

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 №5 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 1 4](#_Toc182656071)

[2.2 Результаты задания 2 5](#_Toc182656072)

[2.3 Результаты задания 3 7](#_Toc182656073)

[2.4 Результаты задания 4 8](#_Toc182656074)

[2.5 Результаты задания 5 8](#_Toc182656075)

[2.6 Результаты задания 6 9](#_Toc182656076)

[2.7 Результаты задания 7 9](#_Toc182656077)

[2.8 Результаты задания 8 9](#_Toc182656078)

[ПРИЛОЖЕНИЕ А Листинг программы 11](#_Toc182656079)

1. Постановка задачи

В процессе выполнения заданий ознакомиться с guard, interceptor, директивами, pipe, работой с формами и обменом данными с сервером приложений.

1. Результаты работы
   1. Результаты задания 1

Создайте json-файл описывающий структуру групп и студентов. Используйте этот файл для эмуляции CRUD операций с json Rest Full сервером.

1. В папке, где будут лежать файлы нашего сервера, открываем терминал и выполняем команду npm install json-server.
2. Создаем файл базы данных db.json, который содержит информацию о студентах и группах (Приложение А).
3. Редактируем package.json (Приложение А).
4. В терминале вводим npm run start (Рис. 1) и открываем в браузере http://localhost:3000/ (Рис. 2).

