



REACT - SPRINGBOOT

Optimizando Backend

Package Entities

Entities Producto:

- Añadiremos atributo activo de tipo Boolean, aplicado para dar de baja un producto sin necesidad de eliminar físicamente el producto.
- Aplicaremos `@JsonIgnore` en la clase Categoría para el despliegue de productos integrando la categoría.

```

src > main > java > com > vivitasol > projectbackend > entities > J Producto.java > ...
11 import lombok.NoArgsConstructor;
12
13 @Data
14 @NoArgsConstructor
15 @AllArgsConstructor
16 @Entity
17 public class Producto {
18
19     @Id
20     @GeneratedValue(strategy=GenerationType.IDENTITY)
21     private Long id;
22     private String nombre;
23     private String descripcion;
24     private Long precio;
25     private Boolean activo = true;
26
27     @ManyToOne(fetch = FetchType.EAGER)
28     @JoinColumn(name="categoria_id")
29     private Categoria categoria;
30 }
31
32
33
34

```

```

src > main > java > com > vivitasol > projectbackend > entities > J Categoria.java > ...
12 import com.fasterxml.jackson.annotation.JsonIgnore;
13
14
15
16 @Data
17 @NoArgsConstructor
18 @AllArgsConstructor
19
20 @Entity
21 public class Categoria {
22
23     @Id
24     @GeneratedValue(strategy = GenerationType.IDENTITY)
25     private Long id;
26     private String nombre;
27
28     @OneToOne(mappedBy = "categoria")
29     @JsonIgnore
30     private List<Producto> productos;
31
32
33
34 }

```

Package Services:

- Definimos método desactivar, implementado en la clase del servicio.

```

src > main > java > com > vivitasol > projectbackend > services > J ProductoServices.java > ...
1 package com.vivitasol.projectbackend.services;
2
3 import java.util.List;
4 import com.vivitasol.projectbackend.entities.Producto;
5
6 public interface ProductoServices {
7
8     Producto crear(Producto producto);
9     Producto obtenerId(Long id);
10    List<Producto> listarTodas();
11    void eliminar(Long id);
12    Producto actualizar(Long id, Producto productoActualizado);
13    Producto desactivar(Long id);
14 }
15

```

```

src > projectbackend > services > J ProductoServicesImpl.java > Language Support for Java(TM) by Red Hat, Inc.
public class ProductoServicesImpl implements ProductoServices {
    ...
    @Override
    public Producto desactivar(Long id){
        Producto producto = obtenerId(id);
        producto.setActivo(activo:false);
        return productoRepositories.save(producto);
    }
}

```

Package RestControllers:

- Implementamos petición patch() para modificar el estado del producto a través del servicio.

En la base de datos cargamos una nueva data: **base.sql**

```

src > projectbackend > controllers > J ProductoRestControllers.java > Language Support for Java(TM) by Red Hat, Inc.
14 public class ProductoRestControllers {
15
16     ...
17     @PatchMapping("/{id}/desactivar")
18     public ResponseEntity<Producto> desactivar(@PathVariable Long id) {
19         return ResponseEntity.ok(productoServices.desactivar(id));
20     }
21 }

```

Optimizando FrontEnd

Componente App.jsx: agregamos id en el path de editar.

```
2  return (
3    <Router>
4      <Routes>
5        <Route path="/" element={<Home />} />
6        <Route path="/contacto" element={<Contacto />} />
7        <Route path="/inventario" element={<Inventario />} />
8        <Route path="/productos" element={<Productos />} />
9        <Route path="/crear-producto" element={<CrearProducto />} />
10       <Route path="/editar-producto/:id" element={<EditarProd />} />
11     </Routes>
12   </Router>
13 )
14 
```

Componente productos.jsx:

