Backend (Spring Boot) – Noticias: Public / Secured (JWT)

Este proyecto se entrega **al alumnado** como **base**. Compila y arranca, pero **NO está terminado**.

Tu objetivo es implementar:

1. Los **servicios** (lógica de negocio) para Noticias.
2. La **securización JWT** de la ruta `/api/secured/**`.

> Los repositorios ya incluyen datos de ejemplo y funcionan.

---

1) Qué ya está hecho

- Modelo `Noticia`
- DTO `NoticiaRequest`
- Repositorio en memoria (`InMemoryNoticiaRepository`) con datos de ejemplo
- Controllers con rutas:
  - `/api/public/noticias/**`
  - `/api/secured/noticias/**`
  - `/auth/login`
- Swagger/OpenAPI (si está configurado en el proyecto)
- Handler de errores base (`ApiExceptionHandler`)

---

2) Qué tienes que implementar (obligatorio)

2.1 Servicio de Noticias (CRUD)

Implementa en `NoticiaServiceImpl`:

- `listar()`
- `obtener(id)`
- `crear(request)`
- `actualizar(id, request)`
- `borrar(id)`

Reglas:

- `id` no puede ser `null` → lanzar `IllegalArgumentException`
- `titulo` y `contenido` no pueden ser `null` ni vacios → `IllegalArgumentException`
- Si no existe el id → lanzar `NotFoundException`

Códigos HTTP esperados:

- GET lista → 200 + array JSON
- GET id → 200, si no existe → 404
- POST → 201 + objeto con `id`
- PUT → 200 + objeto actualizado, si no existe → 404
- DELETE → 204

2.2 Login y JWT

Implementa `AuthServiceImpl#login`:

- Usuario en memoria: `admin/admin` o `user/user`
- Si credenciales OK → devolver `{ "token": "..." }`

Implementa `JwtUtil`:

- `generateToken(username)`
- `validateAndGetUsername(token)`

Implementa `JwtAuthFilter`:

- Leer header `Authorization: Bearer <token>`
- Validar token
- Autenticar en `SecurityContext`

Implementa `SecurityConfig`:

- Permitir:
  - `/auth/login`
  - `/api/public/**`
  - Swagger (si se usa)
- Proteger:
  - `/api/secured/**` → requiere JWT (401/403 si no token)

> Ahora mismo TODO está permitido para que compile, pero debes securizarlo.

---

3) Rutas rápidas

- Swagger UI: `http://localhost:8080/swagger-ui/index.html`
- OpenAPI JSON: `http://localhost:8080/v3/api-docs`

