5. Plataformas para el Desarrollo en Java

5.1. Introducción y objetivos

En este tema se estudian algunas de las plataformas que nos ayudan a desarrollar sistemas basados en la tecnología Java.

En primer lugar, estableceremos el contexto de una tecnología independiente de la plataforma que requiere de una máquina virtual que interpreta el código en la plataforma destino, característica fundamental que marca las ventajas e inconvenientes de su uso.

Posteriormente, veremos algunos de los principales entornos de desarrollo integrado que facilitan la escritura, compilación, depuración y despliegue de soluciones basadas en Java, así como algunas plataformas que facilitan la automatización de la construcción y el despliegue de este tipo de aplicaciones. Por último, expondremos algunas de las plataformas que aceleran el desarrollo de soluciones basadas en tecnologías Java*.*

Con el estudio de este tema pretendemos alcanzar los siguientes objetivos:

* Comprender el contexto, el valor y la utilidad que tiene el desarrollo con una tecnología independiente de la plataforma como Java*.*
* Conocer los principales entornos de desarrollo integrado para Java*.*
* Introducir algunas herramientas de automatización para la gestión de dependencias y procesos de compilación y despliegue sobre Java*.*
* Conocer algunas plataformas aceleradoras del desarrollo a nivel de interfaz de usuario y servidores empresariales*.*

5.2. Contexto de las plataformas para el desarrollo en Java

Java es, por un lado, un lenguaje de programación y por otro una plataforma. Como lenguaje de programación, es un lenguaje de alto nivel que se puede caracterizar por las siguientes características: orientado a objetos, multiproceso, de arquitectura neutra, portable, seguro y robusto (Bloch, 2017).

Aunque originalmente era un lenguaje orientado a objetos, Java ha comenzado a adoptar conceptos de otros paradigmas de programación como la programación funcional. Algunas de sus características principales como lenguaje son (Sierra, Bates, y Gee, 2022):

* Orientado a objetos.
* Fuertemente tipado estáticamente.
* Independiente de la plataforma.
* Con un recogedor de elementos no usados para liberar la memoria (*garbage collector*).
* Multiproceso.

Java es un lenguaje de programación de propósito general que adopta el paradigma orientado a objetos. Una característica central del lenguaje es la portabilidad multiplataforma, lo que significa que los programas escritos en una plataforma se pueden ejecutar en cualquier combinación de software y hardware mediante una máquina virtual.

En el lenguaje de programación Java, todo el código fuente se escribe primero en archivos de texto sin formato que terminan con la extensión .java. Esos archivos fuente luego se compilan en archivos .class por el compilador javac. Un archivo *.*class no contiene código nativo de su procesador, en su lugar, contiene *bytecodes*, el lenguaje de máquina de máquina virtual de Java (*Java virtual machine*, JVM). Las instrucciones de código de *bytes* de Java son análogas al código de máquina, pero son interpretadas por una máquina virtual de Java (JVM) específica para la combinación de sistema operativo *host* y hardware. De esta forma, se consigue esta característica de portabilidad mencionada. Los *bytecodes* pueden ser decompilados y se puede obtener una visión del código fuente escrito, salvo los comentarios.

Como plataforma, se compone de dos elementos: la máquina virtual de Java y la interfaz de programación de aplicaciones Java (*Java application programming interface*,API).

Debido a que JVM está disponible en muchos sistemas operativos diferentes, los mismos archivos .class pueden ejecutarse en Microsoft Windows, el sistema operativo Solaris (Solaris OS), Linux o macOS. Algunas máquinas virtuales, como Java SE HotSpot, realizan pasos adicionales en tiempo de ejecución para mejorar el rendimiento de su aplicación. Esto incluye varias tareas, como encontrar cuellos de botella en el rendimiento y volver a compilar (en código nativo) las secciones de código que se usan con frecuencia.

La API es una gran colección de componentes de software listos para usar que brindan muchas capacidades útiles. Se agrupa en bibliotecas de clases e interfaces relacionadas. Estas bibliotecas se conocen como paquetes.

A nivel de plataforma, podemos encontrar un diverso grupo de tecnologías:

A screenshot of a phone

Description automatically generated

Figura 1. Tecnologías en la plataforma Java. Fuente: Oracle, s. f.

