Capítulo 1

Introducción

1.1 Definición del problema

El Departamento de Ingeniería en Computación desempeña un papel importante como enlace entre los estudiantes, profesores, universidades y el campo laboral, coordinando aspectos de la carrera tales como plan de estudios, intercambios universitarios, investigación, programación de horarios, así como la contratación de personal docente capaz y comprometido con el aprendizaje y superación de los estudiantes.

Se encarga también de trámites propios de dichas actividades como la recepción de documentos de profesores de nuevo ingreso y los procedimientos para la generación de nombramientos de todos los profesores, ayudantes de profesor y técnicos académicos del Departamento a partir de una base de datos.

La cantidad de información manejada por el personal aumenta en complejidad y responsabilidad al administrar el proceso de recepción de calificaciones de los laboratorios de 8 asignaturas que deben ser organizadas y enviadas a cada profesor de teoría, que van de los 6 a los 34 grupos cada una con un aproximado de 16 a 55 alumnos por grupo. Además deben ser incorporadas las calificaciones de semestres lectivos previos como parte del proceso de revalidación que tiene como fin brindar un apoyo al estudiante que acreditó el laboratorio pero no así la teoría.

De esta forma el alumno regular y el que está en artículo 33 del Reglamento General de Inscripciones (Alumno Sin Derecho a ReInscripción o ASDRI) pueden cursar de nueva cuenta la teoría conservando su calificación aprobatoria de laboratorio si es el caso con lo que se reduce la demanda de grupos de laboratorio, asimismo será válida para los alumnos que presenten examen extraordinario quienes también deben tener el laboratorio aprobado.

1.1.1 Panorama actual

Desde hace algunos años el personal del Departamento ha desarrollado y adaptado un sistema que tiene como fin facilitar algunas etapas del proceso de recepción, revalidación y envío de calificaciones de laboratorio, sin embargo no es del todo fiable y es propenso a errores humanos.

En resumen, el desarrollo actual se da de la siguiente forma:

En la última semana de clases de cada semestre el profesor de laboratorio debe enviar las evaluaciones de sus estudiantes con información específica: nombre, número de cuenta, carrera, grupo de teoría y calificación obtenida. Parte de esta información les es entregada al descargar su lista de alumnos inscritos del portal de la Unidad de Servicios de Cómputo Administrativo (USECAD) que brinda este servicio a los profesores de la Facultad de Ingeniería.

El primer obstáculo se presenta cuando estos profesores no entregan en tiempo y forma, al no cumplir los requisitos del formato ni las fechas establecidas, lo que lleva a revisar minuciosamente documento por documento y contactar constantemente al docente para solicitar su evaluación, siendo ésta una actividad muy tediosa y tardada.

Si esto sucede los profesores de teoría se ven afectados en su cronograma al retrasar su evaluación final en la que contemplan cierto porcentaje para el laboratorio y para la cual deben tomar en cuenta además que es requisito indispensable aprobar el laboratorio para acreditar la asignatura.

Una vez recibidas las listas de todos los grupos de una asignatura y verificado su formato se pasan por una macro programada en Microsoft Excel, la cual mediante comandos de Visual Basic y sentencias de SQL agrupa las calificaciones y las exporta a un archivo por asignatura con sus correspondientes grupos.

Como parte imprescindible de la metodología actual está el descargar en las primeras semanas de iniciado el semestre las listas de alumnos inscritos incluidos ASDRI del portal de USECAD para cada grupo de teoría. A dichos archivos se les da un tratamiento especial a fin de adaptarlo al formato requerido por la macro, que una vez ejecutada y generado el archivo se extrae la lista con las calificaciones finales y se envía mediante correo electrónico al profesor de teoría correspondiente.

Sin embargo, pueden presentarse casos donde el número de cuenta, nombre o calificación no estén escritos correctamente y al no contar con un módulo de verificación no se produce ningún aviso de estos errores y pasan desapercibidos.

Otra carencia del método actual consiste en la opción de búsqueda de alumno de forma individual dado que en ocasiones deben realizarse correcciones de calificación, afrontando esta situación con una revisión de cada uno de los archivos de forma manual lo que representa una pérdida de tiempo y esfuerzo para el personal.

También destaca la falta de seguridad y restricciones para el acceso a la información, donde cualquiera con cercanía al equipo donde ésta se almacena puede realizar alteraciones a las calificaciones sin generarse ningún reporte de tal evento.

Lo anterior constituye un largo y complejo proceso de muchos pasos, que recibe información de terceros y se convierte en factor de error al no ser entregada como se requiere, propenso a fallas por parte del personal del Departamento de Computación y vulnerable a alteraciones, lo cual repercutiría directamente en la evaluación y avance de los estudiantes.

1.1.2 Objetivos

Desarrollar e implementar un sistema que sea capaz de gestionar la información de calificaciones y revalidaciones de los laboratorios pertenecientes al Departamento de Ingeniería en Computación obteniendo la información de una base de datos eficiente para ser puesta a disposición oportunamente a los profesores de teoría y así éstos lleven a cabo su evaluación.

Optimizar el proceso de asignación de calificaciones mediante un portal Web y automatizar el relacionado al tratamiento y distribución de las mismas incluidas las correspondientes a las revalidaciones mediante un sistema que estará alojado en los servidores de la Unidad de Servicios de Cómputo Administrativo (USECAD).

1.1.3 Alcances

Al concluir el proyecto conjunto entre el Departamento de Ingeniería en Computación y la USECAD se conseguirá dar respuesta a las problemáticas mencionadas a través de un sistema que cubrirá las siguientes expectativas:

- Integral.

El sistema estará conformado por diferentes módulos que atiendan las necesidades del Departamento en cuanto al manejo de las calificaciones y garanticen su funcionamiento y eficacia.

- Seguro.

Con la finalidad de reducir las vulnerabilidades del sistema y que éste sea resistente a amenazas, serán implementadas las características señaladas por la arquitectura de seguridad OSI que contempla 5 categorías de servicios de seguridad:

1. *Confidencialidad*. Busca garantizar que la información pueda ser accedida por las partes autorizadas, previniendo su divulgación al viajar por la red por lo que se recurre a la ocultación de información mediante herramientas como la criptografía.

2. *Autenticación.* Sirve para verificar la identidad del usuario a partir de un dato conocido como una contraseña, que al ser validada por la parte con quien desea identificarse demuestra ser quien dice ser. La validación puede darse también a partir de un objeto físico como una llave o por características inherentes al individuo como su patrón de voz. Por lo anterior está estrechamente relacionada con los servicios de Control de Acceso.

3. *Integridad.* Garantiza que la información es recibida exactamente como es enviada protegiendo al sistema contra alteraciones, modificaciones, borrado o inserción de los datos. Cuando algunas de estas acciones ocurren el servicio genera un aviso y utiliza mecanismos para la recuperación de los datos.

4. *Control de Acceso.* Una vez que el usuario ha pasado por el servicio de autenticación tiene acceso al sistema y a sus recursos bajo ciertas condiciones que son determinadas por los privilegios de cada usuario, los atributos de la información y las acciones de escritura, lectura y ejecución que le sean permitidas.

5. *No repudio.* Evita que tanto emisor como receptor nieguen su participación en la comunicación verificando tanto que el mensaje fuera enviado por la entidad especificada como que fuera recibido por la otra parte, empleando generalmente herramientas como las firmas digitales.

- Fiable.

Desde que el alumno sin derecho a reinscripción en laboratorio se registre y que el profesor de laboratorio asigne calificaciones hasta que el profesor de teoría las descargue, todo esto vía un portal Web será un proceso en tiempo real, sin errores y ejecutado por el sistema con lo que se podrá confiar en la veracidad de la información.

- Multiplataforma.

Una ventaja importante que presentará el sistema será el correr en diversas plataformas operativas gracias a las herramientas sobre las cuales estará desarrollado.

- Amigable.

Las interfaces tanto en los portales como con el propio sistema serán amigables con los diferentes usuarios, que bien podrán descargar formatos como el de ASDRI, asignar y descargar calificaciones vía Web, realizar modificaciones y respaldos de una forma sencilla e intuitiva.

1.2 Antecedentes

La Facultad de Ingeniería forma parte del conjunto de Facultades, Institutos y escuelas de la Universidad Nacional Autónoma de México. Nace del interés del gobierno virreinal por resolver los problemas resultantes de la explotación irracional de los recursos mineros y ve su evolución a través de la historia en el Real Seminario de Minería, el Colegio de Minería, la Escuela Nacional de Ingenieros y finalmente con el rango de Facultad en el año de 1959 gracias a la iniciativa de Javier Barros Sierra.

Se organiza en Divisiones que coordinan las Ciencias Básicas y todas las carreras que se imparten, en Secretarías orientadas al desarrollo, la investigación y la administración de actividades y servicios (figura 1.1)

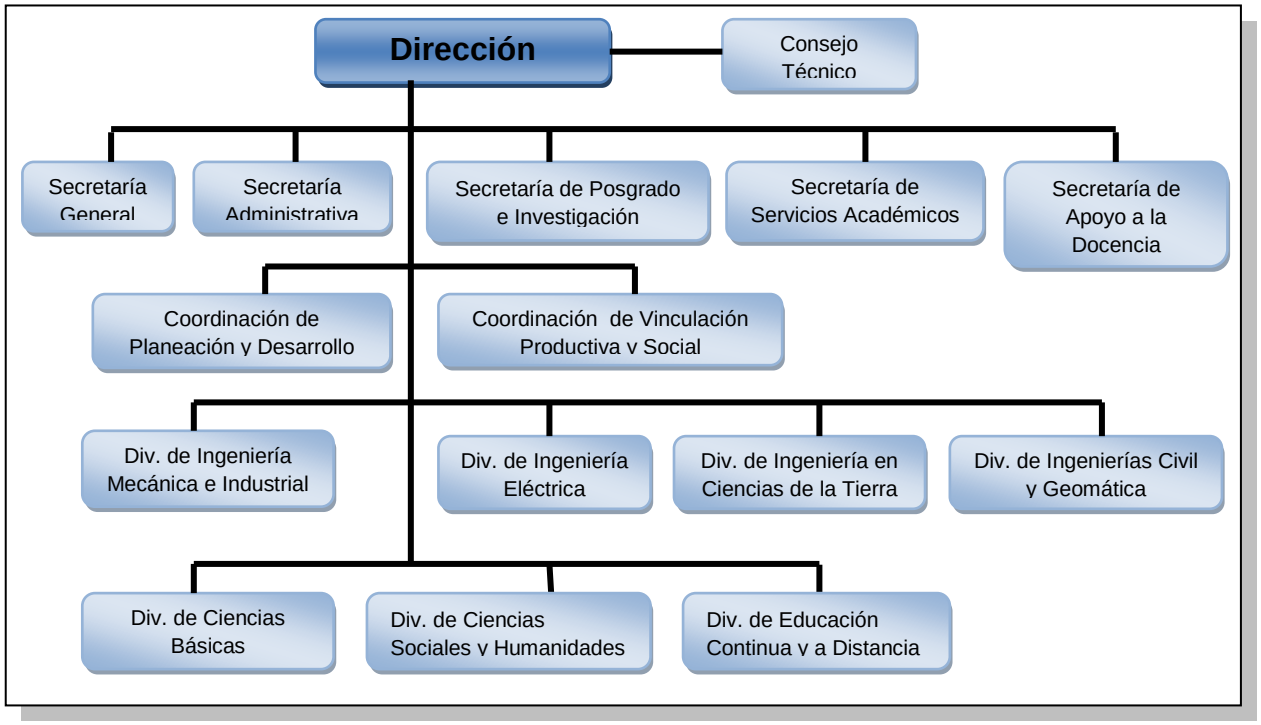

Figura 1.1 Organigrama de la Facultad de Ingeniería

Misión.

Formar integralmente ingenieros competitivos, comprometidos con su país, con valores, habilidades y actitudes que se desempeñen profesionalmente y capaces de orientarse a la investigación y la docencia.

Fomentar la investigación y la difusión de la cultura.

Visión.

Ser la institución al frente de la formación de ingenieros actualizados mediante las opciones de educación continua y a distancia, además de fuente de nuevos conocimientos fruto de la investigación con un impacto óptimo en la sociedad, la cultura y la ecología.

