Aplicación Presupuesto en Angular

Vamos a crear a continuación la aplicación de Manejo de Presupuesto con Angular.

Pondremos en práctica todos los conceptos estudiados hasta el momento.

A partir del archivo html que les entregamos, el objetivo del ejercicio es hacer la aplicación dinámica usando Angular.

Archivo index.html base:

```
<!doctype html>
```

```
<html lang="en" data-bs-theme="dark">
<head>
  <meta charset="utf-8">
  <title>AppPresupuesto</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <!-- Bootstrap CSS -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
rel="stylesheet">
  <!-- Bootstrap Icons -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap-icons/font/bootstrap-icons.css"
rel="stylesheet">
</head>
<body>
  <div class="container my-5">

    <!-- Cabecero centrado -->
    <!-- Componente: Cabecero -->
    <div class="row justify-content-center">
      <div class="p-4 rounded border border-primary text-center">
        <h4 class="mb-3">Presupuesto Disponible:</h4>
        <div class="display-3 fw-bold text-warning">$3,400.00</div>
        <div class="row mt-4 justify-content-center">
          <div class="col-md-5 text-center">
            <div class="text-uppercase">INGRESOS</div>
            <div class="text-success h5">$4,500.00</div>
          </div>
          <div class="col-md-5 text-center">
            <div class="text-uppercase">EGRESOS</div>
            <div class="text-danger h5">
              $1,100.00
              <span class="badge bg-success ms-2">24%</span>
            </div>
          </div>
        </div>
      </div>
    </div>

    <!-- Formulario centrado -->
    <!-- Componente: Formulario -->
    <div class="row justify-content-center mt-4">
      <div class="col-md-8">
        <form class="row g-2">
          <div class="col-auto">
            <select class="form-select bg-success">
```

```
              <option value="ingresoOperacion" selected>+</option>
              <option value="egresoOperacion">-</option>
            </select>
          </div>
          <div class="col">
            <input type="text" class="form-control border-success" placeholder="Agregar Descripción"
required />
          </div>
          <div class="col-auto">
            <input type="number" class="form-control border-success" placeholder="Valor" required />
          </div>
          <div class="col-auto">
            <button type="submit" class="btn btn-success">
              <i class="bi bi-check-circle"></i> <!-- Bootstrap Icon -->
            </button>
          </div>
        </form>
      </div>
    </div>

    <!-- Ingresos y Egresos en columnas de igual tamaño -->
    <div class="row mt-4">
      <!-- Componente: Ingresos -->
      <div class="col-md-6">
        <h2 class="text-center text-primary">Ingresos</h2>
        <ul class="list-group mt-3">
          <li class="list-group-item d-flex justify-content-between align-items-center">
            <span class="flex-grow-1">Salario</span>
            <span class="text-success mx-3">$4,000.00</span>
            <button class="btn btn-sm btn-danger">
              <i class="bi bi-x-circle"></i> <!-- Bootstrap Icon -->
            </button>
          </li>
          <li class="list-group-item d-flex justify-content-between align-items-center">
            <span class="flex-grow-1">Venta Coche</span>
            <span class="text-success mx-3">$500.00</span>
            <button class="btn btn-sm btn-danger">
              <i class="bi bi-x-circle"></i> <!-- Bootstrap Icon -->
            </button>
          </li>
        </ul>
      </div>

      <!-- Componente: Egresos -->
      <div class="col-md-6">
```

```
        <h2 class="text-center text-danger">Egresos</h2>
        <ul class="list-group mt-3">
          <li class="list-group-item d-flex justify-content-between align-items-center">
            <span class="flex-grow-1">Renta Depto</span>
            <span class="d-flex align-items-center mx-3">
              <span class="text-danger">- $900.00</span>
              <span class="badge bg-success ms-2">20%</span>
            </span>
            <button class="btn btn-sm btn-danger">
              <i class="bi bi-x-circle"></i> <!-- Bootstrap Icon -->
            </button>
          </li>
          <li class="list-group-item d-flex justify-content-between align-items-center">
            <span class="flex-grow-1">Ropa</span>
            <span class="d-flex align-items-center mx-3">
              <span class="text-danger">- $200.00</span>
              <span class="badge bg-success ms-2">4%</span>
            </span>
            <button class="btn btn-sm btn-danger">
              <i class="bi bi-x-circle"></i> <!-- Bootstrap Icon -->
            </button>
          </li>
        </ul>
      </div>
    </div>

  </div>
</body>
</html>
```

