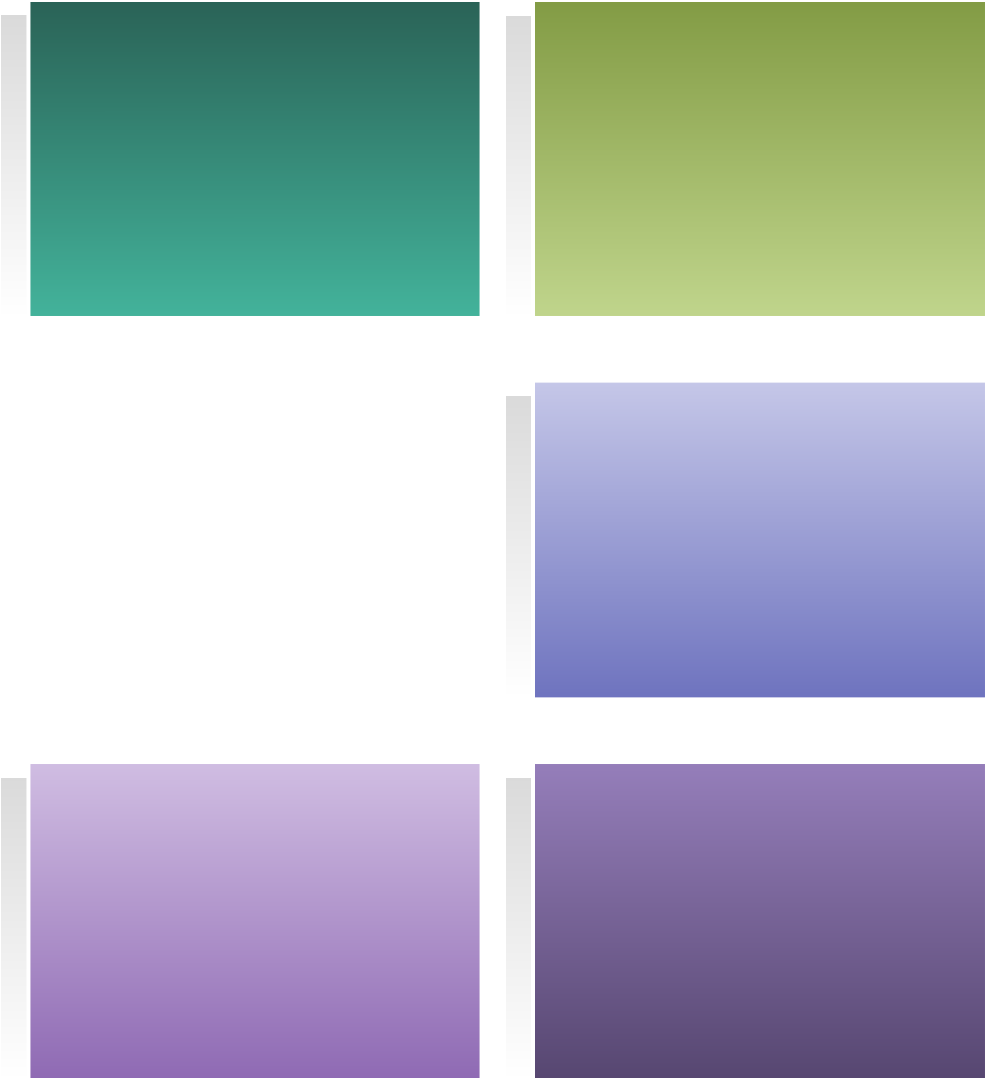
React Form + Validaciones zod



Contenidos

01 INTRO

02 CONCEPTOS BÁSICOS

03 GLOSARIO

04 ROADMAP

05 VENTAJAS - DESVENTAJAS

intro

Zod y React-Hook-Form son herramientas que permiten la validación de formularios en aplicaciones React de manera eficiente.

Zod se utiliza para definir esquemas de validación de datos, mientras que **React-Hook-Form proporciona** ganchos personalizados para manejar la lógica del formulario.

Zod

Es la Biblioteca de esquemas en TypeScript que facilita la definición de estructuras de datos y reglas de validación.

Su importancia radica en la capacidad de establecer un contrato claro sobre cómo deben ser los datos, lo que mejora la robustez y la mantenibilidad del código.

React-Form

Es la Biblioteca de gestión de formularios que utiliza hooks de React.

