3 ESTUDIO DE TECNOLOGÍAS EXISTENTES PARA EL DESARROLLO DE APLICACIONES EMPRESARIALES CON MOVILIDAD

Para poder elegir una tecnología adecuada a nuestros propósitos, tendremos que definir los requisitos que las aplicaciones empresariales imponen en el mercado actual. Buscaremos soluciones para abordar estos requisitos y nos centraremos en el mercado de la movilidad, sus exigencias y soluciones.

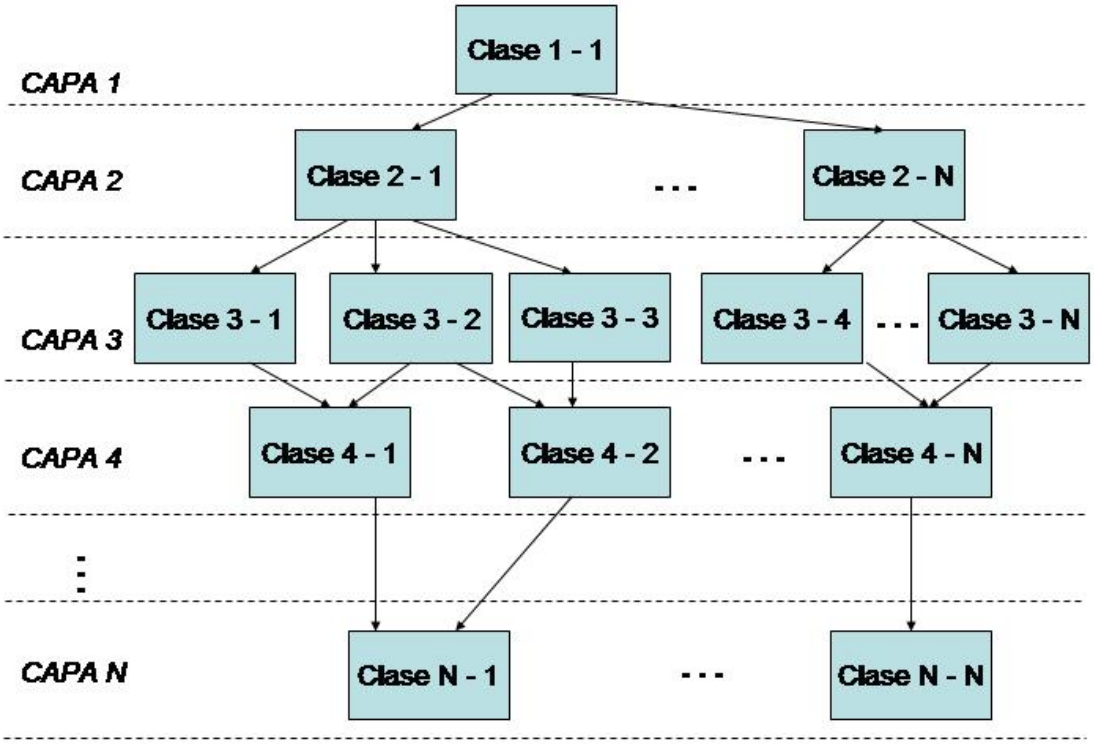
3.1 DESARROLLO DE APLICACIONES EMPRESARIALES

A lo largo de los años se han proporcionando herramientas y tecnologías orientadas al desarrollo de sistemas distribuidos, sin embargo, han sido optimizadas para intranets y escenarios de uso LAN. En los últimos años, han aparecido nuevas tecnologías adaptadas a nuevas necesidades, que nos permiten crear aplicaciones distribuidas utilizando Internet como infraestructura de comunicaciones consiguiendo una más rápida y flexible integración de procesos de negocios entre compañías de socios.

Para diseñar una aplicación empresarial siempre ha sido importante considerar los factores de mantenibilidad y reusabilidad. Hoy en día, las aplicaciones son cada vez más complejas, y conforme aumenta la complejidad, la mantenibilidad y reusabilidad en una aplicación adquieren más valor.

Para conseguir una aplicación fácil de mantener y con componentes reusables, es interesante tener en cuenta el patrón arquitectónico *Layers*. Un patrón arquitectónico es un patrón de alto nivel que fija la arquitectura global de la aplicación.

En el patrón arquitectónico Layers el software está estructurado en capas. Permite ocultar las tecnologías que utiliza nuestro software. Cambios en una capa, no tienen porqué afectar a capas superiores o inferiores, por ejemplo, si existe un cambio de versión en una de las capas, no siempre supone un impacto sobre capas superiores.



F. 3-1: Patrón Arquitectónico Layers.

Con este patrón, el software está divido en capas.

El tiempo siempre es un factor a tener en cuenta. Se proyectan sistemas en varios canales de clientes y se debe hacer de un modo fiable, productivo y capaz de hacer frente a frecuentes actualizaciones tanto de información como servicios. Es necesario mantenerse al ritmo de los retos de negocio y mantener el valor de los recursos existentes. En este entorno productividad, seguridad y predictibilidad se convierten en los mayores aliados.

Las aplicaciones deben pasar rápidamente del prototipo a la producción y deben continuar evolucionando incluso después de ser desplegadas, es por tanto que deben ser escalables. Sin un único modelo la arquitectura de las aplicaciones se vuelve compleja, el poseer medios estándares para acceder a los servicios requeridos por las aplicaciones puede contribuir tanto a una rápida respuesta como a la productividad.

