1. Endpoint de la API

Necesitamos que el equipo de backend que te pasen la documentación o, al menos, que te confirmen estos endpoints clave para la sección de **operadores**:

Funcionalidad	Método	Endpoint sugerido	Descripción
Obtener todos los operadores	**GET**	`/api/operadores`	Devuelve la lista completa de operadores
Obtener un operador	**GET**	`/api/operadores/:id`	Devuelve la info de un operador puntual
Crear operador	**POST**	`/api/operadores`	Alta de un nuevo operador
Modificar operador	**PUT**	`/api/operadores/:id`	Editar datos de un operador existente
Eliminar operador	**DELETE**	`/api/operadores/:id`	Dar de baja un operador
Cambiar estado	**PATCH**	`/api/operadores/:id/estado`	Habilitar o deshabilitar operador

🔹 **Importante**: Confirmar con backend **los nombres reales** de los endpoints y métodos.
Vamos a usar esos nombres exactos en el **JavaScript** para cargar y gestionar datos.

---

2. Estructura de datos

Necesitamos saber **cómo viene la información** para poder mostrarla bien en la tabla y en los formularios.
Por ejemplo, para la lista de operadores, necesitamos que confirmen que la API responde algo así:

```
[
  {
    "id": 1,
    "nombre": "Pablo",
    "apellido": "Larralde",
    "dni": "30234567",
    "email": "pablo@example.com",
    "estado": "Habilitado"
  },
  {
    "id": 2,
    "nombre": "Emilia",
    "apellido": "Gonzales",
    "dni": "28456789",
    "email": "emilia@example.com",
```

```
    "estado": "No habilitado"
  }
]
```

Esto es clave para que podamos poblar la **tabla de operadores** sin inventar datos de prueba.

---

3. Manejo de estados y errores

Necesitamos que la API:

- Devuelva **mensajes claros** cuando algo sale mal (por ejemplo, error de validación).
- Envíe **códigos HTTP correctos**:
  - **200** → OK
  - **201** → Creado
  - **400** → Error de validación
  - **401** → No autorizado
  - **404** → No encontrado
  - **500** → Error interno

Esto nos va a ayudar a mostrar mensajes más claros en el front (por ejemplo, “DNI ya registrado” o “Email inválido”).

---

4. Autenticación y roles

¿El sistema usa **JWT, cookies de sesión o tokens** para autenticar?
Es importante porque:

- Los administradores podrán **crear, editar y dar de baja operadores**.
- Los operadores normales **solo verán su perfil**.

Necesito que backend nos indique cómo vamos a **recibir el token** y **validar los permisos** desde el front.

---

5. Contraseñas

Como en tu diseño, la contraseña inicial parece ser el **DNI del operador**. Confirmar con backend:

- Si se genera automáticamente.
- Si el operador puede cambiarla luego.
- Si debemos mostrar un campo editable o solo informativo.

---

6. CORS y pruebas locales

Necesitamos que habiliten **CORS** para tu dominio local, por ejemplo:

```
http://localhost:3000
```

De lo contrario, las peticiones fetch/axios no van a funcionar durante el desarrollo.

---

7. Documentación de la API

Si tienen **Swagger, Postman o Insomnia**, necesitamos un archivo de colección o la URL de la documentación.
Eso te va a ahorrar muchísimo tiempo para probar las rutas antes de conectar el front.

Resumen para el equipo de backend

- Confirmar **endpoints** y **métodos** para operadores.
- Definir **estructura exacta** de los datos que devuelve la API.
- Establecer manejo correcto de **errores y códigos HTTP**.
- Confirmar **método de autenticación y roles**.
- Acordar cómo manejar **contraseñas iniciales**.
- Pedir habilitación de **CORS** para desarrollo.
- Solicitar **documentación oficial** si la tienen.