* Java Platform,Standard Edition (Java SE): permite desarrollar e implementar aplicaciones Java en equipos de escritorio y servidores.
* Oracle JavaEmbedded: se orienta a soluciones de Internet de las cosas y dispositivos embebidos con menos recursos computacionales.
* Java Platform,Enterprise Edition (Java EE): es el estándar en software empresarial impulsado por la comunidad. Java EE se desarrolla utilizando el proceso de la comunidad de Java, con contribuciones de expertos de la industria, organizaciones comerciales y de código abierto, grupos de usuarios de Java e innumerables personas.
* Java Platform,Micro Edition (Java ME): proporciona un entorno sólido y flexible para aplicaciones que se ejecutan en dispositivos integrados y móviles en el Internet de las cosas, como microcontroladores, sensores, puertas de enlace, teléfonos móviles, televisores o impresoras. Las aplicaciones basadas en Java ME son portátiles en muchos dispositivos, pero aprovechan las capacidades nativas de cada dispositivo.
* Java Card: permite que elementos seguros, como tarjetas inteligentes y otros chips de seguridad resistentes a la manipulación, alojen aplicaciones que emplean tecnología Java. Ofrece una plataforma de ejecución segura e interoperable que puede almacenar y actualizar múltiples aplicaciones en un solo dispositivo con recursos limitados, al mismo tiempo que conserva los más altos niveles de certificación y compatibilidad con los estándares.
* Java TV: es una tecnología basada en Java ME que proporciona una solución eficaz, segura y fácil de implementar para desarrollar aplicaciones Java que se ejecutan en televisores y decodificadores. Al usar el tiempo de ejecución de Java TV, los desarrolladores pueden crear fácilmente aplicaciones, como guías electrónicas de programas (EPG), clientes de vídeo a pedido (VOD), juegos y aplicaciones educativas, aplicaciones para acceder a datos de Internet (por ejemplo, clima, teletipos de noticias, redes sociales o redes) y en la mayoría de los títulos de disco Blu-ray, la interfaz de usuario y el contenido adicional.
* Java DB es una distribución de la base de datos de código abierto Apache Derby. Admite SQL ANSI/ISO estándar a través de las API JDBC y Java EE.

A partir de la versión 11, Oracle modificó el modelo de licenciamiento de la JVM, por lo que es posible utilizar las JDK para desarrollar una aplicación, pero es necesario pagar una subscripción. Oracle Java SE Subscription y Oracle Java SE Desktop Subscription son soluciones de Oracle para desarrolladores empresariales que ejecutan aplicaciones e infraestructura de misión crítica en la plataforma Java SE.

La suscripción de Java SE es una suscripción mensual simple (en abril de 2022, el precio de escritorio es de 2,5 dólares por mes por usuario) que incluye licencias y soporte de Java SE para usar en escritorios, servidores o implementaciones en la nube. Sigue un modelo de uso común, popular con las distribuciones de Linux. La suscripción proporciona acceso a actualizaciones de rendimiento, estabilidad y seguridad probadas y certificadas para Java SE, directamente desde Oracle.

Existen diversas implementaciones de la máquina virtual de Java:

* Azul Platform Core: es una compilación OpenJDK compatible con Azul Systems que cumple con los estándares Java SE 17, 16, 15, 13, 11, 8, 7 y 6.
* Codename One: utiliza el ParparVM de código abierto.
* Eclipse OpenJ9: código abierto de IBM J9 para Windows, AIX, Linux (x86, Power y Z), macOS, MVS, OS/400, Pocket PC, z/OS.
* GraalVM: se basa en HotSpot/OpenJDK y tiene una función políglota para mezclar y combinar de forma transparente los idiomas admitidos.
* HotSpot: la implementación de Java VM de código abierto de Oracle.
* Jikes RVM (Jikes Research Virtual Machine): es un proyecto de investigación. Admite las bibliotecas Apache Harmony y GNU Classpath con una licencia Eclipse.
* leJOS: la *suite* de robótica, un reemplazo de *firmware* para los bloques funcionales programables Lego Mindstorms, proporciona un entorno de programación Java para los robots Lego RCX y NXT.
* Maxine: máquina virtual de investigación de código abierto de Oracle Labs y la Universidad de Manchester.
* JamaicaVM (aicas): una máquina virtual Java en tiempo real para sistemas integrados.

Las más habituales son la oficial de Oracle (OracleJDK), la versión abierta oficial (OpenJDK) o la distribución binaria basada en la anterior ([Adoptium](https://adoptium.net)).

Además de Java, otros lenguajes pueden ejecutarse en la máquina virtual de Java:

* Scala: significa ‘lenguaje escalable’. Creado en 2004, es un lenguaje de tipo estático que combina dos importantes paradigmas de programación: la programación funcional y la orientada a objetos. Scala es un lenguaje orientado a objetos puro, ya que no admite primitivas. Brinda la capacidad de definir clases, objetos, métodos junto con características de programación funcional como rasgos, tipos de datos algebraicos o clases de tipos.
* Kotlin: es un lenguaje de código abierto, de propósito general y tipificado estáticamente desarrollado por el equipo de JetBrains, que reúne los paradigmas funcional y orientado a objetos. El enfoque principal durante el desarrollo de Kotlin fue la interoperabilidad de Java, la seguridad (manejo de excepciones), la concisión y un mejor soporte de herramientas. Desde el lanzamiento de Android Studio 3.0, Kotlin es un lenguaje de programación totalmente compatible con Google en la plataforma Android. También se incluye en el paquete IDE de Android Studio como una alternativa al compilador estándar de Java (Jemerov, 2016).
* Groovy: es un lenguaje específico de dominio dinámico (*domain specific language,*DSL) orientado a objetos, tipado opcionalmente, con soporte para capacidades de compilación y tipado estáticos. Su objetivo es mejorar la productividad de los desarrolladores con una sintaxis fácil de aprender. Se integra fácilmente con cualquier programa de Java e inmediatamente agrega funciones potentes, como capacidades de secuencias de comandos, metaprogramación en tiempo de ejecución y tiempo de compilación y funciones de programación funcional.
* Clojure: es un lenguaje de programación funcional de propósito general. El lenguaje se ejecuta en JVM, así como en la *common language runtime* (CLR) de Microsoft. Sigue siendo un lenguaje compilado y dinámico, ya que sus funciones son compatibles en tiempo de ejecución. Los diseñadores de Clojure querían diseñar un Lisp moderno que pudiera ejecutarse en JVM. Es por eso por lo que también se le conoce como un dialecto del lenguaje de programación Lisp y, como sucede en este último lenguaje mencionado, trata el código como datos y también tiene un sistema de macros.
* Jython: es la implementación de la plataforma Java de Python que se ejecuta en la JVM. Este lenguaje se diseñó inicialmente para permitir la escritura de aplicaciones de alto rendimiento sin sacrificar la interactividad. Jython está orientado a objetos, tiene múltiples subprocesos y utiliza el recolector de basura de Java para limpiar la memoria de manera eficiente. Incluye la mayoría de los módulos que forman parte del lenguaje Python pero también puede importar y usar cualquier clase de las bibliotecas de Java.
* JRuby: es una implementación del lenguaje de programación Ruby para ejecutarse en la máquina virtual Java. Es un lenguaje de alto rendimiento y multiproceso, con una gran variedad de bibliotecas disponibles de Java y Ruby. Además, combina características de ambos lenguajes como la programación orientada a objetos y el tipado pato (*duck typing*).