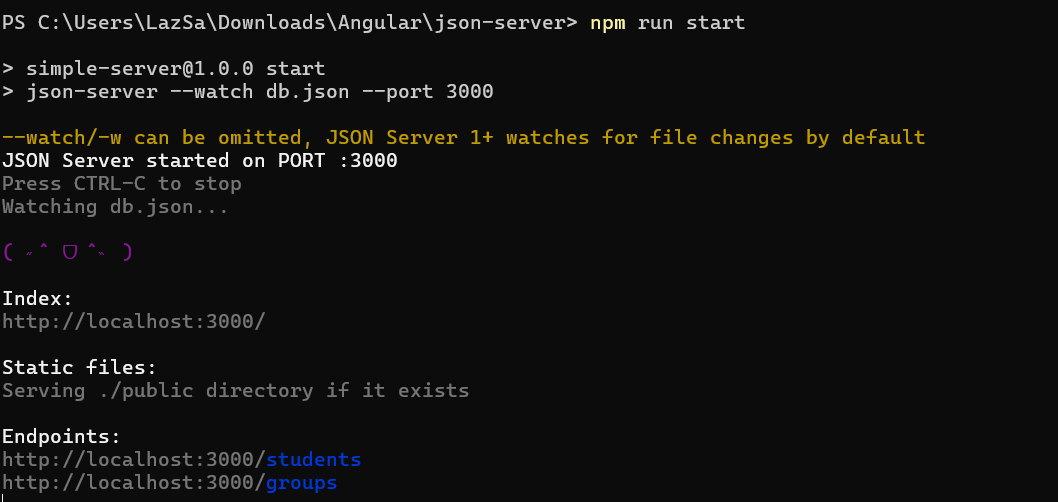


Рисунок 1 – Результат работы терминала

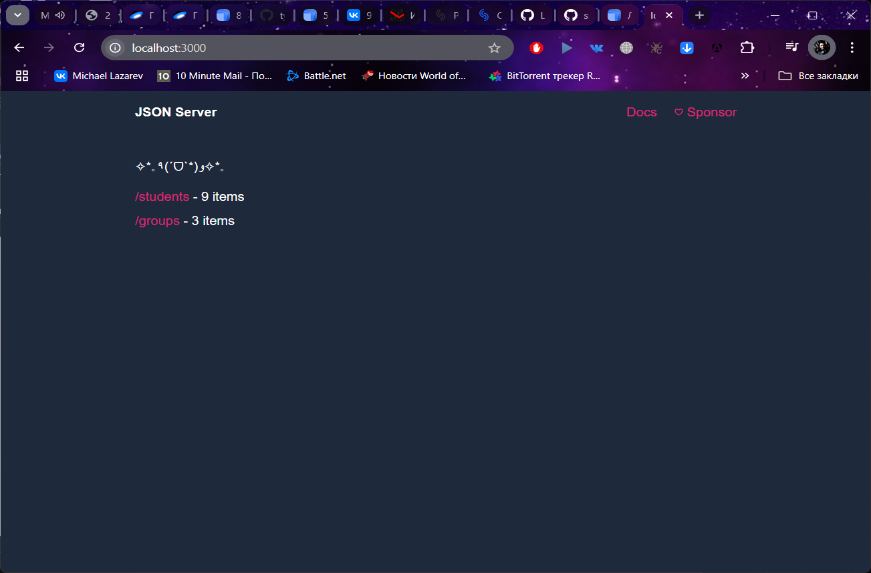


Рисунок 2 – Работа сервера в браузере

* 1. Результаты задания 2

Создайте два сервиса для взаимодействия с вашим Rest full server. Один для работы с группами (Ид, номер группы), второй – со студентами (ФИО, год рождения, ссылка на группу, курс, год поступления).

1. Открываем терминал в папке нашего Angular приложения. Вводим команды для создания шаблонов сервисов ng generate service services/student и ng generate service services/group для взаимодействия со студентами и группами в нашем db.json соответственно.
2. Реализуем в app/services/group.service.ts экспортируемый класс GroupService с CRUD операциями. Производим импорт HttpClient из @angular/common/http и Observable из rxjs. Далее в нашем классе указываем URL для доступа к группам по http://localhost:3000/groups и реализуем CRUD операции для получения списка всех групп getGroups(), получения группы по ID getGroupById(), добавления новой группы addGroup(), обновления существующей группы updateGroup() и удаления группы deleteGroup() (Приложение А).
3. Реализуем в app/services/student.service.ts экспортируемый класс StudentService с CRUD операциями. Аналогично как и в GroupService, в нашем классе указываем URL для доступа к студентам по http://localhost:3000/students и реализуем CRUD операции для получения списка всех студентов getStudents(), получения студента по ID getStudentById(), добавления нового студента addStudent(), обновления данных студента updateStudent(), удаления студента deleteStudent() и получения студентов по ID группы getStudentsByGroupId() (Приложение А).
   1. Результаты задания 3

Поправьте страницу группы из предыдущей лабораторной работы. На странице должен выводиться дропдаун со списком существующих групп (данные берутся, при выборе группы, должен показываться список студентов в произвольной форме).

1. В файле app.config.ts в providers добавляем provideHttpClient() для взаимодействия с нашим сервером.
2. Редактируем ts файл нашего компонента GroupComponent. Импортируем в класс компонента наши сервисы, а также inject и применяем инжекцию к импортируемому HttpClient. Производим загрузку списка групп при помощи loadGroups(), а также загрузку списка студентов loadStudents(). Добавляем методы обработки изменения группы onGroupChange(), добавления нового студента addStudent(), удаления студента deleteStudent(), редактирования startEdit(), сохранения изменений saveStudent() и отмены редактирования cancelEdit() (Приложение А).
3. Стилизуем страницу в css и оформляем в html (Приложение А).
4. Видим результат в браузере (Рис.3).

