

UNIVERSIDAD DEL VALLE DE TLAXCALA

PROPUESTA DE ARQUITECTURA SOA EN MBN

TESIS PRESENTADA POR EFRAIN TLAPALE PULIDO
PARA OBTENER EL GRADO DE INGENIERO EN SISTEMAS COMPUTACIONALES

2017

Agradecimientos

Gracias a mi familia, al Maestro Roberto Capistran, a MBN y al PhD Alberto Portilla

Índice general

Agradecimientos	2
1. Introducción	4
1.1. Planteamiento del Problema	5
1.2. Estado del Arte	5
1.2.1. Sistemas de información distribuidos	5
1.2.2. Arquitecturas distribuidas modernas	6
1.2.3. Transferencia de conocimiento en MBN	8
1.3. Justificación	8
1.4. Objetivos	8
2. Marco Teórico	9
2.1. Servicios Web	9
2.1.1. Estructura interna de un servicio Web	10
2.1.2. Estructura externa de un Servicio Web	10
2.2. Java	10
2.3. Java EE	12
2.4. Spring	13
2.5. Java Server Faces	15
2.6. PrimeFaces	16
2.7. PostgreSQL	16
2.8. ORM	17
2.9. Hibernate	18

Capítulo 1

Introducción

MBN es una empresa dedicada a proporcionar consultoría de negocios y desarrollo de software enfocada en apoyar a las organizaciones que requieren la adopción de tecnologías de información como estrategia para acelerar su crecimiento, facilitando la optimización y simplificación de su proceso de negocio.

Esta empresa desarrolla sistemas de información modernos, por lo que deben considerar aspecto como alta disponibilidad, rendimiento, portabilidad, separación de intereses, y facilidad de mantenimiento. Según los requerimientos antes mencionados, se desarrollan sistemas de información distribuidos bajo la arquitectura SOA (Service Oriented Architecture o Arquitectura Orientada a Servicios), este modelo permite separar las capas de la aplicación creando subsistemas independientes que se unifican para lograr una plataforma completa.

La abstracción en capas de un sistema mejora aspectos organizacionales en el desarrollo de un proyecto, pero genera problemas si no se definen y especifican todos los componentes que conforman la arquitectura, tal es el caso de MBN, pues no se cuenta con una arquitectura definida para los sistemas que desarrollan.

La arquitectura de los sistemas desarrollados debe ser entendida y adoptada por todos los miembros involucrados en el proceso, debido a que no se cuenta con una bien establecida, el proceso de adopción y de transferencia del conocimiento no se lleva a cabo de manera eficiente, en ocasiones prolongándose hasta 8 meses, lo cual consume recursos y genera retrasos que una empresa de alto nivel como MBN no se puede permitir.

1.1. Planteamiento del Problema

La empresa Miracle Bussines Network no cuenta con una arquitectura estandarizada para el desarrollo de sistemas de información, por lo que se complica el proceso de construcción de software así como el de transferencia de conocimiento.

1.2. Estado del Arte

1.2.1. Sistemas de información distribuidos

Los sistemas de información han evolucionado de acuerdo a las necesidades generadas a partir de avances tecnológicos empleados en la construcción de hardware y software. Los sistemas de información siempre se han podido abstraer en 3 capas: presentación, lógica y administración de recursos. Los primeros sistemas de información mantenían las 3 capas antes mencionadas en una sola computadora, por lo que eran consideradas aplicaciones monolíticas, esto producía componentes fuertemente acoplados que no podían ser utilizados en ambientes externos

Con la aparición de la red de área local, comenzó una separación de capas que distribuía procesos en diferentes computadoras, la capa de presentación se designó exclusivamente a los clientes y la de lógica y recursos a los servidores. Muchos estilos arquitectónicos se crearon basándose en esta separación de capas. Cuando el avance en la teoría de redes de computadoras logró aumentar el ancho de banda disponible, los sistemas de información comenzaron a aislar las capas y se logró una completa distribución, se creó una capa extra entre el cliente y el servidor llamada “middleware”, que permitía comunicar todas las capas entre sí y mantener organizado el flujo de información.

