

Arquitecturas de Software para Aplicaciones Empresariales

Validando un formulario mediante angular

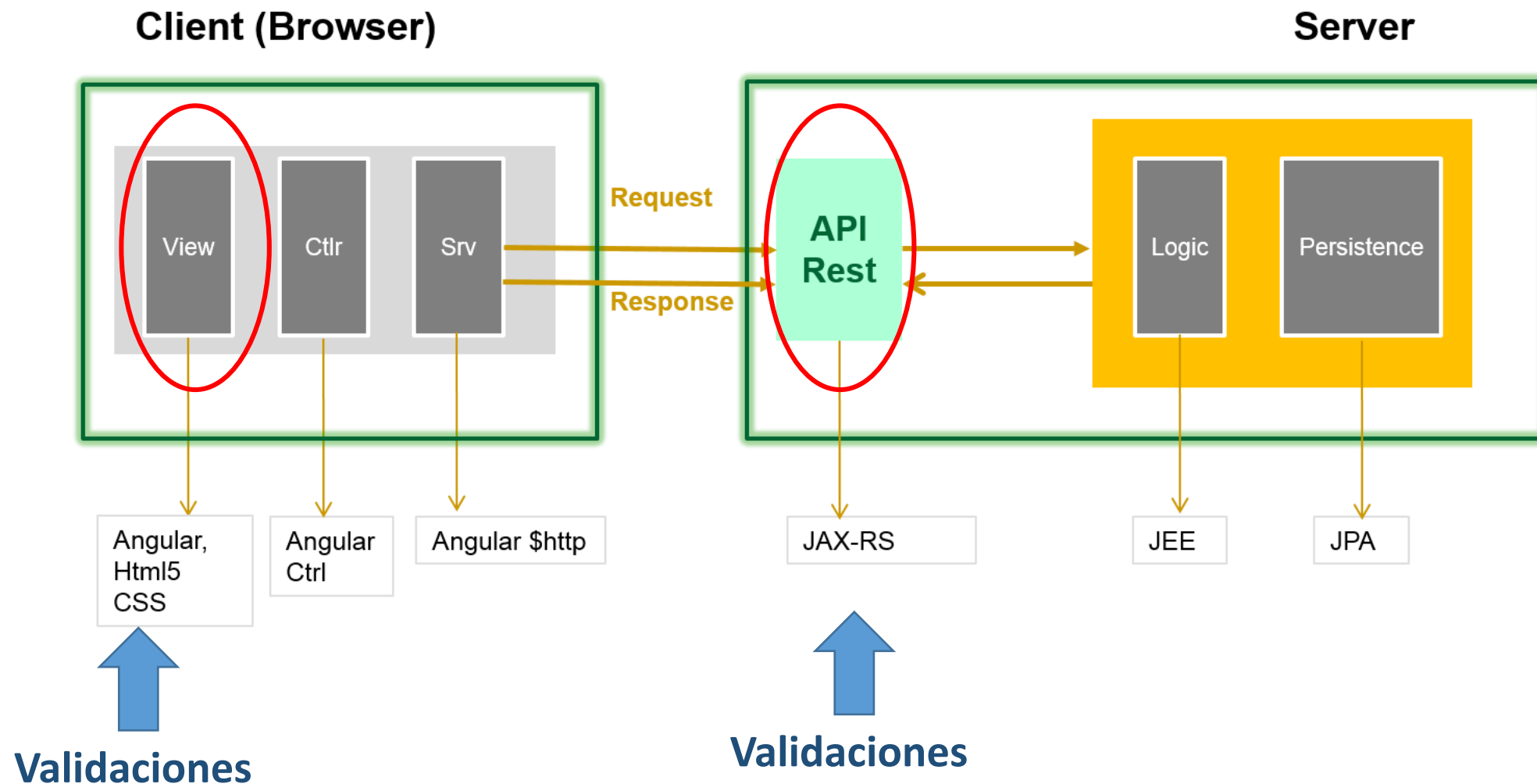


PROGRAMA DE INGENIERIA DE SISTEMAS

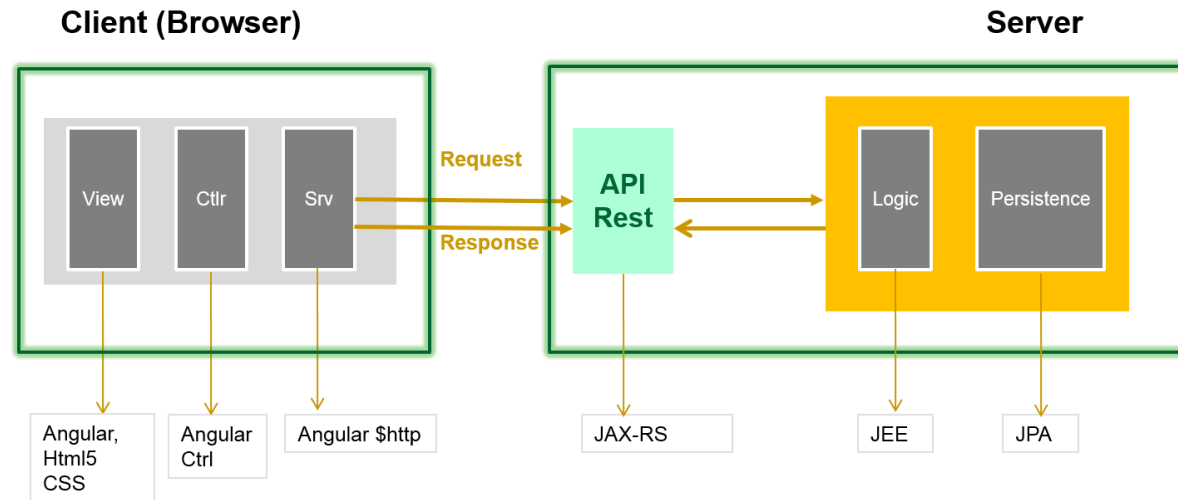
Ing. Daniel Eduardo Paz Perafán (danielp@Unicauca.edu.co)

Ing. Pablo A. Magé (pmage@Unicauca.edu.co)

Donde realizar las validaciones



Donde realizar las validaciones



- ❖ En la medida de lo posible se deben validar los datos ingresados en el FrontEnd y en el BackEnd.
- ❖ En el FrontEnd es necesario validar los datos de entrada. Cosas como formato, completitud de los campos, complejidad de las contraseñas, entre otras, son cosas que se pueden validar en el front. De esta forma se alivia el trabajo a los servidores.
- ❖ El BackEnd siempre debe desconfiar en los datos que el cliente le envía. En un caso extremo, un hacker podría implementar su propio FrontEnd y enviar datos incorrectos.

- ❖ Bean Validation es el estándar de facto para implementar la lógica de validación en el ecosistema Java
- ❖ JSR 380 es una especificación de la API de Java para la validación de beans, que forma parte de Jakarta EE y JavaSE. Esto asegura que las propiedades de un bean cumplen con criterios específicos, utilizando anotaciones como @NotNull , @Min y @Max .
- ❖ Esta versión requiere Java 8 o superior y aprovecha las nuevas funciones agregadas en Java 8, como las anotaciones de tipo y la compatibilidad con nuevos tipos como Optional y LocalDate .

Prueba de las validaciones en el backEnd

