



  
CJAVA



**CJAVA**

siempre para apoyarte



**Visión**

Poder aportar al desarrollo del País usando tecnología Java.

# Quienes Somos

Somos una organización orientada a **desarrollar, capacitar e investigar tecnología JAVA** a través de un prestigioso staff de profesionales a nivel nacional.





**CJAVA**

siempre para apoyarte

**cjavaperu.com**  
**info@cjavaperu.com**

COMUNIDAD

ACADEMICA



**CJAVA**

siempre para apoyarte





**CJAVA**  
siempre para apoyarte

# Servicio de Capacitación

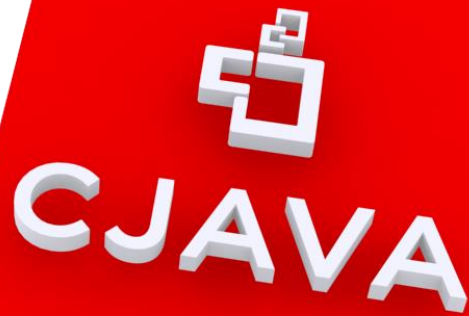
- **Programer** (Antes Java Developer Junior - 80 horas)  
[Certificado: Java Programer]
- **Developer** (Antes Java Web Developer - 80 horas)  
[Certificado: Java Developer]
- **Expert** (Antes Java Developer Senior - 80 horas)  
[Certificado: Java Expert]
- **Arquitect** (Antes ADS-RUP - 80 horas)  
[Certificado: Java Arquitect]
- **Carrera** (12 meses de contenido Java)  
[Diploma: Carrera Java]

Architect

Expert

Developer

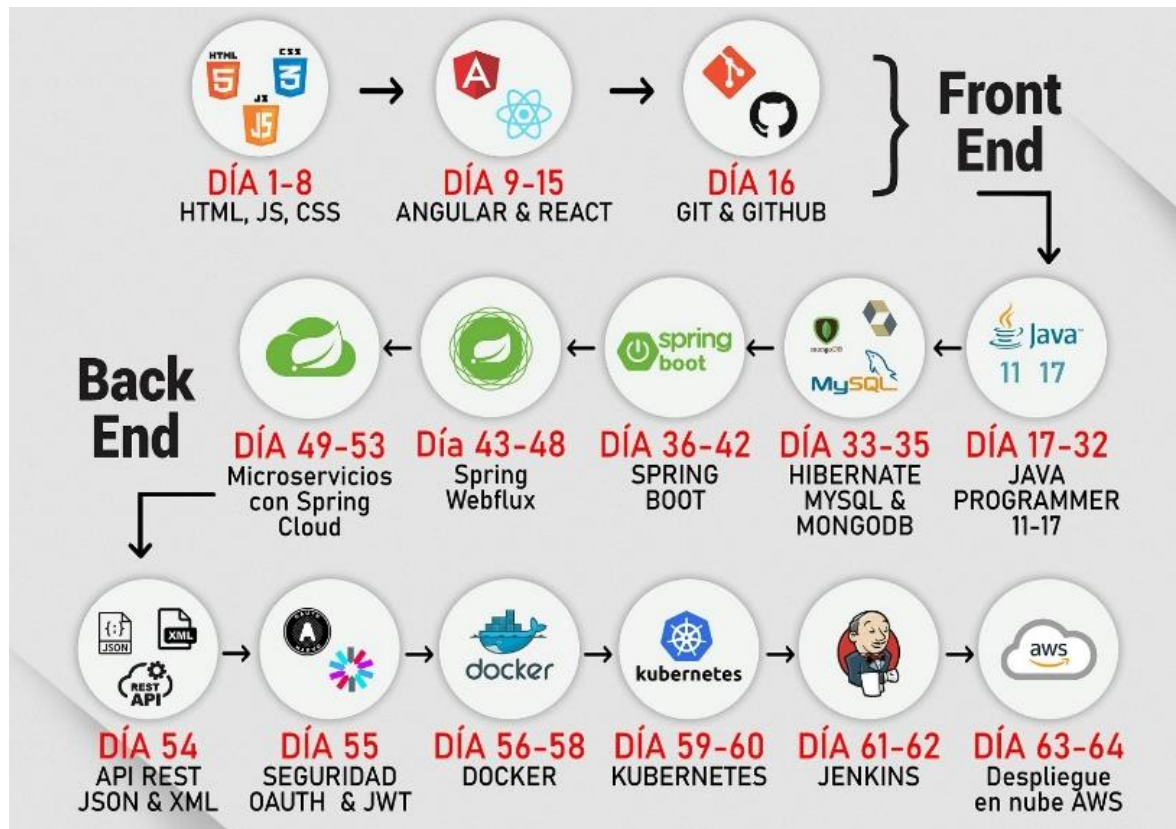
Programmer



# Java FullStack Developer

emaravi@cjavaperu.com

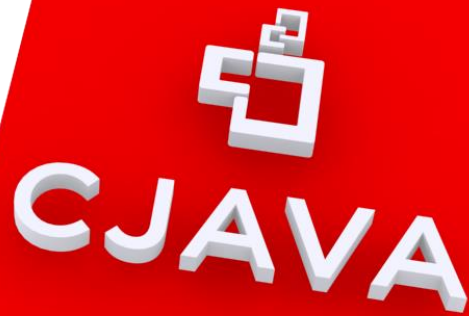
# Introducción a Full Stack Development



# Introducción a Full Stack Development

Front-end Development: Discutir las tecnologías y herramientas utilizadas en el desarrollo front-end, como HTML, CSS, JavaScript y React.





# Estructura de un app y componentes

Edwin Maravi

# Resultado de aprendizaje

Comprende y analiza el concepto del Framework Angular. Asimismo, instala de forma correcta Angular y crea un nuevo proyecto con la herramienta CLI.

