<!--TIDSM-->

Forms y validaciones {

<Por="Equipo 2"/>

}

Formularios reactivos {

Son aquellos que no utilizan directivas de angular en html, en cambio usan objetos como FormControl, FormGroup y FormArray para manejar el estado y la validación de los campos del formulario. Esto nos da un mayor control en los formularios que requieren una validación avanzada, manejo de eventos y/o cambios dinámicos en el estado del forms.

}

Formularios por plantilla

{

Estos formularios son asíncronos. La lógica y estado de estos se maneja desde el html usando las directivas de angular tales como ngModel, ngForm, ngSubmit, etc. El manejar su estado y validaciones de esa manera los hace que sean más rápidos y sencillos de implementar.

}

<!--TIDSM-->

Validaciones {

<...="..."/>

}

Formularios por plantilla

{

Estos usan un tipo de validación en conjunto de directivas, lo que se hace es con NgModel, se hace la conexión de la variable en el archivo TypeScript y el Html.
En el que cada vez que cambia, se comprueba la validación y es cuando entra la directiva *ngIF para ver si es que existe un error, para evitar que, con cada pequeño cambio, salte el mensaje de error y no permita al usuario terminar, se usa “dirty” o “touched”; En donde el Touched salta cuando el usuario deja de seleccionar el input y pasa a otro, el dirty salta cuando un valor ya establecido sufre un cambio.

}

Validate {

Formularios reactivos

Los formularios reactivos hacen uso de sus validaciones desde el archivo TypeScript, Angular hace el cambio de estas, cada vez que se hace cambio en el valor del control.

Funciones de validador

Las funciones del validador pueden ser síncronas o asincrónicas.

}

Funciones de validador

{

Validadores síncronos

Los validadores síncronos son los que pueden validar algo “al momento”, los que pueden especificar algo sin necesidad de hacer operaciones. Como ver si un campo esta vacío o comprobar la sintaxis de un correo electrónico, esta se devuelve como segundo argumento cuando creas una instancia de un archivo.

Validadores asíncronos

Los validadores asíncronos son los que necesitan hacer operaciones, como por ejemplo hacer una petición, para ver si el correo ingresado está en uso. Esta devuelve un Promise u Observable que luego emite un conjunto de errores de validación, este se devuelve como tercer argumento.

- **Promise: El promise es un objeto que representa la finalización o el fracaso de operación asíncrona, está maneja un solo evento cuando una operación falla o se finaliza.**
- **Observable: Este es una secuencia de eventos que se envían a lo largo del tiempo.**

}

Funciones {

Clases de CSS de estado de control

Angular cuenta con clases de css, que reflejan los estados en los que se encuentra el campo, unas como: .ng-valid, .ng-invalid, .ng-touched, etc. Por ejemplo, en una respuesta invalida, al tener en el css la clase .ng-invalid, se aplicará el css que le hayas dado en el archivo.

Validación entre campos

La validación entre campos es cuando dependiendo de una respuesta en un campo, se decidirá la validación del otro campo.

}

Validación entre campos

{

Adición de validación cruzada a formularios reactivos

Lo que se hace, es que al momento de obtener los campos del form, se hace la función de comparar los campos, luego de esto se muestra un mensaje si es que es un error, el ejemplo proporcionado por angular, es que el nombre de un superhéroe y alias, no sea el mismo, una vez obtenido los dos campos, verifica que no sean igual, si es así, regresa el error.

}

Validación entre campos

{

Adición de validación cruzada a formularios basados en plantillas

Al igual manera que la adicción de validaciones, se hará uso de Directivas, con el provide de NG_VALIDATORS. Una vez creada la directiva y la función que valida, se usará directamente en el html. Sin olvidar poner una visualización amigable para los errores.

```
@Directive({
  selector: '[appForbiddenName]',
  providers: [
    {
      provide: NG_VALIDATORS,
      useExisting: ForbiddenValidatorDirective,
      multi: true,
    },
  ],
  standalone: true,
})
export class ForbiddenValidatorDirective implements Validator {
  @Input('appForbiddenName') forbiddenName = '';

  validate(control: AbstractControl): ValidationErrors | null {
    return this.forbiddenName
      ? forbiddenNameValidator(new RegExp(this.forbiddenName, 'i'))(control)
      : null;
  }
}
```

}

Directivas de Formularios en Angular{

En Angular, las directivas desempeñan un papel crucial en la manipulación de formularios. Algunas directivas comunes incluyen:

NgForm

Esta directiva se utiliza para rastrear el estado y la validez del formulario en su conjunto.

NgModel

Permite la vinculación bidireccional entre un control del formulario y una propiedad en el componente.

NgSubmit

Utilizada para capturar el evento de envío del formulario y ejecutar una función específica en el componente.

}

Puntos clave {

Angular proporciona clases de CSS específicas para reflejar el estado del control del formulario. Algunas de ellas son:

.ng-pristine

Indica que el control del formulario no ha sido tocado.

.ng-dirty

Indica que el control del formulario ha sido modificado.

.ng-valid

Indica que el valor del control cumple con todas las validaciones.

.ng-invalid

Indica que el valor del control no cumple con una o más validaciones.

}

{

Control de Errores en el Formulario:

En Angular, puedes acceder a los errores de un control del formulario mediante la propiedad errors. Esto permite personalizar la respuesta del usuario según los errores específicos, proporcionando mensajes de error más significativos.

Angular Form Builder:

El uso del constructor de formularios (FormBuilder) en Angular facilita la creación y configuración de formularios de manera programática. Permite definir controles, grupos y formularios completos de manera más eficiente.

Estilos Condicionales en Angular:

Además de las clases de estado del formulario, puedes aplicar estilos condicionales directamente en el HTML utilizando la directiva ngClass para mejorar la apariencia visual basada en el estado y la validez del formulario.

}

<!--TIDSM-->

Gracias {

<Por="Equipo 2"/>

}