handleChange}*

*required*

*/>*

*</div>*

*<div className="form-actions">*

*<Link to="/">*

*<button type="button">Cancel</button>*

*</Link>*

*<button type="submit" className={editMode ? 'edit' : 'add'}>*

*{editMode ? 'Update Character' : 'Add Character'}*

*</button>*

*</div>*

*</form>*

*);*

*}*

*export default App;*

# Web-страница