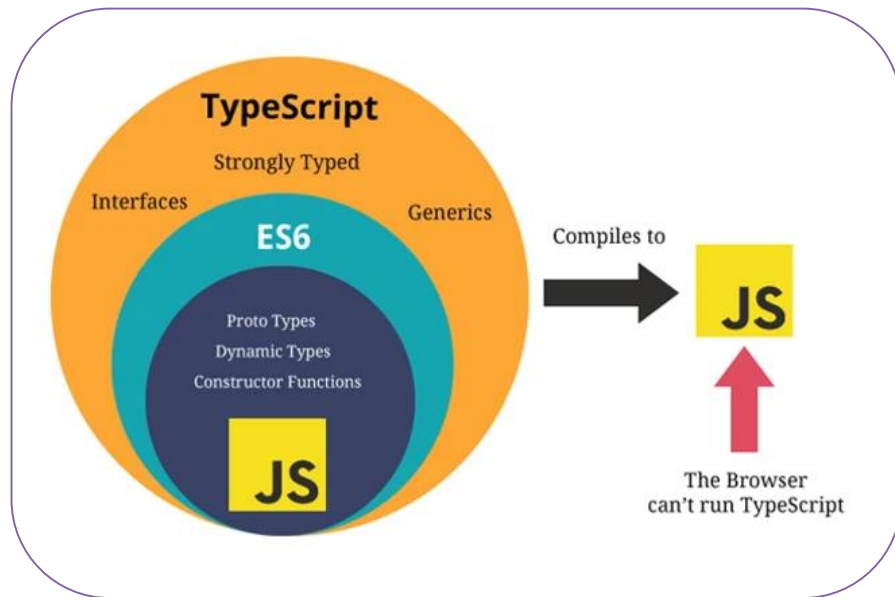
# Contenidos o temas

- TypeScript
- Diferencia entre JavaScript y TypeScript
- ¿Qué es Angular?
- Características de Angular
- Instalación de Angular
- Estructura de un proyecto en Angular



# TypeScript

- TypeScript es un lenguaje de desarrollo de JavaScript de la era moderna.
- Es un lenguaje compilado estáticamente para escribir código JavaScript claro y simple.
- Se puede ejecutar en Node JS o en cualquier navegador que admita ECMAScript 3 versiones más recientes.



Recuperado de <https://openwebinars.net/blog/que-es-typescript/>

# TypeScript vs. JavaScript

Observa el siguiente cuadro comparativo con las diferencias:

TypeScript	JavaScript
Se conoce como un lenguaje de programación orientado a objetos.	Es un lenguaje basado en prototipos.
Tiene una función conocida como escritura estática	No admite esta función.
Admite interfaces	JavaScript no admite interfaces.

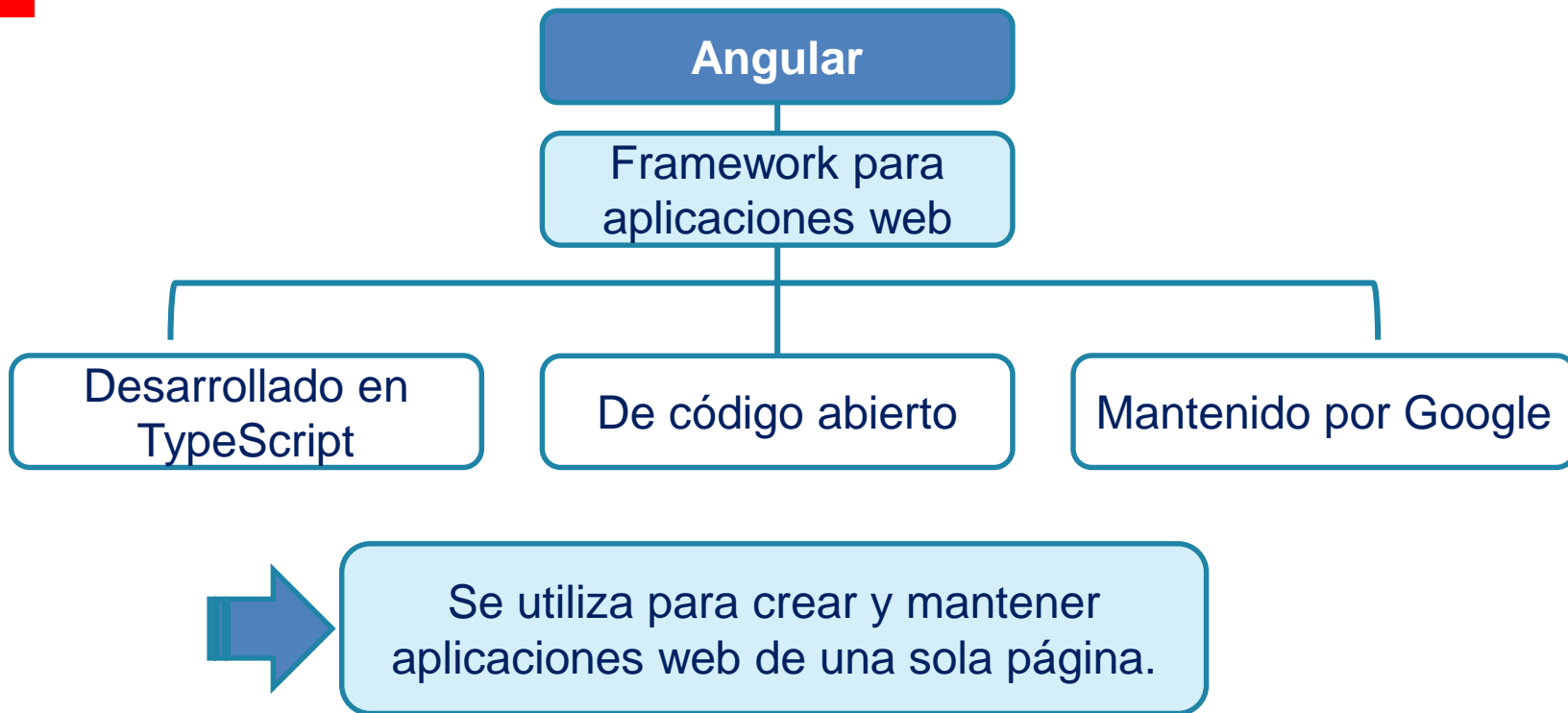
# TypeScript vs. JavaScript

Observa el siguiente cuadro comparativo con las diferencias:

TypeScript	JavaScript
Es un superconjunto de Javascript.	Es un lenguaje de secuencias de comandos que le ayuda a crear páginas web interactivas.
El código TypeScript debe compilarse.	El código Javascript no necesita compilarse.
Admite una función de creación de prototipos	No admite esta función.

