

# Arquitecturas de Software para Aplicaciones Empresariales

## Introducción a aplicaciones SPA y Angular

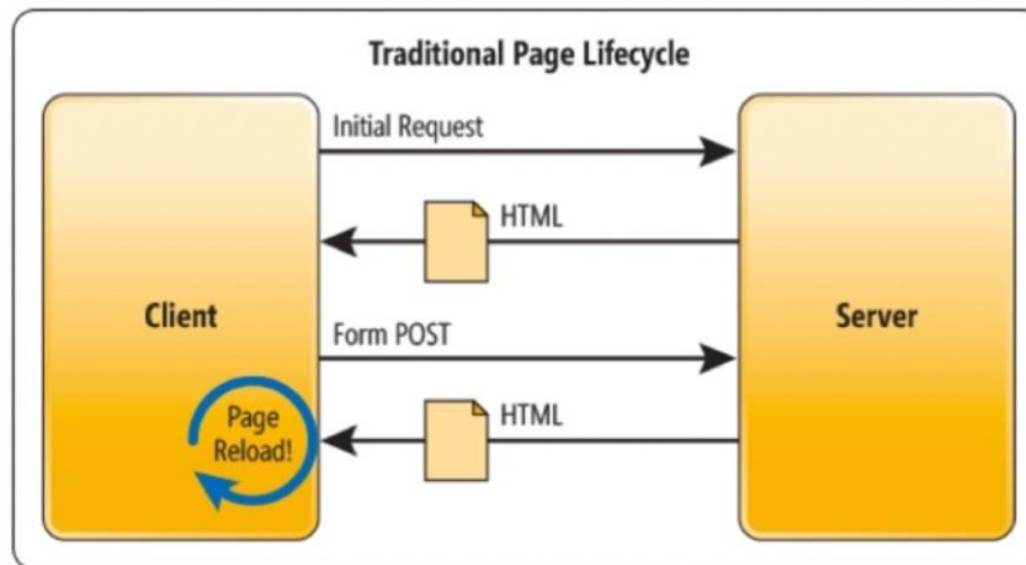


### PROGRAMA DE INGENIERIA DE SISTEMAS

Ing. Daniel Eduardo Paz Perafán ([danielp@Unicauca.edu.co](mailto:danielp@Unicauca.edu.co))

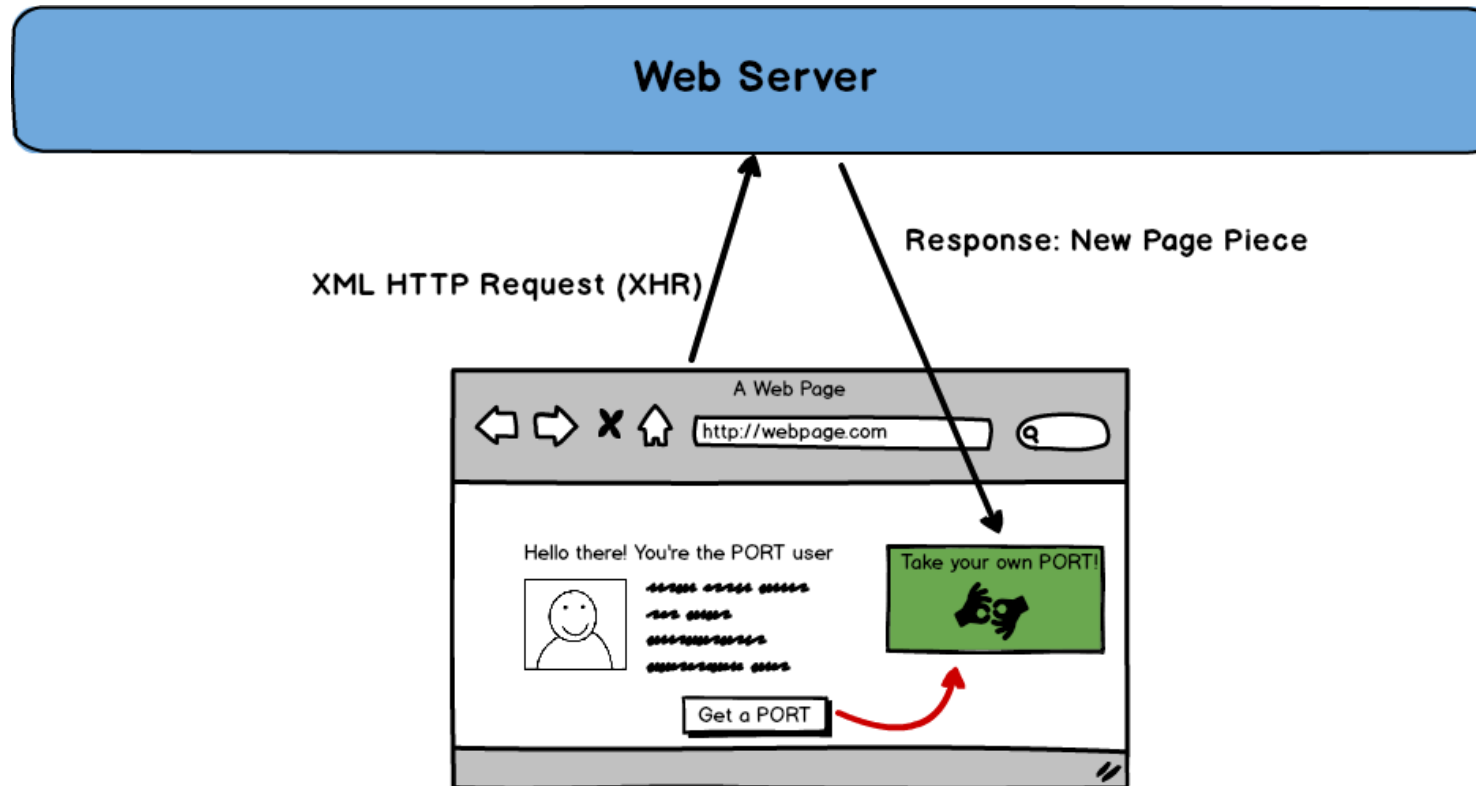
Ing. Pablo A. Magé ([pmage@Unicauca.edu.co](mailto:pmage@Unicauca.edu.co))

- ❖ EL navegador enviaba una petición (GET, POST) al servidor, la cual estaba asociada a una URL.
- ❖ El servidor seleccionaba una pagina o construía las páginas con datos y devolvía el HTML, CSS, y JavaScript al navegador. Esto genera una gran carga al servidor.
- ❖ EL navegador interpretaba el resultado.