Actualmente ofrece formación en las carreras de:

- Ing. Civil
- Ing. Geomática
- Ing. Geofísica
- Ing. Geológica
- Ing. de Minas y Metalurgia
- Ing. Petrolera
- Ing. Eléctrica y Electrónica
- Ing. en Computación
- Ing. en Telecomunicaciones
- Ing. Mecánica
- Ing. Industrial
- Ing. Mecatrónica

En 2011 se reportó que la carrera de Ingeniería en Computación tuvo una matrícula de 423 alumnos de primer ingreso y 1887 de reingreso de acuerdo a estadísticas de la Facultad de Ingeniería lo que la convierte en la carrera con mayor número de estudiantes con 2310 de un total de 11715 que abarca todas las carreras de la Facultad, apenas seguida por Ingeniería Eléctrica Electrónica e Ingeniería Civil, con 1759 y 1559 respectivamente[1].

Brindar servicios a tal número de estudiantes se vuelve una tarea compleja que es solventada por diversas entidades y Departamentos, tales servicios cubren los rubros académico, administrativo, cómputo y la investigación. Uno de los aspectos más importantes es el proceso de reinscripción que se lleva a cabo vía Internet y del cual se encarga la USECAD semestre a semestre, así mismo el de programación de horarios y administración de calificaciones y revalidaciones de laboratorios que es responsabilidad del Departamento de Ingeniería en Computación, entre muchos otros servicios imprescindibles para el funcionamiento del sistema educativo.

[1].: UNAM :: Facultad de Ingeniería :. Consultada en Abril de 2011 en: http://www.ingenieria.unam.mx/paginas/estadisticas/matricula.php

1.2.1 Departamento de Ingeniería en Computación

El Departamento de Ingeniería en Computación conforma junto con otros seis Departamentos la División de Ingeniería Eléctrica de la Facultad de Ingeniería de la UNAM como puede observarse en la figura 1.2.

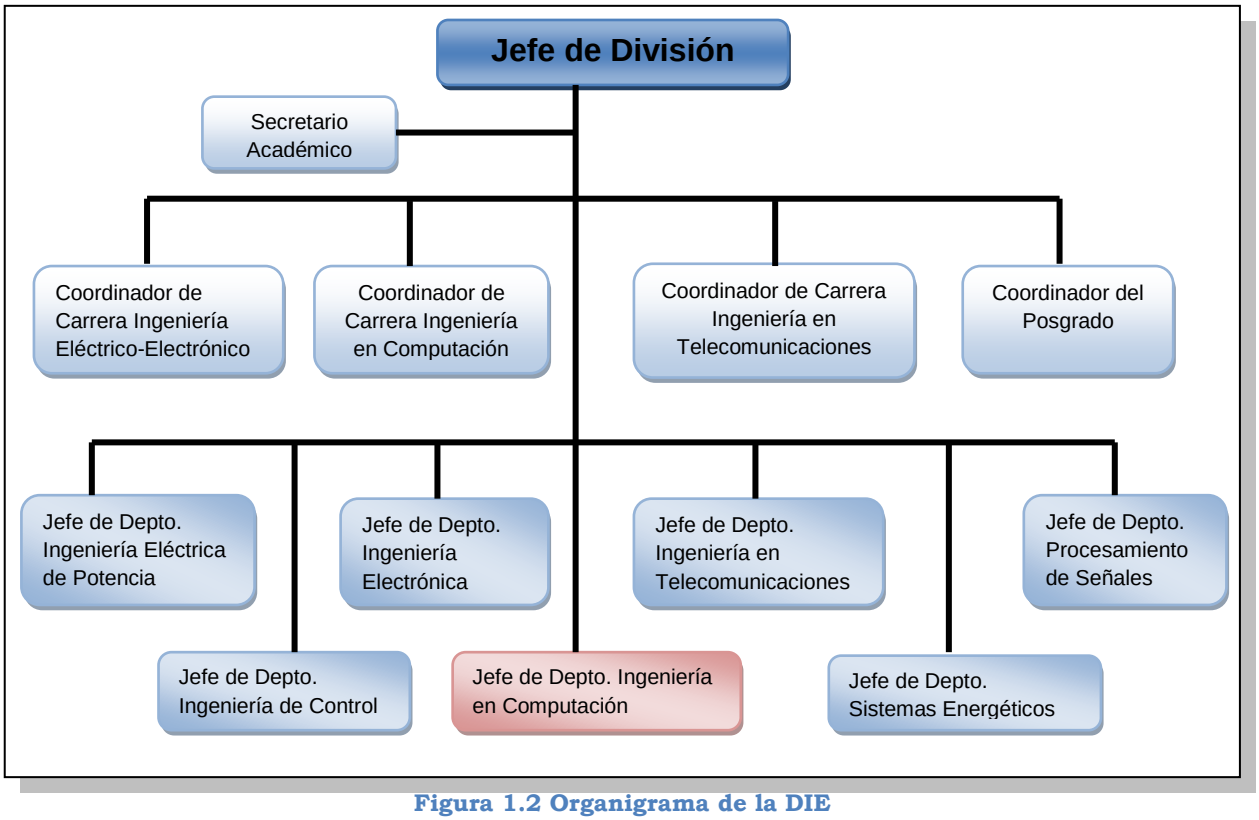

Figura 1.2 Organigrama de la DIE

Misión.

Coordinar todos los aspectos relacionados con la carrera de Ingeniería en Computación.

Proponer los cambios que requiera el Plan de Estudios.

Formar estudiantes de alto nivel en todas las áreas de la computación brindándoles las herramientas necesarias para que sean competitivos en el ámbito profesional tanto nacional e internacional y puedan aplicar sus conocimientos al desarrollo de sistemas complejos que resuelvan problemas y simplifiquen las diversas actividades del quehacer humano.

Favorecer la especialización de alumnos y docentes en campos de la ingeniería de hardware y software, redes y seguridad, bases de datos, sistemas inteligentes y computación gráfica e ingeniería biomédica, preservando el interés por mantenerse actualizado y a la vanguardia.

Visión.

Fomentar la Investigación y la Docencia.

Promover la vinculación con empresas y otras instituciones educativas nacionales e internacionales.

Algunas funciones del Departamento de Ingeniería en Computación.

- Mantenimiento y actualización de la agenda del personal académico.
- Mantener al día la base de horas/profesor y horas/ayudante con las que cuenta.
- Organización, planeación y realización de horarios de cada semestre.
- Elaboración de los movimientos (contrataciones, bajas, altas, prórrogas, etc.) del personal académico y sus justificaciones.
- Revalidación de calificaciones de los laboratorios.
- Recopilación de calificaciones de los laboratorios que pertenecen al Departamento, así como la distribución de calificaciones de éstas hacia los profesores de teoría.

1.2.2 Unidad de Servicios de Cómputo Administrativo

La Unidad de Servicios de Cómputo Administrativo (USECAD) forma parte de la Secretaría de Servicios Académicos (SSA) que es el organismo de la Facultad de Ingeniería encargado de proporcionar los servicios de cómputo para las actividades de administración escolar. A continuación se muestra en la figura 1.3 el organigrama de la SSA.

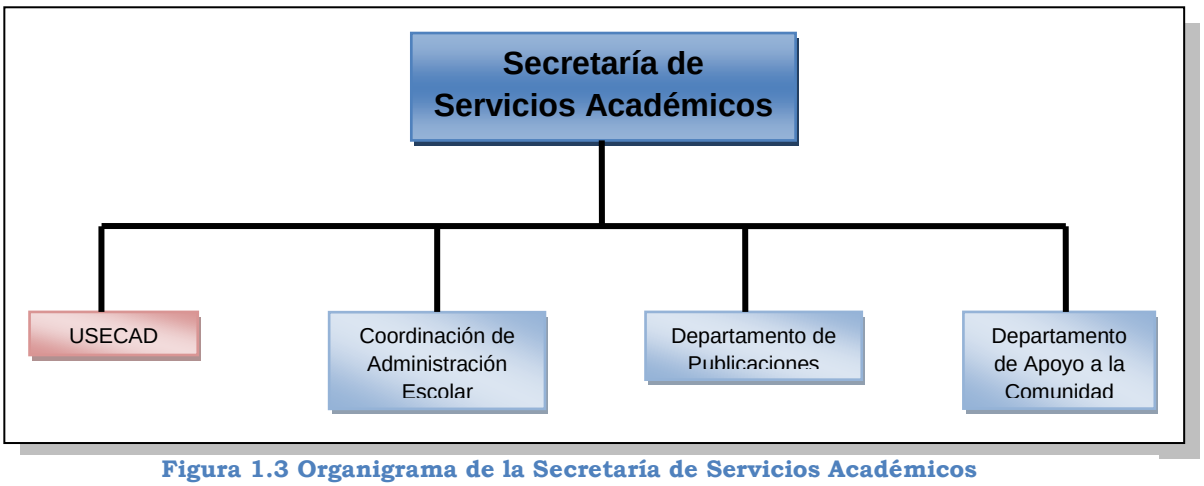

Figura 1.3 Organigrama de la Secretaría de Servicios Académicos

Misión.

Brindar los servicios de cómputo, soporte técnico, desarrollo de sistemas y consulta a su base de datos tanto alumnos como a los funcionarios de la Facultad de Ingeniería.

Visión.

Actualizar la infraestructura y los sistemas que mantienen los procesos académicos y administrativos para mejorar la calidad de servicios a los usuarios.

Algunas funciones de la USECAD.

- Administrar la red de cómputo que permite a los funcionarios de la Facultad el acceso a las bases de datos de inscripciones, información de profesores y alumnos, horarios e historiales.

- Inscripción y reinscripción de los alumnos vía Web.

- Proporcionar servicios de consulta vía Web sobre horarios, cupos, vacantes y resultados de la inscripción.

- Actualizar los recursos de los servidores de bases de datos y el equipo de cómputo de la SSA para asegurar su correcto funcionamiento.

1.3 Marco de referencia

Conceptos básicos de programación

Un programa es lo que marca la diferencia entre una máquina de escribir y una computadora –sin hacer de lado la riqueza del hardware- dado que permite recrear y resolver situaciones al familiarizarlas con metodologías y lenguaje matemático atendiendo a cada variable con una acción.

Un **programa** es un conjunto de instrucciones que guían a la computadora y sus recursos para resolver algún problema, que interactúan con el usuario y los datos que procesa.

Mediante un programa se puede representar cualquier situación de la realidad, simular fenómenos y realizar cálculos extensos y complejos.

El proceso de un programa implica varias interrogantes tales como:

- ¿Para qué servirá?
- ¿Qué datos usará?
- ¿Qué resultados producirá?
- ¿Cómo se operaran los datos para obtener lo esperado?

A partir de las respuestas y teniendo en mente la estructura interna y funcionamiento más convenientes se procede a diseñarlo con ayuda de alguna representación simbólica o textual resultando en un algoritmo que posteriormente se codificará con un lenguaje de programación.

Este proceso consta de un conjunto de instrucciones legibles por un individuo que reciben el nombre de código fuente, pero al ser sólo palabras y símbolos no son identificados por la computadora ya que ésta sólo es capaz de procesar elementos binarios, es decir, una serie de ceros y unos.

Tal secuencia que puede ser de millones conocida como lenguaje máquina es el resultado de traducir el código fuente mediante un compilador[2], el cual será

[2] Compilador. Programa que lee el código fuente y lo divide en instrucciones, estructuras de control, operadores, etc. propias del lenguaje de programación validando que sea correcto sintáctica y semánticamente, posteriormente lo traduce a código intermedio o código máquina que ya es ejecutable por el equipo.

elegido de acuerdo a la plataforma operativa donde será ejecutado el programa en base a las necesidades, finalidad y complejidad del mismo.

Sin embargo el mayor peso en la creación del programa recae en el razonamiento del desarrollador, en su forma de analizar, diseñar y aprovechar los recursos a su alcance para ofrecer soluciones considerando en el proceso cada detalle y cada posible falla.

Una aptitud importante es también cómo se comunica con la computadora, tener en mente que hablarle es como instruir a un niño pequeño en cierta actividad, a quien deben darse instrucciones precisas en un lenguaje que comprenda y recordar que tanto sus experiencias como su poca capacidad debidas a su corta edad le impiden la toma de decisiones ante algún tropiezo dado que sólo seguirá el camino que se le haya trazado.

En conclusión la computadora será el perfecto asistente que ejecute todo lo que se le indique pero no podrá afrontar problemas por sí misma a menos que el programador los haya contemplado y le enseñe una posible solución como parte de las instrucciones que reciba en cierto lenguaje de programación, que si bien existen los de alto nivel distan mucho del natural.

