Angular. Routing.

Contents

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 **src/index.html**

```
<title>Angular 2: Routing</title>
<base href="/">
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Step 02. Components

1. Создайте папку **app/components**.
2. Создайте три компонента-заглушки **HomeComponent, AboutComponent, PageNotFoundComponent** выполнив следующие команды из командной строки папки app[1]

```
app>ng g c components/home[2] --spec=false -ve=Emulated --skip-import=true[3]
app>ng g c components/about --spec=false -ve=Emulated --skip-import=true
app>ng g c components/page-not-found --spec=false -ve=Emulated --skip-import=true
```

3. Создайте файл **app/components/index.ts** и добавьте в него следующий фрагмент кода

```
export * from './about/about.component';
export * from './home/home.component';
export * from './page-not-found/page-not-found.component';
```

[1] Все следующие команды генерации сущностей выполнять из папки app.

[2] Компоненты, которыми управляет роутер не нуждаются в селекторе, поэтому селектор можно либо оставить, либо удалить.

[3] Флаги означают следующее

--spec=false – не генерировать файл теста.

--skip-import – не регистрировать компоненты в модуле

-ve=Emulated – включить инкапсуляцию стилей

Step 03. Routes Config

Создайте файл **app/app.routing.module.ts**[4]. 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 {}
```

[4] 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**.

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: [
    AppComponent,
    appRouterComponents
  ],

// 4
```

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
              (activate)='onActivate($event)'
              (deactivate)='onDeactivate($event)'>
     </router-outlet>
     <!-- Routed views go here -->
</div>
```

2. Добавьте два метода в класс компонента **AppComponent** используя следующий фрагмент кода

```
  onActivate($event) {
    console.log('Activated Component', $event);
  }

  onDeactivate($event) {
    console.log('Deactivated Component', $event);
  }
```

Step 06. routerLink

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

```
// 1
<a class="navbar-brand" routerLink="/home">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 модель задачи выполнив следующую команду из командной строки

```
app>ng g cl models/task --spec=false
```

2. Замените содержание класса на следующее

```
export class Task {
	constructor(
		public id: number,
		public action: string,
		public priority: number,
		public estHours: number,
		public actHours?: number,
		public done?: boolean
	) {
		this.actHours = actHours || 0;
		this.done = done || false;
	}
}
```

3. Создайте модуль **tasks/tasks.module.ts** выполнив следующую команду из командной строки

```
app>ng g m tasks
```

4. 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 {}
```

5. Создайте сервис **TaskArrayService** выполнив следующую команду из командной строки

```
app>ng g s tasks/services/task-array --spec=false -m=tasks
```

6. Замените код сервиса используя следующий фрагмент кода

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

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

const 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)
];
```

```
const 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);
    }
  }

  completeTask(task: Task): void {
    task.done = true;
  }
}
```

7. Создайте компонент **TaskListComponent** выполнив следующую команду из командной строки и замените код класса компонента используя фрагмент кода ниже

```
app>ng g c tasks/task-list --spec=false -ve=Emulated
```

```
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 taskArrayService: TaskArrayService) { }
```

```
  ngOnInit() {
    console.log(this.tasks);
    this.taskArrayService.getTasks()
      .then(tasks => this.tasks = tasks)
      .catch((err) => console.log(err));
  }

  completeTask(task: Task): void {
    this.taskArrayService.completeTask(task);
  }
}
```

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

```
<task
  *ngFor="let task of tasks"
  [task]="task"
  (onComplete)="completeTask($event)">
</task>
```

9. Создайте компонент **TaskComponent** выполнив следующую команду из командной строки и замените код класса компонента используя фрагмент кода ниже

```
app>ng g c tasks/task --spec=false -ve=Emulated
```

```
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(): void {
    this.onComplete.emit(this.task);
  }

  editTask() {

  }
}
```

10. 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()">
                Done
            </button>
            <button class="btn btn-warning btn-sm"
                (click)="editTask()">
                Edit
            </button>
        </div>
</div>
```

11. 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';
```

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

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

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

@NgModule({
  providers: [
    TaskArrayService
  ]
})
export class TasksModule {}
```

