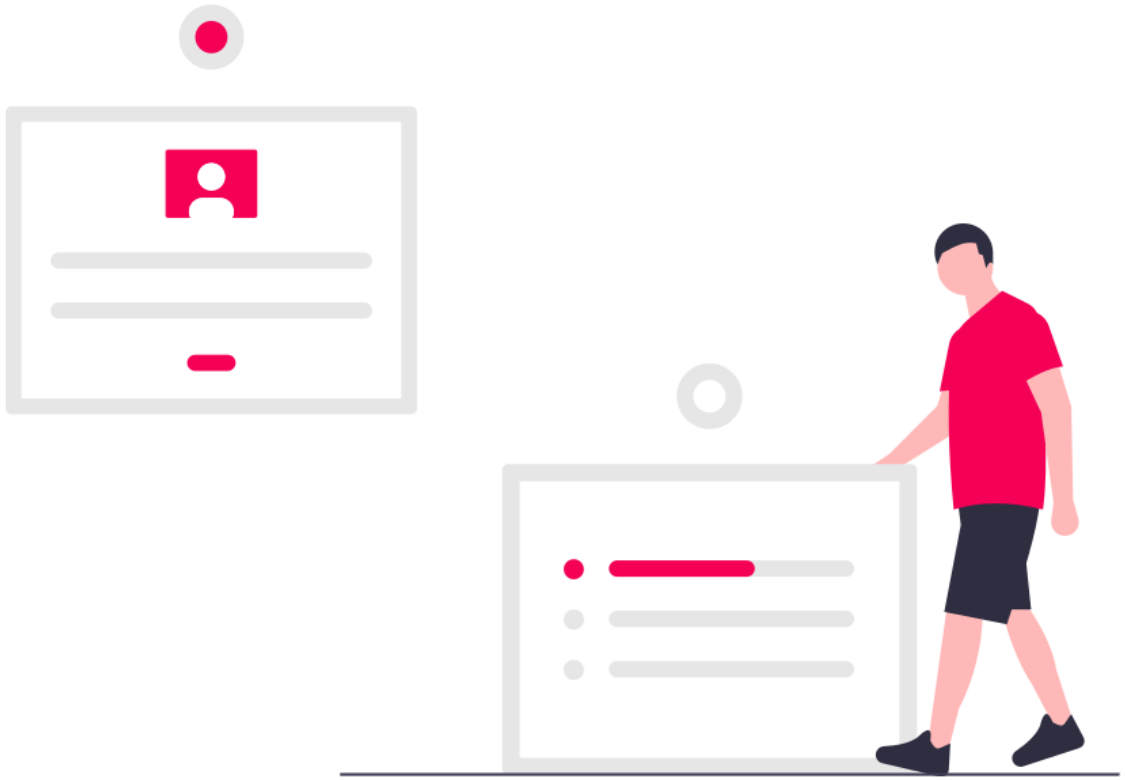
Formularios

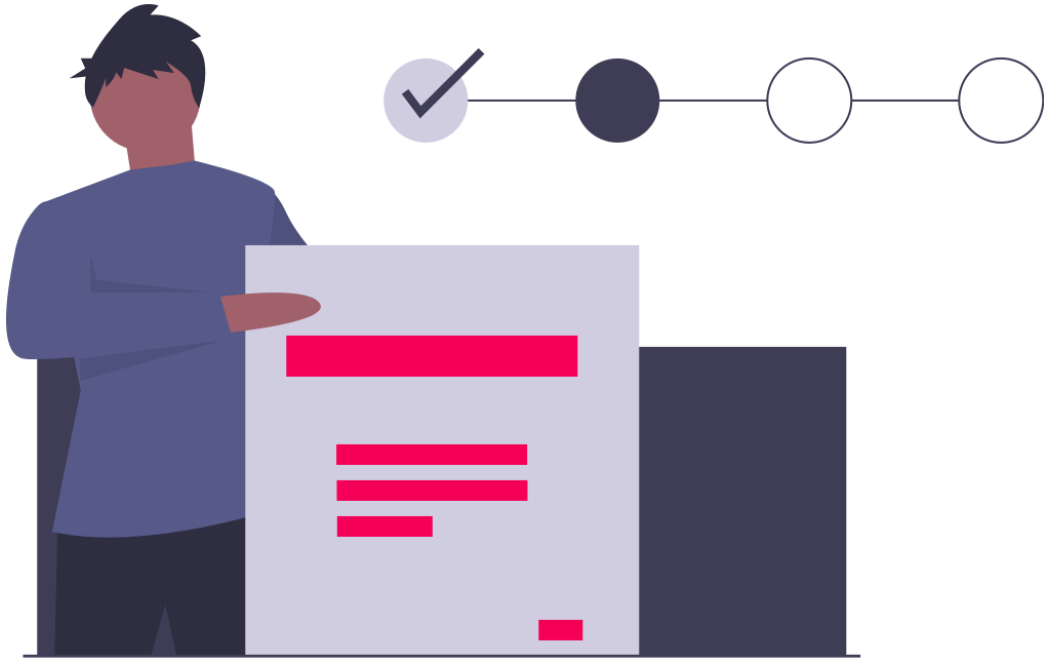
¿Qué son los Formularios?

Los formularios son el punto de entrada de datos a nuestras aplicaciones, se podría decir sin duda que es el elemento mas importante en el desarrollo de aplicaciones.
Constan de un numero de campos de datos que se solicitaran al usuario para su almacenamiento y posterior procesamiento.



