REPORTE ETAPA 3

REPORTE DE CODIGO Y ETAPA

Reporte Técnico: Módulos, Controladores, Formularios y Comunicación entre Componentes en Angular JS

LUCIO SAUCEDA LISSETTE

3.4 Evidencia de Aprendizaje: Aplicación en angular (reporte)

PARTICIPANTES:

Leidy Naomi

Lizandro Noel

Bestaide Abigail

Desiry Shekinna

Clemente Hernandez

1. Módulos en Angular

Los módulos en Angular son contenedores organizacionales que agrupan componentes, directivas, pipes y servicios relacionados funcionalmente. Cada aplicación Angular tiene al menos un módulo principal (AppModule) que sirve como punto de entrada.

Características Principales

- NgModule: Decorador que define la estructura del módulo
- Declaraciones: Incluye componentes, directivas y pipes que pertenecen exclusivamente al módulo
- Importaciones: Permite utilizar funcionalidades de otros módulos
- Exportaciones: Hace visibles ciertos elementos del módulo para otros módulos
- Proveedores: Define servicios disponibles dentro del ámbito del módulo

Controladores en Angular

En Angular, los controladores son representados por las clases de componentes que gestionan la lógica de presentación y la interacción con la vista. Cada componente actúa como controlador para su template correspondiente.

Responsabilidades

- Gestión del estado del componente
- Manejo de eventos del usuario
- Interacción con servicios
- Control del ciclo de vida del componente

Partes de una seta: ¿cómo están formadas?

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Formularios en Angular

Los formularios basados en plantillas utilizan directivas en el template HTML para definir la estructura y validación del formulario. La lógica reside principalmente en la vista.

Elementos Clave

- **ngForm: Directiva que convierte un formulario HTML estándar en un formulario Angular**
- **ngModel: Implementa two-way data binding entre el modelo y la vista**
- **Validadores integrados: required, minlength, maxlength, pattern**
- **Estados de validación: Control automático de estados como valid, invalid, pristine, dirty**

Ventajas

- **Rápida implementación para formularios simples**
- **Menos código TypeScript requerido**
- **Familiar para desarrolladores con experiencia en AngularJS**

Limitaciones

- **Dificultad para testing unitario**
- **Complejidad en formularios dinámicos**
- **Menor control programático**

Validaciones en Formularios

- Validaciones Sincrónicas: Verificaciones inmediatas durante la interacción del usuario
- Validaciones Asincrónicas: Validaciones que requieren consultas externas (API, base de datos)
- Validadores Personalizados: Lógica de validación específica del negocio

Estados de Validación

- valid/invalid: Indica si el campo cumple con todas las validaciones
- pristine/dirty: Refleja si el usuario ha modificado el valor
- touched/untouched: Indica si el campo ha recibido y perdido el foco

Características

Modelo explícito e inmutable
Validación síncrona y asíncrona
Testing más straightforward
Mayor control programático
Capacidad para crear formularios dinámicos

FormBuilder

FormBuilder es un servicio que proporciona métodos factory para crear instancias de FormControl, FormGroup y FormArray de manera más concisa y legible.

Ventajas

Sintaxis más compacta y mantenible

Reducción de código boilerplate

Mejor organización de la estructura del formulario

Facilita la creación de formularios complejos

Comunicacion entre componentes

01

4.1 Comunicación Padre → Hijo (@Input)
Definición
El decorador @Input permite que un componente padre pase datos a un componente hijo mediante property binding.

02

Mecanismo

- El componente hijo declara una propiedad con @Input
- El componente padre vincula datos a esta propiedad mediante sintaxis de corchetes
- Los cambios en el padre se propagan automáticamente al hijo
- Soporta detección de cambios automática

03

Casos de Uso

- Pasar datos de configuración
- Enviar estado del componente padre
- Propagar información de contexto
- Compartir datos maestros

Comunicación Hijo → Padre (@Output)

El decorador @Output permite que un componente hijo emita eventos hacia su componente padre mediante event binding.

