Curso Avanzado de Angular

Plataforma: Udemy |Instructor: Fernando Herrera |Fecha: Marzo 2023

Nota del Autor:

El siguiente documento constituye una exhaustiva recopilación de los conocimientos y prácticas derivados del curso de Angular ofrecido en la plataforma Udemy. Su propósito fundamental radica en servir como un recurso detallado y de fácil acceso para futuras referencias en mi trayectoria profesional.

Además de presentar el código proporcionado por el instructor, este documento incluye explicaciones detalladas de ciertos conceptos que, aunque se abordaron en el curso de Angular, no fueron explorados en profundidad. Se ha procurado enriquecer la comprensión de estos temas mediante análisis más detallados. Esta extensión va más allá de la enseñanza estándar del curso, proporcionando una perspectiva más completa y facilitando la asimilación de conceptos clave. Así, este material no solo actúa como una recopilación de lo aprendido, sino también como un recurso complementario que busca ofrecer una comprensión más holística de los temas tratados en el curso.

Quisiera enfatizar que este material no tiene como finalidad generar lucro alguno. En lugar de ello, busca únicamente consolidar y mantener frescos los conocimientos adquiridos durante el curso. Es importante señalar que la mayor parte del código presente en este documento ha sido proporcionado por el instructor, **Fernando Herrera**. Solo en casos excepcionales se han incorporado modificaciones o funcionalidades adicionales como resultado de prácticas complementarias.

Esta recopilación se presenta como una herramienta personal, creada con el objetivo de fortalecer y consolidar los conceptos aprendidos en el curso de Angular. Agradezco profundamente al instructor por compartir su experiencia y conocimientos, los cuales han sido fundamentales para mi desarrollo en esta tecnología.

Espero que este documento no solo sirva como recordatorio para mí, sino también como una fuente de conocimiento para otros estudiantes interesados en profundizar en Angular. Cabe destacar que cualquier beneficio derivado de este material debe ser atribuido principalmente al esfuerzo y dedicación del instructor y la plataforma Udemy, a quienes agradezco por facilitar este valioso aprendizaje.

Mas información Aquí: https://www.udemy.com/course/angular-avanzado-fernando-herrera/

__06/Marzo 2024 - ... __

C O N T E N I D O

Instalación

Se requieren las mismas herramientas instaladas en el curso Básico Vew guía de Instalación

Angular Cli Tests

Durante la participación en este curso (Marzo 2024) esta es la versión del Angular Cli / Node y Angular.

```
[fcruz@fedora 01-clitest]$ ng version


     _                      _                 ____ _     ___
    / \   _ __   __ _ _   _| | __ _ _ __     / ___| |   |_ _|
   / △ \ | '_ \ / _` | | | | |/ _` | '__|   | |   | |    | |
  / ___ \| | | | (_| | |_| | | (_| | |      | |___| |___ | |
 /_/   \_\_| |_|\__, |\__,_|_|\__,_|_|       \____|_____|___|
                |___/


Angular CLI: 17.2.3
Node: 20.10.0
Package Manager: npm 10.2.3
OS: linux x64

Angular: 17.2.4
... animations, common, compiler, compiler-cli, core, forms
... platform-browser, platform-browser-dynamic, router

Package                         Version
---------------------------------------------------------
@angular-devkit/architect       0.1702.3
@angular-devkit/build-angular   17.2.3
@angular-devkit/core            17.2.3
@angular-devkit/schematics      17.2.3
@angular/cli                    17.2.3
@schematics/angular             17.2.3
rxjs                            7.8.1
typescript                      5.2.2
zone.js                         0.14.4
```

Comando help:

--help podemos usarlo de forma global `ng --help` o sobre un comando específico, por ejemplo podemos obtener ayuda para el comando que genera un componente:

```
$ ng g c --help
ng g component [name]

Creates a new, generic component definition in the given project.

Arguments:
  name  The name of the component.
[string]
```

```
Options:
      -c, --change-detection    The change detection strategy to use in the
new component.

      [string] [choices: "Default", "OnPush"] [default: "Default"]
```

NOTA: se mostrará todas las opciones, en este caso a manera de ejemplo mostramos el parámetro **-c** para especificar la estrategia de detección de cambios.