Trabajar con Formularios en Angular

Cuando hablamos de trabajar con formularios en cualquier tecnología nos referimos al tratamiento de dato que le daremos a dichos formularios prestándole mucha mas atención a la parte de “**validación de formularios**”.



Existen dos maneras trabajar con formularios en Angular:

- Formularios reactivos.
- Formularios basados en plantillas.

Formularios basados en plantillas

son útiles para agregar un formulario sencillo a una aplicación, como un formulario de suscripción de lista de correo electrónico. Son fáciles de agregar a una aplicación, pero no escalan tan bien como los formularios reactivos.

Trabajar con formularios basados en plantillas:

1. Importamos el modulo de formularios de angular en el app.module.ts:

```
import { FormsModule } from '@angular/forms';
imports: [ BrowserModule, FormsModule ],
```

2. Creamos la plantilla de un formulario en algún componente:

```
<form>
        <label for="name">Name</label>
        <input type="text" class="form-control" id="name" required>
        <button type="submit" class="btn btn-success">Submit</button>
</form>
```

3. Usamos la directiva ngModel en nuestro formulario para enlazar datos bidireccionales (Two-way data binding):

```
<form #heroForm="ngForm">
<input type="text" class="form-control" id="name" required [(ngModel)]="model.name" name="name“
#name="ngModel">
```

En resumen:

El formulario Angular que se describe en esta página aprovecha las siguientes características de marco para proporcionar compatibilidad con la modificación de datos, la validación y mucho más:

- Una plantilla de formulario HTML angular.
- Una clase de componente de formulario con un decorador.@Component
- Controlar el envío del formulario mediante el enlace a la propiedad de evento.NgForm.ngSubmit
- Variables de referencia de plantilla como y .#heroForm#name
- [(ngModel)] sintaxis para el enlace de datos bidireccional.
- El uso de atributos para la validación y el seguimiento de cambios de elementos de formulario.name
- Propiedad de la variable de referencia en los controles de entrada para comprobar si un control es válido y mostrar u ocultar mensajes de error.valid
- Controlar el estado habilitado del botón Enviar enlazando a validez.NgForm
- Clases CSS personalizadas que proporcionan comentarios visuales a los usuarios sobre controles no válidos

Formularios reactivos

Los formularios reactivos proporcionan un enfoque basado en modelos para controlar las entradas de formulario cuyos valores cambian con el tiempo.

son más escalables, reutilizables y comprobables. Si los formularios son una parte clave de la aplicación o ya está utilizando patrones reactivos para compilar la aplicación, use formularios reactivos.

Trabajar con formularios reactivos:

1. Importamos el modulo de formularios reactivos de angular en el app.module.ts:

```
import { ReactiveFormsModule } from '@angular/forms';

@NgModule({
  imports: [
    // other imports ...
    ReactiveFormsModule
  ],
})
export class AppModule { }
```

2. Creamos la plantilla de un formulario en algún componente:

```
<form>
        <label for="name">Name</label>
        <input type="text" class="form-control" id="name" required>
        <button type="submit" class="btn btn-success">Submit</button>
</form>
```

3. Importamos los módulos que necesitamos para utilizar formularios reactivos en la lógica de nuestro componente, y creamos un FormGroup:

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

formContact: FormGroup;
```

4. Construimos un formulario, el cual esta compuesto por un FromControl que controlara cada campo input, este FromControl esta compuesto por un valor por defecto y un array de validadores:

```
this.formContact = new FormGroup({
  firstName: new FormControl(
     this.contact.firstName,
     [
        Validators.required
     ]
  )
});
```

5. En nuestro formulario creamos el atributo formGroup que será igual al formulario que creamos en la parte de la lógica, también asignaremos el atributo formControlName al input con el valor del FormControl que creamos dentro del FormGroup:

```
<form [formGroup]="formContact">
     <label for="name">Name</label>
     <input formControlName="firstName" type="text" class="form-control" id="name" required>
     <button type="submit" class="btn btn-success">Submit</button>
</form>
```

6. Finalmente capturamos el evento submit del formulario para procesar los datos:

```
<form [formGroup]="formContact" (ngSubmit)="updateContact()“>
```

5. Deshabilitamos el botón para enviar el formulario en caso de que este no sea valido:

```
<button [disabled]="formContact.invalid" type="submit" class="btn btn-success">Submit</button>
```

Diferencias claves

	REACTIVA	BASADO EN PLANTILLAS
Configuración (modelo de formulario)	Más explícito, creado en clase de componente	Menos explícito, creado por directivas
Modelo de datos	Estructurado	Desestructurado
Previsibilidad	Síncrono	Asincrónica
Validación del formulario	Funciones	Directivas
Mutabilidad	Inmutable	Mutables
Escalabilidad	Acceso a API de bajo nivel	Abstracción sobre las API

Referencias:

- Estructura de un formulario:
https://developer.mozilla.org/es/docs/Learn/HTML/Forms/How_to_structure_an_HTML_form

- Input tag:
https://www.w3schools.com/tags/tag_input.asp

- Validación de formularios:
https://developer.mozilla.org/es/docs/Learn/HTML/Forms/Validacion_formulario_datos

- Trabajar con formularios en Angular:
https://angular.io/guide/forms-overview