Angular. Routing.

# Contents

[Contents 1](#_Toc479749881)

[Знакомство со структурой проекта 3](#_Toc479749882)

[Усовные обозначения 3](#_Toc479749883)

[Step 01. Basic Setup. <base> Tag. 4](#_Toc479749884)

[Step 02. Components 5](#_Toc479749885)

[Step 03. Routes Config 6](#_Toc479749886)

[Step 04. Import Routes 7](#_Toc479749887)

[Step 05. <router-outlet> 8](#_Toc479749888)

[Step 06. routerLink 9](#_Toc479749889)

[Step 07. routerLinkActive 10](#_Toc479749890)

[Step 08. Task Feature Module 11](#_Toc479749891)

[Step\_09. Tasks Feature Route Configuration 15](#_Toc479749892)

[Step\_10. Register Task Feature Routing 16](#_Toc479749893)

[Step\_11. Register Tasks Feature Module 17](#_Toc479749894)

[Step\_12. Tasks List on Home Page 18](#_Toc479749895)

[Step\_13. Navigate 19](#_Toc479749896)

[Step\_14. Getting the route parameter 22](#_Toc479749897)

[Step\_15. Navigate Back 23](#_Toc479749898)

[Step\_16. Users Components 24](#_Toc479749899)

[Step\_17. Users Feature Area 29](#_Toc479749900)

[Step\_18. Users Nested Routing 30](#_Toc479749901)

[Step\_19. Relative Navigation 32](#_Toc479749902)

[Step\_20. Optional Parameters 33](#_Toc479749903)

[Step\_21. Admin Feature Area 35](#_Toc479749904)

[Step\_22. canActivate Guard 38](#_Toc479749905)

[Step\_23. Auth Service 39](#_Toc479749906)

[Step\_24. Login Component 41](#_Toc479749907)

[Step\_25. canActivateChild Guard 43](#_Toc479749908)

[Step\_26. canDeactivate Guard 44](#_Toc479749909)

[Step\_27. resolve Guard 46](#_Toc479749910)

[Step\_28. Query Parameters and Fragment 48](#_Toc479749911)

[Step\_29. Lazy-Loading Route Configuration 50](#_Toc479749912)

[Step\_30. canLoad Guard 51](#_Toc479749913)

# Знакомство со структурой проекта

>git clone <https://github.com/VZhyrytskiy/An2-4-Routing.git>

# Усовные обозначения

**Черный цвет** – фрагмент кода, который необходимо полностью использовать для создания нового файла, а в сочетании с зеленым или красным – фрагмент кода, который был добавлен раньше.

**Зеленый цвет** – фрагмент кода, который необходимо добавить.

**Красный цвет** – фрагмент кода, который необходимо удалить.

# Step 01. Basic Setup. <base> Tag.

Добавте тег base в файле **srs/index.html**

<title>Angular 2: Routing</title>

**<base href="/">**

<meta name="viewport" content="width=device-width, initial-scale=1">

Если нет доступа к тегу <head> то можно добаить программно в **app/app.module.ts**

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

**import {APP\_BASE\_HREF} from '@angular/common';**

@NgModule({

**providers: [{provide: APP\_BASE\_HREF, useValue: '/my/app'}]**

})

class AppModule {}

# Step 02. Components

Создайте папку **app/components** и перейдите в нее в командной строке.

1. Создайте три компонента-заглушки и удалите селекторы у этих компонентов[[1]](#footnote-1):

* HomeComponent (>ng g c home -spec=false)
* AboutComponent (>ng g c about -spec=false)
* PageNotFoundComponent (>ng g c page-not-found -spec=false)

-spec=false – обозначает не генерить файл теста.

При этом в файл **app.module.ts** будет автоматически добавлен следующий фрагмент кода:

import { AboutComponent } from './components/about/about.component';

import { HomeComponent } from './components/home/home.component';

import { PageNotFoundComponent } from './components/page-not-found/page-not-found.component';

@NgModule({

declarations: [

TodoAppComponent,

AboutComponent,

HomeComponent,

PageNotFoundComponent

],

...

1. Создайте файл **app/components/index.ts** и добавте следующий фрагмент кода

export \* from './about/about.component';

export \* from './home/home.component';

export \* from './page-not-found/page-not-found.component';

1. Внесите изменения в файл **app.module.ts**

import { AboutComponent } from './components/about/about.component';

import { HomeComponent } from './components/home/home.component';

import { PageNotFoundComponent } from './components/page-not-found/page-not-found.component';

import { AboutComponent, HomeComponent, PageNotFoundComponent } from './components';

# Step 03. Routes Config

Создайте файл **app/app.routing.module.ts[[2]](#footnote-2)**. Добавьте в него следующий фрагмент кода.

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

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

import { AboutComponent, HomeComponent, PageNotFoundComponent } from './components';

const routes: Routes = [

{

path: 'home',

component: HomeComponent

},

{

path: 'about',

component: AboutComponent

},

{

path: '',

redirectTo: '/home',

pathMatch: 'full'

},

{

// The router will match this route if the URL requested

// doesn't match any paths for routes defined in our configuration

path: '\*\*',

component: PageNotFoundComponent

}

];

export let appRouterComponents = [AboutComponent, HomeComponent, PageNotFoundComponent];

@NgModule({

imports: [

RouterModule.forRoot(routes)

],

exports: [RouterModule]

})

export class AppRoutingModule {}

# Step 04. Import Routes

Внесите изменения в файл **app.module.ts**:

// 1

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

import { AppRoutingModule, appRouterComponents } from './app.routing.module';

// 2

Добавьте импортированный модуль **AppRoutingModule** в раздел imports

imports: [

BrowserModule,

FormsModule,

AppRoutingModule

]

// 3

Добавьте **appRouterComponents** в секцию декларирования компонентов

declarations: [

TodoAppComponent,

appRouterComponents

AboutComponent,

HomeComponent,

PageNotFoundComponent

],

// 4

Удалите импорт

import { AboutComponent, HomeComponent, PageNotFoundComponent } from './components';