Los lenguajes de programación se pueden clasificar de acuerdo a la figura 1.4:

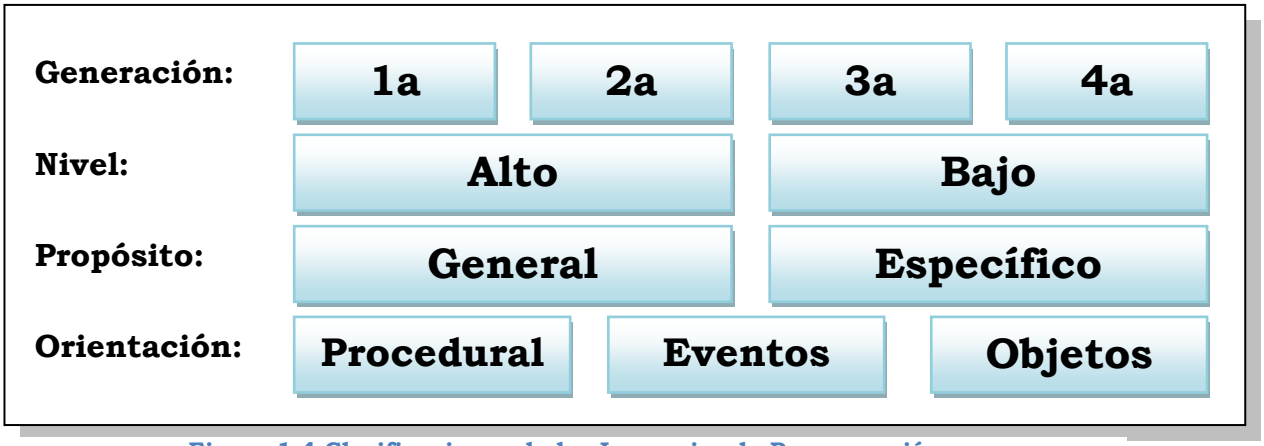

Figura 1.4 Clasificaciones de los Lenguajes de Programación

La *generación* refiere a que tan compleja y numerosa sea la secuencia de números binarios representada por cada símbolo que compone el lenguaje.

El *nivel* comprende el número de plataformas operativas en las que el programa desarrollado en el lenguaje podrá ejecutarse.

El *propósito* contempla las áreas del conocimiento humano hacia las que está destinada la aplicación y sus limitantes, como puede ser una actividad específica como la graficación hasta la apertura del lenguaje para desarrollar aplicaciones de bases de datos, científicas, etc.

La *orientación* está asociada a la forma en que las instrucciones que conforman el código fuente se organizan y se estructuran internamente, así como la forma en que se ejecutarán.

El caso de la orientación a objetos destaca por su creciente popularidad en la actualidad debida a sus promesas de reutilización de código, de construir aplicaciones modulares, distribuidas y fácilmente interconectables con un mayor nivel de abstracción[3].

Tiene un marcado contraste con lenguajes no orientados a objetos como C y Basic de tipo procedural y que basan su estructura en el concepto de procedimiento o función.

La función deriva de la filosofía de seccionar un problema en subproblemas más simples y que se enfocan a una tarea específica a través de un bloque de instrucciones que operan sobre los datos que recibe llamados argumentos y produce un resultado. Así el programa se integra por una sucesión de funciones y las llamadas entre ellas que pueden estar incorporadas en el lenguaje, en el sistema operativo o creadas por el usuario.

La programación estructurada busca implantar un procedimiento como solución a la cuestión: ¿cómo se resolverá el problema?, y trata de guiar secuencialmente la estructura del código. Mientras que la programación orientada a objetos (POO) no se preocupa por las instrucciones ni su orden, sino por el quién efectuará el procedimiento al abstraer la naturaleza de las cosas y la conexión que hay entre ellas.

[3] La abstracción es la capacidad de notar sólo lo esencial, la acción misma sin detenerse en los detalles, de enfocarse en el ¿qué hace? y no en el ¿cómo lo hace?

1.3.1 Programación orientada a objetos

Conceptos básicos

La POO ofrece la ventaja de representar entidades de la realidad sobre las cuales realizar operaciones y provocar comportamientos. Tales entidades poseen características propias que las distinguen, acciones que las manipulan y tienen su equivalente en la computación en los objetos.

Un programa orientado a objetos se constituye por un conjunto de objetos que interactúan entre si y cuya funcionalidad, entradas y salidas permiten la resolución de un problema como se observa en la figura 1.5.

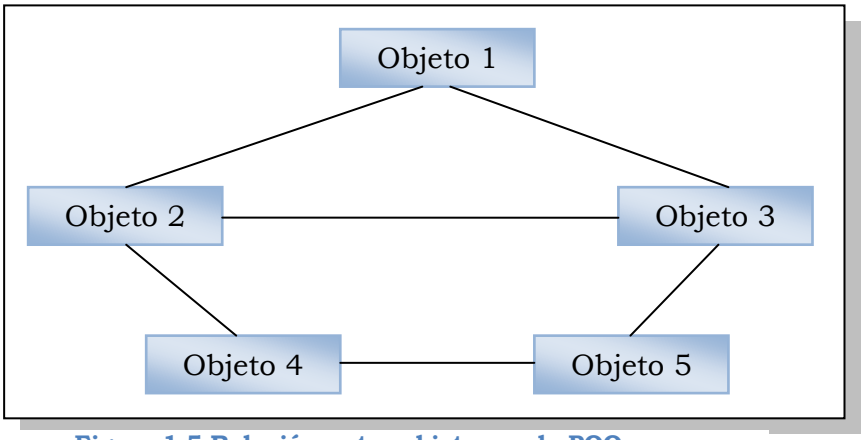

Figura 1.5 Relación entre objetos en la POO

Un objeto es la representación en un programa de un concepto y contiene toda la información necesaria para abstraerlo: datos que describen sus atributos y operaciones que pueden realizarse sobre los mismos.[4]

Los **atributos** también conocidos como datos expresan la información que se conoce del objeto, su forma, apariencia y características a través de valores.

Los **métodos** son comportamientos equivalentes a las funciones en los lenguajes procedurales que representan las acciones que el objeto podrá realizar al manipular sus atributos.

[4] CUENCA, Pedro, *Programación en Java,* p. 272.

A pesar de ser un estilo muy diferente de programar, la POO sigue también una metodología que inicia en el planteamiento del problema a partir de los datos de entrada, el proceso y la salida esperada; identificar los objetos como sustantivos; el diseño del algoritmo en base al diagrama de clases y las pruebas de escritorio.

Una vez explicado el concepto de objeto será más sencillo explicar el de clase que es fundamental en el diseño porque con una basta para representar objetos iguales que hacen lo mismo, es decir, con características y comportamientos comunes.

La **clase** es una representación abstracta y formal que describe un conjunto de objetos del mismo tipo. Puede verse como un molde de helado o gelatina la cual posee atributos definidos como forma, tamaño y sabor que sirve para crear muestras que si bien poseen el mismo tipo de atributos, sus valores pueden cambiar como el caso del sabor como se observa en la figura 1.6.

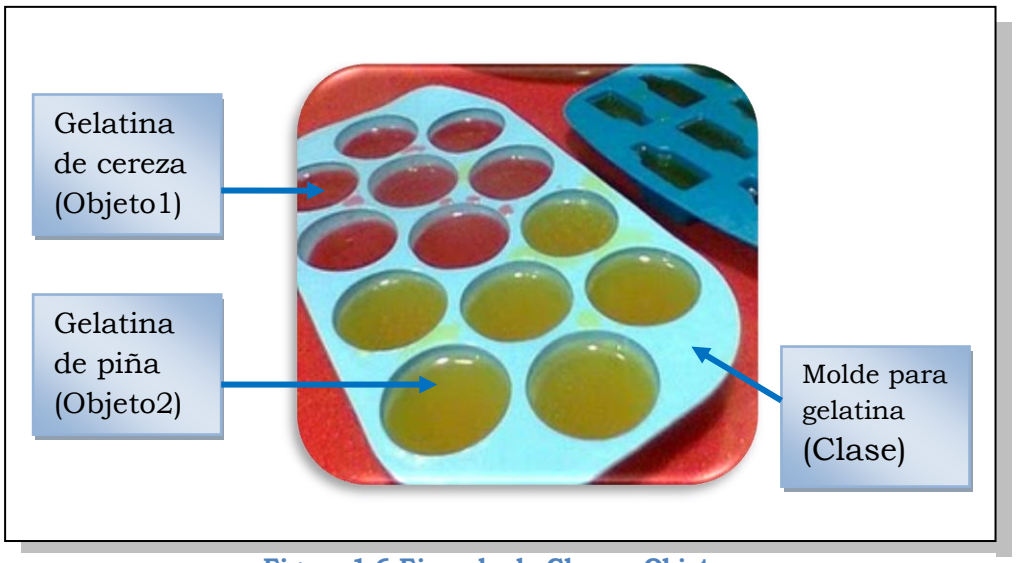

Figura 1.6 Ejemplo de Clase y Objetos

En términos de computación, la clase será la plantilla que define los atributos y métodos (llamados miembros de clase) comunes que serán copiados a cada objeto (llamado instancia) creado a partir de ella pero que tendrá valores propios.

Por ejemplo en la figura 1.7 se observa un diagrama de la Clase Empleado con atributos y métodos definidos, de ésta se crean dos instancias con valores distintos que representan a dos empleados.

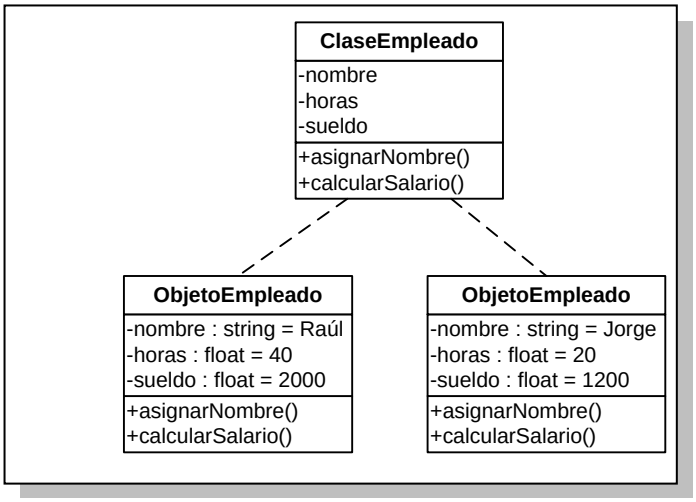

Figura 1.7 Diagrama de Clase y Objetos

Las clases constituyen la unidad mínima modular que permiten crear instancias, nuevos tipos de datos y estructurar código. Se diseñan teniendo en mente el código y los datos de forma conjunta, imaginando si son leídos o generados, así como los métodos que se encargarán de acceder a ellos.

Los objetos no están aislados, hacen uso de métodos para comunicarse con otro objeto pidiéndole que realice cierta acción y con los datos que contiene a través de mensajes.

Para que una aplicación realmente esté desarrollada orientada a objetos y no sólo basada en ellos debe cumplir con tres condiciones: encapsulamiento, herencia y polimorfismo.

El **encapsulamiento** permite un control total sobre "quién" o "qué" puede acceder a los miembros de un objeto haciendo invisible su estructura interna y protegiendo su información frente a accesos sin autorización.

A pesar de que el objeto adquiere la cualidad de caja negra suele presentar métodos de acceso para que otros objetos consulten o modifiquen sus atributos de una forma segura sin la necesidad de dar a conocer como se integra ni su comportamiento dentro de sí mismo. Observe la figura 1.8 donde la clase empleado posee miembros de tipo privado (su nombre está precedido por signo

"-") invisible para otras y de tipo público (su nombre está precedido por signo "+") a la vista y disposición del resto.

