|  |  |
| --- | --- |
| **Российский университет транспорта (МИИТ)**  **Институт транспортной техники и систем управления**  **Кафедра «Управление и защита информации»** | |
| **Отчет**  **по практическому заданию**  **по теме «Разработка текстовой базы данных и веб-**  **оболочки на тему: «Издательство». Задание 8»**  **по дисциплине**  **«Web-программирование»** | |
|  | Выполнил:  студент группы ТКИ-541,  Петросян А. А.  Проверил:  доцент кафедры УиЗИ к.т.н.,  Сафронов А. И. |
| г. Москва - 2023 | |

ОГЛАВЛЕНИЕ

[Цель работы 3](#_Toc150296207)

[Формулировка задачи 3](#_Toc150296208)

[Описание предметной области и сформулированных к ней задач 3](#_Toc150296209)

[*ERD*-диаграмма данных 3](#_Toc150296210)

[Таблицы соответствия переменных 4](#_Toc150296211)

[Архитектура системы 4](#_Toc150296212)

[1 Содержательная часть 5](#_Toc150296213)

[1.1 Server.js 5](#_Toc150296214)

[1.2 Main.ts 9](#_Toc150296215)

[1.3 Index.ts 9](#_Toc150296216)

[1.4 App.vue 10](#_Toc150296217)

[1.5 Abaut.vue 11](#_Toc150296218)

[1.6 logo.svg 13](#_Toc150296219)

[1.7 Authors.vue 13](#_Toc150296220)

[1.8 Works.vue 15](#_Toc150296221)

[1.9 AuthorWorks.vue 16](#_Toc150296222)

[1.10 Скриншоты страниц 21](#_Toc150296223)

[1.11 Сеть Петри 26](#_Toc150296224)

[Вывод 27](#_Toc150296225)

Цель работы

Разработка веб-приложения для информационного обслуживания сотрудников издательства

Формулировка задачи

Вариант №14

* Создание интерфейса для отображения списка трудов и авторов
* Реализация функционала для просмотра и редактирования списка и отношений

Описание предметной области и сформулированных к ней задач

Предметная область: Издательство.

Задачи: информационного обслуживания издательства

Таблица «труды (works)» содержит информацию о трудах, название, тип (журнал, книга, методическое пособие) и год издания.

Таблица «Авторы (authors)» содержит информацию об Авторах. Каждый автор имеет уникальный идентификатор, имя, Фамилию и электронную почту.

Таблица «авторы и труды (authorworks)» содержит информацию об отношении автора к труду.

*ERD*-диаграмма данных

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Рисунок 1 - ERD-диаграмма

Таблицы соответствия переменных

**Works:**

|  |  |
| --- | --- |
| Название переменной в коде | Описание переменной |
| Work\_id | Уникальный идентификатор труда |
| title | Наименование труда |
| work\_type | Тип труда |
| publication\_year | Год публикации |

**Authors:**

|  |  |
| --- | --- |
| Название переменной в коде | Описание переменной |
| author\_id | Уникальный идентификатор Автора |
| first\_name | Имя Автора |
| last\_name | Фамилия Автора |
| email | Электронная почта Автора |

**Authorsworks:**

|  |  |
| --- | --- |
| Название переменной в коде | Описание переменной |
| author\_id | Уникальный идентификатор Автора |
| work\_id | Уникальный идентификатор труда |
| role | Роль автора к этому труду |

Архитектура системы

Изображение выглядит как текст, снимок экрана, Цвет электрик, Шрифт

Автоматически созданное описание

Рисунок 2 - Архитектура системы

# Содержательная часть

## Server.js

const express = require('express');

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

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

const cors = require('cors');

const app = express();

const port = 3000;

const pool = new Pool({

  user: 'postgres',

  host: 'localhost',

  database: 'arutdb',

  password: '123',

  port: 5432,

});

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended: true }));

app.use(cors());

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

  res.send('Подключился молодец!');

});

