

# Module frontend with angular - class

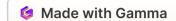


Today's agenda - Feb 5th 🔆

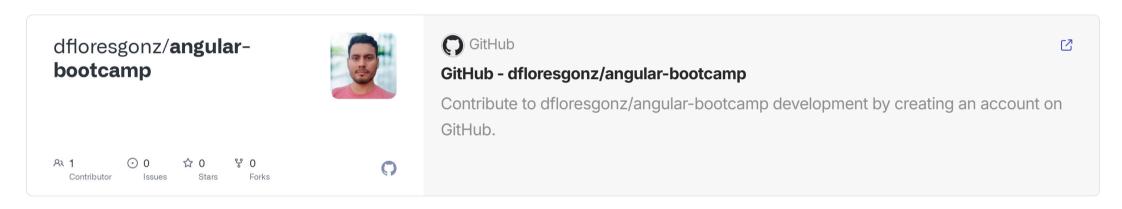


#### **Table of contents**

- Repository
- **Angular directives**
- <u>Angular pipes</u>
- **Angular routing**
- Let's review
- Thank you



### Repository



https://github.com/dfloresgonz/angular-bootcamp/







Permiten crear interfaces dinámicas y reutilizables

```
// Directiva *nglf para renderizado condicional
@Component({
 template: `
  <div *ngIf="isVisible">
   Este contenido aparece solo si isVisible es true
  </div>
  <!-- También podemos usar else -->
  <div *ngIf="isLoggedIn; else loginTemplate">
   Bienvenido usuario
  </div>
  <ng-template #loginTemplate>
   Por favor, inicia sesión
  </ng-template>
})
export class MiComponente {
 isVisible = true;
 isLoggedIn = false;
}
```

```
@Component({
 template: `
  <!-- Iteración básica -->
  {{ item }}
  <!-- Con índice y otras variables -->
  <div *ngFor="let user of users;</pre>
         let i = index;
         let isFirst = first;
         let isLast = last">
   Usuario {{i + 1}}: {{user.name}}
   <span *nglf="isFirst">(Primer usuario)</span>
   <span *ngIf="isLast">(Último usuario)</span>
  </div>
})
export class ListaComponent {
 items = ['Manzana', 'Naranja', 'Plátano'];
 users = [
  { name: 'Ana' },
  { name: 'Juan' },
  { name: 'María' }
];
}
```

```
// Directiva ngClass para manipular clases CSS
@Component({
 template: `
  <div [ngClass]="{'active': isActive,</pre>
            'error': hasError,
            'highlight': isHighlighted}">
   Contenido con clases dinámicas
  </div>
  <!-- ngStyle para estilos inline -->
  <div [ngStyle]="{'color': textColor,</pre>
            'font-size': fontSize + 'px',
           'background-color': bgColor}">
   Estilos dinámicos
  </div>
})
export class EstilosComponent {
 isActive = true;
 hasError = false;
 isHighlighted = true;
 textColor = 'blue';
 fontSize = 16;
 bgColor = '#f0f0f0';
```

// Componente con múltiples directivas

```
@Component({
 template: `
  <div class="task-list">
   <!-- Usar nglf para mostrar mensaje cuando no hay tareas -->
   <div *ngIf="tasks.length === 0" class="empty-message">
    No hay tareas pendientes
   </div>
   <!-- Lista de tareas con ngFor y múltiples directivas -->
   <div *ngFor="let task of tasks; let i = index"</pre>
      [ngClass]="{'completed': task.completed,
            'urgent': task.priority === 'alta'}"
      [ngStyle]="{'opacity': task.completed?'0.6':'1'}"
      appResaltado="lightyellow">
    <span class="task-number">{{i + 1}}.</span>
    <span class="task-title">{{task.title}}</span>
    <!-- Botones condicionales -->
    <button *nglf="!task.completed"</pre>
         (click)="completeTask(task)">
     Completar
    </button>
    <button *nglf="task.completed"</pre>
         (click)="undoComplete(task)">
     Deshacer
    </button>
   </div>
  </div>
})
export class TaskListComponent {
 tasks = [
  { title: 'Comprar víveres', completed: false, priority: 'normal' },
  { title: 'Pagar facturas', completed: false, priority: 'alta' },
  { title: 'Llamar al médico', completed: true, priority: 'alta' }
 ];
 completeTask(task: any) {
  task.completed = true;
 undoComplete(task: any) {
  task.completed = false;
}
@Component({
```

```
template: `
  <div [ngSwitch]="userRole">
   <div *ngSwitchCase=""admin"">
    Panel de administrador
   </div>
   <div *ngSwitchCase=""editor"">
    Panel de editor
   </div>
   <div *ngSwitchCase=""user"">
    Panel de usuario
   </div>
   <div *ngSwitchDefault>
    Acceso denegado
   </div>
  </div>
})
export class PanelComponent {
 userRole = 'admin';
}
```



