



M4S1 CONSTRUYE BASES DE DATOS PARA APLICACIONES WEB

PRACTICA “HOLA MUNDO”

Esta es la guía definitiva para empezar con IntelliJ IDEA correctamente. La "mejor práctica" no es solo instalarlo, sino **entender por qué se hacen ciertos pasos** y adoptar buenos hábitos desde el primer día.

Aquí tienes el paso a paso, diseñado para principiantes.

Fase 0: El Pre-requisito (Lo que necesitas ANTES)

La práctica más importante de todas es entender esto: **IntelliJ IDEA no es Java**.

- **IntelliJ IDEA** es el taller (el IDE).
- **Java (JDK)** son las herramientas y la materia prima (el lenguaje y el kit de desarrollo).

Necesitas instalar las herramientas (el JDK) antes de organizar tu taller (IntelliJ).

1. Descarga el JDK (Java Development Kit):

- Ve al sitio oficial de **Eclipse Temurin (Adoptium)** o **Oracle**.
- **Mejor Práctica:** Descarga una versión **LTS (Long-Term Support)**. Las más comunes y recomendadas actualmente son **Java 17** o **Java 21**.

2. Instala el JDK:

- Sigue el instalador. Es muy sencillo. Asegúrate de que durante la instalación marques la opción que dice "**Set JAVA_HOME variable**" (o "Establecer variable JAVA_HOME"). Esto le dice a tu computadora dónde encontrar Java.

Con el JDK instalado, ahora sí puedes instalar el IDE.

Fase 1: Instalación de IntelliJ IDEA

1. Elige tu Versión (Community vs. Ultimate):

- Ve al sitio web oficial de **JetBrains**.
- Verás dos opciones:
 - **Ultimate (De pago):** Para desarrollo web avanzado (Spring, JavaScript, bases de datos) y empresarial.
 - **Community (Gratuita):** De código abierto. Es **perfecta** para empezar y para todo el desarrollo de Java, Kotlin y Android.
- **Mejor Práctica: Descarga la versión Community.** Es gratuita para siempre y tiene todo lo que necesitas (y más) para aprender.

2. Ejecuta el Instalador:

- Abre el archivo descargado.

M4S1 CONSTRUYE BASES DE DATOS PARA APLICACIONES WEB

- **Mejor Práctica (Opciones de Instalación):** Cuando te pregunte, marca estas casillas. Son muy útiles:
 - **Crear un acceso directo en el escritorio.**
 - **Agregar "Abrir Carpeta como Proyecto":** (Muy importante). Te permite hacer clic derecho en una carpeta y abrirla como un proyecto.
 - **Crear asociaciones:** Marca .java, .kt, .groovy (para que estos archivos se abran con IntelliJ por defecto).
 - **Agregar "bin" al PATH:** (Recomendado).

Fase 2: Configuración Inicial (Tu Primer Lanzamiento)

La primera vez que abras IntelliJ, te hará algunas preguntas.

1. **Importar Ajustes:** Te preguntará si quieras importar configuraciones. Como es tu primera vez, selecciona "**Do not import settings**" (No importar configuraciones).
2. **Tema (Theme):** Elige el que más te guste. El tema oscuro (Dark) se llama "**Darcula**" y es el más popular.
3. **Plugins:** Te sugerirá instalar *plugins* (pequeños programas que añaden funciones).
 - **Mejor Práctica: No instales nada todavía.** Es mejor empezar con el IDE limpio. Haz clic en "Skip Remaining and Set Defaults" (Omitir y usar valores predeterminados).

Fase 3: Tu Primer Proyecto ("Hola, Mundo")

Aquí es donde la mayoría se confunde. Vamos a hacerlo de la forma correcta.

1. **Crear Nuevo Proyecto:**
 - En la pantalla de bienvenida, haz clic en "**New Project**" (Nuevo Proyecto).
2. **Configurar el Proyecto (La parte clave):**
 - **Name (Nombre):** Escribe un nombre, por ejemplo: MiPrimeraApp.
 - **Location (Ubicación):** Elige dónde guardarlo.
 - **Language (Lenguaje):** Selecciona **Java**.
 - **Build System (Sistema de Construcción):** Para empezar, selecciona **IntelliJ**. (Más adelante aprenderás a usar Maven o Gradle, que son estándar en la industria).
 - **JDK:** ¡Este es el paso vital! Aquí debes decirle a IntelliJ dónde está el JDK que instalaste en la Fase 0.
 - Normalmente, IntelliJ lo detecta automáticamente (ej: Temurin-17).

