

20 - Formularios reactivos : FormBuilder

La clase FormBuilder nos permite crear objetos de la clase FormGroup y FormControl con una sintaxis más corta. Es decir que el resultado final será el mismo al que obtenemos con la sintaxis vista anteriormente.

Queda en decisión del desarrollador utilizar una u otra sintaxis.

Resolveremos un ejercicio de conceptos anteriores pero ahora con la sintaxis propuesta con la clase FormBuilder.

Problema

Confeccionar un formulario de contacto que permita cargar el nombre, mail y un mensaje. Implementar las siguientes validaciones a los controles del formulario:

- Los tres controles no pueden quedar vacíos.
- El nombre debe tener como mínimo 10 caracteres.
- El email ingresado debe ser correcto.
- El mensaje no puede superar los 500 caracteres.

Cuando se presione un botón mostrar un mensaje indicando si todos los controles se encuentran correctamente cargados.

- Crearemos primero el proyecto:

```
ng new proyecto015
```

- Modificamos la vista de la componente que muestra el formulario reactivo y eventualmente los mensajes de error de entrada de datos (app.component.html):

```

<form [formGroup]="formularioContacto" (ngSubmit)="submit()">
  <p>Nombre:
    <input type="text" formControlName="nombre" required>
    @if(this.formularioContacto.get('nombre')?.errors?.['required']
  ) {
    <span>(El nombre no puede quedar vacío)</span>
    }
    @if(this.formularioContacto.get('nombre')?.errors?.['minlength']
  ) {
    <span>(Debe tener como mínimo 10 caracteres)</span>
    }
  </p>
  <p>Mail:
    <input type="text" formControlName="mail">
    @if(this.formularioContacto.get('mail')?.errors?.['required'])
  {
    <span>(El mail no puede quedar vacío)</span>
    }
    @if(this.formularioContacto.get('mail')?.errors?.['email']) {
    <span>(El mail no es válido)</span>
    }
  </p>
  <p>Mensaje<br>
    <textarea rows="10" cols="70" formControlName="mensaje"></text
area>
  </p>
  @if(this.formularioContacto.get('mensaje')?.errors?.['required']
) {
  <div>(El mensaje no puede quedar vacío)</div>
  }
  @if(this.formularioContacto.get('mensaje')?.errors?.['maxlength']
) {
  <div>(Debe tener como máximo 500 caracteres)</div>
  }
  <button type="submit">Confirmar</button>
</form>
<div>{{resultado}}</div>
<router-outlet />

```

Analizaremos este archivo en conjunto luego de presentar 'app.component.ts'

- La clase asociada a la vista es (app.component.ts):

```

import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';
import { ReactiveFormsModule, Validators, FormBuilder, FormGroup }
  from '@angular/forms';
@Component({
  selector: 'app-root',
  imports: [RouterOutlet, ReactiveFormsModule],
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  resultado!: string;
  formularioContacto: FormGroup;

  constructor(private fb: FormBuilder) {
    this.formularioContacto = this.fb.group({
      nombre: ['', [Validators.required, Validators.minLength(10)]],
      mail: ['', [Validators.required, Validators.email]],
      mensaje: ['', [Validators.required, Validators.maxLength(500)]]
    });
  }

  submit() {
    if (this.formularioContacto.valid)
      this.resultado = "Todos los datos son válidos";
    else
      this.resultado = "Hay datos inválidos en el formulario";
  }
}

```

Lo primero que debemos hacer es importar la clase FormBuilder (ya no requerimos importar las clases FormGroup y FormControl):

```
import { ReactiveFormsModule, Validators, FormBuilder, FormGroup } from '@angular/forms';
```

Se inyecta al constructor un objeto de la clase FormBuilder:

```
constructor(private fb: FormBuilder) {
}
```