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';
```

2. 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';
```

2. Внесите изменения в следующий фрагмент кода:

```
imports: [
    BrowserModule,
    FormsModule,
    TasksModule,
    AppRoutingModule
  ],
```

Step_12. Tasks List on Home Page

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

```
const routes: Routes = [
  {
    path: 'task-list',
    path: 'home',
    component: TaskListComponent
  }
];
```

2. 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];
```

3. 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';
```

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

Step_13. Navigate

1. Создайте компонент **TaskFormComponent** в папке **tasks** (ng g c task-form -m tasks) 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 taskArrayService: TaskArrayService,
  ) { }

  ngOnInit(): void {
    this.task = new Task(null, '', null, null);
  }

  ngOnDestroy(): void {
  }

  saveTask() {
    const task = {...this.task};

    if (task.id) {
      this.taskArrayService.updateTask(task);
    }
    else {
      this.taskArrayService.addTask(task);
    }
  }

  goBack(): void {
  }
}
```

2. 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
                type="button"
                class="btn btn-primary"
                (click)="goBack()">Back
            </button>
          </form>
      </div>
</div>
```

3. 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';
```

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

```
// 1

import { TaskFormComponent } from './task-form/task-form.component';
import { TaskListComponent, TaskComponent, TaskFormComponent, TaskArrayService } from '.';
```

5. 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
  }
];
```

6. Внесите изменения в компонент **TaskComponent**

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

constructor(
    private router: Router
) { }


editTask() {
    const 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**

```
// 1
import { ActivatedRoute, Params } from '@angular/router';

import 'rxjs/add/operator/switchMap';

// 2
constructor(
    private taskArrayService: TaskArrayService,
    private route: ActivatedRoute
  ) { }

// 3
ngOnInit(): void {
    this.task = new Task(null, '', null, null);

    // it is not necessary to save subscription to route.paramMap
    // it handles automatically
    this.route.paramMap
      .switchMap((params: Params) => this.taskArrayService.getTask(+params.get('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';
```

2. Внесите изменения в конструктор

```
constructor(
    private taskArrayService: TaskArrayService,
    private router: Router,
    private route: ActivatedRoute
  ) { }
```

3. Внесите изменения в метод goBack()

```
goBack(): void {
    this.router.navigate(['/home']);
  }
```

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

```
if (task.id) {
      this.taskArrayService.updateTask(task);
}
else {
      this.taskArrayService.addTask(task);
}

this.goBack();
```

Step_16. Secondary Router Outlet

1. Внесите изменения в **app.component.html** используя следующий фрагмент разметки

```
<div class="container">
     <router-outlet
         (activate)='onActivate($event)'
         (deactivate)='onDeactivate($event)'>
     </router-outlet>
     <!-- Routed views go here -->
</div>

<div class="container">
     <div class="col-md-10">
          <router-outlet
              (activate)='onActivate($event)'
              (deactivate)='onDeactivate($event)'>
          </router-outlet>
          <!-- Routed views go here -->
     </div>
     <div class="col-md-2">
          <router-outlet name="popup"></router-outlet>
     </div>
 </div>
```

2. Создайте сервис **MessagesService** выполнив следующую команду из командной строки и замените код класса используя следующий фрагмент кода

```
app>ng g s services/messages --spec=false -m=app
```

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

@Injectable()
export class MessagesService {
  isDisplayed = false;

  private messages: string[] = [];

  constructor() { }

  addMessage(message: string): void {
    const currentDate = new Date();
    this.messages.unshift(`${message} at ${currentDate.toLocaleString()}`);
  }

  getMessages(): Array<string> {
    return this.messages;
  }

  clearMessageList(): void {
    this.messages.length = 0;
  }
}
```

3. Создайте файл **services/index.ts** используя следующий фрагмент кода

```
export * from './messages.service';
```

1. Создайте компонент **MessagesComponent** выполнив следующую команду из командной строки и внесите изменения в шаблон компонента используя следующий фрагмент разметки

