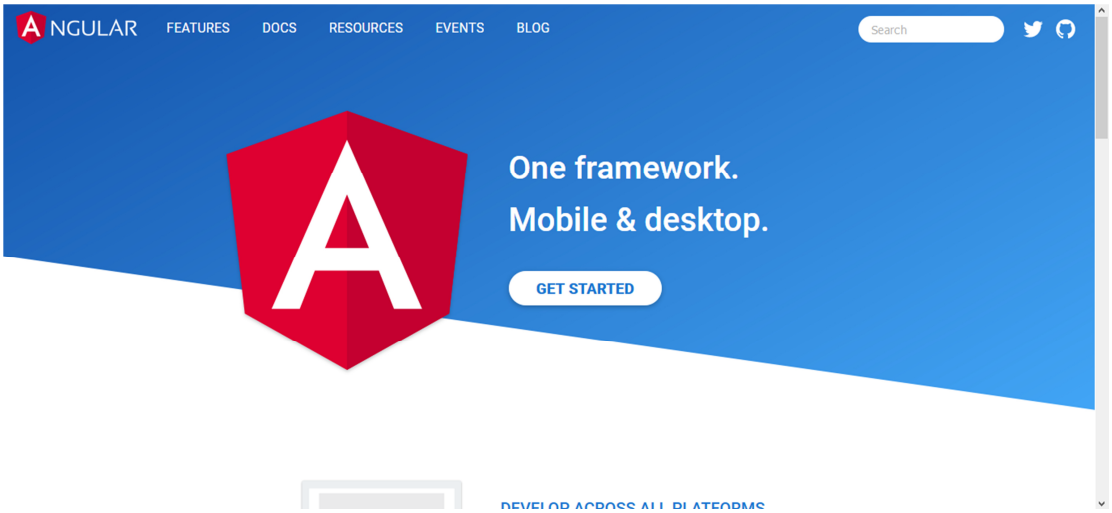
PROGRAMACIÓN ANGULAR – Laboratorio Prog. IV – UTN / TUP

Página WEB Angular.io



Términos Importantes:

- C = Clase
- I = Interface
- K = Constante
- P = Pipe
- @ = Decorador
- F = Funciones
- D = Directivas
- E = Enumeración

Instalar Angular

```
npm install -g @angular/cli
```

Versión Angular

ng version

Crear Proyecto Angular

Abrir una consola Windows o Linux y ejecutar el comando

Navegar hasta la carpeta deseada para el proyecto y ejecutar el comando

ng new NOMBRE_APLICACION

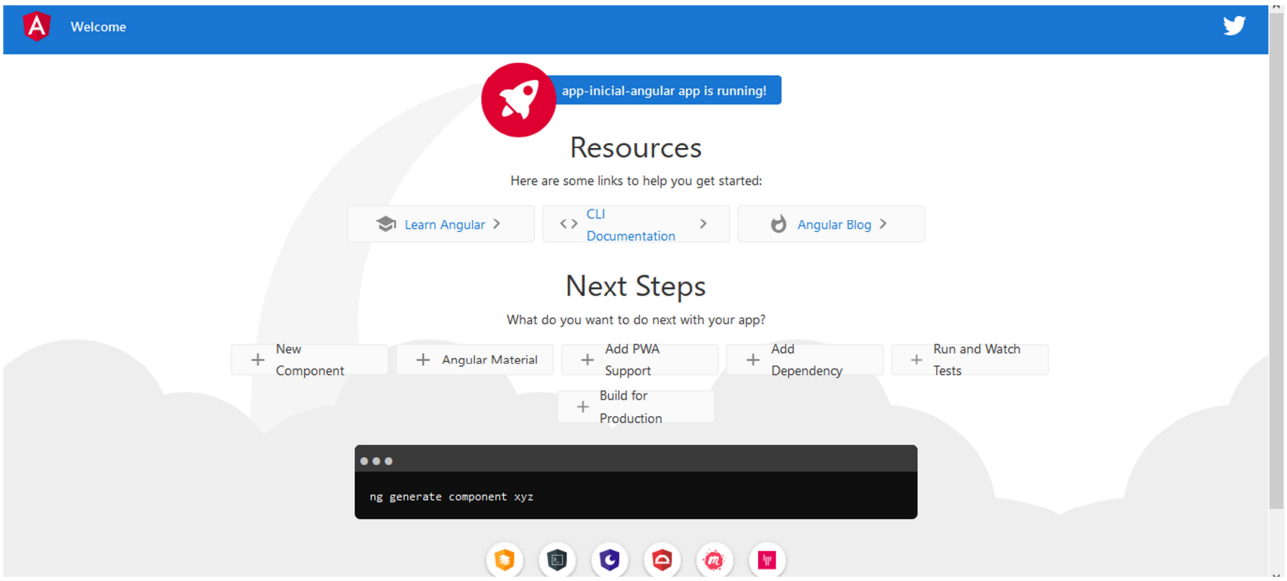
Ejemplo: ng new app-inicial-angular

Cuando les pregunte si quieren agregar el archivo de routes, confirmen con 'y'

