МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

“ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

КАФЕДРА ПРОГРАМНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГИЙ УПРАВЛІННЯ

«Створення клієнтського додатку до REST API»

Лабораторна робота №5

Виконав:

студент групи КН 36-В

Кулик В.В.

Перевірив:

Товстокоренко О. Ю.

Харків 2018

**Тема:** Создание клиентского приложения к REST API

**Цель:** Создание веб-приложения с использованием Angular

**Вариант: 8)**

**Задание:** Собаки, породы собак

**Ход работы:**

1. Реализован фронтенд с помощью Angular.

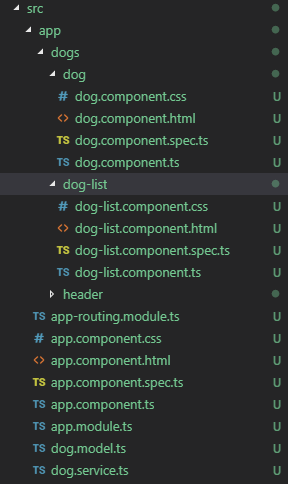


Рисунок 1 – Структура Angualr

2. Начальный интерфейс с выводом всех данных из БД.

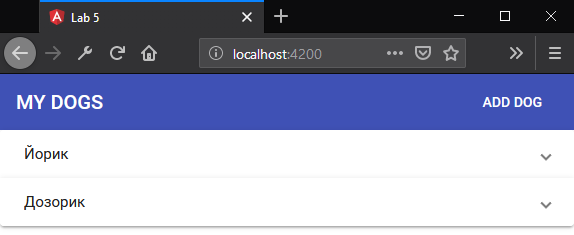


Рисунок 2 – Вывод в браузере

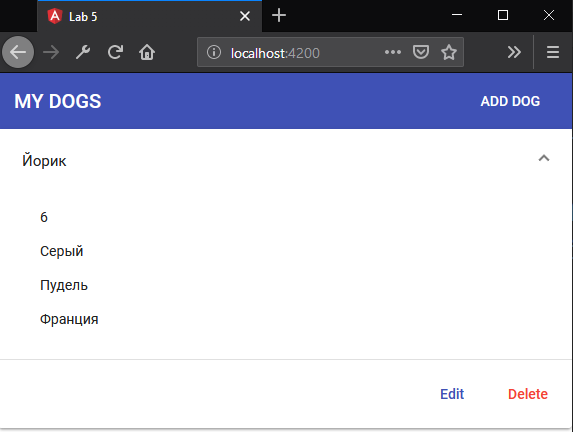


Рисунок 3 – Вывод в браузере

<mat-accordion \*ngIf="dogs.length > 0" xmlns="http://www.w3.org/1999/html">

<mat-expansion-panel \*ngFor="let dog of dogs">

<mat-expansion-panel-header>

{{dog.name}}

</mat-expansion-panel-header>

<p> {{dog.age}} </p>

<p> {{dog.color}}</p>

<p> {{dog.typedog.typedog}} </p>

<p> {{dog.typedog.size}} </p>

<mat-action-row>

<a mat-button color="primary" [routerLink]="['/edit' , dog.id]">Edit </a>

<button mat-button color="warn" (click)="onDelete(dog.id)">Delete</button>

</mat-action-row>

</mat-expansion-panel>

</mat-accordion>

<p class="mat-paginator-page-size" \*ngIf="dogs.length <=0">No dogs</p>

Рисунок 4 – Код страницы

3. Форма добавления нового элемента.

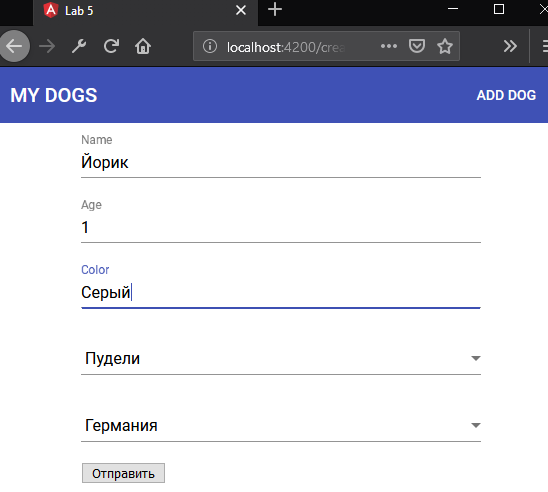


Рисунок 5 – Форма добавления элемента

<form (submit)="onSavePost(myForm)" #myForm="ngForm" class="conteinerRegist">

<mat-form-field>

<input type="text" matInput

placeholder="Name"

value=""