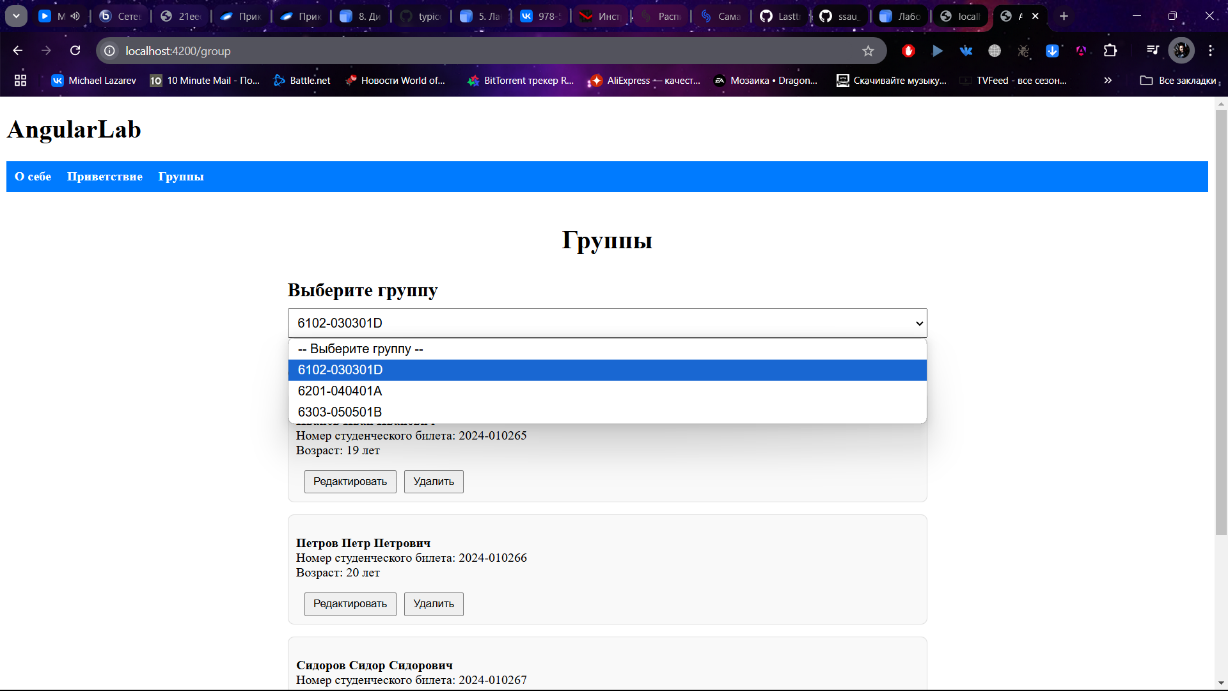


Рисунок 3 – Страница «Группы»

* 1. Результаты задания 4

Напишите директиву, которая будет преобразовывать текст в курсив, если студенту меньше 18 лет. Подключите директиву на странице с группами.

1. Открываем терминал в папке Angular приложения и вводим команду ng generate directive directives/under18 для создания шаблона нашей директивы.
2. Редактируем файл app/directives/under18.directive.ts. Импортируем из @angular/core Directive, ElementRef, Renderer2, Input и OnInit. Назначаем директиву как appUnder18, которая в свою очередь определяет возраст студента по возрасту Age и добавляет стиль курсива (Приложение А).
3. В ts-файле компонента GroupComponent производим импорт созданной директивы Under18Directive, а в html-файле добавляем нашу директиву, применяемую к списку студентов. Получаем результат в браузере (Рис.4).

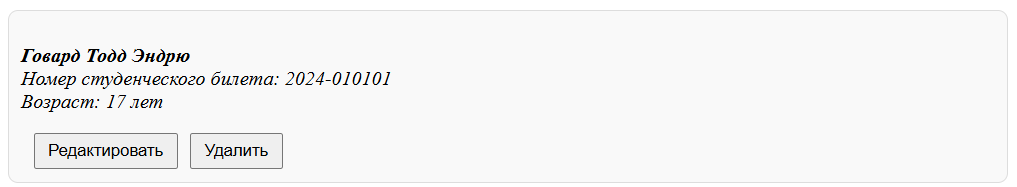


Рисунок 4 – Результат выполнения директивы

* 1. Результаты задания 5

Напишите Pipe который будет форматировать текст с датой рождения в виде «Родился в таком-то веке нашей эры». Примените его к странице групп.

1. Открываем терминал в папке Angular приложения и вводим ng generate pipe pipes/birth-century для создания шаблона.
2. Редактируем файл app/pipes/birth-century.pipe.ts. Импортируем из @angular/core Pipe и PipeTransform. Класс BirthCenturyPipe имплементирует PipeTransform, вычисляет год рождения от возраста и определяет век (Приложение А).
3. Вносим изменения в компонент GroupComponent. В html-файле добавляем строку для отображения века рождения студента. В ts-файл импортируем наш BirthCenturyPipe. Получаем результат в браузере (Рис.5).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 5 – Отображение века

* 1. Результаты задания 6

Сделайте реактивную форму по добавлению нового студента. Реализуйте полностью функционал. При попытке ухода с данной страницы должен срабатывать Guard, который подтверждает, что мы переходим без сохранения данных. При добавлении данных мы должны быть перенаправлены на станицу групп – данные о студентах должны быть обновлены.

