МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий Кафедра «Электроника и сети ЭВМ»

## Лабораторная работа №2

«Введение в JavaScript. Программное взаимодействие с HTML документами на основе DOM API»

## по дисциплине

«WEB - технологии»

РУКОВОДИТЕЛЬ:

Соменков Ф.В.

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

Скорынин С.С.

(подпись) (фамилия, и.,о.)

22-ПО

(шифр группы)

Работа защищена « »

С оценкой

Нижний Новгород

2024 г

**ВВЕДЕНИЕ**

**Задание**:

Написать приложение News:

* Приложение должно отображать список новостей в виде: Title, Author, Content.
* Так же должна присутствовать страница добавления новости.
* Так же должна присутствовать страница удаления новости.
* Хранилище реализовать в памяти с ограничением в 50 записей.
* Должны быть реализованы кнопки навигации.
* Должно быть реализовано интерактивное взаимодействие с пользователем.

Подключить swagger к веб-сервису(обязательно описать реализованные API в swagger)

1. Реализовать следующие методы или типов запросов HTTP для взаимодействия с сервером:

* GET
* POST
* PUT
* DELETE

1. Для документирования API добавить в проект Swagger UI
2. Продемонстрировать работу сервиса через браузер (запросы, возвращаемы ответ).

**Код**

AddNews.css:

.add-news-form {

    display: flex;

    flex-direction: column;

    align-items: center;

  }

  .add-news-form button {

    background-color: orange;

    color: white;

    padding: 10px 20px;

    border: none;

    border-radius: 5px;

    cursor: pointer;

    transition: background-color 0.3s ease;

    display: flex;

    align-items: center;

    justify-content: center;

  }

  .add-news-form button:hover {

    background-color: darkorange;

  }

AddNews.js:  
import React, { useState } from 'react';

import axios from 'axios';

import './AddNews.css'; // Импортируем CSS файл

const AddNews = () => {

  // Храненим значения полей

  const [title, setTitle] = useState('');

  const [author, setAuthor] = useState('');

  const [content, setContent] = useState('');

  // Обработчик отправки формы

  const handleSubmit = (e) => {

    e.preventDefault();

    axios.post('http://localhost:8080/news', { title, author, content })

      .then(response => {

        console.log('News added:', response.data); // Логи успехи

      })

      .catch(error => {

        console.error('There was an error adding the news!', error); // Логи ошибок

      });

  };

  return (

    <div>

      <h1>Add News</h1>

      <form className="add-news-form" onSubmit={handleSubmit}>

        <div>

          <label>Title:</label>

          <input

            type="text"

            value={title}

            onChange={(e) => setTitle(e.target.value)} // Обновление состояние title при изменении поля

          />

        </div>

        <div>

          <label>Author:</label>

          <input

            type="text"

            value={author}

            onChange={(e) => setAuthor(e.target.value)} // Обновление состояние author при изменении поля

          />

        </div>

        <div>

          <label>Content:</label>

          <textarea

            value={content}

            onChange={(e) => setContent(e.target.value)} // Обновление состояние content при изменении поля

          />

        </div>

        <button type="submit">Add News</button>

      </form>

    </div>

  );

};

export default AddNews;

DeleteNews.css:  
.delete-news-form {

    display: flex;

    flex-direction: column;

    align-items: center;

  }

  .delete-news-form button {

    background-color: orange;

    color: white;

    padding: 10px 20px;

    border: none;

    border-radius: 5px;

    cursor: pointer;

    transition: background-color 0.3s ease;

    display: flex;

    align-items: center;

    justify-content: center;

  }

  .delete-news-form button:hover {

    background-color: darkorange;

  }

DeleteNews.js:  
import React, { useState } from 'react';

import axios from 'axios';

import './DeleteNews.css'; // Импортируем CSS файл

const DeleteNews = () => {

  // ID новости, которую нужно удалить

  const [id, setId] = useState('');

  const handleSubmit = (e) => {

    e.preventDefault();

    axios.delete(`http://localhost:8080/news/${id}`)

      .then(response => {

        console.log('News deleted:', response.data); // Логи успехов

      })

      .catch(error => {

        console.error('There was an error deleting the news!', error); // Логи ошибок

      });

  };

  return (

    <div>

      <h1>Delete News</h1>

      <form className="delete-news-form" onSubmit={handleSubmit}>

        <div>

          <label>ID:</label>

          <input

            type="text"

            value={id}

            onChange={(e) => setId(e.target.value)}

          />

        </div>

        <button type="submit">Delete News</button>

      </form>

    </div>

  );

};

export default DeleteNews;

NewsList.css:

.news-id {

    font-size: 0.8em;

    color: gray;

    margin-right: 10px;

  }

NewsList.js:  
import React, { useEffect, useState } from 'react';

import axios from 'axios';

import './NewsList.css'; // Импортируем CSS файл

const NewsList = () => {

  // Хранение новостей

  const [news, setNews] = useState([]);

  useEffect(() => {

    axios.get('http://localhost:8080/news')