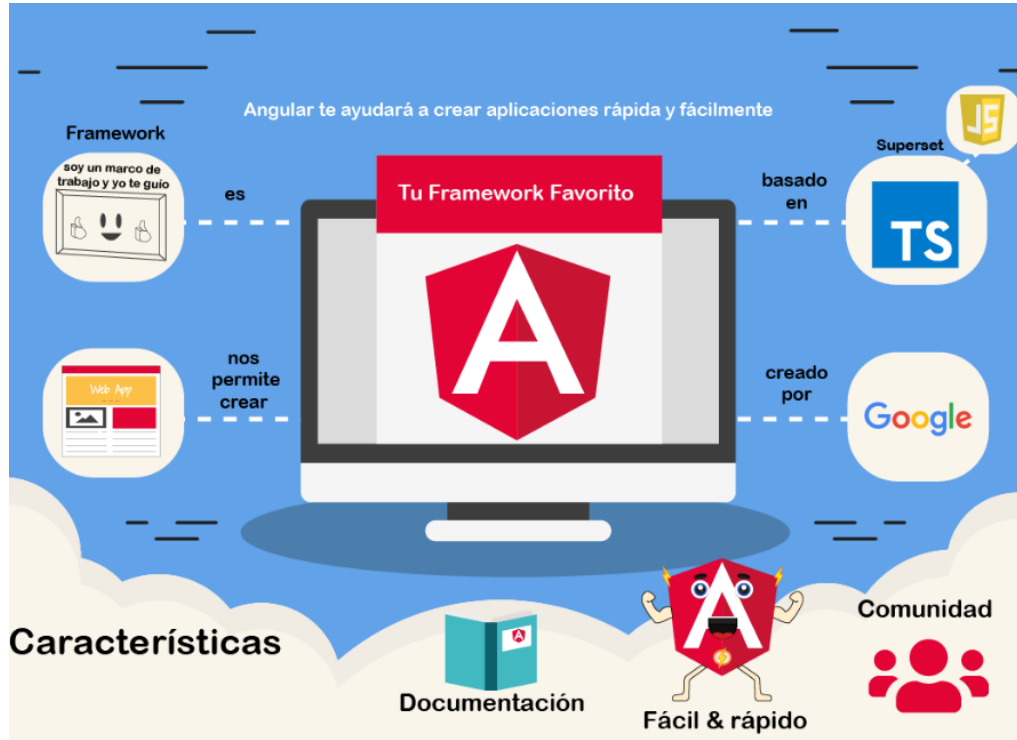


# ¿Qué es Angular?



# ¿Qué es Angular?

Observa la siguiente imagen para que puedas comprender mejor qué es Angular.



Recuperado de <https://medium.com>

# Instalación de Angular

## PREREQUISITOS



JavaScript



HTML



Css



TypeScript (es útil,  
pero no obligatorio).

## REQUISITOS



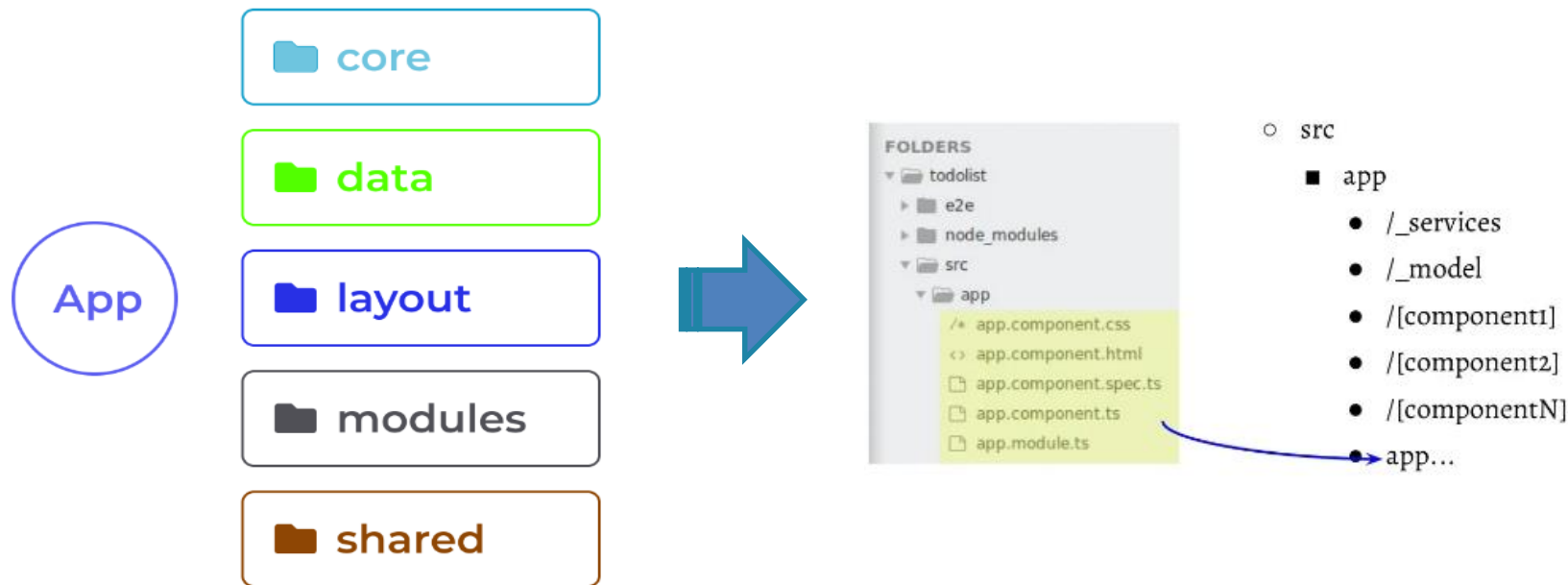
Node Js



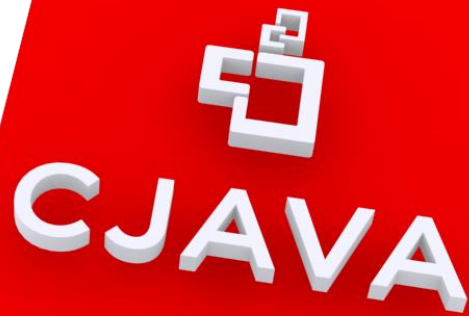
Npm Package  
Manager

# Estructura de un proyecto en Angular

Observa la estructura de un proyecto en Angular:



Recuperado de <https://designicode.com>



# Arquitectura de aplicaciones Angular

Edwin Maravi

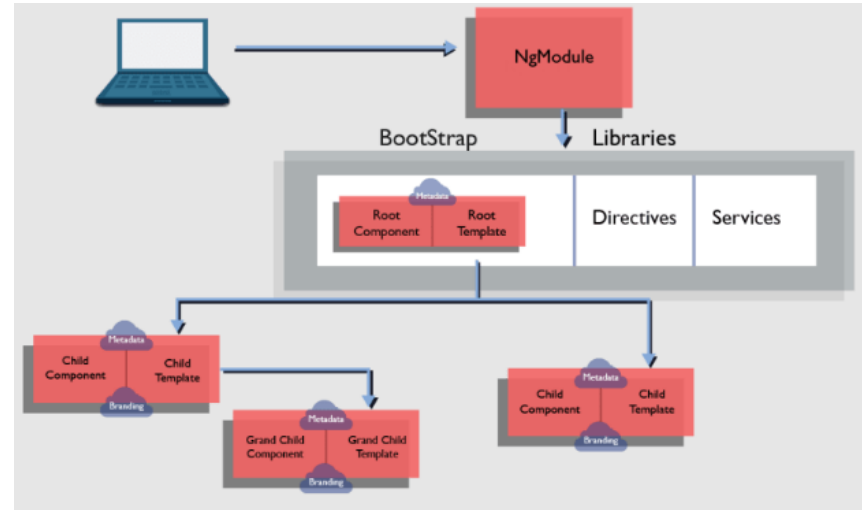
# Contenidos o temas

- Arquitectura Angular
- Arquitectura de un SPA ( Single Page Application)
- SPA vs. App Tradicional
- Ciclo de vida



# Arquitectura Angular

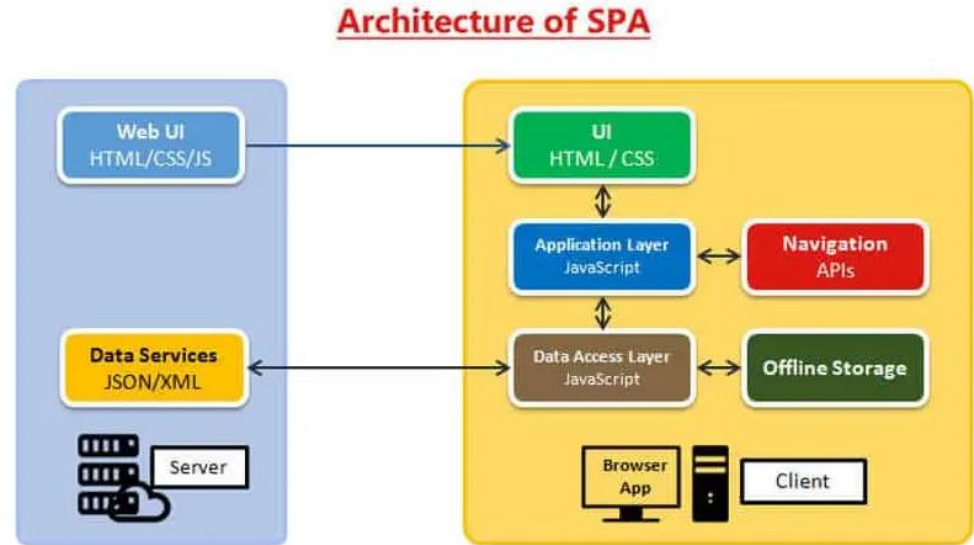
- Angular es un framework para construir aplicaciones web Front-End.
- Los bloques de construcción básicos del marco angular son componentes angulares que están organizados en NgModules.



Recuperado de <https://dev.to/vanessamarely/arquitectura-en-angular-5c23>

# Arquitectura SPA ( Single Page Application)

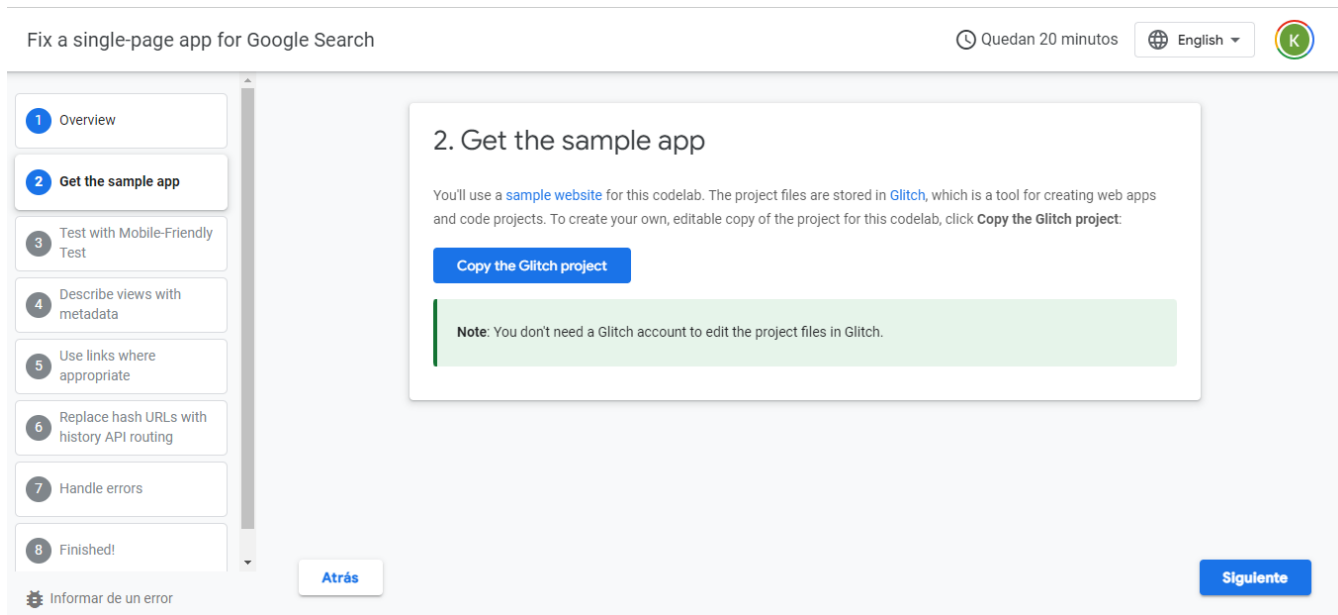
Es una aplicación web o es un sitio web que cabe en una sola página con el propósito de dar una experiencia más fluida a los usuarios, como si fuera una aplicación de escritorio.