Solución de la aplicación de Presupuesto con Angular

En primer lugar, vamos a crear de la aplicación de Presupuesto con Angular:

```
ng new app-presupuesto
```

Para esta aplicación crearemos varios componentes, desde el cabecero de la aplicación de presupuesto, el componente de egreso, el componente de ingreso y un componente de formulario para capturar un ingreso o egreso. Vamos a aplicar varios conceptos más a lo largo de la aplicación, así que será una aplicación bastante completa para poner en práctica todo lo estudiado hasta el momento. Comencemos con la creación de componentes:

Componente cabecero:

```
ng g c cabecero --skip-tests
```

Componente ingreso:

ng g c ingreso --skip-tests

Componente egreso:

ng g c egreso --skip-tests

Componente formulario:

ng g c formulario --skip-tests

Creamos ahora las **Clases de Modelo** de ingreso y egreso dentro de las carpetas respectivas de cada componente. Estos archivos los creamos manualmente. Esta es la ruta y nombre de cada archivo:

Archivo Ruta: ingreso/ingreso.model.ts

Archivo Ruta: egreso/egreso.model.ts

Creamos ahora algunos **Servicios** para poder trabajar con los arreglos de ingresos y egresos. Cada uno de ellos los creamos dentro de la carpeta respectiva de ingreso o egreso según corresponda.

```
ng g s ingreso/ingreso --skip-tests
```

```
ng g s egreso/egreso --skip-tests
```

Una vez creados los componentes y servicios, empecemos a trabajar sobre cada uno de ellos:

Agregamos Bootstrap, y los íconos de Bootstrap ya que también los vamos a utilizar en nuestra aplicación:

```
npm install bootstrap@latest --save
```

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

Configurar Bootstrap en Angular

Después de instalar Bootstrap, debes agregar los estilos de Bootstrap en tu proyecto. Esto se hace incluyendo el archivo CSS de Bootstrap en el archivo `angular.json`.

1. Abre el archivo `angular.json`.
2. Busca la sección `"styles"` y agrega la ruta al archivo CSS de Bootstrap. También puedes agregar la ruta a Popper.js de manera opcional en la sección `"scripts"` si planeas usar componentes que lo necesiten.

```
          "styles": [
            "node_modules/bootstrap/dist/css/bootstrap.min.css",
```

```
              "node_modules/bootstrap-icons/font/bootstrap-icons.css",
              "src/styles.css"
            ],
            "scripts": [
              "node_modules/bootstrap/dist/js/bootstrap.min.js"
            ]
```

Aplicar el modo dark (oscuro) a toda la aplicación usando Bootstrap. Agregamos lo siguiente al archivo index.html:

```
<html lang="en" data-bs-theme="dark">
```

Código del componente app.component.html (Recordar importar los componentes utilizados):

```
<div class="container my-5">
  <!-- Cabecero centrado -->
  <div class="row justify-content-center">

      <app-cabecero
        [presupuestoTotal]="getPresupuestoTotal()"
        [ingresoTotal]="getIngresoTotal()"
        [egresoTotal]="getEgresoTotal()"
        [porcentajeTotal]="getPorcentajeTotal()"
      ></app-cabecero>
      </div>

  <!-- Formulario centrado -->
  <div class="row justify-content-center mt-4">
    <div class="col-md-8">
      <app-formulario></app-formulario>
    </div>
  </div>

  <!-- Ingresos y Egresos en columnas de igual tamaño -->
  <div class="row mt-4">
    <div class="col-md-6">
      <app-ingreso></app-ingreso>
    </div>
    <div class="col-md-6">
      <app-egreso [ingresoTotal]="getIngresoTotal()"></app-egreso>
    </div>
  </div>
</div>
```

Aquí tienes un resumen de las clases de **Bootstrap** utilizadas en el código:

1. **`container`**: Crea un contenedor con márgenes laterales automáticos, ideal para el diseño centrado y con márgenes predefinidos.
2. **`my-5`**: Añade un margen superior e inferior de 5 unidades (espaciado vertical – eje y).
3. **`row`**: Define una fila en el sistema de rejilla de Bootstrap, utilizada para organizar las columnas.
4. **`justify-content-center`**: Centra horizontalmente el contenido de la fila.
5. **`mt-4`**: Añade un margen superior de 4 unidades (margin-top, espaciado vertical entre elementos).
6. **`col-md-8`**: Define una columna que ocupa 8 de las 12 columnas disponibles en pantallas medianas (a partir de 768px).
7. **`col-md-6`**: Define una columna que ocupa 6 de las 12 columnas disponibles en pantallas medianas (a partir de 768px), dividiendo el espacio entre ingresos y egresos en partes iguales.