Su importancia radica en la eficiencia y la simplicidad que proporciona al gestionar el estado del formulario y los eventos, así como en la integración sin problemas con el ecosistema de React.

Ventajas y Desventajas

Parámetro	ZOD + REACT-HOOK-FORM	NO ZOD + NO REACT-FORM
Performance	✅ **Mayor eficiencia** en la reactividad de la web gracias a react-hook-form. ✅ **Menos renders innecesarios**.	☒ **Mayor carga manual** de la reactividad de la web. ☒ **Bastantes Renders innecesarios** al gestionar la reactividad.
Mantenibilidad	✅ **Mayor mantenibilidad,** centralización del manejo de formularios con react-hook-form.	☒ **Menos mantenible,** cada campo tiene su propia función de cambio y validación.
Usabilidad	✅ **Simplifica el manejo del formulario** ✅ **Interfaz más amigable**.	☒ **Requiere más código** para gestionar cambios y validaciones.
Escalabilidad	✅ **Estructura organizada y centralizada.**	☒ **Más complejo** a medida que crece el formulario.
Semántica	✅ Más semántico.	☒ Menos semántico.
Conexión con Backend	✅ **Facilita la conexión** al utilizar zod para definir **esquemas de validación compartidos** entre frontend y backend.	☒ **Puede requerir ajustes manuales** para alinear con las validaciones del backend.
Lógica de Negocio	✅ Útil para **formularios grandes o con lógica compleja, donde la conexión con el backend** es necesaria	✅ Útil para **formularios pequeños con lógica simple,** donde el rendimiento no es crítico.
Cuándo No Usar	⚠️ En formularios pequeños y simples donde la complejidad de react-hook-form sea excesiva.	⚠️ En formularios grandes o complejos donde la gestión del estado puede volverse confusa.

RoadMap

Paso 1

Instala las dependencias necesarias. Necesitarás react, react-dom, @hookform/resolvers, react-hook-form y zod.

pnpm install react react-dom @hookform/resolvers react-hook-form zod

Paso 2

Importa los módulos necesarios en tu archivo de componente.

```
import React from 'react';
import { useForm, Controller } from 'react-hook-form';
import { zodResolver } from '@hookform/resolvers/zod';
import { object, string, number } from 'zod';
```

RoadMap

Paso 3
(equivalente al validator)

Define tu esquema de validación con Zod.

```
const schema = object({
        name: string().nonempty({ message: 'El nombre es requerido' }).regex(/^[^\d]+$/, {
            message: 'El nombre no debe contener números' }),
        age: number().nonnegative({ message: 'La edad debe ser un número positivo' }),
        email: string().email({ message: 'El email no es válido' }),
        password: string().min(6, { message: 'La contraseña debe tener al menos 6
            caracteres' }), });
```

Paso 4

Configura useForm con el esquema de validación.

```
const { handleSubmit, control, formState: { errors }, reset } = useForm({
    resolver: zodResolver(schema)
  });
```

RoadMap

Paso 5

Crea tu formulario usando Controller para conectar los campos del formulario con react-hook-form.

```
return (

    <form onSubmit={handleSubmit(onSubmit)}>
        {formFields.map(({ name, label, type }, index) => (
            <div key={index}>
                <label>{label}</label>

                <Controller
                    name={name}
                    control={control}
                    defaultValue={type === 'number' ? 0 : ''}
                    render={({ field }) => (

                      <input
                        {...field}
                        type={type}
                        placeholder={`Enter your ${name.toLocaleLowerCase()}`}
                        value={field.value}
                        onChange={(e) => field.onChange(
                            type === 'number' ? parseInt(e.target.value, 10) : e.target.value)}
                      />
                    )}
                />
            </div>
        ))}

         <Button width="100%" height='30px' />

    </form>
);
```

Glosario

useState	Hook de React utilizado para gestionar el estado de componentes funcionales.
Hook	Función especial que permite acceder a características de React, sin necesidad de convertirse en componentes de clase
Zod	Resuelve los esquemas de validación de zod para ser utilizados con react-hook-form.
Controller	Componente de react-hook-form que se utiliza para vincular un controlador a un campo de entrada.
useForm	Hook de react-hook-form que inicializa y configura el formulario.
reset	Función de react-hook-form que reinicia los valores de los campos del formulario
react-hook-form	Biblioteca de React que proporciona herramientas para manejar formularios de manera eficiente.
zodResolver	Biblioteca de validación de esquemas para JavaScript y TypeScript.