// 5

Добавьте конструктор классу

constructor(router: Router) {

console.log('Routes: ', JSON.stringify(router.config, undefined, 2));

}

# Step 05. <router-outlet>

1. В файле **app.component.html** добавьте директиву router-outlet

<div class="container">

<router-outlet></router-outlet>

<!-- Routed views go here -->

</div>

# Step 06. routerLink

1. В файле **app.component.html** добавьте директиву **routerLink**

// 1

<a class="navbar-brand" **routerLink="/"**>Task Manager</a>

// 2

<a **routerLink="/about"**>About</a>

# Step 07. routerLinkActive

В файле **app.component.html** добавьте директиву **routerLinkActive**

<li **routerLinkActive="active"**>

<a [routerLink]="[**'**about**']**">About</a>

</li>

# Step 08. Task Feature Module

1. Создайте папку **models**
2. Создайте файл **models/task.ts** следующего содержания

export class Task {

constructor(

public id: number,

public action: string,

public priority: number,

public estHours: number,

public actHours?: number,

public done?: boolean

) {

this.actHours = 0;

this.done = false;

}

}

1. Создайте модуль **tasks/tasks.module.ts** выполнив следующую команду из командной строки   
   …/app> **ng g m tasks**
2. Внесите изменения в модуль, используя следующий фрагмент кода:

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

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

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

@NgModule({

declarations: [

],

imports: [

CommonModule,

FormsModule

],

providers: [

]

})

export class TasksModule {}

1. Создайте папку **tasks/services**
2. Создайте файл **task-array.service.ts** (**ng g s task-array**) следующего содержания

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

import 'rxjs/add/operator/toPromise';

import { Task } from './../../models/task';

let taskList = [

new Task(1, **'**Estimate**'**, 1, 8, 8, true),

new Task(2, **'**Create**'**, 2, 8, 4, false),

new Task(3, **'**Deploy**'**, 3, 8, 0, false)

];

let taskListPromise = Promise.resolve(taskList);

@Injectable()

export class TaskArrayService {

getTasks(): Promise<Task[]> {

return taskListPromise;

}

getTask(id: number | string): Promise<Task> {

return this.getTasks()

.then(tasks => tasks.find(task => task.id === +id))

.catch(() => Promise.reject('Error in getTask method'));;

}

addTask(task: Task): void {

taskList.push(task);

}

updateTask(task: Task): void {

let i = -1;

taskList.forEach((item, index) => {

if (item.id === task.id ) {

i = index;

return false;

}

});

if (i > -1) {

taskList.splice(i, 1, task);

}

}

}

1. Внесите изменения в модуль, используя следующий фрагмент кода:

import { TaskListComponent, TaskComponent, TaskFormComponent, TaskArrayService } from '.';

@NgModule({

providers: [

TaskArrayService

]

})

export class TasksModule {}

1. В папке **tasks** cоздайте компонент **TaskListComponent** (ng g c task-list) следующего содержания:

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

import { Task } from './../../models/task';

import { TaskArrayService } from './../services/task-array.service';

@Component({

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

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

})

export class TaskListComponent implements OnInit {

tasks: Array<Task>;

constructor(

private tasksService: TaskArrayService) { }

ngOnInit() {

console.log(this.tasks);

this.tasksService.getTasks()

.then(tasks => this.tasks = tasks)

.catch((err) => console.log(err));

}

completeTask(task: Task): void {

task.done = true;

}

}

Проверьте, чтобы компонент был зарегистрирован в модуле **tasks/tasks.module.ts**

1. Добавьте в файл темплейта **task-list.component.html** следующую разметку:

<task

\*ngFor="let task of tasks"

[task]="task"

(onComplete)="completeTask($event)">

</task>

1. В папке **tasks** создайте компонент **TaskComponent** (ng g c task) следующего содержания:

import { Component, EventEmitter, Input, Output } from '@angular/core';

import { Task } from './../../models/task';

@Component({

selector: 'task',

templateUrl: 'task.component.html',

styleUrls: ['task.component.css']

})

export class TaskComponent {

@Input() task: Task;

@Output() onComplete = new EventEmitter<Task>();

constructor() { }

completeTask(event: any): void {

this.onComplete.emit(this.task);

}

editTask() {

}

}

Проверьте, чтобы компонент был зарегистрирован в модуле **tasks/tasks.module.ts**

1. Добавьте в файл темплейта **task.component.html** следующую разметку:

<div class="panel panel-default">

<div class="panel-heading">Task</div>

<div class="panel-body">

<ul>

<li>Action: {{task.action}}</li>

<li>Priority: {{task.priority}}</li>

<li>Estimate Hours: {{task.estHours}}</li>

<li>Actual Hours: {{task.actHours}}</li>

<li>Done: {{task.done}}</li>

</ul>

<button class="btn btn-primary"

(click)="completeTask($event)">

Done

</button>

<button class="btn btn-warning btn-sm"

(click)="editTask()">

Edit

</button>

</div>

</div>

1. Создайте файл **tasks/index.ts** следующего содержания:

export \* from './services/task-array.service';

export \* from './task/task.component';

export \* from './task-list/task-list.component';

1. Внесите изменения в файл **tasks.module.ts**

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

import { TaskComponent } from './task/task.component';

import { TaskArrayService } from './services/task-array.service';

import { TaskListComponent, TaskComponent, TaskArrayService } from '.';

# Step\_09. Tasks Feature Route Configuration

1. Создайте файл **tasks/tasks.routing.module.ts** следующего содержания:

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

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

import { TaskListComponent } from '.';

const routes: Routes = [

{

path: 'task-list',

component: TaskListComponent

}

];

@NgModule({

imports: [

RouterModule.forChild(routes)

],

exports: [RouterModule]