Recuperado de <https://gearheart.io/articles/pros-and-cons-building-single-page-applications-2019/>

# Arquitectura SPA ( Single Page Application)

Observa el siguiente ejemplo:

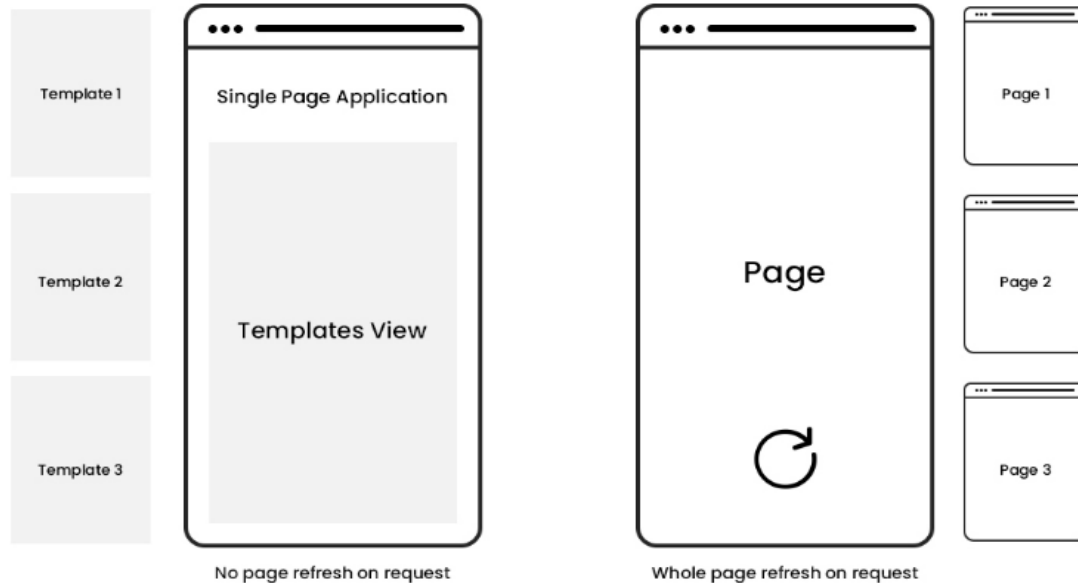


The screenshot shows a Codelabs interface for a tutorial titled "Fix a single-page app for Google Search". The interface includes a sidebar with a list of steps: 1. Overview, 2. Get the sample app (highlighted), 3. Test with Mobile-Friendly Test, 4. Describe views with metadata, 5. Use links where appropriate, 6. Replace hash URLs with history API routing, 7. Handle errors, and 8. Finished!. The main content area for step 2, "Get the sample app", contains the text: "You'll use a [sample website](#) for this codelab. The project files are stored in [Glitch](#), which is a tool for creating web apps and code projects. To create your own, editable copy of the project for this codelab, click **Copy the Glitch project**". Below this text is a blue button labeled "Copy the Glitch project". A green note box states: "Note: You don't need a Glitch account to edit the project files in Glitch." At the bottom of the main content area, there are two buttons: "Atrás" (Back) and "Sigiente" (Next). The top right of the interface shows a timer "Quedan 20 minutos", a language dropdown set to "English", and a user profile icon.

Recuperado de <https://codelabs.developers.google.com/codelabs/making-a-single-page-app-search-friendly#1>

# Arquitectura SPA ( Single Page Application)

Observa la estructura de una página single page application (SPA) y una estructura web tradicional:

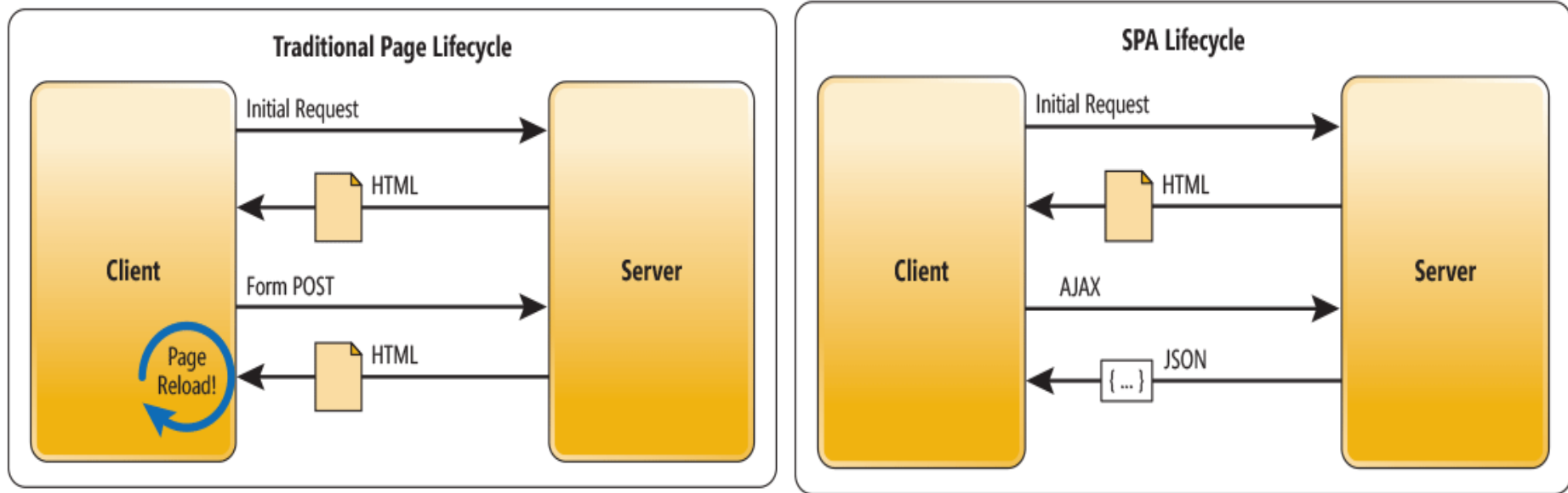


# SPA vs. App Tradicional

<b>Aplicaciones web tradicionales</b>	<b>Single Page Application</b>
<ul style="list-style-type: none"><li>▪ Los requisitos del lado del cliente de su aplicación son básicos o solo de lectura.</li></ul>	<ul style="list-style-type: none"><li>▪ Su aplicación debe mostrar una interfaz de usuario dinámica con muchas características.</li></ul>
<ul style="list-style-type: none"><li>▪ Su aplicación debe funcionar en navegadores sin compatibilidad con JavaScript.</li></ul>	<ul style="list-style-type: none"><li>▪ La aplicación debe funcionar en navegadores que soporten JavaScript.</li></ul>
<ul style="list-style-type: none"><li>▪ El equipo de desarrollo no está familiarizado con las técnicas de desarrollo de JavaScript o TypeScript.</li></ul>	<ul style="list-style-type: none"><li>▪ El equipo de desarrollador cuenta con experiencia con el desarrollo de JavaScript o TypeScript.</li></ul>

