✅ FASE 4.2 – Estructura de un Proyecto Spring Boot + Gestión de Dependencias con Maven / Gradle

🧠 Objetivo

Entender cómo se estructura un proyecto Spring Boot moderno, cómo funciona Maven (o Gradle) como herramienta de construcción y gestión de dependencias, y cómo esto se integra con todo el stack backend.

🔍 ¿Qué es Maven?

Maven es una **herramienta de automatización de compilación y gestión de proyectos** para Java. Permite:

- Declarar dependencias de librerías (como Spring Boot, Hibernate, etc.)
- Compilar el código fuente
- Ejecutar tests
- Empaquetar el proyecto (por ejemplo, en un `.jar`)
- Gestionar el ciclo de vida del proyecto

📁 Maven se basa en una estructura de carpetas estándar y un archivo central: `pom.xml`.

🔸 ¿Qué es Gradle (comparación)?

Gradle es otra herramienta similar a Maven, más moderna y basada en un enfoque declarativo y más flexible (usa Groovy o Kotlin DSL en lugar de XML). También sirve para compilar, testear y manejar dependencias.

Característica	Maven	Gradle
Configuración	XML (`pom.xml`)	DSL (Groovy/Kotlin, `build.gradle`)

Velocidad	Lento en proyectos grandes	Más rápido con compilación incremental
Curva de aprendizaje	Más simple	Más flexible, pero más compleja
Comunidad	Enorme	Creciendo rápido

💡 **En esta formación usamos Maven**, por su simplicidad y amplia adopción.

🧱 Estructura Estándar de un Proyecto Spring Boot (con Maven)

Cuando generás un proyecto en https://start.spring.io/, se te descarga una estructura similar a esta:

bash
CopyEdit

```
mi-proyecto/
├── src/
│   ├── main/
│   │   ├── java/
│   │   │   └── com/miempresa/miproyecto/
│   │   │       └── MiProyectoApplication.java
│   │   └── resources/
│   │       ├── application.properties
│   │       └── static/ (HTML, CSS, JS)
│   └── test/
│       └── java/
│           └── com/miempresa/miproyecto/
│               └── MiProyectoApplicationTests.java
├── pom.xml  ← Archivo central de configuración de Maven
```

📄 ¿Qué contiene `pom.xml`?

Es el "corazón" de Maven. Define:

- El **ID del proyecto**, versión, nombre.
- Las **dependencias** (librerías necesarias como Spring Web, JPA, Security).

- Plugins para empaquetar, testear, correr la app.
- Repositorios donde buscar las librerías (por defecto: Maven Central).

Ejemplo de sección de dependencias:

xml
CopyEdit

```xml
<dependencies>
    <!-- Spring Web -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <!-- Base de datos en memoria H2 -->
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <scope>runtime</scope>
    </dependency>
</dependencies>
```

🛠️ ¿Cómo se genera un proyecto?

1. Ir a 👉 https://start.spring.io
2. Elegir:
    - **Project:** Maven Project
    - **Language:** Java
    - **Spring Boot:** (usar la última estable)
    - **Metadata:** nombre, grupo, artefacto
    - **Dependencies:** como Spring Web, Spring Data JPA, Lombok, etc.
3. Descargar el `.zip` y descomprimirlo.

4. Abrir en IntelliJ (como proyecto Maven, detecta `pom.xml` automáticamente).

🔁 ¿Cómo funciona Maven en el día a día?

Comandos básicos (consola o integrados en IDE):

Comando Maven	Función
`mvn compile`	Compilar el proyecto
`mvn test`	Ejecutar pruebas unitarias
`mvn package`	Empaquetar el proyecto en `.jar`
`mvn spring-boot:run`	Ejecutar la app directamente

IntelliJ IDEA también ejecuta Maven automáticamente al importar el proyecto.

🧩 ¿Cómo se integra con Spring Boot y el stack?

- **Spring Boot Starter** son conjuntos de dependencias preconfiguradas. Ej: `spring-boot-starter-web` ya incluye Tomcat, Jackson, etc.
- Esto **reduce drásticamente la configuración manual**.
- Gracias a Maven, **no necesitas descargar nada a mano**: todo se resuelve con un `pom.xml`.

🎯 Ventajas de usar Maven con Spring Boot

✅ Proyectos ordenados y estándar
✅ Cero configuración manual: se descarga todo automáticamente
✅ Facilita la colaboración entre programadores
✅ Fácil integración con herramientas como GitHub, Jenkins, Docker, etc.

📌 Buenas prácticas

- **Nunca modifiques la estructura base de Maven.**
- **Usá siempre `pom.xml` para agregar dependencias.**
- **Controlá la versión de Java y Spring Boot que usás.**
- Commit inicial = proyecto limpio generado desde Spring Initializr.

🧠 En entrevistas técnicas te pueden preguntar:

- ¿Qué es el archivo `pom.xml` y qué hace?
- ¿Qué diferencia hay entre Maven y Gradle? ¿Cuál usás y por qué?
- ¿Cómo agregás una nueva dependencia al proyecto?
- ¿Qué es un Starter en Spring Boot?
- ¿Dónde vive la configuración de la aplicación?

✅ Listo para avanzar...

Una vez que entendiste esto, estás listo para pasar a la **Fase 4.3: Controladores REST y Peticiones HTTP con Spring Boot**.