La integración de distintos sistemas y procesos es otro de los retos ante los que se enfrentan las aplicaciones empresariales. Una de las dificultades de integrar los distintos sistemas, es proveer un único modelo de seguridad.

En una aplicación empresarial, el conjunto de reglas específicas de negocio determinan las necesidades de la aplicación. Los requerimientos comunes para los distintos objetos de negocio suelen ser el mantenimiento de estado, operaciones con datos compartidos, participación en transacciones, necesidad de una diversidad de clientes, disponibilidad, acceso remoto a datos, control de acceso y reusabilidad.

En resumen, para conseguir una potente aplicación proponemos las siguientes soluciones para la consecución de distintos objetivos:

— Los **sistemas distribuidos** nos proporcionarían una mayor integración de procesos de negocios.

— El **patrón arquitectónico Layer**, en la estructura del software, nos facilitaría el desarrollo de aplicaciones con buenos resultados en cuanto a escalabilidad, mantenibilidad y reusabilidad.

— La utilización de medios y **métodos estándares** nos permitirían conseguir una rápida respuesta, mayor productividad, más seguridad y mejor predictibilidad.

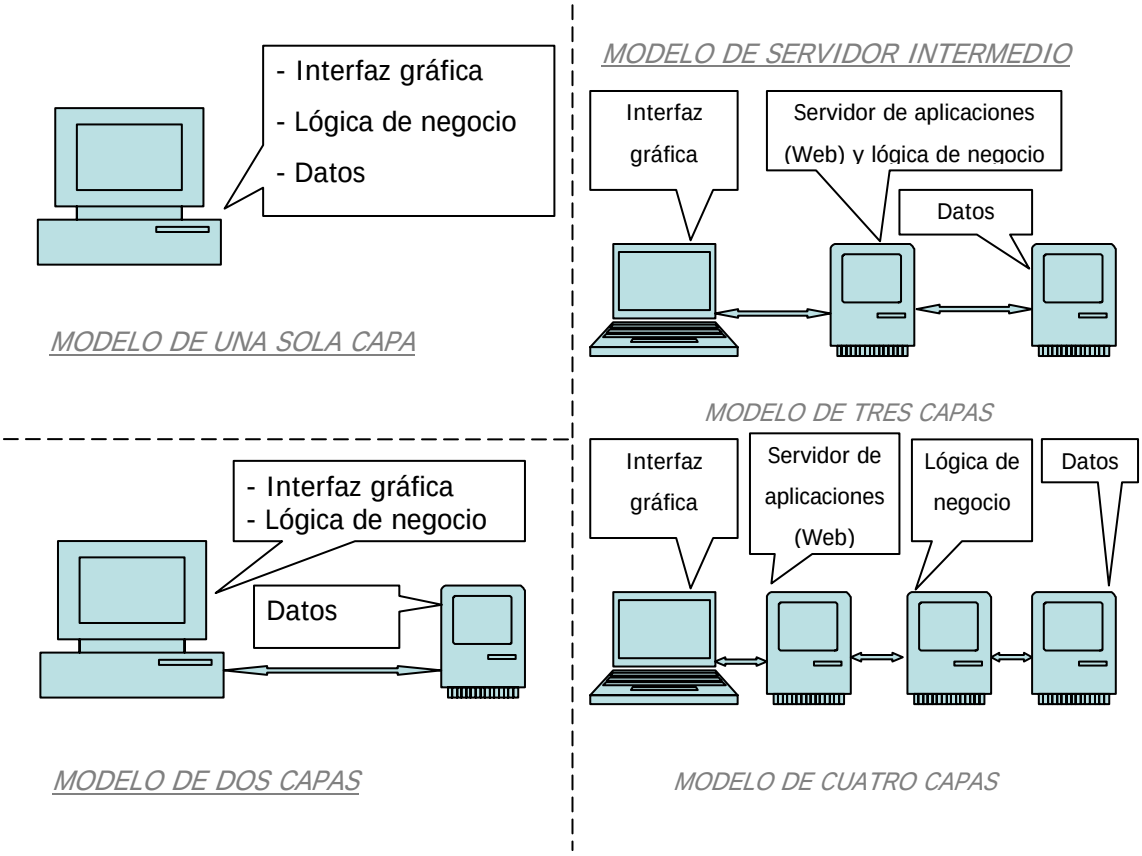
3.2 TECNOLOGÍAS WEB COMO PROPUESTA A LA MOVILIDAD

La arquitectura física es necesaria tenerla en cuenta en el diseño de nuestras aplicaciones. Una arquitectura física con una sola capa, en la que el acceso a datos, la lógica de negocio y la interfaz gráfica están integrados, no nos proporciona ningún tipo de movilidad. Sería necesario desplazar toda la aplicación para poder utilizarla en otra máquina y aún así, la información sería difícilmente compartida. Por esto no centraremos en las **arquitecturas multicapa**.

En el modelo de dos capas con clientes stand-alone, tanto el modelo como la interfaz gráfica se encuentran en los clientes, y la intranet se utiliza para el acceso a datos. Sin embargo, esta arquitectura también presenta varias desventajas. Cambios en la implementación de la capa modelo provocan una necesidad de recompilación de toda la aplicación y reinstalación en los clientes. Aún cuando estos cambios pueden estar provocados por un cambio en el tipo de la base de datos, la necesidad de cambios de drivers de acceso provoca cambios en la lógica del modelo. El modelo de dos capas, además, no nos proporciona toda la movilidad deseada, el cliente necesita tener instalada la aplicación para poder acceder a los datos.