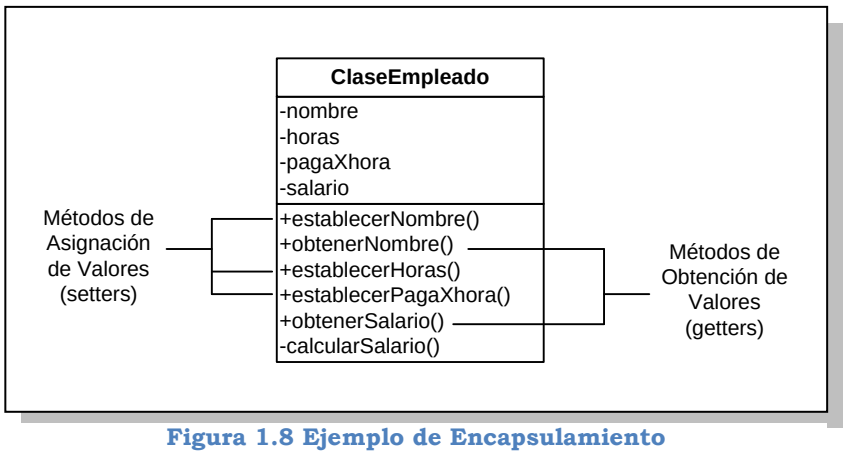

Figura 1.8 Ejemplo de Encapsulamiento

El encapsulado reduce la complejidad al usar un objeto dado que sólo se emplea su interfaz de comunicación sin prestar atención a su implementación. Se distinguen dos tipos de interfaces esenciales, la pública constituida por los métodos que son conocidos y sirven de enlace con otros objetos; y la interfaz privada que incluye lo necesario para funcionar y comunicarse internamente siendo visible sólo por objetos de la misma clase o incluso únicamente por el objeto mismo. Cualquier intento de otro objeto por acceder a la parte privada arrojará un error del compilador que garantiza el cumplimiento de los modificadores de acceso que además de ser público y privado, también se usan de tipo protegido, estático o abstracto.

El ser humano abstrae los sistemas complejos de forma jerárquica dividiéndolos sucesivamente en sistemas más simples hasta conseguir su unidad mínima. Puede verse de forma similar a un árbol genealógico identificando primero al objeto más genérico de la especie y después a sus derivados.

La **herencia** se basa en este principio para organizar jerárquicamente las clases de un programa, donde una clase puede crearse a partir de una existente llamada clase padre o superclase y tomar todos sus atributos y métodos.

No obstante, la clase derivada o hija puede redefinir los miembros heredados o añadir nuevos que la especializan en cierta tarea y delimitan su accionar. Así se logra simplificar diseños y reutilizar código en común.

La clase padre debe diseñarse con cuidado pues sus miembros afectarán a todas las clases herederas, por lo cual si un error es encontrado o se pretende redefinir un método basta con corregirlo en la clase padre y los cambios se verán reflejados inmediatamente en sus descendientes.

La clasificación de jerarquías con la herencia es una tarea compleja dado que deben establecerse las relaciones y agrupaciones más convenientes, seleccionando los métodos y atributos más genéricos y comunes a todas las clases hijas lo que convierte al diseño en una actividad de prueba y error. Una técnica que ayuda a identificar está relación se basa en la expresión "es un" que se observa en "automóvil es un vehículo" pero sin caer en la expresión "tiene un" como por ejemplo "Juan tiene un automóvil".

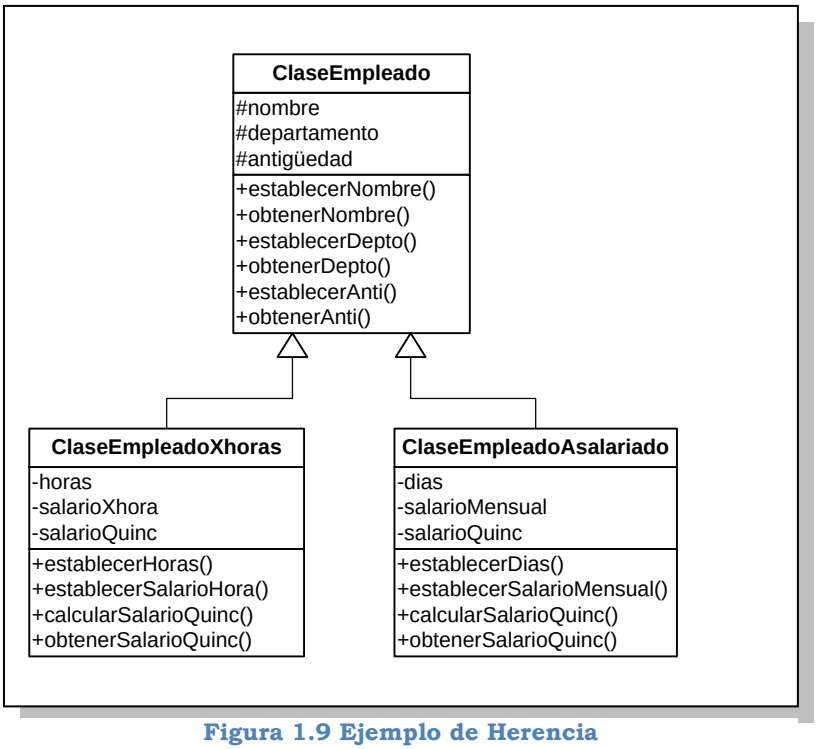

Figura 1.9 Ejemplo de Herencia

Al retomar la clase genérica Empleado, pueden extenderse dos clases que hereden sus miembros pero se especialicen en dos tipos de empleado diferente que poseen atributos y métodos propios. Por un lado se tiene el empleado que trabaja por horas cuyo contexto se define por el número de horas trabajadas y el salario correspondiente por fracción de tiempo; por otro se tiene al empleado asalariado con un salario mensual fijo como se observa en la figura 1.9.

El **polimorfismo** es la propiedad que permite al objeto modificar su forma y adaptar su comportamiento en función del ambiente y la exigencia. Se observa cuando un método realiza la misma operación sin importar el número de argumentos que recibe o cuando éstos son de diferente tipo de dato, tal caso recibe el nombre de Overloaded o sobrecarga. Otro caso de polimorfismo es el de Overridden o sobreescritura que consiste en reemplazar un atributo o método heredado por el que realmente necesita la clase, es decir, la implementación del código cambia de acuerdo a la nueva clase pero a fin de respetar el encapsulamiento es necesario respetar el número de argumentos y tipos de datos.

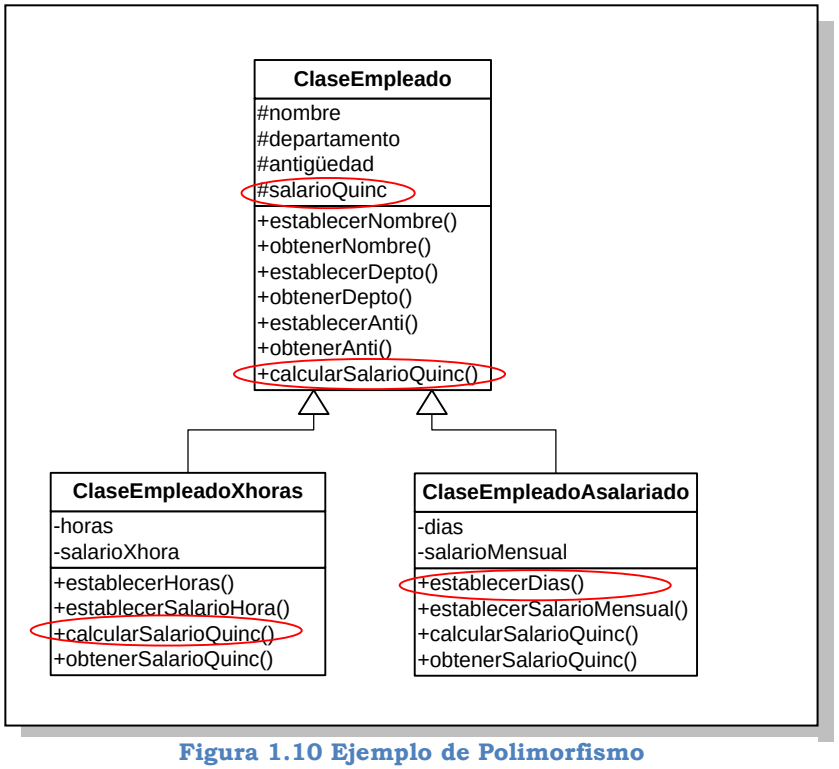

Figura 1.10 Ejemplo de Polimorfismo

En la figura 1.10 se observa el uso del polimorfismo sobre la clase Empleado, que se ve modificada al incluir el atributo salarioQuinc y el método calcularSalarioQuinc que suben de nivel para poder ser heredados. Las clases hijas modifican este método de acuerdo a sus especificaciones, mientras que en la ClaseEmpleadoXhoras el cálculo utiliza las *horas* trabajadas por el *salarioXhora*; en la ClaseEmpleadoAsalariado el cálculo resulta del *salarioQuinc* fijo en función de los días trabajados. En conclusión el método lleva el mismo nombre, hace la misma función pero el mecanismo y los datos que emplea cambian.

Ventajas:

- Abordar la resolución de un problema de forma metódica que lleva de lo general a lo particular.
- El diseño tiende a reflejar fielmente las condiciones del problema dado por el mayor nivel de abstracción.
- Reutilización del código que reduce la extensión del programa.
- Las clases pueden desarrollarse de forma separada dado que no influyen en las demás sólo tomando en cuenta sus interfaces, hasta las dispuestas por otros programadores en Internet.
- Mejora el mantenimiento del software y su flexibilidad al preservar la interfaz de una clase a pesar de re-implementarse.
- Modificadores de visibilidad para ofrecer seguridad y condiciones de acceso a las clases.

Desventajas:

- Requiere mucho mayor planeación, buen diseño y técnica de programación.
- La abstracción y la jerarquía de clases no son tareas triviales.
- El análisis del problema es más complejo dado que deben identificarse todos los elementos que componen las clases y la relación entre ellos.
- El tiempo de ejecución presenta un leve retardo ya que los métodos usados en la herencia llevan más tiempo que la llamada a una función de lenguaje estructurado.

1.3.2 Lenguaje de programación Java

¿Qué es Java?

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems cuya sintaxis está basada en C++ pero elimina todos los elementos causantes de error como el manejo de apuntadores.

Tiene su origen en el mercado de la electrónica de consumo y los electrodomésticos cuyos reducidos recursos de cómputo y memoria requerían un lenguaje sencillo y de poco tamaño.

Los lenguajes como C y C++ se enfrentaban además al surgimiento de nuevos chips de menor costo pero con arquitecturas diferentes lo que conducía a re-compilar el software frecuentemente.

Java pretende resolver el problema de arquitecturas incompatibles tanto en equipos de cómputo, sistemas operativos, sistemas de ventanas y dispositivos electrónicos, es decir, que la aplicación corra sobre cualquier plataforma de hardware y software, pero además brindar soporte para aplicaciones en red y distribuidas.

Sin embargo, el ámbito de la Web fue quien lanzó a Java al éxito al dotarlo del dinamismo del que carecían los documentos Web escritos en lenguaje HTML que constaban originalmente de texto, imágenes y vínculos a otros documentos.

Posteriormente evolucionó para incluir contenido dinámico como audio y video, no obstante era necesario contar con plug-ins en el navegador para interpretarlo o en su defecto descargarlo y así usar el programa apropiado instalado en el equipo.

Con el auge del contenido interactivo se evidenció la necesidad de un lenguaje de programación con las facultades para incluir en las páginas Web interfaces de usuario y conectar los datos recibidos en formularios con los repositorios necesarios.

Surge Java para sustituir a los limitados lenguajes, proporcionar a los navegadores la capacidad de ejecutar programas y reproducir contenido multimedia y desarrollar todo tipo de aplicaciones.

Tipos de aplicaciones en Java

- *Applets.*

Son mini aplicaciones que se ejecutan dentro de un navegador al cargar una página HTML desde un servidor Web y que no requieren instalación.

El navegador puede interpretarlos dado que se incrustan en la página con una etiqueta al igual que una animación de flash.

Mediante los applets se logra también integrar sonidos, video y contenido 3D.

- *Aplicaciones de Consola.*

Programas independientes que se desarrollan al igual que los lenguajes tradicionales pero limitados a sólo texto en una línea de comandos como en el ambiente de MS-DOS.

- *Aplicaciones Gráficas.*

Utilizan las clases destinadas para gráficos como AWT y Swing para brindar una amigable interfaz de usuario a partir de un entorno visual sencillo con ventanas, imágenes y botones de acción.

- *Servlets.*

Programas sin interfaz gráfica que se ejecuta en el servidor Web y devuelve una página HTML.

¿En qué se usa?

La versatilidad, portabilidad y potencia de Java han colocado aplicaciones en diversos ámbitos de la vida común y en las tecnologías de la información, se encuentran en consolas de videojuegos, menús de películas en alta definición, integración de juegos y medios en teléfonos móviles, navegadores y servicios Web, procesamiento de datos de foros en línea y tiendas, comercio electrónico, facilitando la carga de imágenes en redes sociales, aplicaciones de escritorio y empresariales, chat, juegos en línea, dispositivos electrónicos como impresoras y cámaras, etc.