UNIVERSIDAD DEL CAUCA – FIET
DEPARTAMENTO DE SISTEMAS

The screenshot shows a REST client interface with a POST request to `localhost:8085/api/clientes/`. The request body is a JSON object with the following fields:

```
1 {
2   "nombre": "jua",
3   "apellido": "per",
4   "email": "juane@unicauca.edu.co",
5   "fechaRegistro": "2022-08-09T05:00:00.000+00:00"
6 }
```

The response status is **400 Bad Request**, with a time of 63 ms and a size of 413 B. The response body, shown in the 'Pretty' view, contains an array of validation errors:

```
1 {
2   "errors": [
3     "Campo 'nombre: ' la cantidad de caracteres del nombre debe estar entre 4 y 12",
4     "Campo 'apellido: ' la cantidad de caracteres del apellido debe estar entre 4 y 12"
5   ]
6 }
```

- ❖ Para validar formularios en html5 no hay que cargar librerías, ni crear reglas de validación.
- ❖ En esta última versión de este lenguaje existe una nueva propiedad que reciben los elementos de formulario llamada require.
- ❖ En HTML5 los nuevos atributos de las entradas como required y pattern usados en combinación con selectores de CSS, facilitan la programación y mejora de la interfaz del usuario.

Para agregar validación a un formulario basado en plantillas, agregue los mismos atributos de validación que haría si validara formularios en [HTML nativo](#)

Cada vez que cambia el valor de un elemento de formulario, angular ejecuta la validación y genera una lista de errores que da como resultado un estado **NO VÁLIDO, NULO, o VÁLIDO**.

