# **Introducción al Framework**

Angular es un framework de desarrollo **Front-End** diseñado para construir **Single Page Applications (SPA)**, donde la experiencia del usuario es fluida y dinámica, sin necesidad de recargar completamente la página al interactuar con ella. Fue desarrollado por Google y lanzado por primera vez en 2010 como AngularJS, su versión inicial.

## **Origen y Evolución: AngularJS vs Angular**

1. **AngularJS (2010):**Una versión basada en JavaScript que introdujo el concepto de **data binding** bidireccional, facilitando la sincronización entre el modelo de datos y la vista. Sin embargo, con el tiempo, su arquitectura demostró limitaciones en aplicaciones grandes debido a problemas de rendimiento y escalabilidad.
2. **Angular (desde 2016):**Un rediseño completo utilizando **TypeScript**, un superconjunto de JavaScript que ofrece tipado estático. Angular eliminó las restricciones de AngularJS y adoptó una arquitectura basada en **Componentes y Módulos**, lo que mejora la mantenibilidad, escalabilidad y rendimiento en aplicaciones modernas.

## **Diferencias Clave entre AngularJS y Angular**

| **Característica** | **AngularJS (1.x)** | **Angular (2+)** |
| --- | --- | --- |
| **Lenguaje** | JavaScript | TypeScript |
| **Arquitectura** | MVC (Model-View-Controller) | Basada en Componentes |
| **Data Binding** | Bidireccional | Bidireccional, pero más eficiente |
| **Rendimiento** | Limitado en aplicaciones grandes | Altamente optimizado |
| **Soporte** | Descontinuado | Actualizaciones regulares |

## **Comparación con Otros Frameworks**

Angular es frecuentemente comparado con otros frameworks populares como **React** y **Vue.js**:

| **Característica** | **Angular** | **React** | **Vue.js** |
| --- | --- | --- | --- |
| **Lenguaje Base** | TypeScript | JavaScript/JSX | JavaScript |
| **Arquitectura** | Completa (framework) | Biblioteca para UI | Progresiva (adoptable) |
| **Curva de Aprendizaje** | Empinada | Moderada | Baja |
| **Escalabilidad** | Alta | Alta | Moderada |

## **Aplicaciones Prácticas Recomendadas**

Angular es ideal para desarrollar aplicaciones que requieren una estructura sólida y un alto rendimiento, como:

* **Dashboards analíticos**: Herramientas interactivas con gráficos en tiempo real.
* **Sistemas empresariales**: ERP o CRM que gestionen grandes volúmenes de datos.
* **Aplicaciones web progresivas (PWA)**: Experiencias que funcionan offline.

## **Gráfico Conceptual: Estructura de una Aplicación Angular**

1. **App Module**Contiene los módulos principales y dependencias de la aplicación.
2. **Components**Definen las interfaces de usuario y la lógica específica de cada vista.
3. **Services**Gestionan la lógica de negocio y el acceso a datos.

css

[ App Module ] --> [ Components ] <--> [ Services ]

|

[ Templates ]

# 

# **Comparativa: Angular vs. AngularJS**

| **Criterio** | **AngularJS (2010)** | **Angular (Desde 2016)** | **Diferencia Clave** |
| --- | --- | --- | --- |
| **Lenguaje Base** | **JavaScript** | **TypeScript** | TypeScript permite tipado estático y mejor depuración. |
| **Arquitectura** | **MVC (Model-View-Controller)** | **Basada en Componentes** | Angular usa componentes reutilizables y modulares. |
| **Data Binding** | **Bidireccional** | **Bidireccional (optimizado)** | Angular tiene mejor rendimiento con la detección de cambios. |
| **Inyección de Dependencias** | **No disponible** | **Sí, jerárquica** | La inyección jerárquica mejora la eficiencia. |
| **Rendimiento** | **Limitado en grandes apps** | **Optimizado, hasta 5x más rápido** | Angular usa detección de cambios unidireccional. |
| **Curva de Aprendizaje** | **Moderada** | **Alta al inicio, luego estable** | Angular requiere aprendizaje de TypeScript y CLI. |
| **Reutilización de Código** | **Difícil** | **Fácil con módulos y componentes** | La componentización permite reutilización de lógica y vista. |
| **Manejo de Archivos** | **Sin estructura estándar** | **Estructura generada con CLI** | Angular CLI crea una estructura lista para trabajar. |
| **Modularización** | **Limitada** | **Total (componentes y módulos)** | Angular tiene módulos para separar la lógica. |
| **Soporte y Actualizaciones** | **Descontinuado** | **Actualizaciones regulares** | Angular recibe actualizaciones constantes de Google. |
| **Control de Versiones** | **Versión única (1.x)** | **Versiones 2+ (actualmente Angular 15)** | Angular sigue un ciclo de actualizaciones periódicas. |
| **Uso de CLI** | **No disponible** | **CLI para automatizar tareas** | La CLI permite crear, compilar y ejecutar proyectos con comandos. |
| **Soporte para Móviles (PWA)** | **No soportado** | **Soporte para PWA** | Angular permite crear Aplicaciones Web Progresivas (PWA). |
| **SEO** | **Difícil de optimizar** | **Mejor con Angular Universal** | Angular Universal permite la renderización del lado del servidor. |
| **Compatibilidad** | **Obsoleto, no recomendado** | **Recomendado para apps modernas** | AngularJS ya no recibe soporte oficial. |

