Clase grabada

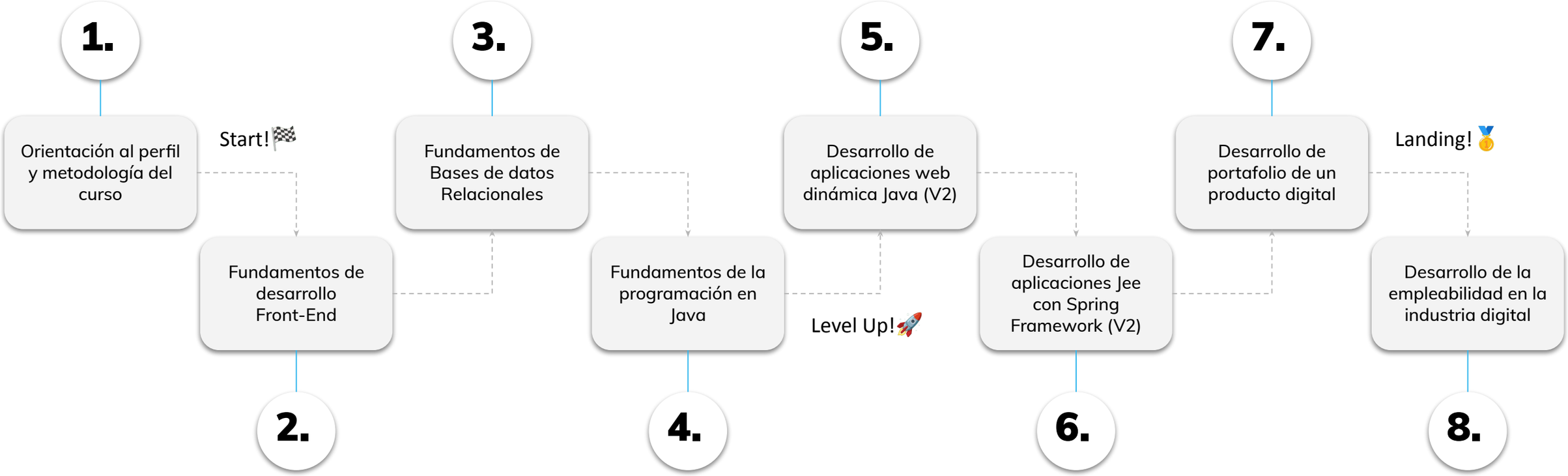
El gestor de Proyectos

Plan formativo: Desarrollo de Aplicaciones Full Stack Java Trainee V2.0

English Always | alkemy

HOJA DE RUTA

¿Cuáles **skill** conforman el programa?



English Always | alkemy

REPASO CLASE ANTERIOR

En la clase anterior trabajamos 📚:

- ✓ *Gestor de dependencias: Maven y el POM*
- ✓ *Spring Framework-MVC-Spring Boot*

English Always | alkemy

LEARNING PATHWAY

. ¿Sobre qué temas trabajaremos?

6.1 Start!🏁

El gestor de Proyectos → Maven → Dependencias

El manejo de dependencias con Maven
El repositorio general Maven y el repositorio local

English Always | alkemy

OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?

- *Aprender a manejar dependencias con Maven*
- *Conocer el repositorio local y genera de Maven*

Dependencias con Maven

Manejo de dependencias con Maven

Maven permite a los desarrolladores declarar las dependencias que su proyecto necesita para funcionar correctamente en un archivo especial llamado **pom.xml** (Project Object Model). Esto es beneficioso por varias razones. En primer lugar, proporciona una forma estructurada y legible de especificar las bibliotecas necesarias. En segundo lugar, Maven automatiza la gestión de versiones, lo que significa que puedes asegurarte de que tu proyecto use las versiones correctas de las bibliotecas, evitando conflictos y problemas de compatibilidad.