Mecanismo

- El componente hijo declara un EventEmitter con @Output
- El componente hijo emite eventos mediante el método emit()
- El componente padre se suscribe al evento mediante sintaxis de paréntesis
- Comunicación basada en el patrón observer

Casos de Uso

- Notificar acciones del usuario
- Emitir resultados de operaciones
- Comunicar cambios de estado

Patrones de Comunicación Adicionales

Servicios Compartidos

- Utilización de servicios como bus de eventos
- Patrón Observer/Subject para comunicación cruzada
- Ideal para componentes no directamente relacionados

ViewChild y ContentChild

- Acceso directo a componentes hijos
- Comunicación imperativa
- Útil para casos específicos que requieren control directo

Consideraciones de Arquitectura

Mejores Prácticas

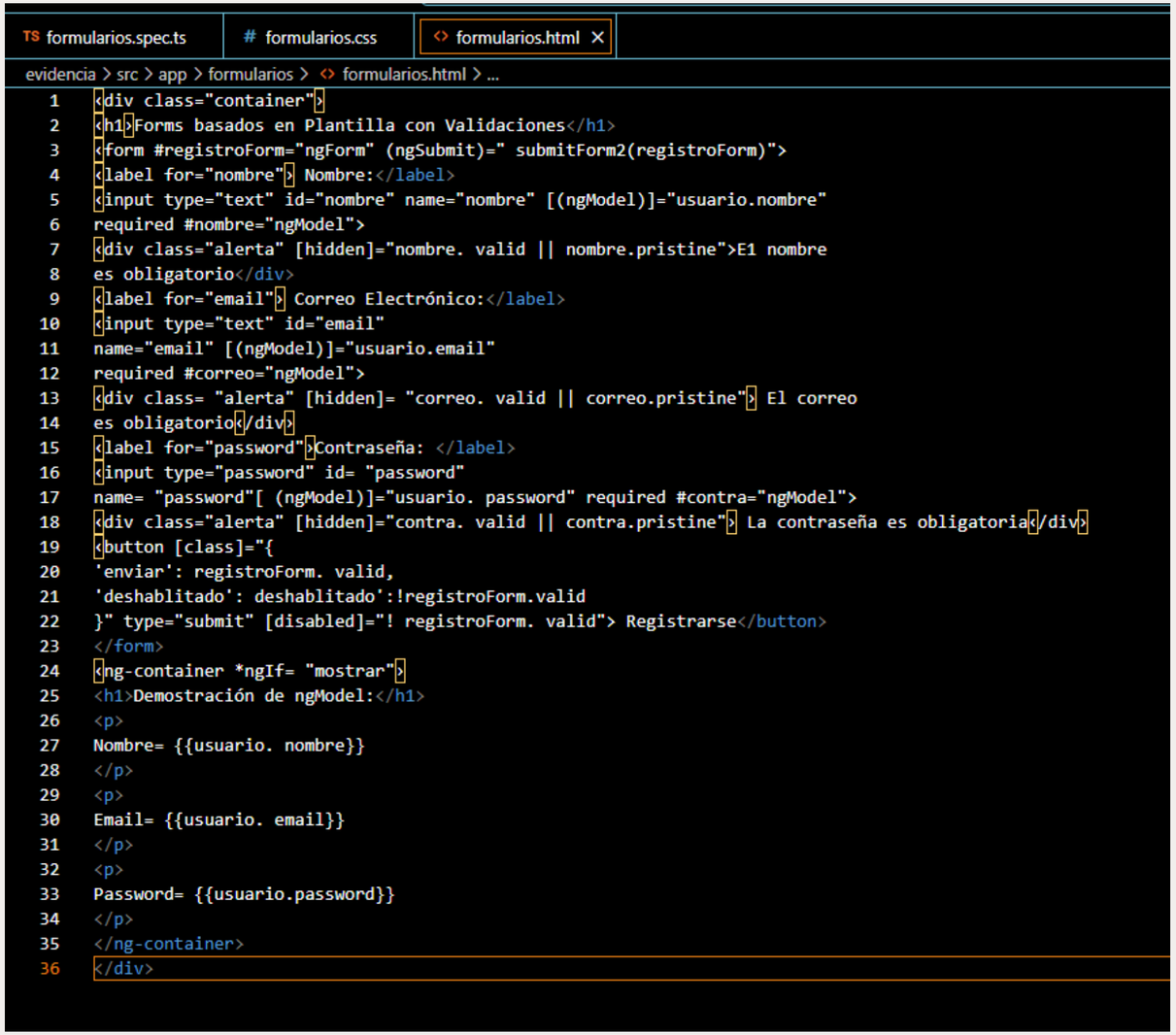
- **Utilizar formularios reactivos para formularios complejos**
- **Implementar validaciones personalizadas para reglas de negocio**
- **Estructurar módulos por dominio funcional**
- **Limitar la profundidad de comunicación entre componentes**
- **Preferir comunicación mediante servicios para componentes distantes**

