# Angular 4

Angular 4 (de ahora en adelante solo Angular) es un framework para construir aplicaciones cliente en HTML y un lenguaje bien sea Javascript, o un lenguaje que compile a Javascript tales como TypeScript o Dart.

Escribimos la aplicación componiendo templates HTML con etiquetas de Angular, escribiendo clases componentes para controlar dichos templates, agregando lógica de la aplicación en servicios, y agrupando los componentes y servicios en módulos.

## Angular CLI

Es un plugin para NodeJS que simplifica la generación y gestión de proyectos de Angular 4.

Para comenzar instalaremos el plugin de manera global con el siguiente comando:

npm install -g @angular/cli

## Hola Mundo Angular

A continuación, crearemos nuestra primera aplicación Angular usando Angular CLI.

1. Creamos el proyecto con Angular CLI ejecutando el siguiente comando:

ng new contacts-ui

El anterior comando creara el proyecto configurando las dependencias para trabajar con Typescript y Angular 4; así como, las dependencias de desarrollo necesarias para ejecutar las pruebas de desarrollo y automáticas de la aplicación.

1. Para iniciar el proyecto ejecutamos el siguiente nos movemos al directorio de la aplicación (cd contacts-ui) y ejecutamos el siguiente comando:

ng serve

El anterior comando inicia un servidor de desarrollo que nos permitirá acceder a la aplicación en el URL: <http://localhost:4200>, es posible cambiar el puerto de escucha del servidor pasando como parámetro --port <puerto> o la forma abreviada –p <puerto>.

Ahora que hemos visto “Welcome to app!!” en nuestro navegador WEB vamos a seguir con los conceptos de Angular y como creamos una aplicación para el mundo real.

Para este proyecto de ejemplo usaremos Angular Material que es una librería de componentes de interface gráfica para Angular.

Para instalar y configurar la libraría realizaremos los siguientes pasos:

1. Instalamos las dependencias

npm install --save @angular/material hammerjs

1. En src/styles.css importamos el tema y las fuentes usadas para los iconos

@import '~@angular/material/prebuilt-themes/deeppurple-amber.css';

@import '~https://fonts.googleapis.com/icon?family=Material+Icons';

1. En el módulo importamos los módulos de animaciones, material y hammerjs

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

import { MaterialModule } from '@angular/material';

import 'hammerjs';

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule,

BrowserAnimationsModule,

MaterialModule

],

providers: [],

bootstrap: [AppComponent]

})

Desarrollaremos la aplicación de contactos (para la cual ya desarrollamos la API de servicios REST)

Angular CLI brinda la facilidad de generar automáticamente módulos, componentes, directivas, pipes, servicios, clases, guards, interfaces y enumeraciones usando el comando ng generate <blueprint> <nombre> o la versión abreviada ng g <blueprint> <nombre>. A continuación, la descripción de cada uno de los blueprints soportados por Angular CLI y lo que representa cada uno de ellos al interior del framework.

## Módulo

Es una clase adornada con el decorador ***@NgModule***, pasamos como parámetro al decorador un objeto metadata que indica a Angular como compilar y ejecutar el módulo.

Ejemplo:

ng g module authorization

## Componente

Es una clase adornada con el decorador ***@Component***, que controla parte de la pantalla de la aplicación llamada vista, pasamos al decorador como parámetro un objeto metadata que indica a Angular el comportamiento del componente, como debe inicializarse, el selector que indica donde debe ser inyectado en el DOM, el template que representa el HTML renderizado por el componente, el archivo de estilos usado por el componente.

Ejemplo:

ng g component contact-list

## Servicio

Es una clase adornada con el decorador @Injectable y provee funcionalidades comunes a múltiples componentes o módulos (reutilización de funcionalidades).

Ejemplo:

ng g service oauthv2-token-handler

## Templates

Usamos los templates para definir la representación gráfica de los diferentes componentes usando HTML regular, excepto por las directivas de Angular que usamos para controlar como Angular interactúa en la renderización del componente. El template puede ser definido en la metadata del decorador del componente inline usando la propiedad template o en un archivo independiente usando la propiedad templateURL.

## Data binding

Es un mecanismo soportado por Angular que permite coordinar la interacción de los datos asociados al componente y el template o viceversa. Agregamos etiquetas de binding a los templates HTML para indicar a Angular como conectar ambos extremos.

## Directivas

Son clases adornadas con el decorador ***@Directive***, pasamos como parámetro al decorador un objeto metadata para indicar a Angular como la directiva interactúa con el DOM para transformarlo. Existen directivas estructurales que alteran la distribución de los elementos del DOM, agregando, eliminando y reemplazando, y directivas de atributo que alteran la apariencia de un elemento existente.

Ejemplo:

ng g directive contacts-map