```
app>ng g c components/messages --spec=false -m=app
```

```
<div class="row">
  <h4 class="col-md-10">Message Log</h4>
  <span class="col-md-2">
    <a class="btn btn-default" (click)="close()">x</a>
  </span>
</div>
<div *ngFor="let message of messagesService.getMessages(); let i=index">
  <div class="message-row">
    {{message}}
  </div>
</div>
```

2. Внесите изменения в файл стилей компонента **MessagesComponent** используя следующий код

```
.message-row {
  margin-bottom: 10px;
}
```

3. Внесите изменения в класс компонента **MessagesComponent** используя следующий фрагмент кода

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

import { MessagesService } from './../../services';

@Component({
  selector: 'app-messages',
  templateUrl: './messages.component.html',
  styleUrls: ['./messages.component.css']
})
export class MessagesComponent implements OnInit {

  constructor(
    public messagesService: MessagesService,
    private router: Router
  ) { }

  ngOnInit() {
  }

  close() {
    this.router.navigate([{ outlets: { popup: null } }]);
    this.messagesService.isDisplayed = false;
  }
}
```

4. Внесите изменения в файл **components/index.ts** используя следующий фрагмент кода

```
export * from './about/about.component';
export * from './messages/messages.component';
export * from './page-not-found/page-not-found.component';
```

5. Внесите изменения в файл **app.module.ts** используя следующий фрагмент кода

```
import { MessagesComponent } from './components/messages/messages.component';
import { MessagesService } from './services/messages.service';
```

6. Внесите изменения в файл **app.routing.module.ts** используя следующий фрагмент кода

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

{
    path: 'messages',
    component: MessagesComponent,
    outlet: 'popup'
  },
```

7. Внесите изменения в компонент **AppComponent** используя следующий фрагмент кода

```
import { MessagesService } from './services';

constructor(public messagesService: MessagesService) { }
```

8. Внесите изменения в файл **app.component.html** используя следующий фрагмент разметки и посмотрите результат.

```
<div class="col-md-2">
  <button class="btn btn-success"
      *ngIf="!messagesService.isDisplayed"
      [routerLink]="[{outlets: {popup: ['messages']}}]">
    Show Messages
  </button>
  <router-outlet name="popup"></router-outlet>
</div>
```

9. Внесите изменения в компонент **AppComponent** используя следующий фрагмент кода

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


constructor(
    public messagesService: MessagesService,
    private router: Router
  ) { }


displayMessages(): void {
    this.router.navigate([{ outlets: { popup: ['messages'] } }]);
    this.messagesService.isDisplayed = true;
  }
```

10. Внесите изменения в разметку компонента **AppComponent** используя следующий фрагмент разметки

```
<button class="btn btn-success"
      *ngIf="!messagesService.isDisplayed"
      [routerLink]="[{outlets: {popup: ['messages']}}]">
    Show Messages
  </button>
<button class="btn btn-success"
        *ngIf="!messagesService.isDisplayed"
        (click)="displayMessages()">Show Messages</button>
```

Step_17. Users Components

1. Создайте модель пользователя выполнив следующую команду из командной строки

```
app>ng g cl models/user --spec=false
```

2. Замените содержание класса на следующее

```
export class User {
   constructor(
          public id: number,
          public firstName: string,
          public lastName: string
   ) {}
}
```

3. Создайте модуль **users/users.module.ts** выполнив следующую команду из командной строки

```
app>ng g m users
```

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

```
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { CommonModule } from '@angular/common';

@NgModule({
  imports: [
    CommonModule,
    FormsModule,
  ],
  declarations: [
  ],
  providers: [
  ]
})
export class UsersModule {}
```

5. Создайте сервис **UserArrayService** выполнив следующую команду из командной строки

```
app>ng g s users/services/user-array --spec=false -m=users
```

6. Замените код сервиса используя следующий фрагмент кода

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

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

const userList = [
  new User(1, 'Anna', 'Borisova'),
  new User(2, 'Boris', 'Vlasov'),
  new User(3, 'Gennadiy', 'Dmitriev')
];

const 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);
    }
  }
}
```

7. Создайте компонент **UserListComponent** выполнив следующую команду из командной строки и замените код класса компонента используя фрагмент кода ниже