Una mejor solución es la que ofrece la **arquitectura de modelo de servidor intermedio**. En esta arquitectura nos encontramos modelos de tres o cuatro capas. En el modelo de tres capas, un cambio en la implementación del modelo sólo afecta al servidor. Si existen clientes stand-alone éstos sólo dispondrán de la interfaz gráfica, accederán al servidor que implementa el modelo, y éste es el que accede a los datos. Entre las arquitecturas de tres capas están aquellas aplicaciones puramente Web. En estas aplicaciones los clientes sólo poseen un navegador desde el que acceden al servidor.

Por último otra posibilidad es la del modelo de cuatro capas en las que existen clientes tanto Web como stand-alone y dos tipos de servidores, el primero un servidor de aplicaciones Web que accede al siguiente servidor donde se encuentra el modelo. Este último es el que accede a los datos. Al estar separado el servidor de aplicaciones Web y el del modelo, los clientes stand-alone pueden acceder directamente al servidor donde se implementa la lógica de negocio.



F. 3-2: Arquitectura multicapa.

La arquitectura más completa y que permite mucha más flexibilidad en la configuración es el modelo de cuatro capas, sin embargo, este modelo en muchos casos excede las necesidades de nuestras aplicaciones. Nos centraremos en un modelo de tres capas con clientes Web. Este modelo, con unas características de diseño adecuadas es fácilmente migrable a una estructura de cuatro capas.

Las aplicaciones Web residen en el servidor y los clientes pueden acceder a la aplicación en el servidor por algún medio. El cliente más común es un navegador, que muestra la interfaz gráfica y se comunica con el servidor normalmente por peticiones HTTP a través de Internet.

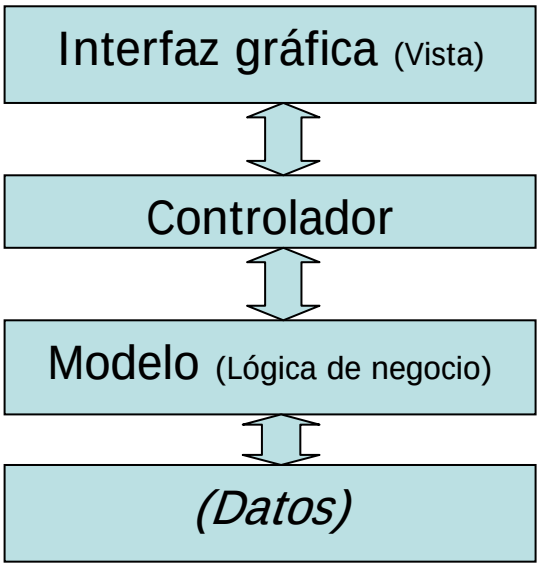
La movilidad implica la disponibilidad en cualquier momento y lugar de la información. Las tecnologías Web, al situar las aplicaciones en el servidor, y poder acceder a ellas desde los clientes, nos proporciona esta disponibilidad total, siempre y cuando, dispongamos del cliente adecuado. Por ejemplo, si normalmente accedemos desde un navegador a través de Internet, lo podremos hacer desde cualquier navegador, y por tanto desde cualquier PC, no restringiéndonos a un solo ordenador, lo que como mínimo nos proporciona la movilidad entre todos los ordenadores con acceso a Internet.

Los primeros sitios Web consistían tan sólo en colecciones de páginas estáticas enlazadas. Hoy día, incluyen multimedia, existen aplicaciones de comercio electrónico y banca online, aplicaciones basadas en Voz IP, etc. Estos avances en contenido, se han visto apoyados por los importantes avances en las tecnologías de servidor y en tecnologías de creación dinámica de contenido.

Las tecnologías Web no incluyen tan sólo el modelo de servidor al que acceden los navegadores a través de peticiones HTTP. En el diseño de la aplicación intentaremos separar lo que es la interfaz gráfica de la lógica del modelo de negocio que debe ser reusable con distintas interfaces gráficas. Con esto pretendemos conseguir mayor movilidad ya que el modelo no restringiría el cliente a utilizar. Por ejemplo, para un mismo modelo podríamos tener clientes con interfaces Web que accedan desde Internet o desde una Intranet, clientes stand-alone que accedan desde la Intranet con un interfaz gráfico de ventanas, clientes móviles que accedan desde WAP, SMS o a través de Internet con UMTS. En este último caso hay que tener en cuenta que el interfaz del dispositivo móvil con una pantalla pequeña tiene menor capacidad que la interfaz que podremos ver desde la pantalla de un PC.

Para conseguir nuestro objetivo de diversidad de clientes es interesante tener en cuenta el **patrón arquitectónico MVC, Model - View - Controller**. El patrón arquitectónico MVC propone una separación clara entre el modelo y la vista, gracias a un controlador que los mantiene desacoplados. Este modelo es reusable con distintas vistas y nos facilita la utilización de diferentes clientes. La arquitectura Layers, de estructuración del software en capas, dará soporte a la arquitectura MVC. A su vez, el patrón MVC facilitará la separación física de componentes en una arquitectura multicapa.

