ANGULAR

Es un framework para JavaScript que nos va a ayudar a desarrollar aplicaciones web SPA, es decir, una aplicación que no se necesita recargar la página en ningún momento y que está completamente separada del **backend**. La comunicación con el *Backend* es mediante peticiones AJAX y mediante comunicación asincrona.

En caso de tenerlo instalado debemos hacer lo siguiente:

> Desinstalar angular/cli:
> - npm uninstall -g @angular/cli

> Borrar cache: - npm cache verify - npm cache clear --force

INSTALACIÓN DE ANGULAR

> npm install -g @angular/cli

Generar un nuevo proyecto angular

> ng new "nombre de la carpeta"

> Arrancar el proyecto: ng serve --open (se debe estar dentro de la carpeta creada)

CONCEPTOS TEÓRICOS DE ANGULAR

1- Componente

Un componente al final va a controlar un trozo de pantalla o de la vista. Todo lo que se puede ver en pantalla es controlado y gestionado por este tipo de elementos. La lógica de un componente dentro de una clase en Angular es que da soporte a una vista interactuando con ella a través de una API con propiedades y métodos.

El componente hace de mediador entre la vista a través de la plantilla y la lógica de la app donde se incluirá el modelo de datos,es decir una especie de controlador.

2- Plantillas

Las plantillas van a definir la vista de los componentes.

Son htmls y tienen sintaxis especial de Angular. Trabajando con el databinding y las directivas.

3- Decoradores y metadatos

Con los decoradores (patrón de diseño) vamos a configurar dinamicamente atributos/metadatos de las clases y componentes.

Los metadatos van a describir a las clases pero también describen relaciones, por ejemplo si tenemos un componente y una plantilla, el metadato se va a encargar de decirle a Angular que ese componente y esa plantilla van juntos, entre otras muchas cosas.

4- Servicios

Son clases con un objetivo claro, facilita la reutilización, son un tipo de elemento dentro de la arquitectura de Angular y mediante la inyección de dependencias los podemos usar en otro componentes principales.

5- Providers

Son servicios que nos proveen de datos o funcionalidades mediante sus métodos. Existen providers/servicios propios de Angular o creados por nosotros mismos.

6- Directivas

Son funcionalidades aplicables al DOM y a los elementos HTML en las plantillas de un componente. Por ejemplo una directiva puede servir para controlar que un div se muestre o no, recorrer un array en la vista (directivas estructurales, estructuras condicionales y de control) o incluso también para interactuar con el modelo de datos del componente.

Basicamente son nuevos atributos para aplicarlea cualquier cosa en nuestra plantilla/vista.

COMPONENTE

Es una parte de nuestra aplicación. Puedo tener un componente para el header, otro componente para un sidebar, otro para un calendario. Todo en angular está formado por componentes.

> Los componentes se crearán en la carpeta "app", en el archivo app.component.ts

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

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

> Los componentes se inicializan en el archivo "app.module.ts"

EJEMPLO DE DEFINIR UNA TEMPLATE EN LINEA

Creamos una carpeta "videojuego", donde colocaremos 2 archivos "videojuego.component.ts" (para los componentes) y el otro "videojuego.component.html"

En el archivo "videojuego.component.ts" creamos el componente:

```
    import { Component } from "@angular/core"; //Indicar el paquete donde estará
ubicado el componente

    @Component({ //Decorador (no colocar punto y coma)
        selector: 'videojuego', //nombre de la etiqueta
        template: `
            <h2>Componente de Videojuegos</h2>
            <ul>
                <li>GTA</li>
                <li>Prince of Persia</li>
                <li>Tekken</li>
                <li>Mario</li>
            </ul>
            `
    })
    export class VideojuegoComponent {
        constructor(){
            console.log("Se ha cargado un componente")
        }
    }
```

Para poder utilizar este componente, vamos al archivo "app.module.ts" e importo la clase "VideojuegoComponent" y luego declararlo en el array de "declarations"

```
    import { VideojuegoComponent } from './videojuego/videojuego.component';

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

Para utilizar el componente, solo debemos colocar la nueva etiqueta en "app.component.html"

```
    <div class="content" role="main">
    <h1>Bienvenido al {{ title }}</h1>
    <p>Vamos a aprender Angular juntos</p>