- Mostramos los productos del backend. Al desactivar o dar de baja un producto, el despliegue se actualiza dentro del mismo componente. Se modifíco el componente agregando el método cargarProductos(), el cual es llamado a través de useEffect().
- Se ha creado el método handleDesactivar(), el cual modifica el estado del producto y actualizando el componente.

```
src > componentes > Productos > Produkto.jsx > Produkto.js > Produkto.js
1  import './Productos.css';
2
3  export function Produkto() {
4
5    const [productos, setProductos] = useState([]);
6
7    const cargarProductos = async () => {
8      try {
9        const response = await fetch('http://localhost:8080/api/productos');
10       if (!response.ok) {
11         throw new Error('Error en la respuesta del servidor');
12       }
13
14       const data = await response.json();
15       console.log("Respuesta del backend:", data);
16
17       const productosNormalizados = data.map(p => ({
18         ...p,
19         activo: p.activo === true || p.activo === 'true',
20       }));
21
22       setProductos(productosNormalizados);
23     } catch (error) {
24       console.error('Error al obtener los productos:', error);
25     }
26   };
27
28   useEffect(() => {
29     cargarProductos();
30   }, []);
31
32   const handleDesactivar = (id, nombre) => {
33     const respuesta = window.confirm(`Estás seguro de desactivar el producto "${nombre}"?`);
34     if (respuesta) {
35       fetch(`http://localhost:8080/api/productos/${id}/desactivar`, {
36         method: 'PATCH'
37       })
38       .then(response => {
39         if (!response.ok) {
40           throw new Error('Error al desactivar el producto');
41         }
42         return response.json();
43       })
44       .then(data => {
45         console.log('Producto desactivado:', data);
46         alert('Producto desactivado exitosamente');
47         cargarProductos();
48       })
49       .catch(error => {
50         console.error('Error al desactivar:', error);
51         alert('Error al desactivar el producto');
52       });
53     }
54   };
55
56   return (
57     <>
58   );
59 }
```

```
useEffect(() => {
  cargarProductos();
}, []);

const handleDesactivar = (id, nombre) => {
  if (window.confirm(`Estás seguro de desactivar el producto "${nombre}"?`)) {
    fetch(`http://localhost:8080/api/productos/${id}/desactivar`, {
      method: 'PATCH'
    })
    .then(response => {
      if (!response.ok) {
        throw new Error('Error al desactivar el producto');
      }
      return response.json();
    })
    .then(data => {
      console.log('Producto desactivado:', data);
      alert('Producto desactivado exitosamente');
      cargarProductos();
    })
    .catch(error => {
      console.error('Error al desactivar:', error);
      alert('Error al desactivar el producto');
    });
  }
};

return (
  <>
</tr>
</thead>
<tbody>
  {productos.map((prod) => (
    <tr key={prod.id}>
      <td>{prod.activo ? 1 : 0.5}</td>
      <td>{prod.id}</td>
      <td>{prod.nombre}</td>
      <td>{prod.descripcion}</td>
      <td>{prod.precio}</td>
      <td>
        {prod.activo ? (
          <Link className="btn btn-outline-primary"
            style={{ fontSize: '13px' }}
            to={`/editar-producto/${prod.id}`}
            onClick={() => handleDesactivar(prod.id, prod.nombre)}
          > Editar Producto </Link>
        ) : (
          <button className="btn btn-outline-secondary"
            style={{ cursor: 'not-allowed',
              pointerEvents: 'none',
              disabled: true
            }}
            onClick={() => handleDesactivar(prod.id, prod.nombre)}
          > Editar Producto </button>
        )}
      </td>
    </tr>
  ))
}
```

```
<td>
  {prod.activo ? (
    <Link className="btn btn-outline-primary"
      style={{ fontSize: '13px' }}
      to={`/editar-producto/${prod.id}`}
      onClick={() => handleDesactivar(prod.id, prod.nombre)}
    > Editar Producto </Link>
  ) : (
    <button className="btn btn-outline-secondary"
      style={{ cursor: 'not-allowed',
        pointerEvents: 'none',
        disabled: true
      }}
      onClick={() => handleDesactivar(prod.id, prod.nombre)}
    > Editar Producto </button>
  )}
</td>

```

Componente crearProducto:

En este componente debemos codificar el acceso a RestController de Categorías y de Productos.