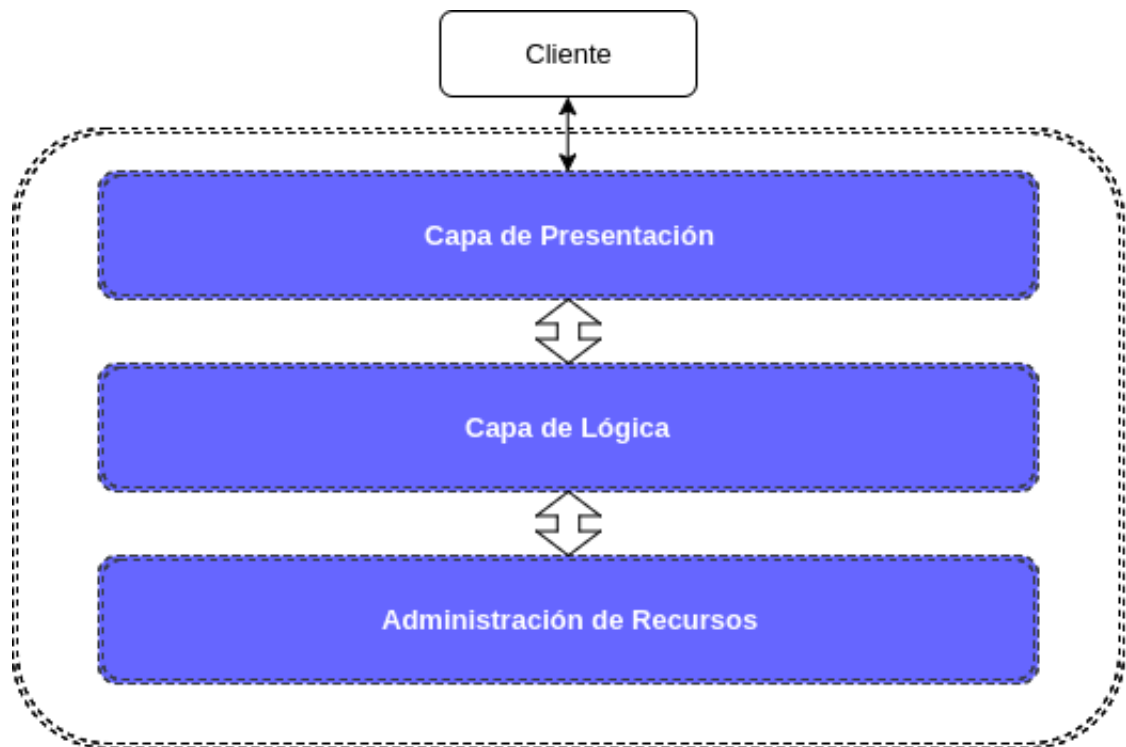


Figura 1.1: Arquitectura de un Sistema Distribuido

1.2.2. Arquitecturas distribuidas modernas

Como se mencionó en el punto anterior, los sistemas de información modernos utilizan estilos arquitectónicos distribuidos para mejorar el rendimiento, construcción y mantenimiento. A continuación se presentan las principales arquitecturas distribuidas:

Cliente-Servidor

Esta arquitectura distribuida reparte los requerimientos de procesamiento entre dos entidades, una que brinda servicios y recursos, llamada servidor y una que solicita esos servicios, llamada cliente. Una de las principales ventajas de esta arquitectura no se refiere al rendimiento del sistemas sino a que permite una separación de responsabilidades y un esquema centralizado de las capas que lo conforman. Esta

arquitectura permite a las organizaciones mantener un control sobre los equipos de trabajo que interfieren en la construcción de software, pues los equipos se dedican exclusivamente a modificar una parte del sistema. Si los equipos no mantienen un estándar ni expresan correctamente las especificaciones de la capa que desarrollan, el sistema puede tener problemas de integración.

N-Capas

Las arquitecturas a n-capas (también referenciadas como tiers, tercios o niveles) se presenta como una extensión al modelo clásico de 3 capas. En esta arquitectura se vinculan múltiples subsistemas creados debido a la complejidad que las capas pueden adquirir, por ejemplo, la capa de recursos tradicionalmente consiste en un sistema de base de datos, esta puede ser sustituida por un sistema de dos o tres capas donde se centralicen varias BD. Otra capa que se puede extender es la de presentación. Para distribuir eficientemente los procesos del sistema se puede implementar un servidor web encargado de generar las páginas html solicitadas por los clientes. Conforme se van añadiendo más capas a los sistemas de información también se hace más complejo su desarrollo, mantenimiento y gestión.