Código del componente app.component.ts

```
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';
import { FormularioComponent } from "./formulario/formulario.component";
import { IngresoComponent } from "./ingreso/ingreso.component";
import { EgresoComponent } from "./egreso/egreso.component";
import { CabeceroComponent } from "./cabecero/cabecero.component";
import { Ingreso } from './ingreso/ingreso.model';
import { Egreso } from './egreso/egreso.model';
import { IngresoService } from './ingreso/ingreso.service';
import { EgresoService } from './egreso/egreso.service';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [RouterOutlet, FormularioComponent, IngresoComponent, EgresoComponent,
CabeceroComponent],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {
  ingresos:Ingreso[]=[];
  egresos:Egreso[]=[];

  constructor(
    private ingresoServicio: IngresoService,
    private egresoServicio: EgresoService
  ){
```

```
    this.ingresos = this.ingresoServicio.ingresos;
    this.egresos = this.egresoServicio.egresos;
  }

  getIngresoTotal(){
    let ingresoTotal:number=0;
    this.ingresos.forEach(ingreso =>{
      ingresoTotal += ingreso.valor;
    });
    return ingresoTotal;
  }

  getEgresoTotal(){
    let egresoTotal:number=0;
    this.egresos.forEach(egreso =>{
      egresoTotal += egreso.valor;
    });
    return egresoTotal;
  }

  getPorcentajeTotal(){
    return this.getEgresoTotal()/this.getIngresoTotal();
  }

  getPresupuestoTotal(){
    return this.getIngresoTotal() - this.getEgresoTotal();
  }
}
```

Código del componente cabecero.component.html:

```
<div class="p-4 rounded border border-primary text-center">
    <h4 class="mb-3">Presupuesto Disponible:</h4>

    <div class="display-3 fw-bold text-warning">
        {{presupuestoTotal | currency:'MNX':'$':'1.2-2'}}
    </div>

    <div class="row mt-4 justify-content-center">
        <div class="col-md-5 text-center">
            <div class="text-uppercase">INGRESOS</div>
            <div class="text-success h5">
                {{ingresoTotal | currency:'MNX':'$':'1.2-2'}}
            </div>
```

```
        </div>
        <div class="col-md-5 text-center">
            <div class="text-uppercase">EGRESOS</div>
            <div class="text-danger h5">
                {{egresoTotal | currency:'MNX':'$':'1.2-2'}}
                <span class="badge bg-success ms-2">{{porcentajeTotal | percent}}</span>
            </div>
        </div>
    </div>
</div>
```

Aquí tienes el resumen con la explicación en español e inglés de cada clase de **Bootstrap** utilizada, indicando también el significado de los términos en inglés:

1. **`p-4`**: Añade relleno (padding) de 4 unidades. *(p = padding)*
2. **`rounded`**: Aplica bordes redondeados. *(rounded = redondeado)*
3. **`border`**, **`border-primary`**: Añade un borde y lo colorea de azul (primario). *(border = borde)*
4. **`text-center`**: Centra el texto. *(center = centrado)*
5. **`mb-3`**: Añade margen inferior de 3 unidades. *(mb = margin-bottom)*
6. **`display-3`**: Hace el texto grande. *(display = tamaño de texto)*
7. **`fw-bold`**: Aplica negrita al texto. *(fw = font-weight, bold = negrita)*
8. **`text-warning`**: Colorea el texto de amarillo. *(warning = advertencia)*
9. **`row`**: Define una fila en el sistema de rejilla. *(row = fila)*
10. **`mt-4`**: Añade margen superior de 4 unidades. *(mt = margin-top)*
11. **`justify-content-center`**: Centra las columnas horizontalmente. *(justify-content = justificación de contenido, center = centro)*
12. **`col-md-5`**: Define columnas que ocupan 5/12 del ancho en pantallas medianas. *(col = columna, md = tamaño mediano)*
13. **`text-uppercase`**: Convierte el texto a mayúsculas. *(uppercase = mayúsculas)*
14. **`text-success`**: Colorea el texto de verde. *(success = éxito)*
15. **`h5`**: Aplica estilo de encabezado de nivel 5. *(h = heading, encabezado)*
16. **`text-danger`**: Colorea el texto de rojo. *(danger = peligro)*
17. **`badge`**: Aplica estilo de etiqueta o insignia. *(badge = insignia)*
18. **`bg-success`**: Colorea el fondo de la etiqueta de verde. *(bg = background, success = éxito)*
19. **`ms-2`**: Añade margen izquierdo de 2 unidades. *(ms = margin-start, start = inicio, izquierdo)*

Este resumen ahora incluye tanto el significado en inglés de las abreviaciones de **Bootstrap** como su explicación en español, lo que te permitirá entender mejor qué se está aplicando con cada clase.