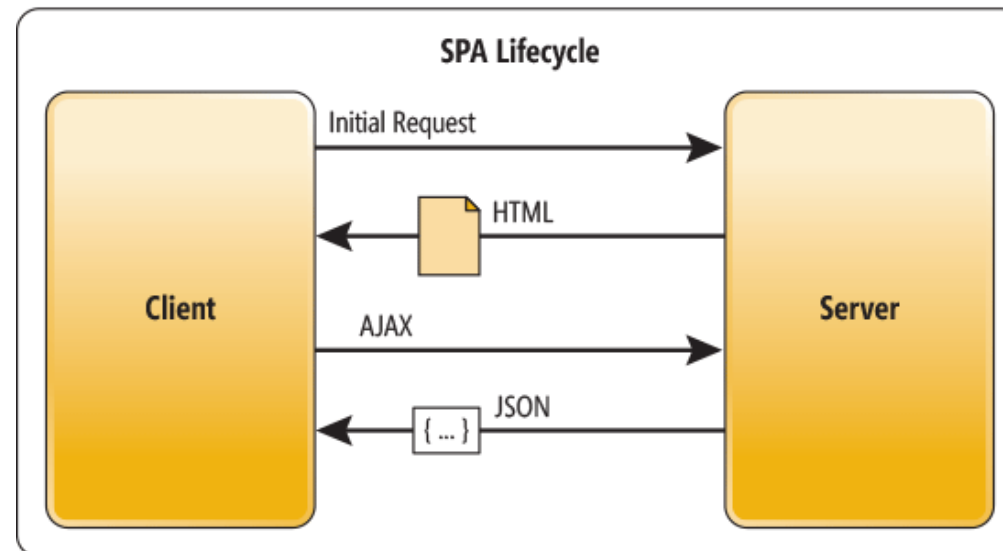
# Asynchronous JavaScript And XML (AJAX)

En una pagina fue posible realizar peticiones asíncronas al navegador a través de Javascript, con el fin de actualizar secciones específicas, repintar bloques con datos actualizados, modificar vistas, etc



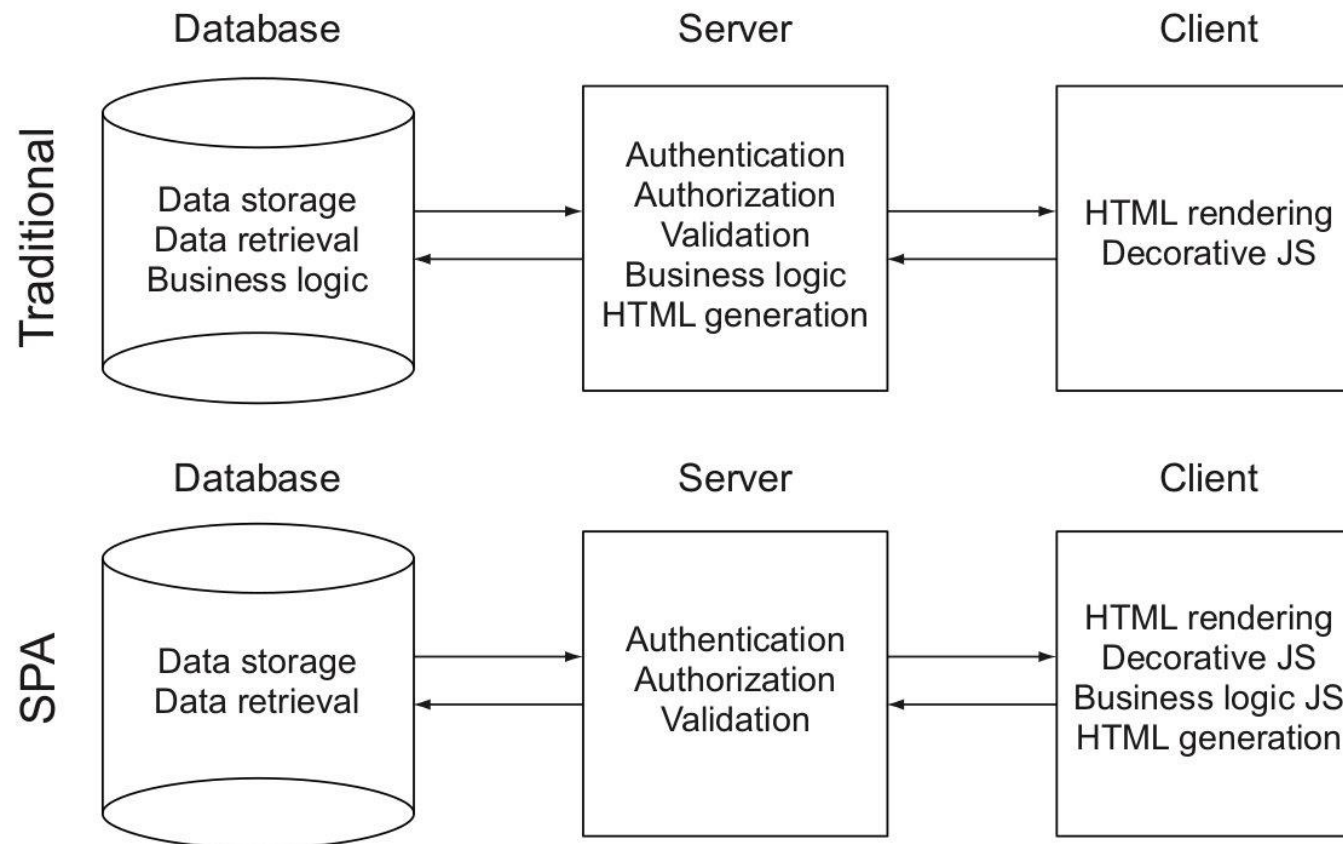
# Aplicaciones de una sola página (SPA)

- ❖ Por SPA (Single Page Application ) se conocen las aplicaciones de una sola página o Single Page Applications.
- ❖ Una SPA, es un sitio web que nunca recarga la pagina. Todo el contenido html, css, JavaScript esta en el navegador del cliente.
- ❖ El navegador hace peticiones al servidor en formato Json con el fin de realizar inicios de sesión, validaciones, consultar o gestionar datos del medio de persistencia



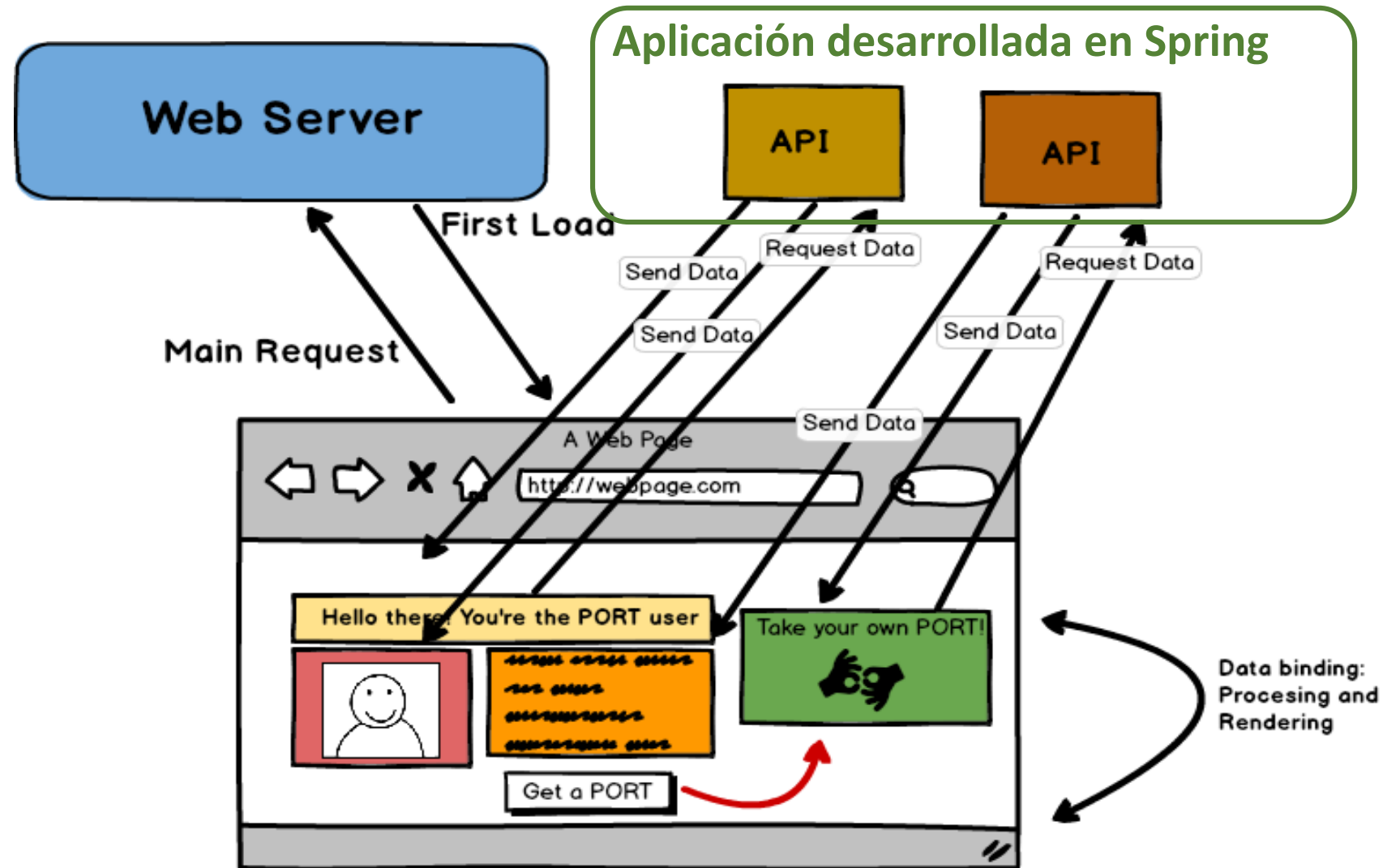
# Aplicaciones de una sola página (SPA)

