

## Introducción al desarrollo Web con Ant Design Table y getProducts

### *git branch getProducts*

Como primer paso establece el endpoint **PRODUCTS: "api/products"**, en tu archivo de constantes.

En la carpeta de **services**, crea el archivo **products.js** para crear las solicitudes al endpoint establecido en la instrucción anterior.

Realiza los imports necesarios (axios, ENV) y construye la primera solicitud con el método get para obtener todos los productos.

```
src > services > .js products.js > ...
1  import axios from 'axios';
2  import { ENV } from '../utils/constants';
3
4
5  export const getProducts = async () => {
6    try {
7      const response = await axios.get(`${ENV.API_URL}/${ENV.ENDPOINTS.PRODUCTS}`);
8      return response.data;
9    } catch (error) {
10     console.error('Error al obtener los productos:', error);
11     throw error;
12   }
}
```

Para continuar, en tu componente Productos (pages/index.jsx) implementa la vista, yo utilice el componente Table de Antd, ve a su documentación y observa los imports necesarios para una tabla de estructura básica con un select y paginación incluida. Importa el componente **import { Divider, Table } from 'antd';** e incluye una constante columns que será un arreglo de objetos que servirá para nombrar cada una de las columnas de la tabla, esto lo haremos de acuerdo a los parámetros que obtengamos de la solicitud. Aquí tienes un ejemplo, pero recuerda que la estructura depende de las características de API y los datos que quieras mostrar. El render se usa en la columna de imgURL porque necesitas renderizar una imagen en lugar de solo mostrar una URL.

```
8  const columns = [
9  |
10 |   {
11 |     title: 'Nombre',
12 |     dataIndex: 'name',
13 |   },
14 |   {
15 |     title: 'Precio',
16 |     dataIndex: 'price',
17 |   },
18 |   {
19 |     title: 'Categoría',
20 |     dataIndex: 'category',
21 |   },
22 |   {
23 |     title: 'Imagen',
24 |     dataIndex: 'imgURL',
25 |     render: (url) => <img src={url} alt="product" style={{ width: 100 }} />,
26 |   },
27 | ];
```

También incluiremos la función `rowSelection` para seleccionar cada fila, esta viene directo con el componente de Antd.

```
29 // rowSelection object indicates the need for row selection
30 const rowSelection = {
31   onChange: (selectedRowKeys, selectedRows) => {
32     console.log(`selectedRowKeys: ${selectedRowKeys}`, 'selectedRows: ', selectedRows);
33   },
34   getCheckboxProps: (record) => ({
35     disabled: record.name === 'Disabled User',
36     // Column configuration not to be checked
37     name: record.name,
38   })),
39 };
40
```

Para comenzar con la solicitud, no te olvides de importar la función `getProducts` **`import { getProducts } from '../services/products'`**; dentro del componente crea el estado para `products` (incluye `selectionType` de antd).

Construye la estructura de `useEffect` que permite que la solicitud se realice una vez se cargue el componente, y no siga rehaciendo la solicitud, creamos la constante **`data`** que almacenará la respuesta de `getProducts` de manera asíncrona, se utiliza el método `map` para iterar sobre cada producto.

En esta iteración, se crea un nuevo objeto “producto” que incluye todas las propiedades del producto original (`...product`) y se añade una nueva propiedad `key` que se establece como el valor de `_id` del producto y se actualiza el estado.

Por último manejamos el error y ejecutamos esta función.

```
const Productos = () => {
  const [products, setProducts] = useState([]);
  const [selectionType, setSelectionType] = useState('checkbox');

  useEffect(() => {
    const fetchProducts = async () => {
      try {
        const data = await getProducts();
        const productsWithKey = data.map(product => ({
          ...product,
          key: product._id,
        }));
        setProducts(productsWithKey);
      } catch (error) {
        console.error('Error al obtener los productos', error);
      }
    };

    fetchProducts();
  }, []);
};
```

Ahora si en nuestro componente Table podemos pasarle las propiedades columns y datasource con los valores que acabamos de construir.



Observa que se añade la propiedad scroll cuando la tabla llega a una altura de 400.




Los demás componentes que se muestran son parte de la UI.





```
62     return (  
63       <div>  
64         <Nav />  
65         <CrudButton />  
66  
67         <Divider />  
68  
69         <div className="products-container">  
70           <Table  
71             rowSelection={{  
72               type: selectionType,  
73               ...rowSelection,  
74             }}  
75             columns={columns}  
76             dataSource={products}  
77             scroll={{ y: 400 }}  
78           />  
79         </div>  
80       </div>  
81     );  
82   };  
83  
84   export default Productos;  
85
```

Por ejemplo, el componente CrudButton renderiza los botones necesarios para implementar el Crud.

Al final obtendrás una vista como esta.

 Productos Servicios Contacto 

<input type="checkbox"/>	Nombre	Precio	Categoría	Imagen
<input type="checkbox"/>	iPhone 15 pro	34300	Smartphone	
<input type="checkbox"/>	iPhone 15 pro max	39300	Smartphone	
<input type="checkbox"/>	Samsung Galaxy S23	28000	Smartphone	
<input type="checkbox"/>	MacBook Pro 16	36000	Laptop	

< 1 2 >