1. Создаем через терминал в папке Angular приложения шаблон командой ng generate guard guards/form-unsaved.
2. В app/guards/form-unsaved.guard.ts прописываем необходимые импорты и реализуем экспортируемый класс FormUnsavedGuard, имплементирующий CanDeactivate, при этом применяется булево значение к компоненту GroupComponent. Если форма изменена и не сохранена, показываем предупреждение (Рис.6) (Приложение А).
3. Редактируем ts-файл GroupComponent. Импортируем FormBuilder, ReactiveFormsModule и Validators из форм Angular. В constructor подключаем FormBuilder для создания формы и составляем инициализацию формы studentForm с валидацией. Реализуем метод добавления студента с перенаправлением на страницу «Группы» и alert для корректного заполнения формы (Рис.7) (Приложение А).
4. Обновляем html-файл в соответствии с реализацией реактивной формы в ts-файле, привязываем реализацию studentForm. В файле роутинга импортируем наш guard для GroupComponent (Приложение А).

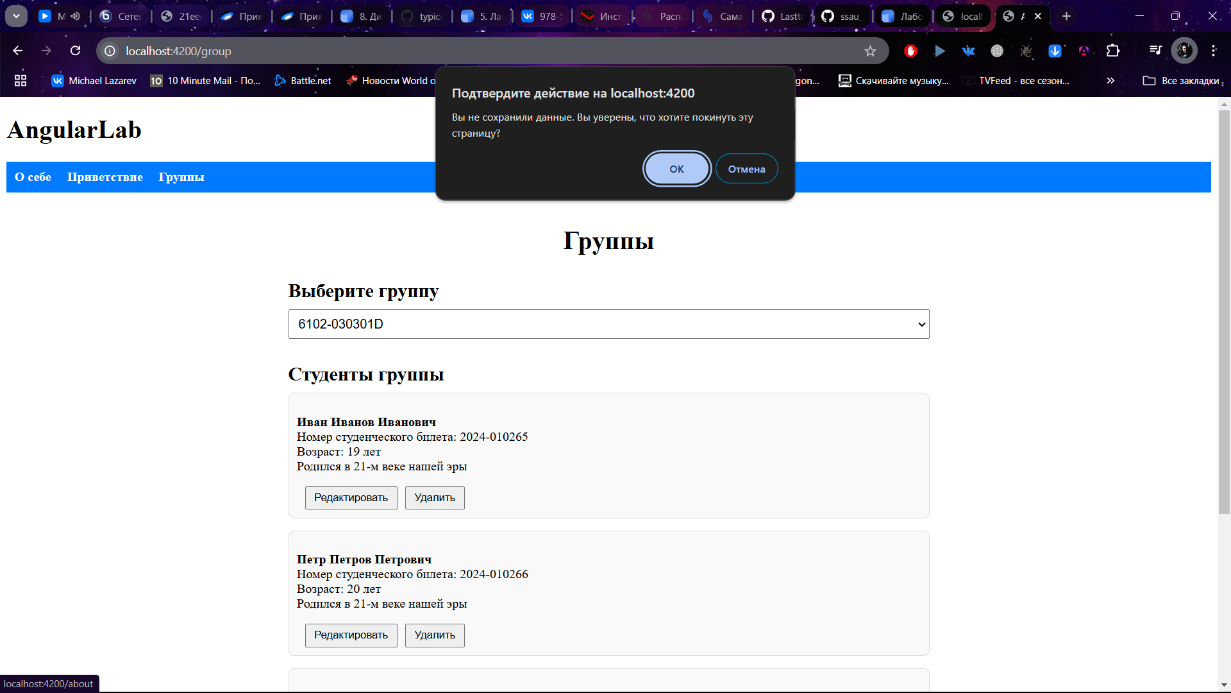


Рисунок 6 - Предупреждение

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 7 – Некорректное заполнение данных

* 1. Результаты задания 7

Сделайте template-driven форму для редактирования студента. Реализуйте полностью функционал. При обновлении данных мы должны быть перенаправлены на станицу групп – данные о студентах должны быть обновлены.

