

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА  
(САМАРСКИЙ УНИВЕРСИТЕТ)»

Институт Информатики и кибернетики   
Кафедра Программных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**  
  
к лабораторной работе №6 по дисциплине «Прикладные информационные системы»

Обучающийся группы 6132-020402D \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дьяконов А.В.

Обучающийся группы 6132-020402D \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лазарев М.Ю.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лобанков А.А.

Самара 2024

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc182656069)

[2 Результаты работы 4](#_Toc182656070)

[2.1 Результаты задания 4](#_Toc182656071)

[ПРИЛОЖЕНИЕ А Листинг программы 11](#_Toc182656079)

1. Постановка задачи

В процессе выполнения заданий ознакомиться с созданием приложений на Angular.

1. Результаты работы
   1. Результаты задания

Реализовать упрощённую версию таск-трекера. В трекер можно добавлять задачи.

Каждая задача содержит следующие поля для заполнения:

• Номер (генерируется автоматически)

• Тип (bug или task)

• Приоритет (critical/high/medium/low)

• Статус

• Заголовок

• Описание

• Исполнитель

• Создатель

• Дата и время создания (генерируется автоматически)

• Дата и время последнего изменения (генерируется автоматически)

Продумайте страницы с отображением списка задач, наличие фильтра по исполнителю, приоритету и типу для задач (фильтр работает по AND), страницы с созданием и обновлением задачи. Список задач по умолчанию должен сортироваться по дате последнего изменения.

1. Реализуем сервер json-server с базой данных сотрудников и задач db.json. Открываем терминал и прописываем npm install json-server. Редактируем package.json и создаем db.json (Приложение А). Для запуска вводим npm run start в терминале (Рис.1)(Рис.2).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок 1 – Терминал json-server

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 2 – Json-server в браузере

1. Создаём Angular приложение. Открываем терминал и вводим ng new task-tracker. Открываем терминал в созданной папке приложения и вводим ng generate component components/navigation, ng generate component components/task-list, ng generate component components/task-details, ng generate component components/employee-list и ng generate component components/task-form для создания шаблонов страниц со списком задач, навигации, списком сотрудников, формы ввода задачи и формы ввода сотрудника. Создаём шаблоны сервисов для работы с сервером, вводим в терминал ng generate service services/task и ng generate service services/employee (Приложение А).
2. Редактируем app.routes.ts и app.component.ts. Прописываем роутинг нашего приложения и основные компоненты, а также реализуем страницу с навигацией в app.components.html (Приложение А). Запускаем терминал в папке нашего приложения, вводим ng serve и переходим в браузере по адресу localhost:4200 для отслеживания изменений нашего приложения.
3. Редактируем файлы сервисов сотрудников и задач. В обоих сервисах добавляем связь с нашим сервером посредством HttpClient из Angular и указываем URL нашего сервера, а также URL для каждого метода. В employee.service.ts реализуем добавление нового сотрудника addEmployee(), удаление сотрудника deleteEmployee(), обновление данных сотрудника updateEmployee(), вывод сотрудников getEmployees() и вывод данных конкретного сотрудника по ID getEmployeeById(). В task.service.ts реализуем интерфейс для детализации задачи Task, вывод задач getTasks(), вывод конкретной задачи по ID getTaskById(), добавление новой задачи addTask(), обновление задачи updateTask() и удаление задачи deleteTask(), при этом создание и обновление задач реализуются с актуализацией времени на текущие дату и время (Приложение А).
4. Редактируем компонент навигации для отображения простой навигации по всем страницам, в данном случае актуальные страницы «Сотрудники» и «Задачи». В ts-файле реализуем навигацию к компонентам сотрудников navigateToEmployees() и задач navigateToTasks(). В html-файле реализуем разметку из 2ух кнопок навигации. В css-файле задаем стиль. Получаем простую навигационную панель (Приложение А) (Рис.3).

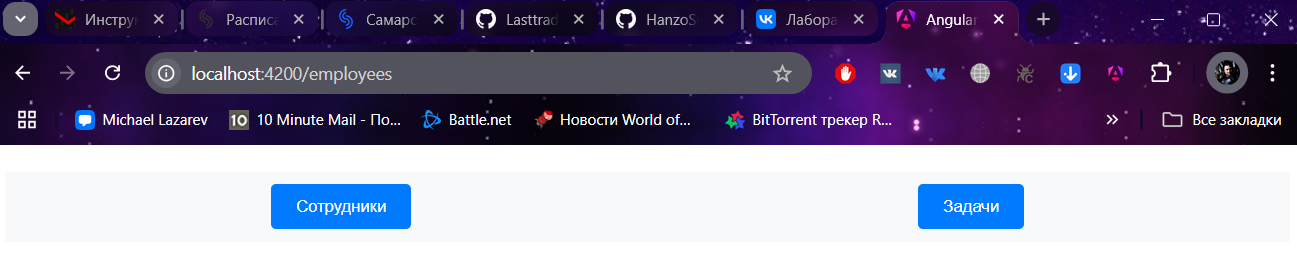


Рисунок 3 – Навигационная панель

1. Реализуем компонент employee-form для формы добавления нового и редактирования существующего сотрудника. В ts-файле дополнительно к модулям для реализации нашей формы импортируем наш сервис EmployeeService. В класс EmployeeFormComponent имплементирующий OnInit добавляем имя и роль сотрудника, а также проверку на редактирование isEditMode, реализуем в методе ngOnInit(). Метод saveEmployee() в зависимости от состояния редактирования isEditMode с параметрами false или true даёт нам непосредственно сохранить данные нового или изменить данные существующего сотрудника. В html-файле реализуем соответствующие поля и кнопки, в том числе кнопку отмены. Также после нажатия кнопок «Сохранить», «Добавить» и «Отмена» реализован возврат на страницу сотрудников. В css-файле создаём общий стиль формы. Получаем результат готовую форму (Приложение А) (Рис.4) (Рис.5).