El navegador hace peticiones al servidor en formato Json con el fin de realizar inicios de sesión, validaciones, consultar datos del medio de persistencia o gestionar los datos del medio de persistencia.



# Aplicaciones de una sola página (SPA)

UNIVERSIDAD DEL CAUCA – FIET  
DEPARTAMENTO DE SISTEMAS

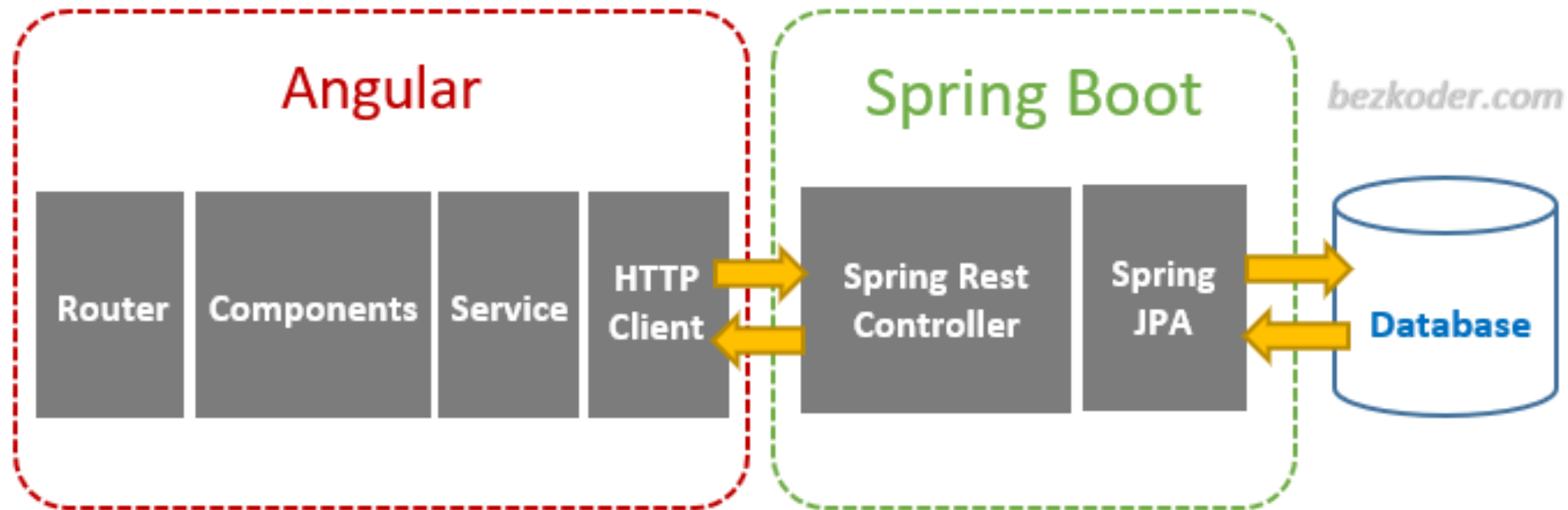


- ❖ Angular es un framework de desarrollo para JavaScript creado por Google que se utiliza para crear y mantener aplicaciones web de una sola página.
- ❖ Cada 6 meses se libera una versión
- ❖ Angular separa completamente el front-end y el back-end (es la capa de acceso a datos de un software o cualquier dispositivo, que no es directamente accesible por los usuarios)
- ❖ **AngularJS** (Angular 1.x) y **Angular** (Angular 2) son frameworks diferentes



- ❖ *Aplicaciones web responsivas*: aplicaciones que se adaptan según la resolución, a todo tipo de dispositivos pc, tablet, celulares etc. Utilizando herramientas como Bootstrap o angular material.
- ❖ *Single Page Application-SPA*: Es la esencia de Angular, van a existir muchos elementos que van cambiando dinámicamente y están en una sola página, no se recarga la página.
- ❖ *Reactivo RXJs*: RxJS is una librería de programación reactiva cuyo fin es simplificar la composición de código asíncrono y basado en eventos a través de observables.
- ❖ *Asíncrono*: Durante el intercambio de la información no se recarga la página, solo se intercambia información de forma asíncrona y reactiva.





- ❖ Con Angular se va a trabajar con el front end, todo lo relacionado con el cliente y con Spring se va a trabajar el back end, todo lo relacionado con la lógica del negocio
- ❖ En el back end se trabajarán con API Restful, Spring MVC, anotaciones, validaciones de datos, Spring Data JPA.
- ❖ Se va a utilizar el API Reactive X utilizando operadores para realizar la conexión con el back end.

- ❖ La programación en Angular se hace usando TypeScript, el cual es un lenguaje que es un superconjunto de JavaScript que agrega capacidades de tipado estático.
- ❖ Esto nos da la ventaja de poder tipar cosas como variables, funciones, devoluciones, además de poder crear clases , atributos , métodos, constructores e interfaces.
- ❖ TypeScript la cual fue desarrollada y es mantenida por Microsoft. Es un lenguaje de programación libre y es open source.
- ❖ El lenguaje utilizado es TypeScript pero una vez compilado se convierte en Javascript.