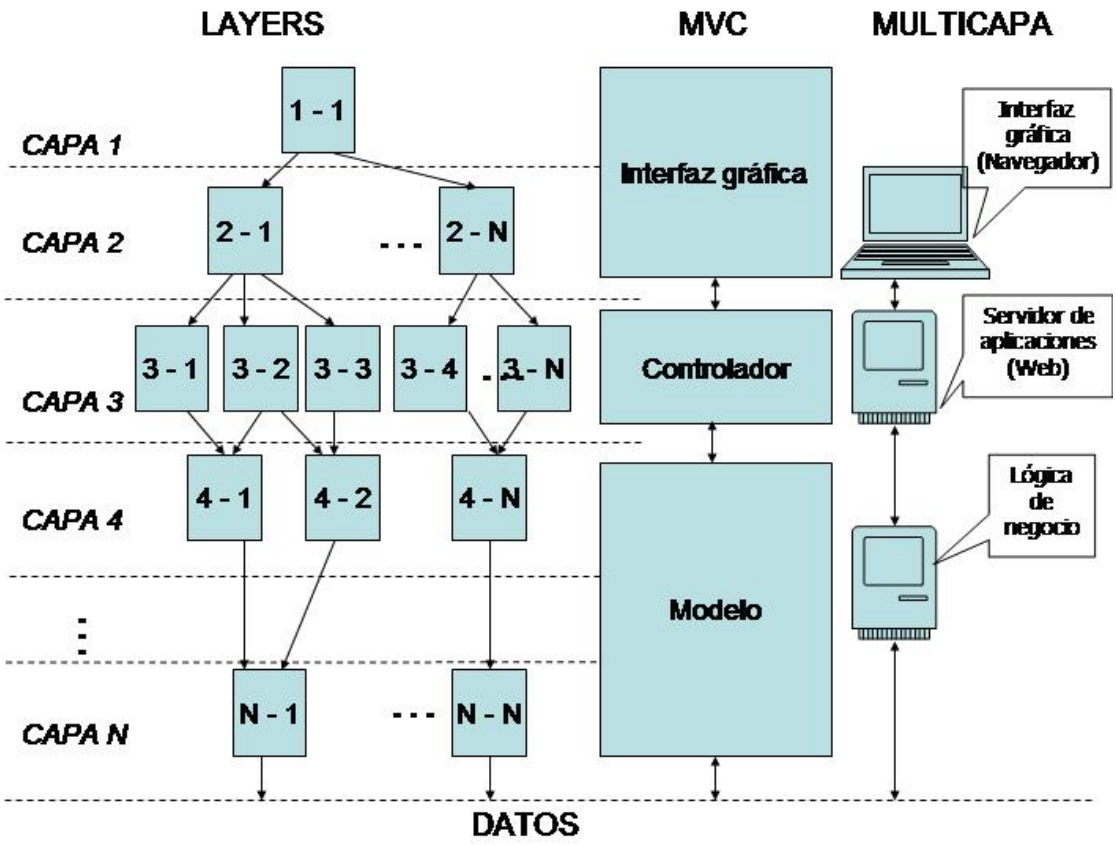
No todas las tecnologías Web utilizan el patrón MVC, en el ejemplo de un navegador que accede al servidor a través de Internet, puede utilizar o no esta arquitectura. Esto vendrá determinado por la separación o no del modelo de negocio y la generación de la vista. Puede que el código que implementa los casos de uso esté inmerso en la generación de la interfaz gráfica, en este caso, no se implementa el modelo MVC, y el uso de distintos tipos de clientes sería complejo y poco reusable.



F. 3-3: Patrón Model-View-Controller (MVC).

Es necesario insistir que este modelo de arquitectura tipo Web no es útil sólo en entornos Internet, sino para todos los entornos empresariales. La utilización de este modelo, nos permitirá, además de una mayor diversidad de clientes, utilizar tanto una arquitectura física de tres o de cuatro capas. Al separar la interfaz gráfica de la lógica de negocio, es fácil separar el modelo (lógica de negocio) en un servidor distinto al que genera la vista.

Por lo tanto, para conseguir mayor movilidad de nuestras aplicaciones utilizaremos tecnologías Web con las que se propone una arquitectura física de servidor intermedio, y utilizaremos el patrón arquitectónico Model-View-Controller.



F. 3-4: Relación entre patrones arquitectónicos.

NOTA: Relación de patrones arquitectónicos

Hasta ahora se ha propuesto la utilización de tres patrones arquitectónicos. El patrón arquitectónico multicapa, hace referencia a la distribución de las distintas partes de la aplicación en uno o distintos componentes físicos. Este patrón puede soportarse sobre el patrón MVC. Al dividir el software en modelo, vista y controlador, la distribución de cada una de las partes en distintas máquinas se realiza de forma inmediata. Sin embargo, la estructura MVC, no es requisito indispensable para las arquitecturas multicapa. Por ejemplo, en un modelo de dos capas, la vista y el modelo pueden estar mezclados, al situarse en la misma máquina.