Lanzar el servidor de la aplicación

Navegar hasta la carpeta del proyecto creado anteriormente y ejecutar el comando:

ng serve –o



Estructura

e2e: Archivos para realizar pruebas
angular-cli.json: archivos de configuración de la app
editor.config: configuraciones del editor
gitignore: archivos a ignorar en caso de compartir la app ejemplo via git
karma.conf.js: realizar pruebas unitarias
package.json: en caso de desastre por perdidas inesperadas, podemos reconstruir nuestro proyecto
protractor.conf.js: archivo configuración para pruebas

tsconfig.json: archivo configuración de TypeScript
tslint.json: archivo que nos ayuda a que los errores se visualicen de forma amigable
src -> app ->app.module.ts: declaramos los componentes, modulos, proveedores, importaciones, etc
assets-> imágenes, videos y contenido estatico
index.html -> archivo inicial de la app
styles.css -> css principal de la App

Introducción a los componentes y directivas estructurales

Los Componentes serán pequeñas clases que representan a las diferentes secciones de nuestra aplicación, ejemplo encabezado, menú, pie de página.

Las **Directivas estructurales** son instrucciones que le dicen a las interfaces html de nuestra aplicación como debe comportarse. Ejemplo ngIf, nfFor.

Primer App:

Elimine la totalidad del contenido del archivo app.component.html y reemplace el mismo por

```
<h1>Apellido: {{apellido}}</h1>
<h3>Nombre: {{nombre}}</h3>
```

Abrir el archivo app.component.ts y reemplazar la definición de la clase por

```
export class AppComponent {
  apellido:string="Magni";
  nombre:string="Gerardo";
}
```

Debe notar claramente el vínculo que existe entre ambos archivos.

Incorporar Bootstrap:

En la raíz del proyecto

Instalación Automática ejecutar (recomendado Angular actualizado):

```
ng add @ng-bootstrap/ng-bootstrap
```

Verificar que en el archivo app.module.ts figure la línea

```
import { NgbModule } from '@ng-bootstrap/ng-bootstrap';
```

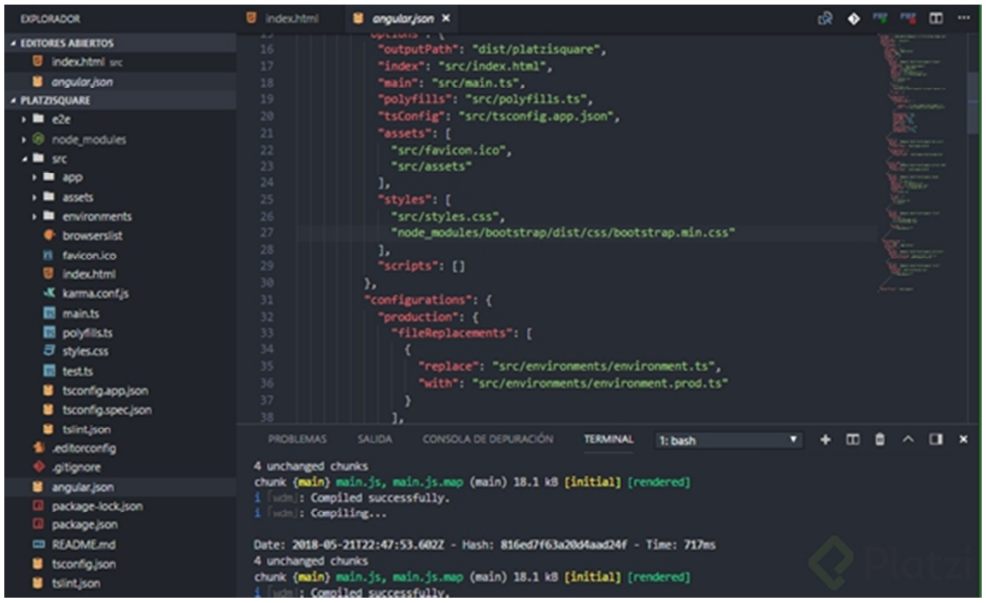
Instalación Manual:

Ejecutar estos pasos

1- Ejecutar

```
npm install bootstrap --save
```

2- Configurar angular.json



3- Escribir el código

```
"styles": [
              "src/styles.css",
              "node modules/bootstrap/dist/css/bootstrap.min.css"
            ]
```

4- Instalar ng-bootstrap, ejecutar los comandos:

```
ng add @angular/localize

npm install @ng-bootstrap/ng-bootstrap
```

5- Ir a app.module.ts y escribir lo siguiente

```
import {NgbModule} from '@ng-bootstrap/ng-bootstrap';

@NgModule({
  imports: [ NgbModule, ... ],
```

```
  // ...
})
export class AppModule {}
```

6- Guardar y ejecutar

Si el texto de la página se modifica es señal de que se incorporó exitosamente las librerías.

