Desarrollando un CRUD en ANGULAR

Parte 2
Routing

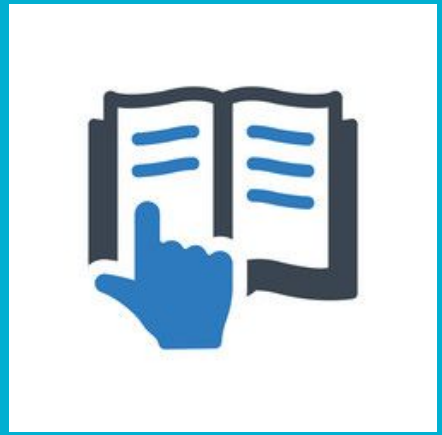
Mct. Esteban Calabria

Enunciado

En el laboratorio anterior logramos implementamos la funcionalidad de routing e hicimos de nuestra aplicación una PWA.

Ahora vamos a implementar la siguiente funcionalidad:

- Crear un formulario de contacto con Formularios Reactivos
- Validar el formulario
- Estilizar el formulario
- Obtener el contenido del formulario como un JSON



Antes de continuar debe saber lo siguiente

¿Qué son los formularios reactivos?

¿Que diferencia hay entre los Formularios Reactivos y los Template Driven Forms?

Antes de continuar ver el siguiente link

👉👉👉👉👉👉

📖📖📖

Angular ReactiveForms

https://angular.dev/guide/forms/reactive-forms

📖📖📖

Definir el formulario en la clase del componente

Paso 1

- Crear el formulario por código

⚠️⚠️⚠️

Los Formularios Reactivos se definen primero en el archivo del componente. Su desarrollo se centra en el código antes que en el template

⚠️⚠️⚠️

Importar....

Primero debemos importar:

ReactiveFormsModule

En el CrearContactoComponent

El ReactiveFormsModule en Angular permite crear forms de forma programática . Los forms se actualizan automáticamente en respuesta a cambios en los datos o en el estado.

```
import { Component } from '@angular/core';
import { ReactiveFormsModule } from '@angular/forms';
import { Router } from '@angular/router';

@Component({
  selector: 'app-crear-contacto',
  standalone: true,
  imports: [ReactiveFormsModule],
  templateUrl: './crear-contacto.component.html',
  styleUrl: './crear-contacto.component.css'
})
export class CrearContactoComponent {
```

Archivo crear-contacto.component.ts

El *ReactiveFormsModule* proporciona...

Soporte para Formularios

Reactivos: Permite la creación y manejo de formularios de manera reactiva y programática.

Validación de Formularios

Facilita la configuración de validaciones dinámicas y personalizadas en los formularios.

Control Preciso

Ofrece un control total sobre el estado y los datos de los formularios, permitiendo la vinculación directa con modelos de datos en el componente.

Componentes Reactivos:

Permite la integración de formularios con componentes y servicios de Angular para reactividad avanzada y manipulación de datos.

Crear form...

Ahora vamos a crear el formulario programáticamente antes de definir el template.

Observar:

- FormGroup
- FormBuilder
- Validators

```
export class CrearContactoComponent {

  form: FormGroup;

  constructor(private fb: FormBuilder,
    private router: Router) {

    this.form = this.fb.group({
      id: [null], // Campo opcional
      documento: [null, [Validators.required, Validators.min(1)]],
      nombreUsuario: ['', [Validators.required,
Validators.minLength(3)]],
      password: ['', [Validators.required, Validators.minLength(6)]],
      email: ['', [Validators.required, Validators.email]],
      fechaDeNacimiento: [null, Validators.required],
    });
  }
```

Archivo crear-contacto.component.ts

FormGroup

Angular agrupa y gestiona un conjunto de controles de formulario reactivos.

Agrupación de Controles

Maneja un conjunto de FormControls o FormArrays.

Estado Compartido

Controla el estado (válido, sucio, tocado, etc.) de los controles agrupados.

Validación en Grupo

Permite aplicar validaciones a nivel de grupo.

Control Anidado

Soporta formularios anidados y estructurados.

Acceso Centralizado

Proporciona un punto de acceso único a los valores y estados de todos los controles incluidos.