En el índice de la comunidad de programación TIOBE, que mide la popularidad de los lenguajes de programación, los principales (en marzo 2022) que funcionan bajo la JVM son: Java (núm. 3), Groovy (núm. 25), Kotlin (núm. 31), Sala (núm. 36) y Clojure (núm. 47).

5.3. Valor y utilidad de las plataformas para el desarrollo en Java

El valor y la utilidad del desarrollo en Javase centra en las características asociadas a la ejecución sobre la máquina virtual, en particular:

* Independencia de la plataforma: el *bytecode* de Java se puede escribir una vez y luego ejecutar en múltiples plataformas. No necesita ser portado a un entorno de hardware específico porque se ejecuta en la máquina virtual de Java. Cualquier aplicación compilada en el código de *bytes* de Java se puede ejecutar en una plataforma informática que tenga una máquina virtual de Java.
* API común: el otro gran beneficio de usar JVM es el de una API común para trabajar con recursos específicos de la plataforma. Por ejemplo, no hay necesidad de pensar en cómo manejar la entrada/salida de archivos (E/S) en todas las plataformas posibles, ya que JVM lo hará por nosotros. JVM manejará no solo la E/S de archivos específica de la plataforma, sino también otros recursos como la memoria, las redes, los dispositivos de entrada y salida, etc.
* Control de la corrección: cada vez que se carga un *bytecode*, la JVM realiza una verificación de acuerdo con la especificación de la máquina virtual de Java. Garantiza que el código generado esté correctamente escrito y compilado por un compilador válido. Se realizan muchas comprobaciones de verificación, como la verificación de acuerdo con las restricciones de JVM y la asignación de memoria.
* Seguridad: Java fue escrito con la seguridad en mente. La máquina virtual de Java tiene características de seguridad integradas que permiten a los programadores escribir programas Java altamente seguros. También evita que el software malicioso comprometa el sistema operativo (SO) porque evita que las aplicaciones Java interactúen con los recursos del sistema operativo.
* Gestión de la memoria: no hay necesidad de reservar memoria para las variables o gestionar punteros a espacios de memoria. La JVM se encarga de la gestión de la memoria al asignar la memoria requerida para variables y objetos. Luego, se realiza un seguimiento del acceso a la memoria y el recolector de basura (*garbage collector,*GC) del subsistema JVM puede recuperar la memoria no utilizada cuando sea el momento. Esta funcionalidad libera al desarrollador de gestionar cuándo asignar memoria y cuándo liberarla, lo que reduce los problemas de fuga de memoria.

Adicionalmente, el uso de un lenguaje interpretado conlleva una serie de desventajas:

* Menor eficiencia de ejecución: los programas Java que se ejecutan en una máquina virtual Java tienden a funcionar más lentamente que los programas equivalentes escritos en C++. La neutralidad del sistema del código de *bytes* actúa como una desventaja en lo que respecta al rendimiento. Esto se debe a que la optimización del código depende, en gran medida, de las funciones específicas del sistema. Dado que el código de *bytes* de Java es neutral para el sistema, no se puede optimizar para un conjunto de hardware específico (Oaks, 2014).
* Mayor consumo de memoria: la JVM, necesaria para ejecutar una aplicación en Java, consume memoria, por lo que el consumo es superior. Esto supone un inconveniente en dispositivos con menos recursos, como los teléfonos inteligentes, que justifica la implementación de JVM específicas para los mismos que optimizan el consumo de memoria, pero no ofrecen todas las funcionalidades. Sin embargo, hay proveedores de la nube que tarifican según los recursos utilizados por los programas, por lo que en este caso sí que estaríamos pagando un sobrecoste adicional necesario derivado de la JVM.
* Posibles errores en la JVM: dado que un programa Java se basa en la máquina virtual de Java para ejecutarlo, la JVM debe estar libre de errores para que el programa funcione correctamente. Esta dependencia de la máquina virtual de Java introduce un posible punto de falla para el programa. Afortunadamente, el software Java virtual machine se produce con estándares muy altos y, por lo tanto, no es probable que se envíe con algún error. De todos modos, una falla en la máquina virtual de Java es una posibilidad que debe considerarse.