    <videojuego></videojuego> <!-- Nuevo componente -->
    </div>
```

> **IMPORTANTE:** Lo más recomendable es definir una template en una vista externa.

Para hacer esto llevamos lo que colocamos el archivo "videojuego.component.ts" en la parte de *Template*, a otro archivo llamado "videojuego.component.html". Hacemos los siguientes cambios:

```
// En videojuego.component.ts
import { Component } from "@angular/core";

@Component({
    selector: 'videojuego',
    templateUrl: './videojuego.component.html'
})

export class VideojuegoComponent {

    constructor() {
        console.log("Se ha cargado el componente")
    }
}
```

```
<!-- En el nuevo archivo videjuego.component.html -->
<h2>Componente de Videojuegos</h2>
<ul>
    <li>GTA</li>
    <li>Prince of Persia</li>
    <li>Tekken</li>
    <li>Mario</li>
</ul>
```

PROPIEDADES

Dentro de las clases se pueden colocar propiedades Publicas.

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

@Component({
    selector: 'videojuego',
    templateUrl: './videojuego.component.html'
})

export class VideojuegoComponent {
    public titulo: string;
    public listado: string;

    constructor() {
        this.titulo = "Videojuegos";
```

```
            this.listado = "Listado de los juegos más populares";
            console.log("Se ha cargado el componente");
        }
    }
```

En el videojuego.component.html

```
    <h2>{{titulo}}</h2> <!--Se incrusta la variable creada en el constructor  -->
    <p>{{listado}}</p>
    <ul>
        <li>GTA</li>
        <li>Prince of Persia</li>
        <li>Tekken</li>
        <li>Mario</li>
    </ul>
```

MULTIPLES / VARIOS COMPONENTES

Si creamos otros componentes, estos podemos reutilizarlos de la manera que queramos, podemos invocarlos de manera directa en el html principal, o colocarlo dentro de otro componente, para que este componente lo muestre luego. En ambas situaciones va a funcionar el componente.

Ejemplo: el componente "zapatillas" a sido creado como otro componente pero puede utilizarse sin ningún problema en el componente "videojuego"

```
    <!-- videojuego.component.html -->
    <h2>{{titulo}}</h2>
    <p>{{listado}}</p>
    <ul>
        <li>GTA</li>
        <li>Prince of Persia</li>
        <li>Tekken</li>
        <li>Mario</li>
    </ul>
    <zapatillas></zapatillas>
```

CREAR COMPONENTES DESDE LA CONSOLA

Se recomienda utilizar CMD en Windows Ingresar a la ruta del proyecto y colocar:

> ng g component (nombre del componente)

Al crearlo de esta manera, el sistema crea automáticamente el import y el declaration en el app.module.ts

HOOKS / EVENTOS CICLO DE VIDA

> Hooks: Eventos de un componente que se ejecutan en un momento dado del ciclo de vida del componente. Se lanzan dependiendo del estado del componente.

OnInit

Se implenta en una interfaz de la siguiente manera:

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

@Component({
    selector: 'cursos',
    templateUrl: './cursos.component.html',
    styleUrls: ['./cursos.component.css']
})
export class CursosComponent implements OnInit {

constructor() { }

ngOnInit(): void {
}

}
```

Este Hook brinda acceso a los siguientes métodos:

- **ngOnInit:** Este es el método que se ejecuta nada más cargar un componente.

```
ngOnInit(): void {
    }
```

- **ngDoCheck:** este metodo se ejecuta cada vez que hay algún cambio en el código.

> export class VideojuegoComponent implements OnInit, DoCheck{ }

```
ngDoCheck(): void {
    console.log("DoCheck ejecutado");
}
```

- **OnDestroy:** este metodo se ejecuta cuando se elimina un componente.

> export class VideojuegoComponent implements OnInit, DoCheck, OnDestroy{ }

```
ngOnDestroy(): void {
    console.log("OnDestroy ejecutado")
}
```

¿Cómo crear una clase e importarla dentro de otra clase?

Creamos una nueva carpeta con el archivo "configuracion.ts", dentro de ese archivo colocamos este objeto:

```
    export var Configuracion = { //se debe colocar "export"
        color: "red",
        fondo: "blue",
        titulo: "Aplicación con Angular",
        descripcion: "Aprendiendo Angular con Victor Robles"
    }
```

Para acceder a este objeto debebos importarlo en el archivo donde lo deseamos aplicar, en este caso en el "app.component.ts"

> import { Configuracion } from './models/configuracion';

```
    export class AppComponent {
        //Inicializamos todos lo atributos
        public title = 'Master en JavaScript';
        public descripcion: string;
        public mostrar_videojuegos: boolean = true;

        constructor(){ //Con el constructor enlazamos a la variable y al objeto
            this.config = Configuracion;// Al asignar el objeto a una variable
podré acceder a él a través de dicha variable, no se necesitaría declarar todos
los atributos del objeto.
            this.title = Configuracion.titulo;
            this.descripcion = Configuracion.descripcion
    }
```

En el HTML colocamos las variables creadas en el constructor

```
    <!-- De esta manera accederemos desde la vista a los atributos del objeto
Configuracion -->
    <div [ngStyle]="{'background': config.color}" class="content" role="main">
    <!-- Puede reemplazase por: <div style="background: {{config.color}}"
class="content" role="main"> -->
    <h1>Bienvenido al {{ title }}</h1>
    <p>{{ descripcion }}</p>
```

En la directiva [ngStyle] se pueden colocar más atributos:

```
<div [ngStyle]="{
                'background': config.fondo,
                'padding': '20px',
                'border' : '5px solid black',
                'border-color': config.color
            }" class="content" role="main">
```

MODELO DE DATOS

Un modelo de una entidad puede tener diversas propiedades, que pueden servir como molde para la creación de diversos objetos.

> **Importante:** Cuando se vaya a crear un nuevo modelo, se recomienda que su nombre esté en singular, dado que el modelo representa a un solo objeto.

Crear un nuevo modelo de datos

En un nuevo archivo (zapatillas.ts) colocamos lo siguiente:

```
// De manera larga, menos óptima
export class Zapatilla{
    public nombre: string;
    public marca: string;
    public color: string;
    public precio: string;
    public stock: boolean;

    constructor(nombre, marca, color, precio, stock){
        this.nombre = nombre;
        this.marca = marca;
        this.color = color;
        this.precio = precio;
        this.stock = stock;
    }
}
```

Todo lo colocado anteriormente lo podemos resumir de la siguiente manera:

```
export class Zapatilla{
    constructor(
        public nombre: string,
        public marca: string,
        public color: string,
        public precio: number,
        public stock: boolean
    ){}
}
```

Importar el nuevo modelo de datos

En el archivo donde se desee importar (zapatillas.component.ts), colocamos lo siguiente:

```
    import { Component, OnInit } from "@angular/core";
    import { Zapatilla } from "../models/zapatillas";

    @Component({
        selector: 'zapatillas',
        templateUrl: './zapatillas.component.html'
    })

    export class ZapatillasComponent implements OnInit{
        public titulo: string = "Componente de Zapatillas"
        public zapatillas: Array<Zapatilla>; // Crear array para colocar los
objetos
        constructor(){
            // Haciendo referencia al Array "zapatillas"
            this.zapatillas = [
                new Zapatilla('Nike Airmax', 'Nike', 'Blancas', 40, true),
                new Zapatilla('Reebook Clasic', 'Reebok', 'Blanco', 80, true),
                new Zapatilla('Nike Runner MD', 'Nike', 'Negras', 60, true),
                new Zapatilla('Adidas Yezzy', 'Adidas', 'Azul', 180, false)
            ];
        }

        ngOnInit(): void {
            // Al implementar onInit obligatoriamente debemos utilizar esta
interfaz para mostrar los resultados
            console.log(this.zapatillas);
        }
    }
```

Para mostrar todos los objetos almacenados en el array en la vista(zapatillas.component.html) solo es necesario utilizar la directiva *ngFor*

```
    <h2>{{titulo}}</h2>
    <ul>
        <li *ngFor="let deportiva of zapatillas">
        {{deportiva.nombre}} - <strong>{{deportiva.precio}}€</strong>
        </li>
    </ul>
```