Proceso de compilación e interpretado

La independencia de la Plataforma y eficacia en la diversidad de aplicaciones se deben a que Java realiza tanto compilación como interpretación de código lo que convierte al ambiente de desarrollo un tanto particular como se observa en la figura 1.11. Para programar en Java se requiere:

- *Entorno de desarrollo.*

Sun distribuye gratuitamente el JDK (Java Development Kit) que incluye un conjunto de programas y librerías necesarios para codificar, compilar y ejecutar los programas en Java.

- *Editor de programas.*

La codificación puede escribirse en cualquier editor de texto plano o hacer uso de herramientas de desarrollo de tipo IDE[5] como JBuilder, NetBeans, JCreator, Eclipse y Visual Studio.

- *Compilador.*

En el JDK el compilador se llama javac.exe y está encargado de realizar un análisis de sintaxis del código fuente con extensión *.java devolviendo después de la compilación archivos con extensión *.class con código intermedio que recibe el nombre de ByteCode.

- *Biblioteca de clases.*

Un programa en Java no parte de cero, es necesario invocar las clases necesarias que le otorgan cierta funcionalidad ya desarrollada y que vienen predefinidas en la biblioteca de clases o comúnmente llamada API (Application Programming Interface). Al instalar el entorno de desarrollo quedan disponibles en la carpeta include dentro de jdk y están organizadas en una jerarquía de clases a través de paquetes contenidos en archivos con extensión *.H. Además

[5] IDE: Entorno Integrado de Desarrollo que provee un marco de trabajo amigable para uno o más lenguajes de programación al incorporar en la misma aplicación un editor de código, un compilador, un depurador o debugger, un constructor de interfaz gráfica y ambiente de ejecución para evitar cambiar de aplicaciones.

de éstas puede hacerse uso de clases comerciales o desarrolladas por otros programadores.

- *Máquina Virtual de Java (JVM).*

El código intermedio resultante de la compilación no es código ejecutable en ninguna plataforma de hardware pues no corresponde con las especificaciones de procesadores reales, pero si con una máquina virtual de carácter puramente teórico con set de instrucciones, tamaño en bits y tipos de datos bien definidos. La llamada Máquina Virtual Java (también conocida como JRE o entorno de ejecución) se encarga de interpretar el código intermedio pasándolo a código máquina del procesador donde se está ejecutando. En este punto es donde se consigue la portabilidad dado que una vez compilado el código fuente y obtenido el código intermedio ya no es necesario modificarlo para cambiar de plataforma, sólo se requiere que posea el entorno de ejecución adecuado que ya es dependiente de la máquina.

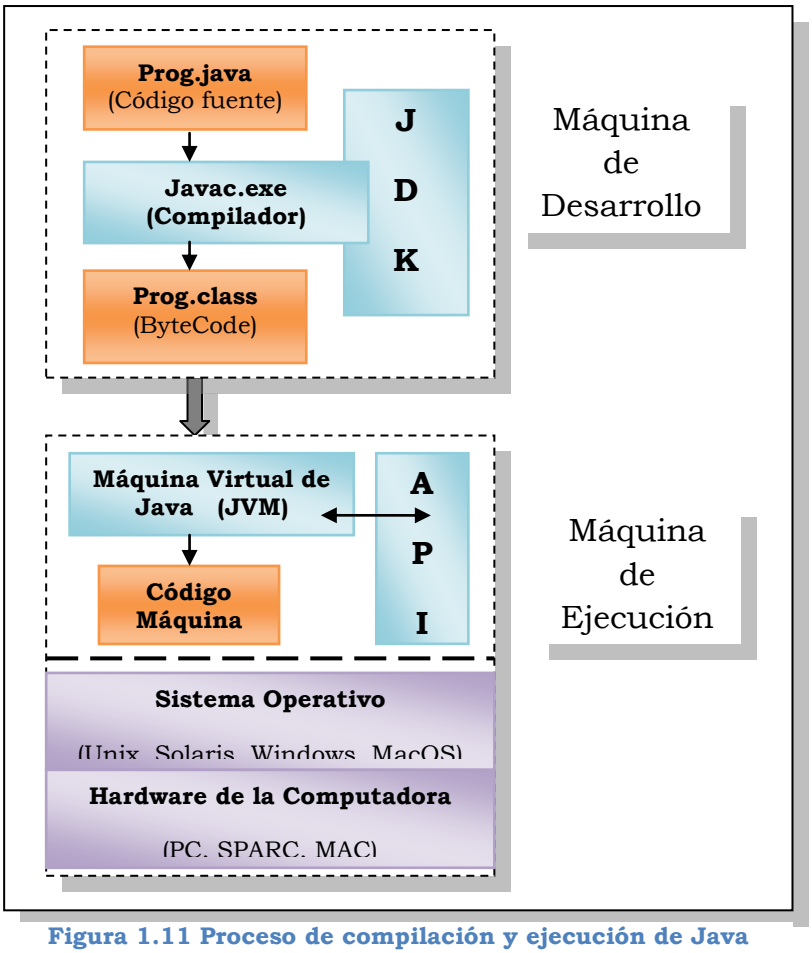

Figura 1.11 Proceso de compilación y ejecución de Java

Características

- *Sencillo.*

Aunque recoge sólo las características esenciales de C y C++ puede abordar cualquier tipo de problema, además es de rápida adopción y permite construir aplicaciones ligeras que no consumen muchos recursos.

Reduce los errores al eliminar el uso de apuntadores, registros, macros y la necesidad de liberar memoria y el tiempo perdido derivado de esto puede destinarse al diseño del programa.

- *Orientado a Objetos.*

A comparación de los programas en C que ejecutan sus instrucciones de forma secuencial y de C++, que a pesar de estar pensado como extensión de C para emplear entidades independientes pueden no usarse, Java es un lenguaje que basa plenamente su estructura interna y funcionalidad en los objetos y la comunicación entre ellos. Además soporta las características esenciales de esta orientación como la herencia, encapsulación y polimorfismo.

- *Distribuido.*

Las aplicaciones Java pueden ajustarse a la arquitectura cliente – servidor, donde el programa puede actuar como servidor permitiendo el enlace con otros o bien con bases de datos, mientras que el usuario accede a estos servicios mediante una interfaz que puede ser brindada por un applet.

Con esto se evita la instalación de software y la descarga de actualizaciones ampliando su alcance dado que la corrección de errores y mejoras sólo se realizan del lado del servidor pasando inadvertidas para el usuario.

Además de soportar los servidores más populares, las librerías estándar de Java favorecen el desarrollo de aplicaciones distribuidas donde los servicios pueden estar alojados en diferentes sitios pero comunicados entre sí por la red.

- *Robusto.*

La robustez de un programa radica en su resistencia a errores y su capacidad para manejarlos con el fin de evitar un alto en la ejecución de forma repentina.

Por tanto deben realizarse verificaciones tanto en tiempo de compilación como de ejecución.

Entre las singularidades de Java es que obliga a la comprobación de tipos y la declaración explícita de métodos, además realiza el manejo de la memoria, la comprobación de límites de arreglos y la verificación del código intermedio.

Otro aspecto importante es que pueden recogerse los errores que se produzcan en tiempo de ejecución por medio del uso de excepciones que permiten darles posibles soluciones a los que se piensen puedan ocurrir y así garantizar la continuidad del programa.

- *Independiente de la Plataforma.*

Un programa puede ejecutarse sin cambios en cualquier ambiente de hardware y software sin importar en que máquina fue generado, abarcando desde una PC, Mac o servidor hasta dispositivos electrónicos como televisiones o teléfonos móviles. Esto es posible porque el código pre-compilado es independiente de la plataforma por lo cual sólo se requiere el entorno de ejecución o JRE (conocido también como Java Runtime Environment, Runtime, Java Virtual Machine) que se encargará de interpretar las instrucciones y está disponible para los sistemas operativos Solaris, SunOS, Windows, MacOS, Linux.

- *Seguro.*

Java se basa en 3 herramientas para garantizar una de sus mayores fortalezas, la seguridad.

1. *Verificador de código intermedio.*
   Comprueba el correcto comportamiento del código y el cumplimiento de las reglas de Java, detecta fragmentos de éste de carácter ilegal que viole los derechos de acceso o convierta tipos y clases de un objeto.

2. *Verificador de Clases.*
   Proporciona las clases necesarias y se asegura que sean las originales de Java buscando primero entre las locales y después entre las procedentes del exterior, impidiendo así los Caballos de Troya y la suplantación de clases.
   Puede verificar además el origen del código mediante firmas digitales validando sus privilegios antes de instanciar un objeto y cargarlo en memoria.

Por otro lado, en el caso de los applets existe una mayor seguridad dado que los intérpretes de los navegadores Web presentan más restricciones que bloquean la ejecución de aplicaciones nativas de la plataforma, abrir archivos o usar el equipo como "zombie[6]" sin el conocimiento del usuario.

3. *Administrador de Seguridad.*
   Permite al usuario elegir niveles de seguridad para la ejecución de programas y de confianza de acuerdo al sitio de procedencia del código que puede ser identificada por ejemplo mediante firmas digitales.

En conclusión, las medidas de seguridad evitan que los programas accedan directamente a los recursos de la computadora como memoria y disco duro previniendo su interacción con ciertos virus. Además cada operación es vigilada tanto en la compilación como al momento de ser ejecutada por el intérprete.

- *Portable.*

El concepto de portabilidad se centra en la ya mencionada independencia de la plataforma del código compilado que se extiende también a los tipos de datos, los cuales ocupan el mismo espacio a diferencia de C y C++, así por ejemplo un entero siempre será de 32 bits en complemento a 2.

Asimismo se recurre a un sistema abstracto de ventanas aprovechando la semejanza que presentan en cuanto a componentes y funcionalidad en todos los sistemas operativos.

- *Interpretado.*

Java involucra 2 procesos casi antagónicos en la programación para obtener una aplicación, la compilación y la interpretación de código, éste último permite la independencia de la plataforma y el uso de menos recursos pero a costa de una reducida velocidad de ejecución donde en comparación con C es de 10 a 30 veces más lento.

Sin embargo, deben tomarse en cuenta los mecanismos de seguridad mencionados anteriormente y la búsqueda de la mayor optimización durante la compilación los parcialmente causantes de esta lentitud.

[6] Equipo que al ser infectado por algún tipo de malware es usado por otro individuo sin conocimiento del usuario para infectar equipos, distribuir SPAM o generar tráfico con más zombies para atacar un servidor.

- *Multithreading.*

Haciendo alusión a las traducciones más populares multithreading quiere decir multihilo, multitarea o multihilvanado. Los threads por tanto son pequeños procesos independientes que integran uno mayor.

Un programa en Java puede crear varios threads o hilos de ejecución que se ejecutan en paralelo con una tarea específica cada uno, por ejemplo, la recepción de datos del usuario por el teclado, la lectura de archivos, cálculos, la descarga de cierto contenido Web como imágenes mientras se observa el resto de la información sin tener que esperar a que concluya, etc.

Además los threads pueden asignarse a diferentes microprocesadores si se cuenta con sistemas de este tipo. No obstante, el rendimiento multitarea es acorde al sistema operativo y a su capacidad para manejarlo.

- *Dinámico.*

Bajo el supuesto de que un programa en Java no es un ente compacto cada vez que se compila sino un conjunto de módulos que se cargan y se conectan hasta el tiempo de ejecución, es posible actualizar las librerías añadiendo nuevos métodos y atributos a sus clases sin afectar la aplicación actual. En caso de que ésta se ejecute en red serán traídos automáticamente los fragmentos necesarios o que no se entiendan.

- *Difundido.*

Al ser un lenguaje versátil en cuanto a las aplicaciones y áreas de desarrollo abarca el entretenimiento, programas simples y empresariales, teléfonos móviles, etc. Tal alcance lo ha llevado a contar con una vasta documentación, fuentes en Web, foros y libros de gran dominio entre los programadores.

- *Garbage Collector.*

Una carga importante para el programador de C++ es la administración de la memoria de forma manual, donde la falla en su asignación y desalojo puede llevar a fugas de memoria, consumir más de la necesaria o cuelgues del programa.

Java retira esta responsabilidad al programador mediante el recolector de basura que de forma automática e invisible asigna y libera los recursos.

