

Documentación API: Galactic Cargo

Esta API simula un sistema de gestión logística interplanetaria.

- **Repositorio:** [GitHub - joadelviajaca/galacticCargo](#)
- **Base URL:** `http://localhost:3000`
- **Latencia:** La API tiene una demora artificial aleatoria de **200ms a 1000ms** para obligar a gestionar estados de carga (`isLoading`).
- **Persistencia:** Los datos están en memoria. Si reinicias el servidor (`node server.js`), los datos vuelven al estado inicial.

Usuarios Pre-cargados (Seed Data)

Utiliza estos usuarios para probar los diferentes roles y permisos en tu aplicación Angular. **La contraseña es la misma para ambos.**

Rol	Email	Password	Nombre	Permisos
COMMANDER (Admin)	<code>admin@galaxy.com</code>	<code>1234</code>	Cmdr. Shepard	Ver, Crear, Borrar
PILOT (User)	<code>han@solo.com</code>	<code>1234</code>	Han Solo	Ver, Crear

Autenticación

1. Login

Obtiene el Token JWT para acceder a las rutas privadas.

- **Método:** `POST`
- **Endpoint:** `/auth/login`
- **Body:**

JSON

```
{
  "email": "han@solo.com",
  "password": "1234"
}
```

- **Respuesta (200 OK):**

JSON

```
{
  "token": "eyJhbGciOi...",
  "user": {
    "id": "65a...",
    "email": "han@solo.com",
    "name": "Han Solo",
    "role": "pilot"
  }
}
```

2. Registro

Crea un nuevo piloto (siempre se crea con rol `pilot` por defecto).

- **Método:** `POST`
- **Endpoint:** `/auth/register`
- **Body:**

JSON

```
{
  "name": "Luke Skywalker",
  "email": "luke@force.com",
  "password": "1234"
}
```

3. Verificar Estado (Re-Login / F5)

Valida si el token actual sigue siendo válido y recupera los datos del usuario. Ideal para usar en el `APP_INITIALIZER` o en el `constructor` del `AuthService`.

- **Método:** `GET`

- **Endpoint:** `/auth/check-status`
 - **Header:** `Authorization: Bearer <TOKEN>`
 - **Respuesta (200 OK):** Devuelve el objeto usuario actualizado.
-



Endpoints Públicos

1. Noticias Galácticas

No requiere autenticación. Sirve para probar que el `Interceptor` no bloquee rutas públicas.

- **Método:** `GET`
 - **Endpoint:** `/news`
 - **Respuesta:** Array de noticias.
-



Endpoints de Carga (Protegidos)

Requisito: Todas estas peticiones deben incluir la cabecera `Authorization: Bearer <TOKEN>`.

1. Listar Cargas

Disponible para **Pilotos** y **Comandantes**.

- **Método:** `GET`
- **Endpoint:** `/carga`
- **Respuesta:** Array de objetos `Cargo`.

JSON

```
[
  {
    "_id": "...",
    "code": "CRGO-001",
    "description": "Cristales Kyber",
    "origin": "Ilum",
    "destination": "Coruscant",
    "value": 5000,
    "status": "Disponible"
  },
  ...
]
```

2. Crear Nueva Carga

Disponible para **Pilotos** y **Comandantes**.

- **Método:** POST
- **Endpoint:** /cargo
- **Body:**

JSON

```
{
  "code": "CRGO-999",
  "description": "Suministros Médicos",
  "origin": "Alderaan",
  "destination": "Endor",
  "value": 200,
  "status": "Disponible"
}
```

3. Eliminar Carga (⚠ Solo Comandantes)

Si intentas acceder con un usuario `pilot`, recibirás un error **403 Forbidden**.

- **Método:** DELETE
 - **Endpoint:** /cargo/:id
 - **Ejemplo:** /cargo/65a9f...
-



Códigos de Error Comunes

Tu aplicación Angular (especialmente el Interceptor de Errores) debe saber manejar estos códigos:

- **400 Bad Request:** Datos faltantes o incorrectos (ej: email duplicado en registro).
- **401 Unauthorized:** No enviaste token o el token ha caducado/es falso. -> **Acción:** Redirigir a Login.
- **403 Forbidden:** Tienes token válido, pero no tienes permisos para hacer esto (ej: Han Solo intentando borrar una carga). -> **Acción:** Mostrar alerta "Permisos insuficientes".
- **500 Internal Server Error:** Fallo en el servidor (o en la BD).