5.4. Entornos de desarrollo integrados

Un entorno de desarrollo integrado (*integrated development environment,*IDE) agrupa todas las herramientas que un programador necesita para escribir, depurar y probar su código. Un IDE permite a los desarrolladores trabajar en un solo entorno y mejorar su flujo de trabajo de programación. Cada IDE tiene diferentes recursos, pero todos incluyen un editor de texto, herramientas de compilación y un depurador.

Las principales funcionalidades que nos ofrecen los IDE para el desarrollo con Java son:

* Edición de código: esta herramienta identifica e inserta componentes de código para ahorrarle tiempo y reducir el riesgo de errores tipográficos y errores. El resaltado de la sintaxis incorporado es crucial para la legibilidad y la eficiencia. Esta herramienta puede ayudar a encontrar errores en su código y permite filtrar su código a una velocidad más rápida.
* Automatización de compilación: un buen IDE permitirá escribir y ejecutar el código Java en la misma ventana, lo que aumenta la productividad y la eficiencia de su programa.
* Entorno de ejecución: para lanzar la aplicación desarrollada.
* Depuración: si bien los depuradores varían entre los diferentes IDE, es importante tener una herramienta para resaltar posibles errores y ayudarlo a revisar su código, con ejecuciones paso a paso y visualización del valor de variables.

Algunos de los IDE más populares para el desarrollo en Java son:

A screenshot of a phone

Description automatically generated

Tabla 1. IDE para Java. Fuente: elaboración propia.

Eclipse

Es uno de los IDE de Java más populares del mercado. Es gratuito, de código abierto y tiene un extenso ecosistema de complementos que permite a los usuarios personalizar las funcionalidades para el desarrollo de aplicaciones. El IDE de Eclipse viene en ediciones de escritorio y en la nube. Esta última permite a los desarrolladores codificar en un navegador web. Eclipse es una de las mejores opciones debido a sus herramientas de desarrollo. Ofrece un *marketplace* que contiene una variedad de complementos disponibles para descargar, junto con un entorno de desarrollo de complementos que permite a los desarrolladores crear sus propias funciones (Vogel, 2013).

A screenshot of a computer

Description automatically generated

Figura 2. Captura del IDE eclipse. Fuente: elaboración propia.

Apache NetBeans

Es el IDE oficial para Java 8, lo que lo convierte en uno de los principales IDE de Java para crear aplicaciones de escritorio, móviles y web. A su vez, es de código abierto. Es fácil personalizar las aplicaciones de software porque resalta el código Java sintáctica y semánticamente. Con potentes herramientas de refactorización y depuración, NetBeans ayuda a aumentar la precisión y la eficiencia.

A screenshot of a computer

Description automatically generated

Figura 3. Captura del IDE Apache NetBeans. Fuente: elaboración propia.

IntelliJ

Es uno de los mejores IDE para el desarrollo de Java. Es un IDE capaz y ergonómico, con características para aumentar la productividad sin saturar la interfaz de usuario. Incluye un conjunto de herramientas que facilitan la programación, como finalización inteligente, refactorización entre idiomas, análisis de flujo de datos e inyección de lenguaje. IntelliJ está disponible en dos ediciones: la edición comunitaria con licencia de Apache 2 y la edición Ultimate propietaria que ofrece funcionalidades avanzadas para el desarrollo web y empresarial.

Oracle JDeveloper

Es un IDE gratuito ofrecido por Oracle que cubre todo el ciclo de vida del desarrollo: codificación, diseño, depuración, optimización, creación de perfiles e implementación. Se integra con Oracle Application Development Framework (Oracle ADF) para simplificar el desarrollo y ofrece editores visuales y declarativos y permite a los desarrolladores editar directamente sus aplicaciones desde el entorno de codificación. Tiene una función de arrastrar y soltar que simplifica el desarrollo de aplicaciones y es compatible con Java EE y Java SE.

Greenfoot

Fue diseñado con la intención educativa de hacer que la programación Java sea más simple y agradable para los jóvenes desarrolladores. Utiliza gráficos bidimensionales para crear programas, juegos y simulaciones interactivos. Este IDE atrae la atención de profesores y estudiantes y proporciona una plataforma para que interactúen en línea. Ofrece acceso gratuito a sonidos y animaciones e ilustra conceptos tales como programación orientada a objetos, interacciones de objetos y parámetros.

DrJava

Es un IDE liviano que fue desarrollado para principiantes y estudiantes. Aunque no es tan poderoso como algunos de los otros IDE en esta lista, tiene una interfaz ordenada y una apariencia consistente en diferentes plataformas. Permite a los desarrolladores evaluar de forma interactiva su código desde una consola y presentar sus resultados en la misma consola. Da soporte con características como buscar y reemplazar, finalización automática, sangría automática, comentarios y coloreado de sintaxis.

MyEclipse

Es un IDE de Java comercial que se basa en la plataforma Eclipse. Utiliza tanto código propietario como código abierto en el entorno de desarrollo y sus herramientas se pueden utilizar para el desarrollo de aplicaciones Java y el desarrollo web. Es compatible con conectores de servidores de aplicaciones y bases de datos y tiene soporte mejorado para Maven y Spring, lo que ayuda a construir *front-ends* y *back-ends*potentes.

