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

**1. Spring Web**

* Permite crear controladores REST (API) para que el frontend se comunique con el backend.
* Dependencia: spring-boot-starter-web

**2. Spring Data JPA**

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

**3. Base de datos (PostgreSQL)**

* Se puede usar PostgreSQL para producción.
* Dependencia: org.postgresql:postgresql

**4. Lombok**

* Ahorra tiempo escribiendo getters, setters, constructores, etc.
* Dependencia: org.projectlombok:lombok

**Seguridad**

**5. Spring Security**

* Protege los endpoints, permite login/logout, y funciona bien con JWT.
* Dependencia: spring-boot-starter-security

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

* 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**

* Permite ejecutar tareas automáticamente en intervalos definidos (útil para tareas recurrentes).
* Incluido en Spring, solo hay que agregar @EnableScheduling.

**8. Spring Mail**

* Dependencia: spring-boot-starter-mail

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

**9. Apache POI**

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

**10. iText o OpenPDF**

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

**Pruebas**

**11. Spring Boot Starter Test**

* Incluye JUnit, Mockito y otras herramientas para probar la aplicación.
* Dependencia: spring-boot-starter-test

**12. MapStruct**

* Para mapear entidades ↔ DTOs de forma rápida y limpia.
* Dependencia: org.mapstruct:mapstruct

**13. Swagger / Springdoc OpenAPI**

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

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

**Componentes que necesitas:**

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

* 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)**

* 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)

* Permite enviar notificaciones push usando VAPID keys desde Java a los navegadores compatibles.