A continuación se muestran las opciones completas para la versión del ***cli 17.2.3***

Option	Desc	Values
--help	Shows a help message for this command in the console.	[boolean]
--interactive	Enable interactive inputprompts.	[boolean] [default: true]
-d, --dry-run	Run through and reports activity without writing out results.	[boolean] [default: false]
--defaults	Disable interactive input prompts for options with a default.	[boolean] [default: false]
--force	Force overwriting of existing files.	[boolean] [default: false]
-c, --change-detection	The change detection strategy to use in the new component.	[string] [choices: "Default", "OnPush"] [default: "Default"]
-b, --display-block	Specifies if the style will contain `:host { display: block; }`.	[boolean] [default: false]
--export	The declaring NgModule exports this component.	[boolean] [default: false]
--flat	Create the new files at the top level of the current project.	[boolean] [default: false]
-s, --inline-style	Include styles inline in the component.ts file. Only CSS styles can be included inline. By default, an external styles file is created and referenced in the component.ts file.	[boolean] [default: false]
-t, --inline-template	Include template inline in the component.ts file. By default, an external template file is created and referenced in the component.ts file.	[boolean] [default: false]
-m, --module	The declaring NgModule.	[string]
-p, --prefix	The prefix to apply to the generated component selector.	[string]
--project	The name of the project.	[string]

Option	Desc	Values
--selector	The HTML selector to use for this component.	[string]
--skip-import	Do not import this component into the owning NgModule.	[boolean] [default: false]
--skip-selector	Specifies if the component should have a selector or not.	[boolean] [default: false]
--skip-tests	Do not create "spec.ts" test files for the new component.	[boolean] [default: false]
--standalone	Whether the generated component is standalone.	[boolean] [default: true]
--style	The file extension or preprocessor to use for style files, or 'none' to skip generating the style file.	[string] [choices: "css", "scss", "sass", "less", "none"] [default: "css"]
--type	Adds a developer-defined type to the filename, in the format "name.type.ts".	[string] [default: "Component"]
-v, --view-encapsulation	The view encapsulation strategy to use in the new component.	[string] [choices: "Emulated", "None", "ShadowDom"]

Primera APP

```
$ ng new 01-clitest --standalone false
```

> Vamos a usar la forma anterior basada en módulo. Esto lo hacemos con la opción **--standalone false**

Comandos básicos del Ng Cli.

Podemos usar la opción **-d** la cual nos permite probar varios opciones sin aplicar cambios en el proyecto, por ejemplo: Creemos un componente home con y sin el flag **--flat**

Agregar componente

```
$ ng g c home -d
CREATE src/app/home/home.component.css (0 bytes)
CREATE src/app/home/home.component.html (19 bytes)
CREATE src/app/home/home.component.spec.ts (587 bytes)
CREATE src/app/home/home.component.ts (191 bytes)
UPDATE src/app/app.module.ts (467 bytes)
```

Crea el Componente en su propio directorio ./src/app/home, ahora usemos el --flat.

```
$ ng g c home -d --flat
CREATE src/app/home.component.css (0 bytes)
CREATE src/app/home.component.html (19 bytes)
CREATE src/app/home.component.spec.ts (587 bytes)
CREATE src/app/home.component.ts (191 bytes)
UPDATE src/app/app.module.ts (462 bytes)
```

Crea el componente en la ruta ./src/app

Coloquemos el Inline CSS **-s, --inline-style** y Inline Template **-t, --inline-template**

```
$ ng g c home -d --flat -s -t
CREATE src/app/home.component.spec.ts (587 bytes)
CREATE src/app/home.component.ts (183 bytes)
UPDATE src/app/app.module.ts (462 bytes)
```

Finalmente creemos el componente sin los archivos de prueba con la opción **--skip-tests**

```
$ ng g c home -d --flat -s -t --skip-tests
CREATE src/app/home.component.ts (183 bytes)
UPDATE src/app/app.module.ts (462 bytes)
```

Ejecutamos este último comando sin el -d para aplicar los cambios:

```
$ ng g c home --flat -s -t --skip-tests
CREATE src/app/home.component.ts (183 bytes)
UPDATE src/app/app.module.ts (462 bytes)
```

El componente creado es el siguiente:

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

@Component({
  selector: 'app-home',
  template: `
    <p>
      home works!
    </p>
  `,
  styles: ``
})
export class HomeComponent {
```

```
}
```

Adicionalmente, Ng Cli actualiza el **AppModule** y declara el nuevo componente. Pero, esto no lo hace visible al mundo exterior (fuera del componente) simplemente lo declara dentro del mismo módulo para su uso interno.