[ngModel]="dog?.name"

name="name">

</mat-form-field>

<mat-form-field>

<input type="text"

matInput placeholder="Age"

value=""

name="age"

[ngModel]="dog?.age">

</mat-form-field>

<mat-form-field>

<input type="text"

matInput

placeholder="Color"

name="color"

[ngModel]="dog?.color">

</mat-form-field>

<mat-form-field>

<select matNativeControl required [ngModel]="dog?.typedog.typedog" name="typedog">

<option value="Пудель">Пудели</option>

<option value="Такса">Таксы</option>

<option value="Гончая">Гончие</option>

</select>

</mat-form-field>

<mat-form-field>

<select matNativeControl required [ngModel]="dog?.typedog.size" name="size">

<option value="Франция">Франция</option>

<option value="Германия">Германия</option>

<option value="Англия">Англия</option>

</select>

</mat-form-field>

<input type="submit" value="Отправить" />

</form>

Рисунок 6 – Код формы добавления элемента

4. Вывод данных БД с добавленных элементом.

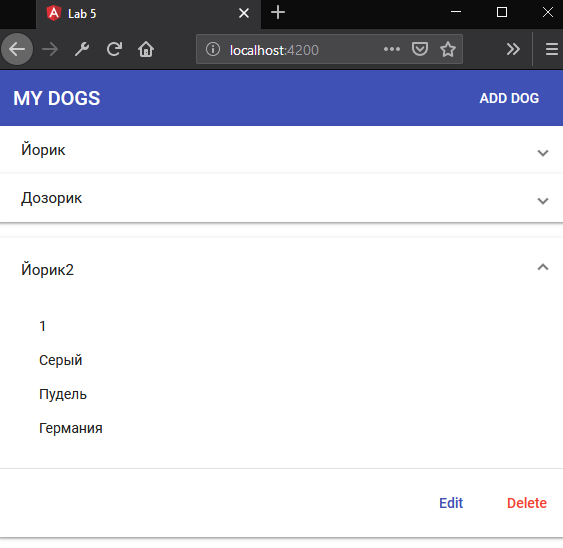


Рисунок 3 – Вывод с добавленных элементом

**Код компонента управления данными:**

import {Dog} from './dog.model';

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

import {Subject} from 'rxjs';

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

import {map} from 'rxjs/operators';

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

@Injectable({

providedIn: 'root',

})

export class DogService {

private dogs: Dog [] = [];

private dogUpdates = new Subject<Dog []>();

constructor(private http: HttpClient , private router : Router) {

}

getDogs() {

this.http.get<any>('http://localhost:3000/dogs').pipe(map(function (postData) {

return postData.map(post => {

return {

id: post.\_id,

name: post.name,

age: post.age,

color: post.color,

typeDog: {

typeDog: post.typeDog.typeDog,

size: post.typeDog.size,

}

};

});

})).subscribe((transDog) => {

this.dogs = transDog;

this.dogUpdates.next([...this.dogs]);

});

}

getDog(id: string) {

return this.http.get<Dog>('http://localhost:3000/dogs/' + id);

}

deleteDog(dogId: string) {

console.log('delete Elem');

this.http.delete('http://localhost:3000/dogs/' + dogId).subscribe(() => {

});

const newDogList = this.dogs.filter(dog => dog.id !== dogId);

this.dogs = newDogList;

this.dogUpdates.next([...this.dogs]);

}

getDogUpdateListener() {

return this.dogUpdates.asObservable();

}

addDog(dog: Dog) {

this.http.post<any>('http://localhost:3000/dogs', dog).pipe(map(function (addDog) {

return {

id: addDog.\_id,

name: addDog.name,

age: addDog.age,

color: addDog.color,

typeDog: {

typeDog: addDog.typeDog.typeDog,

size: addDog.typeDog.size,

}

};

})).subscribe((newDog) => {

this.dogs.push(newDog);

this.dogUpdates.next([...this.dogs]);

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

});

}

updatePost(updateDog: Dog) {

this.http.put<Dog>('http://localhost:3000/dogs/' + updateDog.id, updateDog).subscribe(response => {

const updatedPosts = this.dogs;

const oldPostIndex = updatedPosts.find(p => p.id === updateDog.id);

this.dogs = updatedPosts;

this.dogUpdates.next([...this.dogs]);

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

});

}

}

**Вывод:** в этой лабораторной работе была задействована работа с языком NodeJS, фрейворком Express, базой данных MongoDB и библиотекой Mongoose.