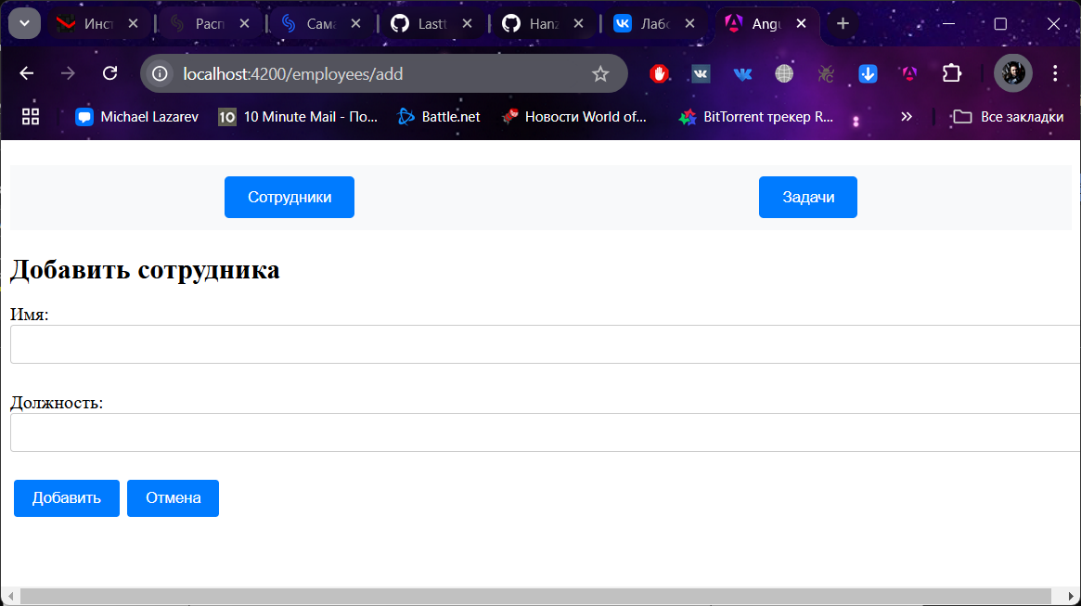


Рисунок 4 – Форма добавления нового сотрудника

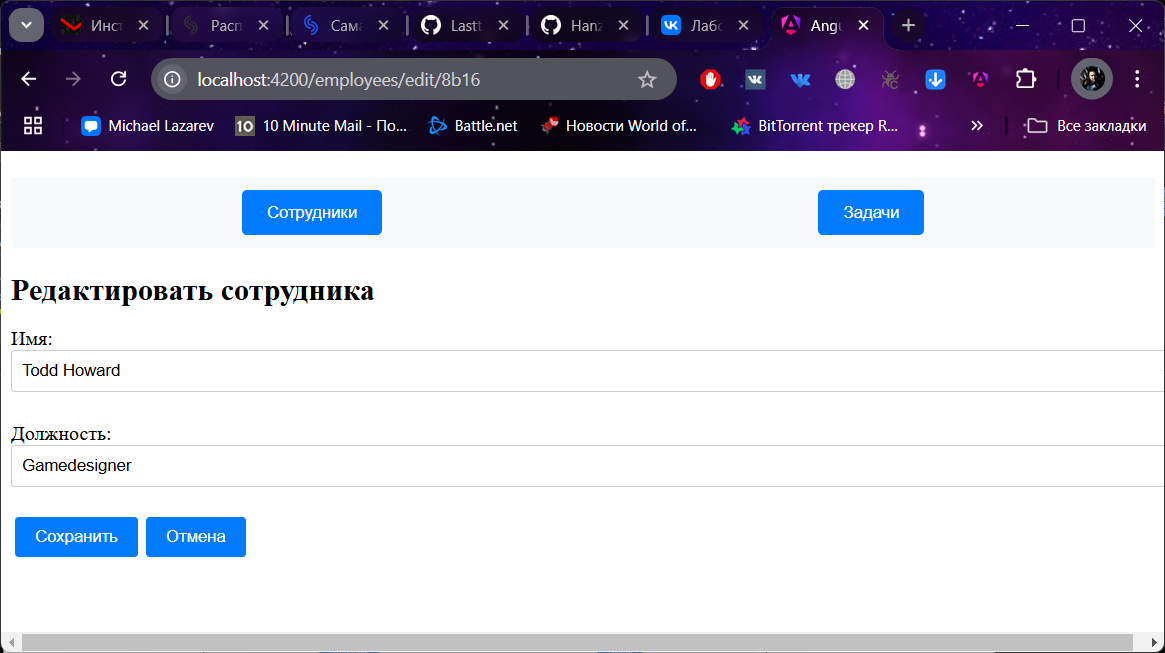


Рисунок 5 – Форма редактирования существующего сотрудника

1. Реализуем компонент employee-list для вывода списка сотрудников с соответствующими функциями. В ts-файле реализуем список сотрудников с нашего сервера, посредством метода loadEmloyees() через наш сервис EmloyeeService. Добавляем методы добавления нового сотрудника addEmployee(), редактирования сотрудника editEmployee() и удаления сотрудника deleteEmployee() с загрузкой обновленного списка. В html-файле организуем список с именем, должностью и действиями в отношении данных наших сотрудников с кнопками «Редактировать» и «Удалить». Добавляем кнопку «Добавить сотрудника». В css-файле реализуем стиль. Получаем готовую страницу EmployeeListComponent с переходом к нашей форме EmployeeFormComponent (Приложение А) (Рис.6).