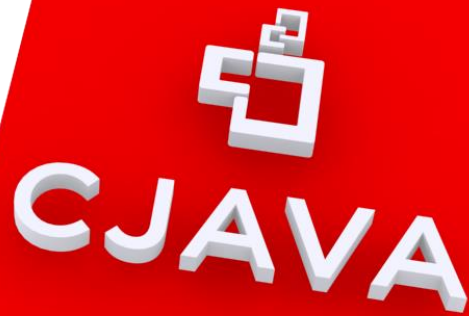
# Ciclo de vida

Observa el ciclo de vida de una página tradicional y de una SPA:



Recuperado de [https://juanda.gitbooks.io/webapps/content/spa/arquitectura\\_de\\_un\\_spa.html](https://juanda.gitbooks.io/webapps/content/spa/arquitectura_de_un_spa.html)





# Componentes Angular

Edwin Maravi

# Resultado de aprendizaje

Comprende la importancia del uso de componentes en una aplicación web. Asimismo, crea un nuevo componente usando la herramienta CLI de Angular.

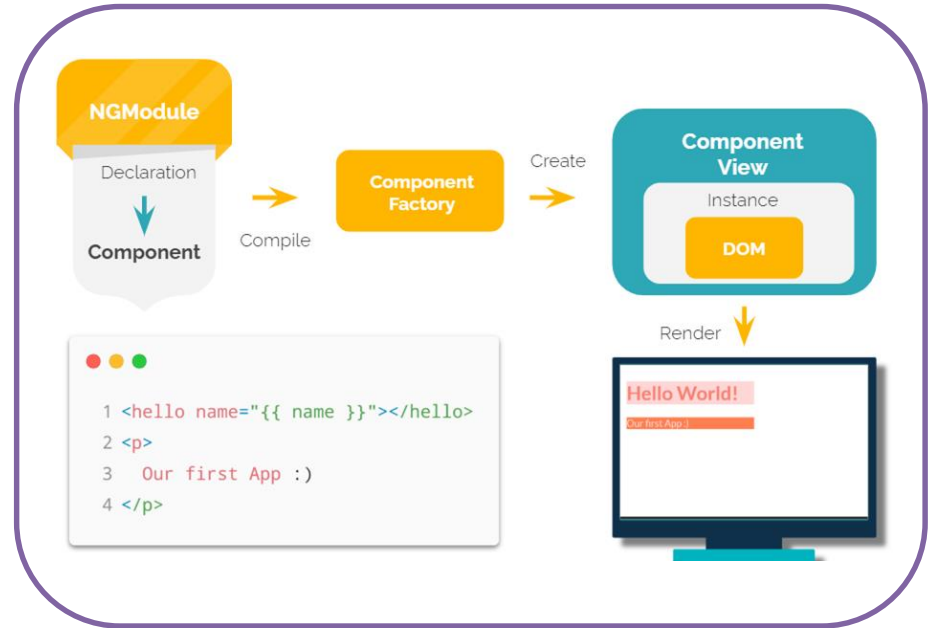


# Contenidos o temas

- ¿Qué es un componente angular?
- Ciclo de vida de los componentes angular
- Creación de un componente con Angular-CLI

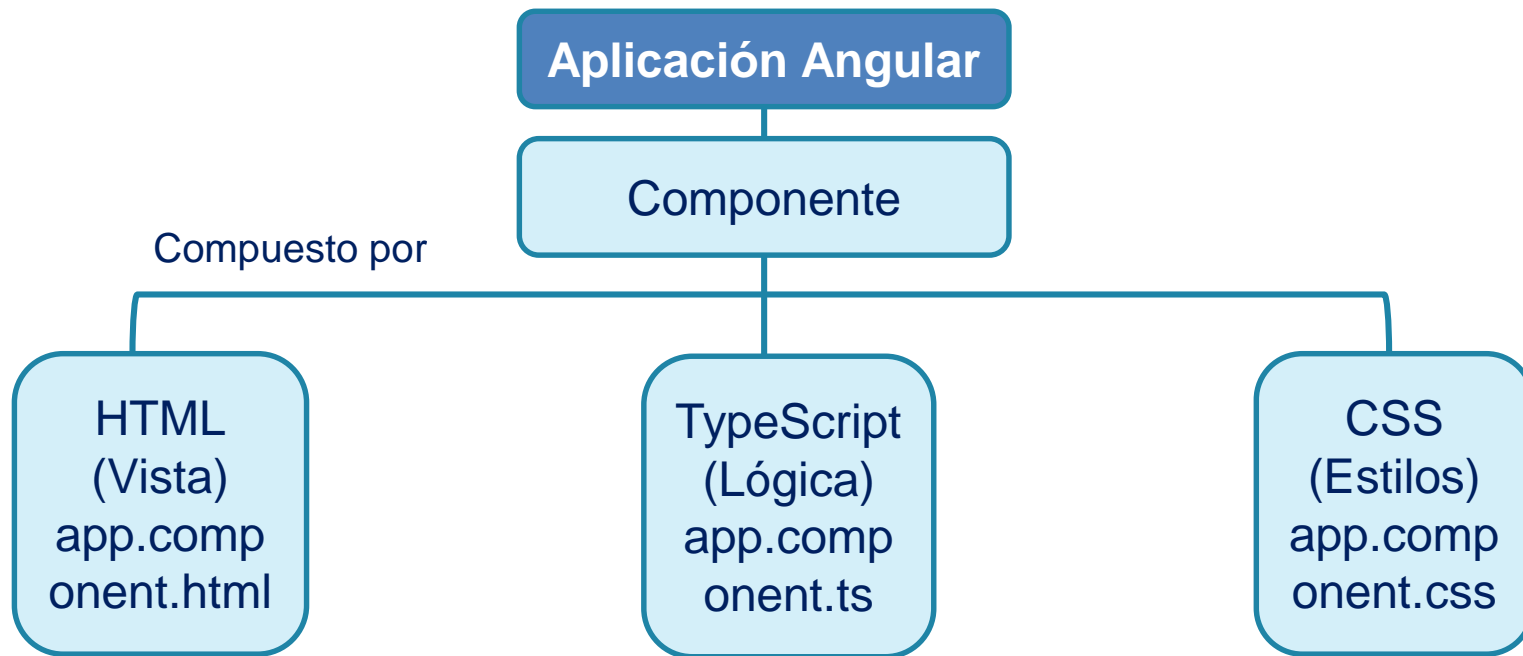
# ¿Qué es un componente angular?

