# Documentación Técnica - API de Publicaciones y Comentarios

### 1. Introducción

La **API de Publicaciones y Comentarios** es un servicio **RESTful** que permite gestionar publicaciones tipo blog y comentarios asociados. El sistema incorpora:

- **Autenticación JWT** para acceso seguro.
- **Control de permisos** para proteger recursos.
- **Validaciones estrictas** y **sanitización HTML** contra ataques XSS.
- **Paginación** y filtros de búsqueda para optimizar consultas.

La API está diseñada para integrarse fácilmente con aplicaciones web o móviles y puede ejecutarse localmente o en entornos en la nube.

---

### 2. Instalación y Configuración

#### 2.1 Requisitos Previos

- Node.js 18+
- MySQL 8.0+
- Docker (opcional, para levantar base de datos)

#### 2.2 Clonar el Proyecto

git clone https://github.com/Cesar0902/api-pubs-comments.git

cd api-pubs-comments

#### 2.3 Instalar Dependencias

npm install

#### 2.4 Configurar Variables de Entorno

Crear un archivo .env en la raíz con:

PORT=5000

DB_HOST=localhost

DB_PORT=3307

DB_USER=bloguser

DB_PASSWORD=blogpass

DB_NAME=blogdb

JWT_SECRET=Tengo_la_camisa_negra

JWT_EXPIRES_IN=3h

**2.5 Levantar Base de Datos con Docker**

docker compose -f docker-compose.yml up -d --build

Este comando:

- Crea un contenedor MySQL 8.0 en el puerto **3307**.
- Inicializa la base de datos blogdb.
- Crea usuario bloguser con contraseña blogpass.

**2.6 Ejecutar la API**

npm run dev

La API estará disponible en:

http://localhost:5000

---

**3. Autenticación**

La API utiliza **JWT (JSON Web Token)**. Para acceder a rutas protegidas:

Authorization: Bearer <tu_token_jwt>

---

**4. Endpoints**

**4.1 Autenticación**

| Método | Ruta | Descripción | Protegido |
|---|---|---|---|
| POST | /auth/register | Registro de usuario | ❌ |
| POST | /auth/login | Inicio de sesión | ❌ |

---

**4.2 Publicaciones**

| Método | Ruta | Descripción | Protegido |
|---|---|---|---|
| GET | /publicaciones | Lista publicaciones con paginación y búsqueda | ❌ |
| GET | /publicaciones/:id | Ver publicación por ID | ❌ |
| POST | /publicaciones | Crear publicación | ✅ |
| PUT | /publicaciones/:id | Editar publicación (solo autor) | ✅ |
| DELETE | /publicaciones/:id | Eliminar publicación (solo autor) | ✅ |

---

### 4.3 Comentarios

| Método | Ruta | Descripción | Protegido |
|---|---|---|---|
| GET | /publicaciones/:id/comentarios | Lista comentarios de una publicación | ❌ |
| POST | /publicaciones/:id/comentarios | Crear comentario | ✅ |

---

### 5. Ejemplos de Uso con cURL

**Registro de usuario**

```
curl -X POST http://localhost:5000/auth/register
  -H "Content-Type: application/json"
  -d '{"nombre": "David", "email": "david@example.com", "password": "MiPassword123"}'
```

**Login de usuario**

```
curl -X POST http://localhost:5000/auth/login
  -H "Content-Type: application/json"
  -d '{"email": "david@example.com", "password": "MiPassword123"}'
```

**Listar publicaciones**

```
curl "http://localhost:5000/publicaciones?page=1&limit=5"
```

**Crear publicación (requiere token)**

```
curl -X POST http://localhost:5000/publicaciones
  -H "Authorization: Bearer <tu_token>"
  -H "Content-Type: application/json"
  -d '{"titulo": "Mi post", "contenido": "Contenido de prueba"}'
```

---

### 6. Seguridad Implementada

- **Sanitización HTML:** Limpieza automática contra XSS.
- **Validaciones estrictas:**
  - Passwords: 8-20 caracteres, mayúscula, número y símbolo.
  - Emails con formato válido.
  - UUIDs correctos.
- **JWT** con expiración configurable.
- **Control de permisos** por autor.

---

**7. Códigos de Estado HTTP**

| Código | Significado |
|---|---|
| 200 | OK |
| 201 | Created |
| 400 | Bad Request |
| 401 | Unauthorized |
| 403 | Forbidden |
| 404 | Not Found |
| 409 | Conflict |
| 500 | Internal Server Error |

---

**8. Estructura del Proyecto**

```
├── src/
|   ├── config/
|   ├── controllers/
|   ├── middlewares/
|   ├── models/
|   ├── routes/
|   ├── schemas/
|   └── utils/
├── test/
├── docker-compose.yml
├── package.json
└── server.js
```

---

**9. Conclusión**

Esta API ofrece una solución segura, escalable y fácil de integrar para sistemas de publicaciones y comentarios. El uso de JWT, validaciones estrictas y sanitización HTML garantizan la seguridad y confiabilidad del servicio.