```
app>ng g c users/user-list --spec=false -ve=Emulated --skip-import
```

```
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 userArrayService: UserArrayService,
  ) { }

  ngOnInit() {
    this.userArrayService.getUsers()
      .then(users => this.users = [...users])
      .catch(err => console.log(err));
  }

  ngOnDestroy() {
  }
}
```

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

```
<user
  *ngFor="let user of users"
  [user]="user">
</user>
```

9. Создайте компонент **UserComponent** выполнив следующую команду из командной строки и замените код класса компонента используя фрагмент кода ниже

```
app>ng g c users/user --spec=false -ve=Emulated
```

```
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() {
  }
}
```

10. 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>
```

11. Создайте компонент **UserFormComponent** выполнив следующую команду из командной строки и замените код класса компонента используя фрагмент кода ниже

```
app>ng g c users/user-form --spec=false -ve=Emulated --skip-import
```

```
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;
  originalUser: User;

  constructor(
    private userArrayService: UserArrayService,
    private route: ActivatedRoute,
  ) { }

  ngOnInit(): void {
    this.user = new User(null, '', '');

    const id = +this.route.snapshot.paramMap.get('id');
    this.userArrayService.getUser(id)
      .then(user => {
        this.user = {...user};
        this.originalUser = {...user};
      })
      .catch(err => console.log(err));
  }

  ngOnDestroy(): void {
  }

  saveUser() {
    const user = {...this.user};

    if (user.id) {
      this.userArrayService.updateUser(user);
    } else {
      this.userArrayService.addUser(user);
    }
    this.originalUser = {...this.user};
  }

  goBack() {
  }
}
```

12. 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>
```

13. 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';
```

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

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

import { UserComponent } from './user/user.component';
import { UserComponent, UserArrayService } from '.';
```

Step_18. Users Feature Area

1. Создайте компонент **UsersComponent** выполнив следующую команду из командной строки и замените код класса компонента используя фрагмент кода ниже

app>ng g c users/users --spec=false -ve=Emulated –flat --skip-import

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

@Component({
  templateUrl: './users.component.html',
  styleUrls: ['./users.component.css']
})
export class UsersComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

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

```
<h2>Users</h2>
```

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

```
export * from './users.component';
export * from './user/user.component';
```

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

```
import { UsersComponent } from './users.component';
import { UsersComponent, UserListComponent, UserComponent, UserFormComponent, UserArrayService }
from '.';
```

5. 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_19. 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>
```

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

```
<h2>Users</h2>
<router-outlet></router-outlet>
```

3. Создайте файл роутинга **users/users.routing.module.ts** используя 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 { }
```

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

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

```
imports: [
    CommonModule,
    FormsModule,
    UsersRoutingModule
  ]
declarations: [
    usersRouterComponents,
    UserComponent,
  ],
```

Step_20. Relative Navigation

1. Внесите изменения в **UserComponent** используя следующий фрагмент кода

```
// 1
import { Router, ActivatedRoute } from '@angular/router';

// 2
constructor(
    private router: Router,
    private route: ActivatedRoute
  ) { }

// 3
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});
  }
```

2. Внесите изменения в компонент **UserFormComponent** используя следующий фрагмент кода

```
// 1
import { ActivatedRoute, Params, Router } from '@angular/router';

// 2
constructor(
    private userArrayService: UserArrayService,
    private route: ActivatedRoute,
    private router: Router
  ) { }
```

3. Внесите изменения в метод **saveUser**

```
if (user.id) {
      this.userArrayService.updateUser(user);
} else {
      this.userArrayService.addUser(user);
}
this.originalUser = {...this.user};

this.goBack();
```

4. Внесите изменения в метод **goBack**

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

Step_21. Optional Parameters

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

```
if (user.id) {
      this.userArrayService.updateUser(user);
      this.router.navigate(['/users', {id: user.id}]);
    } else {
      this.userArrayService.addUser(user);
      this.goBack();
    }
this.originalUser = {...this.user};
this.goBack();
```

2. Внесите изменения в компонент **UserListComponent** используя следующий фрагмент кода

```
// 1
import { ActivatedRoute, Params } from '@angular/router';
import 'rxjs/add/operator/switchMap';

// 2
private editedUser: User;