})

export class TasksRoutingModule { }

# Step\_10. Register Task Feature Routing

1. Добавьте в файл **tasks/tasks.module.ts** следующий фрагмент кода

import { TasksRoutingModule } from './tasks.routing.module';

1. Внесите изменения в следующий фрагмент кода:

imports: [

CommonModule,

FormsModule,

TasksRoutingModule

]

# Step\_11. Register Tasks Feature Module

1. Добавьте в файл **app.module.ts** следующий фрагмент кода:

import { TasksModule } from './tasks/tasks.module';

1. Внесите изменения в следующий фрагмент кода:

imports: [

BrowserModule,

CommonModule,

FormsModule,

TasksModule,

AppRoutingModule

],

# Step\_12. Tasks List on Home Page

1. Внесите изменения в файл **tasks/tasks.routing.module.ts**

const tasksRoutes: Routes = [

{

**~~path: 'task-list',~~**

**path: 'home',**

component: TaskListComponent

}

];

1. Внесите изменения в файл **app.routing.module.ts**

// 1

import { AboutComponent, HomeComponent, PageNotFoundComponent } from './components';

{

path: 'home',

component: HomeComponent

},

// 2

export let appRouterComponents = [AboutComponent, HomeComponent, PageNotFoundComponent];

1. Внесите изменения в файл **app/components/index.ts**

export \* from './about/about.component';

export \* from './home/home.component';

export \* from './page-not-found/page-not-found.component';

1. Удалите компонент **HomeComponent** (папка components/home)

# Step\_13. Navigate

1. Создайте компонент **TaskFormComponent** в папке **tasks** (ng g c task-form) используя следующий фрагмент кода:

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

import { Task } from './../../models/task';

import { TaskArrayService } from './../services/task-array.service';

@Component({

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

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

})

export class TaskFormComponent implements OnInit, OnDestroy {

task: Task;

constructor(

private tasksService: TaskArrayService,

) { }

ngOnInit(): void {

this.task = new Task(null, '', null, null);

}

ngOnDestroy(): void {

}

saveTask() {

let task = new Task(

this.task.id,

this.task.action,

this.task.priority,

this.task.estHours

);

if (task.id) {

this.tasksService.updateTask(task);

}

else {

this.tasksService.addTask(task);

}

}

goBack(): void {

}

}

1. Создайте темплейт для компонента **TaskFormComponent** использу следующий фрагмент разметки

<div class="panel panel-default">

<div class="panel-heading">

<h4 class="pannel-title">

Task Form

</h4>

</div>

<div class="panel-body">

<form \*ngIf="task" (ngSubmit)="saveTask()" id="task-form">

<div class="form-group">

<label for="action">Action</label>

<input type="text"

class="form-control"

id="action" name="action"

placeholder="Action"

required

[(ngModel)]="task.action">

</div>

<div class="form-group">

<label for="priority">Priority</label>

<input type="number"

min="1" max="3"

class="form-control"

id="priority" name="priority"

placeholder="Priority"

[(ngModel)]="task.priority">

</div>

<div class="form-group">

<label for="estHours">Est. Hours</label>

<input type="number"

min="0"

step="2"

class="form-control"

id="estHours" name="estHours"

placeholder="Est. Hours"

[(ngModel)]="task.estHours">

</div>

<button

type="submit"

class="btn btn-primary"

form="task-form">Save

</button>

<button

class="btn btn-primary"

(click)="goBack()">Back

</button>

</form>

</div>

</div>

1. Внесите изменения в файл **tasks/index.ts**

export \* from './task/task.component';

export \* from './task-form/task-form.component';

export \* from './task-list/task-list.component';

1. Внесите изменения в файл **tasks/tasks.module.ts**

// 1

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

import { TaskListComponent, TaskComponent, TaskFormComponent, TaskArrayService } from '.';

1. Внесите изменения в файл **tasks/tasks.routing.module.ts**

// 1

import { TaskListComponent, TaskFormComponent } from '.';

// 2

const routes: Routes = [

{

path: 'home',

component: TaskListComponent

},

**{**

**path: 'edit/:id',**

**component: TaskFormComponent**

**}**

];

1. Внесите изменения в компонент **TaskComponent**

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

constructor(

**private router: Router**

) { }

editTask() {

**let link = ['/edit', this.task.id];**

**this.router.navigate(link);**

}

# Step\_14. Getting the route parameter

1. Внесите изменения в компонент **TaskFormComponent** в файле **tasks/task-form.component.ts**

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

**import 'rxjs/add/operator/switchMap';**

1. Внесите изменения в конструктор

constructor(

private tasksService: TaskArrayService,

**private route: ActivatedRoute**

) { }

1. Внесите изменения в ngOnInit

ngOnInit(): void {

this.task = new Task(null, '', null, null);

// it is not necessary to save subscription to route.params

// it handles automatically

this.route.params

.switchMap((params: Params) => this.tasksService.getTask(+params['id']))

.subscribe(

task => this.task = Object.assign({}, task),

err => console.log(err)

);

}

# Step\_15. Navigate Back

1. Внесите изменения в компонент TaskFormComponent в файле **tasks/task-form.component.ts**

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

1. Внесите изменения в конструктор

constructor(

private tasksService: TaskArrayService,

**private router: Router,**

private route: ActivatedRoute

) { }

1. Внесите изменения в метод goBack()

goBack(): void {

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

}

1. Внесите изменения в метод **saveTask()**

if (task.id) {

this.tasksService.updateTask(task);

}

else {

this.tasksService.addTask(task);

}

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

# Step\_16. Users Components

1. создайте файл **models/user.ts** используя слудющий фрагмент кода:

export class User {

constructor(

public id: number,

public firstName: string,

public lastName: string

) {}

}

1. Создайте файл **users/users.module.ts** (**ng g m users** - команду запустить в папке app) используя следующую разметку