## Instalación Node.js

Sitio: <https://nodejs.org/en/>

a) Debemos instalarlo debido a que contiene un componente denominado **Node Package Manager-NPM**, utilizado para administrar y descargar dependencias.

b) Para instalarlo se recomienda que sea la versión más estable, en este caso 16.13.2LTS, donde LTS significa soporte a largo plazo.

c) Una vez descargado e instalado debemos abrir una consola para comprobar la versión del nodejs y la versión NPM.

d) Con el comando **node -v** se tiene la versión de nodejs

```
C:\Users\LENOVO>node -v  
v14.15.0
```

e) Con el comando **npm -v** se tiene la versión de NPM.

```
C:\Users\LENOVO>npm -v  
6.14.8
```



## Instalación de TypeScript

Sitio: <https://www.typescriptlang.org>

Nos ayuda a detectar errores en tiempo de escritura, este es el lenguaje con el cual se programa en Angular, desde la versión 2 en adelante.

a) Para instalar se abre una ventana de comando con cmd o powershell como administrador y se ingresa el comando `npm install -g typescript`

La opción -g significa que se va a instalar de manera global

```
C:\WINDOWS\system32>npm install -g typescript
C:\Users\LENOVO\AppData\Roaming\npm\tsc -> C:\Users\LENOVO\AppData\Roaming\npm\node_modules\typescript\bin\tsc
C:\Users\LENOVO\AppData\Roaming\npm\tsserver -> C:\Users\LENOVO\AppData\Roaming\npm\node_modules\typescript\bin\tsserver+ typescript@4.0.5
added 1 package from 1 contributor in 46.072s
```

b) Para probar utilizar el comando `tsc -v`

```
C:\WINDOWS\system32>tsc -v
Version 4.0.5
```

Sitio: <https://angular.io>

Como pre requisitos se necesitan node.js y el gestor de paquetes npm.

a) Para instalar se abre una ventana de comando con cmd o powershell como administrador y se ingresa el comando `npm install -g @angular/cli`

```
C:\WINDOWS\system32>npm install -g @angular/cli
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
C:\Users\LENOVO\AppData\Roaming\npm\ng -> C:\Users\LENOVO\AppData\Roaming\npm\node_modules\@angular\cli\bin\ng

> @angular/cli@10.2.0 postinstall C:\Users\LENOVO\AppData\Roaming\npm\node_modules\@angular\cli
> node ./bin/postinstall/script.js

? Would you like to share anonymous usage data with the Angular Team at Google under
Google's Privacy Policy at https://policies.google.com/privacy? For more details and
how to change this setting, see http://angular.io/analytics. Yes

Thank you for sharing anonymous usage data. If you change your mind, the following
command will disable this feature entirely:


  ng analytics off

+ @angular/cli@10.2.0
added 279 packages from 206 contributors in 1053.747s
```

b) Para probar utilizar el comando `ng version`

Se desplegarán los datos del Angular CLI (command line interface), junto con los paquetes de desarrollo específicos.

```
C:\WINDOWS\system32>ng version
```



```
Angular CLI: 10.2.0  
Node: 14.15.0  
OS: win32 x64  
  
Angular:  
...  
Ivy Workspace:
```

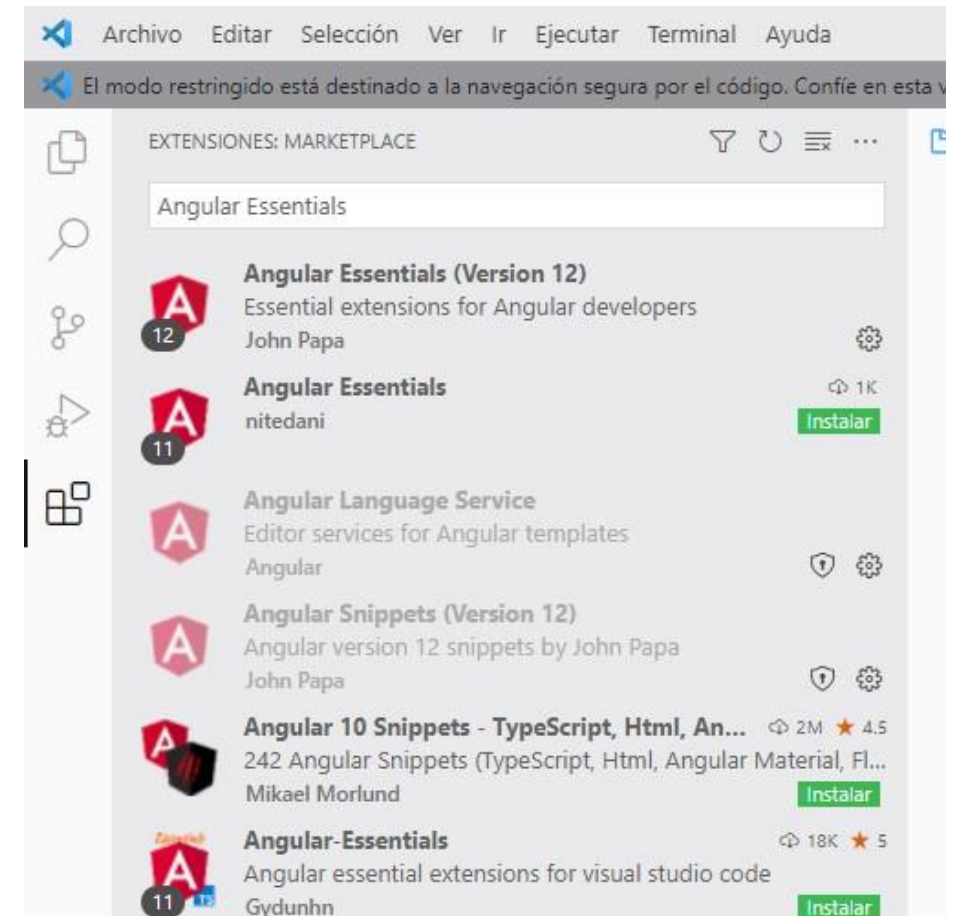
Package	Version
@angular-devkit/architect	0.1002.0 (cli-only)
@angular-devkit/core	10.2.0 (cli-only)
@angular-devkit/schematics	10.2.0 (cli-only)
@schematics/angular	10.2.0 (cli-only)
@schematics/update	0.1002.0 (cli-only)

# Configuración de VSCode para desarrollar con Angular

UNIVERSIDAD DEL CAUCA – FIET  
DEPARTAMENTO DE SISTEMAS

Los siguientes son las extensiones que se recomienda instalar en el editor VSCode para desarrollar con Angular

- Angular Essentials
  - Angular Snippets: Adiciona snippets para Angular TypeScript y HTML.
  - Angular Language Service: Enriquece la experiencia de edición para Templates Angular.
  - Debugger from Chrome: Para depurar el código JavaScript ejecutando en Google Chrome.
  - Material Icon Theme: Obtener diseño de iconos para VSCode.