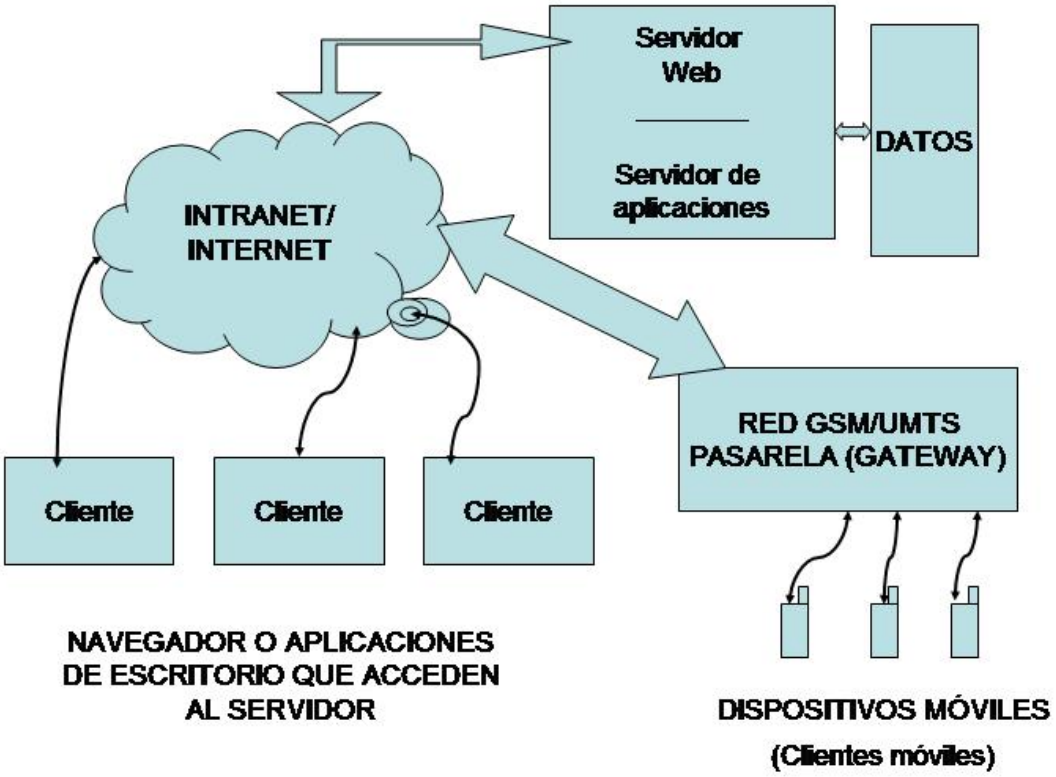
A su vez, el uso del patrón Layers para la división del software es un gran soporte para el uso del patrón MVC. Con una correcta división del software en capas, los componentes de la vista serán fácilmente identificados y diferenciados de los correspondientes al modelo y al controlador. También se podría considerar el patrón MVC, como una aplicación del patrón arquitectónica Layers con tres capas principales. Además, cada una de estas capas podría seguir el patrón de división del software Layers.

3.3 TECNOLOGÍAS

A continuación se muestra un resumen de las tecnologías para el desarrollo de aplicaciones empresariales más importantes en el mercado actual.

Se incluyen las tecnologías para el desarrollo Web, ya que son las que consideramos más adecuadas para el desarrollo de aplicaciones con movilidad.

Con este proyecto no se pretende abarcar la integración de aplicaciones, sin embargo, desde un principio se ha querido resaltar la importancia de la integración de éstas en el mercado actual. Por esto, también incluimos entre las tecnologías actuales, algunas tecnologías de integración, para que se tengan presente las distintas posibilidades que hoy en día nos permiten afrontar este reto.



F. 3-5: Tecnologías Web como propuesta para la movilidad.

3.3.1 Tecnologías tradicionales

3.3.1.1 HTML estático

Las primeras páginas Web dependían de servidores HTTP básicos que, a través de peticiones, se limitaban a servir páginas HTML estáticas a los navegadores. Las páginas HTML están formadas por etiquetas que definen formato y contenido fácilmente interpretado por los navegadores.

3.3.1.2 Extensiones de servidor

Sin embargo, rápidamente empezó a demandarse una plataforma Web para servir aplicaciones y contenidos dinámicos.

Se desarrollaron varios mecanismos para permitir a los servidores Web generar contenido bajo demanda, a través de extensiones funcionales de los servidores Web.

Las API de extensión de servidor permiten a los desarrolladores crear librerías que generan contenido dinámico. Sin embargo estas API son propias de cada servidor, no siendo portadles entre servidores de distintos distribuidores.

Uno de los problemas que presentan estas extensiones de servidor, es que pueden comprometer la estabilidad de un sistema, ya que si fallan pueden hacer caer el servidor.

3.3.1.3 Common Gateway Interface, CGI.

El primer estándar de extensión de servidor fue la Common Gateway Interface (CGI) que define un tipo de programa ejecutable usado por un servidor para producir contenido dinámico. CGI engloba el conjunto de normas que definen la comunicación entre los servidores HTTP y programas ejecutados en el servidor. Estos programas son los que procesan la información que envían al cliente Web y generan la vista de forma dinámica.

Aún son muy populares, pero tienen algunas limitaciones importantes. El mismo programa genera tanto las partes estáticas como las partes dinámicas. Cada petición HTTP es enviada al CGI lo que suele resultar en la creación de procesos pesados para el sistema operativo donde corre el servidor. No ofrece soporte portable para servicios de sistemas de alto nivel como balance de carga, escalabilidad, disponibilidad, seguridad, mantenimiento de estado y manejo de recursos.

Los CGI son fáciles de entender, pero las soluciones escalables son difíciles de desarrollar y mantener. Actualmente, los lenguajes más utilizados para aplicaciones CGI son Perl y C.