// Получение списка авторов

app.get('/api/authors', async (req, res) => {

  try {

    const result = await pool.query('SELECT \* FROM authors');

    res.status(200).json(result.rows);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

// Получение списка работ

app.get('/api/works', async (req, res) => {

  try {

    const result = await pool.query('SELECT \* FROM works');

    res.status(200).json(result.rows);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

// Получения работ определенного автора

app.get('/api/author-works/:authorId', async (req, res) => {

  const authorId = req.params.authorId;

  try {

    const result = await pool.query(`

      SELECT works.\*

      FROM works

      INNER JOIN authorsworks ON works.work\_id = authorsworks.work\_id

      INNER JOIN authors ON authors.author\_id = authorsworks.author\_id

      WHERE authors.author\_id = $1

    `, [authorId]);

    res.status(200).json(result.rows);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

// Получения авторов и их роли для каждой работы

app.get('/api/work-authors/:workId', async (req, res) => {

  const workId = req.params.workId;

  try {

    const result = await pool.query(`

      SELECT authors.\*, authorsworks.role

      FROM authors

      INNER JOIN authorsworks ON authors.author\_id = authorsworks.author\_id

      WHERE authorsworks.work\_id = $1

    `, [workId]);

    res.status(200).json(result.rows);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

// Добавление автора

app.post('/api/authors', async (req, res) => {

  const { first\_name, last\_name, email } = req.body;

  try {

    const result = await pool.query('INSERT INTO authors (first\_name, last\_name, email) VALUES ($1, $2, $3) RETURNING \*', [first\_name, last\_name, email]);

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

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

// Добавление работы

app.post('/api/works', async (req, res) => {

  const { title, work\_type } = req.body;

  try {

    const result = await pool.query('INSERT INTO works (title, work\_type) VALUES ($1, $2) RETURNING \*', [title, work\_type]);

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

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

// Удаление автора

app.delete('/api/authors/:authorId', async (req, res) => {

  const authorId = req.params.authorId;

  try {

    await pool.query('DELETE FROM authors WHERE author\_id = $1', [authorId]);

    // Обновление нумерации авторов

    await pool.query('SELECT setval(\'authors\_author\_id\_seq\', (SELECT COALESCE(MAX(author\_id), 0) + 1 FROM authors))');

    res.status(204).send();

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

// Удаление работы

app.delete('/api/works/:workId', async (req, res) => {

  const workId = req.params.workId;

  try {

    await pool.query('DELETE FROM works WHERE work\_id = $1', [workId]);

    // Обновление нумерации работ

    await pool.query('SELECT setval(\'works\_work\_id\_seq\', (SELECT COALESCE(MAX(work\_id), 1) FROM works))');

    res.status(204).send();

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

// Связывание автора с работой и установления статуса (автор или соавтор)

app.post('/api/assign-author-work', async (req, res) => {

  const { authorId, workId, role } = req.body;

  try {

    await pool.query('INSERT INTO authorsworks (author\_id, work\_id, role) VALUES ($1, $2, $3)', [authorId, workId, role]);

    res.status(201).send();

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

app.post('/api/assign-author-work', async (req, res) => {

  const { workId, authorId, role } = req.body;

  try {

    const result = await pool.query('INSERT INTO authorsworks (work\_id, author\_id, role) VALUES ($1, $2, $3) RETURNING \*', [workId, authorId, role]);

    res.status(200).json(result.rows[0]);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

// Обновление роли автора в связи

app.put('/api/assign-author-work/:workId/:authorId/:role', async (req, res) => {

  const { workId, authorId, role } = req.params;

  try {

    // Проверка наличия связи

    const result = await pool.query('SELECT \* FROM authorsworks WHERE work\_id = $1 AND author\_id = $2', [workId, authorId]);

    if (result.rows.length === 0) {

      return res.status(404).json({ error: 'Связь автора и работы не найдена.' });