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

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

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

@NgModule({

imports: [

CommonModule,

FormsModule,

],

declarations: [

],

providers: [

]

})

export class UsersModule {}

1. Создайте сервис **UserArrayService** в файле **users/services/user-array.service.ts** используя следующий фрагмент кода

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

import { User } from './../../models/user';

let userList = [

new User(1, 'Anna', 'Borisova'),

new User(2, 'Boris', 'Vlasov'),

new User(3, 'Gennadiy', 'Dmitriev')

];

let userListPromise = Promise.resolve(userList);

@Injectable()

export class UserArrayService {

getUsers(): Promise<User[]> {

return userListPromise;

}

getUser(id: number | string): Promise<User> {

return this.getUsers()

.then(users => users.find(user => user.id === +id))

.catch(() => Promise.reject('Error in getUser method'));

}

addUser(user: User): void {

userList.push(user);

}

updateUser(user: User): void {

let i = -1;

userList.forEach((item, index) => {

if (item.id === user.id ) {

i = index;

return false;

}

});

if (i > -1) {

userList.splice(i, 1, user);

}

}

}

1. Внесите изменения в файл **users/users.module.ts** используя следующий фрагмент кода

import { UserListComponent, UserComponent, UserFormComponent, UserArrayService } from '.';

@NgModule({

providers: [

UserArrayService,

]

})

1. Создайте компонент **UserListComponent** (**ng g c user-list** )используя следующий код

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

import { User } from './../../models/user';

import { UserArrayService } from './../services/user-array.service';

@Component({

templateUrl: 'user-list.component.html',

styleUrls: ['user-list.component.css']

})

export class UserListComponent implements OnInit, OnDestroy {

users: Array<User>;

constructor(

private usersService: UserArrayService,

) { }

ngOnInit() {

this.usersService.getUsers()

.then(users => this.users = users)

.catch((err) => console.log(err));

}

ngOnDestroy() {

}

}

1. Создайте разметку для **UserListComponent** в файле **user-list.component.html** используя следующий фрагмент разметки

<user

\*ngFor="let user of users"

[user]="user">

</user>

1. Создайте компонент **UserComponent** (**ng g c user**) используя слудующий фрагмент кода

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

import { User } from './../../models/user';

@Component({

selector: 'user',

templateUrl: 'user.component.html',

styleUrls: ['user.component.css']

})

export class UserComponent {

@Input() user: User;

constructor(

) { }

editUser() {

}

}

1. Создайте разметку для **UserComponent** в файле **user.component.html** используя следующий фрагмент разметки

<div class="panel panel-default">

<div class="panel-heading">User</div>

<div class="panel-body">

<ul>

<li>FirtsName: {{user.firstName}}</li>

<li>LastName: {{user.lastName}}</li>

</ul>

<button class="btn btn-warning btn-sm"

(click)="editUser()">

Edit

</button>

</div>

</div>

1. Создайте компонент **UserFormComponent** (**ng g c user-form**) использу следующий фрагмент кода

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

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

import { User } from './../../models/user';

import { UserArrayService } from './../services/user-array.service';

@Component({

templateUrl: 'user-form.component.html',

styleUrls: ['user-form.component.css'],

})

export class UserFormComponent implements OnInit, OnDestroy {

user: User;

oldUser: User;

constructor(

private usersService: UserArrayService,

private route: ActivatedRoute,

) { }

ngOnInit(): void {

this.user = new User(null, '', '');

this.route.params

.switchMap((params: Params) => this.usersService.getUser(+params['id']))

.subscribe(

user => {

this.user = Object.assign({}, user);

this.oldUser = user;

},

err => console.log(err)

);

}

ngOnDestroy(): void {

}

saveUser() {

const user = new User(

this.user.id,

this.user.firstName,

this.user.lastName

);

if (user.id) {

this.usersService.updateUser(user);

// if success

this.oldUser = this.user;

}

else {

this.usersService.addUser(user);

// if success

this.oldUser = this.user;

}

}

goBack() {

}

}

}

1. Создайте разметку для **UserFormComponent** в файле **user-form.component.html** используя следующий фрагмент разметки

<div class="panel panel-default">

<div class="panel-heading">

<h4 class="pannel-title">

User Form

</h4>

</div>

<div class="panel-body">

<form \*ngIf="user" (ngSubmit)="saveUser()" id="user-form">

<div class="form-group">

<label for="firstName">First Name</label>

<input type="text"

class="form-control"

id="firstName" name="firstName"

placeholder="First Name"

required

[(ngModel)]="user.firstName">

</div>

<div class="form-group">

<label for="lastName">Last Name</label>

<input type="text"

class="form-control"

id="lastName" name="lastName"

placeholder="Last Name"

[(ngModel)]="user.lastName">

</div>

<button

type="submit"

class="btn btn-primary"

form="user-form">Save

</button>

<button class="btn btn-primary"

type="button"

(click)="goBack()">Back

</button>

</form>

</div>

</div>

1. Создайте файл **users/index.ts** следующего содержания

export \* from './services/user-array.service';

export \* from './user/user.component';

export \* from './user-form/user-form.component';

export \* from './user-list/user-list.component';

1. Внесите изменения в файл **users.module.ts**

import { UserArrayService } from './services/user-array.service';

import { UserListComponent } from './user-list/user-list.component';

import { UserComponent } from './user/user.component';

import { UserFormComponent } from './user-form/user-form.component';

import { UserListComponent, UserComponent, UserFormComponent, UserArrayService } from '.';

# Step\_17. Users Feature Area

1. Создайте компонент **UsersComponent** в папке **users (ng g c users** --flat**)** используя следующий фрагмент кода

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

@Component({

selector: 'users',

templateUrl: 'users.component.html',

styleUrls: ['users.component.css']

})

export class UsersComponent implements OnInit {

constructor() { }

ngOnInit() {

}

}

1. Создайте разметку для **UsersComponent** в файле **users/users.component.html** используя следующий фрагмент разметки