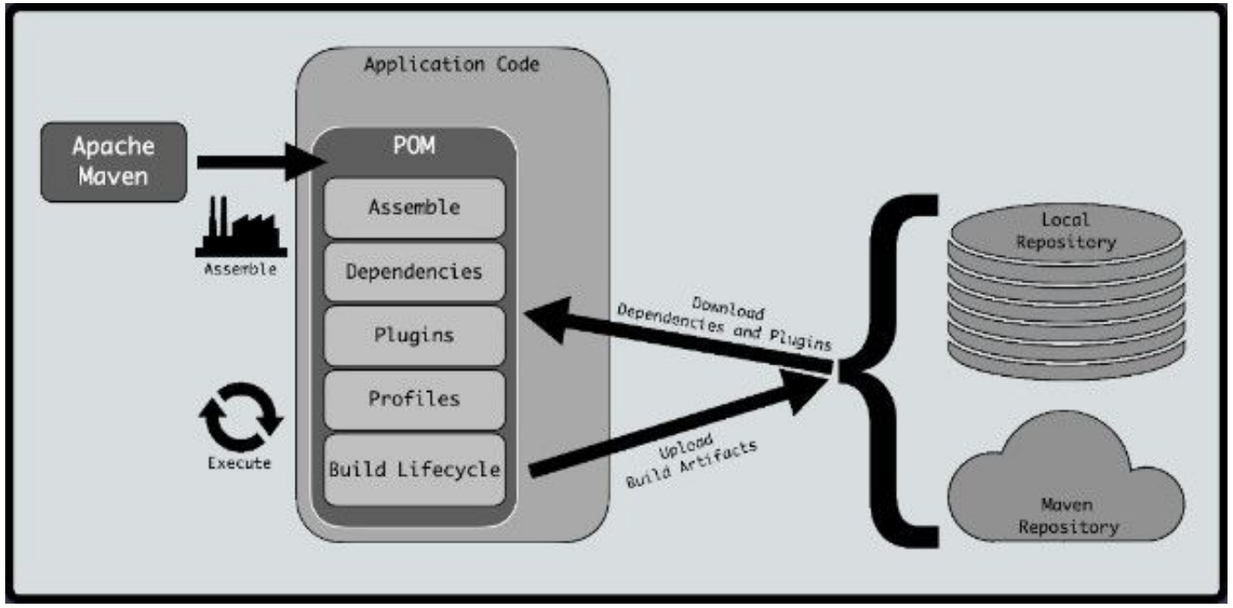
Manejo de dependencias con Maven

El archivo **pom.xml** también incluye información sobre el grupo (groupId), el artefacto (artifactId), y la versión (version) de las dependencias. **Esto permite a Maven buscar y descargar automáticamente las bibliotecas** desde repositorios remotos, como el Repositorio Central Maven.

Además, Maven es capaz de gestionar las dependencias transitivas, lo que significa que, si una biblioteca A depende de la biblioteca B, Maven también descargará B cuando necesites A.

Manejo de dependencias con Maven

Una ventaja significativa de este enfoque es la facilidad de mantenimiento. Cuando necesitas actualizar una biblioteca, simplemente modificas la versión en el pom.xml, y Maven se encargará del proceso de actualización sin necesidad de intervención manual.



Manejo de dependencias con Maven

En el archivo pom.xml, puedes especificar las bibliotecas que tu proyecto necesita para funcionar correctamente. Maven se encarga de descargar estas bibliotecas desde repositorios y gestionar las versiones de manera eficiente.

En este ejemplo, estamos declarando una dependencia en el proyecto hacia Spring Framework JPA. Maven buscará y descargará esta biblioteca y sus dependencias desde los repositorios configurados.