```
$ tree
.
├── app.component.css
├── app.component.html
├── app.component.spec.ts
├── app.component.ts
├── app.module.ts
├── app-routing.module.ts
└── home.component.ts  <=====
```

Opciones del NG Generate

Comando	Descripcion
ng g app-shell	Generates an application shell for running a server-side version of an app.
ng g application [name]	Generates a new basic application definition in the "projects" subfolder of the workspace.
ng g class [name]	Creates a new, generic class definition in the given project.
ng g component [name]	Creates a new, generic component definition in the given project.
ng g config [type]	Generates a configuration file in the given project.
ng g directive [name]	Creates a new, generic directive definition in the given project.
ng g enum [name]	Generates a new, generic enum definition in the given project.
ng g environments	Generates and configures environment files for a project.
ng g guard [name]	Generates a new, generic route guard definition in the given project.
ng g interceptor [name]	Creates a new, generic interceptor definition in the given project.
ng g interface [name] [type]	Creates a new, generic interface definition in the given project.
ng g library [name]	Creates a new, generic library project in the current workspace.
ng g module [name]	Creates a new, generic NgModule definition in the given project.
ng g pipe [name]	Creates a new, generic pipe definition in the given project.
ng g resolver [name]	Generates a new, generic resolver definition in the given project.

Comando	Descripcion
ng g service [name]	Creates a new, generic service definition in the given project.
ng g service-worker	Pass this schematic to the "run" command to create a service worker
ng g web-worker [name]	Creates a new, generic web worker definition in the given project.

Nueva Sección: Nueva Aplicación:

¿Qué veremos en esta sección?

En esta sección trabajaremos sobre:

- Uso del template administrativo
- Código fuente del template
- Uso de librerías externas
- Creación de los primeros componentes
- Separar el Login del template administrativo, ya que tienen estructuras diferentes
- Animaciones por CSS
- Respaldos en GitHub
- Preparar el proyecto que usaremos a lo largo del curso

Creando COmponentes y Estructura de Directorios

```
#Auth
$ ng g  c auth/login         --skip-tests -s
$ ng g  c auth/register      --skip-tests -s
#Pages
$ ng g  c pages/nopagefound  --skip-tests -s
$ ng g  c pages/dashboard    --skip-tests -s

$ ng g  c shared/breadcrumbs --skip-tests -s
$ ng g  c shared/side-bar    --skip-tests -s
$ ng g  c shared/header      --skip-tests -s
```

Importanto Plantilla existente

Usaremos una plantilla previamente diseñada, la cual contiene todas las dependeicnas, tales como imágenes, estilos, plugins, librerias de terceros, etc.

Copiaremos a nuestra carpeta **Assets** todos esos elementos y modificaremos nuestro index.html para incluir dichos elementos en nuestra página.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>AdminPro</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/png" sizes="16x16"
href="./assets/images/favicon.png">
```

```
  <!-- Bootstrap Core CSS -->
  <link href="./assets/plugins/bootstrap/css/bootstrap.min.css"
rel="stylesheet">
  <!-- Custom CSS -->
  <link href="./assets/css/style.css" rel="stylesheet">
  <!-- You can change the theme colors from here -->
  <link href="./assets/css/colors/default-dark.css" id="theme"
rel="stylesheet">


</head>
<body class="fix-header card-no-border fix-sidebar">
  <app-root></app-root>

   <!-- ============================================================== -->
    <!-- All Jquery -->
    <!-- ============================================================== -->
    <script src="./assets/plugins/jquery/jquery.min.js"></script>
    <!-- Bootstrap tether Core JavaScript -->
    <script src="./assets/plugins/bootstrap/js/popper.min.js"></script>
    <script src="./assets/plugins/bootstrap/js/bootstrap.min.js"></script>
    <!-- slimscrollbar scrollbar JavaScript -->
    <script src="./assets/js/perfect-scrollbar.jquery.min.js"></script>
    <!--Wave Effects -->
    <script src="./assets/js/waves.js"></script>
    <!--Menu sidebar -->
    <script src="./assets/js/sidebarmenu.js"></script>
    <!--stickey kit -->
    <script src="./assets/plugins/sticky-kit-master/dist/sticky-
kit.min.js"></script>
    <script src="./assets/plugins/sparkline/jquery.sparkline.min.js">
</script>
    <!--Custom JavaScript -->
    <script src="./assets/js/custom.min.js"></script>
    <!-- ============================================================== -->
    <!-- Style switcher -->
    <!-- ============================================================== -->
    <script src="./assets/plugins/styleswitcher/jQuery.style.switcher.js">
</script>
</body>
</html>
```

El proceso para incorporar una plantilla consiste en tomar cada parte y colocarla en el lugar adecuado, tal como si comenzó con el index.html.