# **Herramientas esenciales**

Angular requiere un entorno de desarrollo adecuado para garantizar un flujo de trabajo eficiente y organizado. A continuación, se describen las herramientas y pasos necesarios para configurar el entorno inicial.

## **Herramientas Necesarias**

1. **Node.js**Angular utiliza Node.js para gestionar las dependencias del proyecto mediante su gestor de paquetes, npm (Node Package Manager).
2. **npm (Node Package Manager)**npm se instala automáticamente con Node.js y se usa para instalar Angular y sus dependencias.
3. **Angular CLI (Command Line Interface)**La CLI de Angular simplifica la creación, configuración y administración de proyectos Angular mediante comandos específicos.
4. **Editores de Código Recomendados**
   * **Visual Studio Code (VS Code):** Popular por su soporte de extensiones, autocompletado y depuración.
   * **WebStorm:** Ideal para proyectos complejos con herramientas integradas.
   * **Sublime Text:** Ligero, aunque requiere configuraciones adicionales.

## **Pasos para la Instalación y Configuración Inicial**

### **1. Instalar Node.js y npm**

* Descarga la última versión LTS de Node.js desde [nodejs.org](https://nodejs.org/).
* Durante la instalación, asegúrate de incluir npm.

Verifica la instalación abriendo la terminal y ejecutando:  
  
  
node -v

npm -v

### **2. Instalar Angular CLI**

Una vez que Node.js y npm estén instalados, instala Angular CLI globalmente:  
  
  
npm install -g @angular/cli

Verifica la instalación de Angular CLI:  
bash  
ng version

### **3. Crear un Proyecto Angular**

Usa Angular CLI para generar un proyecto base:  
  
ng new nombre-del-proyecto

* Durante la configuración, selecciona las opciones según tus necesidades (por ejemplo, incluir Angular routing y el formato de hojas de estilo preferido como CSS, SCSS, etc.).

### **4. Ejecutar el Proyecto**

Navega al directorio del proyecto creado:  
  
cd nombre-del-proyecto

Inicia el servidor de desarrollo:  
  
ng serve

* Accede a la aplicación en tu navegador en http://localhost:4200.

## **Configuración Adicional Opcional**

1. **Instalar Extensiones en VS Code**
   * Recomendadas:
     + **Angular Language Service:** Autocompletado y ayuda contextual.
     + **Prettier:** Formateo automático del código.
     + **ESLint:** Herramienta para asegurar la calidad del código.
2. **Configurar Git (Opcional)**

Inicia un repositorio Git para gestionar el control de versiones:  
  
git init

git add .

git commit -m "Proyecto Angular inicial"

# **Instalación del entorno**

Angular requiere la instalación de **Node.js** y su gestor de paquetes **npm**, además de la **Angular CLI**. A continuación, se presenta una guía detallada para instalar estas herramientas, junto con la explicación de comandos esenciales.

## **1. Instalar Node.js y npm**

Node.js es una plataforma que permite ejecutar JavaScript fuera del navegador y es esencial para gestionar dependencias en Angular.

### **Paso 1: Descargar Node.js**

1. Visita la página oficial:<https://nodejs.org>.
2. Descarga la versión LTS (Long Term Support) para garantizar estabilidad en el desarrollo.

### **Paso 2: Instalar Node.js**

1. Ejecuta el instalador descargado.
2. Sigue los pasos del asistente:
   * Acepta los términos y condiciones.
   * Selecciona la carpeta de instalación.
   * Asegúrate de que la casilla **"Add to PATH"** esté marcada.