TECNOLOGÍA	CARACTERÍSTICAS
HTML estático	*-Contenido estático*
Exten. de servidor	*-Contenido dinámico* *-Mezcla de contenido estático y dinámico, difícil de mantener, poco escalable, propias de cada servidor, pueden provocar comportamientos inestables.*
CGI	*-Contenido dinámico, estándar.* *-Mezcla de contenido estático y dinámico, difícil de mantener, poco escalable,procesos pesados en el servidor.*

F. 3-6: Comparación de tecnologías Web - I.

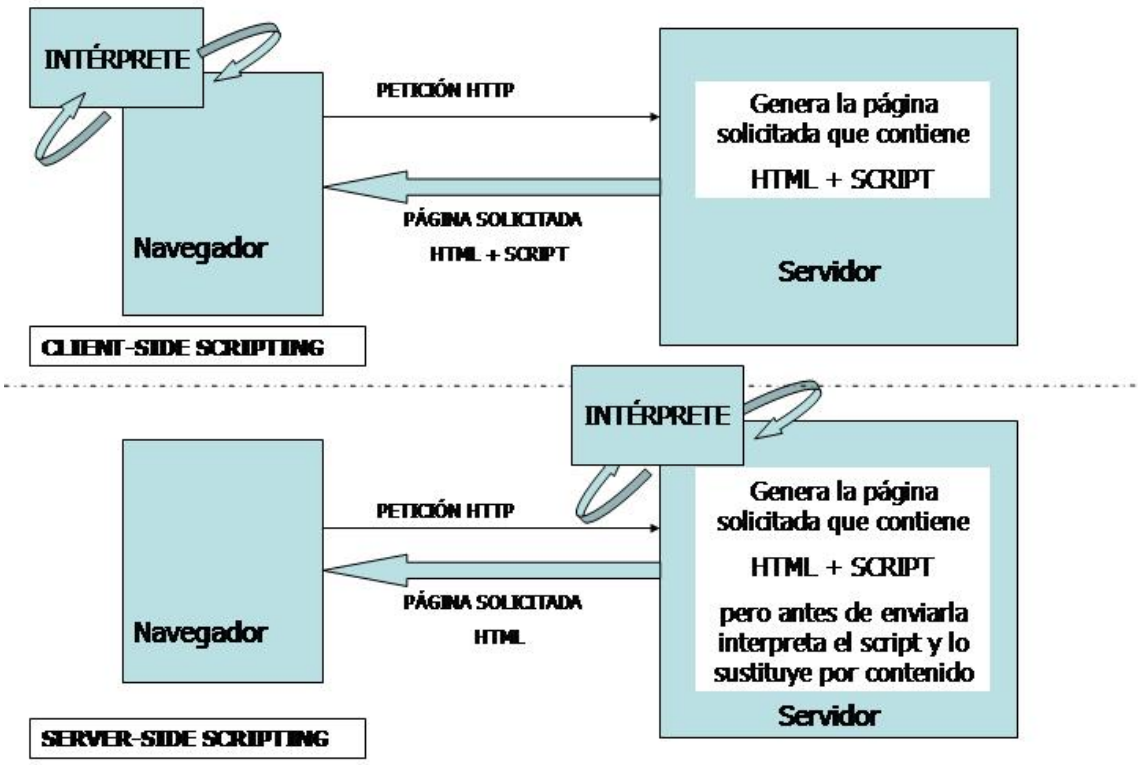
3.3.1.4 Scripting

El uso de scripting en las páginas Web, se refiere a la introducción de código de algún lenguaje interpretado entre las etiquetas HTML. Los lenguajes interpretados, son aquellos lenguajes de programación que no necesitan compilarse, ya que son ejecutados por un intérprete. Los compiladores traducen los programas escritos en lenguajes de alto nivel a otro lenguaje capaz de ejecutarse en la máquina. El intérprete lee directamente las líneas de código y las ejecuta.

Podemos diferenciar dos tipos de scripting según dónde se ejecute el script. Si el script se ejecuta en el lado cliente recibe el nombre de *client-side scripting*, si lo hace en el servidor se trata de *server-side scripting*.

En la figura F. 3-7, se muestran los casos de scripting orientado a Web más comunes, en el que las páginas Web contienen etiquetas HTML y scripts. En el caso de client-side scripting, el navegador recibe una página con etiquetas HTML y código inmerso que interpreta. En el caso de server-side scripting, al navegador sólo le llega una página con etiquetas HTML ya que el servidor interpreta los scripts sustituyéndolos por contenido dinámicamente.

El server-side scripting se puede considerar una mejora de las extensiones de servidor, donde código que se ejecuta en el servidor produce contenido dinámico. Si el código falla, normalmente no hace caer el servidor ya que el intérprete de código puede fácilmente actuar para proteger al servidor de los fallos de código.



F. 3-7: Ejemplos de scripting orientado a Web.

Este tipo de tecnología por si sola, aún no proporciona acceso portable y uniforme a servicios de sistemas de alto nivel.

A continuación veremos algunos de los lenguajes de script más utilizados en entornos Web.

3.3.1.4.a JavaScript

Es un lenguaje interpretado orientado a Web para ejecutar en el cliente. Este lenguaje tiene una sintaxis semejante al lenguaje de programación Java™ y fue desarrollado por Netscape para utilización en su navegador aunque inicialmente tuvo otros nombres como Mocha y LiveScript.

Este lenguaje fue adoptado como estándar ECMA con el nombre de ECMAScript y actualmente también es un estándar ISO.

Actualmente, este lenguaje ha vuelto a adquirir importancia debido a su utilización en la realización de aplicaciones Web con AJAX.