Para crear, construir y ejecutar en un servidor de desarrollo un nuevo proyecto básico de Angular, debe ir al directorio principal de su nuevo espacio de trabajo y use los siguientes comandos:

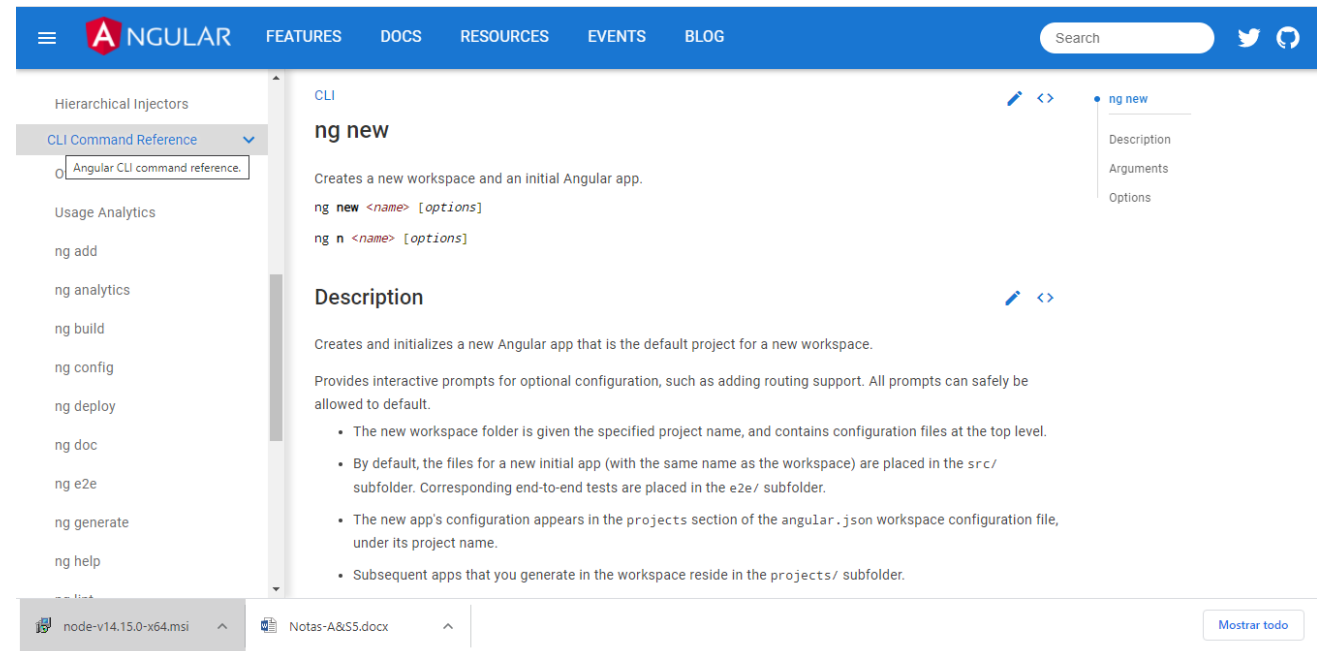
```
ng new miPrimerProyecto
```

```
cd miPrimerProyecto
```

```
ng serve
```

Para obtener información de todos los comandos del Angular CLI, ir al sitio de la herramienta y buscar el enlace CLI COMMANDS.

<https://angular.io>



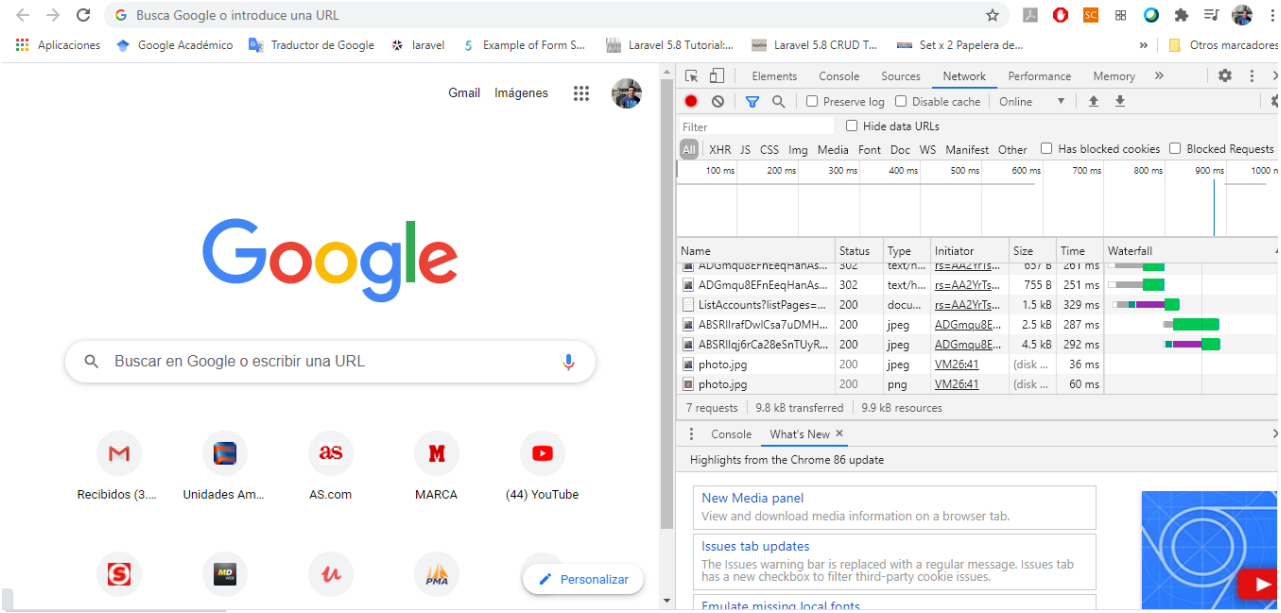
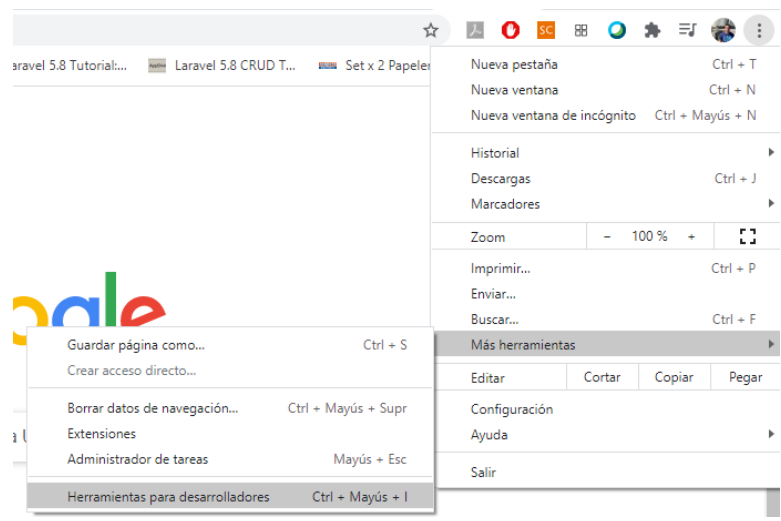


# Construir un proyecto

## Navegador Chrome

Se recomienda que se instale el navegador Chrome ya que posee una herramienta para detectar errores. Para activarla en el navegador ubicarse en la esquina superior derecha en el icono (Personalizar y configurar Google Chrome). Click en More tools y seleccionar **Developers Tools**.

Se encontrarán diferentes herramientas, por ejemplo **Console** para visualizar todos los errores, **Elements** para inspeccionar elementos de una página.