Codenvy

Es un IDE comercial de Java construido sobre Eclipse Che. Hay un total de tres variantes del IDE entre las que los programadores de Java pueden elegir: desarrollador, empresa y equipo. Es un espacio de trabajo para desarrolladores bajo demanda y basado en la nube. Además de escribir código Java que utiliza el editor basado en navegador que ofrece, los programadores de Java pueden beneficiarse de potentes funciones como finalización de código, refactorización y resaltado de sintaxis. El editor también facilita la vida de los programadores al ofrecer una forma eficaz de compilar, depurar, editar y ejecutar varios proyectos.

Si se atiende a la última encuesta para profesionales de Stack Overflow (2021), IntelliJ es usado por el 28 % de los desarrolladores, Eclipse por el 15 % y NetBeans por el 7 %, aunque en estos resultados no se acotan al uso de Java*.*

A screenshot of a graph

Description automatically generated

Figura 4. IDE más populares. Fuente: Stack Overflow, 2021.

5.5. Plataformas para la gestión de las dependencias y procesos de compilación

Dentro del ciclo de vida del desarrollo, una vez que hemos escrito el código fuente, es necesario compilarlo, empaquetarlo, testearlo y desplegarlo. Existen una serie de herramientas de automatización de la construcción (*building automation tools*) que se utilizan para automatizar todo el proceso de creación de compilación de software y otros procesos relacionados, como el empaquetado de código binario y la ejecución de pruebas automatizadas.

En este tema vamos a exponer alguna de las herramientas principales para el desarrollo con Java como Ant*,*Maven y Gradle.

Apache Ant

Es una biblioteca de Java y una herramienta de línea de comandos cuya misión es impulsar los procesos descritos en los archivos de compilación como objetivos y puntos de extensión que dependen unos de otros. Proporciona una serie de tareas integradas que permiten compilar, ensamblar, probar y ejecutar aplicaciones Java. Es extremadamente flexible y no impone convenciones de codificación o diseños de directorios a los proyectos Java que lo adoptan como herramienta de construcción.Para usarlo, basta con crear un fichero build.xml en el que definimos los objetivos para poder hacer una compilación (*compile*), un empaquetado (*jar*) o una ejecución (*run*) que se podrá hacer desde un IDE o desde la propia línea de comandos (por ejemplo: *ant build*).

![A computer code with text

Description automatically generated with medium confidence]()

Apache Maven

Es una herramienta de gestión de dependencias y automatización de compilaciones, utilizada principalmente para aplicaciones Java. Facilita el proceso de construcción, proporciona un sistema de construcción uniforme y una información de proyectos de calidad y fomenta mejores prácticas de desarrollo. Utiliza archivos XML como Ant, pero de una manera mucho más manejable.

Para usarlo deberemos crear el fichero configuración de Maven, que contiene instrucciones de gestión de dependencias y compilación y que se llama por convención pom.xml:

![A screenshot of a computer code

Description automatically generated]()

![A computer code with text

Description automatically generated with medium confidence]()

Con Maven, el proyecto debe tener una estructura de proyecto fija, diferenciando el código fuente (*src*) de las pruebas (*test*) e incorporando ficheros de configuración o recursos externos que no sea código (*resources*).

![A computer code with text

Description automatically generated with medium confidence]()

Podemos compilar el proyecto simplemente con un mvn compile, o empaquetarlo con un mvn package, o bien ejecutarlo con un mvn compile exec:java -Dexec.mainClass="net.unir.missi.HelloWorld".

Gradle

Es una herramienta de automatización de compilación de código abierto centrada en la flexibilidad y el rendimiento. Los *scripts* de compilación de Gradle se escriben con Groovy o Kotlin DSL. Es altamente personalizable ya que está modelado de una manera que es extensible en las formas más fundamentales. Completa las tareas rápidamente al reutilizar los resultados de ejecuciones anteriores y procesa solo aquellos elementos que cambiaron y puede ejecutar tareas en paralelo. Es la herramienta de compilación oficial para Android y viene con soporte para muchos lenguajes y tecnologías populares (Muschko, 2014).

Para usarlo crearemos un fichero build.gradle en el que definiremos las dependencias y tareas:

![A screenshot of a computer code

Description automatically generated]()

A partir de aquí podremos compilar (*gradle classes*) o empaquetar (*gradle jar*).

Accede al código fuente de ejemplos de proyectos con Ant, Maven y Gradle a través del aula virtual

5.6. Plataformas aceleradoras del desarrollo en Java

Existen una serie de plataformas a modo de librerías o *frameworks* que nos permiten acelerar el desarrollo de sistemas con Java. De forma general, veremos con más detalle Spring Framework, las librerías para el desarrollo de interfaces gráficas de usuario y alguna utilidad práctica como Lombok*.*

Las principales plataformas aceleradoras del desarrollo en Java que veremos son:

A screenshot of a computer

Description automatically generated

Tabla 2. Plataformas aceleradoras del desarrollo en Java. Fuente: elaboración propia.

Spring Framework

Es una plataforma Java que proporciona soporte de infraestructura integral para el desarrollo de aplicaciones Java. Spring maneja la infraestructura para que podamos concentrarnos en la aplicación. Nos permite la creación de aplicaciones a partir de objetos Java simples y antiguos (*plain old Java object,*POJO) y aplicar servicios empresariales de forma no invasiva a los POJO. Esta capacidad se aplica al modelo de programación Java SE y a Java EE completo y parcial.