      .then(response => {

        setNews(response.data); // Обновление новостей

      })

      .catch(error => {

        console.error('There was an error fetching the news!', error); // Логи ошибок

      });

  }, []);

  return (

    <div>

      <h1>News List</h1>

      <ul>

        {news.map(item => (

          <li key={item.id}>

            <span className="news-id">ID: {item.id}</span>

            <h2>{item.title}</h2>

            <p>Author: {item.author}</p>

            <p>{item.content}</p>

          </li>

        ))}

      </ul>

    </div>

  );

};

export default NewsList;

App.css:  
.App {

  text-align: center;

}

.App-logo {

  height: 40vmin;

  pointer-events: none;

}

@media (prefers-reduced-motion: no-preference) {

  .App-logo {

    animation: App-logo-spin infinite 20s linear;

  }

}

.App-header {

  background-color: #282c34;

  min-height: 100vh;

  display: flex;

  flex-direction: column;

  align-items: center;

  justify-content: center;

  font-size: calc(10px + 2vmin);

  color: white;

}

.App-link {

  color: #61dafb;

}

@keyframes App-logo-spin {

  from {

    transform: rotate(0deg);

  }

  to {

    transform: rotate(360deg);

  }

}

/\* Styles for navigation links \*/

nav ul {

  list-style-type: none;

  padding: 0;

}

nav ul li {

  display: inline;

  margin: 0 10px;

}

nav ul li a {

  background-color: orange;

  color: white;

  padding: 10px 20px;

  text-decoration: none;

  border-radius: 5px;

  transition: background-color 0.3s ease;

}

nav ul li a:hover {

  background-color: darkorange;

}

App.js:  
import React from 'react';

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

import NewsList from './components/NewsList';

import AddNews from './components/AddNews';

import DeleteNews from './components/DeleteNews';

import './App.css'; // Импортируем CSS файл

function App() {

  return (

    <Router>

      <div>

        <nav>

          <ul>

            <li><Link to="/">News List</Link></li> {/\* Ссылка на новости \*/}

            <li><Link to="/add">Add News</Link></li> {/\* Ссылка на добавления новости \*/}

            <li><Link to="/delete">Delete News</Link></li> {/\* Ссылка на удаления новости \*/}

          </ul>

        </nav>

        <Routes>

          <Route path="/" element={<NewsList />} /> {/\* Маршрут для отображения новостей \*/}

          <Route path="/add" element={<AddNews />} /> {/\* Маршрут для отображения добавления новости \*/}

          <Route path="/delete" element={<DeleteNews />} /> {/\* Маршрут для отображения удаления новости \*/}

        </Routes>

      </div>

    </Router>

  );

}

export default App;

news.go:

package models

type News struct {

    ID      int    `json:"id"`

    Title   string `json:"title"`

    Author  string `json:"author"`

    Content string `json:"content"`

}

news.go:  
package news

import (

    "github.com/gin-gonic/gin"

    "news-service/models"

    "news-service/service"

    "strconv"

)

// Управление

type NewsController struct {

    service \*service.NewsService

}

// Создание NewsController

func NewNewsController(service \*service.NewsService) \*NewsController {

    return &NewsController{service: service}

}

// Получение всех новостей.

func (c \*NewsController) GetNews(ctx \*gin.Context) {

    newsList := c.service.GetAll()

    ctx.JSON(200, newsList)

}

// Создание новой новости

func (c \*NewsController) CreateNews(ctx \*gin.Context) {

    var newNews models.News // Переменная для хранения новости

    if err := ctx.ShouldBindJSON(&newNews); err != nil {

        ctx.JSON(400, gin.H{"error": err.Error()})

        return

    }

    if newNews.Title == "" || newNews.Author == "" || newNews.Content == "" { // Проверка заполнености

        ctx.JSON(400, gin.H{"error": "Title, Author, and Content are required"})

        return

    }

    createdNews := c.service.Create(newNews)

    ctx.JSON(201, createdNews)

}

// Обновления новости

func (c \*NewsController) UpdateNews(ctx \*gin.Context) {

    idStr := ctx.Param("id") // Получаем ID

    id, err := strconv.Atoi(idStr)

    if err != nil {

        ctx.JSON(400, gin.H{"error": "Invalid ID"})

        return

    }

    var updatedNews models.News // Переменная для хранения обновленной новости

    if err := ctx.ShouldBindJSON(&updatedNews); err != nil {

        ctx.JSON(400, gin.H{"error": err.Error()})

        return

    }

    if updatedNews.Title == "" || updatedNews.Author == "" || updatedNews.Content == "" {

        ctx.JSON(400, gin.H{"error": "Title, Author, and Content are required"})

        return

    }

    news, err := c.service.Update(id, updatedNews) // Обновляем новость

    if err != nil {

        ctx.JSON(404, gin.H{"error": err.Error()})

        return

    }

    ctx.JSON(200, news)

}

// Удаления новости.

