

## PROFORMA DR. DSFDS

Lima, 20/06/2025

### DATOS DEL DOCTOR

**Nombre:** dsfds**CMP:** 32**Teléfono:** 2222222222222222**Dirección:** kikik mota**Tipo de entrega:** Entrega a domicilio**Observación:**

www

### MENSAJE DE GROBDI

Estimado Doctor,

Nos es grato dirigirnos a usted para presentarle la información detallada acerca de las formulaciones que ponemos a su disposición.

A continuación, encontrará un desglose completo de las cantidades, descripciones y precios correspondientes a las formulaciones cotizadas.

### DETALLE DE LA PROFORMA

Formulación	Cantidad	Precio Unitario	Subtotal
dfg	3	S/ 32.00	S/ 96.00
TOTAL:			S/ 96.00

**Gracias por su preferencia.****Nota:** Los precios incluyen el IGV. Condiciones de pago: Transferencia bancaria. Entrega: Según coordinación previa.

© 2025 - Todos los derechos reservados.

**Tipo de Pago:** Transferencia

Código de transacción: fdgdf3443

**Voucher:**

```

1 // Importar librerías
2 import React, { useState } from 'react';
3 import './styles.css';
4
5 // Definir componentes de UI
6 const Header = () => {
7   return (
8     <div>
9       <h1>Formulario de Registro</h1>
10     </div>
11   );
12 };
13
14 // Definir el controlador de datos
15 const Controller = {
16   // Método para validar los datos de entrada
17   validateForm: (formData) => {
18     // Validar nombre
19     if (!formData.name) {
20       return { valid: false, message: 'El nombre es obligatorio' };
21     }
22     // Validar correo electrónico
23     if (!formData.email) {
24       return { valid: false, message: 'El correo electrónico es obligatorio' };
25     }
26     // Validar contraseña
27     if (!formData.password) {
28       return { valid: false, message: 'La contraseña es obligatoria' };
29     }
30     // Validar confirmación de contraseña
31     if (formData.password !== formData.confirmPassword) {
32       return { valid: false, message: 'Las contraseñas no coinciden' };
33     }
34     // Si todos los campos son válidos
35     return { valid: true, message: '' };
36   },
37   // Método para guardar los datos en el servidor
38   saveForm: (formData) => {
39     // Aquí se simula la llamada al servidor
40     // En un entorno real, se usaría axios o fetch
41     // Simulamos un retraso de 2 segundos
42     setTimeout(() => {
43       // Guardamos los datos en un array local
44       Controller.formData.push(formData);
45       // Mostramos un mensaje de éxito
46       Controller.showMessage('Registro exitoso');
47     }, 2000);
48   },
49   // Método para mostrar un mensaje
50   showMessage: (message) => {
51     // Aquí se mostraría el mensaje en el UI
52     // Simulamos un retraso de 2 segundos
53     setTimeout(() => {
54       // Eliminamos el mensaje
55       Controller.showMessage('');
56     }, 2000);
57   },
58   // Estado para almacenar los datos de entrada
59   formData: [],
60 };
61
62 // Definir el componente principal
63 const App = () => {
64   // Estado para almacenar los datos de entrada
65   const [formData, setFormData] = useState({
66     name: '',
67     email: '',
68     password: '',
69     confirmPassword: ''
70   });
71   // Estado para almacenar el mensaje de éxito
72   const [message, setMessage] = useState('');
73   // Estado para almacenar si se está cargando
74   const [isLoading, setIsLoading] = useState(false);
75   // Función para manejar el cambio de un campo
76   const handleChange = (e) => {
77     const { name, value } = e.target;
78     setFormData({ ...formData, [name]: value });
79   };
80   // Función para manejar el envío del formulario
81   const handleSubmit = (e) => {
82     e.preventDefault();
83     // Validar los datos de entrada
84     const validation = Controller.validateForm(formData);
85     // Si no es válido, mostrar el mensaje de error
86     if (!validation.valid) {
87       Controller.showMessage(validation.message);
88       return;
89     }
90     // Si es válido, guardar los datos en el servidor
91     Controller.saveForm(formData);
92     // Limpiar el formulario
93     setFormData({
94       name: '',
95       email: '',
96       password: '',
97       confirmPassword: ''
98     });
99   };
100   // Renderizar el formulario
101   return (
102     <div>
103       <Header />
104       <form>
105         <input type="text" value={formData.name} onChange={handleChange} />
106         <input type="email" value={formData.email} onChange={handleChange} />
107         <input type="password" value={formData.password} onChange={handleChange} />
108         <input type="password" value={formData.confirmPassword} onChange={handleChange} />
109         <button type="submit">Registrar</button>
110       </form>
111       <div>{message}</div>
112     </div>
113   );
114 };
115
116 // Exportar el componente principal
117 export default App;

```