Código del componente cabecero.component.ts:

```
import { CommonModule } from '@angular/common';
```

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

@Component({
  selector: 'app-cabecero',
  standalone: true,
  imports: [CommonModule],
  templateUrl: './cabecero.component.html',
  styleUrl: './cabecero.component.css'
})
export class CabeceroComponent {
  @Input() presupuestoTotal!:number;
  @Input() ingresoTotal!:number;
  @Input() egresoTotal!:number;
  @Input() porcentajeTotal!:number;
}
```

El operador `!` en TypeScript se llama **non-null assertion operator** (operador de afirmación de no nulo). Este operador le dice al compilador que confíe en ti, es decir, que el valor de esa variable **nunca será `null` o `undefined`**, a pesar de que en el contexto actual podría no estar inicializado.

Código del componente formulario.component.html:

```
<form (ngSubmit)="f.form.valid && agregarValor()" #f="ngForm" class="row g-2">
    <div class="col-auto">
        <select (change)="tipoOperacion($event)" class="form-select"
            [ngClass]="{ 'bg-danger':
             tipo === 'egresoOperacion', 'bg-success': tipo === 'ingresoOperacion' }">
            <option value="ingresoOperacion" selected>+</option>
            <option value="egresoOperacion">-</option>
        </select>
    </div>

    <div class="col">
        <input type="text" class="form-control"
            placeholder="Agregar Descripción" name="descripcionInput"
            [(ngModel)]="descripcionInput" required
            [ngClass]="{ 'border-danger':
        tipo === 'egresoOperacion', 'border-success': tipo === 'ingresoOperacion' }" />
    </div>

    <div class="col-auto">
        <input type="number" class="form-control" placeholder="Valor"
            name="valorInput" [(ngModel)]="valorInput" required
            [ngClass]="{ 'border-danger':
         tipo === 'egresoOperacion', 'border-success': tipo === 'ingresoOperacion' }" />
```

```
    </div>

    <div class="col-auto">
        <button type="submit" class="btn"
            [ngClass]="{ 'btn-danger':
            tipo === 'egresoOperacion', 'btn-success': tipo === 'ingresoOperacion' }">
            <i class="bi bi-check-circle"></i>
        </button>
    </div>
</form>
```

Veamos con más detalle cómo funciona la condición en el código **`[ngClass]`**.

Explicación de la condición en `[ngClass]`:

```
[ngClass]="{ 'btn-danger': tipo === 'egresoOperacion', 'btn-success': tipo ===
'ingresoOperacion' }"
```

El atributo **`[ngClass]`** en Angular permite agregar o eliminar clases dinámicamente a un elemento HTML según ciertas condiciones. El formato que ves es una **expresión de objeto** que asocia las clases de CSS con una condición booleana (una que evalúa si es `true` o `false`).

Estructura:

- La estructura es un objeto de JavaScript que tiene este formato:

```
{
  'nombreClase1': condicion1,
  'nombreClase2': condicion2,
  ...
}
```

- Cada propiedad (o "clave") es el **nombre de la clase de CSS** que deseas aplicar, y su valor es una **condición** que, cuando es verdadera (`true`), hará que la clase se aplique al elemento.
- Si la condición es falsa (`false`), la clase no se aplica.

¿Cómo funciona en nuestro código?

1. **`'btn-danger': tipo === 'egresoOperacion'`**:
    - Si la variable `tipo` es igual a `'egresoOperacion'`, se aplicará la clase **`btn-danger`** (que pinta el botón de rojo).

- Si `tipo` **NO es** `'egresoOperacion'`, no se aplicará la clase **`btn-danger`**.

2. **`'btn-success': tipo === 'ingresoOperacion'`**:
   - Si la variable `tipo` es igual a `'ingresoOperacion'`, se aplicará la clase **`btn-success`** (que pinta el botón de verde).
   - Si `tipo` **NO es** `'ingresoOperacion'`, no se aplicará la clase **`btn-success`**.

Aquí tienes un resumen de las clases de **Bootstrap** utilizadas en el código anterior:

1. **`row`**: Crea una fila en el sistema de rejilla de Bootstrap.
2. **`g-2`**: Añade un **gap** (espacio) de 2 unidades entre las columnas de la fila.
3. **`col-auto`**: Crea una columna que se ajusta automáticamente al tamaño de su contenido.
4. **`form-select`**: Aplica el estilo de Bootstrap a un menú desplegable (`select`).
5. **`form-control`**: Aplica el estilo de Bootstrap a los campos de entrada, dándoles una apariencia uniforme.
6. **`bg-danger`**: Aplica un fondo rojo al elemento (usado en el `select` para egresos).
7. **`bg-success`**: Aplica un fondo verde al elemento (usado en el `select` para ingresos).
8. **`border-danger`**: Aplica un borde rojo al campo (para egresos).
9. **`border-success`**: Aplica un borde verde al campo (para ingresos).
10. **`btn`**: Aplica los estilos básicos de botón de Bootstrap.
11. **`btn-danger`**: Aplica el color rojo al botón (usado para egresos).
12. **`btn-success`**: Aplica el color verde al botón (usado para ingresos).
13. **`bi bi-check-circle`**: Usa un ícono de Bootstrap para mostrar un check en el botón de envío.

Código del componente formulario.component.ts:

```
import { Component } from '@angular/core';
import { IngresoService } from '../ingreso/ingreso.service';
import { EgresoService } from '../egreso/egreso.service';
import { FormsModule } from '@angular/forms';
import { CommonModule } from '@angular/common';
import { Ingreso } from '../ingreso/ingreso.model';
import { Egreso } from '../egreso/egreso.model';

@Component({
  selector: 'app-formulario',
  standalone: true,
  imports: [FormsModule, CommonModule],
  templateUrl: './formulario.component.html',
  styleUrl: './formulario.component.css'
})
export class FormularioComponent {
  tipo:string="ingresoOperacion";
  descripcionInput:string | null = null;
```

```
  valorInput:number | null = null;

  constructor(private ingresoServicio: IngresoService,
    private egresoServicio:EgresoService
  ){}

  tipoOperacion(evento: Event){
    const target = evento.target as HTMLSelectElement;
    this.tipo = target.value;
  }

  agregarValor(){
    if(this.descripcionInput != null && this.valorInput != null)
      if(this.tipo == 'ingresoOperacion')
        this.ingresoServicio.ingresos.push(
            new Ingreso(this.descripcionInput, this.valorInput));
      else
        this.egresoServicio.egresos.push(
            new Egreso(this.descripcionInput, this.valorInput));
    else
      console.log('Introduce valores en descripcion y valor válidos');

  }
}
```

Código del componente ingreso.model.ts:

```
export class Ingreso{
    constructor(public descripcion:string, public valor:number){}
}
```

Código del componente ingreso.service.ts:

```
import { Injectable } from '@angular/core';
import { Ingreso } from './ingreso.model';

@Injectable({
  providedIn: 'root'
})
export class IngresoService {

  ingresos: Ingreso[]=[
    new Ingreso("Salario", 4000),
    new Ingreso("Venta Coche", 500)
```

```
  ];

  eliminar(ingreso:Ingreso){
      const indice: number = this.ingresos.indexOf(ingreso);
      this.ingresos.splice(indice,1);
  }
}
```

Código del componente ingreso.component.html:

```
<div>
    <h2 class="text-center text-primary">Ingresos</h2>

    <ul class="list-group mt-3">
        @for (ingreso of ingresos; track ingreso) {
        <li class="list-group-item d-flex justify-content-between align-items-center">
            <span class="flex-grow-1">{{ingreso.descripcion}}</span>
            <span class="text-success mx-3">
                {{ingreso.valor | currency:'MXN':'$':'1.2-2'}}
            </span>
            <button class="btn btn-sm btn-danger" (click)="eliminarRegistro(ingreso)">
                <i class="bi bi-x-circle"></i>
            </button>
        </li>
        }
    </ul>
</div>
```

Aquí tienes un resumen de las clases de **Bootstrap** utilizadas en el código:

1. **`text-center`**: Centra el texto horizontalmente.
2. **`text-primary`**: Aplica el color azul (primario) al texto.
3. **`list-group`**: Crea una lista con un estilo de grupo de elementos.
4. **`mt-3`**: Añade un margen superior de 3 unidades.
5. **`list-group-item`**: Estiliza los elementos de la lista con un diseño de tarjeta.
6. **`d-flex`**: Aplica el diseño **flexbox** para alinear los elementos en una fila.
7. **`justify-content-between`**: Distribuye los elementos de manera que haya espacio entre ellos.
8. **`align-items-center`**: Centra verticalmente los elementos dentro de la fila.
9. **`flex-grow-1`**: Permite que el elemento ocupe todo el espacio disponible.
10. **`text-success`**: Aplica el color verde al texto (usado para el valor).
11. **`mx-3`**: Añade márgenes izquierdo y derecho de 3 unidades.
12. **`btn`**: Aplica los estilos básicos de botón de **Bootstrap**.
13. **`btn-sm`**: Estiliza el botón en un tamaño pequeño.
14. **`btn-danger`**: Aplica el color rojo (peligro) al botón.

15. **`bi bi-x-circle`**: Usa un ícono de Bootstrap para mostrar una "X" en forma de círculo.

Código del componente ingreso.component.ts:

```
import { Component } from '@angular/core';
import { Ingreso } from './ingreso.model';
import { IngresoService } from './ingreso.service';
import { CommonModule } from '@angular/common';

@Component({
  selector: 'app-ingreso',
  standalone: true,
  imports: [CommonModule],
  templateUrl: './ingreso.component.html',
  styleUrl: './ingreso.component.css'
})
export class IngresoComponent {
  ingresos:Ingreso[]=[];

  constructor(private ingresoServicio: IngresoService){}

  ngOnInit(){
    this.ingresos = this.ingresoServicio.ingresos;
  }

  eliminarRegistro(ingreso: Ingreso) {
    this.ingresoServicio.eliminar(ingreso);
  }
}
```

Código del componente egreso.model.ts:

```
export class Egreso{
    constructor(public descripcion:string, public valor:number){}
}
```

Código del componente egreso.service.ts:

```
import { Injectable } from '@angular/core';
import { Egreso } from './egreso.model';

@Injectable({
  providedIn: 'root',
```

```
})
export class EgresoService {
  egresos: Egreso[] = [
    new Egreso('Renta Depto', 900),
    new Egreso('Ropa', 200)
  ];

  eliminar(egreso: Egreso) {
    const indice: number = this.egresos.indexOf(egreso);
    this.egresos.splice(indice, 1);
  }
}
```

Código del componente egreso.component.html:

```
<div>
    <h2 class="text-center text-danger">Egresos</h2>

    <ul class="list-group mt-3">
        @for (egreso of egresos; track egreso) {
        <li class="list-group-item d-flex justify-content-between align-items-center">
            <span class="flex-grow-1">{{egreso.descripcion}}</span>
            <span class="d-flex align-items-center mx-3">
                <span class="text-danger">
                    - {{egreso.valor | currency:'MXN':'$':'1.2-2'}}
                </span>
                <span class="badge bg-success ms-2">
                   {{calcularPorcentaje(egreso) | percent}}</span>
            </span>
            <button class="btn btn-sm btn-danger" (click)="eliminarEgreso(egreso)">
                <i class="bi bi-x-circle"></i>
            </button>
        </li>
        }
    </ul>
</div>
```

Código del componente egreso.component.ts:

```
import { Component, Input } from '@angular/core';
import { Egreso } from './egreso.model';
import { EgresoService } from './egreso.service';
import { CommonModule } from '@angular/common';
```

```
@Component({
  selector: 'app-egreso',
  standalone: true,
  imports: [CommonModule],
  templateUrl: './egreso.component.html',
  styleUrl: './egreso.component.css'
})
export class EgresoComponent {
  egresos:Egreso[]=[];
  @Input() ingresoTotal!:number;

  constructor(private egresoServicio: EgresoService){}

  ngOnInit(){
    this.egresos = this.egresoServicio.egresos;
  }

  eliminarEgreso(egreso: Egreso) {
    this.egresoServicio.eliminar(egreso);
  }

  calcularPorcentaje(egreso: Egreso){
    return egreso.valor/this.ingresoTotal;
  }
}
```

Resultado final:

Con esto terminamos la aplicación de Control de Presupuesto con Angular.

Saludos!

Ing. Ubaldo Acosta

Fundador de GlobalMentoring.com.mx