Cuando el programa crea un objeto pide al sistema la memoria necesaria siendo vigilado su ciclo de vida por el entorno (Java Run-Time) a través de un thread de baja prioridad que lo examina para determinar en qué momento se ha dejado de usar y no quedan referencias hacia él, entonces lo borra y libera la memoria que estaba ocupando.

Ventajas:

- El código puede ejecutarse sin cambios en cualquier dispositivo que cuente con el entorno Java respectivo.
- Al ser orientado a objetos permite resolver un problema metódica y modularmente facilitando el desarrollo cooperativo.
- Menos errores al eliminar el manejo de apuntadores y la gestión de la memoria dado que su asignación y liberación se realizan de forma automática.
- Admite el uso de técnicas para optimizar el desempeño y acelerar la ejecución.
- Facilita la inclusión de contenido interactivo en páginas Web sin el uso de paquetes multimedia o la instalación de plug-ins.
- La librería de clases es estándar para todas las plataformas y permite afrontar problemas de cálculos matemáticos, acceso a bases de datos, cómputo gráfico, etc.
- Desarrollo de aplicaciones modulares y distribuidas.

Desventajas:

- La más representativa es la reducida velocidad de ejecución debida a la Máquina Virtual de Java que interpreta constantemente el código intermedio consumiendo buena cantidad de recursos de procesamiento y ofrece un bajo rendimiento a comparación de los lenguajes compilados.
- A pesar de la similitud con C++, Java es un lenguaje por sí mismo con sus propias reglas que deben aprenderse.

Servlets

El uso de la Web como se ha mencionado se enfocaba a la consulta de documentos estáticos mediante el navegador del usuario (que bien puede tomar el nombre de cliente). Tales documentos se alojaban en una computadora que fungía como servidor el cual se limitaba a enviar el archivo *.HTML a través de la red cuando el usuario lo solicitaba directamente por su dirección daba click sobre algún vínculo.

La interactividad en este proceso era mínima debido a que el usuario sólo tenía el rol de receptor de la información y no tenía posibilidad de hacer peticiones personalizadas o dejar información propia.

Con el fin de mejorar tal situación se añadió más inteligencia tanto al cliente (navegador del usuario) como al servidor (emisor de páginas HTML).

En el lado del cliente se incorporaron los applets que son verdaderas clases de Java que se cargan y ejecutan en el navegador.

Del lado del servidor se incluyeron las páginas dinámicas de la mano de los formularios HTML. Se les llama dinámicas porque no existen en el disco duro del servidor, son generadas por éste haciendo llamadas a programas específicos conocidos como *scripts* que ejecutan instrucciones de recolección de datos de los formularios y acceso a bases de datos devolviendo como resultado al usuario la página construida para responder a su petición.

Los formularios HTML invierten el sentido del flujo de información ofreciendo al usuario una interfaz para enviar datos con la ayuda de controles simples como cajas de texto, botones de opción y selección.

El formulario transporta el nombre del programa en el servidor que será llamado para procesar los datos, recibirlos e introducirlos en la base de datos correspondiente y que a su vez enviará al cliente una página dinámica como respuesta indicándole que las tareas fueron realizadas con éxito, que se ha producido un error o que falta algún dato. La figura 1.12 muestra el esquema de funcionamiento de un script.

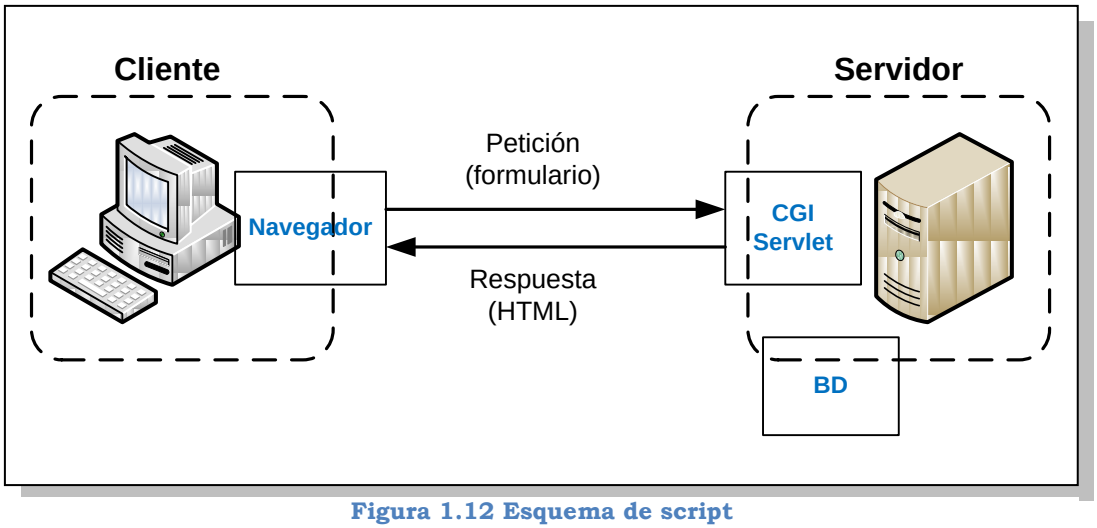

Figura 1.12 Esquema de script

El uso de scripts se debe a varias razones:

- Al residir en el servidor se logra la independencia del navegador.
- Tareas complejas se ejecutan más rápido del lado del servidor.
- Mayor seguridad en la ejecución bajo el control del administrador.
- La página Web se basa en los datos enviados por el usuario.
- La página Web contiene datos que cambian frecuentemente.

La primer tecnología usada en los scripts fueron los programas CGI (Common Gateway Interface) escritos en lenguajes como Perl, C y C++. A pesar de ser simples y ampliamente soportados presentaban la gran desventaja de que el servidor lanzaba una copia del programa por cada petición del usuario, por tanto si el sitio era muy visitado corría el riesgo de saturarse.

Java ofrece una tecnología alternativa que resuelve este problema con la programación de Servlets.

Los Servlets son programas en Java que se ejecutan en servidor Web o en el contenedor Web de un Servidor de Aplicaciones y fungen como una capa intermedia entre la petición de un cliente HTTP en un navegador Web y las bases de datos y aplicaciones del servidor devolviendo finalmente una respuesta al cliente.

Puede compararse con un applet que es cargado y ejecutado en el navegador, mientras que el Servlet es cargado y ejecutado por el servidor.

Ciclo de vida y finalidad del Servlet

1. Leer datos introducidos por el usuario en un formulario o applet.
2. Recoger petición en el servidor, si se observa un Servlet delega la acción al contenedor Web.
3. Obtener información adicional de la petición relacionada al cliente como navegador, cookies, nombre del equipo, etc.
4. Ejecutar el Servlet.
5. Generar resultados a partir de la comunicación con una base de datos, una llamada remota o invocando aplicaciones existentes.
6. Insertar la información resultante en una página HTML.
7. Establecer los parámetros adecuados del documento de respuesta.
8. Devolver el documento al cliente.

¿Qué se necesita para desarrollar Servlets?

Los Servlets son clases de Java que se codifican, compilan, instalan y ejecutan en un espacio restringido del servidor mediante un motor especial que se encargará de vigilar el ciclo de vida del Servlet, crearlo y destruirlo.

Por tanto se requiere instalar un motor de Servlets, en el caso de contar con el JDK en el equipo instalar el contenedor Web Jakarta-Tomcat que permite ejecutar Servlets y páginas JSP. En caso contrario, instalar el J2EE SDK que cuenta con el JDK y un servidor de aplicaciones.

Características:

- Extienden la portabilidad de Java al ser independientes del servidor y de su sistema operativo.
- Distribuyen el trabajo llamando a otros Servlets con tareas específicas redireccionando las peticiones.
- Fácil obtención de información del cliente y procesamiento de formularios.
- Uso de cookies y rastreo de sesiones.
- Actúan de enlace entre el cliente y una o más bases de datos.
- Permiten la generación dinámica de código HTML.
- No requieren soporte para Java en el navegador del cliente porque toda su ejecución se realiza del lado del servidor.
- Son cargados sólo la primera vez que se solicitan y permanecen en memoria. Las siguientes peticiones crean hilos de ejecución.

Las ventajas de los Servlets sobre CGI se observan en la tabla 1.1.

CGI	SERVLET
Si hay "n" peticiones simultáneas al mismo programa se carga "n" veces en memoria.	Una instancia del servlet en memoria y "n" subprocesos.
El programa finaliza cuando da respuesta a la petición lo que retarda los procesos y las conexiones a las bases de datos se pierden.	Permanece en memoria aún después de dar respuesta.
No existe comunicación directa con el servidor Web.	Optimización de conexiones a bases de datos y compartición de datos al comunicarse directamente con el servidor Web.
No hay verificación de código por lo que es propenso a ataques o accidentes.	Existe verificación de límites de matrices y protección de la memoria.

Tabla 1.1 Servlets vs. CGI

JSP

La tecnología de JSP (Java Server Pages) está orientada a la creación de páginas Web con la inclusión del lenguaje Java.

Como ocurre con la mayoría de las tecnologías de programación de páginas dinámicas, se utilizan 2 sintaxis diferentes para distinguir la lógica de la presentación de la lógica de la aplicación: la primera emplea etiquetas HTML que dan forma y aspecto final a la página Web representando el contenido estático; la segunda invoca el lenguaje de programación como VBScript en ASP o Java en JSP con etiquetas especiales que se mezclan en la página Web conformando la parte dinámica, como resultado de ambas partes se tiene un archivo con extensión *.jsp.

Cuando un cliente pide una página JSP del sitio Web por primera vez, la página es pasada al motor JSP y lleva a cabo su traducción en un Servlet que devuelve un archivo *.class. Posteriormente el motor de Servlets carga la clase del Servlet y lo ejecuta generando el HTML dinámico que es enviado al navegador del cliente. El proceso puede observarse en la figura 1.13.

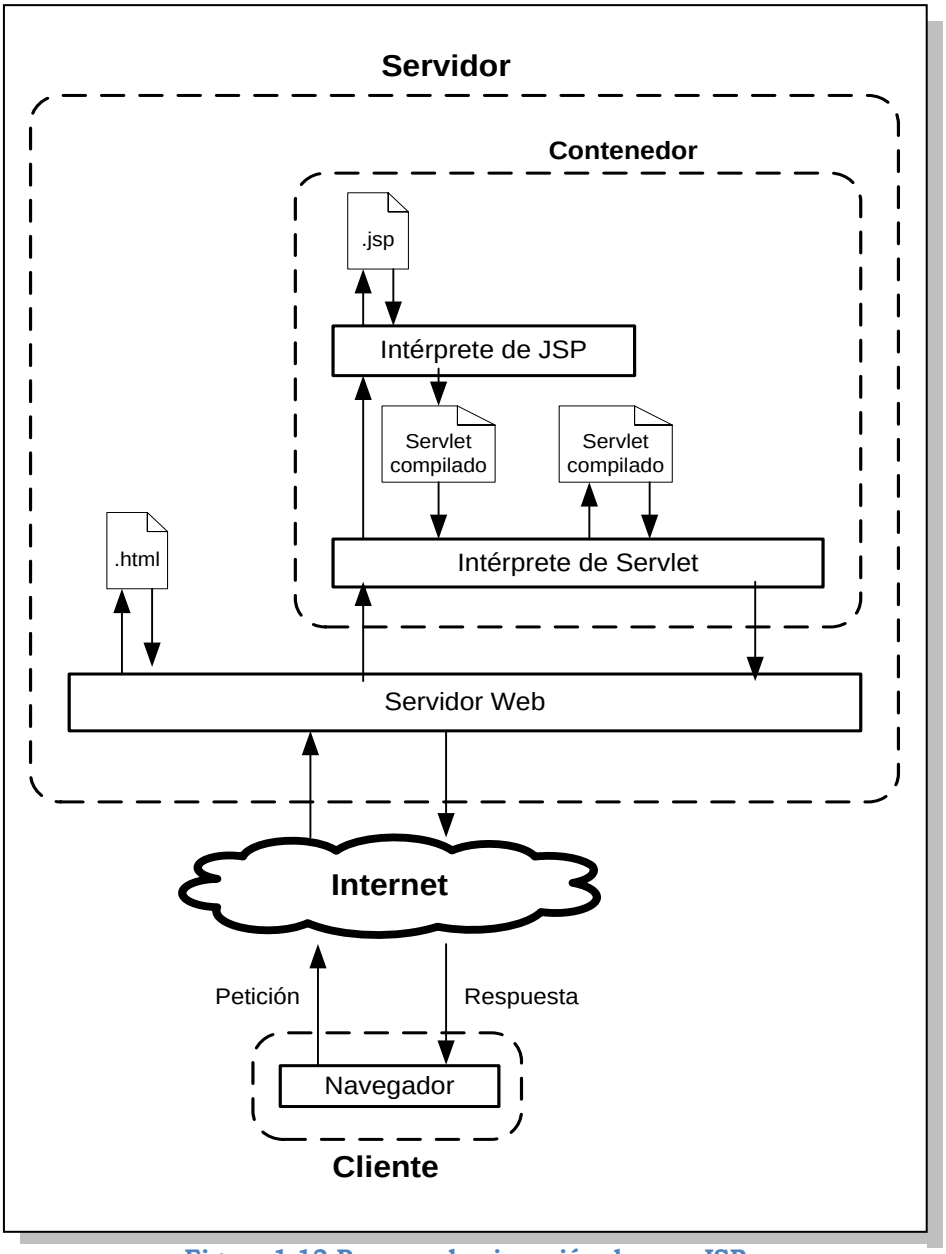

Figura 1.13 Proceso de ejecución de una JSP

Si la página es solicitada de nuevo el Servlet ya estará creado por lo que no es necesaria la conversión otra vez, a menos que el servidor detecte que la página de origen fue modificada, ésta volverá a ser compilada y convertida en Servlet.