---

4) Ejecución

```
mvn clean spring-boot:run
```

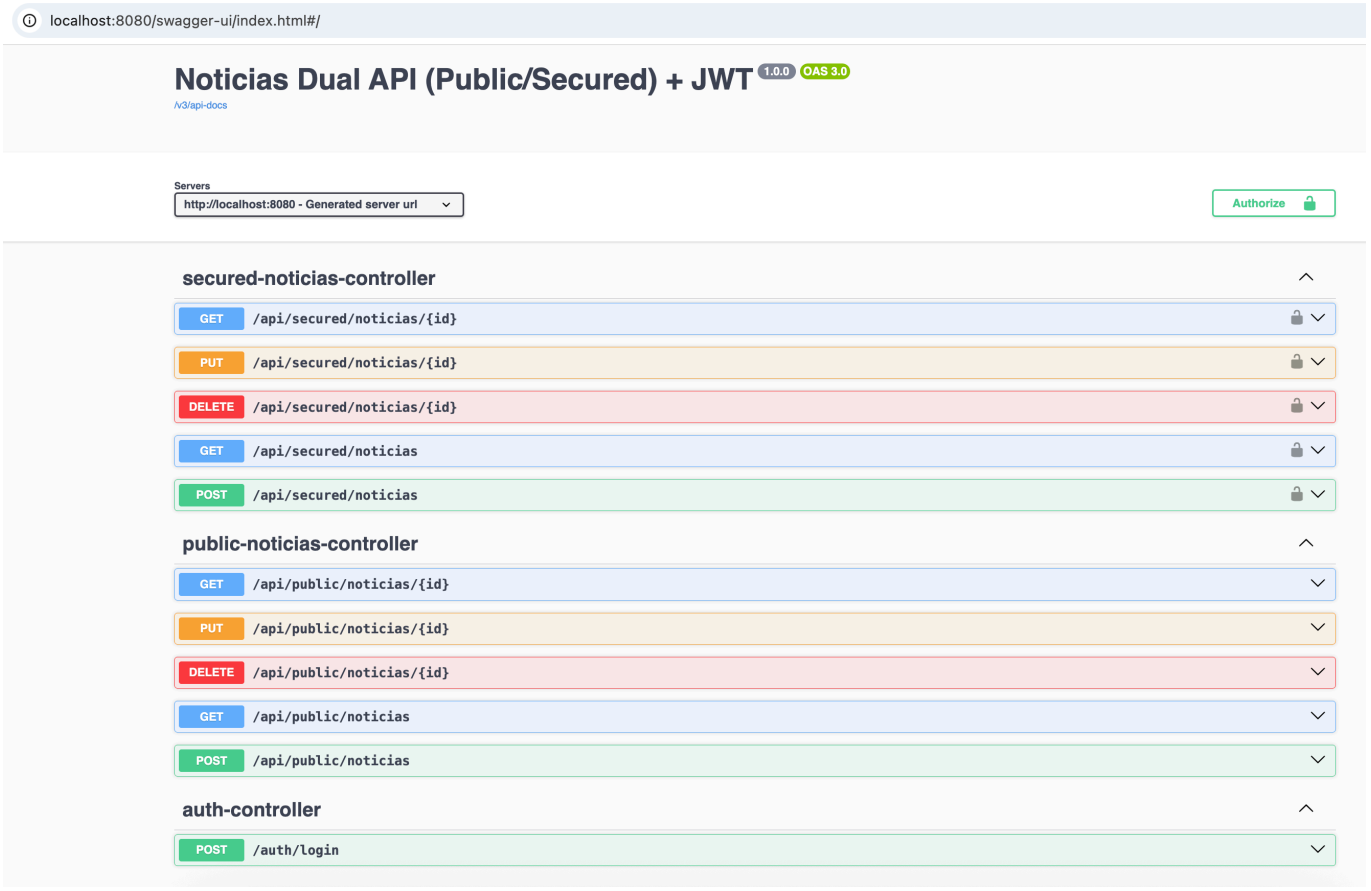
---

5) Evaluación (10 puntos)

- CRUD Público correcto: **4 puntos**
- JWT + CRUD Securizado correcto: **6 puntos**

¿Qué se verifica?

```
== SECURED CRUD (max 6 puntos) ==
POST http://localhost:8080/auth/login -> OK (200)
GET http://localhost:8080/api/secured/noticias -> OK (200)
POST http://localhost:8080/api/secured/noticias -> OK (201)
GET http://localhost:8080/api/secured/noticias/10 -> OK (200)
PUT http://localhost:8080/api/secured/noticias/10 -> OK (200)
DELETE http://localhost:8080/api/secured/noticias/10 -> OK (204)
GET http://localhost:8080/api/secured/noticias/10 -> OK (404)
PUNTUACIÓN SECURED: 6/6
------------------------------------------------
NOTA FINAL: 6,0/10 (public: 0/4, secured: 6/6)
------------------------------------------------
== PUBLIC CRUD (max 4 puntos) ==
GET http://localhost:8080/api/public/noticias -> OK (200)
POST http://localhost:8080/api/public/noticias -> OK (201)
PUT http://localhost:8080/api/public/noticias/11 -> OK (200)
DELETE http://localhost:8080/api/public/noticias/11 -> OK (204)
GET http://localhost:8080/api/public/noticias/11 -> OK (404)
PUNTUACIÓN PUBLIC: 4/4
```


localhost:8080/swagger-ui/index.html#/
Noticias Dual API (Public/Secured) + JWT
1.0.0
OAS 3.0
/v3/api-docs
Servers
http://localhost:8080 - Generated server url
Authorize
secured-noticias-controller
GET /api/secured/noticias/{id}
PUT /api/secured/noticias/{id}
DELETE /api/secured/noticias/{id}
GET /api/secured/noticias
POST /api/secured/noticias
public-noticias-controller
GET /api/public/noticias/{id}
PUT /api/public/noticias/{id}
DELETE /api/public/noticias/{id}
GET /api/public/noticias
POST /api/public/noticias
auth-controller
POST /auth/login