M4S1 CONSTRUYE BASES DE DATOS PARA APLICACIONES WEB

- Si no aparece, haz clic en el menú desplegable, selecciona "Add JDK..." y busca la carpeta donde lo instalaste.
 - **Mejor Práctica:** Marca la casilla que dice "**Add sample code**" (Agregar código de ejemplo). Esto creará automáticamente tu primer "Hola, Mundo".
3. ¡Crear!
- Haz clic en "Create".

Fase 4: Entendiendo la Interfaz (El Primer Vistazo)

El IDE se abrirá y puede ser abrumador. No te preocunes, solo necesitas saber tres cosas:

1. **El Panel de Proyecto (Izquierda):** Aquí ves la estructura de tus archivos. Tu código estará en src > main > java > com.example > Main.java.
2. **El Editor (Centro):** Aquí es donde escribes tu código.
3. **La Barra de Progreso (Abajo):** La primera vez que abres un proyecto, IntelliJ necesita "**Indexar**" tus archivos.
 - **Mejor Práctica: Siempre espera a que termine de indexar.** Esta indexación es lo que hace que el autocompletado y la detección de errores sean tan "inteligentes".

Fase 5: Ejecutando tu Código

1. Abre tu archivo Main.java (haciendo doble clic en él, en el panel izquierdo).
2. Verás el código de ejemplo (public static void main(String[] args)).
3. Busca el **triángulo verde (botón de Play)** que aparece junto a la línea public static void main....
4. Haz clic en él y selecciona "**Run 'Main.main()'**".
5. ¡Felicidades! ¡Verás una nueva ventana en la parte inferior (la "Terminal" o "Run") que muestra el texto "Hello World!".

Mejores Prácticas Esenciales desde el Día 1

1. **Aprende Alt + Enter (La Varita Mágica):**
 - Este es el atajo de teclado más importante.
 - Cuando IntelliJ subraya algo (en rojo para errores, en amarillo para sugerencias), **pon el cursor sobre ello y presiona Alt + Enter**.
 - IntelliJ no solo te dirá qué está mal, sino que te ofrecerá **soluciones automáticas**. Úsalo. Así es como el IDE te enseña a programar mejor.
2. **Aprende a Navegar (Ctrl + Clic):**

M4S1 CONSTRUYE BASES DE DATOS PARA APLICACIONES WEB

- Para entender cómo funciona un programa, mantén presionada la tecla Ctrl y haz clic en cualquier nombre de variable, método o clase.
- IntelliJ te llevará instantáneamente a su definición.

3. Usa el Autocompletado (Ctrl + Espacio):

- No escribas todo el código. Escribe las primeras letras y presiona Ctrl + Espacio. IntelliJ te dará una lista de opciones inteligentes.

4. No le temas a la Refactorización (Shift + F6):

- ¿Quieres cambiar el nombre de una variable? No lo hagas manualmente.
- Haz clic sobre ella y presiona Shift + F6. Escribe el nuevo nombre y el IDE lo cambiará de forma segura en todos los lugares donde se usaba.

La clave de IntelliJ IDEA es dejar que te ayude. No luches contra él; confía en sus sugerencias (especialmente las amarillas) y usa Alt + Enter para todo.

DIFERENCIA ENTRE LOS SISTEMAS DE CONSTRUCCIÓN "MAVEN" Y "GRADLE"

Tanto Maven como Gradle son herramientas de construcción (o *build tools*).

Piensa en ellas como el "jefe de obra" de tu proyecto de software. Tu código son solo los ladrillos, pero el jefe de obra sabe:

1. Qué necesitas: ¿Necesitas "ladrillos especiales" de otra fábrica? (Estas son las dependencias o librerías externas).
2. Cómo construir: ¿En qué orden se debe armar todo?
3. Qué entregar: ¿Cómo se debe empaquetar el producto final? (Por ejemplo, en un archivo .jar o .war).

Ambos hacen el mismo trabajo, pero tienen filosofías muy diferentes.

Maven

- Archivo de configuración: pom.xml
- Lenguaje: XML (Un lenguaje de etiquetas, muy estructurado).
- Filosofía: "Convención sobre Configuración".

Maven es el más antiguo y establecido. Su filosofía significa que *asume* que tu proyecto sigue una estructura de carpetas estándar (ej. src/main/java).

Si sigues sus reglas (convenciones), solo necesitas escribir una configuración mínima. Es muy rígido, pero eso lo hace predecible.