- a) Crear un directorio de trabajo donde se almacenarán los proyectos Angular a crear.
- b) Abrir una ventana de comandos y ubicarse en el directorio **de trabajo**, e ingresar el siguiente comando:

**ng new clientes-app**

- c) Se despliega ¿Would you like to add Angular Routing? Se responde **N**, por que más adelante se hará la instalación de forma manual

```
C:\Users\LENOVO\Documents\workspace-spring-tool-suite-4-4.7.1.RELEASE\angular>ng new clientes-app
? Would you like to add Angular routing? No
? Which stylesheet format would you like to use? CSS
```

- d) Se despliega ¿Which stylesheet format would you like to use? Seleccionar **CSS**

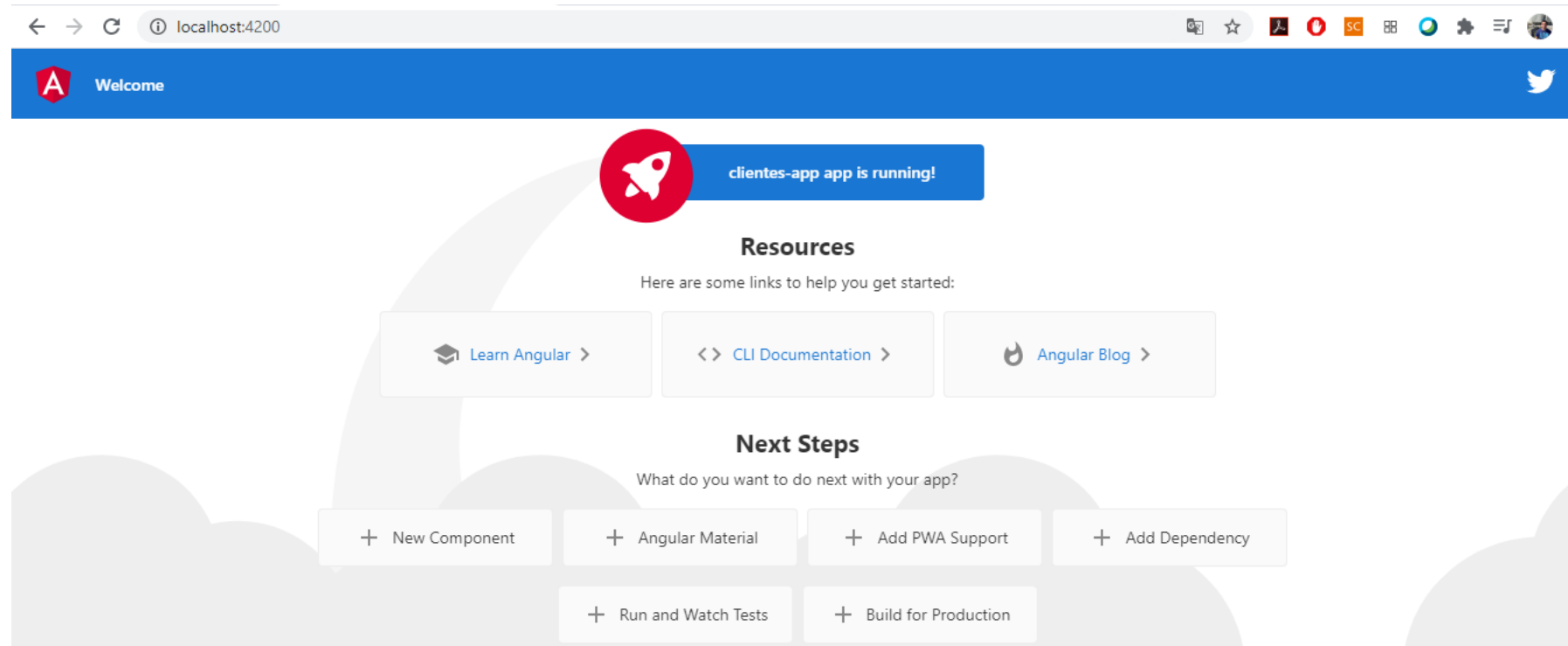
```
CREATE clientes-app/e2e/src/app.po.ts (301 bytes)
/ Installing packages...
```

# Ejecutar un proyecto en el servidor

UNIVERSIDAD DEL CAUCA – FIET  
DEPARTAMENTO DE SISTEMAS

Para ejecutar el proyecto en un servidor de desarrollo se utiliza el comando **ng serve --open** para abrir el proyecto en el navegador por defecto.

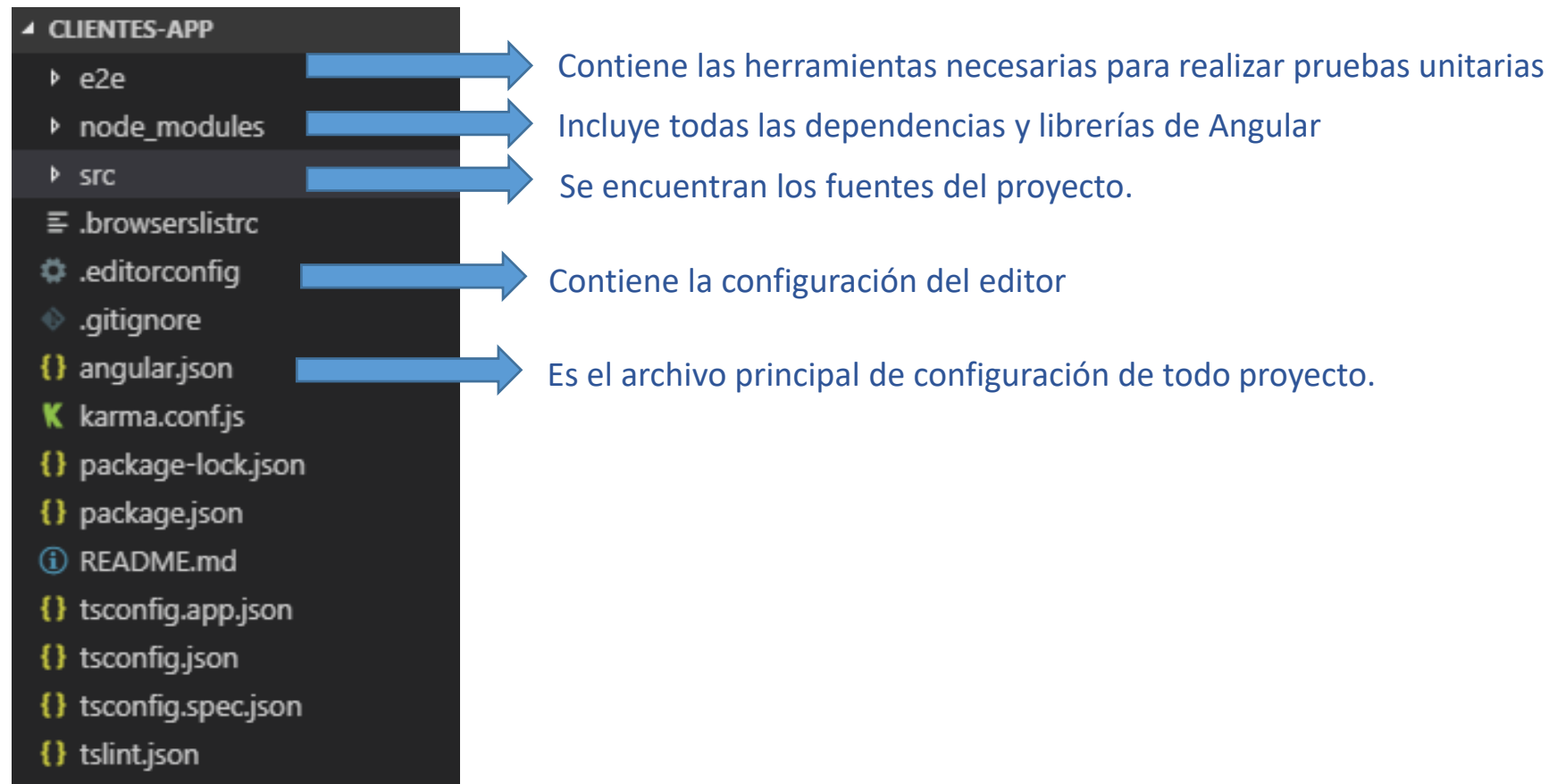
Si deseamos lanzar el servidor en un puerto en particular debemos colocar el comando **ng serve -o --port 4444**