En caso de detectar algún error, el contenedor envía un mensaje hacia el navegador del usuario.

Al emplear Java es imprescindible respetar las limitaciones y obligaciones del lenguaje a pesar de ser usado en el contexto de las páginas Web, no obstante cuenta con toda su potencia, portabilidad y agrega como ventaja la separación entre lo que se conoce como capa de presentación, capa de negocio y datos.

Al igual que los Servlets, la creación y ejecución de páginas JSP requiere de un entorno de desarrollo instalado y configurado correctamente, además de un servidor capaz de interpretar este tipo de páginas llamado contenedor que funciona sobre el servidor Web que recoge las peticiones y le pasa al contenedor las correspondientes a páginas JSP y Servlets.

Comparación con otras tecnologías

- *ASP (Active Server Pages).*

Páginas dinámicas escritas en VBScript (variante de Visual Basic) y JScript (variante de JavaScript) que carecen de portabilidad al estar atadas a sistemas operativos Windows y IIS. Actualmente forma parte de la plataforma .NET bajo la arquitectura cliente – servidor.

- *Servlets.*

Como se ha mencionado son programas en Java que son llamados para procesar datos y generar una respuesta pero su código es sólo visible y ejecutable del lado del servidor además de ser más complejo y laborioso, mientras que la página JSP devuelta al usuario mezcla el código Java con el HTML lo que podría considerarse inseguro pero más simple al separar el desarrollo en presentación y contenido.

- *PHP (PHP Hypertext Preprocessor).*

Lenguaje de tipo gratuito, opensource, multiplataforma y con soporte para las bases de datos más usadas que también se incrusta en las páginas Web dinámicas junto con etiquetas HTML.

- *JavaScript.*

A pesar de la similitud en el nombre no tiene relación con Java. Su dinamismo sólo se produce del lado del cliente y carece de soporte para programación de redes, formularios y bases de datos.

- *ColdFusion.*

Servidor de aplicaciones y lenguaje de programación para desarrollar páginas dinámicas, disponible para varias plataformas e interactúa con diversas bases de datos mediante SQL simple. No es de las más utilizadas pero es útil en la integración con otras aplicaciones como Flash para el que se creó originalmente.

JDBC

La gran parte de las aplicaciones y páginas Web hacen uso de las bases de datos desde que éstas surgieron para agendas, inventarios, validación de usuarios, publicaciones de periódicos en línea, información bancaria y catálogos de productos para comercio electrónico. Un simple sitio Web puede ofrecer gran cantidad de información y variedad de contenido gracias a una base de datos, algunas plantillas para páginas Web y el uso de páginas dinámicas.

Para llevar a cabo la conexión entre la base de datos y la interfaz de usuario en el sitio Web o en una aplicación se requiere un lenguaje de programación como Java que permita el enlace por medio de un driver JDBC.

JDBC (Java DataBase Connectivity) es un API de Java que permite la conexión con un manejador de bases de datos relacionales independientemente del tipo que éste sea; brinda al programador la facilidad de ejecutar instrucciones en el lenguaje estándar de consultas SQL (Structured Query Language) para crear, manipular y gestionar bases de datos.

Para emplear JDBC con un sistema manejador de bases de datos en específico se debe contar con el driver JDBC adecuado con el conjunto de clases distribuidas por el fabricante que haga la traducción de los mensajes de alto nivel de SQL a sentencias de bajo nivel propias de la base de datos usada. Con el API JDBC se deja atrás el escribir un programa que acceda a SyBase y otro para Oracle, con el driver correspondiente un único programa bastará para cualquier manejador soportado.

Existe otro estándar de conexión llamado ODBC (Open DataBase Connectivity) que pretendía conseguir en un inicio que el código fuera independiente del propietario de la base de datos pero al ser propiedad de Microsoft se limitaba a

ambientes Windows por lo que nunca logró plena portabilidad. Otras desventajas radican en que es el único medio de comunicación con algunos manejadores, es complejo de programar y está escrito en lenguaje C.

JDBC se basa en Java, en específico en los servlets para realizar 3 tareas:

1. Establecer la conexión a una base de datos ya sea o no remota.
2. Enviar sentencias SQL.
3. Procesar los resultados obtenidos de la base de datos.

Se emplea a los servlets por ubicarse en la capa media de una arquitectura de 3 capas, donde en todo momento se tiene el control de las operaciones con la base de datos, además porque corren del lado del servidor y no demandan la instalación de software en el cliente.

Tipos de drivers JDBC

La clasificación de drivers JDBC (figura 1.14) es elemental para conseguir un buen funcionamiento e interacción de la aplicación con la base de datos, se lleva a cabo a partir de dos preguntas: ¿el sistema de conexión está completamente escrito en Java? y ¿el cliente necesita un controlador adicional?

- *Tipo 1: Puente JDBC – ODBC.*

Se emplea cuando el manejador carece de soporte para JDBC y sólo dispone de drivers ODBC que si están disponibles en la mayoría de los sistemas. El puente realiza la conversión de los mensajes entre ambos pero es un proceso complejo, con demoras en el acceso, restricciones para red y presenta incompatibilidades de sus lenguajes de programación. Es preciso cargar el código binario del driver ODBC y el del manejador en la máquina cliente.

- *Tipo 2: API nativa.*

El driver contiene código Java que convierte las llamadas en métodos nativos de la base de datos, es decir, a la API propia de la base. En la máquina cliente debe cargarse el cliente de la base de datos como puede ser Sybase, Informix o DB2.

- *Tipo 3: Protocolo de Red.*

Las llamadas JDBC se traducen a un protocolo de red independiente del manejador y se comunican con una aplicación de capa intermedia en el servidor (middleware) que es capaz de conectar al cliente con diferentes manejadores de bases de datos. Es flexible, completamente desarrollado en Java y no necesita la instalación de software en el equipo cliente.

- *Tipo 4: Protocolo Nativo.*

Las llamadas se convierten en el protocolo de red usado por el SMBD. Es el mejor tipo debido a que está totalmente escrito en Java y comunica de forma directa la máquina cliente con el manejador. El driver es específico de la base que en la mayoría de los casos es proporcionado por el fabricante, por lo que si en algún momento debe migrarse de SMBD el driver también deberá cambiarse.

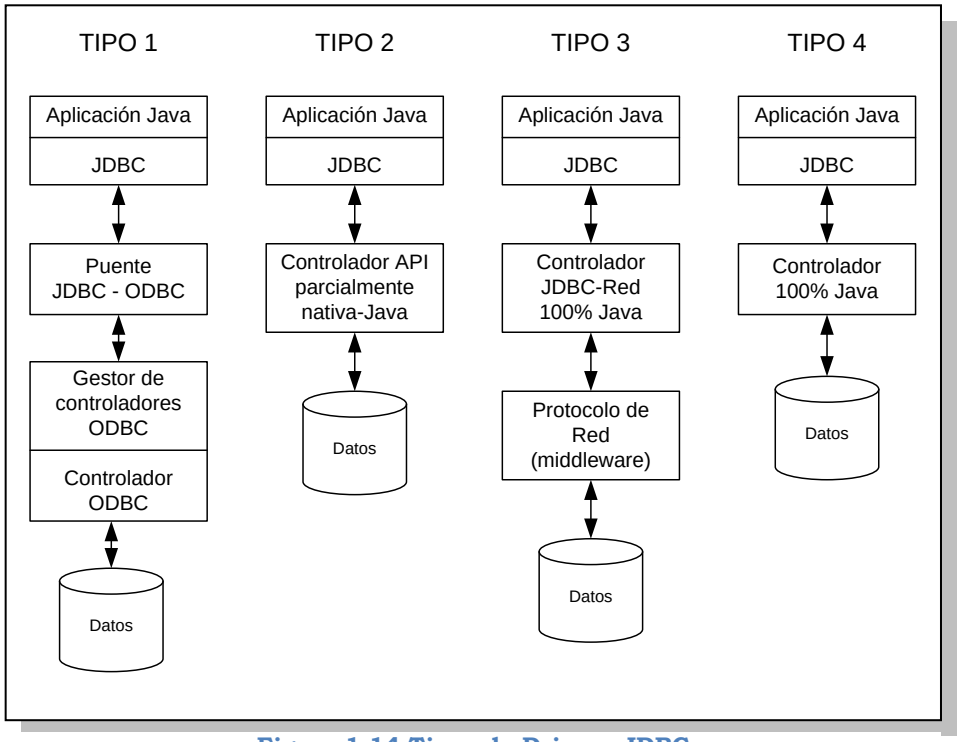

Figura 1.14 Tipos de Drivers JDBC

Ventajas de usar JDBC

- Las funciones y clases de la API son las mismas para cualquier manejador.
- El programador puede desarrollar una aplicación con un manejador sencillo y libre como MySQL, una vez probada llevarla a uno más potente como Oracle con la confianza de que el comportamiento será el mismo sin cambiar una línea de código.
- Actualización de múltiples registros con un sólo comando.
- Acceso a diversos manejadores en una misma transacción.
- Reutilización de conexiones a la base de datos en vez de una por comando.
- Soporta los modelos de acceso a bases de datos de 2 y 3 capas.
- Permite el paso de instrucciones propietarias del manejador aún cuando no respeten el estándar de SQL.
- Los resultados pueden ser devueltos como objetos y los posibles errores como excepciones.

1.3.3 Bases de datos

Una base de datos es una colección de datos clasificados y estructurados pertenecientes a un mismo contexto y almacenados sistemáticamente en uno o más archivos para su uso posterior.

Sistema manejador de bases de datos

Para almacenar y acceder a esta información existen los Sistemas Manejadores de Bases de Datos (SMBD o en inglés DBMS, DataBase Management System) que fungen como interfaz entre la base de datos, el usuario y las aplicaciones que la emplean. Son de utilidad en la manipulación de gran cantidad de información pero con las desventajas de un alto costo de hardware y de requerir varias personas para su administración.

Se distinguen 2 tipos de manejadores:

Libres:

- MySQL
- PostgreSQL

No Libres:

- Microsoft Access
- Microsoft SQL Server
- DB2
- Sybase ASE
- Oracle

Modelo de datos

Un modelo de datos es la descripción de los métodos para estructurar el almacenamiento y el acceso a la información.

Algunos modelos son:

- *Modelo Jerárquico.*

Las relaciones entre los datos son a partir de jerarquías en una estructura de árbol que inicia en un nodo raíz, donde cada nodo derivado o padre puede tener varios descendientes o elementos hijos sucesivamente hasta nodos sin hijos llamados hojas.

Se emplea cuando existe un gran volumen de información con datos compartidos que se relacionan entre sí mediante índices o punteros, o cuando los sistemas permanecen inalterables por mucho tiempo. Cabe señalar que es incapaz de manejar redundancia de datos.

- *Modelo en Red.*

Es una mejora del modelo jerárquico debido a que un hijo puede no tener padre o cualquier número de ellos.

Consta de dos conjuntos de datos: los registros y los enlaces que los relacionan.