M4S1 CONSTRUYE BASES DE DATOS PARA APLICACIONES WEB

Analogía: Es como construir un mueble de IKEA. Te da un manual de instrucciones (el pom.xml) muy detallado y verboso. No puedes ser muy creativo, pero si sigues los pasos, *sabes* que el resultado funcionará.

XML

```
<project>

<modelVersion>4.0.0</modelVersion>

<groupId>com.miempresa</groupId>

<artifactId>mi-app</artifactId>

<version>1.0</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>
```

Gradle

- Archivo de configuración: build.gradle (o build.gradle.kts)
- Lenguaje: Groovy o Kotlin (Lenguajes de *scripting*).
- Filosofía: Flexibilidad y Rendimiento.

Gradle es más moderno. En lugar de un archivo de configuración XML estático, usas un *script* de código. Esto te da un poder y una flexibilidad enormes para crear tareas de construcción personalizadas.

Además, es mucho más rápido que Maven gracias a su sistema de *caché* inteligente (no reconstruye cosas que no han cambiado). Es la herramienta estándar para el desarrollo de Android.

Analogía: Es como contratar a un carpintero experto. En lugar de un manual, le das un boceto (el build.gradle) que es mucho más corto y conciso. Él usará su experiencia (el *scripting*) para construirlo de la forma más eficiente posible.

M4S1 CONSTRUYE BASES DE DATOS PARA APLICACIONES WEB

Groovy

```
// Ejemplo simple de un build.gradle (en Groovy)

plugins {
    id 'java'
}

group = 'com.miempresa'
version = '1.0'

repositories {
    mavenCentral()
}

dependencies {
    testImplementation 'junit:junit:4.13.2'
}
```

Comparativa Rápida

Característica	Maven	Gradle
Sintaxis	XML (Verboso, muy estructurado)	Groovy / Kotlin (Conciso, como código)
Flexibilidad	Baja (Rígido, basado en convenciones)	Alta (Muy personalizable)
Rendimiento	Bueno (Pero reconstruye todo a menudo)	Excelente (Usa caché avanzada)
Curva de Aprendizaje	Fácil al inicio, pero se complica	Un poco más difícil al inicio (es código)
Popularidad	Muy usado en proyectos <i>Enterprise</i> de Java	Estándar en Android y popular en startups

M4S1 CONSTRUYE BASES DE DATOS PARA APLICACIONES WEB

¿Cuál deberías usar para empezar?

- Si tu objetivo es aprender desarrollo de Android: Usa Gradle desde el día uno. Es el estándar y no tienes opción.
- Si tu objetivo es aprender Java "tradicional" (para empresas): Empezar con Maven es una apuesta segura. Es extremadamente común en el mundo corporativo y su rigidez te ayuda a entender la estructura de un proyecto estándar.

La mayoría de los desarrolladores de Java terminan aprendiendo ambos. IntelliJ IDEA los maneja a los dos de manera brillante.

¿CÓMO CREAR UN PROYECTO "HOLA MUNDO" USANDO MAVEN DIRECTAMENTE EN INTELLIJ?

Esta es la forma estándar en que se inician los proyectos en el mundo profesional. Verás que es muy similar, pero con un paso clave diferente.

Paso 1: Abrir el Asistente de Nuevo Proyecto

1. Abre IntelliJ IDEA.
2. En la pantalla de bienvenida, haz clic en "**New Project**" (Nuevo Proyecto).

Paso 2: Configurar el Proyecto (El Paso Clave)

Esta es la pantalla más importante. Rellénala de la siguiente manera:

1. **Name (Nombre):** Escribe el nombre. Por ejemplo: HolaMundoMaven.
2. **Location (Ubicación):** Elige dónde guardarlo.
3. **Language (Lenguaje):** Selecciona **Java**.
4. **Build System (Sistema de Construcción):** Aquí está el cambio. En lugar de "IntelliJ", selecciona **Maven**.
5. **JDK:** Asegúrate de que esté seleccionado tu JDK (ej. Temurin-17).
6. **Add sample code (Agregar código de ejemplo):** ¡Marca esta casilla! Es la forma más fácil de empezar.

Nota sobre GroupId y ArtifactId: Al seleccionar Maven, es posible que veas una sección opcional de "Advanced Settings" (Configuración Avanzada).

- **GroupId:** Es el identificador de tu organización. Se escribe como un dominio al revés. Para practicar, puedes poner: com.tuejemplo.
- **ArtifactId:** Es el nombre de tu proyecto (normalmente coincide con el "Name" que pusiste arriba, HolaMundoMaven).