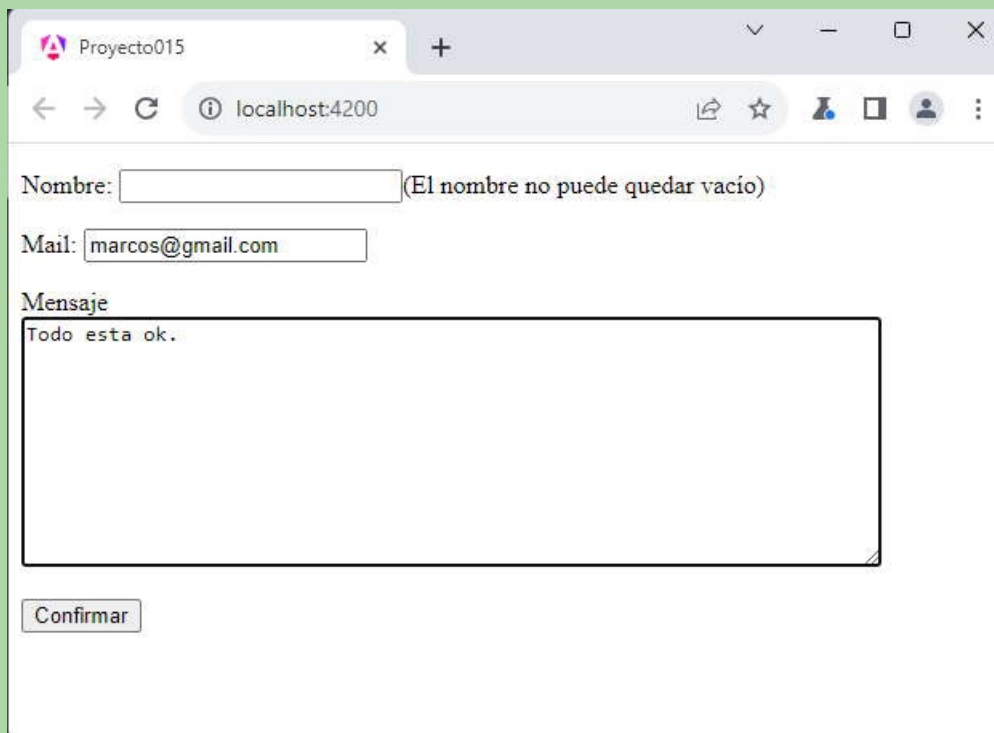
Llamamos al método 'group' de la clase FormBuilder y le pasamos un objeto literal indicando como atributos los nombres de los controles visuales definidos en HTML y como valor se le pasa un arreglo, cuyo primer elemento es el valor inicial del control y en el segundo elemento del arreglo pasamos otro arreglo con las funciones de validación que se le aplican a dicho campo:

```
this.formularioContacto = this.fb.group({
  nombre: ['', [Validators.required, Validators.minLength(10)]],
  mail: ['', [Validators.required, Validators.email]],
  mensaje: ['', [Validators.required, Validators.maxLength(500)]]
});
```

Tener que el resultado es exactamente el mismo si utilizamos la sintaxis vista anteriormente (debemos importar la clase FormControl en dicho caso):

```
this.formularioContacto = new FormGroup({
  nombre: new FormControl('', [Validators.required, Validators.minLength(10)]),
  mail: new FormControl('', [Validators.required, Validators.email]),
  mensaje: new FormControl('', [Validators.required, Validators.maxLength(500)])
});
```

Si ejecutamos la aplicación tenemos una interfaz similar a:



The screenshot shows a web browser window with the title 'Proyecto015' and the address bar displaying 'localhost:4200'. The form contains three fields: 'Nombre:' with a text input and a validation message '(El nombre no puede quedar vacío)', 'Mail:' with a text input containing 'marcos@gmail.com', and 'Mensaje' with a large text area containing 'Todo esta ok.'. Below the text area is a 'Confirmar' button.

Podemos probar esta aplicación en la web [aquí](#).

Retornar