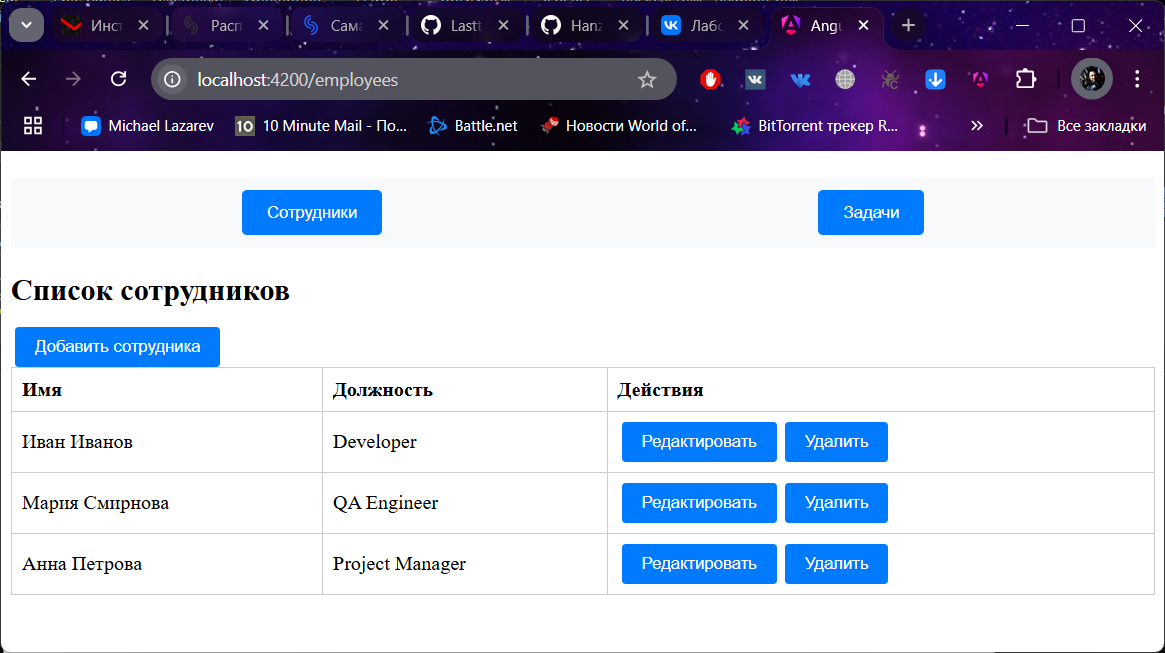


Рисунок 6 – Страница «Сотрудники»

1. Реализуем компонент task-form для создания новой и редактирования текущей задачи. В ts-файле со всеми необходимыми модулями импортируем наш сервис TaskService и интерфейс Task. Добавляем булево значение для редактирования isEditing, для создания новой задачи реализуем наш интерфейс со значениями по умолчанию. Метод ngOnInit() включает в себя загрузку задачи при редактировании и проверку значения isEditing true или false. Метод saveTask() даёт нам сохранить новую задачу или обновить текущую, а также перенаправляет обратно на список задач, также как и метод отмены cancel(). В html-файле реализуем соответствующие кнопки «Сохранить» и «Отмена» для функций нашей формы. В форме обозначаем поля для заполнения заголовка, типа задача Tusk или баг Bug с dropdown, приоритетом c dropdown реализацией, описанием, исполнителем и создателем. В css-файле добавляем стиль нашей формы для задачи. Получаем готовую форму TaskFormComponent для создания и редактирования задач (Приложение А) (Рис.7) (Рис.8).