FormBuider

Facilita la Creación de Formularios

Proporciona métodos para construir formularios complejos de manera concisa.

Métodos Principales

- group: Crea un FormGroup para agrupar controles de formulario.
- control: Crea un FormControl para manejar un solo campo de formulario.
- array: Crea un FormArray para manejar una lista de controles de formulario.

Reducción de Código Boilerplate

Elimina la necesidad de escribir mucho código repetitivo al crear formularios.

Integración con Validaciones

Permite integrar validaciones directamente en la configuración del formulario.

Mejora la Legibilidad

Hace que el código de creación de formularios sea más claro y mantenible.

Definir el formulario en el template

Paso 2

- Mostramos y trataremos con el estado del formulario en el código del componente
- Verificaremos las validaciones

Crear template

Archivo crear-contacto.component.ts

```
<form [formGroup]="form" (ngSubmit)="onSubmit()">
    <!-- Documento -->
    <div>
      <label for="documento">Documento</label>
      <input id="documento" type="number" formControlName="documento" />
      <div *ngIf="form.get('documento').invalid && form.get('documento').touched">
        <small *ngIf="form.get('documento').hasError('required')">
          Documento es requerido.
        </small>
        <small *ngIf="form.get('documento').hasError('min')">
          Debe ser mayor que 0.
        </small>
      </div>
    </div>
    <!-- Otros Campos -->
    <button type="submit" [disabled]="form.invalid">Enviar</button>
</form>
```

Ahora vamos a crear el formulario en el template.

Con mensajes de validaciones

Directiva formGroup

Vincula el formulario HTML a un FormGroup en Angular.

Atributo formControName

Asocia un campo de entrada con un FormControl dentro de un FormGroup.

Form.get

Accede a un control específico dentro de un formulario reactivo. Devuelve un objeto de la clase FormControl

FormControl

Representa y gestiona el estado y valor de un campo individual en el formulario, encargándose de la validación mediante métodos como invalid y hasError, y del estado del campo, como touched para saber si ha sido tocado.

Agregar contacto al servicio

Vamos a agregar un método para agregar contacto al servicio...

```
export class ContactoService {

  private contactos: ContactoModel[] = [...]

//..

  agregarContacto(contacto: ContactoModel): void {
    this.contactos.push(contacto);
  }
```

Obtener el estado del formulario

Paso 3

Asociar el submit del form y mostrar el estado del mismo

Vamos a asocia el onSubmit del formulario para mostrar el estado del mismo que es un json.

⚠️No te olvides de verificar las validaciones ⚠️

```
export class CrearContactoComponent {


  onSubmit(){
    if (this.form.valid) {
      alert(JSON.stringify(this.form.value));
    }
  }
```

Validaciones

Validators en Angular son funciones que se utilizan para validar los datos en formularios reactivos, proporcionando reglas predefinidas para verificar la entrada del usuario.

Tipos de Validators

- required: Campo obligatorio.
- minLength: Longitud mínima del campo.
- maxLength: Longitud máxima del campo.
- pattern: Coincidencia con una expresión regular.
- email: Formato de correo electrónico válido.
- min y max: Valor mínimo y máximo numérico permitido

Estilizar los formularios

Paso 4

Vamos a utilizar las clases que agrega angular por defecto a los input para estilizar el formulario

Clases CSS

- ng-invalid: Aplica a cualquier control de formulario que no cumpla con las validaciones definidas.
- ng-valid: Aplica a los controles de formulario que cumplen con todas las validaciones.
- ng-touched: Aplica a los controles de formulario que han sido tocados (es decir, que han recibido foco y luego se les ha quitado el foco).
- ng-untouched: Aplica a los controles de formulario que aún no han sido tocados.
- ng-pristine: Aplica a los controles de formulario que no han sido modificados desde que se cargó el formulario.
- ng-dirty: Aplica a los controles de formulario que han sido modificados desde que se cargó el formulario.

A darle un poco de estilo

```
/* Estilo para campos
inválidos */
input.ng-invalid {
  border: 1px solid red;
}
```

Archivo crear-contacto.component.css

Sigamos Trabajando...