- Son el bloque de creación de interfaz básico de una aplicación Angular.
- Son un subconjunto de directivas, siempre asociadas a un template o plantilla.
- Es un elemento reutilizable, puede ser desde un elemento HTML como un header, section, footer, etc.



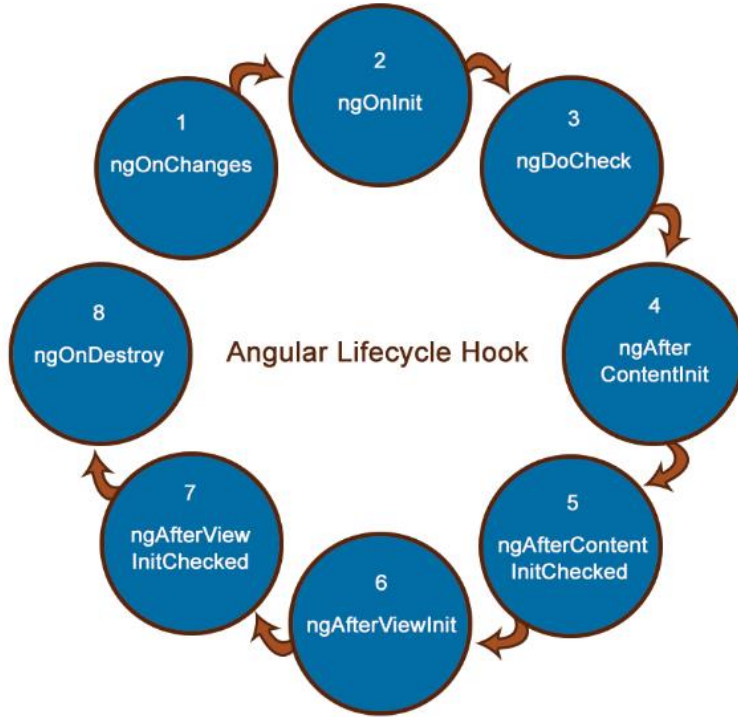
Recuperado de  
<https://medium.com/notasdeangular/componentes-din%C3%A1micos-97595f3f4092>

# ¿Qué es un componente angular?



# Ciclo de vida de los componentes Angular

Observa los métodos del ciclo de vida de Angular:



En Angular, los componentes son los principales pilares de la aplicación, es importante entender el ciclo de vida que estos tienen y la forma en que se ejecuta ese ciclo de vida, para así poder manipular los componentes en nuestra aplicación.

Recuperado de: <https://javadesde0.com/como-crear-un-componente-en-angular/>



# Crear un componente en Angular CLI

Existen dos formas de realizar la creación de componentes:

## 1. Manual

El proceso es más lento pero lo realizaremos con el fin de entender al 100% la estructura y el funcionamiento interno de un componente.

## 2. Mediante a un comando proporcionado por @angular/cli

Es lo recomendable ya que nos ahorra tiempo y trabajo y, además, nos otorga una mayor fiabilidad.

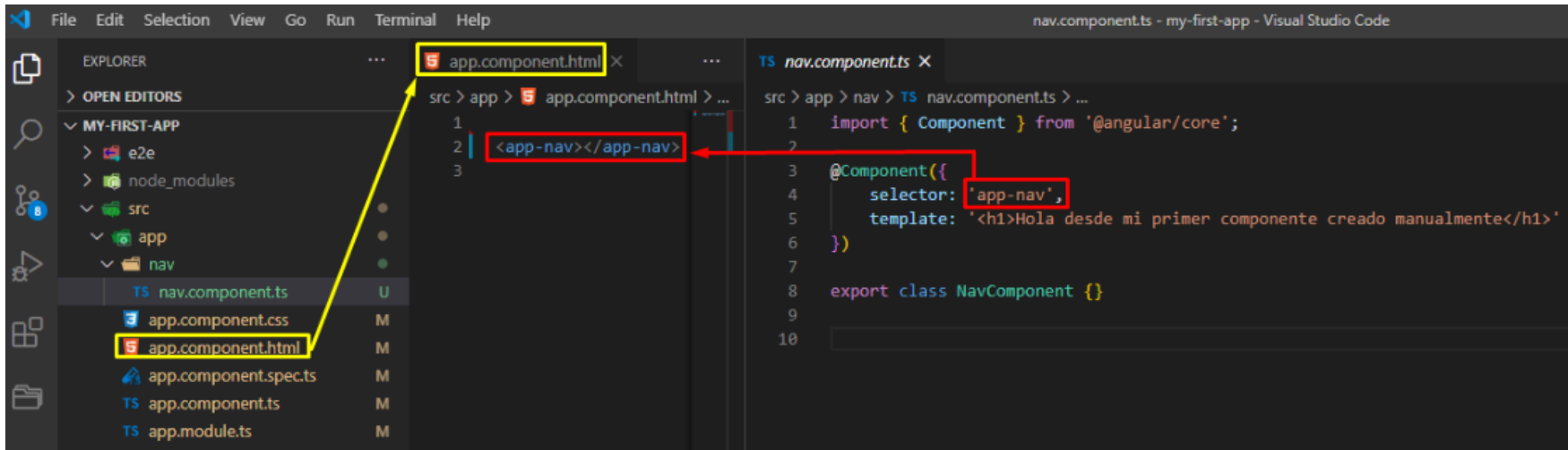
**ng generate component  
"nombre"**



**ng g c "nombre"**

# Crear un componente en Angular CLI

Observa cómo Angular genera la estructura HTML del componente (app.component.ts):



The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER:** The file tree on the left shows the project structure. The file `app.component.html` is highlighted with a yellow box.
- EDITOR:** The main editor area shows the content of `app.component.html`. The code is:

```
1 <app-nav></app-nav>
```

The code is highlighted with a red box.
- TERMINAL:** The terminal at the bottom shows the command `ng generate component nav` and the output `src > app > nav > TS nav.component.ts > ...`.
- FILE EXPLORER:** The file tree on the right shows the generated files for the `nav` component. The file `nav.component.html` is highlighted with a yellow box.
- EDITOR:** The main editor area shows the content of `nav.component.html`. The code is:

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-nav',
5   template: '<h1>Hola desde mi primer componente creado manualmente</h1>'
6 })
7
8 export class NavComponent {}
9
10
```

The code is highlighted with a red box.

Recuperado de <https://javadesde0.com/como-crear-un-componente-en-angular/>



# Módulos en Angular

Edwin Maravi

# Resultado de aprendizaje

Crea e implementa un módulo en angular mediante la clase decorada @NgModule.

# Contenidos o temas

- Módulo principal (Root Module)
- NgModule
- Creación de un módulo en Angular CLI

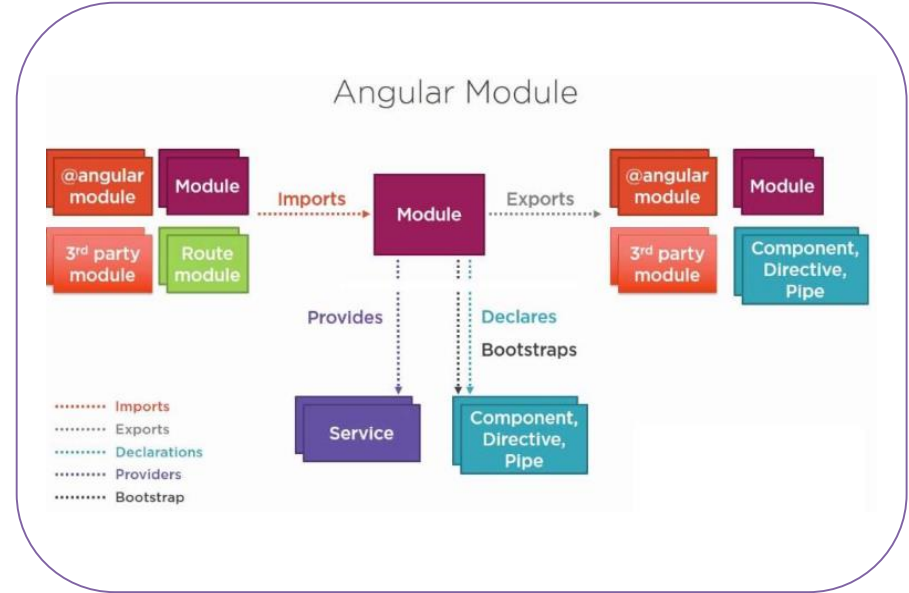
# Modulo Principal ( Root Module)

Aplicaciones desarrolladas  
en Angular

tienen

**Módulo raíz**

Que se compila para  
iniciar la aplicación.



Recuperado de: <https://w3path.com/angular-module-a-complete-guide-for-beginner/>

# Modulo Principal ( Root Module)

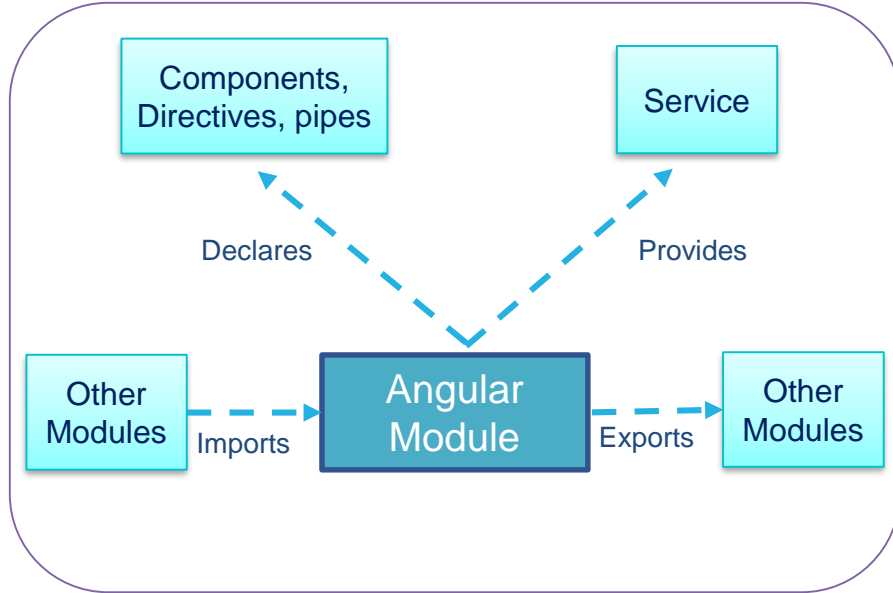
## Módulo raíz

Es todo lo que necesita una aplicación para iniciarla.

A medida que crece la aplicación, refactoriza el módulo raíz en módulos de características que representan colecciones de funcionalidad relacionada.

Luego importa estos módulos al módulo raíz.

# NgModules



- Las aplicaciones Angular son modulares y tienen su propio sistema de modularidad llamado NgModules.
- NgModule es un decorador que se utiliza en `app.module.ts`. `@NgModule` toma un objeto de metadatos que indica a Angular cómo compilar y ejecutar código de módulo.
- Los módulos son una excelente manera de organizar una aplicación y ampliarla con capacidades de bibliotecas externas



# NgModules

Observa el cuadro con las propiedades más importantes de un NgModule:

Metadatos	Definición
declarations	Declara las vistas que pertenecen a tu módulo.
exports	Conjunto de declaraciones que deberían ser visibles y utilizables en las plantillas de componentes de otros NgModules.
imports	Otros módulos NgModules, cuyas clases exportadas son requeridas para templates de componentes de este módulo.

# NgModules

Observa el cuadro con las propiedades más importantes de un NgModule:

Metadatos	Definición
providers	Aquí se declaran todos los servicios que necesita el modulo, y que estarán disponibles para toda la aplicación.
bootstrap	Es la vista principal de la aplicación, llamado el componente raíz, que aloja todas las demás vistas de la aplicación. Sólo el NgModule raíz debe establecer la propiedad bootstrap.

# NgModules

En esta imagen se muestra la estructura de un módulo “app.module.ts”:

src/app/app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
@NgModule({
  imports:      [ BrowserModule ],
  providers:    [ Logger ],
  declarations: [ AppComponent ],
  exports:      [ AppComponent ],
  bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

Recuperado de: <https://docs.angular.lat/guide/architecture-modules>

# Creación de un módulo en Angular CLI

- Cuando creamos una aplicación con Angular CLI, se crea en forma automática el módulo 'AppModule' en el archivo 'app.module.ts'.
- También podemos crear módulos desde la línea de comandos a través de la herramienta de Angular CLI.

**ng generate module [nombre]**



**ng g m [nombre]**

# Creación de un módulo en Angular CLI

Observa cómo NgModule se crea en el archivo app. module.ts de forma predeterminada:

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

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

@NgModule({
  imports: [BrowserModule],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Recuperado de: <https://angular-training-guide.rangle.io/modules/introduction>



**CJAVA**

siempre para apoyarte

Gracias