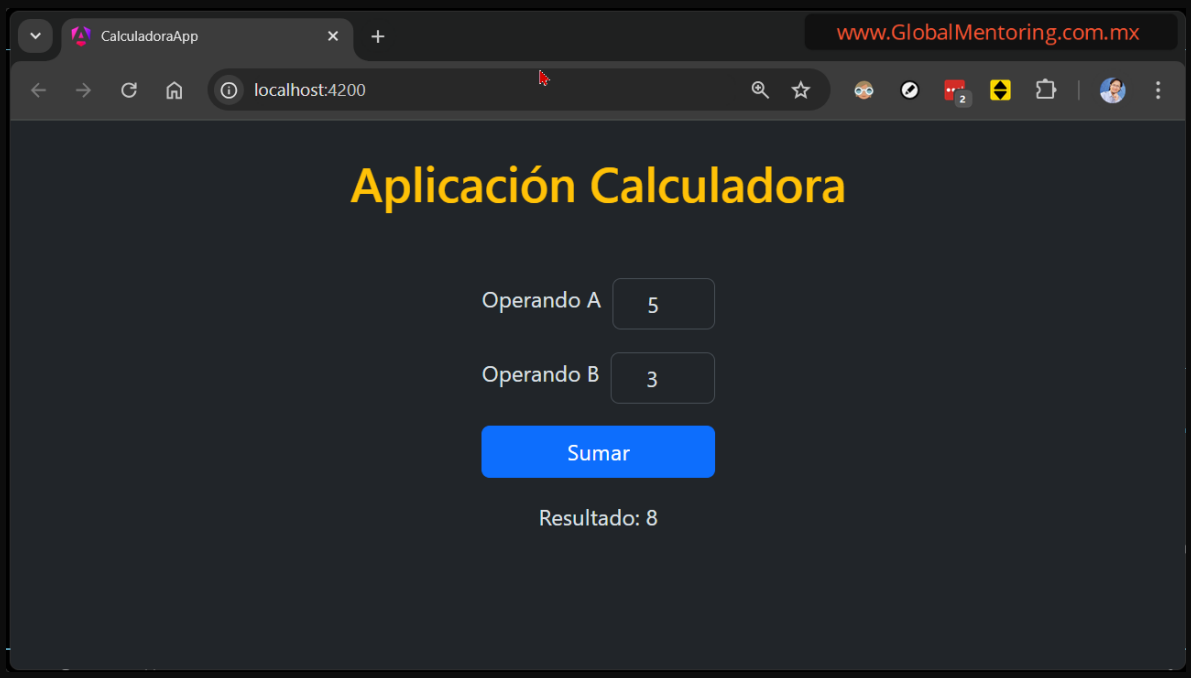
Reto: Aplicación Calculadora en Angular

Aplicación de Calculadora en Angular

Vamos a crear una aplicación de calculadora en Angular, similar a la que se muestra en la imagen que proporcionaste. Utilizaremos Bootstrap 5 para el diseño, componentes standalone, y two-way data binding para manejar la entrada y salida de datos.



Paso 1: Crear el Proyecto en Angular

Primero, creamos un nuevo proyecto:

```
ng new calculadora-app
```

Segundo, abrimos la nueva aplicación en VSC.

Paso 2: Configurar Boostrap.

Agregar el módulo de boostrap y configurarlo en el archivo angular.json como ya se ha visto anteriormente.

Paso 3: Crear el Componente Standalone

A continuación, creamos un componente standalone para la calculadora:

```
ng g c calculadora --skip-tests
```

Paso 4: Implementar el Componente Calculadora

Vamos a implementar el componente standalone que manejará la lógica de la calculadora.

Código del Componente (`calculadora.component.ts`):

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

@Component({
  selector: 'app-calculadora',
  standalone: true,
  imports: [FormsModule],
  templateUrl: './calculadora.component.html',
  styleUrl: './calculadora.component.css'
})
export class CalculadoraComponent {
  operandoA: number = 0;
  operandoB: number = 0;
  resultado: number = 0;