3.3.1.4.b JScript

JScript es la implementación por parte de Microsoft de ECMAScript. Es por tanto, muy similar a JavaScript, sin embargo, a menudo son incompatibles. Actualmente, para hacer frentes a estas incompatibilidades el World Wide Web Consortium ha diseñado el estándar Document Object Model, DOM (Modelo de Objetos del Documento).

3.3.1.4.c VBScript

Es otro lenguaje de script orientado a Web propuesto por Microsoft para Internet Explorer. Al igual que los anteriores ejecuta los scripts en el cliente, aunque a diferencia de los anteriores, su sintaxis es similar a Visual Basic.

3.3.1.4.d Active Server Pages, ASP

Esta es la solución server-side scripting que proporcione Microsoft, páginas activas de servidor, para hacer más fácil crear contenido dinámico generando las páginas en el servidor. Inicialmente sólo se podía utilizar con los servidores de Microsoft (Microsoft IIS o PWS), aunque

ya otros servidores, no propios de Microsoft permiten su utilización. No es independiente de la plataforma.

Utiliza etiquetas para permitir código embebido en una página HTML, seguimiento de sesión y conexión a bases de datos. Las etiquetas están escritas en VBScript y usan componentes ActiveX.

3.3.1.4.e PHP: Hypertext Preprocessor

Es una solución de código abierto del tipo server-side scripting. Este lenguaje de alto nivel es un lenguaje de servidor especialmente pensado para desarrollos Web y el cual puede ser embebido en páginas HTML.

La mayoría de su sintaxis es similar a C, Java y Perl. Este lenguaje es fácil de aprender y permite a los desarrolladores de páginas Web la creación de páginas dinámicas de una manera rápida y fácil. Aunque el uso tradicional de PHP es el desarrollo Web, tiene también otras funcionalidades como el desarrollo de aplicaciones de escritorio.

Este lenguaje se utiliza principalmente integrado en las plataformas tipo LAMP.

3.3.1.5 Servlets

Un *servlet* es un componente Web basado en tecnología Java, manejado por un contenedor que genera contenido dinámico. Básicamente es una clase Java que extiende un servidor Web y produce contenido dinámico en respuesta a las peticiones de servicio de una o más URL.

Son clases Java compiladas, por lo que generalmente son más rápidos que los programas CGI y los scripts de servidor. Son más seguro que las extensiones de servidor y proporcionan mayor variedad de servicios estándares que cualquiera de las anteriores tecnologías.

Al igual que con los CGI, en el código tenemos que incluir tanto el modelo de negocio, como las partes estáticas y dinámicas de la presentación final. Pero los servlets implican menos sobrecarga. Los CGI inician un proceso completo lo que supones hilos pesados, mientras que cada petición a un servlet crea un hilo ligero, un proceso hijo, que es controlado por el proceso

padre, que en este caso es el contenedor de servlets. Esto también mejora el rendimiento respecto a los CGI en el cambio de contexto, que se hace más fácil ya que esto hilos ligeros pasan más fácilmente de activos a inactivos o espera.

TECNOLOGÍA	CARACTERÍSTICAS
Client-side scripting (JavaScript, JScript, VBScript)	*-Contenido dinámico* *-Mezcla de vista y contenido.* *-El cliente necesita un intérprete, normalmente incorporado en el navegador, del lenguaje script. No todas las versiones de scripting son compatibles.*
Server-side scripting (ASP, PHP)	*-Contenido dinámico.* *-Mezcla de vista y contenido.* *-Suele utilizarse integrado en alguna plataforma.*
Servlets	*-Contenido dinámico, estandarizado.* *-Son clases Java, similar a los CGI pero procesos menos pesados.*

F. 3-8: Comparación de tecnologías Web - II.

3.3.1.6 JavaServer Pages

La tecnología *JavaServer Pages* (JSP) es una extensión de la tecnología de Servlets creada para facilitar el uso combinado de datos estáticos y dinámicos en páginas HTML.

Difieren de los servlets en su modelo de programación. Una página JSP es muy similar a una página estática HTML, ya que es esencialmente un documento, aunque este a diferencia de las páginas puramente HTML, especifica contenido dinámico.

Una página JSP es una página Web con etiquetas especiales que puede contener código incrustado, mientras un servlet propiamente dicho es un programa que recibe peticiones y puede generar a partir de ellas una página Web.

JSP resuelve problemas que no resuelven los servlets. Por ejemplo, permite utilizar herramientas de diseño de páginas Web, las actualizaciones de aspecto gráfico no provocan un rearranque del servidor y permiten separación de roles en el equipo de desarrollo.