### **Paso 3: Verificar la Instalación**

Abre una terminal y ejecuta los siguientes comandos para comprobar que Node.js y npm están instalados correctamente:

node -v

npm -v

* **Salida esperada:**
  + Una versión para Node.js, como v18.x.x.
  + Una versión para npm, como 8.x.x.

## **2. Instalar Angular CLI**

La **CLI de Angular** simplifica el desarrollo al proporcionar comandos para generar y gestionar proyectos.

### **Paso 1: Instalar Angular CLI**

Ejecuta el siguiente comando en la terminal para instalar Angular CLI de forma global:

npm install -g @angular/cli

### **Paso 2: Verificar la Instalación**

Comprueba que Angular CLI se instaló correctamente ejecutando:

ng version

* **Salida esperada:** Información sobre las versiones de Angular CLI y dependencias clave.

## **3. Crear un Proyecto Angular**

### **Paso 1: Generar un Nuevo Proyecto**

Usa Angular CLI para crear un proyecto base:

ng new nombre-del-proyecto

* Angular CLI te preguntará:
  + **¿Deseas Angular routing?** Responde "Yes" o "No".
  + **Formato de hoja de estilos:** Elige entre CSS, SCSS, LESS, etc.

### **Paso 2: Navegar al Proyecto**

cd nombre-del-proyecto

### **Paso 3: Ejecutar el Proyecto**

Inicia el servidor de desarrollo:

ng serve

* Abre tu navegador y accede a: http://localhost:4200.

## **Comandos Esenciales**

**Verificar las Versiones Instaladas**  
ng version

1. Muestra información sobre la versión de Angular CLI y las dependencias del proyecto.

**Instalar Dependencias**  
npm install

1. Descarga las dependencias listadas en el archivo package.json.

**Iniciar el Servidor de Desarrollo**  
ng serve

1. Compila y ejecuta la aplicación localmente.

Siguiendo estos pasos, tu entorno estará configurado y listo para comenzar a desarrollar aplicaciones Angular.

# **Estructura del Proyecto**

Cuando se genera un proyecto con Angular CLI, se crea una estructura organizada que facilita el desarrollo y mantenimiento. A continuación, se describen los archivos y carpetas clave, así como sus funciones principales.

## **Carpetas Principales**

### **1. src/**

La carpeta principal donde se encuentra todo el código fuente del proyecto. Contiene subcarpetas y archivos esenciales para la aplicación.

* **src/assets/**Destinada a almacenar recursos estáticos como imágenes, fuentes, y otros archivos que no cambian durante la ejecución.
* **src/environments/**Contiene archivos de configuración para diferentes entornos (por ejemplo, desarrollo y producción). Facilita la gestión de variables específicas según el entorno.

## **Archivos Clave**

### **1. app.module.ts**

* Es el **módulo raíz** de la aplicación.
* Define los **componentes**, **directivas**, y **servicios** que se utilizan en el proyecto.
* Importa otros módulos necesarios y declara las dependencias.

**Ejemplo de funcionalidad:**Si se agrega un nuevo componente, este debe ser declarado aquí para que Angular lo reconozca.

### **2. app.component.ts**

* Es el **componente principal** de la aplicación.
* Define la lógica y estructura de la vista inicial que se muestra al usuario.
* Contiene:
  + **Decorador @Component:** Describe metadatos como el selector y las rutas de los archivos HTML y CSS asociados.
  + **Clase TypeScript:** Gestiona la lógica del componente.

**Ejemplo de funcionalidad:**Cualquier texto o elemento visible en la pantalla inicial de la aplicación suele estar definido en este componente.

### **3. index.html**

* Archivo HTML principal que sirve como contenedor para la aplicación Angular.
* Solo tiene un <app-root> que referencia el **componente raíz**.

**Ejemplo de funcionalidad:**Este archivo nunca se modifica directamente para agregar contenido, ya que Angular inyecta dinámicamente los componentes en el DOM.

### **4. main.ts**

* Punto de entrada de la aplicación.
* Se encarga de inicializar el módulo raíz (AppModule) y arrancar la aplicación.

## **Flujo Básico**

1. **El navegador carga index.html.**
2. **index.html referencia el componente raíz (<app-root>), que está definido en app.component.ts.**
3. **main.ts arranca la aplicación a partir de AppModule.**

# **Integración con Git**

La integración de un proyecto Angular con Git y GitHub permite gestionar versiones, colaborar de manera eficiente y mantener un historial de cambios. A continuación, se explican los pasos para configurar un repositorio Git, vincularlo con GitHub y subir el proyecto inicial.

