API Editorial (JWT) + Documentación + Servicios

Tu objetivo es implementar una API REST de Editoriales con:

1. **4 servicios REST** (Auth, Authors, Books, Publishers)
2. **Securización con JWT** (Bearer token) siguiendo el patrón proporcionado
3. **Documentación REST** (OpenAPI/Swagger) usando anotaciones en controllers
4. Pasar tests y obtener la **nota automática sobre 10**

---

1 Qué debes implementar

1.1 Servicios REST (4)

Debes completar los métodos de estos controllers (actualmente lanzan `UnsupportedOperationException`):

- `AuthController`
- `AuthorController`
- `BookController`
- `PublisherController`

Los controllers deben delegar en la capa service (`IAuthorService`, `IBookService`, `IPublisherService`) y devolver los **HTTP status correctos** `(Se verifican desde los test)`.

1.2 Capa Service

Las implementaciones `*ServiceImpl` que encuentran implementados, y conecgtan con la lógica de los repositorios.

1.3 JWT

Debes securizar los endpoints `/api/**` con JWT **siguiendo el patrón** de estos ficheros:

- `JwtService`
- `JwtAuthenticationFilter`
- `SecurityConfig`

Requisitos:

- Login en `POST /api/auth/login`
- El cliente debe enviar: `Authorization: Bearer <TOKEN>`
- No se usan roles: pero el usuario debe de ser `admin`, y la contraseña `admin123`.

1.4 Documentación REST (Swagger/OpenAPI)

Debes documentar los endpoints con anotaciones OpenAPI, por ejemplo:

- `@Operation(summary = "...")`
- `@ApiResponses(...)`

---

2 Rutas (endpoints) esperados

> Todas las rutas empiezan por `/api`.

2.1 Auth

- `POST /api/auth/login`

2.2 Authors

- `GET /api/authors`
- `GET /api/authors/{id}`
- `GET /api/authors/{id}/books`
- `POST /api/authors`

2.3 Books

- `GET /api/books`
- `GET /api/books/{id}`
- `GET /api/books?authorId={id}`
- `POST /api/books?authorId={id}`

2.4 Publishers

- `GET /api/publishers`
- `POST /api/publishers`

---

3 Códigos de error / respuestas esperadas

Los tests comprueban el tipo de respuesta.

3.1 Securización JWT

A Sin token (sin header Authorization)

Para cualquier endpoint protegido (`/api/authors`, `/api/books`, `/api/publishers`, etc.):

- **401 Unauthorized**

B Token inválido o mal formado

- **401 Unauthorized** (objetivo del ejercicio)

> Si te devuelve 403 en algún caso, debes ajustar la configuración/filtro para unificar a 401.

C Token válido

- Debe devolver el resultado y un `2xx` (200/201) según el endpoint.

3.2 Auth — `POST /api/auth/login`

OK

- **200 OK** + JSON con `{ "token": "..." }`

Credenciales incorrectas

- **401 Unauthorized**

Body inválido (campos vacíos, null, etc.)

- **400 Bad Request**

3.3 Authors

`GET /api/authors`

- OK → **200**
- Sin/Token inválido → **401**

`GET /api/authors/{id}`

- OK → **200**
- No existe → **404 Not Found**
- Sin/Token inválido → **401**

`GET /api/authors/{id}/books`

- OK → **200**
- Autor no existe → **404**
- Sin/Token inválido → **401**

`POST /api/authors`

- OK → **201 Created**

- Body inválido (`name` vacío, etc.) → **400**
- Sin/Token inválido → **401**

3.4 Books

`GET /api/books`

- OK → **200**
- Sin/Token inválido → **401**

`GET /api/books/{id}`

- OK → **200**
- No existe → **404**
- Sin/Token inválido → **401**

`GET /api/books?authorId={id}`

- OK → **200** (lista, puede ser vacía)
- Author no existe → **404** (si tu enunciado lo exige; si no, lista vacía)
- Sin/Token inválido → **401**

`POST /api/books?authorId={id}`

- OK → **201 Created**
- Author no existe → **404**
- Body inválido → **400**
- Sin/Token inválido → **401**

3.5 Publishers

`GET /api/publishers`

- OK → **200**
- Sin/Token inválido → **401**

`POST /api/publishers`

- OK → **201 Created**
- Body inválido → **400**
- Sin/Token inválido → **401**

---

4 ¿Cómo puedes probar el funcionamiento?

Desde el proyecto de test que se te proporciona lanza las pruebas

Ejecutar verificación + nota

La nota se genera desde el proyecto de test aunque hayan errores

```
mvn clean verify
```

Se genera:

- `target/nota.txt`

Nota automática (suma 10)

La nota se calcula por 4 bloques:

1. Auth
2. Authors
3. Books
4. Publishers

Cada bloque aporta una parte y la suma total es **10**.