Por cada input debemos agregar validadores

`required`

Debe haber un valor

`minlength`

Número de caracteres mínimo que debe tener un campo

`maxlength`

Número de caracteres máximo que puede tener un campo

En el input asociado al nombre agregamos dos validadores **required** y **minlength**.

```
<div class="form-group row">
  <label for="nombre" class="col-form-label col-sm-2">Nombre</label>
  <div class="col-sm-6">
    <input type="text" class="form-control" [(ngModel)]="cliente.nombre" name="nombre" required minlength="4">

  </div>
</div>
```

Para inspeccionar el estado de cada input, se exportan automáticamente los errores a una variable identificada con **#nombre**, de la siguiente forma

```
<input type="text" class="form-control" [(ngModel)]="cliente.nombre" name="nombre" required minlength="4" #nombre="ngModel">
```


Para determinar si existe errores necesitamos utilizar la directiva `*ngIf` y un contenedor

```
<div class="form-group row">  
  <label for="nombre" class="col-form-label col-sm-2">Nombre</label>  
  <div class="col-sm-6">  
    <input type="text" class="form-control" [(ngModel)]="cliente.nombre" name="nombre" required minlength="4" #nombre="ngModel">  
    <div class="alert alert-danger" *ngIf="">
```



Contenedor para determinar si existe o no un error



```
</div>  
</div>  
</div>
```

Contenedor para mostrar los errores

Para determinar si existe errores necesitamos utilizar la directiva *ngIf y un contenedor

```
<div class="form-group row">  
  <label for="nombre" class="col-form-label col-sm-2">Nombre</label>  
  <div class="col-sm-6">  
    <input type="text" class="form-control" [(ngModel)]="cliente.nombre" name="nombre" required minlength="4" #nombre="ngModel">  
    <div class="alert alert-danger" *ngIf="nombre.invalid && (nombre.dirty || nombre.touched)">
```



Contenedor para determinar si existe o no un error



```
</div>  
</div>  
</div>
```

Contenedor para mostrar los errores

```
<div class="alert alert-danger" *ngIf="nombre.invalid && (nombre.dirty || nombre.touched)">
```




Sirve para establecer si el campo es valido

```
<div class="alert alert-danger" *ngIf="nombre.invalid && (nombre.dirty || nombre.touched)">
```



Muestra el error a medida que vamos escribiendo

```
<div class="alert alert-danger" *ngIf="nombre.invalid && (nombre.dirty || nombre.touched)">
```



Muestra el error cuando nos salimos del foco del input, es decir pasamos de un input a otro input

Para determinar si existe errores necesitamos utilizar la directiva *ngIf y un contenedor

```
<div class="form-group row">
  <label for="nombre" class="col-form-label col-sm-2">Nombre</label>
  <div class="col-sm-6">
    <input type="text" class="form-control" [(ngModel)]="cliente.nombre" name="nombre" required minlength="4" #nombre="ngModel">
    <div class="alert alert-danger" *ngIf="nombre.invalid && (nombre.dirty || nombre.touched)">
      <div *ngIf="nombre.errors.required">
        Nombre es requerido
      </div>
      <div *ngIf="nombre.errors.minlength">
        Nombre debe tener al menos 4 caracteres
      </div>
    </div>
  </div>
</div>
```

Contenedor para determinar si existe o no un error

Contenedor para mostrar los errores

```
<div *ngIf="nombre.errors.required">  
  Nombre es requerido  
</div>
```

Sirve para establecer si el validador required esta en estado no valido.

```
<div *ngIf="nombre.errors.minlength">  
  Nombre debe tener al menos 4 caracteres  
</div>
```

Sirve para establecer si el validador `minlength` esta en estado no valido

Validación para el input asociado al campo nombre

```
<div class="form-group row">
  <label for="nombre" class="col-form-label col-sm-2">Nombre</label>
  <div class="col-sm-6">
    <input type="text" class="form-control" [(ngModel)]="cliente.nombre" name="nombre" required minlength="4" #nombre="ngModel">
    <div class="alert alert-danger" *ngIf="nombre.invalid && (nombre.dirty || nombre.touched)">
      <div *ngIf="nombre.errors.required">
        Nombre es requerido
      </div>
      <div *ngIf="nombre.errors.minlength">
        Nombre debe tener al menos 4 caracteres
      </div>
    </div>
  </div>
</div>
```

Validación para el input asociado al campo apellido

```
<div class="form-group row">
  <label for="apellido" class="col-form-label col-sm-2">Apellido</label>
  <div class="col-sm-6">
    <input type="text" class="form-control" [(ngModel)]="cliente.apellido" name="apellido" required minlength="4" #apellido="ngModel">
    <div class="alert alert-danger" *ngIf="apellido.invalid && (apellido.dirty || apellido.touched)">
      <div *ngIf="apellido.errors.required">
        Apellido es requerido
      </div>
      <div *ngIf="apellido.errors.minlength">
        Apellido debe tener al menos 4 caracteres
      </div>
    </div>
  </div>
</div>
```