El elemento principal del marco es el control de las dependencias y la inversión del control. Una aplicación Java —un término impreciso que abarca toda la gama desde aplicaciones integradas restringidas hasta aplicaciones empresariales del lado del servidor de n niveles—, por lo general, consta de objetos que colaboran para formar la aplicación propiamente dicha. Por lo tanto, los objetos de una aplicación tienen dependencias entre sí. Aunque la plataforma Java proporciona una gran cantidad de funciones de desarrollo de aplicaciones, carece de los medios para organizar los componentes básicos en un todo coherente, lo que deja esa tarea a los arquitectos y desarrolladores. El componente Spring Framework Inversion of Control(IoC) aborda esta preocupación al proporcionar un medio formalizado para componer componentes dispares en una aplicación completamente funcional lista para usar.

Algunos casos de uso de aplicación del marco son:

* Permite que un método Java se ejecute en una transacción de base de datos sin tener que lidiar con las API de transacciones.
* Facilita la conversión de un método Java local en un punto final HTTP sin tener que lidiar con la API de Servlet.
* Permite convertir un método Java local en un controlador de mensajes sin tener que lidiar con la API de JMS.
* Facilita que un método Java local en una operación de administración sin tener que lidiar con la API JMX.

Spring se basa en la inversión de control como principio de diseño de software, de forma que el flujo de ejecución de un programa se invierte respecto a los métodos de programación tradicionales y es gestionado por la plataforma.

A diagram of a software framework

Description automatically generated with medium confidence

Figura 5. Vista de los componentes de Spring Framework Runtime. Fuente: Spring Framework, s. f.

El marco Spring Framework proporciona un modelo integral de programación y configuración para aplicaciones empresariales modernas basadas en Java, en cualquier tipo de plataforma de implementación.

Un elemento clave de Spring es el soporte de infraestructura a nivel de aplicación: Spring se enfoca en la integración de las aplicaciones empresariales para que los equipos puedan enfocarse en la lógica comercial a nivel de aplicación, sin vínculos innecesarios con entornos de implementación específicos.

Adicionalmente a este marco general, existen un conjunto de proyectos dentro del ecosistema Spring que facilitan el desarrollo de *software* y que se resumen a continuación:

Spring Boot

Facilita la creación de aplicaciones basadas en Spring independientes y de grado de producción que podemos simplemente ejecutar. Es la forma más sencilla de iniciar una solución basada en Spring.

Spring Data

Proporciona un modelo de programación familiar y coherente basado en Spring para el acceso a los datos y, al mismo tiempo, conservar las características especiales del almacén de datos subyacente. Facilita el uso de tecnologías de acceso a datos, bases de datos relacionales y no relacionales, marcos de reducción de mapas y servicios de datos basados en la nube.

Spring Cloud

Proporciona herramientas para que los desarrolladores construyan rápidamente algunos de los patrones comunes en sistemas distribuidos (por ejemplo, gestión de configuración, descubrimiento de servicios, disyuntores, enrutamiento inteligente, micro-proxy, bus de control, tokens únicos, bloqueos globales, elección de liderazgo, distribución sesiones, estado del clúster). La coordinación de los sistemas distribuidos conduce a patrones repetitivos, y al usar Spring Cloud, los desarrolladores pueden implementar rápidamente servicios y aplicaciones que implementan esos patrones.

En este vídeo, *Framework Spring*, aprenderemos a crear un proyecto con el framework Spring.

Spring Cloud Data Flow

Proporciona herramientas para crear topologías complejas para canalizaciones de transmisión de datos por lotes. Las canalizaciones de datos consisten en aplicaciones Spring Boot, creadas con los marcos de microservicios Spring Cloud Stream o Spring Cloud Task. Facilita el procesamiento de datos por lotes y streaming basado en microservicios para Cloud Foundry y Kubernetes. Admite admite una variedad de casos de uso de procesamiento de datos, desde ETL hasta importación/exportación, transmisión de eventos y análisis predictivo.

Spring Security

Es un marco de autenticación y control de acceso potente y altamente personalizable. Es el estándar de facto para proteger las aplicaciones basadas en Spring. Se enfoca en proporcionar autenticación y autorización a las aplicaciones Java.

Spring para GraphQL

Brinda soporte para aplicaciones Spring creadas en GraphQL Java.

Spring Session

Proporciona una API e implementaciones para administrar la información de la sesión de un usuario.

Spring Integration

Amplía el modelo de programación de Spring para admitir los conocidos patrones de integración empresarial (*enterprise integration patterns,*EIP). Permite la mensajería ligera dentro de las aplicaciones basadas en Spring y admite la integración con sistemas externos a través de adaptadores declarativos. Su objetivo principal es proporcionar un modelo simple para crear soluciones de integración empresarial manteniendo la separación de preocupaciones que es esencial para producir código comprobable y mantenible.

Spring HATEOAS

Proporciona algunas API para facilitar la creación de representaciones REST que siguen el principio hipermedia como motor del estado de la aplicación (*Hypermedia As The Engine Of Application State,*HATEOAS) cuando se trabaja con Spring y especialmente con Spring MVC. El problema central que intenta abordar es la creación de enlaces y el ensamblaje de representaciones.