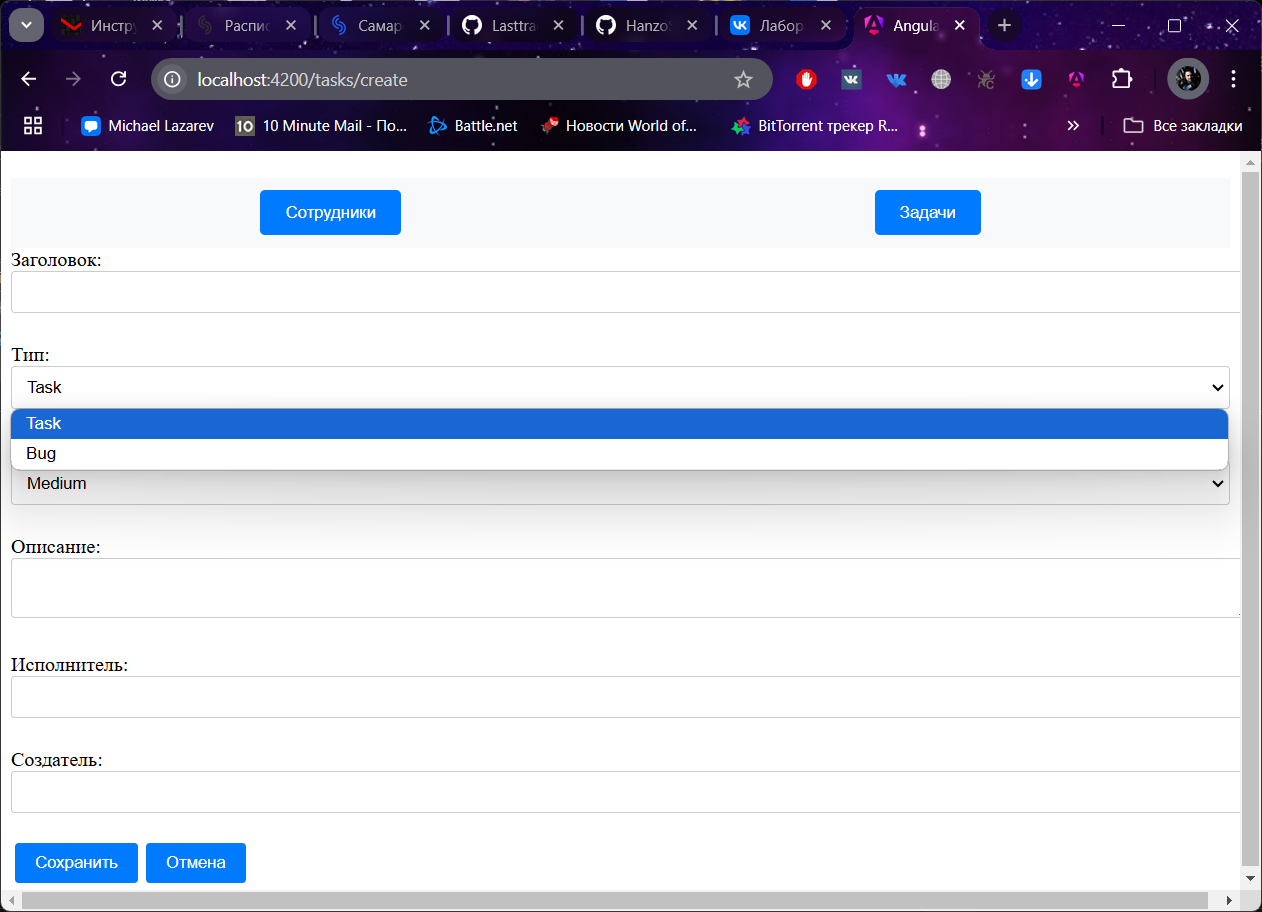


Рисунок 7 – Создание новой задачи

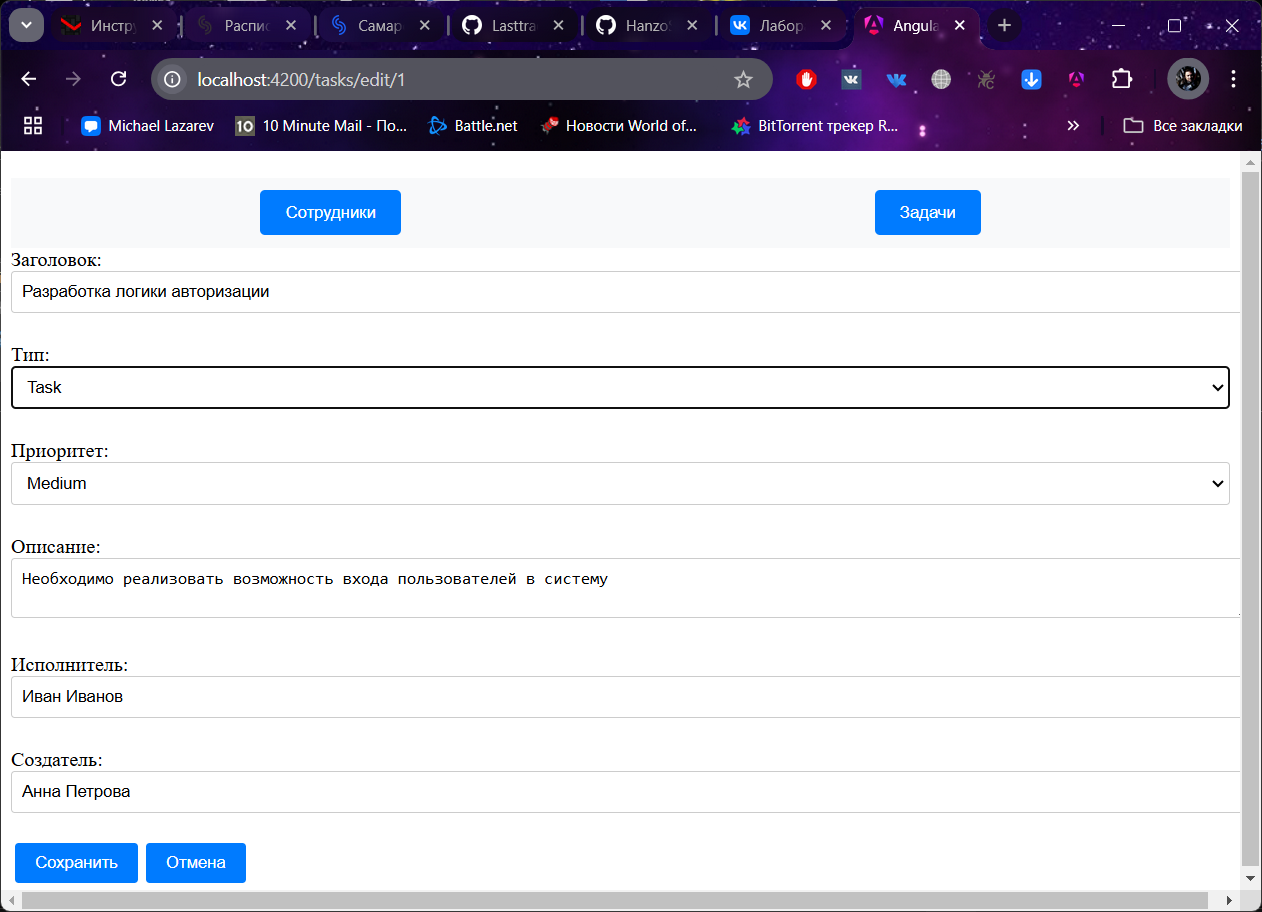


Рисунок 8 – Редактирование выбранной задачи

1. Реализуем компонент task-list для списка наших задач. В ts-файле реализуем список задач и фильтрацию по исполнителю, приоритету и типу c dropdown. Метод загрузки списка задач loadTasks() даёт нам возможность фильтровать и сортировать задачи. Реализуем фильтрацию методом applyFilter() и сортировку по дате изменения задач sortTasks(). Метод formateDate() позволяет отображать время в русском формате ЧЧ:ММ ДД.ММ.ГГГГ. Также реализуем методы добавления задачи addTask(), перенаправления на форму создания новой задачи navigateToCreateTask() и удаления задачи deleteTask() c подтверждением удаления. В html-файле реализуем фильтр по типу, приоритету и исполнителю из нашего ts-файла компонента. Добавляем кнопки для добавления новой задачи, редактирования и удаления. Список задач отображается с информацией о дате и времени обновления, исполнителю, создателю, приоритету и заголовком. В css-файле реализуем стиль нашего списка задач. В итоге у нас есть готовая страница «Задачи» компонента TaskListComponent (Приложение А) (Рис.9) (Рис.10).