- *Modelo Relacional.*

Utilizado en la actualidad para modelar situaciones reales y administrar datos dinámicos, permite la gestión de los datos en función de sus características lógicas sin tomar en cuenta la forma en que se almacenan a comparación con el modelo jerárquico y de red organizándolos en forma de tablas o relaciones.

La relación es un término que designa a una matriz de 2 dimensiones con filas y columnas. Cada columna de la tabla recibe el nombre de **atributo** o campo, mientras que las filas corresponden a los registros o **tuplas**.

Las tablas se enlazan entre sí mediante claves comunes, donde una clave es el conjunto mínimo de columnas que identifica a una fila dentro de una tabla.

La información contenida en la tabla puede manipularse a través de *consultas* escritas en un lenguaje estándar llamado SQL (Structured Query Language).

Normalización

Para el diseño de una base de datos relacional es imprescindible el proceso de normalización de los datos cuya finalidad es evitar la redundancia de datos y problemas de actualización, así como proteger la integridad de los datos.

Existen 5 formas de normalización de las cuales se explican 3 a continuación:

- *Primera forma normal (1FN)*

Un atributo o campo de una misma fila sólo puede contener un dato atómico e indivisible y no un conjunto de valores.

- *Segunda Forma Normal (2FN)*

Llamada de independencia funcional parte del cumplimiento de la primera forma y además marca que los atributos que no son clave deben depender completamente de la clave principal.

- *Tercera Forma Normal (3FN)*

Parte de la segunda forma normal y hace referencia a la existencia de una dependencia entre 2 atributos de una tabla donde uno será la clave de la tabla, visto de otra forma no debe existir dependencia funcional entre campos que no son clave.

Generalmente las bases de datos se normalizan hasta la tercera forma lo que representa un número alto de tablas con pocas columnas.

1.3.4 Sistemas operativos

Un sistema operativo son un conjunto de programas que actúan como intermediarios entre el usuario y el hardware con el fin de brindarle un ambiente para correr sus aplicaciones. Desempeña el papel de administrador de los recursos de hardware y software asignándolos de forma eficiente a programas y usuarios para la realización de sus tareas.

El sistema operativo (SO) combina la asignación de recursos con el control del hardware, que directamente no es fácil de usar y requiere una interfaz más específica como un software de aplicación.

Asimismo mantiene ocultos los procesos de administración mostrando al usuario una interfaz simple sin que le demande intervenir en cada tarea.

Sin embargo el SO no realiza por sí mismo todas las tareas, frecuentemente debe ceder el control al procesador orientándolo a que recursos usar y a que programas enfocarse temporalmente.

Las funciones del SO que son más utilizadas reciben el nombre de kernel o núcleo y se cargan en la memoria principal junto con las funciones empleadas en el momento, otros programas y datos del usuario.

De forma concreta un SO proporciona servicios para las siguientes áreas:

- Desarrollo de programas con utilidades del SO como herramientas de desarrollo y editores.
- Ejecución de programas.
- Acceso a dispositivos de entrada / salida.
- Acceso controlado a archivos mediante la identificación de dispositivos, estructura y permisos.
- Acceso al sistema con limitaciones por usuarios a recursos y datos específicos.
- Detección y respuesta a errores producidos por fallas de hardware o durante la ejecución de programas.
- Contabilidad y estadísticas para evaluar el rendimiento del equipo.

Los SO han evolucionado adaptándose a través del tiempo a los avances tecnológicos tanto en hardware donde todo se hace más compacto, simple y a un menor costo, como en software que es cada vez más potente robusto y demandante de recursos.

En el proceso de desarrollo de los SO surgieron algunos conceptos que marcaron las diferencias en sus versiones: administración de tareas, administración de usuarios y manejo de recursos.

Una tarea o proceso puede ser ejecutado únicamente de inicio a fin en un momento dado y hasta que finalice se ejecuta otro (monotarea), o varios que usan los recursos de forma alternada tan rápido que dan la apariencia de ocurrir al mismo tiempo (multitarea).

Un SO doméstico incluido en una PC sólo permite que un usuario a la vez inicie sesión y haga uso del sistema (monousuario), mientras que uno con orientado a servidores permite que varios usuarios ejecuten programas y empleen los recursos de forma simultánea pero garantizando la confidencialidad de la información y la actividad de cada uno (multiusuario).

Por último se dice que un SO es centralizado sí sólo utiliza los recursos del equipo sobre el que corre o distribuido si puede disponer de los dispositivos de otras computadoras conectadas en red.

El crecimiento de Internet ha impulsado el desarrollo de sistemas modernos que incluyen navegadores Web y aplicaciones para redes, comunicaciones y programación.

Dejando de lado los SO con fines domésticos de Windows y sus variantes para servidor, se observa que Sun Microsystems también ofrece soluciones de este tipo como Solaris.

Solaris es un sistema operativo de tipo UNIX que se basa SunOS y en System V Release 4 (SVR4) que funciona en arquitecturas SPARC, x86 (Intel y AMD), servidores y estaciones de trabajo.

Características:

- Procesamiento en tiempo real.
- Estructuras de datos dinámicamente asignadas.

- Gestión de memoria virtual.
- Sistema de archivos virtual.
- Núcleo multihilo.
- Soporte para multiprocesamiento simétrico (SMP) para arquitecturas con múltiples procesadores, éstos comparten los recursos y realizan las mismas funciones aumentando el rendimiento.
- Interfaz orientada a objetos para el sistema de archivos.

1.3.5 Arquitectura Cliente – Servidor

La arquitectura de red refiere al comportamiento que toman las computadoras conectadas a una red.

Originalmente la arquitectura predominante era la de un mainframe con una computadora central de gran tamaño y una serie de dispositivos de almacenamiento, periféricos y terminales conectados externamente.

Llevaba el nombre de **maestro/esclavo** porque las terminales o esclavos dependían totalmente del equipo maestro para realizar su trabajo con la desventaja de que cualquier fallo en éste repercutía en las terminales. Además el proceso para compartir información entre sistemas era muy complicado.

Posteriormente la arquitectura evolucionó dando paso a las redes de igual a igual y la arquitectura cliente servidor buscando resolver éstos problemas.

Redes de igual a igual

Todos los nodos o dispositivos tienen el mismo nivel de responsabilidad con la restricción de poseer también la misma capacidad de hardware.

Otras características son la comunicación bidireccional donde al ser nodos iguales las peticiones pueden hacerse por cualquiera.

Asimismo, un sistema puede prescindir de otro para funcionar con normalidad.

Arquitectura cliente servidor

Comparte un enfoque similar con la programación modular al considerar la división en módulos de un sistema para lograr rapidez, eficiencia y facilidad de mantenimiento.

Llevado a la arquitectura el módulo que genera las llamadas al sistema es el cliente mientras que el módulo que responde es el servidor, donde cada módulo corre sobre una plataforma de hardware diferente apropiada con la función que desempeña.

En este modelo se distinguen 2 procesos, uno para cada módulo que se comunican entre sí mediante protocolos estándar o particulares.

Cliente

El término cliente tiene a la vez dos acepciones, por un lado denota al nodo físico representado por el hardware, por otro hace referencia a la aplicación o software que corre sobre el nodo.

La aplicación cliente inicia diálogos con la aplicación servidor para solicitarle realice alguna tarea, suele contener la interfaz con el usuario, se encarga de validar datos ingresados y gestionar los recursos del hardware.

Servidor

Presenta 2 acepciones al igual que el cliente: nodo (hardware) y aplicación (software). La aplicación servidor ejecuta las tareas solicitadas por uno o varios clientes de forma simultánea, así como capturar y manipular la información de a base de datos.

Características de la arquitectura cliente servidor

- El cliente proporciona la interfaz y el servidor los medios para hacer uso de los recursos compartidos.
- Los procesos cliente y servidor tienen diferentes requerimientos de hardware.
- Demanda una conexión estable para funcionar bajo diferentes plataformas de hardware y software.
- Es escalable, tanto horizontalmente dado que acepta más nodos cliente, y en vertical con la posibilidad de cambiar a servidores más rápidos y potentes.

- Una aplicación cliente servidor se compone de 3 elementos:
  1. *Presentación.* Interfaz de usuario de tipo gráfica o un navegador Web.
  2. *Lógica de negocio.* Describe el comportamiento del sistema, los cálculos y procesos sobre los datos.
  3. *Gestión de datos.* Representada por el manejador de bases de datos que proporciona los medios para acceso y almacenamiento de información.

Se distinguen dos tipos de arquitectura cliente servidor: de dos y tres capas.

Modelo de 2 capas

El cliente se comunica directamente con el servidor por lo que es recomendable implementarse en ambientes con pocos clientes o de lo contrario puede provocar la saturación del servidor.

En el ámbito de las bases de datos se observa por ejemplo en un manejador de Microsoft Access que reside en el servidor mientras que una sencilla aplicación en Visual Basic en el cliente extrae datos de la base y se los presenta al usuario.

En caso de emplear JDBC, el driver específico para conectarse con el manejador debe instalarse del lado del cliente.

Visto de otra forma, los componentes de la arquitectura pueden agruparse de dos formas: la presentación en el cliente, la gestión de datos en el servidor y ambos compartir la lógica de negocio; la segunda la presentación y la lógica en el cliente y únicamente la persistencia de los datos en el servidor como se observa en la figura 1.15.

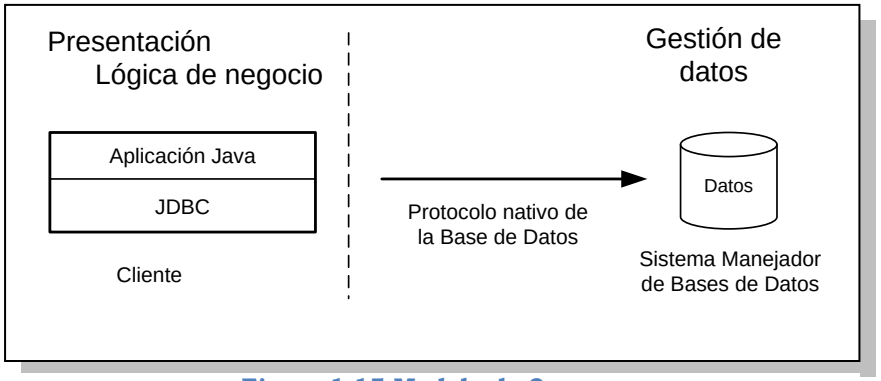

Figura 1.15 Modelo de 2 capas

Modelo de 3 capas

Introduce un elemento conocido como agente entre el cliente y el servidor cuyos objetivos son:

- Liberar de tareas al servidor.
- Traducir las peticiones de los clientes a lenguaje interno del servidor.
- Limitar el número de clientes que acceden.
- Redirigir peticiones a otros servidores.
- Recoger resultados.
- Enviar respuestas a los clientes.

La lógica de negocio toma mayor importancia y puede residir ahora en un servidor de aplicaciones, dejando la presentación en el cliente y la capa de gestión en un servidor de datos como se observa en la figura 1.16. Por ejemplo al usar drivers JDBC éstos ya no tienen que instalarse en la máquina cliente que únicamente contendrá la interfaz con el usuario. Las aplicaciones de tres capas presentan la ventaja de un mayor potencial de crecimiento y un sencillo mantenimiento.

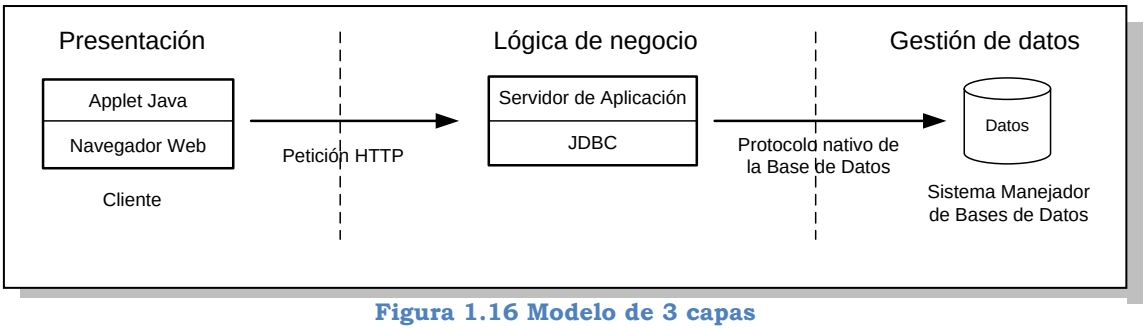

Figura 1.16 Modelo de 3 capas