<h2>Users</h2>

1. Внесите изменения в файл **users/index.ts**

export \* from './users.component';

export \* from './user/user.component';

1. Внесите изменения в файл users/users.module.ts

import { UsersComponent } from './users.component';

import { UsersComponent, UserListComponent, UserComponent, UserFormComponent, UserArrayService } from '.';

1. Внесите изменения в файл **app.module.ts**

// 1

import { TasksModule } from './tasks/tasks.module';

**import { UsersModule } from './users/users.module';**

// 2

imports: [

BrowserModule,

CommonModule,

FormsModule,

TasksModule,

UsersModule,

AppRoutingModule

],

# Step\_18. Users Nested Routing

1. Внесите изменения в файл **app.component.html**

<div>

<ul class="nav navbar-nav">

**<li routerLinkActive="active">**

**<a routerLink="/users">Users</a>**

**</li>**

</ul>

</div>

1. Внесите изменения в файл **users/users.component.html**

<h2>Users</h2>

**<router-outlet></router-outlet>**

1. Создайте файл роутинга **users/users.routing.module.ts** использу следующий фрагмент кода

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

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

import { UsersComponent, UserListComponent, UserFormComponent } from '.';

const routes: Routes = [

{

path: 'users',

component: UsersComponent,

children: [

{

path: 'add',

component: UserFormComponent

},

{

path: 'edit/:id',

component: UserFormComponent,

},

{

path: '',

component: UserListComponent

},

]

}

];

export let usersRouterComponents = [UsersComponent, UserListComponent, UserFormComponent];

@NgModule({

imports: [

RouterModule.forChild(routes)

],

exports: [RouterModule]

})

export class UsersRoutingModule { }

1. Внесите изменения в файл **users/users.module.ts**

import { UsersRoutingModule, usersRouterComponents } from './users.routing.module';

import { UsersComponent, UserListComponent, UserComponent, UserFormComponent, UserArrayService } from '.';

imports: [

CommonModule,

FormsModule,

**UsersRoutingModule**

]

declarations: [

UsersComponent,

UserListComponent,

UserFormComponent,

usersRouterComponents,

UserComponent,

],

# Step\_19. Relative Navigation

1. Внесите изменения в **UserComponent**

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

1. Внесите изменения в конструктор

constructor(

**private router: Router,**

**private route: ActivatedRoute**

) { }

1. Внесите изменения в метод

editUser() {

**const link = ['/users/edit', this.user.id];**

**this.router.navigate(link);**

**// or**

**// const link = ['edit', this.user.id];**

**// this.router.navigate(link, {relativeTo: this.route});**

}

1. Внесите изменения в компонент **UserFormComponent**

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

1. Внесите изменения в конструктор

constructor(

private usersService: UserArrayService,

private route: ActivatedRoute,

**private router: Router**

) { }

1. Внесите изменения в метод **saveUser**

if (user.id) {

this.usersService.updateUser(user);

// if success

this.oldUser = this.user;

}

else {

this.usersService.addUser(user);

// if success

this.oldUser = this.user;

}

**this.router.navigate(['./../../'], { relativeTo: this.route});**

1. Внесите изменения в метод **goBack**

goBack() {

**this.router.navigate(['./../../'], { relativeTo: this.route});**

}

# Step\_20. Optional Parameters

1. Внесите изменения в метод **saveUser** компонента **UserFormComponent**

if (user.id) {

this.usersService.updateUser(user);

this.oldUser = this.user;

**this.router.navigate(['/users', {id: user.id}]);**

}

else {

this.usersService.addUser(user);

this.oldUser = this.user;

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

}

**~~this.router.navigate(['./../../'], { relativeTo: this.route});~~**

1. Внесите изменения в компонент **UserListComponent**

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

**import 'rxjs/add/operator/switchMap';**

1. Добавьте свойство

**private editedUser: User;**

1. Внесите изменения в конструктор

constructor(

private usersService: UserArrayService,

**private route: ActivatedRoute**

) { }

1. Внесите изменения в метод ngOnInit

ngOnInit() {

this.usersService.getUsers()

.then(users => this.users = users)

.catch((err) => console.log(err));

**// listen id from UserFormComponent**

**this.route.params**

**.switchMap((params: Params) => this.usersService.getUser(+params['id']))**

**.subscribe(**

**(user: User) => {**

**this.editedUser = Object.assign({}, user);**

**console.log(`Last time you edit user ${JSON.stringify(this.editedUser)}`);**

**},**

**(err) => console.log(err)**

**);**

}

1. Добавьте метод

isEdited(user: User) {

if (this.editedUser) {

return user.id === this.editedUser.id;

}

return false;

}

1. Внесите изменения в разметку компонента **UserListComponent** , используя следующий фрагмент разметки:

<user

\*ngFor='let user of users'

[user]="user"

[class.edited]="isEdited(user)">

</user>

1. Добавьте правила CSS в файл стилей для компонента UserComponent, используя следующий фрагмент

:host.edited > div {

border: 2px dotted red;

}

# Step\_21. Admin Feature Area

1. Создайте файл **admin/admin.module.ts** (**ng g m admin --flat** из папки app) используя следующий фрагмент кода

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

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

@NgModule({

imports: [

CommonModule,

],

declarations: [

]

})

export class AdminModule {}

1. сгенерируйте три компонента выполнив команду **ng g c component-name** в папке **admin:**
   1. AdminDashboardComponent (ng g c admin-dashboard)
   2. ManageTasksComponent (ng g c manage-tasks)
   3. ManageUsersComponent (ng g c manage-users)
2. сгенерируйте компонент **AdminComponent** выполнив команду **ng g c admin --flat** в папке **admin**
3. Удалите селекторы у этих компоннтов
4. Создайте темплейт для компонента **AdminComponent** используя следующий фрагмент разметки