func (c \*NewsController) DeleteNews(ctx \*gin.Context) {

    idStr := ctx.Param("id") // Получаем ID

    id, err := strconv.Atoi(idStr)

    if err != nil {

        ctx.JSON(400, gin.H{"error": "Invalid ID"})

        return

    }

    if err := c.service.Delete(id); err != nil { // Удаление новости

        ctx.JSON(404, gin.H{"error": err.Error()})

        return

    }

    ctx.JSON(200, gin.H{"message": "News deleted"})

}

news\_repository.go:

package repository

import (

    "errors"

    "news-service/models"

    "sync"

)

// ErrNewsNotFound ошибка, если новость не найдена

var ErrNewsNotFound = errors.New("news not found")

// Структура репозитория

type NewsRepository struct {

    newsList  []models.News // Список

    idCounter int           // Счетчик

    mutex     \*sync.Mutex   // Мьютекс

}

// Создание нового экземпляря NewsRepository

func NewNewsRepository() \*NewsRepository {

    return &NewsRepository{

        newsList:  make([]models.News, 0, 50),

        idCounter: 1,

        mutex:     &sync.Mutex{},

    }

}

// Все новости из репозитория.

func (r \*NewsRepository) GetAll() []models.News {

    r.mutex.Lock()

    defer r.mutex.Unlock()

    return r.newsList

}

// Добавляет новость в репозиторий

func (r \*NewsRepository) Create(news models.News) models.News {

    r.mutex.Lock()

    defer r.mutex.Unlock()

    news.ID = r.idCounter

    r.idCounter++

    r.newsList = append(r.newsList, news)

    return news

}

// Обновляет новость в репозитории

func (r \*NewsRepository) Update(id int, news models.News) (models.News, error) {

    r.mutex.Lock()

    defer r.mutex.Unlock()

    for i, item := range r.newsList {

        if item.ID == id {

            r.newsList[i] = news

            return news, nil

        }

    }

    return models.News{}, ErrNewsNotFound

}

// Удаляет новость из репозитория по ID

func (r \*NewsRepository) Delete(id int) error {

    r.mutex.Lock()

    defer r.mutex.Unlock()

    for i, item := range r.newsList {

        if item.ID == id {

            r.newsList = append(r.newsList[:i], r.newsList[i+1:]...)

            return nil

        }

    }

    return ErrNewsNotFound

}

news\_service.go:

package service

import (

    "news-service/models"

    "news-service/repository"

)

// Структура управления новостями

type NewsService struct {

    repo \*repository.NewsRepository // Репозиторий для доступа к новостям

}

// Создает новый экземпляр NewsService

func NewNewsService(repo \*repository.NewsRepository) \*NewsService {

    return &NewsService{repo: repo}

}

// Возвращает все новости из репозитория

func (s \*NewsService) GetAll() []models.News {

    return s.repo.GetAll() // Получение всех новостей

}

// Добавляет новую новость в репозиторий

func (s \*NewsService) Create(news models.News) models.News {

    return s.repo.Create(news)

}

// Обновляет существующую новость в репозитории

func (s \*NewsService) Update(id int, news models.News) (models.News, error) {

    return s.repo.Update(id, news)

}

// Удаляет новость из репозитория по ID

func (s \*NewsService) Delete(id int) error {

    return s.repo.Delete(id)

}

main.go:  
package main

import (

    \_ "news-service/docs"

    "news-service/news"

    "news-service/repository"

    "news-service/service"

    "github.com/gin-contrib/cors"

    "github.com/gin-gonic/gin"

    swaggerFiles "github.com/swaggo/files"

    ginSwagger "github.com/swaggo/gin-swagger"

)

func main() {

    r := gin.Default()

    // Настройка CORS

    config := cors.DefaultConfig()

    config.AllowOrigins = []string{"http://localhost:3000"}

    config.AllowMethods = []string{"GET", "POST", "PUT", "DELETE", "OPTIONS"}

    config.AllowHeaders = []string{"Origin", "Content-Type", "Accept"}

    r.Use(cors.New(config))

    // Настройка Swagger

    r.GET("/swagger/\*any", ginSwagger.WrapHandler(swaggerFiles.Handler))

    // Инициализация репозитория и сервиса

    newsRepo := repository.NewNewsRepository()

    newsService := service.NewNewsService(newsRepo)

    newsController := news.NewNewsController(newsService)

    // Определение маршрутов

    r.GET("/news", newsController.GetNews)

    r.POST("/news", newsController.CreateNews)

    r.PUT("/news/:id", newsController.UpdateNews)

    r.DELETE("/news/:id", newsController.DeleteNews)

    // Запуск сервера

    r.Run(":8080")

}

Запуск:

D:\ИРИТ\ВЕБ-технологии\VEB\_lab2\_2\news-service>go run main.go  
D:\ИРИТ\ВЕБ-технологии\VEB\_lab2\_2\news-frontend> npm start

http://localhost:8080/swagger/index.html

**Пример:**



