// 3
constructor(
    private userArrayService: UserArrayService,
    private route: ActivatedRoute
  ) { }

// 4
ngOnInit() {
    this.userArrayService.getUsers()
          .then(users => this.users = [...users])
          .catch(err => console.log(err));

    // listen id from UserFormComponent
    this.route.paramMap
      .switchMap((params: Params) => this.userArrayService.getUser(+params.get('id')))
      .subscribe(
        (user: User) => {
          this.editedUser = {...user};
          console.log(`Last time you edit user ${JSON.stringify(this.editedUser)}`);
        },
        err => console.log(err)
      );
  }
```

3. Добавьте метод

```
isEdited(user: User) {
    if (this.editedUser) {
      return user.id === this.editedUser.id;
    }
    return false;
  }
```

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

```
<user
  *ngFor='let user of users'
```

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

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

```
:host.edited > div {
  border: 2px dotted red;
}
```

Step_22. Admin Feature Area

1. Создайте модуль **admin/admin.module.ts** выполнив следующую команду из командной строки

```
app>ng g m admin
```

2. Создайте следующие компоненты-заглушки **AdminDashboardComponent, ManageTasksComponent, ManageUsersComponent**, **AdminComponent,** выполнив следующие команды из командной строки

```
app> ng g c admin/admin-dashboard --spec=false -ve=Emulated --skip-import
app> ng g c admin/manage-tasks --spec=false -ve=Emulated --skip-import
app> ng g c admin/manage-users --spec=false -ve=Emulated --skip-import
app> ng g c admin/admin --spec=false -ve=Emulated --flat --skip-import
```

3. 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>
```

4. 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';
```

5. 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 { }
```

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

```
// 1
import { AdminRoutingModule, adminRouterComponents } from './admin.routing.module';

// 2
declarations: [
   adminRouterComponents
],
imports: [
    CommonModule,
    AdminRoutingModule
]
```

7. 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
  ],
```

8. 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_23. canActivate Guard

1. Создайте защитника **AuthGuard** выполнив следующую команду из командной строки

```
app>ng g g guards/auth --spec=false -m=app
```

2. Внесите изменения в метод canActivate используя следующий фрагмент кода

```
canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean> | Promise<boolean> | boolean {
    console.log('CanActivateGuard is called');
    return true;
  }
```

3. 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_24. 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';
```

2. Создайте сервис **AuthService** выполнив следующую команду из командной строки

```
ng g s services/auth --spec=false -m=app
```

3. Внесите изменения в **AuthService** 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;
  }
}
```

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

```
import {CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, Router } from '@angular/router';
import { AuthService } from './../services/auth.service';
```

5. Добавьте конструктор

```
constructor(
    private authService: AuthService,
    private router: Router
  ) {}
```

6. Добавьте метод checkLogin()

```
private 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;
  }
```

7. Внесите изменения в метод **canActivate**

```
canActivate(next: ActivatedRouteSnapshot, state: RouterStateSnapshot): Observable<boolean> | Promise
<boolean> | boolean {
    console.log('CanActivateGuard is called');
    const url: string = state.url;
    return this.checkLogin(url);
    return true;
  }
```

Step_25. Login Component

1. Создайте компонент **LoginComponent** выполнив следующую команду из командной строки

```
app>ng g c components/login --spec=false -ve=Emulated --skip-import=true
```

2. Внесите изменения в файл **components/index.ts** используя следующий фрагмент кода

```
export * from './about/about.component';
export * from './login/login.component';
export * from './messages/messages.component';
export * from './page-not-found/page-not-found.component';
```

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

```
// 1
import { AboutComponent, MessagesComponent, LoginComponent, PageNotFoundComponent } from
'./components';

// 2
{
    path: 'about',
    component: AboutComponent
},
{
    path: 'login',
    component: LoginComponent
},

// 3
export let appRouterComponents = [AboutComponent, PageNotFoundComponent, LoginComponent];
```

4. Внесите измнения в **LoginComponent**

```
import { Router } from '@angular/router';
import { AuthService } from './../../services/auth.service';
```

5. Добавьте свойство