    }

    // Обновляем роль в связи

    await pool.query('UPDATE authorsworks SET role = $1 WHERE work\_id = $2 AND author\_id = $3', [role, workId, authorId]);

    res.status(200).json({ message: 'Роль автора обновлена.' });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

// Удаление связи между автором и работой

app.delete('/api/assign-author-work/:workId/:authorId', async (req, res) => {

  const workId = req.params.workId;

  const authorId = req.params.authorId;

  try {

    // Проверка наличия связи

    const result = await pool.query('SELECT \* FROM authorsworks WHERE work\_id = $1 AND author\_id = $2', [workId, authorId]);

    if (result.rows.length === 0) {

      return res.status(404).json({ error: 'Связь автора и работы не найдена.' });

    }

    // Удаление связь

    await pool.query('DELETE FROM authorsworks WHERE work\_id = $1 AND author\_id = $2', [workId, authorId]);

    res.status(204).send();

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

app.listen(port, () => {

  console.log(`Server is running on http://localhost:${port}`);

});

## Main.ts

import './assets/main.css'

import { createApp } from 'vue'

import App from './App.vue'

import router from './router'

const app = createApp(App)

app.use(router)

app.mount('#app')

## Index.ts

import { createRouter, createWebHistory } from 'vue-router';

import HomeView from '../views/HomeView.vue';

import AboutView from '../views/AboutView.vue';

import AuthorsView from '../views/AuthorsView.vue';

import WorksView from '../views/WorksView.vue';

import AuthorWorksView from '../views/AuthorWorksView.vue';

const routes = [

  {

    path: '/',

    name: 'home',

    component: HomeView

  },

  {

    path: '/about',

    name: 'about',

    component: AboutView

  },

  {

    path: '/authors',

    name: 'authors',

    component: AuthorsView

  },

  {

    path: '/works',

    name: 'works',

    component: WorksView

  },

  {

    path: '/authorworks',

    name: 'authorworks',

    component: AuthorWorksView

  },

  // Обработка несуществующих маршрутов:

  {

    path: '/:pathMatch(.\*)\*',

    redirect: '/'

  }

];

const router = createRouter({

  history: createWebHistory(import.meta.env.BASE\_URL),

  routes

});

export default router;

## App.vue

<script setup lang="ts">

import { RouterLink, RouterView } from 'vue-router'

import HelloWorld from './components/HelloWorld.vue'

</script>

<template>

  <header>

    <img alt="Vue logo" class="logo" src="@/assets/logo.svg" width="125" height="125" />

    <div>

      <nav>

        <router-link to="/authors">Авторы</router-link>

        <router-link to="/works">Труды</router-link>

        <router-link to="/authorworks">Отчётность</router-link>

      </nav>

      <router-view></router-view>

    </div>

    <div class="wrapper">

      <HelloWorld msg="Издательство!" />

      <nav>

        <RouterLink to="/">Home</RouterLink>

        <RouterLink to="/about">About</RouterLink>

      </nav>

    </div>

  </header>

</template>

<style scoped>

nav {

  display: flex;

  gap: 10px;

  margin-bottom: 20px;

}

router-link {

  padding: 5px 10px;

  border: 1px solid #ccc;

  text-decoration: none;

  color: black;

}

router-link:hover {

  background-color: #f5f5f5;

}

</style>

## About.vue

<template>

  <div class="about">

    <h1>Если не трудно оцените старания</h1>

    <div class="rating">

      <p>Оцените эту страницу:</p>

      <button @click="like">Понравилось</button>

      <button @click="dislike">Не понравилось</button>

      <p>Понравилось: {{ likes }}</p>

      <p>Не понравилось: {{ dislikes }}</p>

    </div>

  </div>

</template>

<script>

export default {

  data() {

    return {

      likes: 0,

      dislikes: 0,

    };

  },

  methods: {

    like() {

      this.likes++;

    },

    dislike() {

      this.dislikes++;

    },

  },

};