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
</dependencies>
```

Manejo de dependencias con Maven

Instalación:

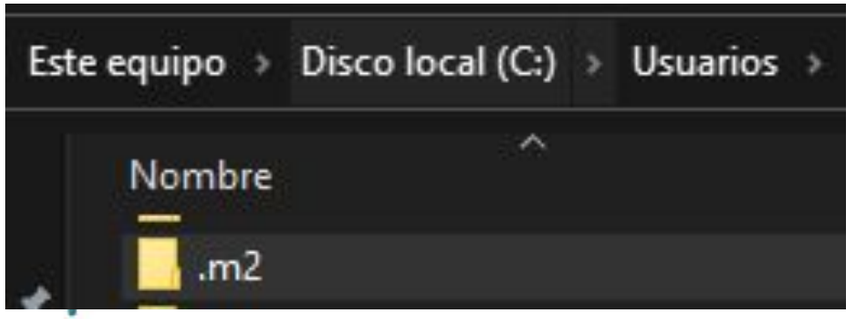
Maven normalmente ya viene instalado en cualquier distribución popular de linux o al menos, está disponible en los repositorios oficiales de la distribución. En otros sistemas, como MacOSX, también está instalado por defecto. Para quienes utilizan Windows, deben descomprimir el fichero zip en algún lugar seguro del directorio. Luego, debes crear la variable de entorno **M2_HOME apuntando al directorio** donde descomprimimos Maven y añadir **M2_HOME al path del sistema** para poder ejecutar maven desde la consola.

Manejo de dependencias con Maven

Instalación:

La primera vez que ejecutemos maven, creará un repositorio local en tu disco duro.

En concreto, creará la carpeta .m2 en la carpeta home del usuario. En ella se guardarán todos los artefactos que maneje maven.



Manejo de dependencias con Maven

Grupos y artefactos

Un artefacto es un componente de software que podemos incluir en un proyecto como dependencia. Normalmente será un jar, pero podría ser de otro tipo, como un war por ejemplo. Los artefactos pueden tener dependencias entre sí, por lo tanto, si incluimos un artefacto en un proyecto, también obtendremos sus dependencias.

Manejo de dependencias con Maven

Grupos y artefactos

Un grupo es un conjunto de artefactos. Es una manera de organizarlos. Así por ejemplo todos los artefactos de Spring Framework se encuentran en el grupo org.springframework.

Esta es la manera de declarar una dependencia de nuestro proyecto con un artefacto. Se indica el identificador de grupo, el identificador del artefacto y la versión.