Si no lo ves, no te preocupes, IntelliJ usará valores predeterminados.

M4S1 CONSTRUYE BASES DE DATOS PARA APLICACIONES WEB

Haz clic en "**Create**" (Crear).

Paso 3: La Sincronización de Maven

Esto es nuevo. Cuando el proyecto se abre, mira la esquina inferior derecha.

- IntelliJ IDEA leerá tu nuevo archivo pom.xml.
- Verás una barra de progreso que dice "**Resolving dependencies**" (Resolviendo dependencias) o "**Syncing**" (Sincronizando).
- Maven está configurando el proyecto según las reglas del pom.xml y descargando de Internet cualquier cosa que necesite (en este caso, casi nada, pero es su trabajo).

Mejor Práctica: Siempre espera a que esta "indexación" o "sincronización" termine antes de empezar a escribir código.

Paso 4: Explorar la Estructura (¡Es Diferente!)

Verás que el panel de proyecto a la izquierda se ve más profesional.

- **HolaMundoMaven** (Tu proyecto)
 - **.idea** (Archivos de configuración de IntelliJ. Puedes ignorarlos).
 - **src** (Source o código fuente)
 - **main/java**: Aquí es donde vivirán tus archivos .java.
 - com.tuejemplo.Main.java (Este es el archivo de ejemplo que pedimos).
 - **test/java**: (Aquí es donde escribirías tus pruebas).
 - **pom.xml**: Este es el cerebro de tu proyecto Maven.

Esta estructura (src/main/java) es la "convención" de la que hablamos. Maven la crea por ti.

Paso 5: Ejecutar el Código

1. Haz doble clic en el archivo pom.xml para abrirlo. Verás que es un archivo XML con el groupId y artifactId que definiste.
2. Ahora, ve a src/main/java/com/tuejemplo/Main.java y ábrelo.
3. Verás el código de ejemplo public static void main(String[] args).
4. Igual que antes, busca el **triángulo verde (Play)** junto a la línea public static void main.
5. Haz clic y selecciona "**Run 'Main.main()'**".

¡El resultado es el mismo! Verás "Hello, world!" en la consola.



M4S1 CONSTRUYE BASES DE DATOS PARA APLICACIONES WEB

¿Cuál es la diferencia?

Has hecho exactamente lo mismo (un "Hola, Mundo"), pero esta vez **Maven estaba al mando** de la estructura del proyecto y del proceso de construcción. IntelliJ IDEA simplemente actuó como un editor inteligente que sabe cómo "hablar" con Maven.

Ahora tienes un proyecto listo para el siguiente paso: añadir librerías externas.

USAR MAVEN PARA AGREGAR UNA LIBRERÍA EXTERNA POPULAR (COMO UNA LIBRERÍA PARA TRABAJAR CON JSON) A ESTE PROYECTO

Vamos a añadir una librería externa. El ejemplo perfecto es **Gson**, una librería de Google que convierte objetos de Java en texto **JSON** (un formato de texto estándar para enviar datos).

Sigue estos pasos en el proyecto HolaMundoMaven que acabamos de crear.

Paso 1: Encontrar la Dependencia

No vamos a descargar un archivo .jar. Simplemente le diremos a Maven "el nombre" de lo que queremos.

1. Abre tu navegador y busca: "**Gson Maven dependency**".
2. El primer resultado será probablemente del sitio "**Maven Central Repository**" (o "mvnrepository.com"). Entra ahí.
3. Haz clic en la versión más reciente (ej. 2.10.1 o similar).
4. Verás un recuadro de texto con un fragmento de código **XML**. Cópialo. Se verá así:

XML

```
<dependency>
  <groupId>com.google.code.gson</groupId>
  <artifactId>gson</artifactId>
  <version>2.10.1</version>
</dependency>
```

Paso 2: Añadir la Dependencia al pom.xml

1. En IntelliJ, abre tu archivo pom.xml.
2. Busca una sección llamada <dependencies>. Si no existe, puedes crearla (pero lo más probable es que ya exista una, quizás con la librería junit dentro).

M4S1 CONSTRUYE BASES DE DATOS PARA APLICACIONES WEB

3. **Pega el código** que copiaste dentro de las etiquetas <dependencies>.