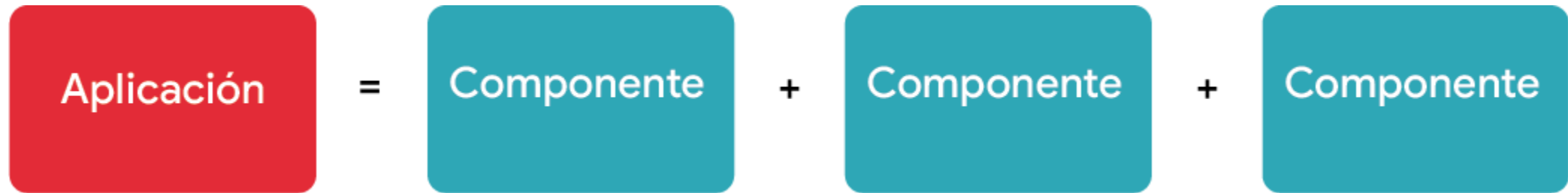
# Estructura de un proyecto en angular

En la carpeta **src**, se encuentran los fuentes del proyecto.

En la carpeta **src/app**, se encuentra el componente principal. Allí encontrará:



Angular 2 se desarrolla en base a componentes.



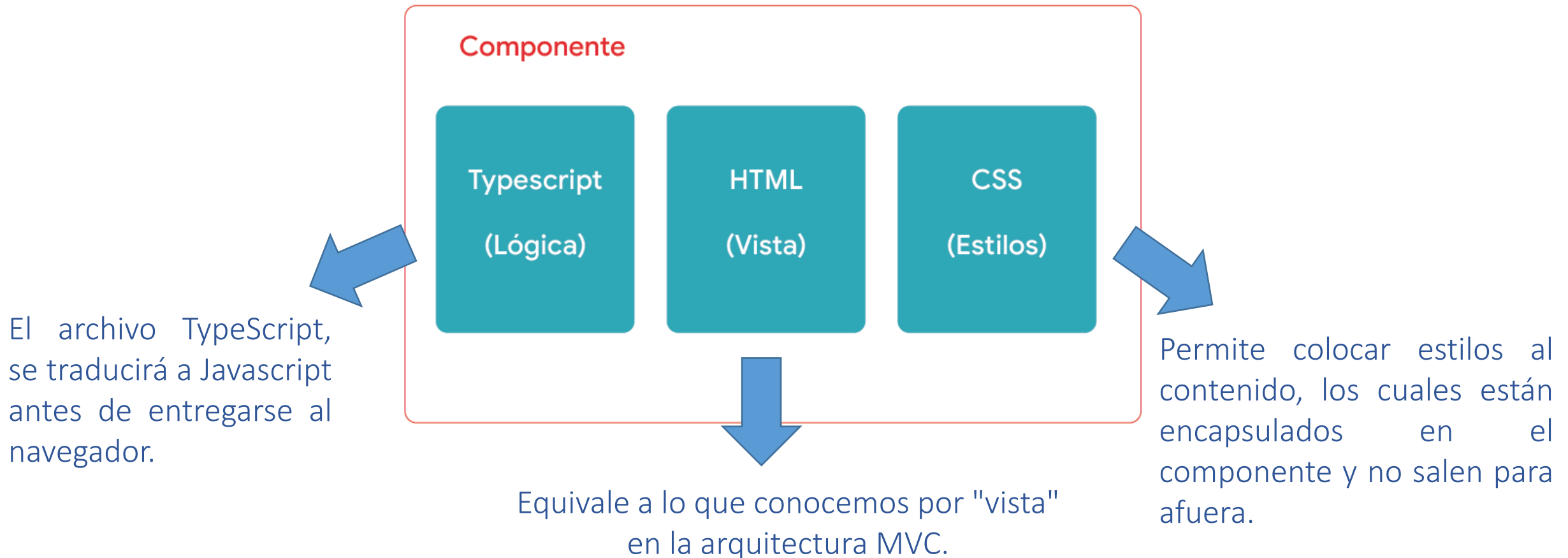
Un componente en Angular es un bloque de código **re-utilizable**.

Los componentes pueden ser:

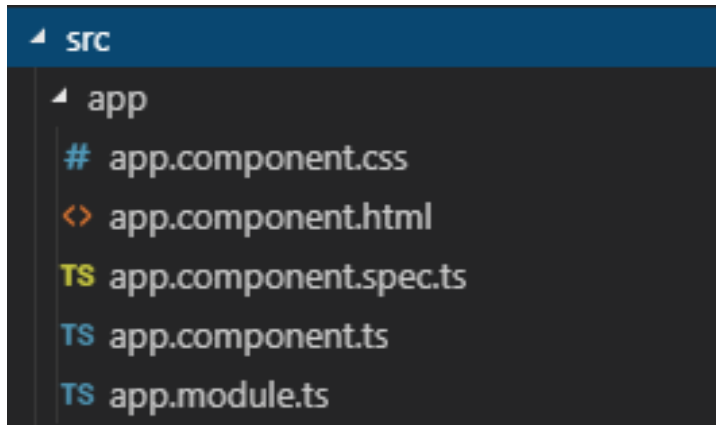
- Un elemento HTML como un `<button>`, una lista (`<ul><li>`), un `<header>`, un `<section>`.
- Un conjunto de etiquetas o elementos `<div>` que tengan una funcionalidad en particular.
- Menú de navegación, un banner, tabla de datos o gráficos

Los componentes son asincrónicos , es decir se ejecutan en su propio proceso

Un componente en Angular consta básicamente de 3 archivos: un CSS, un HTML (también conocido como plantilla o en inglés, *template*) y un TypeScript (en adelante, TS).



En la carpeta app encontramos los archivos que constituyen un solo componente



## app.components.css

Es la hoja de estilo específico del componente, si se desea agregar estilos globales se debe utilizar el archivo `styles.css`

## app.component.html

Corresponde a la vista del componente,

## app.component.spec.ts

Es un archivo para realizar pruebas unitarias

## app.component.ts

Permite mostrar el contenido dinámico del componente

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'clientes-app';
}
```

Archivo `app.component.ts`

El calificador `export` se utiliza para que el componente se pueda exportar y registrar como componente en `app.module.ts`.

## @Component

Decorador que marca una clase como un componente angular y proporciona metadatos de configuración que determinan cómo se debe procesar, instanciar y usar el componente en tiempo de ejecución.

## styleUrls

Una o más rutas relativas o URL absolutas para archivos que contienen hojas de estilo CSS para usar en este componente.

## templateurl

La ruta relativa o la URL absoluta de una plantilla html para un componente angular.

## selector

Indica a Angular que debe crear e instanciar el componente cuando se encuentra un elemento con ese nombre en el HTML



```
export class AppComponent{
```

El calificador export se utiliza para que el componente se pueda exportar y registrar como componente en `app.module.ts`.

El archivo `app.module.ts`: Es una especie de repositorio donde se registran los componentes.

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'clientes-app';
}
```

Archivo `app.component.ts`

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

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

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

Archivo `app.module.ts`

# Elementos de un componente

Archivo **index.html**: donde se refleja el selector del componente principal, a manera de etiquetas:

Index.html

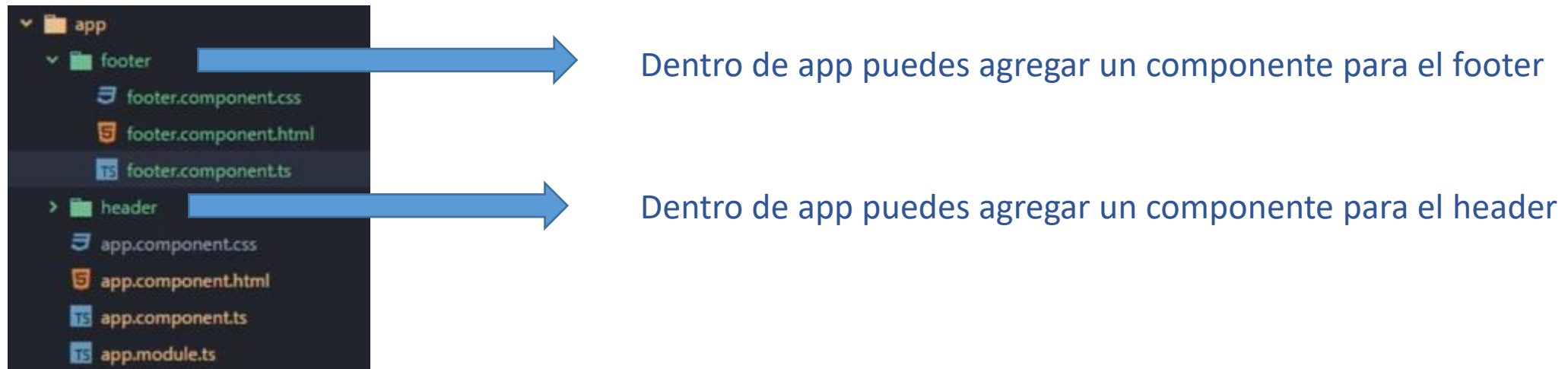
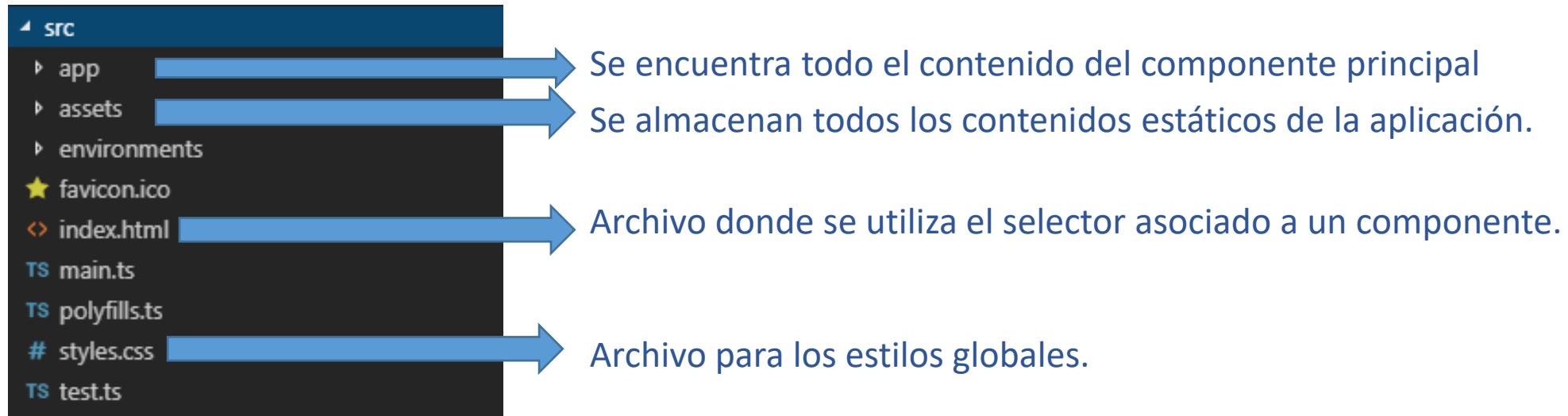
```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>ClientesApp </title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

app.component.ts

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'clientes-app';
}
```

# Estructura de la carpeta src



Como crear un componente de forma automática:

- a) Desde un terminal ubicarse en la carpeta src/app
- b) Generar la clase FooterComponent mediante el comando:
  - ng generate component footer
  - Forma resumida: ng g c footer.
- c) Actualizar la clase footer.component.ts
- d) Incluir en footer.component.html un contenido
- e) Agregar un estilo para el componente footer: Actualizar footer.component.css
- f) Utilizar el componente footer en app.component.html

# Creación de componentes footer y header

```
PS D:\aplicaciones desarrolladas II-2022\clientes2-app> cd .\src\app\  
PS D:\aplicaciones desarrolladas II-2022\clientes2-app\src\app> ng g c footer  
CREATE src/app/footer/footer.component.html (21 bytes)  
CREATE src/app/footer/footer.component.spec.ts (599 bytes)  
CREATE src/app/footer/footer.component.ts (202 bytes)  
CREATE src/app/footer/footer.component.css (0 bytes)  
UPDATE src/app/app.module.ts (396 bytes)  
PS D:\aplicaciones desarrolladas II-2022\clientes2-app\src\app> ng g c header  
CREATE src/app/header/header.component.html (21 bytes)  
CREATE src/app/header/header.component.spec.ts (599 bytes)  
CREATE src/app/header/header.component.ts (202 bytes)  
CREATE src/app/header/header.component.css (0 bytes)  
UPDATE src/app/app.module.ts (478 bytes)  
PS D:\aplicaciones desarrolladas II-2022\clientes2-app\src\app> █
```

# Creación de componentes footer y header

En el archivo app.module.ts se encuentran referenciados los componentes creados

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

import { AppComponent } from './app.component';
import { FooterComponent } from './footer/footer.component';
import { HeaderComponent } from './header/header.component';

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

Editamos el elemento lógico del componente agregando un atributo

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

@Component({
  selector: 'app-header',
  templateUrl: './header.component.html',
  styleUrls: ['./header.component.css']
})
export class HeaderComponent {
  title : string = 'Proyecto de Clase'
}
```

Editamos el elemento html del componente con un navbar

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">{{title}}</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbar"
      <span class="navbar-toggler-icon"></span>
    </button>
  </div>
</nav>
```



Editamos el elemento lógico del componente agregando 3 atributos

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

@Component({
  selector: 'app-footer',
  templateUrl: './footer.component.html',
  styleUrls: ['./footer.component.css']
})
export class FooterComponent {
  public proyecto: any = {anio: '2022', nombreProyecto: 'Proyecto de Clase'};
  public tecnologia: any = {leyenda: 'WebApp desarrollada con ', tec1: 'Angular ', tec2: 'Spring5'};
  public autor: string = 'ASAE';
}
```

Editamos el elemento html del componente

```
<footer class="footer bg-danger rounded top">
  <div class="container py-2">
    <p class="text-center text-white my-2">
      &copy; {{proyecto.anio+' '+proyecto.nombreProyecto+' | '+tecnologia.leyenda+tecnologia.tec1+
        ''+tecnologia.tec2+' | Autor: '+autor}}
    </p>
  </div>
</footer>
```

Editamos el archivo app.component.html invocando los componentes de la siguiente manera

```
<app-header> </app-header>
```

```
<div class="container">
```

```
</div>
```

```
<app-footer></app-footer>
```

Editamos el archivo index.html invocando el componente app.component

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>ClientesApp</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
  <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-Df.
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.bundle.
  <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js" :
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.min.js" i
</body>
</html>
```

**Muchas gracias**

**Preguntas**