Validación para el input asociado al campo email

```
<div class="form-group row">
  <label for="email" class="col-form-label col-sm-2">Email</label>
  <div class="col-sm-6">
    <input type="email" class="form-control" [(ngModel)]="cliente.email" name="email" required email #email="ngModel">
    <div class="alert alert-danger" *ngIf="email.invalid && (email.dirty || email.touched)">
      <div *ngIf="email.errors.required">
        Email es requerido
      </div>
      <div *ngIf="email.errors.email">
        Email debe tener un formato válido
      </div>
    </div>
  </div>
</div>
```


Ingresamos un cliente con todos los campos vacíos

Crear cliente

Nombre

Nombre es requerido

Apellido

Apellido es requerido

Email

Email es requerido

Crear

Para deshabilitar los botones mientras los campos sean inválidos debemos asignar una variable al formulario la cual permite determinar si el formulario tiene errores.

```
<form #clienteForm="ngForm">
```

En los botones de agregar y actualizar debemos agregar la directiva disable usando los []

```
[disabled]="!clienteForm.form.valid"
```

Si el formulario es invalido el botón esta deshabilitado, si el formulario es valido el botón esta habilitado

```
<div class="form-group row">
  <div class="col-sm-6">
    <button class="btn btn-primary" role="button" (click)='crearCliente()'
      *ngIf="!cliente.id else elseBlock" [disabled]="!clienteForm.form.valid">Crear</button>

    <ng-template #elseBlock>
      <button class="btn btn-primary" role="button" (click)='update()' [disabled]="!clienteForm.form.valid">Editar
    </ng-template>
  </div>
</div>
```

De forma predeterminada el botón esta deshabilitado

Crear cliente

Nombre

Nombre es requerido

Apellido

Apellido es requerido

Email

Email es requerido

Crear

Ingresamos un cliente donde el campo nombre únicamente tiene dos caracteres

Crear cliente

Nombre

da

Nombre debe tener al menos 4 caracteres

Apellido

Apellido es requerido

Email

Email es requerido

Crear

Ingresamos un cliente donde el campo nombre únicamente tiene dos caracteres, no ingresamos un apellido, e ingresamos un email que no sigue el formato correcto



The screenshot shows a web form titled "Crear cliente" with three input fields: "Nombre", "Apellido", and "Email". Each field has a validation error message displayed below it in a pink box. The "Nombre" field contains "sd" and the error is "Nombre debe tener al menos 4 caracteres". The "Apellido" field is empty and the error is "Apellido es requerido". The "Email" field contains "danielpunicauca.edu.co" and the error is "Email debe tener un formato válido". A blue "Crear" button is located at the bottom left of the form, highlighted with a red rectangle.

| Crear cliente | |
|--------------------------------------|---|
| Nombre | <input type="text" value="sd"/> Nombre debe tener al menos 4 caracteres |
| Apellido | <input type="text"/> Apellido es requerido |
| Email | <input type="text" value="danielpunicauca.edu.co"/> Email debe tener un formato válido |
| <input type="button" value="Crear"/> | |

Ingresamos un cliente correcto, lo cual activa el botón crear

Crear cliente

Nombre

Daniel Eduardo

Apellido

Paz Perafán

Email

danielp@unicauca.edu.co

Crear

Capturando el error 400

En la clase ClienteService.ts debemos capturar el error 400

```
create(cliente: Cliente) : Observable<Cliente> {  
    return this.http.post<Cliente>(this.urlEndPoint, cliente, {headers: this.httpHeaders}).pipe(  
        catchError(  
            e => {  
                if (e.status == 400) {  
                    return throwError(e);  
                }  
                console.log(e.error.mensaje);  
                swal.fire('Error al crear el cliente', e.error.mensaje, 'error');  
                return throwError(e);  
            })  
        );  
    }  
}
```


Capturando el error 400

En la clase FormComponent debemos almacenar los errores en un vector de cadenas

```
import { ClienteService } from './cliente.service';
import { Component, OnInit } from '@angular/core';
import { Cliente } from './cliente';
import { Router } from '@angular/router';
import swal from 'sweetalert2';

@Component({
  selector: 'app-form',
  templateUrl: './form.component.html',
  styleUrls: ['./form.component.css']
})
export class FormComponent implements OnInit {

  public cliente: Cliente = new Cliente();
  public titulo: string = 'Crear cliente';
  public errores: string[];

  constructor(private clienteService: ClienteService, private router: Router) { }

  ngOnInit(): void {
  }
}
```