El resultado debería verse más o menos así:

XML

```
<project ...>
```

```
...
```

```
<properties>
```

```
  <maven.compiler.source>17</maven.compiler.source>
```

```
  <maven.compiler.target>17</maven.compiler.target>
```

```
</properties>
```

```
<dependencies>
```

```
  <dependency>
```

```
    <groupId>com.google.code.gson</groupId>
```

```
    <artifactId>gson</artifactId>
```

```
    <version>2.10.1</version>
```

```
  </dependency>
```

```
</dependencies>
```

```
...
```

```
</project>
```

Paso 3: Sincronizar el Proyecto

Esto es crucial. Justo después de guardar el archivo pom.xml:

1. Aparecerá un pequeño ícono de **Maven** (una "M") en la esquina superior derecha del editor, o un mensaje flotante.
2. Haz clic en ese ícono (dice "Load Maven Changes" o "Import Changes").
3. ¡Mira la magia! En la barra inferior, verás que IntelliJ y Maven están **descargando la librería "Gson"** y todas sus sub-dependencias desde Internet.

Una vez que termine, la librería gson-2.10.1.jar ya es parte de tu proyecto.

Paso 4: Usar la Librería en tu Código

Ahora, vamos a Main.java para usar la librería que acabamos de importar.

M4S1 CONSTRUYE BASES DE DATOS PARA APLICACIONES WEB

1. Abre tu archivo Main.java (está en src/main/java/com/tuejemplo/Main.java).
2. Borra el contenido de "Hello, world!" y reemplaza el archivo Main.java con este código:

Java

```

package com.tuejemplo;

// 1. Importamos la librería que acabamos de descargar

import com.google.gson.Gson;

import java.util.Map;

public class Main {

    public static void main(String[] args) {

        // 2. Creamos un objeto simple (un Mapa) para convertir

        Map<String, Object> persona = Map.of(
            "nombre", "Ana",
            "edad", 25,
            "esDesarrolladora", true
        );

        // 3. Creamos una instancia de la clase "Gson" de la librería Gson gson = new Gson();

        // 4. Usamos el método .toJson() para convertir el mapa a un string JSON String jsonString
        // = gson.toJson(persona);

        // 5. Imprimimos el resultado

        System.out.println("¡Objeto de Java convertido a JSON!");

        System.out.println(jsonString);

    }
}

```

Nota: Fíjate cómo import com.google.gson.Gson; ya no da error. Esto es porque IntelliJ sabe dónde encontrar ese código gracias a Maven.

Paso 5: Ejecutar y Ver el Resultado

Haz clic en el **botón verde de "Play"** para ejecutar Main.main().

M4S1 CONSTRUYE BASES DE DATOS PARA APLICACIONES WEB

En la consola, ya no verás "Hello, world!". Verás el resultado de la librería que acabas de importar:

¡Objeto de Java convertido a JSON!

```
{"nombre":"Ana","edad":25,"esDesarrolladora":true}
```

¡Felicitaciones! Acabas de usar Maven para gestionar dependencias, que es la tarea número uno de un desarrollador de Java en el día a día.

GUARDAR TU TRABAJO USANDO GIT DIRECTAMENTE DESDE INTELLIJ IDEA

Git es un **sistema de control de versiones**. Es como un "historial de guardado" súper avanzado para tu código. Te permite tomar "fotos" (llamadas **commits**) de tu proyecto en cualquier momento, ver quién cambió qué, y volver a una versión anterior si algo sale mal.

IntelliJ IDEA tiene una de las mejores integraciones con Git del mercado. Vamos a configurar tu proyecto HolaMundoMaven.

Fase 1: Prerrequisito (Instalar Git)

Al igual que con el JDK, IntelliJ no es Git; es una interfaz gráfica para *usar* Git.

1. **Verifica si tienes Git:** Abre una terminal (CMD o PowerShell en Windows) y escribe git --version. Si te da un número de versión, ya lo tienes.
2. **Si no lo tienes:** Ve a git-scm.com y descarga el instalador para tu sistema operativo. Instálalo con todas las opciones por defecto.

Fase 2: Convertir tu Proyecto en un Repositorio Git

Ahora mismo, tu proyecto HolaMundoMaven no está bajo control de versiones. Vamos a cambiar eso.

1. En IntelliJ, con tu proyecto abierto, ve al menú superior.
2. Haz clic en **VCS** (Version Control System).
3. Selecciona **Enable Version Control Integration...** (Habilitar integración de control de versiones...).
4. En el menú desplegable, elige **Git** y presiona OK.

¿Qué acaba de pasar?

- Git ha creado un repositorio *local* en la carpeta de tu proyecto (ha creado una carpeta oculta llamada .git).