```
message: string;
```

6. Добавьте методы

```
private 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
        const redirect = this.authService.redirectUrl ? this.authService.redirectUrl : '/admin';
        // Redirect the user
        this.router.navigate([redirect]);
      }
    });
  }
```

```
  logout() {
    this.authService.logout();
    this.setMessage();
  }
```

7. Внесите изменения в конструктор **LoginComponent**

```
constructor(
    public authService: AuthService,
    public router: Router
  ) { }
```

8. Внесите изменения в метод **ngOnInit** компонента LoginComponent используя следующий фрагмент кода

```
ngOnInit() {
  this.setMessage();
}
```

9. Добавьте разметку для компонента **LoginComponent** используя 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>
```

10. Внесите изменения в темплейт **AppComponent**

```
<li routerLinkActive="active">
     <a routerLink="/admin">Admin</a>
 </li>
<li routerLinkActive="active">
     <a routerLink="/login">Login</a>
</li>
```

Step_26. canActivateChild Guard

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

```
import {CanActivate, CanActivateChild, ActivatedRouteSnapshot, RouterStateSnapshot, Router } from
'@angular/router';

export class AuthGuard implements CanActivate, CanActivateChild {
…
}
```

2. Добавьте метод **canActivateChild**

```
canActivateChild(next: ActivatedRouteSnapshot, state: RouterStateSnapshot) : Observable<boolean> |
Promise<boolean> | boolean {
       console.log('CanActivateChild Guard is called');
       const url: string = state.url;

      return this.checkLogin(url);
  }
```

3. 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_27. canDeactivate Guard

1. Создайте сервис **DialogService** выполнив следующую команду из командной строки

```
app>ng g s services/dialog --spec=false -m=app
```

2. Внесите изменения в **AuthService** используя следующий фрагмент кода

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

@Injectable()
export class DialogService {

  confirm(message?: string): Promise<boolean> {
    return new Promise<boolean>(resolve => {
      resolve(window.confirm(message || 'Is it OK?'));
    });
  }
}
```

3. Создайте интерфейс **CanComponentDeactivate** выполнив следующую команду из командной строки

```
app>ng g i interfaces/can-component-deactivate
```

4. Внесите изменения в интерфейс **CanComponentDeactivate** используя следующий фрагмент кода

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

export interface CanComponentDeactivate {
 canDeactivate: () => Observable<boolean> | Promise<boolean> | boolean;
}
```

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

```
import { Injectable } from '@angular/core';
import { CanDeactivate } from '@angular/router';

import { CanComponentDeactivate } from './../interfaces/can-component-deactivate';

@Injectable()
export class CanDeactivateGuard implements CanDeactivate<CanComponentDeactivate> {
  canDeactivate(component: CanComponentDeactivate) {
    return component.canDeactivate();
  }
}
```

6. Внесите изменения в компонент **UserFormComponent**

```
import { Observable }    from 'rxjs/Observable';
import { DialogService }  from './../../services/dialog.service';
import { CanComponentDeactivate } from './../../interfaces/can-component-deactivate';


export class UserFormComponent implements OnInit, OnDestroy, CanComponentDeactivate {

constructor(
    private userArrayService: UserArrayService,
    private route: ActivatedRoute,
    private router: Router,
    private dialogService: DialogService
```

```
  ) { }


canDeactivate(): Observable<boolean> | Promise<boolean> | boolean {
    const flags = Object.keys(this.originalUser).map(key => {
      if (this.originalUser[key] === this.user[key]) {
        return true;
      }
      return false;
    });

    if (flags.every(el => el)) {
      return true;
    }

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

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

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

{
        path: 'edit/:id',
        component: UserFormComponent,
        canDeactivate: [CanDeactivateGuard]
}

providers: [
    CanDeactivateGuard
  ]
```

Step_28. resolve Guard

1. Создайте **UserResolveGuard** в файле **guards/user-resolve.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> | null {
    console.log('UserResolve Guard is called');
    const id = +route.paramMap.get('id');

    return this.userArrayService.getUser(id).then(user => {
      if (user) {
        return Promise.resolve(user);
      }

