# prueba de pdf

September 18, 2023

# 1 JAVA

# 2 Configurar el ambiente de desarrollo y crear el proyecto

Antes de comenzar a programar debes configurar el ambiente de desarrollo, esto es, instalar las herramientas necesarias para poder programar en Java.

## 2.1 Descarga de aplicaciones

### 2.1.1 JDK (Java Development Kit)

**JDK (Java Development Kit)**: es un paquete con programas para compilar, interpretar y ejecutar el código java así como también un compendio de código prediseñado para facilitar la creación de aplicaciones java. Contiene: 1. *JRE (Java Runtime Environment)*: es el conjunto de programas y librerías necesarias para ejecutar las aplicaciones java, este programa se puede instalar de manera independiente y es posible que ya lo tengas en tu computadora.

2. *Javac (Java compiler)*: es el programa encargado de compilar el código java y pasarlo a bytecode, que es el lenguaje que interpreta la máquina virtual de java *JVM (Java Virtual Machine)* que es parte de JRE.
3. *Java Archive Tool*: es el programa encargado de ensamblar en un paquete las clases java compiladas en un programa ejecutable llamado JAR.

   Recurso: link de descargar de JDK11 (Java SE Development Kit 11.0.19)

Descarga la versión de Oracle JDK específica para tu plataforma y recuerda seguir las instrucciones de instalación. Es importante que configures la variable *PATH* como se indica en la sección *Setting the PATH Environment Variable*, también puedes leer How do I set or change the PATH system variable?

Una vez instalado JDK y configurada la variable de entorno PATH puedes probar la instalación ejecutando en una terminal, cmd (windows) o bash (linux), los comandos `java -version` y `javac -version`, de a uno a la vez, con un espacio luego de java y un guión junto a version. Deberías ver algo parecido a lo siguiente:

### 2.1.2 IntelliJ IDEA

**IntelliJ IDEA**: es un programa conocido como *IDE (Integrated Development Environment)* el cual contiene herramientas útiles para programar tales como un editor de código, un explorador de

archivos, un depurador de código, plugins para trabajar con sistemas de versionado como git,un compilador y ejecutor de código y visor de base de datos, entre otras. La idea de estos programas es que como desarrollador no tengas que usar ningún otro programa para trabajar.

Recurso: link de descarga de IntelliJ IDEA

Recuerda seleccionar la versión community que es gratis aunque carece de algunas características que tiene la versión de pago.

### 2.1.3 Gradle

**Gradle**: es una herramienta que permite la automatización de compilación de código abierto, la cual se encuentra centrada en la flexibilidad y el rendimiento. Los scripts de compilación de Gradle se escriben utilizando *Groovy* o *Kotlin DSL (Domain Specific Language)*.

Gradle tiene una gran flexibilidad y nos deja hacer usos otros lenguajes y no solo de Java. También cuenta con un sistema de gestión de dependencias muy estable.

Gradle es altamente personalizable y rápido, ya que completa las tareas de forma rápida y precisa reutilizando las salidas de las ejecuciones anteriores, sólo procesa las entradas que presentan cambios en paralelo. Además es el sistema de compilación oficial para Android y cuenta con soporte para diversas tecnologías y lenguajes.

Recurso: link de descarga de Gradle

Después de descargar el Zip ponerlo en la misma carpeta que el JDK. Realizar el mismo proceso de las variables de entorno. Para mas información visitar la documentación oficial de Gradle en gradle.org/install.

## 2.2 Crear el proyecto gradle e importarlo a IntelliJ IDEA

Para crear el proyecto utilizarás la herramienta Spring Initializr, con ella se crean los archivos básicos necesarios, así como la estructura de directorios recomendada para comenzar a trabajar en una aplicación de Java, que utilice el Spring Framework implementado con Spring Boot.