1. В нашем ts-файле GroupComponent прописываем метод startEdit() для редактирования, который копирует данные студента в поля. Метод saveStudent() позволяет сохранить и обновить изменения. Метод cancelEdit() реализует отмену редактирования (Приложение А).
2. В нашем html-файле GroupComponent реализуем форму редактирования и привязку данных с помощью ngModule, а директива ngForm позволяет отслеживать состояние элементов формы и их валидность (Приложение А).
3. В браузере получаем соответствующую форму (Рис.8).

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

Автоматически созданное описание

Рисунок 8 – Форма редактирования

* 1. Результаты задания 8

Добавьте интерцептор, который будет логировать каждый вызов API.

1. В папке нашего приложения открываем терминал и вводим ng generate interceptor interceptors/api-logger для создания шаблона.
2. Редактируем получившийся файл app/interceptors/api-logger.interceptor.ts. Так как приложение состоит из standalone компонентов, создаем функциональный интерцептор, который осуществляет логирование в консоли браузера (Приложение А).
3. В main.ts реализуем наш интерцептор и проверяем реализованое логирование в консоли браузера (Рис.9) (Приложение А).

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

Автоматически созданное описание

Рисунок 9 – Работа интерцептора

ПРИЛОЖЕНИЕ А  
Листинг программы

#json-server/db.json

{

"students": [

{

"id": 1,

"fname": "Иванов",

"sname": "Иван",

"mname": "Иванович",

"stud": "2024-010265",

"groupId": 1,

"age": 19

},

{

"id": 2,

"fname": "Петров",

"sname": "Петр",

"mname": "Петрович",

"stud": "2024-010266",

"groupId": 1,

"age": 20

},

{

"id": 3,

"fname": "Сидоров",

"sname": "Сидор",

"mname": "Сидорович",

"stud": "2024-010267",

"groupId": 1,

"age": 18

},

{

"id": 4,

"fname": "Кузнецов",

"sname": "Алексей",

"mname": "Алексеевич",

"stud": "2024-010268",

"groupId": 2,

"age": 21

},

{

"id": 5,

"fname": "Попова",

"sname": "Мария",

"mname": "Андреевна",

"stud": "2024-010269",

"groupId": 2,

"age": 20

},

{

"id": 6,

"fname": "Васильева",

"sname": "Анна",

"mname": "Игоревна",

"stud": "2024-010270",

"groupId": 2,

"age": 22

},

{

"id": 7,

"fname": "Михайлов",

"sname": "Дмитрий",

"mname": "Викторович",

"stud": "2024-010271",

"groupId": 3,

"age": 23

},

{

"id": 8,

"fname": "Егорова",

"sname": "Ольга",

"mname": "Сергеевна",

"stud": "2024-010272",

"groupId": 3,

"age": 18

},

{

"id": 9,

"fname": "Николаева",

"sname": "Екатерина",

"mname": "Александровна",

"stud": "2024-010273",

"groupId": 3,

"age": 19

}

],

"groups": [

{

"id": 1,

"name": "6102-030301D"

},

{

"id": 2,

"name": "6201-040401A"

},

{

"id": 3,

"name": "6303-050501B"

}

]

}

#json-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/services/group.service.ts

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

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

import { Observable } from 'rxjs';

@Injectable({

providedIn: 'root'

})

export class GroupService {

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

constructor(private http: HttpClient) {}

getGroups(): Observable<any[]> {

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

}

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

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

}

addGroup(group: { name: string }): Observable<any> {

return this.http.post(this.apiUrl, group);

}

updateGroup(id: number, updatedData: any): Observable<any> {

return this.http.put(`${this.apiUrl}/${id}`, updatedData);

}

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

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

}

}

#app/services/student.service.ts

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

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

import { Observable } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class StudentService {

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

  constructor(private http: HttpClient) {}

  getStudents(): Observable<any[]> {

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

  }

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

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

  }

  addStudent(student: {

    fname: string;

    sname: string;

    mname: string;

    stud: string;

    groupId: number;

    age: number;