M4S1 CONSTRUYE BASES DE DATOS PARA APLICACIONES WEB

- ¡Mira tus archivos! En el panel de proyecto, todos tus archivos se habrán puesto de **color rojo**. Esto significa que Git los ve, pero aún no están siendo rastreados (son "untracked").

Fase 3: La "Primera Foto" (Initial Commit)

Nuestro objetivo es guardar el estado actual del proyecto. Esto se hace en dos pasos: "preparar" (Stage) y "guardar" (Commit).

Paso 3.1: El .gitignore (La Mejor Práctica N° 1)

¡CRÍTICO! No queremos guardar *todo* en el historial. Hay archivos que IntelliJ o Maven generan automáticamente (como las carpetas target o .idea) que no deben guardarse.

Le decimos a Git que los ignore creando un archivo especial:

1. En el panel de proyecto, haz clic derecho sobre la raíz de tu proyecto (HolaMundoMaven).
2. Selecciona **New > File**.
3. Nombra el archivo exactamente: **.gitignore** (con el punto al inicio).
4. IntelliJ te preguntará si quieres añadir *plugins* de .gitignore. Acepta e instálalos si te lo sugiere.
5. Pega el siguiente contenido dentro de tu archivo .gitignore. Esta es una plantilla estándar de Java/Maven:

Plaintext

```
# Archivos de IntelliJ
.idea/
*.iml

# Archivos de Maven
target/

# Archivos del sistema operativo
.DS_Store
Thumbs.db

# Logs
*.log
```

M4S1 CONSTRUYE BASES DE DATOS PARA APLICACIONES WEB

Ahora fíjate: las carpetas .idea y target deberían volverse de color **marrón** o **gris**. Eso significa que Git las está ignorando correctamente.

Paso 3.2: Hacer el Commit

1. Ve al menú **Git** en la parte superior, o busca la **pestaña Commit** (generalmente a la izquierda o arriba, con un tic ✓ verde).
2. Se abrirá una ventana. Verás una lista de todos tus archivos "nuevos" (en color rojo, que se volverá verde al seleccionarlos).
3. **Selecciona todos los archivos** que quieras guardar (deberían ser pom.xml, .gitignore y toda tu carpeta src).
4. En el campo "**Commit Message**" (Mensaje de Commit), escribe una descripción clara de lo que hiciste. Para el primero, el estandar es:

Initial commit

5. Haz clic en el botón **Commit**.

¡Felicitaciones! Acabas de tomar tu primera "foto" (commit). Tus archivos ahora se pondrán de **color blanco**, lo que significa que están guardados y no tienen cambios pendientes.

Fase 4: Hacer un Cambio y Guardarlo

Ahora veamos el flujo de trabajo normal.

1. **Haz un cambio:** Ve a tu archivo Main.java.
2. Añade un comentario. Por ejemplo, en la línea System.out.println(jsonString); añade arriba:

Java

```
// Este es un cambio para probar Git
System.out.println(jsonString);
```

3. **Observa:** El nombre del archivo Main.java en el panel de proyecto se habrá puesto de **color azul**. Esto significa "modificado".
4. **Haz Commit:**
 - Abre la ventana de **Commit** (o presiona Ctrl + K).
 - Verás tu archivo Main.java en la lista.
 - Escribe un mensaje descriptivo: Agregado comentario de prueba
 - Haz clic en **Commit**.

El archivo volverá a ser blanco.

M4S1 CONSTRUYE BASES DE DATOS PARA APLICACIONES WEB

Fase 5: Ver el Historial

¿Dónde están todas mis "fotos"?

1. Busca la pestaña **Git** en la barra inferior de IntelliJ.
2. Selecciona la sub-pestaña **Log** (Historial).

Verás un historial de todos tus *commits*. Puedes hacer clic en cada uno para ver exactamente qué archivos cambiaron y qué líneas se añadieron o quitaron.

Resumen del Flujo de Trabajo

1. **Codificas** (los archivos se ponen azules).
2. **Haces Commit** (los archivos se guardan en tu historial *local* y vuelven a ser blancos).
3. Repites.

Todo lo que has hecho hasta ahora está **solo en tu computadora** (un repositorio *local*).

¿COMO CONECTAR ESTO A UN SERVICIO EN LA NUBE COMO GITHUB O GITLAB, PARA TENER UNA COPIA DE SEGURIDAD Y PODER COLABORAR CON OTROS?

Vamos a subir tu proyecto HolaMundoMaven a un nuevo repositorio en tu cuenta de GitHub.

Fase 1: Prerrequisito - Tu Cuenta de GitHub

Para este paso, solo necesitas una cosa: **Tener una cuenta de GitHub**.

