**Dependencias esenciales para cualquier app web con Spring Boot**

**1. Spring Web**

* **¿Por qué?** Te permite crear controladores REST (API) para que el frontend se comunique con tu backend.
* 👉 Dependencia: spring-boot-starter-web

**2. Spring Data JPA**

* **¿Por qué?** Facilita el acceso a bases de datos relacionales usando entidades y repositorios.
* 👉 Dependencia: spring-boot-starter-data-jpa

**3. Base de datos (H2 y/o MySQL/PostgreSQL)**

* **¿Por qué?** Puedes usar H2 para pruebas y MySQL/PostgreSQL para producción.
* 👉 Dependencias:
  + com.h2database:h2 (para pruebas en memoria)
  + mysql:mysql-connector-java o org.postgresql:postgresql

**4. Lombok**

* **¿Por qué?** Ahorra tiempo escribiendo getters, setters, constructores, etc.
* 👉 Dependencia: org.projectlombok:lombok

**🔐 Seguridad (si quieres autenticación de usuarios)**

**5. Spring Security**

* **¿Por qué?** Protege tus endpoints, permite login/logout, y funciona bien con JWT.
* 👉 Dependencia: spring-boot-starter-security

**6. JWT (JSON Web Tokens)**

* **¿Por qué?** Para manejar autenticación de forma moderna (token en lugar de sesiones).
* 👉 Librería sugerida: io.jsonwebtoken:jjwt o com.auth0:java-jwt

**🔁 Tareas recurrentes y notificaciones**

**7. Spring Scheduling**

* **¿Por qué?** Permite ejecutar tareas automáticamente en intervalos definidos (útil para tareas recurrentes).
* 👉 Incluido en Spring, solo agrega @EnableScheduling.

**8. Spring Mail (si vas a enviar notificaciones por correo)**

* 👉 Dependencia: spring-boot-starter-mail

**📦 Exportación de tareas (PDF, Excel, etc.)**

**9. Apache POI**

* **¿Por qué?** Para crear archivos Excel (XLSX/CSV) desde Java.
* 👉 Dependencia: org.apache.poi:poi-ooxml

**10. iText o OpenPDF**

* **¿Por qué?** Para generar archivos PDF.
* 👉 Dependencia (alternativas):
  + com.lowagie:itext (versión libre)
  + com.github.librepdf:openpdf

**🧪 Pruebas**

**11. Spring Boot Starter Test**

* **¿Por qué?** Incluye JUnit, Mockito y otras herramientas para probar tu aplicación.
* 👉 Dependencia: spring-boot-starter-test

**🛠️ Otras útiles**

**12. MapStruct (opcional)**

* **¿Por qué?** Para mapear entidades ↔ DTOs de forma rápida y limpia.
* 👉 Dependencia: org.mapstruct:mapstruct

**13. ModelMapper (alternativa a MapStruct, más dinámica)**

* 👉 Dependencia: org.modelmapper:modelmapper

**14. Swagger / Springdoc OpenAPI**

* **¿Por qué?** Para documentar tu API y probarla fácilmente desde el navegador.
* 👉 Dependencia recomendada:
  + org.springdoc:springdoc-openapi-starter-webmvc-ui

¡Buena pregunta, Leo! Si quieres mandar **notificaciones por el navegador** como las de YouTube o Gmail (esas que aparecen en la esquina de la pantalla, incluso si la pestaña está cerrada), necesitas usar una combinación de tecnologías **del lado del cliente (frontend)** y **un empuje desde el servidor (backend)**.

**🧠 Tecnología recomendada: Push Notifications con Web Push API**

**🧱 Componentes que necesitas:**

**🔔 1. Web Push API (en el navegador)**

* **¿Qué es?** Es una API nativa que permite recibir notificaciones aunque el navegador esté en segundo plano o cerrado (si el sitio está permitido).
* **Frontend necesario:** JavaScript + Service Workers

**🌐 2. Service Worker (en el frontend)**

* **¿Qué es?** Es un script que vive en segundo plano y escucha notificaciones push incluso si tu app no está activa.
* **Necesario para:** recibir y mostrar la notificación.

**🚀 3. Backend para enviar el push**

Para Java (Spring Boot), necesitas una librería que pueda **enviar notificaciones Web Push** al navegador.

**👉 Librería recomendada:**

[**web-push-java**](https://github.com/web-push-libs/webpush-java)

* **¿Qué hace?** Te permite enviar notificaciones push usando VAPID keys desde Java a los navegadores compatibles.