## **1. Configurar Git en tu Proyecto**

### **Paso 1: Inicializar un Repositorio Git Local**

1. Abre una terminal en la raíz del proyecto Angular.

Ejecuta el comando:  
  
git init

1. Esto inicializa un repositorio local.

### **Paso 2: Agregar Archivos al Repositorio**

1. Asegúrate de incluir un archivo .gitignore (Angular CLI lo genera automáticamente) para excluir archivos innecesarios como node\_modules.

Agrega todos los archivos al área de preparación:  
  
git add .

### **Paso 3: Realizar el Primer Commit**

Guarda los cambios en el repositorio local con un mensaje descriptivo:

git commit -m "Inicialización del proyecto Angular"

## **2. Vincular el Repositorio Local con GitHub**

### **Paso 1: Crear un Repositorio en GitHub**

1. Accede a [GitHub](https://github.com) y crea un nuevo repositorio.
2. Define un nombre para el repositorio (por ejemplo, mi-proyecto-angular) y selecciona las opciones deseadas (público o privado).

### **Paso 2: Conectar el Repositorio Local con GitHub**

1. Copia la URL del repositorio en GitHub.

En la terminal, configura el repositorio remoto:  
  
git remote add origin <URL-del-repositorio>

Verifica que el repositorio remoto esté vinculado correctamente:  
  
git remote -v

## **3. Subir el Proyecto Inicial a GitHub**

Envía los cambios al repositorio remoto en GitHub:

git push -u origin main

* Si tu rama principal tiene otro nombre, como master, usa ese en lugar de main.

## **4. Gestionar Cambios y Versiones**

### **Realizar Cambios y Guardarlos**

Después de realizar modificaciones en tu proyecto, agrega los cambios al área de preparación:  
  
git add .

Realiza un commit con un mensaje descriptivo:  
  
git commit -m "Descripción de los cambios"

### **Subir los Cambios a GitHub**

Envía los nuevos commits al repositorio remoto:

git push

### **Recuperar Cambios del Repositorio Remoto**

Si trabajas en equipo, asegúrate de obtener los últimos cambios antes de continuar:

git pull

## **5. Buenas Prácticas**

* Realiza commits frecuentes y descriptivos para mantener un historial claro.
* Sincroniza regularmente tu repositorio con git pull antes de trabajar en nuevas funcionalidades.
* Usa ramas (git branch) para trabajar en cambios importantes sin afectar la rama principal.

## 

## **Actividades con Posibles Soluciones**

### **Actividad 1: ¿Por qué una SPA?**

**Duración:** 5 minutos

**Consigna:**Reflexionar sobre la diferencia entre una página web tradicional y una SPA (Single Page Application).

**Posible solución:**

* En una **página web tradicional**, cada vez que se hace clic en un enlace, se recarga toda la página, generando una nueva solicitud al servidor.
* En una **Single Page Application (SPA)**, solo se actualizan ciertas secciones de la página, sin necesidad de recargarla por completo. Esto se logra mediante la manipulación dinámica del DOM y el uso de técnicas de AJAX.

### **Actividad 2: Instalación de herramientas**

**Duración:** No especificada

**Consigna:**Instalar y verificar la instalación de Node.js, npm y Angular CLI.

**Posible solución:**

**Comandos de instalación y verificación**:  
  
node -v # Debería mostrar la versión de Node.js, por ejemplo, v18.x.x

npm -v # Debería mostrar la versión de npm, por ejemplo, 8.x.x

npm install -g @angular/cli # Instala Angular CLI

ng version # Debería mostrar la versión de Angular CLI instalada

1. **Posibles errores y soluciones**:
   * **Error:** "ng no se reconoce como un comando interno o externo".  
     **Solución:** Agregar la carpeta de Angular CLI al **PATH** del sistema.
   * **Error:** Permisos insuficientes en el sistema.  
     **Solución:** Ejecutar el comando con privilegios de administrador o usar sudo.

### **Actividad 3: Creación del primer proyecto Angular**

**Duración:** 10 minutos

**Consigna:**Crear un proyecto Angular y verificar que funcione correctamente.

**Posible solución:**

**Comando para crear el proyecto**:  
  
ng new mi-primera-app

1. **Acciones requeridas durante la creación**:
   * Elegir "No" para la opción de enrutamiento.
   * Seleccionar **CSS** como opción de preprocesador de estilos.