- Se define const de tipo producto, por medio de useState(), la cual será modificada una vez que el objeto se crea en el formulario.
- Se implementa método que nos devuelve las categorías registradas en el backend.

```
componentes > CrearProd > CrearProducto.jsx > CrearProducto > useEffect() callback >
  export function CrearProducto() {
    const [loading, setLoading] = useState(false);
    const [error, setError] = useState(null);
    const [loadingCategorias, setLoadingCategorias] = useState(true);

    useEffect(() => {
      fetch(`${API_BASE_URL}/categorias`)
        .then(response => {
          if (!response.ok) {
            throw new Error('Error al cargar categorías');
          }
          return response.json();
        })
        .then(data => {
          setCategorias(data);
          setLoadingCategorias(false); // ← Termina la carga exitosa
        })
        .catch(error => {
          console.error('Error al obtener las categorías:', error);
          setError('No se pudieron cargar las categorías');
          setLoadingCategorias(false); // ← Termina la carga con error
        });
    }, []);
```

- Método handleChange(): permite capturar los cambios generados en el formulario;

```
const handleChange = (e) => {
  const { name, value } = e.target;
  setProducto(prev => ({
    ...prev,
    [name]: value
  }));
};
```

- e.target: input que genera el evento.
- {name, value}: extrae el name y el value del input.
- setProducto(): actualiza el estado de un objeto, reemplaza el objeto con los valores ingresados en el formulario.

- Método handleSubmit(): se ejecuta cuando se envía el formulario.

```
const handleSubmit = async (e) => [
  e.preventDefault(),
  setError(null),
  setLoading(true),
];
```

- controla que la página no se recargue al enviar el formulario.
- Limpia cualquier error anterior y se activa el estado de "cargando datos".

```
try {
  const response = await fetch(`${API_BASE_URL}/productos`, {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify(productoParaEnviar)
  });

  if (!response.ok) {
    const errordata = await response.json().catch(() => ({}));
    throw new Error(errordata.message || 'Error al crear el producto');
  }

  navigate('/inventario', {
    state: { message: 'Producto creado exitosamente' }
  });

} catch (err) {
  setError(err.message || 'No se pudo crear el producto');
} finally {
  setLoading(false);
}
```

- Petición POST.
- Control de excepciones.
- Se redirecciona a '/inventario' por medio de navigate.

Posteriormente se implementa return() con la creación del formulario.

Implementando return():

```
src > componentes > CrearProd > CargarCategorias.jsx > CrearProducto.jsx
8  export function CrearProducto() {
93
94    return (
95      <>
96        <Navbar />
97
98        <div className="crear-producto-container">
99          <div className="form-card">
100            <h2>Crear Producto</h2>
101            {error && (
102              <div className="error-message">
103                {error}
104                <button onClick={() => setError(null)} className="error-close">x</button>
105              </div>
106            )}
107            {loadingCategorias ? (
108              <div className="loading">Cargando categorías...</div>
109            ) : (
110
111              <form onSubmit={handleSubmit}>
112                <div className="form-row">
113                  <div className="form-group">
114                    <label htmlFor="nombre">
115                      Nombre del producto <span className="required">*</span>
116                    </label>
117                    <input

```

- Botón close (x).
- onSubmit={handleSubmit}
- onChange = {handleChange}
- Cargar categorías: categorías.map
- Button type="submit"
- {handleCancel}

Mi Página

Home Inventario Contacto

Inventario de productos

Id Producto	Nombre	Descripción	Precio	Editar Producto	Desactivar/Eliminar
1	producto1	producto de alta calidad	1000	<button>Editar Producto</button>	<button>Desactivar</button>
2	producto2	producto de alta calidad	2000	<button>Editar Producto</button>	<button>Desactivado</button>
3	producto3	producto de alta calidad	1500	<button>Editar Producto</button>	<button>Desactivar</button>
4	producto4	producto de alta calidad	1800	<button>Editar Producto</button>	<button>Desactivar</button>

Crear Producto

Mi Página

Home Inventario Contacto

Crear Producto

Nombre del producto *

Precio *

Descripción *
0/500 caracteres

Categoría *

Crear Producto Cancelar