Spring REST Docs

Ayuda a documentar los servicios RESTful.

Spring Batch

Es un marco de trabajo por lotes ligero y completo diseñado para permitir el desarrollo de aplicaciones por lotes sólidas vitales para las operaciones diarias de los sistemas empresariales. Proporciona funciones reutilizables que son esenciales en el procesamiento de grandes volúmenes de registros, incluido el registro/rastreo, la gestión de transacciones, las estadísticas de procesamiento de trabajos, el reinicio de trabajos, la omisión y la gestión de recursos.

Spring AMQP

Aplica conceptos básicos de Spring al desarrollo de soluciones de mensajería basadas en el protocolo *Advanced Message Queuing Protocol*(AMQP). Proporciona una plantilla como una abstracción de alto nivel para enviar y recibir mensajes. También proporciona soporte para POJO controlados por mensajes con un contenedor de escucha.

Spring CredHub

Brinda soporte del lado del cliente para almacenar, recuperar y eliminar credenciales de un servidor CredHub que se ejecuta en una plataforma Cloud Foundry. Proporciona una API para almacenar, generar, recuperar y eliminar de forma segura credenciales de varios tipos.

Spring Flo

Es una biblioteca de JavaScript que ofrece un generador visual HTML5 integrado básico para canalizaciones y gráficos simples. Esta biblioteca se utiliza como base del generador de secuencias en Spring Cloud Data Flow.

Spring para Apache Kafka

Aplica conceptos básicos de Spring al desarrollo de soluciones de mensajería basadas en Kafka. Proporciona una plantilla como abstracción de alto nivel para enviar mensajes. También brinda soporte para POJO controlados por mensajes con anotaciones @KafkaListener y un contenedor de escucha.

Spring LDAP

Es una biblioteca para simplificar la programación LDAP en Java, construida sobre los mismos principios que Spring Jdbc.

Spring Shell

Permite crear fácilmente una aplicación de Shell con todas las funciones (también conocida como línea de comandos) dependiendo de los jars de Spring Shell y agregando sus propios comandos (que vienen como métodos en Spring Beans).

Spring StateMachine

Es un marco para que los desarrolladores de aplicaciones utilicen conceptos de máquinas de estado con aplicaciones Spring.

Spring Vault

Proporciona abstracciones familiares de Spring y soporte del lado del cliente para acceder, almacenar y revocar secretos. Ofrece abstracciones de alto y bajo nivel para interactuar con Vault, lo que libera al usuario de preocupaciones de infraestructura.

Spring Web Flow

Se basa en Spring MVC y permite implementar los flujos de una aplicación web. Un flujo encapsula una secuencia de pasos que guían a un usuario a través de la ejecución de alguna tarea empresarial. Abarca múltiples solicitudes HTTP, tiene estado, trata con datos transaccionales, es reutilizable y puede ser dinámico y de larga duración por naturaleza.

Spring Web Services

Es un producto centrado en la creación de servicios web basados en documentos. Tiene como objetivo facilitar el desarrollo de servicios SOAP con contrato primero, lo que permite la creación de servicios web flexibles utilizando una de las muchas formas de manipular las cargas útiles XML.

Spring implementa el patrón de diseño de inyección de dependencias, de forma que se suministran objetos a una clase en lugar de ser la propia clase la que cree dichos objetos. Estos objetos cumplen un contrato (de ahí el concepto de dependencia) que nuestra clase necesita para poderlos usar.

Librerías para el desarrollo de interfaces gráficas de usuario

Las pilas de Java tradicionales se diseñaron para aplicaciones monolíticas con largos tiempos de inicio y grandes requisitos de memoria en un mundo donde la nube, mientras que los contenedores y Kubernetes no existían. Los marcos de Java necesitaban evolucionar para satisfacer las necesidades de este nuevo mundo. Para ello, se ha creado Quarkus como plataforma para permitir a los desarrolladores de Java crear aplicaciones para un mundo moderno nativo de la nube. Quarkus es un *framework* nativo de Java para Kubernetes diseñado para las máquinas virtuales GraalVM y HotSpot, creado a partir de las mejores bibliotecas y estándares de Java. El objetivo es hacer de Java la plataforma líder en Kubernetes y entornos sin servidor al mismo tiempo que ofrece a los desarrolladores un marco para abordar una gama más amplia de arquitecturas de aplicaciones distribuidas.

Para el desarrollo de interfaces gráficas de usuario con Java*,*existen una serie de librerías que nos permiten acelerar el desarrollo y que enumeramos a continuación:

AWT

Abstract Window Toolkit (AWT) fue la primera versión de la interfaz gráfica de usuario (GUI) para Java creada en el año 1995. Se trata de un conjunto de clases básicas que en su visualización son diferentes en función del sistema operativo. Con Java Swing esta librería quedó obsoleta.

Swing

Es un conjunto de clases que se extienden de AWT, incorporan un número elevado de componentes visuales que son independientes de la plataforma y que siguen el patrón modelo-vista-controlador. Java Swing forma parte de las clases básicas de Java (*Java foundation classes,*JFC). Las JFC son un conjunto completo de componentes y servicios de interacción gráfica con el usuario, creados para Java8*,* que simplifican drásticamente el desarrollo y la implementación de aplicaciones de escritorio e Internet/Intranet de calidad comercial (Zukowski, 2000).