SOA

La arquitectura SOA (Service Oriented Architecture) es utilizada para crear sistemas distribuidos, permite una alineación directa de los procesos o lógica de negocio de las organizaciones con las funciones que ofrecen los sistemas desarrollados. Este tipo de sistemas permite una alta escalabilidad al abstraer la lógica de negocios en Servicios. Los servicios son interfaces, por lo que no se describe la tecnología ni el funcionamiento interno de estos, sino que describen el comportamiento. Por lo regular los servicios de una arquitectura SOA se implementan como WebServices, los cuales son un conjunto de tecnologías utilizadas para transferir datos entre aplicaciones a través de Internet, pueden utilizar distintos estándares, como XML, REST, etc. Los servicios web en una arquitectura SOA permiten tener bien definida la manera en que se exponen y consumen los servicios, por lo que la integración de todas las capas del sistema se estandariza y facilita.

1.2.3. Transferencia de conocimiento en MBN

En la empresa MBN actualmente la transferencia de conocimiento respecto a la arquitectura de las sistemas de información que desarrollan se lleva a cabo de manera genérica, pues se da un panorama amplio de cómo está conformada la arquitectura, pero no se cuenta con una guía institucional que cubra aspectos como: configurar el ambiente de desarrollo, comprobar que el ambiente de desarrollo se ejecute correctamente y desarrollar un proyecto integrador para familiarizarse con las tecnologías utilizadas. Según estudios internos la transferencia de conocimientos básicos sobre la arquitectura de aplicaciones de MBN requiere de 4 a 8 meses.

1.3. Justificación

Para que un sistema de información SOA se pueda desarrollar sin inconvenientes, es necesario que todos los involucrados adopten la arquitectura y conozcan todos los componentes que la integran. La propuesta de una arquitectura formal para los sistemas desarrollados en MBN optimizará el tiempo requerido para adoptarla, también facilitará una de las partes más importantes dentro de una empresa de TI: la transferencia de conocimiento.

1.4. Objetivos

El objetivo general de esta tesis es desarrollar una propuesta de arquitectura SOA para los sistemas de información desarrollados en MBN. Para alcanzar el objetivo general se plantean los siguientes objetivos específicos:

- Analizar y caracterizar las principales tecnologías utilizadas en los sistemas ya desarrollados por MBN.
- Documentar la propuesta de Arquitectura
- Elaborar un documento formal que facilite la adopción de la arquitectura de MBN

Capítulo 2

Marco Teórico

2.1. Servicios Web

Un servicio web es un programa independiente que representa un proceso dentro de la lógica de negocio de un sistema con interfaces abiertas utilizando protocolos de Internet.

La descripción de un servicio web involucra los siguientes elementos:

- Lenguaje base en común. Se debe considerar la adopción de un lenguaje base para transmitir los datos que el servicio web exponga o reciba, durante años el estándar más utilizado era XML, pero JSON ha sobresalido en años recientes por ser un formato más ligero y fácil de entender, lo que mejora el rendimiento y universalidad de los servicios.
- Interfaces. Es necesario definir de manera correcta la interfaz de los servicios web, pues se debe indicar el URI y el protocolo con el que se obtiene acceso al servicio, los datos que requiere y los datos que regresan.
- Protocolos de Negocio. En ocasiones para completar una operación requiere llamar a varios servicios para ser completada, por lo que es importante definir qué servicios y en qué orden se llaman.
- Propiedades. Las propiedades pueden representar características no funcionales de los servicios, por ejemplo una descripción textual del servicio o la versión

del mismo.

Para definir la estructura de los servicios web se deben considerar dos aspectos, primero que son una forma de exponer funcionamiento interno de un sistema a clientes externos, y segundo que son aplicaciones independientes y llevan a cabo procesos internos.