- Si no tienes una, ve a github.com y crea una cuenta gratuita.
- **No** necesitas crear el repositorio en la web; IntelliJ lo hará por ti.

Fase 2: Conectar IntelliJ con tu Cuenta de GitHub

Primero, tienes que "presentarle" IntelliJ a tu cuenta de GitHub para que confíe en él. Solo se hace una vez.

1. En IntelliJ, ve al menú superior: File > Settings (en Windows) o IntelliJ IDEA > Preferences (en macOS).
2. Busca la sección: Version Control > GitHub.
3. Haz clic en el icono + (Añadir cuenta).

M4S1 CONSTRUYE BASES DE DATOS PARA APLICACIONES WEB

4. La forma más fácil y segura es seleccionar **Log In via GitHub...** (Iniciar sesión vía GitHub...).
5. Esto abrirá tu navegador web. Te pedirá que inicies sesión en GitHub (si no lo has hecho) y luego que **autorices** a "JetBrains IDE Integration".
6. Haz clic en "Authorize" (Autorizar).
7. Una vez que el navegador te dé el "OK", vuelve a IntelliJ. Verás tu cuenta de GitHub listada. ¡Ya están conectados!

Fase 3: "Compartir" tu Proyecto en GitHub

Esta es la forma más fácil. En lugar de crear un repositorio en la web y conectarlo manualmente, usaremos una acción de IntelliJ que lo hace todo en un solo paso.

1. En el menú superior de IntelliJ, ve a Git > GitHub > **Share Project on GitHub...** (Compartir proyecto en GitHub...).
2. Aparecerá una nueva ventana.
 - **Repository name (Nombre):** Se llenará automáticamente con HolaMundoMaven. Déjalo así.
 - **Private (Privado):** !! ¡MUY IMPORTANTE! Marca esta casilla. Esto asegura que solo tú puedes ver este repositorio de práctica. (Si lo dejas desmarcado, será público).
 - **Description (Descripción):** Escribe algo si quieras, como "Mi primer proyecto Java con Maven y Git". (Es opcional).
3. Haz clic en **Share (Compartir).**

Fase 4: ¿Qué acaba de pasar?

En este momento, IntelliJ está haciendo tres cosas por ti:

1. **Creó** un nuevo repositorio (privado) llamado HolaMundoMaven en tu cuenta de GitHub.
2. **Configuró** tu proyecto local para que sepa que su "destino" remoto (llamado origin) es ese nuevo repositorio.
3. **Subió** (hizo push) todos los *commits* (fotos) que tenías en tu historial local (tu "Initial commit" y el del comentario) al servidor de GitHub.

Fase 5: ¡Verifícalo!

M4S1 CONSTRUYE BASES DE DATOS PARA APLICACIONES WEB

Cuando IntelliJ termine, verás una pequeña notificación de éxito en la esquina inferior derecha. A menudo, **esa notificación incluye un enlace azul con el nombre de tu repositorio.**

1. **Haz clic en ese enlace.**
2. Se abrirá tu navegador y te llevará directamente a tu nuevo repositorio en GitHub.
3. **Verás tu código allí:** tu archivo pom.xml, tu .gitignore y tu carpeta src.

¡Felicitaciones! Tu código ya está seguro en la nube y has completado el ciclo completo del desarrollador.

El Flujo de Trabajo Diario

A partir de ahora, tu rutina diaria será:

1. **Programar** (los archivos se ponen azules).
2. **Hacer Commit** (guardas la "foto" en tu historial *local*).
3. **Hacer Push** (subir tus nuevos *commits* a GitHub).

El paso 3 es nuevo.

COMANDO PUSH (PARA SUBIR) Y PULL (PARA BAJAR CAMBIOS)

Vamos a dividirlo en dos partes:

1. **PUSH (Subir):** Cuando haces cambios en tu PC y quieres subirlos a GitHub.
2. **PULL (Bajar):** Cuando alguien de tu equipo (o tú mismo desde otra PC) sube cambios a GitHub y tú quieres bajarlos a tu PC.

1. El Flujo de "Subir" (PUSH): De tu PC a GitHub

Sigue estos 3 pasos (Cambio -> Commit -> Push).