Crear e incorporar nuevos componentes (Forma Manual)

Para crear nuevos componentes debemos primero crear nuestro nuevo componente el cual será codificado en un archivo .ts

Por ejemplo cree una carpeta components dentro de src->app

Y dentro de esta carpeta cree el archivo header.component.ts

La estructura de un componente debe tener al menos:

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

@Component({

  })
  export class HeaderComponent {

  }
```

Ejemplo de Componente:

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

@Component({
    selector: 'app-header',
    template: `

    <h1>Esta es la plantilla del encabezado</h1>

    `
  })
  export class HeaderComponent {

  }
```

La propiedad **selector** indica el nombre se la etiqueta personalizada que estamos creando.

Hemos creado nuestro componente pero no podemos hacer uso de él, para hacerlo debemos agregarlo en el archivo app.module.ts

```
import {HeaderComponent} from './components/header.component';
```

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

Finalmente agregamos la etiqueta correspondiente al nuevo componente en nuestro archivo app.component.html

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

<h1>Apellido: {{apellido}}</h1>
<h3>Nombre: {{nombre}}</h3>
```

Vista del Navegador:

Esta es la plantilla del encabezado
Apellido: Magni
Nombre: Gerardo

1- Separando el template del component:

Busquemos NavBar en Bootstrap

Copiamos el código ejemplo y lo usamos como nueva definición de **template** en nuestro archivo **header.component.ts**

Esto si bien funciona no es recomendable, por esto vamos a definir y separar el código del template en otro archivo.

Vamos a crear un archivo **header.component.html** dentro de nuestra carpeta components y copiamos dentro del archivo el código del NavBar

Posteriormente debemos modificar nuestro **header.component.ts,** ahora ya no tenemos que referenciar la propiedad template sino que debemos cambiarla por templateURL y colocar la ruta hacia nuestro html.

```
templateUrl: 'header.component.html'
```

Crear e incorporar nuevos componentes (Forma Automatica)

Ejecutar comando: **ng g c** nombre_componente

Ejemplo: ng g c footer

Resultado->

```
CREATE src/app/footer/footer.component.html (25 bytes)
CREATE src/app/footer/footer.component.spec.ts (628 bytes)
CREATE src/app/footer/footer.component.ts (269 bytes)
CREATE src/app/footer/footer.component.css (0 bytes)
UPDATE src/app/app.module.ts (481 bytes)
```

Incorporar el nuevo componente en nuestro archivo **app.component.html**

Si deseamos evitar la creación del archivo **spec ejecutar el comando:**

ng g c nombre_componente **--nospec**

Pueden modificar el aspecto general de la página modificando el archivo styles.css

Ejemplo agregue en footer.component.html el código

```
<footer>
  <div>
    <p style="text-align: center; background-color: black; color: white">
      &copy;Gerardo Magni
    </p>
  </div>
</footer>
```

Modifique el archivo styles.css agregando

```
footer{
    font-size: 50px;
}
```

Directivas estructurales: *ngFor y el *ngIf

Directiva Estructural *ngIf

```
*ngIf ="expresion"
```

Siendo expresión igual a true o false. Equivalente a clausula if()

Ejemplo:

```
<div *ngIf="mostrar">
  {{frase.mensaje}}
</div>
```

Evento Click

```
<div>
  cuerpo works!
</div>
<div *ngIf="mostrar">
  {{frase.mensaje}}
</div>

<div>
  {{frase.libro}}
</div>

<button type="button" (click)="mostrar = !mostrar">MOSTRAR/OCULTAR</button>
```

Variable de Clase

```
mostrar:boolean = false;
```

Clase TS

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

@Component({
  selector: 'app-cuerpo',
  templateUrl: './cuerpo.component.html',
  styleUrls: ['./cuerpo.component.css']
})
export class CuerpoComponent {

    mostrar:boolean = false;
    frase:any={
      mensaje: "Lo escencial es invisible a los ojos",
      libro: "El Principito"
    }
}
```

Directiva Estructural *ngFor

Si deseo repetir cualquier cosa en HTML

HTML

```
<div>
  cuerpo works!
</div>
<div *ngIf="mostrar">
  {{frase.mensaje}}
</div>

<div>
  {{frase.libro}}
</div>

<button type="button" (click)="mostrar = !mostrar">MOSTRAR/OCULTAR</button>

<ul>
  <li *ngFor="let nombre of nombresArray; let i = index">{{i+1}}. {{nombre}}</li>
</ul>
```

Clase TS

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

@Component({
  selector: 'app-cuerpo',
  templateUrl: './cuerpo.component.html',
  styleUrls: ['./cuerpo.component.css']
})
export class CuerpoComponent {

    mostrar:boolean = false;
    frase:any={
      mensaje: "Lo escencial es invisible a los ojos",
      libro: "El Principito"
    }
    nombresArray:string[]=["Juan", "Pepe", "Jose"]
}
```