</script>

<style>

@media (min-width: 1024px) {

  .about {

    min-height: 10vh;

    display: flex;

    align-items: center;

  }

  .rating {

    margin-top: 20px;

    text-align: center;

  }

  .rating button {

    padding: 10px 20px;

    margin: 10px;

    background-color: #4c90af;

    color: white;

    border: none;

    cursor: pointer;

  }

}

</style>

## logo.svg

Немного изменил логотип

<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 200 200">

  <circle cx="100" cy="100" r="90" fill="#3498db" />

  <text x="55" y="120" font-size="80" fill="bleck">AP</text>

</svg>

## Authors.vue

Компонент для отображения авторов и их работ

<template>

  <div>

    <h2>Авторы</h2>

    <table>

      <tr v-for="author in authors" :key="author.author\_id">

        <td>{{ author.first\_name }} {{ author.last\_name }}</td>

        <td><button @click="showAuthorWorks(author.author\_id)">Показать работы</button></td>

      </tr>

    </table>

    <div v-if="selectedAuthorWorks && selectedAuthorWorks.length">

      <h3>Работы автора {{ selectedAuthorName }}</h3>

      <table>

        <tr v-for="work in selectedAuthorWorks" :key="work.work\_id">

          <td>{{ work.title }} ({{ work.work\_type }})</td>

        </tr>

      </table>

    </div>

  </div>

</template>

<script>

import axios from 'axios';

export default {

  name: 'AuthorsView',

  data() {

    return {

      authors: [],

      selectedAuthorWorks: [],

      selectedAuthorName: null,

    };

  },

  methods: {

  // Работы автора

  showAuthorWorks(authorId) {

    // Находим полное имя автора для отображения

    const author = this.authors.find(author => author.author\_id === authorId);

    this.selectedAuthorName = author.first\_name + ' ' + author.last\_name;

    // Получения информации об авторе

    axios.get(`http://localhost:3000/api/author/${authorId}`)

      .then(response => {

        // Обновление

        this.selectedAuthor = response.data;

      })

      .catch(error => {

        console.error("Ошибка при загрузке информации об авторе:", error);

      });

    // Получения работ автора

    axios.get(`http://localhost:3000/api/author-works/${authorId}`)

      .then(response => {

        this.selectedAuthorWorks = response.data;

      })

      .catch(error => {

        console.error("Ошибка при загрузке работ автора:", error);

      });

  },

},

data() {

  return {

    authors: [],

    selectedAuthorWorks: [],

    selectedAuthorName: '',

    selectedAuthor: null,

  };

},

  mounted() {

    axios.get('http://localhost:3000/api/authors')

      .then(response => {

        this.authors = response.data;

      })

      .catch(error => {

        console.error("Ошибка при загрузке данных об авторах:", error);

      });

  },

};

</script>

<style scoped>

</style>

## Works.vue

Компонент для отображения трудов и авторов.

<template>

  <div>

    <h2>Труды</h2>

    <table>

      <tr v-for="work in works" :key="work.work\_id">

        <td>

          {{ work.title }} ({{ work.work\_type }})

        </td>

        <td>

          <button @click="showWorkAuthors(work.work\_id)">Показать авторов</button>

        </td>

        <tr v-if="workAuthors[work.work\_id]">

          <td colspan="2">

            <ul>

              <li v-for="author in workAuthors[work.work\_id]" :key="author.author\_id">

                {{ author.first\_name }} {{ author.last\_name }} ({{ author.role }})

              </li>

            </ul>

          </td>

        </tr>

      </tr>

    </table>

  </div>

</template>

<script>

import axios from 'axios';

export default {

  name: 'WorksView',

  data() {

    return {

      works: [],

      workAuthors: {},

    };

  },

  methods: {

    // Получаем авторов и соавторов для работы

    showWorkAuthors(workId) {

      axios.get(`http://localhost:3000/api/work-authors/${workId}`)

        .then(response => {

          this.workAuthors[workId] = response.data;

        })