■

2.1.1. Estructura interna de un servicio Web

La estructura interna de un servicio web puede definirse en 4 aspectos clave

2.1.2. Estructura externa de un Servicio Web

2.2. Java

Java es un lenguaje de propósito general desarrollado por Sun Microsystems (después adquirida por Oracle) que reúne las siguientes características:

- Compilado
- Orientado a objetos
- Basado en clases
- Concurrente

Una de las características que no se mencionó en el listado anterior, pero que son esenciales para entender por qué java ha disfrutado de una enorme popularidad en la escena del desarrollo de software es la capacidad de ejecutar programas en cualquier ambiente que pueda ejecutar la Virtual Java Machine, para describir mejor el proceso, a continuación se presenta una descripción de la arquitectura de un programa en Java:

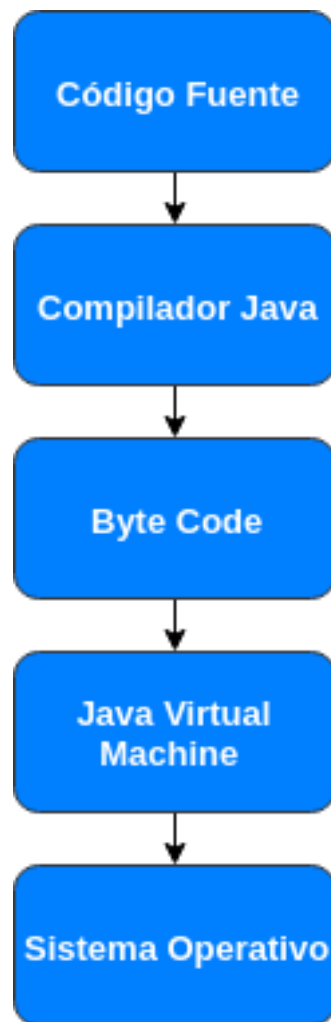


Figura 2.1: Arquitectura del lenguaje Java

Como se puede observar en el diagrama anterior, el proceso parte del código fuente del programa, este pasa por el compilador de Java, generando un ByteCode (archivo .class) el cual será la entrada de la Java Virtual Machine, esta se encargará de transformar el ByteCode a código máquina para su ejecución.

Existen JVM específicas a cada plataforma, es por esto que el mismo programa Java puede ejecutarse en múltiples ambientes y plataformas.

2.3. Java EE

Java EE (Enterprise Edition) es una plataforma para el desarrollo de aplicaciones web a nivel empresarial, proporciona una serie de herramientas que facilitan el desarrollo. Utiliza el lenguaje de programación Java; provee API's para mapear objetos de alguna base de datos, implementar arquitecturas distribuidas y multicapas, y desarrollar servicios web con un enfoque modular. Java EE enfatiza el principio *convención sobre configuración*", lo que permite al desarrollador enfocarse en la construcción de la aplicación mientras la plataforma se encarga de ciertas configuraciones necesarias para ejecutarse. Java EE incluye los siguientes paquetes que extienden la funcionalidad de un proyecto de Java SE:

- javax/servlet
- javax/websocket
- javax/faces
- javax/faces/component
- javax/enterprise/inject
- javax/enterprise/context
- javax/ejb
- javax/validation
- javax/persistence
- javax/transaction
- javax/security/auth/message
- javax/enterprise/concurrent
- javax/jms
- javax/resource

Las aplicaciones Java EE generalmente se justan a un modelo por capas como se muestra en el siguiente diagrama:

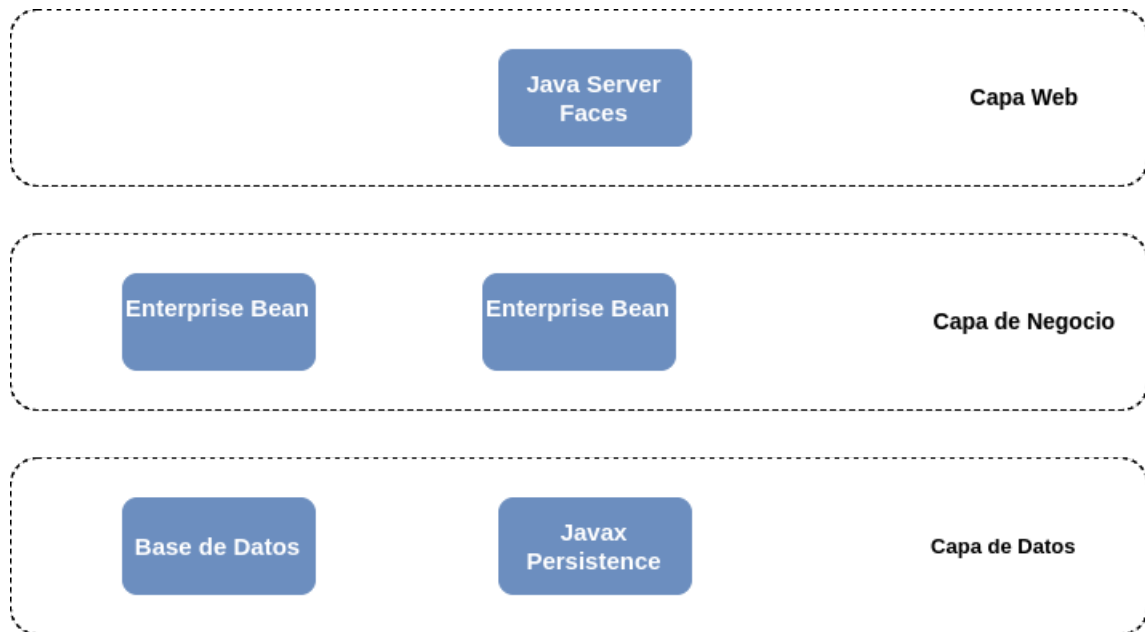


Figura 2.2: Arquitecturad de una aplicación Java EE

2.4. Spring

Spring es un framework para construir aplicaciones compatibles con JVM, esta construido sobre Java EE y proporciona herraminetas para hacer ligero el desarrollo de aplicaciones web.

A continuación se listan algunas de sus características principales:

- Configuración mediante XML
- Manejo de ciclo de vida de componentes
- Administración de dependencia entre componentes
- Se integra con otros frameworks (JSF, Hibernte, etc.)

- Inversión de control
- Inyección de dependencias

Spring promueve el desarrollo modular y la separación de aspectos. Una de las ventajas que ofrece Spring es el poder desarrollar funcionalidades de la aplicación utilizando POJO's (Plain Old Java Object), se utilizan anotaciones para identificar el tipo de objeto y el framework es capaz de unirlos y generar una aplicación completa, a continuación se presenta un ejemplo de un controlador creado bajo el patrón MVC:

```
@Controller
@RequestMapping("/libros")
public class LibrosController{

    public final InventarioLibros inventario;

    @RequestMapping(method = RequestMethod.GET)
    public Map<String> get() {
        return inventario.getLibros();
    }
}
```

Cabe resaltar las anotaciones *@Controller* y *@RequestMapping* son proporcionadas por Spring y la lógica es una simple clase de Java, por lo que se agiliza y simplifica el desarrollo.

2.5. WildFly

WildFly es un servidor de aplicaciones de código libre que permite el despliegue de aplicaciones Java. Es flexible, por lo que se pueden editar diferentes configuraciones dependiendo del proyecto desarrollado. Entre sus principales características se encuentran:

- Soporte para Java EE (última versión de Java EE)
- Compatibilidad con tecnologías web modernas, como WebSockets o estándares REST
- Soporta paradigmas modulares de aplicaciones Java

2.6. Java Server Faces

JSF es un framework MVC para desarrollar interfaces gráficas en aplicaciones Java EE. JSF utiliza componentes definidos en archivos XHTML en forma de Facelets. En general una interfaz construida con JSF está integrada por:

- Vista: Facelets
- Modelo: Objetos Java
- Controlador: ManagedBeans

Entre las ventajas de desarrollar interfaces gráficas con JSF sobre escribir código JavaScript se encuentran:

- Acceder a objetos de lógica de negocio fácilmente
- Ofrece un desarrollo seguro y robusto
- Fácil de probar y mantener

La API de JSF incluye los siguientes elementos:

- Herramientas para representar componentes de interfaz gráfica y administrar su estado.

- Validación.
- Navegación.
- Manejo de Eventos de cliente y servidor.
- Conjunto de componentes por defecto (botones, inputs, etc.).
- Beans administrados (Managed beans).

2.7. PrimeFaces

PrimeFaces es un framework para construir interfaces gráficas con Java EE. Es conocido por ser una librería ligera, fácil de usar, centrada en el rendimiento y la simplicidad. Es el framework recomendado en la documentación oficial de Spring para desarrollar frontend con JSF. Incluye componentes estilizados como inputs, botones, menús, gráficas, además de incluir funcionalidades AJAX.