  }): Observable<any> {

    return this.http.post(this.apiUrl, student);

  }

  updateStudent(id: number, updatedData: {

    fname: string;

    sname: string;

    mname: string;

    stud: string;

    groupId: number;

    age: number;

  }): Observable<any> {

    return this.http.put(`${this.apiUrl}/${id}`, updatedData);

  }

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

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

  }

  getStudentsByGroupId(groupId: number): Observable<any[]> {

    return this.http.get<any[]>(`${this.apiUrl}?groupId=${groupId}`);

  }

}

#app/components/group.component.http

<div class="group-container">

  <h1>Группы</h1>

  <div class="group-select">

    <h2>Выберите группу</h2>

    <select id="groupSelect" (change)="onGroupChange($event)" aria-label="Выберите группу">

      <option value="">-- Выберите группу --</option>

      <option \*ngFor="let group of groups" [value]="group.id">{{ group.name }}</option>

    </select>

  </div>

  <div \*ngIf="selectedGroupId !== null" class="student-list">

    <h2>Студенты группы {{ selectedGroupName }}</h2>

    <div \*ngIf="students.length === 0">Нет студентов в этой группе.</div>

    <div class="student-card" \*ngFor="let student of students">

      <p [appUnder18]="student.age">

        <strong>{{ student.fname }} {{ student.sname }} {{ student.mname }}</strong>

        <br />

        Номер студенческого билета: {{ student.stud }}

        <br />

        Возраст: {{ student.age }} лет

        <br />

        {{ student.age | birthCentury }}

      </p>

      <button (click)="startEdit(student)">Редактировать</button>

      <button (click)="deleteStudent(student.id)">Удалить</button>

    </div>

  </div>

  <div \*ngIf="selectedGroupId !== null && selectedStudent" class="student-form">

    <h2>Редактировать студента</h2>

    <form #editStudentForm="ngForm" (ngSubmit)="saveStudent(editStudentForm)">

      <div>

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

        <input

          id="sname"

          name="sname"

          [(ngModel)]="selectedStudent.sname"

          required

        />

      </div>

      <div>

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

        <input

          id="fname"

          name="fname"

          [(ngModel)]="selectedStudent.fname"

          required

        />

      </div>

      <div>

        <label for="mname">Отчество:</label>

        <input

          id="mname"

          name="mname"

          [(ngModel)]="selectedStudent.mname"

          required

        />

      </div>

      <div>

        <label for="stud">Номер студенческого билета:</label>

        <input

          id="stud"

          name="stud"

          [(ngModel)]="selectedStudent.stud"

          required

        />

      </div>

      <div>

        <label for="age">Возраст:</label>

        <input

          id="age"

          name="age"

          type="number"

          [(ngModel)]="selectedStudent.age"

          required

          min="0"

        />

      </div>

      <button type="submit" [disabled]="editStudentForm.invalid">Сохранить</button>

      <button type="button" (click)="cancelEdit()">Отмена</button>

    </form>

  </div>

  <!-- Форма для добавления студента -->

  <div \*ngIf="selectedGroupId !== null" class="student-form">

    <h2>Добавить студента</h2>

    <form [formGroup]="studentForm" (ngSubmit)="addStudent()">

      <div>

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

        <input id="fname" formControlName="fname" placeholder="Введите имя" />

        <div \*ngIf="studentForm.get('fname')?.invalid && studentForm.get('fname')?.touched" class="error">

          Имя обязательно.

        </div>

      </div>

      <div>

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

        <input id="sname" formControlName="sname" placeholder="Введите фамилию" />

        <div \*ngIf="studentForm.get('sname')?.invalid && studentForm.get('sname')?.touched" class="error">

          Фамилия обязательна.

        </div>

      </div>

      <div>

        <label for="mname">Отчество:</label>

        <input id="mname" formControlName="mname" placeholder="Введите отчество" />

        <div \*ngIf="studentForm.get('mname')?.invalid && studentForm.get('mname')?.touched" class="error">

          Отчество обязательно.