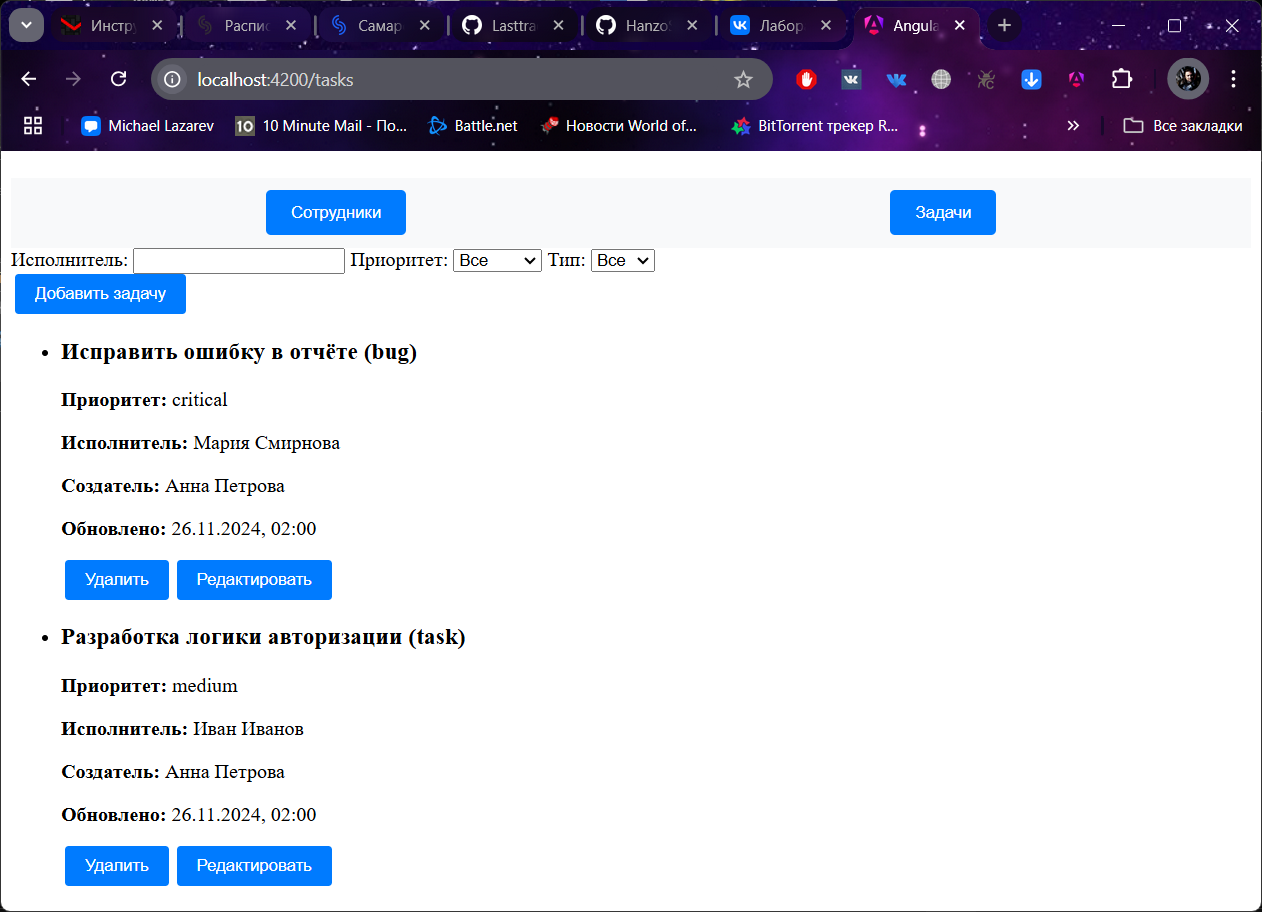


Рисунок 9 – Страница «Задачи»

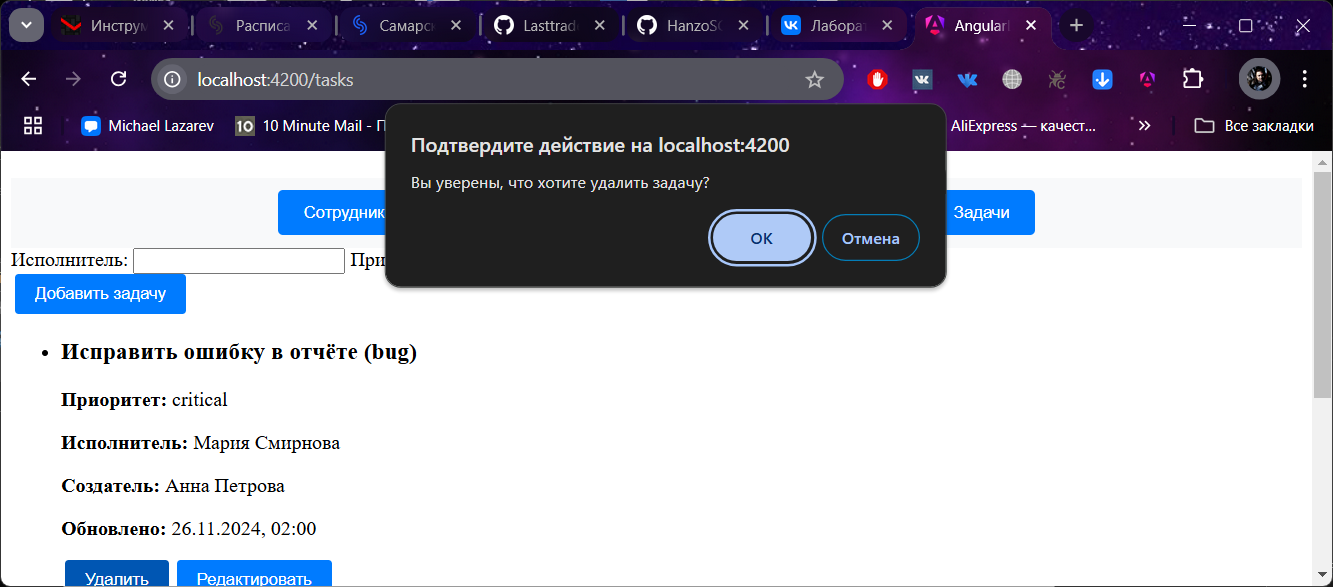


Рисунок 10 – Подтверждение удаления задачи

ПРИЛОЖЕНИЕ А  
Листинг программы

#task-server/db.json

{

  "employees": [

    {

      "id": 1,

      "name": "Иван Иванов",

      "role": "Developer"

    },

    {

      "id": 2,

      "name": "Мария Смирнова",

      "role": "QA Engineer"

    },

    {

      "id": 3,

      "name": "Анна Петрова",

      "role": "Project Manager"

    }

  ],

  "tasks": [

    {

      "id": 1,

      "type": "task",

      "priority": "medium",

      "status": "todo",

      "title": "Разработка логики авторизации",

      "description": "Необходимо реализовать возможность входа пользователей в систему",

      "assignee": "Иван Иванов",

      "creator": "Анна Петрова",

      "createdAt": "2024-11-17T10:00:00",

      "updatedAt": "2024-11-17T12:00:00"

    },

    {

      "id": 2,

      "type": "bug",

      "priority": "critical",

      "status": "inprogress",

      "title": "Исправить ошибку в отчёте",

      "description": "Отчёт по продажам формируется с неверной датой",

      "assignee": "Мария Смирнова",

      "creator": "Анна Петрова",

      "createdAt": "2024-11-16T09:30:00",

      "updatedAt": "2024-11-17T09:45:00"

    }

  ]

