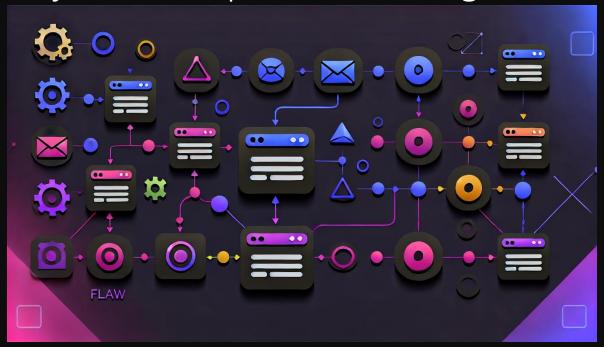
Flujo de una Aplicación en Angular



Flujo de una Aplicación de Angular

En esta lección vamos a ver una explicación detallada y paso a paso del flujo de una aplicación en Angular, junto con un diagrama conceptual para entender cómo se ejecuta una aplicación Angular desde el código hasta el navegador.

1. Configuración Inicial

- package.json: Define las dependencias del proyecto y scripts esenciales. Es el archivo que gestiona la instalación de todos los paquetes necesarios a través de npm install.
- angular.json: Este archivo sigue siendo fundamental para la configuración del proyecto, incluyendo detalles sobre la compilación, los entornos, y la estructura de archivos.

2. Bootstrap de la Aplicación (Punto de Entrada)

• main.ts: Este archivo sigue siendo el punto de entrada de la aplicación. Sin embargo, en lugar de hacer bootstrap a un módulo (AppModule) como se hace con aplicaciones basadas en Módulos, ahora se hace directamente a un componente standalone. El código podría verse así:

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { AppComponent } from './app/app.component';
```

Ing. Ubaldo Acosta Universidad Angular

```
platformBrowserDynamic().bootstrapComponent(AppComponent)
   .catch(err => console.error(err));
```

• Aquí, AppComponent es un standalone component que se usa para arrancar la aplicación directamente, eliminando la necesidad de un módulo raíz (App Module).

3. Componente Standalone Principal

• app.component.ts: Este es el componente raíz de la aplicación. Dado que es un standalone component, se define con standalone: true y se importa directamente en main.ts. Su selector (<app-root>) se utiliza en index.html para renderizar la aplicación.

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

@Component({
    selector: 'app-root',
    standalone: true,
    templateUrl: './app.component.html',
    styleUrls: ['./app.component.css'],
    imports: [/* otros componentes o módulos standalone */]
})
export class AppComponent {
    title = 'Mi Aplicación Angular';
}
```

4. Renderizado en el Navegador

- Carga Inicial: El navegador carga el archivo index.html, que contiene el elemento <app-root>. Angular lo reemplaza con el contenido del AppComponent.
- Angular CLI: Utilizando ng serve, Angular CLI compila la aplicación, convirtiendo TypeScript en JavaScript, y sirve la aplicación en un servidor de desarrollo.
- Rutas y Navegación: Las rutas siguen funcionando de manera similar, pero ahora pueden ser definidas dentro de los standalone components o en archivos de configuración de rutas.

5. Ciclo de Vida de los Componentes

- **Creación**: Los componentes standalone son instanciados y preparados para su renderizado.
- Inicialización: Los hooks de ciclo de vida (ngonīnit, etc) se ejecutan para configurar los componentes. La mayoría de estos métodos (hooks) se ejecutan de manera automática por Angular.
- **Detección de Cambios**: Angular monitorea los cambios en los datos vinculados a la vista y actualiza el DOM en consecuencia.

Ing. Ubaldo Acosta Universidad Angular

 Destrucción: Los componentes que ya no son necesarios se eliminan del DOM, y Angular limpia los recursos asociados.

6. Archivos Opcionales

- polyfills.ts: Este archivo asegura la compatibilidad con navegadores más antiguos, permitiendo que la aplicación funcione en entornos más diversos.
- **styles.css**: Aquí se definen los estilos globales que se aplican a toda la aplicación.

Diagrama Conceptual del Flujo de Ejecución para Standalone Components

El siguiente diagrama conceptual ilustra cómo funciona el flujo de una aplicación Angular 18 utilizando standalone components:

Este flujo actualizado refleja el enfoque más modular y directo de Angular, donde se eliminan los módulos tradicionales en favor de standalone components, permitiendo una estructura más ligera y sencilla para las aplicaciones.

Saludos!

Ing. Ubaldo Acosta

Fundador de GlobalMentoring.com.mx