**Comando para ejecutar la aplicación**:  
  
cd mi-primera-app

ng serve -o

1. **Validación de la solución**:
   * Se debe abrir el navegador con la URL [**http://localhost:4200**](http://localhost:4200/) y mostrar el mensaje **"Welcome to mi-primera-app!"**.

### **Actividad 4: Configuración de Git y GitHub**

**Duración:** No especificada

**Consigna:**Configurar Git y subir el proyecto Angular a un repositorio de GitHub.

**Posible solución:**

**Comandos de configuración de Git**:  
  
git config --global user.name "Tu Nombre"

git config --global user.email "tu\_email@example.com"

**Comandos para subir el proyecto a GitHub**:  
  
git init

git add .

git commit -m "Primer commit"

git remote add origin https://github.com/tu-usuario/mi-primera-app.git

git push -u origin main

**Validación de la solución:**

* Verificar en la interfaz de GitHub que el repositorio contiene todos los archivos del proyecto Angular.

### **Actividad 5: Exploración de app.module.ts**

**Duración:** No especificada

**Consigna:**Explorar el archivo **app.module.ts** e identificar sus secciones principales.

**Posible solución:**

1. **Secciones principales de app.module.ts**:
   * **Imports:** Importación de otros módulos y dependencias necesarias para la aplicación.
   * **Declarations:** Declaración de los componentes utilizados en la aplicación.
   * **Bootstrap:** Indica el componente raíz que se cargará al inicio.

**Ejemplo de app.module.ts**:  
  
import { NgModule } from '@angular/core';

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

import { AppComponent } from './app.component';

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

### **Actividad 6: Publicación en GitHub**

**Duración:** 5 minutos

**Consigna:**Publicar el proyecto Angular en un repositorio de GitHub.

**Posible solución:**

**Comandos de publicación en GitHub**:  
  
git init

git add .

git commit -m "Subida inicial del proyecto"

git remote add origin https://github.com/tu-usuario/mi-primera-app.git

git push -u origin main

1. **Validación de la solución:**
   * Verificar que el proyecto se haya subido correctamente al repositorio de GitHub.

### **Actividad 7: Flujo de desarrollo Angular**

**Duración:** 10 minutos

**Consigna:**Ejecutar, editar y observar el flujo de cambios en la aplicación Angular.

**Posible solución:**

**Comandos para ejecutar la aplicación**:  
  
ng serve -o

1. **Acciones requeridas**:
   * Abrir el archivo **app.component.html** y modificar el contenido.
   * Guardar los cambios y verificar la actualización automática en la página.
2. **Errores comunes y soluciones**:
   * **Error de sintaxis**: Verificar que todas las etiquetas HTML estén cerradas correctamente.
   * **Error de imports**: Asegurarse de que todas las dependencias estén correctamente importadas en **app.module.ts**.

### **Actividad 8: Creación de componentes personalizados**

**Duración:** 15 minutos

**Consigna:**Crear tres componentes personalizados usando Angular CLI y personalizarlos.

**Posible solución:**

**Comandos para generar componentes**:  
  
ng generate component student

ng generate component toolbar

ng generate component navbar

1. **Acciones requeridas**:
   * Editar el contenido de **student.component.html**, **toolbar.component.html** y **navbar.component.html**.
   * Personalizar los archivos de estilo **.css** para cada componente.
2. **Validación de la solución:**

Integrar los componentes en **app.component.html**:  
  
<app-student></app-student>

<app-toolbar></app-toolbar>

<app-navbar></app-navbar>

### **Actividad 9: Componentización**

**Duración:** 15 minutos

**Consigna:**Crear un layout con los componentes **student**, **toolbar** y **navbar** utilizando CSS Grid.

**Posible solución:**

**Definir la disposición del layout usando CSS Grid**:  
  
  
.container {

display: grid;

grid-template-areas:

'toolbar toolbar'

'navbar content';

grid-template-columns: 1fr 3fr;

grid-template-rows: auto 1fr;

}

.toolbar {

grid-area: toolbar;

}

.navbar {

grid-area: navbar;

}

.content {

grid-area: content;

}

**HTML de la página**:  
  
<div class="container">

<app-toolbar class="toolbar"></app-toolbar>

<app-navbar class="navbar"></app-navbar>

<div class="content">

Contenido principal

</div>

</div>

1. **Validación de la solución:**
   * Verificar que la página se divida en tres secciones (toolbar, navbar y content) de acuerdo con la estructura de CSS Grid.