}

#task-server/package.json

{

"name": "simple-server",

"version": "1.0.0",

"description": "Full fake REST API server",

"main": "index.js",

"scripts": {

"start": "json-server --watch db.json --port 3000"

},

"devDependencies": {

"json-server": "^0.17.3"

},

"author": "",

"license": "ISC",

"dependencies": {}

}

#app/app.routes.ts

import { Routes } from '@angular/router';

import { EmployeeListComponent } from './components/employee-list/employee-list.component';

import { EmployeeFormComponent } from './components/employee-form/employee-form.component';

import { TaskListComponent } from './components/task-list/task-list.component';

import { TaskFormComponent } from './components/task-form/task-form.component';

export const routes: Routes = [

{ path: 'employees', component: EmployeeListComponent },

{ path: 'employees/add', component: EmployeeFormComponent },

{ path: 'employees/edit/:id', component: EmployeeFormComponent },

{ path: 'tasks', component: TaskListComponent },

{ path: 'tasks/create', component: TaskFormComponent },

{ path: 'tasks/edit/:id', component: TaskFormComponent },

{ path: '', redirectTo: '/tasks', pathMatch: 'full' },

];

#app/app.component.ts

import { Component } from '@angular/core';

import { RouterOutlet, RouterModule } from '@angular/router';

import { NavigationComponent } from './components/navigation/navigation.component';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrl: './app.component.css',

  standalone: true,

  imports: [RouterOutlet, NavigationComponent, RouterModule]

})

export class AppComponent {

  title = 'Task Manager';

}

#app/app.component.html

<h1><title>{{ title }}</title></h1>

<app-navigation></app-navigation>

<router-outlet></router-outlet>

#app/services/employee.service.ts

import { Injectable } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable({

  providedIn: 'root',

})

export class EmployeeService {

  private baseUrl = 'http://localhost:3000/employees';

  constructor(private http: HttpClient) {}

  getEmployees(): Observable<any[]> {

    return this.http.get<any[]>(this.baseUrl);

  }

  getEmployeeById(id: number): Observable<any> {

    return this.http.get<any>(`${this.baseUrl}/${id}`);

  }

  updateEmployee(id: number, employee: any): Observable<any> {

    return this.http.put<any>(`${this.baseUrl}/${id}`, employee);

  }

  deleteEmployee(id: number): Observable<any> {

    return this.http.delete<any>(`${this.baseUrl}/${id}`);

  }

  addEmployee(employee: any): Observable<any> {

    return this.http.post<any>(this.baseUrl, employee);

  }

}

# app/services/task.service.ts

import { Injectable, inject } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

export interface Task {

  id?: number;

  type: 'task' | 'bug';

  priority: 'critical' | 'high' | 'medium' | 'low';

  status: 'todo'| 'inprogress' | 'done';

  title: string;

  description: string;

  assignee: string;

  creator: string;

  createdAt: string;

  updatedAt: string;

}

@Injectable({

  providedIn: 'root',

})

export class TaskService {

  private http = inject(HttpClient);

  private apiUrl = 'http://localhost:3000/tasks';

  getTasks(): Observable<Task[]> {

    return this.http.get<Task[]>(this.apiUrl);

  }

  getTaskById(id: number): Observable<Task> {

    return this.http.get<Task>(`${this.apiUrl}/${id}`);

  }

  addTask(task: Task): Observable<Task> {

    return this.http.post<Task>(this.apiUrl, {

      ...task,

      createdAt: new Date().toISOString(),

      updatedAt: new Date().toISOString(),

    });

  }

  updateTask(id: number, task: Task): Observable<Task> {

    return this.http.put<Task>(`${this.apiUrl}/${id}`, {

      ...task,

      updatedAt: new Date().toISOString(),

    });

  }

  deleteTask(id: number): Observable<void> {

    return this.http.delete<void>(`${this.apiUrl}/${id}`);

  }

}

#app/components/navigation/navigation.component.ts

import { Component } from '@angular/core';

import { Router } from '@angular/router';

@Component({

  selector: 'app-navigation',

  templateUrl: './navigation.component.html',

  styleUrls: ['./navigation.component.css'],

  standalone: true

})

export class NavigationComponent {

  constructor(private router: Router) {}

  navigateToEmployees() {

    this.router.navigate(['/employees']);

  }

  navigateToTasks() {

    this.router.navigate(['/tasks']);

  }

}

#app/components/navigation/navigation.component.html

<nav class="navigation">

    <button (click)="navigateToEmployees()">Сотрудники</button>

    <button (click)="navigateToTasks()">Задачи</button>

  </nav>

#app/components/navigation/navigation.component.css

.navigation {

    display: flex;

    justify-content: space-around;

    background-color: #f8f9fa;

    padding: 10px;

  }

  button {

    padding: 10px 20px;

    background-color: #007bff;

    color: white;

    border: none;

    border-radius: 4px;

    cursor: pointer;

  }

  button:hover {

    background-color: #0056b3;

  }

#app/components/employee-form/employee-form.component.ts

import { Component, OnInit } from '@angular/core';

import { EmployeeService } from '../../services/employee.service';

import { ActivatedRoute, Router } from '@angular/router';

import { CommonModule } from '@angular/common';

import { FormsModule } from '@angular/forms';

@Component({

  selector: 'app-employee-form',

  standalone: true,

  imports: [CommonModule, FormsModule],

  templateUrl: './employee-form.component.html',

  styleUrls: ['./employee-form.component.css'],

})

export class EmployeeFormComponent implements OnInit {

  employee: any = {

    name: '',

    role: '',

  };

  isEditMode = false;

  constructor(

    private employeeService: EmployeeService,

    public router: Router,

    private route: ActivatedRoute

  ) {}

  ngOnInit(): void {

    const employeeId = this.route.snapshot.params['id'];

    if (employeeId) {

      this.isEditMode = true;

      this.employeeService.getEmployeeById(employeeId).subscribe((data) => {

        this.employee = data;

      });

    }

  }

  saveEmployee(): void {

    if (this.isEditMode) {

      this.employeeService.updateEmployee(this.employee.id, this.employee).subscribe(() => {

        this.router.navigate(['/employees']);

      });

    } else {

      this.employeeService.addEmployee(this.employee).subscribe(() => {

        this.router.navigate(['/employees']);

      });