        </div>

      </div>

      <div>

        <label for="stud">Номер студенческого билета:</label>

        <input id="stud" formControlName="stud" placeholder="Введите номер студенческого билета" />

        <div \*ngIf="studentForm.get('stud')?.invalid && studentForm.get('stud')?.touched" class="error">

          Номер студенческого билета обязателен.

        </div>

      </div>

      <div>

        <label for="age">Возраст:</label>

        <input id="age" type="number" formControlName="age" placeholder="Введите возраст" />

        <div \*ngIf="studentForm.get('age')?.invalid && studentForm.get('age')?.touched" class="error">

          Возраст должен быть положительным числом.

        </div>

      </div>

      <button type="submit" [disabled]="studentForm.invalid">Добавить</button>

    </form>

  </div>

</div>

#app/component/ group.component.css

.group-container {

  max-width: 800px;

  margin: 0 auto;

  padding: 20px;

}

h1 {

  text-align: center;

  margin-bottom: 30px;

}

h2 {

  font-size: 1.5rem;

  margin-bottom: 10px;

}

.group-select {

  margin-bottom: 30px;

}

.group-select select {

  padding: 8px;

  font-size: 1rem;

  width: 100%;

}

.student-list {

  margin-top: 30px;

}

.student-card {

  padding: 10px;

  border: 1px solid #ddd;

  margin-bottom: 15px;

  border-radius: 8px;

  background-color: #f9f9f9;

}

.student-card button {

  padding: 5px 10px;

  margin-left: 10px;

  cursor: pointer;

}

button:hover {

  background-color: #f0f0f0;

}

button:active {

  background-color: #ddd;

}

input {

  padding: 8px;

  margin-right: 10px;

  width: 200px;

  margin-bottom: 10px;

}

input[type="number"] {

  width: 150px;

}

button {

  padding: 10px 20px;

}

#app/components/group.component.ts

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

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

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

import { GroupService } from '../services/group.service';

import { StudentService } from '../services/student.service';

import { FormsModule, ReactiveFormsModule, FormGroup, FormBuilder, Validators } from '@angular/forms';

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

import { Under18Directive } from '../directives/under18.directive';

import { BirthCenturyPipe } from '../pipes/birth-century.pipe';

@Component({

  selector: 'app-groups',

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

  imports: [CommonModule, ReactiveFormsModule, FormsModule, Under18Directive, BirthCenturyPipe],

  standalone: true,

  styleUrls: ['./group.component.css']

})

export class GroupComponent implements OnInit {

  private http = inject(HttpClient);

  groups: any[] = [];

  students: any[] = [];

  selectedGroupId: number | null = null;

  selectedGroupName: string = '';

  studentForm: FormGroup;

  selectedStudent: any = null;

  constructor(

    private groupService: GroupService,

    private studentService: StudentService,

    private route: ActivatedRoute,

    private router: Router,

    private fb: FormBuilder

  ) {

    this.studentForm = this.fb.group({

      fname: ['', [Validators.required]],

      sname: ['', [Validators.required]],

      mname: ['', [Validators.required]],

      stud: ['', [Validators.required]],

      age: ['', [Validators.required, Validators.min(0)]]

    });

  }

  ngOnInit(): void {

    this.loadGroups();

  }

  loadGroups(): void {

    this.groupService.getGroups().subscribe((data) => {

      this.groups = data;

    });

  }

  loadStudents(groupId: number): void {

    this.studentService.getStudentsByGroupId(groupId).subscribe((data) => {

      this.students = data;

    });

  }

  onGroupChange(event: Event): void {

    const selectElement = event.target as HTMLSelectElement;

    const groupId = selectElement.value;

    if (groupId) {

      this.selectedGroupId = parseInt(groupId, 10);

      this.loadStudents(this.selectedGroupId);

    }

  }

  addStudent(): void {

    if (this.studentForm.valid) {

      const newStudent = { ...this.studentForm.value, groupId: this.selectedGroupId };

      this.studentService.addStudent(newStudent).subscribe(() => {

        this.loadStudents(this.selectedGroupId!);

        this.clearStudentForm();

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

      });

    } else {

      alert('Заполните все поля корректно для добавления студента.');