La siguiente sección la dejaremos en el index.html, muestra una animación durante la carga de la página.

```
<!-- ============================================================== -->
<!-- Preloader - style you can find in spinners.css -->
<!-- ============================================================== -->
<div class="preloader">
```

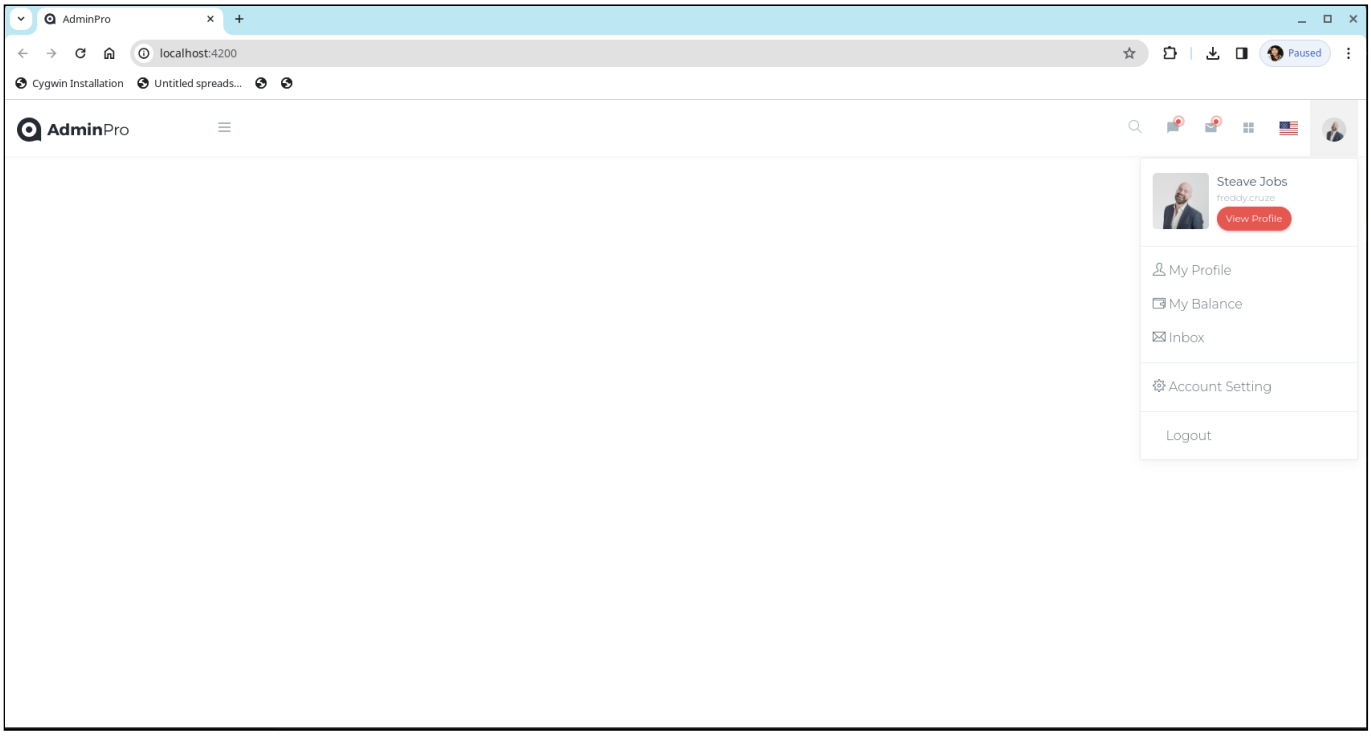
```
    <div class="loader">
        <div class="loader__figure"></div>
        <p class="loader__label">Admin Pro</p>
    </div>
</div>
```

Posteriormente sigue la sección del header. El HTML d esa sección la colocaremos en el componente **HeaderComponent**

```
<!-- ============================================================== -->
<!-- Topbar header - style you can find in pages.scss -->
<!-- ============================================================== -->
<header class="topbar">
  <!-- Copiamos toda la sección del header -->
</header>
```

