

# Memoria 1 (rev4). Backend - WS

## CRUDSeguridad y CORS

### Intercambio de Recursos de Origen Cruzado (CORS)

- **Definición:** CORS es un mecanismo de seguridad que utiliza encabezados HTTP para permitir que un navegador web (o aplicación SPA) acceda a recursos desde un dominio distinto al que originó la petición.
- **Problema AJAX:** Sin una configuración adecuada, el navegador bloquea por defecto las solicitudes entre dominios por seguridad, lo que impediría que una aplicación *frontend* externa consuma nuestro API.
- **Implementación:** Se utiliza el paquete `cors` en Node.js para gestionar estos permisos de forma sencilla a través de *middlewares*.

### Seguridad mediante Autenticación por Token

- **Protección Selectiva:** No todas las rutas requieren el mismo nivel de seguridad. En este proyecto, se protegen las operaciones de escritura (**POST**, **PUT**, **DELETE**) para evitar modificaciones no autorizadas.
- **Acceso Público:** Las consultas de lectura (**GET**) se mantienen abiertas para facilitar el acceso general a la información.
- **Mecanismo de Token:** El servidor valida que el cliente envíe una cabecera personalizada llamada `token` con una clave específica (ej. "password1234") antes de procesar la solicitud.

+1

---

## 2. Incorporación de Seguridad al API CRUD

La seguridad se implementa mediante un *middleware* de autorización denominado `auth()` que actúa como filtro previo a los controladores de las rutas críticas.

### Lógica del Middleware `auth()`

- **Validación de Cabecera:** El sistema verifica primero la existencia de la cabecera `token`.
  - **Verificación de Identidad:** Si el token enviado coincide con el definido en la configuración (`accessToken`), se permite el paso a la función `next()`.
  - **Gestión de Errores:** Si el token es incorrecto o no existe, el servidor responde con un código **401 (No autorizado)** en formato JSON.
-

## 4. Archivo de Configuración

Para mejorar la mantenibilidad del código, se ha extraído la lógica de parámetros a un archivo independiente llamado [config.js](#).

## Pruebas

### DELETE

The screenshot shows a REST client interface with the following details:

- Method:** DELETE
- URL:** http://localhost:3000/api/mascotas/545b1455u5v4ug5vu5
- Status:** 401 Unauthorized
- Response Body (JSON):**

```
{  \"result\": \"KO\",  \"msg\": \"Envia un código válido en la cabecera 'token'\"}
```

Con token:

DELETE ▼ http://localhost:3000/api/mascotas/69948610d9d2ca175cfbcb6e Send ▼

Docs Params Auth Headers (11) Body Scripts Settings Cookies

raw ▼ JSON ▼ Schema Beautify

```
1 {
2   {
3     "id": "5ad65508f6efc76d03ddbe6f",
4     "nombre": "Pepe"
5   }
}
```

Body ▼ ↺ 200 OK • 28 ms • 364 B • 🌐 | Save Response ⋮

{} JSON ▼ ▶ Preview 🖼 Visualize ▼ 🔍 📄 🔗

```
1 {
2   "n": 1,
3   "ok": 1,
4   "deletedCount": 1
5 }
```

**Comando de clonacion:** <https://github.com/RatonDeGranja/api-crud.git>

**Token de acceso:** ghp\_P6Fz2bl6Q4H13vLRamZIIW8tGhxx2e3kYWS5