<h3>Admin</h3>

<nav>

<ul class="nav nav-tabs">

<li routerLinkActive="active" [routerLinkActiveOptions]="{ exact: true }">

<a routerLink="./">Dashboard</a>

</li>

<li routerLinkActive="active">

<a routerLink="./tasks">Manage Tasks</a>

</li>

<li routerLinkActive="active">

<a routerLink="./users">Manage Users</a>

</li>

</ul>

</nav>

<router-outlet></router-outlet>

1. Создайте файл **admin/index.ts** используя следующий фрагмент кода

export \* from './admin.component';

export \* from './admin-dashboard/admin-dashboard.component';

export \* from './manage-tasks/manage-tasks.component';

export \* from './manage-users/manage-users.component';

1. Создайте файл **admin.routing.module.ts** используя следующий фрагмент кода

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

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

import { AdminComponent, AdminDashboardComponent, ManageTasksComponent, ManageUsersComponent } from '.';

const routes: Routes = [

{

path: 'admin',

component: AdminComponent,

children: [

{

path: '',

children: [

{ path: 'users', component: ManageUsersComponent },

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

{ path: '', component: AdminDashboardComponent }

]

}

]

}

];

export let adminRouterComponents = [AdminComponent, AdminDashboardComponent, ManageTasksComponent, ManageUsersComponent];

@NgModule({

imports: [

RouterModule.forChild(routes)

],

exports: [RouterModule]

})

export class AdminRoutingModule { }

1. Внесите измениния в файл **admin/admin.module.ts**

// 1

import { AdminComponent } from './admin.component';

import { AdminDashboardComponent } from './admin-dashboard/admin-dashboard.component';

import { ManageTasksComponent } from './manage-tasks/manage-tasks.component';

import { ManageUsersComponent } from './manage-users/manage-users.component';

import { AdminComponent, AdminDashboardComponent, ManageTasksComponent, ManageUsersComponent } from '.';

// 2

import { AdminRoutingModule } from './admin.routing.module';

// 3

imports: [

CommonModule,

AdminRoutingModule

]

1. Внесите изменения в файл **app.module.ts**

import { TasksModule } from './tasks/tasks.module';

import { UsersModule } from './users/users.module';

**import { AdminModule } from './admin/admin.module';**

imports: [

BrowserModule,

CommonModule,

FormsModule,

TasksModule,

UsersModule,

AdminModule,

AppRoutingModule

],

1. Внесите изменения в файл **app.component.html**

<li routerLinkActive="active">

<a routerLink="/users">Users</a>

</li>

**<li routerLinkActive="active">**

**<a routerLink="/admin">Admin</a>**

**</li>**

# Step\_22. canActivate Guard

1. Создайте защитника AuthGuard в файле **guards/auth.guard.ts** используя следующий фрагмент кода

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

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

@Injectable()

export class AuthGuard implements CanActivate {

canActivate() {

console.log('CanActivateGuard is called');

return true;

}

}

1. Внесите изменения в файл **app.module.ts**

**import { AuthGuard } from './guards/auth.guard';**

**providers: [**

**AuthGuard**

**]**

1. Внесите изменения в файл **admin/admin.routing.module.ts**

**import { AuthGuard } from './../guards/auth.guard';**

const adminRoutes: Routes = [

{

path: 'admin',

component: AdminComponent,

**canActivate: [AuthGuard]**

children: [

{

path: '',

children: [

{ path: 'users', component: ManageUsersComponent },

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

{ path: '', component: AdminDashboardComponent }

]

}

]

}

];

# Step\_23. Auth Service

1. Создайте файл **services/rxjs-extensions.ts** используя следующий фрагмент кода

// Observable class extensions

import 'rxjs/add/observable/of';

import 'rxjs/add/observable/throw';

// Observable operators

import 'rxjs/add/operator/catch';

import 'rxjs/add/operator/debounceTime';

import 'rxjs/add/operator/delay';

import 'rxjs/add/operator/distinctUntilChanged';

import 'rxjs/add/operator/do';

import 'rxjs/add/operator/filter';

import 'rxjs/add/operator/map';

import 'rxjs/add/operator/switchMap';

1. Создайте сервис **AuthService** в файле **services/auth.service.ts** используя следующий фрагмент кода

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

import { Observable } from 'rxjs/Observable';

import './rxjs-extensions';

@Injectable()

export class AuthService {

isLoggedIn = false;

// store the URL so we can redirect after logging in

redirectUrl: string;

login(): Observable<boolean> {

return Observable.of(true).delay(1000).do(val => this.isLoggedIn = true);

}

logout(): void {

this.isLoggedIn = false;

}

}

1. Внесите изменения в файл **guards/auth.guard.ts**

import {

CanActivate, **Router,**

**ActivatedRouteSnapshot,**

**RouterStateSnapshot**

} from '@angular/router';

**import { AuthService } from './../services/auth.service';**

1. Добавьте конструктор

**constructor(**

**private authService: AuthService,**

**private router: Router**

**) {}**

1. Добавьте метод checkLogin()

**checkLogin(url: string): boolean {**

**if (this.authService.isLoggedIn) { return true; }**

**// Store the attempted URL for redirecting**

**this.authService.redirectUrl = url;**

**// Navigate to the login page**

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

**return false;**

**}**

1. Внесите изменения в метод **canActivate**

canActivate(**route: ActivatedRouteSnapshot, state: RouterStateSnapshot**): boolean {

**let url: string = state.url;**

**return this.checkLogin(url);**

**~~console.log('CanActivateGuard is called');~~**

**~~return true;~~**

}

1. Внесите изменения в файл **app.module.ts**

import { AuthGuard } from './guards/auth.guard';

**import { AuthService } from './services/auth.service';**

providers: [

AuthGuard,

**AuthService**

]

# Step\_24. Login Component

1. Сгенерируйте новый компонент **LoginComponent** в папке components, выполнив команду **ng g c login**
2. Удалите селектор этого компонента
3. Внесите изменения в файл **components/index.ts** используя следующий фрагмент кода

export \* from './about/about.component';

export \* from './login/login.component';

export \* from './page-not-found/page-not-found.component';