      // id not found
      this.router.navigate(['/users']);
      return null;
    });
  }
}
```

2. 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: [
    CanDeactivateGuard,
    UserResolveGuard
]
```

3. Внесите изменения в **UserFormComponent**

```
ngOnInit(): void {
    this.user = new User(null, '', '');

    this.route.data.subscribe(data => {
```

```
      this.user = {...data.user};
      this.originalUser = {...data.user};
    });

const id = +this.route.snapshot.paramMap.get('id');
    this.userArrayService.getUser(id)
      .then(user => {
        this.user = Object.assign({}, user);
        this.originalUser = Object.assign({}, user);
      })
      .catch(err => console.log(err));

}
```

Step_29. Query Parameters and Fragment

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

```
import { CanActivate, CanActivateChild, Router,
  ActivatedRouteSnapshot, RouterStateSnapshot, NavigationExtras
} from '@angular/router';
```

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

```
private 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;
  }
```

3. 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);
}
```

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

```
<p>Session ID: {{ sessionId | async }}</p>
<a id="anchor"></a>
<p>Token: {{ token | async }}</p>
```

5. Внесите изменения в **AdminDashboardComponent**

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { Observable } from 'rxjs/Observable';
import './../../services/rxjs-extensions';
```

6. Добавьте свойства

```
sessionId: Observable<string>;
token: Observable<string>;
```

7. Внесите изменения в контроллер

```
constructor(
    private route: ActivatedRoute
  ) { }
```

8. Внесите изменения в метод **ngOnInit**

```
ngOnInit() {
    this.sessionId = this.route
      .queryParamMap
      .map(params => params.get('session_id') || 'None');

    this.token = this.route
      .fragment
      .map(fragment => fragment || 'None');
  }
```

9. 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_30. 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'
  },
```

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

```
const routes: Routes = [
  {
    path: 'admin',
    path: '',
    component: AdminComponent,
    canActivate: [AuthGuard],
    children: [
      …
    ]
  }
];
```

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

```
const routes: Routes = [
  {
    path: 'users',
    path: '',
    component: UsersComponent,
    children: [
      …
    ]
  }
];
```

4. 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
  ],
```

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

Step_31. 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';
```

2. Внесите изменения в описание класса

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

3. Добавьте метод **canLoad**

```
canLoad(route: Route): Observable<boolean> | Promise<boolean> | boolean {
    console.log('CanLoad Guard is called');
    const url = `/${route.path}`;
    return this.checkLogin(url);
}
```

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

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

  {
    path: 'admin',
    canLoad: [AuthGuard],
    loadChildren: 'app/admin/admin.module#AdminModule'
  },
```

Step_32. Default Preloading Strategy

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

```
// 1
import { Routes, RouterModule, PreloadAllModules, ExtraOptions } from '@angular/router';

// 2
const extraOptions: ExtraOptions = {
  preloadingStrategy: PreloadAllModules,
  enableTracing: true // Makes the router log all its internal events to the console.
};

// 3
@NgModule({
  imports: [
    RouterModule.forRoot(routes, extraOptions)
  ]
})
```

Step_33. Custom Preloading Strategy

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

```
{
    path: 'users',
    loadChildren: 'app/users/users.module#UsersModule',
    data: { preload: true }
},
```

2. Создайте сервис в файле **app/services/custom-preloading-strategy.service.ts** используя следующий фрагмент кода

```
import { Injectable } from '@angular/core';
import { PreloadingStrategy, Route } from '@angular/router';
import { Observable } from 'rxjs/Observable';
import './rxjs-extensions';

@Injectable()
export class CustomPreloadingStrategyService implements PreloadingStrategy {
  private preloadedModules: string[] = [];

  preload(route: Route, load: () => Observable<any>): Observable<any> {
    if (route.data && route.data['preload']) {
      this.preloadedModules.push(route.path);
      return load();
    } else {
      return Observable.of(null);
    }
  }
}
```

3. Внесите изменения в файл **app/services/index.ts** используя следующий фрагмент кода

```
export * from './auth.service';
export * from './custom-preloading-strategy.service';
export * from './dialog.service';
export * from './messages.service';
```