```
<dependency>
    <groupid>org.springframework</groupid>
    <artifactid>spring-orm</artifactid>
    <version>3.0.5.RELEASE</version>
    <scope>runtime</scope>
</dependency>
```

Manejo de dependencias con Maven

Scope (alcance): sirve para indicar el alcance de nuestra dependencia y su transitividad. Hay 6 tipos:

compile: es el alcance que tenemos por defecto si no especificamos scope. Indica que **la dependencia es necesaria para compilar**. La dependencia además se propaga en los proyectos dependientes.

provided: Es como la anterior, pero espera que el contenedor ya tenga esa librería. Un claro ejemplo es cuando desplegamos en un servidor de aplicaciones, que por defecto, tiene bastantes librerías que utilizaremos en el proyecto, así que no necesitamos desplegar la dependencia.

Manejo de dependencias con Maven

runtime: La dependencia es necesaria en tiempo de ejecución pero no es necesaria para compilar.

test: La dependencia es solo para testing que es una de las fases de compilación con maven. JUnit es un claro ejemplo de esto.

system: Es como **provided** pero tienes que incluir la dependencia **explícitamente**. Maven no buscará este artefacto en tu repositorio local. Habrá que especificar la ruta de la dependencia mediante la etiqueta <systemPath>

import: este solo se usa en la sección **dependencyManagement**.

Manejo de dependencias con Maven

En nuestro proyecto podemos indicar que necesitamos un **jar** (por ejemplo, log4j o el conector de MySQL) y **Maven es capaz de buscar esos jar en internet y descargarlos automáticamente**. Incluso si alguno de esos jar necesita de otros jar para funcionar, Maven "tira del hilo" y va descargando todos los jar que sean necesarios.

Las dependencias se recopilan en el archivo **pom.xml**, dentro de una etiqueta **<dependencies>**.

Un repositorio en Maven contiene artefactos de compilación y dependencias de diferentes tipos.

Repositorios Maven: locales y remotos

Repositorio Local

El repositorio local de Maven es una parte crítica de su funcionamiento y, aunque a menudo pasa desapercibido, es fundamental para la eficiencia en el desarrollo.

El repositorio local es una carpeta ubicada en la máquina del desarrollador, por defecto en la carpeta **.m2** en el directorio de inicio del usuario. Este repositorio sirve como almacén local de todas las dependencias descargadas y los artefactos generados por los proyectos.

Repositorio Local

La función principal del repositorio local es evitar la descarga repetida de dependencias. Cuando un proyecto necesita una biblioteca que ya ha sido descargada previamente, Maven la tomará del repositorio local en lugar de descargarla nuevamente. Esto no solo ahorra tiempo, sino que también ayuda a reducir la carga en los servidores remotos.

Repositorio Local

El repositorio local **es específico para cada usuario** y cada proyecto tiene su propio directorio dentro de este repositorio, lo que garantiza que las dependencias sean almacenadas de manera aislada para cada proyecto. Esto evita conflictos entre proyectos y garantiza que cada proyecto tenga acceso a sus propias versiones de las bibliotecas.

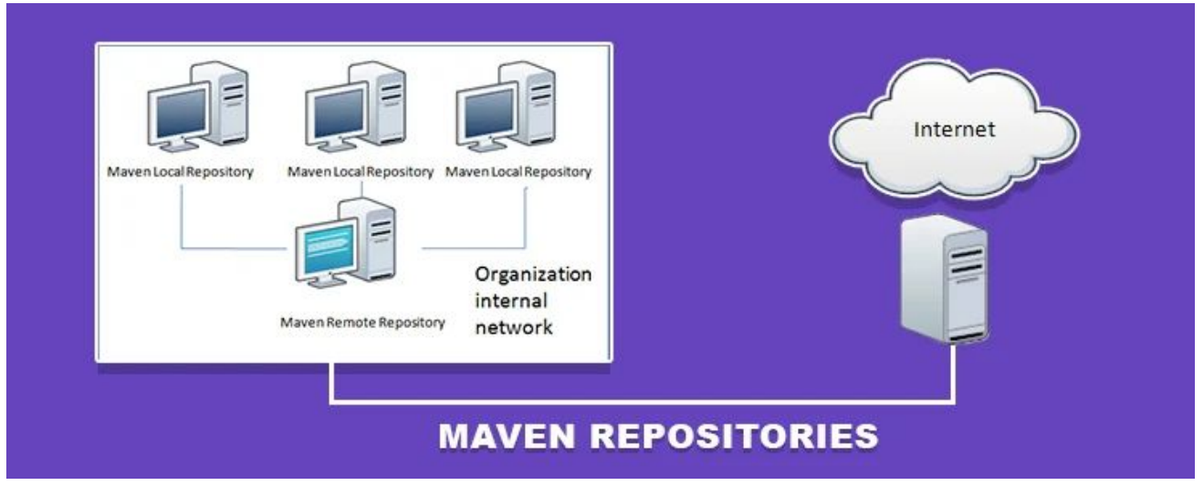
Repositorio Local