1. Внесите изменения в файл **app.module.ts**,

**import { LoginComponent } from './components/login/login.component';**

declarations: [

TodoAppComponent,

LoginComponent,

appRouterComponents

],

1. Внесите изменения в файл **app.routing.module.ts**

// 1

import { AboutComponent, PageNotFoundComponent, LoginComponent } from './components';

// 2

{

path: 'about',

component: AboutComponent

},

**{**

**path: 'login',**

**component: LoginComponent**

**},**

// 3

export let appRouterComponents = [AboutComponent, PageNotFoundComponent, LoginComponent];

1. Внесите измнения в **LoginComponent**

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

**import { AuthService } from './../../services/auth.service';**

1. Добавьте свойство

**message: string;**

1. Добавьте методы

setMessage() {

this.message = 'Logged ' + (this.authService.isLoggedIn ? 'in' : 'out');

}

login() {

this.message = 'Trying to log in ...';

this.authService.login().subscribe(() => {

this.setMessage();

if (this.authService.isLoggedIn) {

// Get the redirect URL from our auth service

// If no redirect has been set, use the default

let redirect = this.authService.redirectUrl ? this.authService.redirectUrl : '/admin';

// Redirect the user

this.router.navigate([redirect]);

}

});

}

logout() {

this.authService.logout();

this.setMessage();

}

1. Внесите изменения в конструктор **LoginComponent**

constructor(

**public authService: AuthService,**

**public router: Router**

) {

**this.setMessage();**

}

1. Добавьте разметку для компонента **LoginComponent** импользуя следующий фрагмент разметки

<h2>LOGIN</h2>

<p>State: {{message}}</p>

<p>

<button (click)="login()" \*ngIf="!authService.isLoggedIn">Login</button>

<button (click)="logout()" \*ngIf="authService.isLoggedIn">Logout</button>

</p>

1. Внесите изменения в темплейт **TodoAppComponent**

<li routerLinkActive="active">

<a routerLink="/admin">Admin</a>

</li>

**<li routerLinkActive="active">**

**<a routerLink="/login">Login</a>**

**</li>**

# Step\_25. canActivateChild Guard

1. Внесите изменения в **guards/auth.guard.ts**

import {

CanActivate, **CanActivateChild**, Router,

ActivatedRouteSnapshot,

RouterStateSnapshot

} from '@angular/router';

export class AuthGuard implements CanActivate, **CanActivateChild** {

…

}

1. Добавьте метод **canActivateChild**

**canActivateChild(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): boolean {**

**console.log('canActivateChild Guard is called');**

**return this.canActivate(route, state);**

**}**

1. Внесите изменения в файл **admin/admin.routing.module.ts**

{

path: '',

**canActivateChild: [AuthGuard],**

children: [

{ path: 'users', component: ManageUsersComponent },

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

{ path: '', component: AdminDashboardComponent }

]

}

# Step\_26. canDeactivate Guard

1. Создайте новый сервис в файле **services/dialog.service.ts** используя следующий фрагмент кода

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

@Injectable()

export class DialogService {

confirm(message?: string) {

return new Promise<boolean>(resolve => {

resolve(window.confirm(message || 'Is it OK?'));

});

};

}

1. Внесите изменения в файл **app.module.ts**

import { DialogService } from './services/dialog.service';

providers: [

AuthGuard,

AuthService,

**DialogService**

],

1. Создайте CanDeactivateGuard в файле **guards/can-deactivate.guard.ts** используя следующий фрагмент кода

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

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

import { Observable } from 'rxjs/Observable';

export interface CanComponentDeactivate {

canDeactivate: () => Observable<boolean> | Promise<boolean> | boolean;

}

@Injectable()

export class CanDeactivateGuard implements CanDeactivate<CanComponentDeactivate> {

canDeactivate(component: CanComponentDeactivate) {

return component.canDeactivate ? component.canDeactivate() : true;

}

}

1. Внесите изменения в компонент **UserFormComponent**

**import { DialogService } from './../../services/dialog.service';**

constructor(

private usersService: UserArrayService,

private route: ActivatedRoute,

private router: Router,

**public dialogService: DialogService**

) { }

**canDeactivate(): Promise<boolean> | boolean {**

**if (!this.oldUser || this.oldUser.firstName === this.user.firstName) {**

**return true;**

**}**

**return this.dialogService.confirm('Discard changes?');**

**}**

1. Внесите изменения в файл **users/users.module.ts**

**import { CanDeactivateGuard } from './../guards/can-deactivate.guard';**

providers: [

UserArrayService,

**CanDeactivateGuard**

]

1. Внесите изменения в файл **users.routing.module.ts**

**import { CanDeactivateGuard } from './../guards/can-deactivate.guard';**

{

path: 'edit/:id',

component: UserFormComponent,

**canDeactivate: [CanDeactivateGuard]**

}

# Step\_27. resolve Guard

1. Создайте **UserResolveGuard** в файле **guards/user-resove.guard.ts** используя следующий фрагмент кода

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

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

import { User } from './../models/user';

import { UserArrayService } from './../users/services/user-array.service';

@Injectable()

export class UserResolveGuard implements Resolve<User> {

constructor(

private userArrayService: UserArrayService,

private router: Router

) {}

resolve(route: ActivatedRouteSnapshot): Promise<User> {

const id = +route.params['id'];

return this.userArrayService.getUser(id).then(user => {

// todo: check maybe -1 if id not found

if (user) {

return user;

}

else { // id not found

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

return null;

}

});

}

}

1. Внесите изменения в файл **users/users.routing.module.ts**

import { CanDeactivateGuard } from './../guards/can-deactivate.guard';