- **Framework**: en programación, un framework es un marco de trabajo que tiene como objetivo facilitar la solución de problemas que pueden surgir al programar. Los frameworks aceleran el proceso de programar facilitando tareas como la organización del código o el trabajo en equipo dentro de un proyecto. Fuente.
- **Spring framework**: es un framework para backend que se puede usar con Java, Kotlin y Groovy, Spring framework existe para facilitar la creación de aplicaciones, simplificando la estructura del proyecto y mejorando la velocidad de una aplicación empresarial. *Una aplicación empresarial* es una gran aplicación con fines comerciales, que será usada por muchas personas. Una aplicación empresarial es compleja de hacer, debe ser escalable (dar la posibilidad de ser modificada sin generarle pesadillas a los encargados de mantenerla), brindarle comodidad y seguridad al usuario. Fuente.
- **Spring Boot**: Spring Boot funciona con Spring core (Spring framework). Spring Boot es una variación de Spring que viene con la configuración, que anteriormente, se tenía que hacer a mano, facilitando mucho el desarrollo con Spring, permitiendo al programador no tener que configurar nada, solo programar. Fuente.

- **Spring Initializr**: es una herramienta web intuitiva que nos permite autogenerar el esqueleto de la aplicación. El inicializador genera el `pom.xml` o `build.gradle`, el `main` de la aplicación, y un test que comprueba que la aplicación arranca correctamente. Fuente.

### 2.2.1 Spring Initializr

En Spring Initializr configuramos el proyecto de la siguiente forma (Fuente):

1. **Project**: es el tipo de proyecto. Podemos elegir entre un proyecto *Maven* o un proyecto *Gradle*. El artefacto que se va a generar lo hará con un archivo `pom.xml` con su script y archivos para el *Maven Wrapper*, o con un archivo `build.gradle` y sus correspondientes archivos para el *gradle wrapper*. Nosotros vamos a seleccionar Gradle.
    - *Gradle* se utiliza para compilar, ejecutar pruebas y construir tu aplicación. Gradle proporciona una manera fácil de crear tareas y ejecutarlas para llevar a cabo diferentes pasos en un modo ordenado y controlado. Al mismo tiempo, Gradle proporciona un sólido sistema de dependencias. Fuente.
    - *Gradle Wrapper* es un script que descarga e invoca una versión específica de Gradle. Esto supone dos ventajas. La primera es que no necesitas tener Gradle en tu entorno. La segunda es que puedes garantizar que el build script siempre se ejecuta con la misma versión de Gradle. Fuente.
    - *Las dependencias* son útiles para poder acceder a códigos escritos por otros (como la biblioteca *Mockk* o *Apache Commons Collections*). Cuando instalamos Gradle, se crea un repositorio local en una carpeta que por defecto suele ser `HOME/.gradle`, donde descargamos los componentes localmente una sola vez (muchos proyectos que usan la misma dependencia van a buscar el artefacto en ese mismo lugar). Fuente.
2. **Language**: es el lenguaje de programación que vamos a utilizar en el proyecto. Nosotros vamos a seleccionar Java.
3. **Spring Boot**: es la versión de Spring Boot de la que vamos a depender. Aquí nos permiten incluso depender de las versiones de desarrollo que aparecen entre paréntesis. Nosotros vamos a seleccionar la menos reciente y más estable (que no tenga ninguna marca entre paréntesis).
4. **Project Metadata**:
    41. *Group*: será el campo groupId en el descriptor de maven y el nombre del paquete base de las clases de nuestra aplicación.. Nosotros vamos a indicar com.mindhub.
    42. *Artifact*: nombre de nuestro artefacto. En maven se va a convertir en los campos `artifactId` y `name`. En gradle irá a parar al campo `jar.baseName`. Este será además el nombre del archivo zip que se va a generar. Nosotros vamos a indicar homebanking.
    43. *Name*: ca a parar al campo name de nuestro archivo pom.xml en maven. En gradle no tiene efecto. El campo se completa de forma automática con el contenido de artifact.
    44. *Description*: va a parar al campo description de nuestro archivo pom.xml. En gradle no tiene efecto.
    45. *Package Name*: nombre del paquete base de las clases de la aplicación en caso de que sea diferente a nuestro campo Group. El campo se completa de forma automática con el contenido de artifact y group.

5. **Packaging**: empaquetado de nuestro artefacto donde podemos seleccionar `jar` o `war`. Esto afecta a los plugins de construcción que se especifican en el descriptor de la aplicación (`pom.xml` o `build.gradle`) y a la estructura de archivos que se crea.