Paso 1: Haz un Cambio

Ve a tu archivo Main.java y haz una modificación. Por ejemplo, añade un nuevo System.out.println:

Java

```
public class Main {
```

```
    public static void main(String[] args) {
```

M4S1 CONSTRUYE BASES DE DATOS PARA APLICACIONES WEB

```
// ... tu código de Gson ...

System.out.println(jsonString);

// AÑADE ESTA LÍNEA NUEVA

System.out.println("Este es mi primer PUSH!");

}

}

• Observa: Tu archivo Main.java se pondrá de color azul, indicando que está modificado.
```

Paso 2: Haz Commit (Guardado Local)

Como antes, guardamos esta "foto" en tu historial local.

1. Abre la pestaña **Commit** (o Ctrl + K).
2. Verás tu archivo Main.java modificado.
3. Escribe un mensaje de commit claro. Por ejemplo: Añadido mensaje de prueba para Push.
4. **Importante:** Esta vez, fíjate en el botón de abajo. Tendrás "Commit" y "Commit and Push...".
 - **Commit:** Solo lo guarda en tu PC.
 - **Commit and Push...:** Hace los dos pasos a la vez.

Para aprender, vamos a hacerlo en dos pasos. Haz clic en **Commit**.

Tus archivos vuelven a estar blancos. Tu "foto" está guardada localmente, pero GitHub *aún no la tiene*.

Paso 3: Haz Push (Subir a la Nube)

Ahora, vamos a enviar ese *commit* que acabas de hacer a GitHub.

1. Ve al menú superior: Git > Push... (O presiona Ctrl + Shift + K).
2. **La forma más fácil:** Busca el **ícono de flecha verde hacia arriba** en la esquina superior derecha de IntelliJ.
3. Aparecerá una ventana que te resume lo que estás a punto de hacer. Dirá algo como:
 - main (tu rama local) → origin/main (la rama en GitHub)
 - Y listará el *commit* que hiciste: "Añadido mensaje de prueba para Push".

M4S1 CONSTRUYE BASES DE DATOS PARA APLICACIONES WEB

4. Haz clic en el botón **Push**.

¡Listo! Si ahora vas a tu repositorio en GitHub.com y recargas la página, verás que tu archivo Main.java incluye la nueva línea System.out.println.

2. El Flujo de "Bajar" (PULL): De GitHub a tu PC

Esto es lo que harías al empezar tu día de trabajo, o cuando un compañero te dice "¡ya subí mis cambios!".

Vamos a **simular** que un compañero hizo un cambio.

Paso 1: Simula un Cambio Remoto (en GitHub)

1. Abre tu navegador y ve a tu repositorio de HolaMundoMaven en **GitHub.com**.
2. Navega hasta tu archivo: src > main > java > com > tuejemplo > Main.java.
3. Haz clic en el **ícono de lápiz** en la esquina derecha para "Editar este archivo".
4. Añade un comentario en cualquier parte. Por ejemplo, al inicio:

Java

```
package com.tuejemplo;
```

```
// ¡¡ESTE CAMBIO LO HICIMOS DESDE LA WEB DE GITHUB!!
```

```
import com.google.gson.Gson;
// ... etc ...
```

5. Ve al final de la página. Escribe un mensaje de commit (ej. "Editado desde la web") y haz clic en el botón verde **"Commit changes..."**.

Situación actual: Tu repositorio en GitHub tiene un cambio (un *commit*) que tu computadora local *no* tiene.

Paso 2: Haz Pull (Bajar a tu PC)

1. Vuelve a **IntelliJ IDEA**.
2. Ve al menú superior: Git > Pull...
3. **La forma más fácil:** Busca el **ícono de flecha azul hacia abajo** (⬇) en la esquina superior derecha (justo al lado del de Push).
4. Aparecerá una ventana. Te sugerirá "Pull" (bajar) los cambios desde origin/main a tu rama main.

M4S1 CONSTRUYE BASES DE DATOS PARA APLICACIONES WEB

5. Haz clic en el botón **Pull**.

¡Observa la magia! Verás que tu archivo Main.java se actualiza solo. El comentario // ¡¡ESTE CAMBIO LO HICIMOS DESDE LA WEB DE GITHUB!! aparecerá mágicamente en tu editor.

Resumen del Ciclo Diario

Has aprendido el flujo de trabajo más importante de Git:

1. **Pull (Bajar):** Siempre haz esto *antes* de empezar a programar para asegurarte de que tienes la última versión del código.
2. **(Tú trabajas):** Haces cambios y haces varios **Commit** locales.
3. **Push (Subir):** Al final del día, o cuando terminas una tarea, subes todos tus *commits* a GitHub.

Has dominado el flujo de trabajo básico en la rama principal (main). El siguiente concepto esencial en Git, y la clave de la colaboración, es el trabajo con **ramas** (Branches).