JavaFX

Es una plataforma de aplicaciones cliente de próxima generación de código abierto para sistemas integrados, móviles y de escritorio construidos en Java. Se compone de un conjunto de herramientas moderno, eficiente y con todas las funciones para desarrollar aplicaciones de clientes enriquecidos. Permite a los desarrolladores integrar gráficos vectoriales, animación, sonido y activos web de vídeo en una aplicación interactiva, completa y atractiva (Chin, Vos y Weaver, 2021).

JGoodies

Ofrece software, productos y servicios gratuitos para desarrolladores de escritorio Java. Nos enfocamos en el diseño de la interfaz de usuario de Java y su implementación en Swing. Como organización, ofrecen capacitación, consultoría, bibliotecas, aplicaciones de ejemplo, patrones de escritorio, plantillas de diseño, una arquitectura de aplicaciones y un proceso de producción para acelerar el proceso de desarrollo de aplicaciones de escritorio con Java.

Qtjambi

Es la biblioteca Qt disponible para Java. Qt es el marco C ++ estándar de facto para el desarrollo de software multiplataforma de alto rendimiento. Es una tecnología con soporte oficial dirigida a todos los programadores de escritorio que desean escribir clientes GUI enriquecidos utilizando el lenguaje Java, mientras que al mismo tiempo aprovechan la potencia y el rendimiento de Qt.

Apache Pivot

Permite a los desarrolladores crear fácilmente aplicaciones conectadas, multiplataforma y visualmente atractivas en Java. Es también el único marco para *installable internet applications* (IIA) abierto y está impulsado en su totalidad por la comunidad de desarrollo de software. Permite a los desarrolladores crear soluciones utilizando las herramientas que ya conocen, lo que reduce el tiempo de entrega y la expansión tecnológica.

Java Servlets

Se utilizan para crear una aplicación web, reside en el lado del servidor y genera una página web dinámica. Podemos verlo como una forma de construir una página HTML dinámica mediante una programación Java. En un servidor web que sea capaz de contener *servlets*: al recibir la petición, se ejecuta el código Java correspondiente y, como resultado, se genera la página web.

JavaServer Pages (JSP)

Conceptualmente similar a Java Servlets, sirve para construir una página HTML dinámica mediante una programación Java. La diferencia es que se separa la vista de la página de la lógica de control, por lo que la visualización queda en el archivo JSP mientras desarrollamos en Java el resto de la lógica que es invocadadesde el JSP siempre en el lado del servidor (Murach, 2014).

PrimeFaces

Es un marco de código abierto popular para JavaServer Faces que presenta más de cien componentes, kit móvil, validación del lado del cliente y motor de temas. Dispone de un amplio conjunto de componentes (HtmlEditor, Dialog, autocompletar, gráficos y muchos más), integra AJAX basado en el API JSF AJAX estándar. Simplemente requiere la integración de un fichero .jar, sin más dependencias. Tiene un marco para crear temas, amplia documentación y una gran comunidad de usuarios (Caliskan y Varaksin, 2015).

En este vídeo, *Primefaces: desarrollando interfaces de usuario web en Java,*aprenderemos a ejecutar el proyecto primefaces-showcase para ver todos sus componentes y crear una solución Java cuya interfaz gráfica web está basada en el *framework* PrimeFaces.

Google Web Toolkit (GWT)

GWT es un conjunto de herramientas de desarrollo para crear y optimizar aplicaciones complejasbasadas en navegador web. Su objetivo es permitir el desarrollo productivo de aplicaciones web de alto rendimiento sin que el desarrollador tenga que ser un experto en peculiaridades del navegador, XMLHttpRequest y JavaScript. Es de código abierto, completamente gratuito y lo utilizan miles de desarrolladores de todo el mundo. Fue creado por Google para ocultar la complejidad de varios aspectos de la tecnología AJAX, al ser compatible con los principales navegadores.

Vaadin

Es una plataforma comercial para desarrollar rápidamente aplicaciones web en*back-ends*Java. Permite crear interfaces de usuario escalables en Java o TypeScript y provee herramientas, componentes y un sistema de diseño integrados para iterar más rápido, diseñar mejor y simplificar el proceso de desarrollo (Duarte, 2021).

Lombok

A nivel de utilidades, el proyecto Lombok es una biblioteca de Java que se conecta automáticamente a su editor y herramientas de compilación. Permite no tener que volver a escribir métodos *get*o *set*, sino que en su lugar con una anotación la clase tiene un constructor con todas las funciones. De esta manera, podemos simplificar el código necesario y en tiempo de compilación será añadido. La integración con los principales IDE hace que no se generen errores aunque esos métodos no los veamos escritos como tal.

De esta forma una clase quedará escrita como:

![A computer code with text

Description automatically generated with medium confidence]()

Y podrá ser utilizada como:

![A computer code with black text

Description automatically generated]()

Para instalarlo descargaremos el fichero lombok.jar de la web del proyecto y ejecutaremos java -jar lombok.jar, y empezará a buscar posibles IDE instalados, o también podremos indicarle la ruta en la que están desplegados.

A screenshot of a computer

Description automatically generated

Figura 6. Captura de la instalación de Lombok con los IDE del equipo. Fuente: elaboración propia

Accede al código fuente del ejemplo con Lombok a través del aula virtual.