**import { UserResolveGuard } from './../guards/user-resolve.guard';**

{

path: 'edit/:id',

component: UserFormComponent,

canDeactivate: [CanDeactivateGuard],

**resolve: {**

**user: UserResolveGuard**

**}**

}

providers: [

**UserResolveGuard**

]

1. Внесите изменения в **UserFormComponent**

ngOnInit(): void {

this.user = new User(null, '', '');

**this.route.data.forEach((data: { user: User }) => {**

**this.user = Object.assign({}, data.user);**

**this.oldUser = data.user;**

**});**

~~this.route.params~~

~~.switchMap((params: Params) => this.usersService.getUser(+params['id']))~~

~~.subscribe(~~

~~user => {~~

~~this.user = Object.assign({}, user);~~

~~this.oldUser = user;~~

~~},~~

~~err => console.log(err)~~

~~);~~

}

# Step\_28. Query Parameters and Fragment

1. Внесите изменения в файл **guards/auth.guard.ts**

import { CanActivate, CanActivateChild, Router,

ActivatedRouteSnapshot, RouterStateSnapshot, **NavigationExtras**

} from '@angular/router';

1. Внесите изменения в метод **checkLogin()** в файле **guards/auth.guard.ts**

checkLogin(url: string): boolean {

if (this.authService.isLoggedIn) { return true; }

// Store the attempted URL for redirecting

this.authService.redirectUrl = url;

**// Create a dummy session id**

**const sessionId = 123456789;**

**const navigationExtras: NavigationExtras = {**

**queryParams: { 'session\_id': sessionId },**

**fragment: 'anchor'**

**};**

// Navigate to the login page with extras

this.router.navigate(['/login'], **navigationExtras**);

return false;

}

1. Внесите изменения в **LoginComponent**

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

if (this.authService.isLoggedIn) {

const redirect = this.authService.redirectUrl

? this.authService.redirectUrl : '/admin';

**const navigationExtras: NavigationExtras = {**

**queryParamsHandling: 'preserve'**

**preserveFragment: true**

**};**

// Redirect the user

this.router.navigate([redirect], **navigationExtras**);

}

1. Добавьте в темплейт **AdminDashboardComponent** следующий фрагмент разметки

<p>Session ID: {{ sessionId | async }}</p>

<a id="anchor"></a>

<p>Token: {{ token | async }}</p>

1. Внесите изменения в **AdminDashboardComponent**

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

**import { ActivatedRoute } from '@angular/router';**

**import { Observable } from 'rxjs/Observable';**

**import './../services/rxjs-extensions';**

1. Добавьте свойства

sessionId: Observable<string>;

token: Observable<string>;

1. Внесите изменения в контроллер

constructor(

**private route: ActivatedRoute**

) { }

1. Внесите изменения в метод ngOnInit()

ngOnInit() {

**this.sessionId = this.route**

**.queryParams**

**.map(params => params['session\_id'] || 'None');**

**this.token = this.route**

**.fragment**

**.map(fragment => fragment || 'None');**

}

1. Внесите изменения в темплейт компонента **AdminComponent**

<nav>

<ul class="nav nav-tabs">

<li routerLinkActive="active" [routerLinkActiveOptions]="{ exact: true }">

<a [routerLink]="['./']" queryParamsHandling="preserve" preserveFragment>Dashboard</a>

</li>

<li routerLinkActive="active">

<a [routerLink]="['./tasks']" queryParamsHandling="preserve" preserveFragment>Manage Tasks</a>

</li>

<li routerLinkActive="active">

<a [routerLink]="['./users']" queryParamsHandling="preserve" preserveFragment>Manage Users</a>

</li>

</ul>

</nav>

# Step\_29. Lazy-Loading Route Configuration

1. Внесите изменения в файл **app.routing.module.ts**

**{**

**path: 'admin',**

**loadChildren: 'app/admin/admin.module#AdminModule'**

**},**

**{**

**path: 'users',**

**loadChildren: 'app/users/users.module#UsersModule'**

**},**

1. Внесите изменения в файл **admin/admin.routing.module.ts**

const routes: Routes = [

{

**path: 'admin',**

**path:** '',

component: AdminComponent,

canActivate: [AuthGuard],

children: [

…

]

}

];

1. Внесите изменения в **users/users.routing.module.ts**

const routes: Routes = [

{

path: 'users',

path: '',

component: UsersComponent,

children: [

…

]

}

];

1. Внесите изменения в файл **app.module.ts**

import { AdminModule } from './admin/admin.module';

import { UsersModule } from './users/users.module';

imports: [

BrowserModule,

CommonModule,

FormsModule,

TasksModule,

UsersModule,

AdminModule**,**

RouterModule,

AppRoutingModule

],

1. **Может понадобиться перезапустить ng serve!**

# Step\_30. canLoad Guard

1. Внесите изменения в AuthGuard в файле **guards/auth.guard.ts**

import {

CanActivate, CanActivateChild, **CanLoad**, Router, **Route**,

ActivatedRouteSnapshot, RouterStateSnapshot, NavigationExtras

} from '@angular/router';

1. Внесите изменения в описание класса

export class AuthGuard implements CanActivate, CanActivateChild, **CanLoad** {

1. Добавьте метод

**canLoad(route: Route): boolean {**

**let url = `/${route.path}`;**

**return this.checkLogin(url);**

**}**

1. Внесите изменения в файл **app.routing.module.ts**

**import { AuthGuard } from './guards/auth.guard';**

{

path: 'admin',

**canLoad: [AuthGuard],**

loadChildren: 'app/admin/admin.module#AdminModule'

},

1. Для компонентов, которыми управляет роутер, селектор не нужен. [↑](#footnote-ref-1)
2. Согласно стайлгайду, этот файл должен называтся **app-routing.module.ts**, а класс AppRoutingModule. Но я изменил это название на **app.routing.module.ts** для того, чтобы этот файл находился рядом с файлом **app.module.ts**. [↑](#footnote-ref-2)