  sumar(): void {
    this.resultado = this.operandoA + this.operandoB;
  }

}
```

Aquí tienes un resumen de los puntos importantes de Angular en este código:

1. `standalone: true`

- **Componente Standalone**: Este componente se define como standalone, lo que significa que no forma parte de un módulo Angular tradicional (`NgModule`). Esto permite que el componente sea autónomo y pueda importar directamente sus dependencias necesarias.

2. `imports: [FormsModule]`

- **Importación Directa de Módulos**: Al ser un componente standalone, se importan directamente los módulos necesarios en el componente. Aquí se importa `FormsModule`, que es necesario para utilizar `ngModel` para two-way data binding en la plantilla.

3. Propiedades del Componente

- **`operandoA` y `operandoB`:** Son propiedades numéricas que almacenan los valores ingresados en los campos de entrada correspondientes.
- **`resultado`:** Es una propiedad numérica que almacena el resultado de la operación de suma.

4. Método `sumar()`

- **Lógica del Componente**: Este método realiza la suma de `operandoA` y `operandoB` y asigna el resultado a la propiedad `resultado`. Este método se invoca mediante un evento `click` en la plantilla, lo que demuestra el uso de **event binding**.

5. Uso de Tipos de Datos

- **Tipado Estricto**: Las propiedades `operandoA`, `operandoB`, y `resultado` están tipadas como `number`, lo que aprovecha la fuerte tipificación de TypeScript para evitar errores en tiempo de compilación y mejorar la robustez del código.

Paso 5: Implementar la Plantilla del Componente

Usamos Bootstrap para estilizar la aplicación y `ngModel` para el two-way data binding.

Código de la Plantilla (`calculadora.component.html`):

```
<div class="container mt-5">
    <div class="d-flex justify-content-center">
        <div class="w-25">
            <div class="mb-3 d-flex align-items-center">
```

```
                <label for="operandoA" class="form-label me-2 text-nowrap">Operando A</label>
                <input type="number" id="operandoA" class="form-control text-center"
                        [(ngModel)]="operandoA">
            </div>
            <div class="mb-3 d-flex align-items-center">
                <label for="operandoB" class="form-label me-2 text-nowrap">Operando B</label>
                <input type="number" id="operandoB" class="form-control text-center"
                        [(ngModel)]="operandoB">
            </div>
            <button class="btn btn-primary mb-3 w-100" (click)="sumar()">Sumar</button>
            <p class="text-center">Resultado: {{ resultado }}</p>
        </div>
    </div>
</div>
```

Aquí tienes una explicación de los puntos importantes de Angular presentes en el código:

1. `[(ngModel)]="operandoA"` y `[(ngModel)]="operandoB"`

- **Two-Way Data Binding**: Estas directivas `[(ngModel)]` permiten el enlace bidireccional entre las propiedades del componente (`operandoA` y `operandoB`) y los campos de entrada (`<input>`). Esto significa que cualquier cambio en los campos de entrada se reflejará automáticamente en las propiedades del componente y viceversa.
- **FormsModule Necesario**: Para que `ngModel` funcione, el módulo `FormsModule` debe estar importado en el componente o módulo correspondiente.

2. `(click)="sumar()"`

- **Event Binding**: Esta es una forma de enlazar un evento DOM (en este caso, el evento `click` del botón) a un método del componente (`sumar`). Cuando el usuario hace clic en el botón, se ejecuta el método `sumar()`, que realiza la operación deseada (sumar los valores de `operandoA` y `operandoB`).

3. `{{ resultado }}`

- **Interpolación**: Esta técnica se usa para mostrar el valor de una propiedad del componente (`resultado`) directamente en la vista. Angular reemplaza la expresión dentro de `{{ }}` con el valor actual de la propiedad `resultado`, que se actualiza cada vez que se llama al método `sumar()`.

4. Estructura de Componente Standalone

- **Componentes Standalone**: El código asume que el componente es standalone, lo que significa que no está dentro de un módulo tradicional, sino que se importa y

utiliza directamente. En este contexto, `FormsModule` debe ser importado explícitamente dentro del componente si se está utilizando `ngModel`.

5. Tipo de Dato en `input type="number"`

- **Validación de Tipo**: Al utilizar `type="number"` en el campo de entrada, nos aseguramos que sólo se puedan ingresar valores numéricos, lo que facilita la manipulación de los datos en el componente sin necesidad de validaciones o conversiones adicionales.

También, les dejamos un resumen rápido de las clases Bootstrap usadas en el código:

- **`container`**: Centra el contenido y aplica márgenes automáticos.
- **`mt-5`**: Añade un margen superior de `3rem` (48px).
- **`text-center`**: Centra el texto horizontalmente.
- **`d-flex`**: Aplica un layout de flexbox para alinear los elementos horizontalmente.
- **`justify-content-center`**: Centra los elementos hijos dentro del contenedor flexbox.
- **`w-25`**: Define el ancho del contenedor al 25% del contenedor padre.
- **`mb-3`**: Añade un margen inferior de `1rem` (16px).
- **`align-items-center`**: Centra verticalmente los elementos dentro del contenedor flexbox.
- **`form-label`**: Estiliza las etiquetas de formulario.
- **`me-2`**: Añade un margen derecho de `0.5rem` (8px).
- **`text-nowrap`**: Evita que el texto se divida en varias líneas.
- **`form-control`**: Estiliza los campos de entrada.
- **`w-100`**: Hace que el botón ocupe el 100% del ancho del contenedor.

Este es un resumen conciso de las clases utilizadas en el código y su propósito dentro del diseño.

Paso 6: Asegurarse de Tener `FormsModule` Importado

Ya que estás usando un componente standalone, `FormsModule` debe ser importado directamente en el componente, lo cual ya se ha hecho en el código del componente anterior.

Paso 7: Configurar la Aplicación Principal

Configura la aplicación principal para que use el componente `CalculadoraComponent`.

Código de `app.component.ts`:

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