        .catch(error => {

          console.error("Ошибка при загрузке авторов работы:", error);

        });

    },

  },

  mounted() {

    axios.get('http://localhost:3000/api/works')

      .then(response => {

        this.works = response.data;

      })

      .catch(error => {

        console.error("Ошибка при загрузке данных о работах:", error);

      });

  },

};

</script>

<style scoped>

</style>

## AuthorWorks.vue

<template>

  <div>

    <h2>Авторы и Работы</h2>

    <form @submit.prevent="addAuthor" class="form">

  <div class="form-group">

    <label for="authorFirstName">Имя:</label>

    <input type="text" id="authorFirstName" v-model="newAuthorFirstName" required>

  </div>

  <div class="form-group">

    <label for="authorLastName">Фамилия:</label>

    <input type="text" id="authorLastName" v-model="newAuthorLastName" required>

  </div>

  <div class="form-group">

    <label for="authorEmail">Почта:</label>

    <input type="email" id="authorEmail" v-model="newAuthorEmail" required>

  </div>

  <button type="submit" class="form-button">Добавить автора</button>

</form>

<form @submit.prevent="addWork" class="form">

  <div class="form-group">

    <label for="workTitle">Название работы:</label>

    <input type="text" id="workTitle" v-model="newWorkTitle" required>

  </div>

  <div class="form-group">

    <label for="workType">Тип работы:</label>

    <input type="text" id="workType" v-model="newWorkType" required>

  </div>

  <div class="form-group">

    <label for="workYear">Год издания:</label>

    <input type="number" id="workYear" v-model="newWorkYear" required>

  </div>

  <button type="submit" class="form-button">Добавить работу</button>

</form>

    <table>

      <tr>

        <h3>Список авторов</h3>

      </tr>

      <tr v-for="author in authors" :key="author.author\_id">

        <td>

          {{ author.first\_name }} {{ author.last\_name }} ({{ author.email }})

        </td>

        <td>

          <button @click="deleteAuthor(author.author\_id)">Удалить</button>

        </td>

      </tr>

    </table>

    <h3>Список работ</h3>

    <ul class="list">

      <li v-for="work in works" :key="work.work\_id">

        <div class="list-item">

          {{ work.title }} ({{ work.work\_type }}), {{ work.publication\_year }}

          <div class="work-actions">

            <button @click="deleteWork(work.work\_id)" class="list-button">Удалить</button>

            <button @click="selectWork(work)" class="list-button">Выбрать</button>

          </div>

          <div v-if="selectedWork === work" class="work-actions">

            <select v-model="selectedAuthorId" class="form-select">

              <option value="">Выберите автора</option>

              <option v-for="author in authors" :key="author.author\_id" :value="author.author\_id">

                {{ author.first\_name }} {{ author.last\_name }}

              </option>

            </select>

            <button @click="assignAuthorWork(work.work\_id, selectedAuthorId, 'Основной автор')" class="list-button">Установить автором</button>

            <button @click="assignAuthorWork(work.work\_id, selectedAuthorId, 'Соавтор')" class="list-button">Установить соавтором</button>

            <button @click="deleteAuthorWork(work.work\_id, selectedAuthorId)" class="list-button">Удалить связь</button>

          </div>

        </div>

      </li>

    </ul>

  </div>

</template>

<script>

import axios from 'axios';

export default {

  name: 'AuthorWorksView',

  data() {

    return {

      newAuthorFirstName: '',

      newAuthorLastName: '',

      newAuthorEmail: '',

      newWorkTitle: '',

      newWorkType: '',

      newWorkYear: null,

      authors: [],

      works: [],

      selectedWork: null,

      selectedAuthorId: '',

    };

  },

  methods: {

    addAuthor() {

      axios.post('http://localhost:3000/api/authors', {

        first\_name: this.newAuthorFirstName,

        last\_name: this.newAuthorLastName,

        email: this.newAuthorEmail,

      })