Decisiones de Diseño

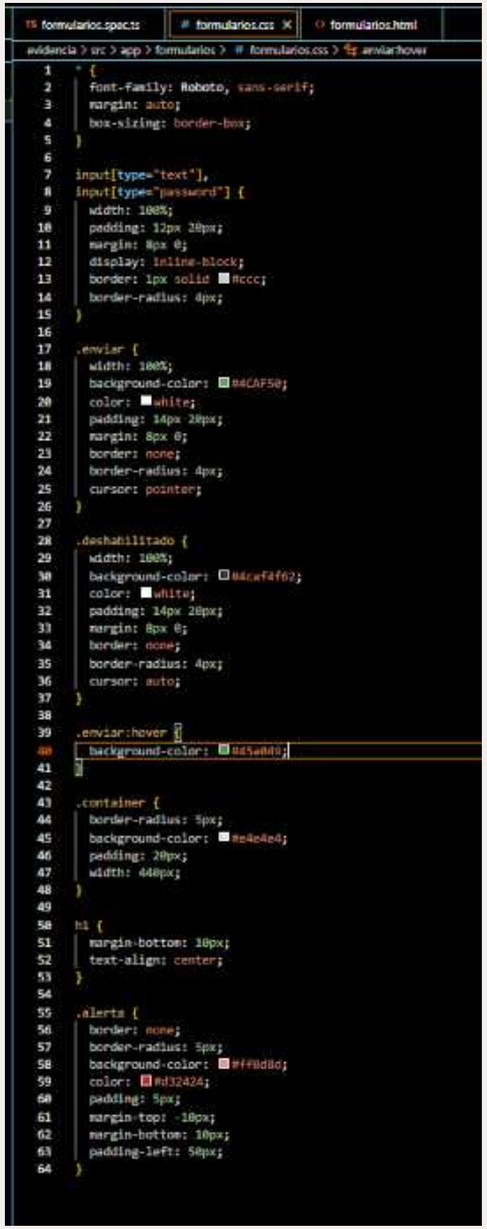
La elección entre formularios template-driven y reactivos depende de la complejidad del formulario, los requisitos de testing y la preferencia del equipo de desarrollo. La comunicación entre componentes debe diseñarse considerando la mantenibilidad y la claridad del flujo de datos.

Este componente se encarga de manejar un formulario en Angular, específicamente para validar y procesar datos de usuario. Cuando el usuario llena el formulario y lo envía, el componente verifica si toda la información es correcta antes de proceder.

Este archivo HTML implementa un formulario de registro con validaciones integradas usando el enfoque de Angular "Formularios Basados en Plantillas". El formulario incluye campos para nombre, email y contraseña, con validación visual y mensajes de error.



Este archivo CSS define todos los estilos visuales para el formulario de registro analizado anteriormente. Implementa un diseño limpio y profesional con feedback visual claro para el usuario, incluyendo estados de botones habilitados/deshabilitados y mensajes de error.



Muchas gracias

Presentado por Fernando Morales