Capturando el error 400

En la clase FormComponent debemos capturar los errores y guardarlos en el vector de errores

```
public crearCliente(): void{
  this.clienteService.create(this.cliente)
    .subscribe(
      response =>
      {
        console.log(response.nombre);
        this.router.navigate(['/clientes']);
        swal.fire('Nuevo cliente', `Cliente ${response.nombre} creado con éxito!`, 'success');
      },
      err => {
        thiserrores = err.error.errors as string[];
        console.error('Código del error desde el backend: ' + err.status);
        console.error(err.error.errors);
      }
    )
}
```

En el FormComponent.html debemos determinar si hay errores con la directiva ngIf

```
<ul class="alert alert-danger" *ngIf="errores?.length > 0">  
  
</ul>
```

En el FormComponent.html debemos recorrer la lista de errores con la directiva ngFor

```
<ul class="alert alert-danger" *ngIf="errores?.length > 0">  
  <li *ngFor="let err of errores">  
    {{ err }}  
  </li>  
</ul>
```

Desactivamos la validación del lado del cliente

```
<div class="form-group row">
  <label for="nombre" class="col-form-label col-sm-2">Nombre</label>
  <div class="col-sm-6">
    <input type="text" class="form-control" [(ngModel)]="cliente.nombre" name="nombre" #nombre="ngModel">
  </div>
</div>

<div class="form-group row">
  <label for="apellido" class="col-form-label col-sm-2">Apellido</label>
  <div class="col-sm-6">
    <input type="text" class="form-control" [(ngModel)]="cliente.apellido" name="apellido" #apellido="ngModel">
    <!-- required -->
  </div>
</div>

<div class="form-group row">
  <label for="email" class="col-form-label col-sm-2">Email</label>
  <div class="col-sm-6">
    <input type="email" class="form-control" [(ngModel)]="cliente.email" name="email" #email="ngModel">
  </div>
</div>
```

Probando únicamente las validaciones en el BackEnd

UNIVERSIDAD DEL CAUCA – FIET
DEPARTAMENTO DE SISTEMAS

Es posible que el FrontEnd realice unas validaciones antes de enviar los datos al servidor, y el backEnd realice unas validaciones de los datos que ingresan, y el backEnd encuentre inconsistencias.

- Campo 'apellido: ' la cantidad de caracteres del apellido debe estar entre 4 y 12
- Campo 'nombre: ' la cantidad de caracteres del nombre debe estar entre 4 y 12
- Campo 'email: ' Correo no valido

Crear cliente

Nombre

ss

Apellido

ss

Email

danielpunicauca.edu.co

Crear

Activando validaciones en el FrontEnd y en el backEnd

UNIVERSIDAD DEL CAUCA – FIET
DEPARTAMENTO DE SISTEMAS

Es posible que el FrontEnd realice unas validaciones antes de enviar los datos al servidor, y el backEnd realice unas validaciones de los datos que ingresan, y el backEnd encuentre inconsistencias.

- Campo 'nombre: ' la cantidad de caracteres del nombre debe estar entre 4 y 12

Crear cliente

Nombre

Daniel Eduardo

Apellido

Paz Perafán

Email

danielp@unicauca.edu.co

Crear

Considerar el código fuente del taller 1

Requisito funcional a implementar

Yo como coordinador de la maestría quiero gestionar un nuevo estudiante para poder matricularlo en un curso, gestionar sus solicitudes y asociarlo a un proyecto de investigación. Los atributos del estudiante son los definidos en el taller 1.

Condiciones de entrega

Realizar validaciones en el FrontEnd y BackEnd asociadas a correo electrónico y cantidad de caracteres máximo y mínimo para los nombres y apellidos.

Debe entregar un video de máximo 5 minutos donde se muestre:

- 1) (v 1.0) Explicación de la estructura del proyecto FrontEnd y BackEnd
- 2) (v 1.0) Explicación de los errores y fallos de validación implementados en el FrontEnd y en el BackEnd.
- 3) (v 1.0) Prueba de las funcionalidades de listar, crear, actualizar y eliminar.
- 4) (v 1.0) Prueba de los manejos de los mensajes de error o mensajes por fallos de validación implementados en el FrontEnd .
- 5) (v 1.0) Desactivar las validaciones en el FrontEnd y comprobar que el back está validando los datos y retornando el código 400.

El enlace de acceso al video debe ser subido a la plataforma de classroom en la asignación de la tarea correspondiente.

Muchas gracias
Preguntas