      .then(response => {

        this.authors.push(response.data);

        this.newAuthorFirstName = '';

        this.newAuthorLastName = '';

        this.newAuthorEmail = '';

      })

      .catch(error => {

        console.error("Ошибка при добавлении автора:", error);

      });

    },

    addWork() {

      axios.post('http://localhost:3000/api/works', {

        title: this.newWorkTitle,

        work\_type: this.newWorkType,

        publication\_year: this.newWorkYear,

      })

      .then(response => {

        this.works.push(response.data);

        this.newWorkTitle = '';

        this.newWorkType = '';

        this.newWorkYear = null;

      })

      .catch(error => {

        console.error("Ошибка при добавлении работы:", error);

      });

    },

    deleteAuthor(authorId) {

      axios.delete(`http://localhost:3000/api/authors/${authorId}`)

      .then(() => {

        this.authors = this.authors.filter(author => author.author\_id !== authorId);

      })

      .catch(error => {

        console.error("Ошибка при удалении автора:", error);

      });

    },

    deleteWork(workId) {

      axios.delete(`http://localhost:3000/api/works/${workId}`)

      .then(() => {

        this.works = this.works.filter(work => work.work\_id !== workId);

      })

      .catch(error => {

        console.error("Ошибка при удалении работы:", error);

      });

    },

    deleteAuthorWork(workId, authorId) {

      axios.delete(`http://localhost:3000/api/assign-author-work/${workId}/${authorId}`)

        .then(() => {

        })

        .catch(error => {

          console.error("Ошибка при удалении связи автора и работы:", error);

        });

    },

    assignAuthorWork(workId, authorId, role) {

  axios.post('http://localhost:3000/api/assign-author-work', {

    workId,

    authorId,

    role

  })

    .then(response => {

    })

    .catch(error => {

      console.error("Ошибка при установлении связи автора и работы:", error);

    });

},

 selectWork(work) {

      this.selectedWork = work;

    },

  },

  mounted() {

    axios.get('http://localhost:3000/api/authors')

      .then(response => {

        this.authors = response.data;

      })

      .catch(error => {

        console.error("Ошибка при загрузке данных об авторах:", error);

      });

    axios.get('http://localhost:3000/api/works')

      .then(response => {

        this.works = response.data;

      })

      .catch(error => {

        console.error("Ошибка при загрузке данных о работах:", error);

      });

  },

};

</script>

<style scoped>

</style>

## Скриншоты страниц

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3 – Страница Home

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рисунок 4 – Страница About

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рисунок 5 - Страница [Авторы](http://localhost:5173/medical-workers)

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 6 - Страница Труды

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

Рисунок 7 - Страница [Отчётность](http://localhost:5173/wards)

|  |  |
| --- | --- |
|  |  |
| Рисунок 8 – Добавление Автора Михаил Игнатов | |
|  |  |
| Рисунок 9 – Добавим Журнал 1998 года «Истории» | |
|  |  |
| Рисунок 10 – Обозначим что Михаил автор этого журнала | |
| Рисунок 11 – посмотри на эту работу во вкладке Труды | |
| Рисунок 12 – Удалим связь | | |
|  | |
|  | |  |
| Рисунок 13 – Удаление Автора Михаил Игнатов | | |
|  | |  |
| Рисунок 14 – Удаление журнала 1998 года «Истории» | | |

## Сеть Петри

Изображение выглядит как черно-белый, белый, диаграмма, дизайн

Автоматически созданное описание

Рисунок 15 - Сеть Петри

Вывод

В ходе разработки и реализации системы управления данными издательства были изучены и применены ключевые принципы и технологии веб-разработки. Проект включал в себя создание структурированной базы данных, серверной части на Node.js и клиентской части на Vue.js. Успешно разработана ERD, отражающая структуру данных издательства, что позволило эффективно организовать хранение информации. Реализована интеграция между серверной и клиентской частями, что обеспечивает гладкое и быстрое взаимодействие системы в целом.