Los repositorios locales y remotos están estructurados de la misma manera para que los scripts se puedan ejecutar en cualquier lado o se puedan sincronizar para su uso **sin conexión**. Sin embargo, el diseño de los repositorios es completamente transparente para el usuario de Maven.

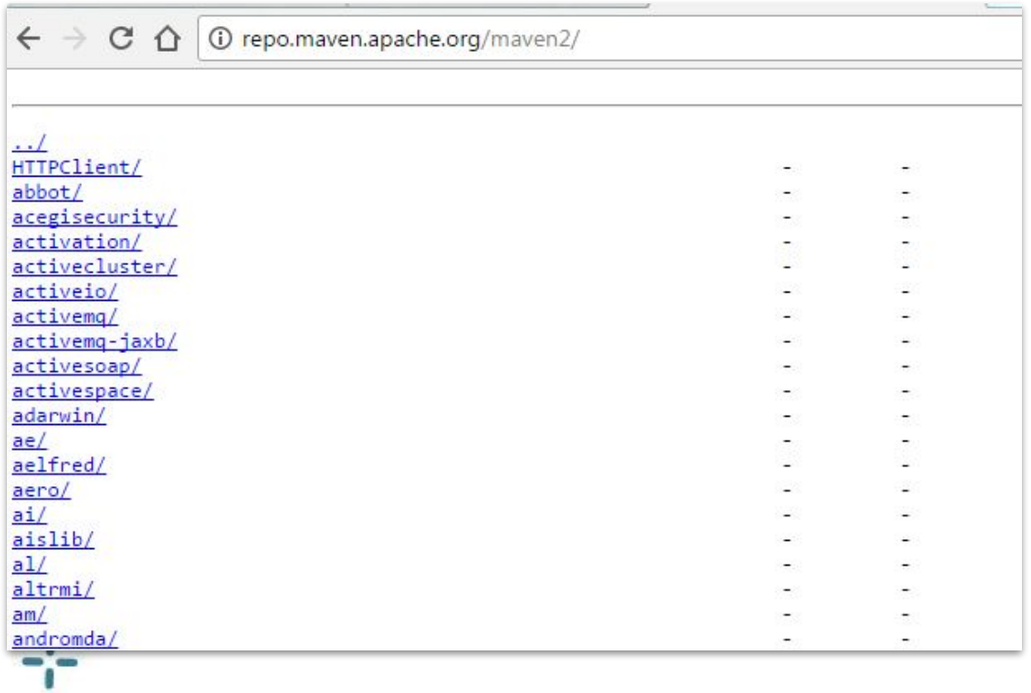
Repositorio remoto

El **Repositorio Central Maven**, también conocido como el repositorio Maven Central, es un recurso esencial para los desarrolladores de Java.

Es un repositorio público y remoto que almacena miles de bibliotecas de código abierto disponibles para su uso en proyectos Maven. Está accesible a través de Internet en https://repo.maven.apache.org/maven2/



Repositorio remoto



Este repositorio es mantenido por la comunidad Maven y es una fuente invaluable de bibliotecas de código abierto listas para ser integradas en tus proyectos.

Repositorio remoto

Cuando declaras una dependencia en tu archivo pom.xml, **Maven buscará primero en tu repositorio local, y si no encuentra la dependencia, buscará en el Repositorio Central Maven**. Si la dependencia está disponible, la descargará automáticamente y la almacenará en tu repositorio local para futuros usos. Esto significa que, en la mayoría de los casos, no es necesario preocuparse por la descarga y gestión manual de bibliotecas.

Repositorio remoto

Además del Repositorio Central Maven, e**s posible configurar Maven para utilizar repositorios personalizados o privados**. Esto es útil cuando necesitas bibliotecas específicas para tu organización o proyecto que no están disponibles en el Repositorio Central.

LIVE CODING

Ejemplo en vivo

Local vs Remoto:

En este ejemplo en vivo, veremos la carpeta .m2 para reconocer las dependencias instaladas en el repositorio local.

Luego, vamos a buscar dependencias remotas en el Maven Central para reconocer diferentes dependencias que suelen utilizarse con proyectos Spring: conector mysql, spring web, spring jpa.

Tiempo: 15 minutos

Evaluación Integradora ✨

¿Listos para un nuevo desafío? En esta clase comenzamos a construir nuestro....

Trabajo Integrador del Módulo 💪🏽

Iremos completándolo progresivamente clase a clase.

English Always | alkemy

Ejercicio N° 1

En remoto

En remoto

Manos a la obra: 🙌

En este ejercicio te invitamos a recorrer el repositorio central de Maven y elegir 5 dependencias que consideres serían necesarias para implementar en un proyecto de Billetera Virtual.

Consigna: ✍️

1- Dirigirse al Repositorio Central de Maven

2- Buscar dependencias que consideres importantes para un proyecto de Wallet

3- Realizar una puesta en común y evaluar juntos a los demás compañeros cuántas dependencias coincidieron

Tiempo🕐: *15 minutos*

¿Alguna consulta?

RESUMEN

¿Qué logramos en esta clase?

- ✓ *Comprender el concepto de dependencias*
- ✓ *Aprender las diferencias entre repositorio local y remoto*

English Always | alkemy

#WorkingTime

Continuemos ejercitando

¡Antes de cerrar la clase! Te invitamos a: 👇👇👇

1. Repasar nuevamente la grabación de esta clase
2. Revisar el material compartido en la plataforma de Moodle (lo que se vio en clase y algún ejercicio adicional)
    a. *Lectura Modulo 6, Lección 1: páginas 5 - 6*
3. Traer al próximo encuentro, todas tus dudas y consultas para verlas antes de iniciar nuevo tema.

¡Muchas Gracias!

Nos vemos en la próxima clase 👋🏻

English Always | alkemy

Momento:

Time-out!

⏰5 min.