    }

  }

}

#app/components/employee-form/employee-form.component.html

<div class="employee-form">

  <h2>{{ isEditMode ? 'Редактировать сотрудника' : 'Добавить сотрудника' }}</h2>

  <form (ngSubmit)="saveEmployee()">

    <label>

      Имя:

      <input [(ngModel)]="employee.name" name="name" required />

    </label>

    <label>

      Должность:

      <input [(ngModel)]="employee.role" name="role" required />

    </label>

    <button type="submit">{{ isEditMode ? 'Сохранить' : 'Добавить' }}</button>

    <button type="button" (click)="router.navigate(['/employees'])">Отмена</button>

  </form>

</div>

#app/components/employee-form/employee-form.component.css

h2 {

    font-size: 1.5rem;

    margin-bottom: 1rem;

  }

  table {

    width: 100%;

    border-collapse: collapse;

    margin-bottom: 1rem;

  }

  table, th, td {

    border: 1px solid #ccc;

  }

  th, td {

    padding: 0.5rem;

    text-align: left;

  }

  button {

    margin: 0 0.2rem;

    padding: 0.5rem 1rem;

    background-color: #007bff;

    color: white;

    border: none;

    border-radius: 3px;

    cursor: pointer;

  }

  button:hover {

    background-color: #0056b3;

  }

  form label {

    display: block;

    margin-bottom: 0.5rem;

  }

  form input, form select, form textarea {

    width: 100%;

    margin-bottom: 1rem;

    padding: 0.5rem;

    border: 1px solid #ccc;

    border-radius: 3px;

  }

#app/components/employee-list/employee-list.component.ts

import { Component, OnInit } from '@angular/core';

import { EmployeeService } from '../../services/employee.service';

import { Router } from '@angular/router';

import { CommonModule } from '@angular/common';

@Component({

selector: 'app-employee-list',

standalone: true,

imports: [CommonModule],

templateUrl: './employee-list.component.html',

styleUrls: ['./employee-list.component.css'],

})

export class EmployeeListComponent implements OnInit {

employees: any[] = [];

constructor(private employeeService: EmployeeService, private router: Router) {}

ngOnInit(): void {

this.loadEmployees();

}

loadEmployees(): void {

this.employeeService.getEmployees().subscribe((data) => {

this.employees = data;

});

}

deleteEmployee(employeeId: number): void {

if (confirm('Вы уверены, что хотите удалить сотрудника?')) {

this.employeeService.deleteEmployee(employeeId).subscribe(() => {

this.loadEmployees();

});

}

}

editEmployee(employeeId: number): void {

this.router.navigate([`/employees/edit/${employeeId}`]);

}

addEmployee(): void {

this.router.navigate(['/employees/add']);

}

}

#app/components/employee-list/employee-list.component.html

<div class="employee-list">

  <h2>Список сотрудников</h2>

  <button (click)="addEmployee()">Добавить сотрудника</button>

  <table>

    <thead>

      <tr>

        <th>Имя</th>

        <th>Должность</th>

        <th>Действия</th>

      </tr>

    </thead>

    <tbody>

      <tr \*ngFor="let employee of employees">

        <td>{{ employee.name }}</td>

        <td>{{ employee.role }}</td>

        <td>

          <button (click)="editEmployee(employee.id)">Редактировать</button>

          <button (click)="deleteEmployee(employee.id)">Удалить</button>

        </td>

      </tr>

    </tbody>

  </table>

</div>

#app/components/employee-list/employee-list.component.css

h2 {

    font-size: 1.5rem;

    margin-bottom: 1rem;

  }

  table {

    width: 100%;

    border-collapse: collapse;

    margin-bottom: 1rem;

  }

  table, th, td {

    border: 1px solid #ccc;

  }

  th, td {

    padding: 0.5rem;

    text-align: left;

  }

  button {

    margin: 0 0.2rem;

    padding: 0.5rem 1rem;

    background-color: #007bff;

    color: white;

    border: none;

    border-radius: 3px;

    cursor: pointer;

  }

  button:hover {

    background-color: #0056b3;

  }

  form label {

    display: block;

    margin-bottom: 0.5rem;

  }

  form input, form select, form textarea {

    width: 100%;

    margin-bottom: 1rem;

    padding: 0.5rem;

    border: 1px solid #ccc;

    border-radius: 3px;

  }

#app/components/task-form/task-form.component.ts

import { Component, inject } from '@angular/core';

import { CommonModule } from '@angular/common';

import { FormsModule } from '@angular/forms';

import { Router, ActivatedRoute } from '@angular/router';

import { TaskService, Task } from '../../services/task.service';

@Component({

  selector: 'app-task-form',

  standalone: true,

  imports: [CommonModule, FormsModule],

  templateUrl: './task-form.component.html',

  styleUrls: ['./task-form.component.css'],

})

export class TaskFormComponent {

  private taskService = inject(TaskService);

  private router = inject(Router);

  private route = inject(ActivatedRoute);

  task: Task = {

    type: 'task',

    priority: 'medium',

    status: 'todo',

    title: '',

    description: '',

    assignee: '',

    creator: '',

    createdAt: '',

    updatedAt: '',

  };

  isEditing = false;

  ngOnInit() {

    const id = this.route.snapshot.params['id'];

    if (id) {

      this.isEditing = true;

      this.taskService.getTaskById(id).subscribe((task) => {

        this.task = task;  // Если редактируем, загружаем задачу

      });

    }

  }

  saveTask() {

    if (this.isEditing) {

      this.taskService.updateTask(this.task.id!, this.task).subscribe(() => {

        this.router.navigate(['/tasks']);

      });

    } else {

      this.taskService.addTask(this.task).subscribe(() => {

        this.router.navigate(['/tasks']);

      });

    }

  }

  cancel() {

    this.router.navigate(['/tasks']);

  }

}

#app/components/task-form/task-form.component.html

<form (ngSubmit)="saveTask()" class="task-form">

  <label>

    Заголовок:

    <input type="text" [(ngModel)]="task.title" name="title" required />

  </label>

  <label>

    Тип:

    <select [(ngModel)]="task.type" name="type" required>

      <option value="task">Task</option>

      <option value="bug">Bug</option>

    </select>

  </label>

  <label>

    Приоритет:

    <select [(ngModel)]="task.priority" name="priority" required>

      <option value="critical">Critical</option>

      <option value="high">High</option>

      <option value="medium">Medium</option>

      <option value="low">Low</option>

    </select>

  </label>

  <label>

    Описание:

    <textarea [(ngModel)]="task.description" name="description" required></textarea>

  </label>

  <label>

    Исполнитель:

    <input type="text" [(ngModel)]="task.assignee" name="assignee" required />

  </label>

  <label>

    Создатель:

    <input type="text" [(ngModel)]="task.creator" name="creator" required />

  </label>

  <div class="form-buttons">

    <button type="submit" class="save-button">Сохранить</button>

    <button type="button" (click)="cancel()" class="cancel-button">Отмена</button>

  </div>

</form>

#app/components/task-form/task-form.component.css

h2 {

    font-size: 1.5rem;

    margin-bottom: 1rem;

  }

  table {

    width: 100%;

    border-collapse: collapse;

    margin-bottom: 1rem;

  }

  table, th, td {

    border: 1px solid #ccc;

  }

  th, td {

    padding: 0.5rem;

    text-align: left;

  }

  button {

    margin: 0 0.2rem;

    padding: 0.5rem 1rem;

    background-color: #007bff;

    color: white;

    border: none;

    border-radius: 3px;

    cursor: pointer;

  }

  button:hover {

    background-color: #0056b3;

  }

  form label {

    display: block;

    margin-bottom: 0.5rem;

  }

  form input, form select, form textarea {

    width: 100%;

    margin-bottom: 1rem;

    padding: 0.5rem;

    border: 1px solid #ccc;

    border-radius: 3px;

  }

#app/components/task-list/task-list.component.ts

import { Component, inject } from '@angular/core';

import { CommonModule } from '@angular/common';

import { RouterModule, Router } from '@angular/router';

import { FormsModule } from '@angular/forms';

import { TaskService, Task } from '../../services/task.service';

@Component({

  selector: 'app-task-list',

  standalone: true,

  imports: [CommonModule, FormsModule, RouterModule],

  templateUrl: './task-list.component.html',

  styleUrls: ['./task-list.component.css'],

})

export class TaskListComponent {

  private taskService = inject(TaskService);

  private router = inject(Router);

  tasks: Task[] = [];

  filteredTasks: Task[] = [];

  filter = {

    assignee: '',

    priority: '',

    type: '',

  };

  ngOnInit() {

    this.loadTasks();

  }

  loadTasks() {

    this.taskService.getTasks().subscribe((tasks) => {

      this.tasks = tasks;

      this.filteredTasks = [...tasks];

      this.sortTasks();

    });

  }

  applyFilter() {

    this.filteredTasks = this.tasks.filter((task) => {

      const matchesAssignee = this.filter.assignee

        ? task.assignee.includes(this.filter.assignee)

        : true;

      const matchesPriority = this.filter.priority

        ? task.priority === this.filter.priority

        : true;

      const matchesType = this.filter.type ? task.type === this.filter.type : true;

      return matchesAssignee && matchesPriority && matchesType;

    });

  }

  sortTasks() {

    this.filteredTasks.sort(

      (a, b) => new Date(b.updatedAt).getTime() - new Date(a.updatedAt).getTime()

    );

  }

  formatDate(date: string): string {

    const options: Intl.DateTimeFormatOptions = {

      hour: '2-digit',

      minute: '2-digit',

      day: '2-digit',

      month: '2-digit',

      year: 'numeric',

    };

    return new Date(date).toLocaleString('ru-RU', options);

  }

  deleteTask(id: number | undefined) {

    if (id === undefined) {

      console.error('Невозможно удалить задачу: ID не указан.');

      return;

    }

    if (confirm('Вы уверены, что хотите удалить задачу?')) {

      this.taskService.deleteTask(id).subscribe(() => {

        this.loadTasks();

      });

    }

  }

  addTask(newTask: Task) {

    this.taskService.addTask(newTask).subscribe(() => {

      this.loadTasks();

    });

  }

  navigateToCreateTask() {

    this.router.navigate(['/tasks/create']);

  }

}

#app/components/task-list/task-list.component.html

<div class="filters">

  <label>

    Исполнитель:

    <input type="text" [(ngModel)]="filter.assignee" (input)="applyFilter()" />

  </label>

  <label>

    Приоритет:

    <select [(ngModel)]="filter.priority" (change)="applyFilter()">

      <option value="">Все</option>

      <option value="critical">Critical</option>

      <option value="high">High</option>

      <option value="medium">Medium</option>

      <option value="low">Low</option>

    </select>

  </label>

  <label>

    Тип:

    <select [(ngModel)]="filter.type" (change)="applyFilter()">

      <option value="">Все</option>

      <option value="task">Task</option>

      <option value="bug">Bug</option>

    </select>

  </label>

</div>

<div class="add-task-btn">

  <button (click)="navigateToCreateTask()">Добавить задачу</button>

</div>

<ul>

  <li \*ngFor="let task of filteredTasks">

    <h3>{{ task.title }} ({{ task.type }})</h3>

    <p><strong>Приоритет:</strong> {{ task.priority }}</p>

    <p><strong>Исполнитель:</strong> {{ task.assignee }}</p>

    <p><strong>Создатель:</strong> {{ task.creator }}</p>

    <p><strong>Обновлено:</strong> {{ formatDate(task.updatedAt) }}</p>

    <button (click)="deleteTask(task.id)" class="btn btn-danger">Удалить</button>

 <button [routerLink]="['/tasks/edit', task.id]" class="btn btn-primary">Редактировать</button>

  </li>

</ul>

#app/components/task-list/task-list.component.css

h2 {

    font-size: 1.5rem;

    margin-bottom: 1rem;

  }

  table {

    width: 100%;

    border-collapse: collapse;

    margin-bottom: 1rem;

  }

  table, th, td {

    border: 1px solid #ccc;

  }

  th, td {

    padding: 0.5rem;

    text-align: left;

  }

  button {

    margin: 0 0.2rem;

    padding: 0.5rem 1rem;

    background-color: #007bff;

    color: white;

    border: none;

    border-radius: 3px;

    cursor: pointer;

  }

  button:hover {

    background-color: #0056b3;

  }

  form label {

    display: block;

    margin-bottom: 0.5rem;

  }

  form input, form select, form textarea {

    width: 100%;

    margin-bottom: 1rem;

    padding: 0.5rem;

    border: 1px solid #ccc;

    border-radius: 3px;

  }