```
import { CalculadoraComponent } from "./calculadora/calculadora.component";

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [RouterOutlet, CalculadoraComponent],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {
  titulo = 'Aplicación Calculadora';
}
```

Código de `app.component.html`:

```
<section>
  <div class="container text-center">
      <h1 class="text-center text-warning my-4">{{ titulo }}</h1>
      <app-calculadora/>
  </div>
</section>
```

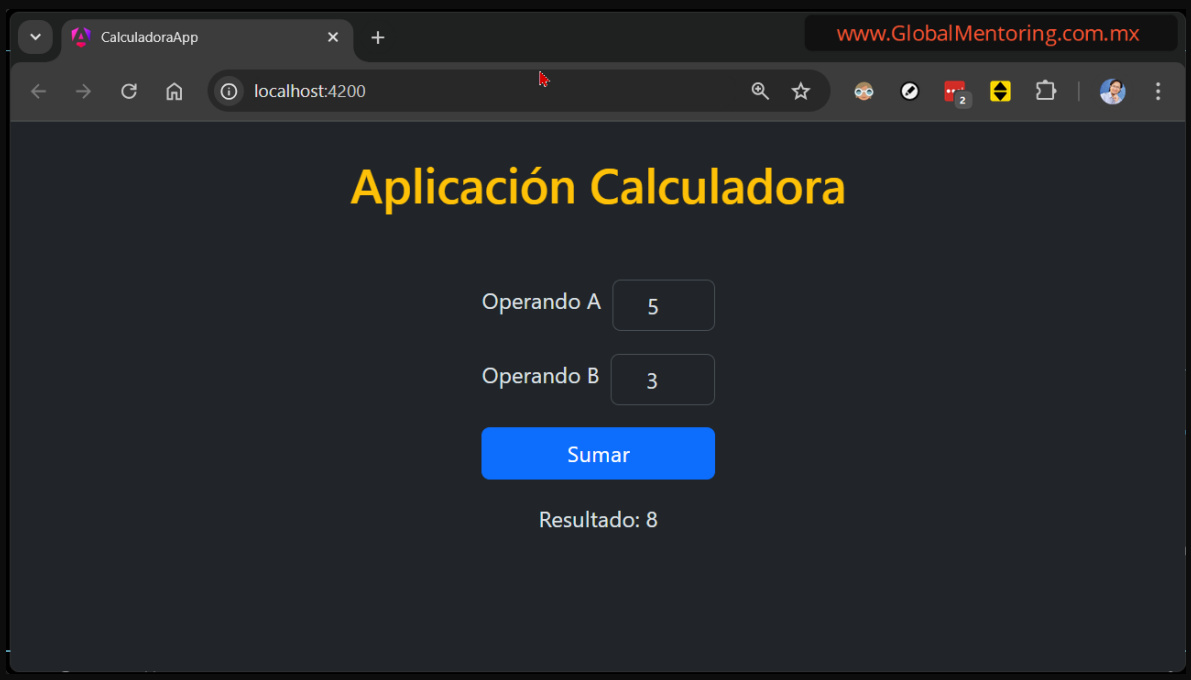
Paso 8: Ejecutar la Aplicación

Finalmente, ejecuta la aplicación para ver el resultado:

```
ng serve -o
```

Resultado Esperado:

La aplicación debe verse similar a la imagen proporcionada, con una interfaz simple que permite ingresar dos números y mostrar su suma cuando se presiona el botón "Sumar".



Saludos!

Ing. Ubaldo Acosta

Fundador de GlobalMentoring.com.mx