2.8. PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional *open-source*, utiliza un modelo cliente/servidor y hace uso de multiprocesos para garantizar la estabilidad del sistema. La arquitectura de la BD se representa en el siguiente diagrama que puede ser encontrado en la documentación oficial:

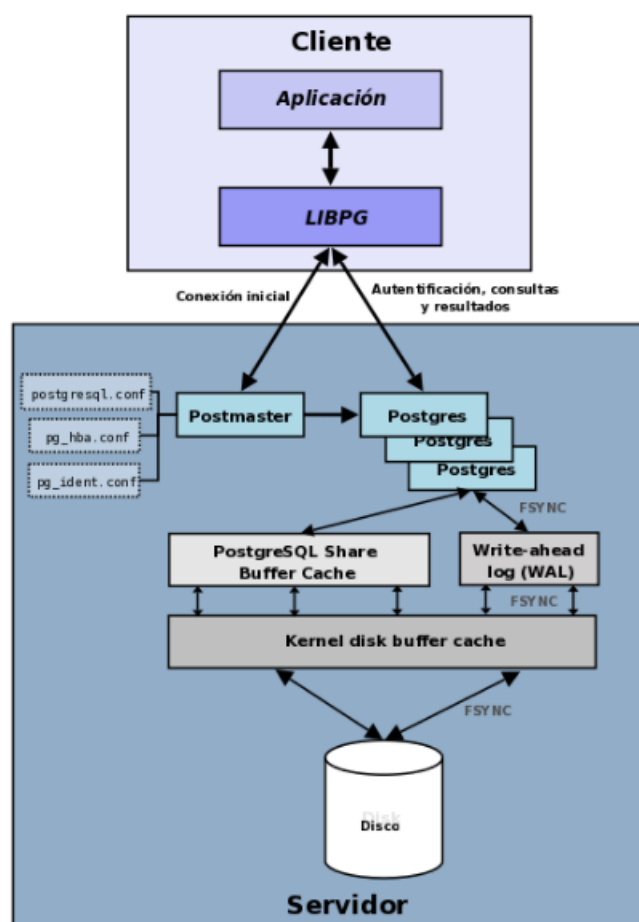


Figura 2.3: Arquitectura de PostgreSQL

2.9. ORM

El mapeo de objeto-relación (Object-Relational Mapping) es un modelo de programación que convierte las tablas de una base de datos en entidades que faciliten el acceso a los datos a los programadores. Las herramientas ORM evitan que los programadores tengan que aplicar consultas complejas en lenguaje SQL, al aplicar una capa de abstracción sobre la base de datos se agiliza el desarrollo y se puede escribir código con una legibilidad mayor. Los ORM también aportan herramientas como la validación de datos, generación de estructuras, protección contra ataques

SQL, definición de relaciones, entre otras.

El proceso de transformación utilizado por un ORM se describe en el siguiente diagrama:

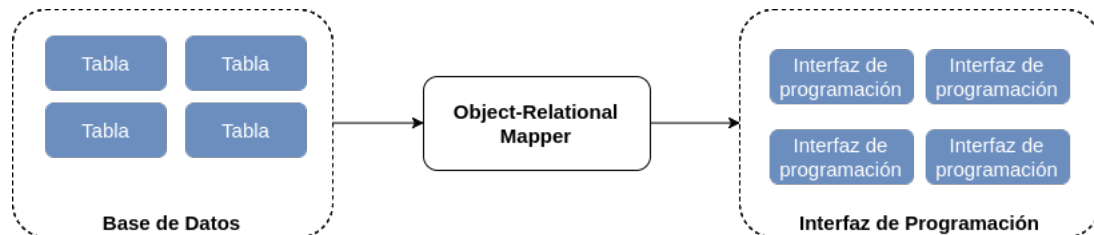


Figura 2.4: Proceso de mapeo de un ORM

2.10. Hibernate

Hibernate es un framework ORM, que, además de contar con su propia API nativa, también es una implementación de la JPA (Java Persistence API), por lo que es posible integrarla con aplicaciones Java SE, Java EE, y otras tecnologías relacionadas. Hibernate es conocido por su alto desempeño, escalabilidad, estabilidad y calidad, lo que hace de este ORM la opción por excelencia a la hora de desarrollar aplicaciones Java.