    }

  }

  clearStudentForm(): void {

    this.studentForm.reset();

    this.selectedStudent = null;

  }

  startEdit(student: any): void {

    this.selectedStudent = { ...student };

  }

  saveStudent(form: any): void {

    if (form.valid && this.selectedStudent) {

      this.studentService.updateStudent(this.selectedStudent.id, this.selectedStudent).subscribe(() => {

        this.loadStudents(this.selectedGroupId!);

        this.cancelEdit();

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

      });

    }

  }

  cancelEdit(): void {

    this.selectedStudent = null;

  }

  deleteStudent(studentId: number): void {

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

      this.studentService.deleteStudent(studentId).subscribe(() => {

        this.loadStudents(this.selectedGroupId!);

      });

    }

  }

}

#app/directives/under18.directive.ts

import { Directive, ElementRef, Renderer2, Input, OnInit } from '@angular/core';

@Directive({

  selector: '[appUnder18]',

  standalone: true

})`

export class Under18Directive implements OnInit {

  @Input() studentAge: number | null = null;

  constructor(private el: ElementRef, private renderer: Renderer2) {}

  ngOnInit(): void {

    if (this.studentAge && this.studentAge < 18) {

      this.renderer.setStyle(this.el.nativeElement, 'font-style', 'italic');

    }

  }

}

#app/pipes/birth-century.pipe.ts

import { Pipe, PipeTransform } from '@angular/core';

@Pipe({

  name: 'birthCentury',

  standalone: true

})

export class BirthCenturyPipe implements PipeTransform {

  transform(age: number): string {

    if (!age || age <= 0) {

      return 'Некорректный возраст';

    }

    const currentYear = new Date().getFullYear();

    const birthYear = currentYear - age; // Вычисляем год рождения

    const century = Math.ceil(birthYear / 100); // Определяем век

    return `Родился в ${century}-м веке нашей эры`;

  }

}

#app/guards/form-unsaved.guard.ts

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

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

import { GroupComponent } from '../group/group.component';

@Injectable({

  providedIn: 'root'

})

export class FormUnsavedGuard implements CanDeactivate<GroupComponent> {

  canDeactivate(component: GroupComponent): boolean {

    // Если форма изменена и не сохранена, показываем предупреждение

    if (component.studentForm.dirty) {

      return confirm('Вы не сохранили данные. Вы уверены, что хотите покинуть эту страницу?');

    }

    return true;

  }

}

#app/app.routes.ts

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

import {GreetComponent} from "./greet/greet.component";

import {AboutComponent} from "./about/about.component";

import {GroupComponent} from "./group/group.component";

import { FormUnsavedGuard } from './guards/form-unsaved.guard';

export const routes: Routes = [

    { path: "greet", component: GreetComponent},

    { path: "about", component: AboutComponent},

    { path: "group", component: GroupComponent, canDeactivate: [FormUnsavedGuard] },

    { path: "group/:groupNumber", component: GroupComponent, canDeactivate: [FormUnsavedGuard] },

    { path: "\*\*", redirectTo: "/greet", pathMatch:"full"}

  ];

#app/interceptors/api-logger.interceptor.ts

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

import { tap } from 'rxjs/operators';

export const apiLoggerInterceptor: HttpInterceptorFn = (req, next) => {

  console.log('HTTP Request:', req);

  return next(req).pipe(

    tap({

      next: (event) => {

        console.log('HTTP Response:', event);

      },

      error: (error) => {

        console.error('HTTP Error:', error);

      }

    })

  );

};

#app/app.config.ts

import { ApplicationConfig, provideZoneChangeDetection } from '@angular/core';

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

import { routes } from './app.routes';

import { provideClientHydration } from '@angular/platform-browser';

import { provideHttpClient, withFetch, withInterceptors } from '@angular/common/http';

import { apiLoggerInterceptor } from './interceptors/api-logger.interceptor';

export const appConfig: ApplicationConfig = {

  providers: [provideZoneChangeDetection({ eventCoalescing: true }), provideRouter(routes), provideClientHydration(), provideHttpClient(withInterceptors([apiLoggerInterceptor])), provideHttpClient(withFetch())]

};