Ahora podemos agregar el **main-wrapper** y el componente Header en el template del **AppComponent**

```
<div id="main-wrapper">
  <app-header></app-header>
</div>
```



Moveremos toda la sección del **Left Sidebar** a nuestro SideBarComponent.

```
<!-- ============================================================== -->
<!-- Left Sidebar - style you can find in sidebar.scss  -->
```

```
<!-- ============================================================ -->
  <aside class="left-sidebar"></aside>
```

Y luego lo agregamos a nuestro AppComponent:

```
<div id="main-wrapper">
  <app-header></app-header>
  <app-side-bar></app-side-bar>
</div>
```

Luego copiaremos algunas secciones directamentene en nuestro template principal, porque ellas mostraran internamente todas nuestras rutas

```
<div id="main-wrapper">
  <app-header></app-header>
  <app-side-bar></app-side-bar>

  <div class="page-wrapper">
    <div class="container-fluid">
      <!-- Sistema de Rutas -->
    </div>
  </div>
</div>
```

La siguiente sección que debemos de mover es la **Bread crumb** y lo hacemos a nuestro componente **BreadcrumbsComponent** Y movemos el footer directamente al template principal. No usaremos el Rigth Side Bar. El template principal queda así:

```
<div id="main-wrapper">
  <app-header></app-header>
  <app-side-bar></app-side-bar>

  <div class="page-wrapper">
    <div class="container-fluid">
      <app-breadcrumbs></app-breadcrumbs>

      <!-- Sistema de Rutas -->


      <!-- ============================================================ -
->
      <!-- Start Page Content -->
      <!-- ============================================================ -
->
      <div class="row">
        <div class="col-12">
            <div class="card">
```

```
                <div class="card-body">
                    This is some text within a card block.
                </div>
            </div>
          </div>
        </div>
        <!-- ============================================================== -
->
        <!-- End PAge Content -->
        <!-- ============================================================== -
->
    </div>
    <!-- ============================================================== -->
    <!-- footer -->
    <!-- ============================================================== -->
    <footer class="footer">
      © 2024 Admin Pro by wrappixel.com
    </footer>
    <!-- ============================================================== -->
    <!-- End footer -->
    <!-- ============================================================== -->
  </div>
</div>
```

Rutas Principales

Creamos un par de componentes más:

```
$ ng g c pages/progress -s --skip-tests
$ ng g c pages/grafica1 -s --skip-tests
```

Y luego definimos las rutas principales

```
const routes: Routes = [
  { path: 'dashboard', component: DashboardComponent },
  { path: 'login', component: LoginComponent },
  { path: 'register', component: RegisterComponent },
  { path: 'progress', component: ProgressComponent },
  { path: 'grafica1', component: Grafica1Component },
  { path: '', redirectTo: '/dashboard', pathMatch: 'full' },
  { path: '**', component: NopagefoundComponent }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Finalmente implementamos nuestro **router-outlet** en el AppComponent para mostrar cada uno de los componentes dentro del contenedor **container-fluid**

Rutas Secundarias

En este momento todos los componente, incluyendo el Login y Register se muestran como un componente más dentro del Dashboard. El Dashboard debe de mostrarse únicamente si el usuario está logeado.

La idea es crear un componente que muestre todo el Layout actual, el dashdoard con sus componentes, para ello vamos a mover todo el código HTML del AppComponent a un nuevo componente:

```
$ ng g c pages/pages -s --skip-tests --flat
```

Este componente se va a mostrar únicamente cuando el usuario esté logeado.

Haremos estos cambios en nuestro sistema de Rutas:

```
const routes: Routes = [

  {
    path: '',
    component: PagesComponent,
    children: [
      { path: 'dashboard', component: DashboardComponent },
      { path: 'progress', component: ProgressComponent },
      { path: 'grafica1', component: Grafica1Component },
      { path: '', redirectTo: '/dashboard', pathMatch: 'full' },
    ],
  },
  { path: 'login', component: LoginComponent },
  { path: 'register', component: RegisterComponent },
  { path: '**', component: NopagefoundComponent }

];
```

Aca lo que estamos indicando es que login y register van a renderizarse fuera del componente Pages, que es el que renderiza el template con los headers, sideBar, etc.

De esta forma tenemos las rutas de login y register independientes y podemos implementar su propio template.