4. Внесите изменения в **app.routing.module.ts**

```
// 1
import { Routes, RouterModule, PreloadAllModules, ExtraOptions } from '@angular/router';
import { CustomPreloadingStrategyService } from './services';

// 2
const extraOptions: ExtraOptions = {
  preloadingStrategy: PreloadAllModules CustomPreloadingStrategyService,
  // enableTracing: true // Makes the router log all its internal events to the console.
};

// 3
@NgModule({
  imports: [
    RouterModule.forRoot(routes, extraOptions)
  ],
  providers: [
    CustomPreloadingStrategyService
  ],
  exports: [
```

```
    RouterModule
  ]
})
```

Step_34. Router Events and Title Service

1. Внесите изменения в **app.routing.module.ts** используя следующий фрагмент кода

```
const routes: Routes = [
  {
    path: 'about',
    component: AboutComponent,
    data: { title: 'About' }
  },
  {
    path: 'login',
    component: LoginComponent,
    data: { title: 'Login' }
  },
  {
    path: 'admin',
    canLoad: [AuthGuard],
    loadChildren: 'app/admin/admin.module#AdminModule',
    data: { title: 'Admin' }
  },
  {
    path: 'users',
    loadChildren: 'app/users/users.module#UsersModule',
    data: {
      preload: true,
      title: 'Users'
    }
  },
  {
    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,
    data: { title: 'Page Not Found' }
  }
];
```

2. Внесите изменения в **tasks.routing.module.ts** используя следующий фрагмент кода

```
const routes: Routes = [
  {
    path: 'home',
    component: TaskListComponent,
    data: { title: 'Task Manager'}
  },
  {
    path: 'edit/:id',
    component: TaskFormComponent
  }
];
```

3. Внесите изменения в файл **app.component.ts** используя следующий фрагмент кода

```
// 1
import { Component, OnInit, OnDestroy } from '@angular/core';
```

```
import { Title } from '@angular/platform-browser';
import { Router, NavigationEnd } from '@angular/router';
import { Subscription } from 'rxjs/Subscription';

// 2
export class AppComponent implements OnInit, OnDestroy

// 3
private sub: Subscription;

  constructor(
    public messagesService: MessagesService,
    private titleService: Title,
    private router: Router
  ) { }

  ngOnInit() {
    this.setPageTitles();
  }

  ngOnDestroy() {
    this.sub.unsubscribe();
  }

  private setPageTitles() {
    this.sub = this.router.events
      .filter(event => event instanceof NavigationEnd)
      .map(() => this.router.routerState.root)
      .map(route => {
        while (route.firstChild) {
          route = route.firstChild;
        }
        return route;
      })
      .filter(route => route.outlet === 'primary')
      .switchMap(route => route.data)
      .subscribe(
         data => this.titleService.setTitle(data['title'])
      );
  }
```

Step_35. Meta Service (Meta only available in 4.X)

1. Внесите изменения в **tasks.routing.module.ts** используя следующий фрагмент кода

```
const routes: Routes = [
  {
    path: 'home',
    component: TaskListComponent,
    data: { title: 'Task Manager'}
    data: {
       title: 'Task Manager',
       meta: [{
         name: 'description',
         content: 'Task Manager Application. This is an ASP application'
       },
       {
         name: 'keywords',
         content: 'Angular 4 tutorial, SPA Application, Routing'
       }]
    }
  },
];
```

2. Внесите изменения в файл **app.component.ts** используя следующий фрагмент кода

```
// 1
import { Title, Meta } from '@angular/platform-browser';

// 2
constructor(
    public messagesService: MessagesService,
    private titleService: Title,
    private metaService: Meta,
    private router: Router
  ) { }
```

3. Переименуйте метод **setPageTitles** в **setPageTitlesAndMeta** в файле **app.component.ts**
4. Внеситеизменения в файл **app.component.ts** используя следующий фрагмент кода

```
// 1
ngOnInit() {
  this.setPageTitles();
  this.setPageTitlesAndMeta();
}

// 2
.subscribe(
   data => this.titleService.setTitle(data['title'])
   data => {
        this.titleService.setTitle(data['title']);
        this.metaService.addTags(data['meta']);
   }
);
```