{{ 'Hola' | miPipe }} <!-- Resultado: "aloH" -->



Los pipes son herramientas para transformar datos directamente en las plantillas. Se usan con el operador

```
{{ valor | uppercase }}: Convierte el texto a mayúsculas.

{{ valor | lowercase }}: Convierte el texto a minúsculas.

{{ valor | date:'dd/MM/yyyy' }}: Formatea una fecha.

{{ valor | currency:'USD' }}: Formatea un número como moneda.

{{ valor | json }}: Convierte un objeto a formato JSON.
```

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({ name: 'miPipe' })
export class MiPipe implements PipeTransform {
    transform(value: string): string {
      return value.split(").reverse().join(");
    }
}
```

```
@Component({
 template: `
  <!-- DatePipe - Formatea fechas -->
  Fecha: {{ fecha | date }}
  Fecha personalizada: {{ fecha | date:'dd/MM/yyyy' }}
  Hora: {{ fecha | date:'shortTime' }}
  <!-- UpperCasePipe y LowerCasePipe - Transformación de texto -->
  Mayúsculas: {{ nombre | uppercase }}
  Minúsculas: {{ nombre | lowercase }}
  <!-- DecimalPipe - Formato de números -->
  Número: {{ precio | number:'1.2-2' }}
  Moneda: {{ precio | currency: 'EUR' }}
  Porcentaje: {{ valor | percent:'1.0-2' }}
  <!-- SlicePipe - Recorta arrays o strings -->
  Texto recortado: {{ texto | slice:0:10 }}
  Array recortado: {{ array | slice:1:3 }}
  <!-- JsonPipe - Útil para debugging -->
  <{cobjeto | json }}</pre>
})
export class EjemploPipesComponent {
 fecha = new Date();
 nombre = 'Juan Pérez';
 precio = 23.5678;
 valor = 0.8543;
 texto = 'Este es un texto largo de ejemplo';
 array = [1, 2, 3, 4, 5];
 objeto = { nombre: 'Juan', edad: 25 };
}
```

```
// Pipe personalizado para filtrar una lista
@Pipe({
 name: 'filtrar'
})
export class FiltrarPipe implements PipeTransform {
 transform(items: any[], searchText: string): any[] {
  if (!items) return [];
  if (!searchText) return items;
  searchText = searchText.toLowerCase();
  return items.filter(item => {
  return item.nombre.toLowerCase().includes(searchText);
  });
 }
}
// Componente que usa el pipe personalizado
@Component({
 template: `
  <input [(ngModel)]="busqueda" placeholder="Buscar...">
  ul>
   {{ item.nombre }}
   })
export class ListaComponent {
 busqueda = ";
 items = [
  { nombre: 'Manzana' },
  { nombre: 'Banana' },
  { nombre: 'Naranja' }
 ];
}
```

# **Angular routing**



Mecanismo que permite a una app web dirigir y gestionar las vistas que se muestran al usuario en función de la URL o la ubicación dentro del sitio web.

- Basic routing
- Route parameters
- Lazy loading
- Route guards
- Child Routes





#### Let's review

Did you learn?







# Thank you

Thanks for paying attention!