6. **Java**: versión de java que vamos a especificar para nuestro artefacto. Nosotros vamos a seleccionar la 11, que es la versión del JDK que instalamos al comienzo.

7. **Dependencies**: vamos a descargar las siguientes dependencias

    71. *Spring Data JPA*: permite la persistencia de datos en tablas SQL con JPA utilizando Spring Data e Hibernate.

        - JPA (Java Persistent API): es una especificación que indica cómo se debe realizar la persistencia (almacenamiento) de los objetos en programas Java. Es una *especificación* porque JPA no tiene una implementación concreta, sino que, existen diversas tecnologías que implementan JPA para darle concreción. JPA nos provee una serie de interfaces que podemos utilizar para implementar la capa de persistencia de nuestra aplicación. Fuente.

        - Spring Data: es uno de los frameworks que se encuentra dentro de la plataforma de Spring. Su objetivo es simplificar al desarrollador la persistencia de datos contra distintos repositorios de información. Fuente.

        - Hibernate: es una herramienta de ORM para Java, que facilita el mapeo de atributos en una base de datos tradicional y el modelo de objetos en una aplicación, Fuente. El ORM (Object Relational Mappig o mapeo relacional de objetos) tiene la función es abstraer la base de datos, de modo que tú como programador puedas hacer consultas sin conocer SQL, y en su lugar, seguir usando el lenguaje de programación que ya conoces. Fuente.

    72. *H2 database*: es una base de datos relacional que se encuentra escrita en Java y funciona como una base de datos en memoria, cuyo punto fuerte es que puede trabajar como una base de datos embebida en aplicaciones Java o ejecutarse en modo cliente servidor. Se añade en nuestras aplicaciones como una dependencia más y una vez configurada la conexión, nos va a permitir realizar pruebas y trabajar como si fuera una base de datos relacional. Fuente.

        - Base de datos relacional: es una colección de información que organiza datos en relaciones predefinidas, en la que los datos se almacenan en una o más tablas (o "relaciones") de columnas y filas, lo que facilita su visualización y la comprensión de cómo se relacionan las diferentes estructuras de datos entre sí. Fuente.

Una vez seleccionadas las dependencias pulsar el botón "generate" y se descargara un archivo comprimido.

Extrae la carpeta homebanking en un directorio de tu preferencia y ejecuta el IntelliJ IDEA, una vez iniciado pulsa el botón de abrir o importar, te preguntará por el directorio del proyecto y deberás indicar la ruta a la carpeta homebanking previamente extraída, una vez que se seleccione la carpeta pulsar el botón ok, luego espera a que el proyecto se cargue por completo. Fuente.

### 2.2.2 Prueba

Para ejecutar el proyecto y probar que todo esté funcionando correctamente ve al pluging de Gradle ubicado en la parte superior derecha del IntelliJ IDEA, allí encontrarás un árbol de directorios con las tareas de Gradle, deberás hacer doble click sobre la tarea `bootRun` ubicada dentro de la carpeta `task/application`.

Verás que en la parte inferior del IntelliJ IDEA aparece la consola de ejecución con los registros de qué está sucediendo a medida que inicia la aplicación. Cuando veas el siguiente mensaje quiere decir que ya inició la misma:

```
Started HomeBankingApplication in ... seconds (JVM running for 4.787)
```

En el navegador y colocamos la url http://localhost:8080/ y deberemos ver un JSON con una entrada llamada profile, parece poco pero hasta ahora lograste crear una aplicación Gradle de Java con Spring Boot, importarla en el IntelliJ IDEA, compilarla y ejecutarla.

Para detener la aplicación pulsa sobre el botón rojo ubicado a la izquierda del panel de ejecución o arriba a la derecha del IntelliJ IDEA.

Si más adelante deseas detener e iniciar de nuevo la aplicación (reiniciar), solo tienes que pulsar el botón de la flecha verde como símbolo de retorno.

## 2.3 Crear la entidad `Client`

prueba

[ ]:
```java
import java.util.Objects;
import java.lang.Long;


@Entity
public class Author {
    @Id
    @GeneratedValue
    private Integer id;

    public Author(Integer id){
        this.id = id;
    }

    public Integer getId(){return this.id;}
}
```

```
Author author = new Author(15);

System.out.println(author.getId());
```