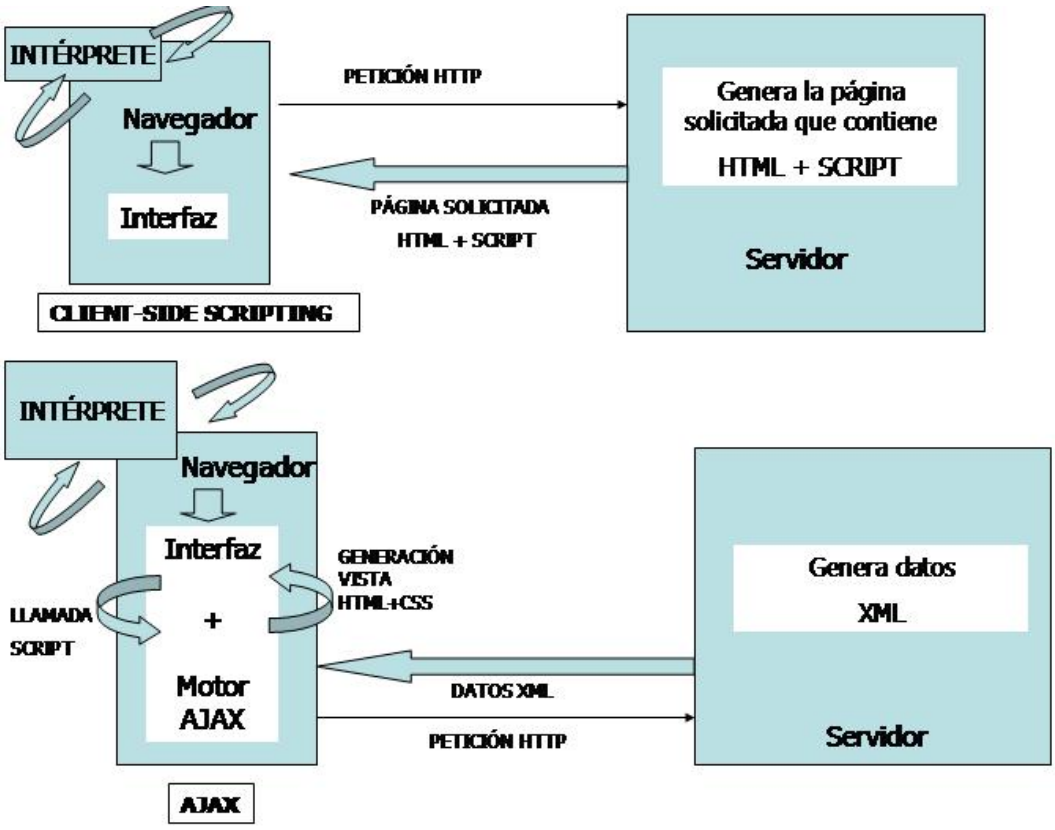
La tecnología JSP proporciona una alternativa centrada en el documento frente a la alternativa centrada en la programación que ofrecen los servlets. Los servlets son programas que producen contenido dinámico, las páginas JSP especifican contenido dinámico.

Representan una solución que se puede combinar fácilmente con algunas de las soluciones anteriores, creando contenido dinámico de un modo más fácil y rápido consiguiendo aplicaciones que trabajan con una variedad de servidores Web, navegadores, servidores de aplicación y otras herramientas de desarrollo.

Las páginas JSP pueden tener similitudes con otras tecnologías de servidor como las páginas ASP. Ambos utilizan un lenguaje de etiquetas que utilizan para funciones como seguimiento de sesión y conexiones a bases de datos. Además en ambas tecnologías se permite código embebido en las páginas. Sin embargo, JSP presenta mejores características en velocidad y estabilidad ya que una página ASP es siempre interpretada. Se diferencian también en el lenguaje que utilizan, mientras ASP utiliza VBScript, JSP utiliza Java, por lo que es mucho más independiente de la plataforma. Por último, otra de las diferencias básicas, la marca el lenguaje de etiquetas extensible propio de JSP. Esta característica, que no poseen otras tecnologías, permite a los desarrolladores crear etiquetas personalizadas para extender la sintaxis de las páginas JSP creando así nuevas funciones.

3.3.2 AJAX

AJAX es el acrónimo de Asyncronous JavaScript And XML y de Advanced JavaScript And XML. Se puede considerar como una tecnología que engloba otras tecnologías que trabajan conjuntamente. Utiliza, entre otras, las tecnologías X/HTML y CSS para la presentación de datos, DOM y JavaScript para interactuar con los datos asíncronos, XML y XSLT para el intercambio y manipulación de datos.



F. 3-9: Comparación de AJAX y client-side scripting.

Su objetivo es conseguir una mayor interacción por parte del cliente, en las aplicaciones Web. Esto se consigue con la carga inicial de una página en la que se mantiene el usuario, mientras scripts en JavaScript, trabajan en modo background, buscando en el servidor los datos necesarios para actualizar la página. Aunque la carga inicial de la página se hace más lenta por la carga inicial de los scripts, la interacción con la página es mucho más rápida, ya que sólo se vuelven a cargar las partes de la página necesaria, evitando tener que cargar aquellas partes que no se han modificado.

Durante la carga inicial de la página, se carga un motor AJAX en JavaScript. Este motor es el que permite que la interacción con la aplicación se realice de forma asíncrona. La idea es no generar directamente una petición HTTP por cada acción del usuario, algunas acciones como validación, edición y a veces algo de navegación, pueden ser realizadas por el motor AJAX cargado en el cliente. Cuando es necesario llamar al servidor, el motor lo hace modo asíncrono sin frenar la interacción del usuario.

Esta tecnología acerca las aplicaciones Web a las aplicaciones de escritorio en cuanto a rapidez e interfaz. Exige la existencia de un intérprete de JavaScript en la máquina cliente (normalmente incorporado al navegador).

3.3.3 Plataformas

Junto a las tecnologías tradicionales, hoy en día han surgido plataformas completas que integran algunas de estas tecnologías. Esto permite la utilización de distintas tecnologías para conseguir distintos objetivos pero en un entorno de integración flexible y potente.

3.3.3.1 Plataforma LAMP

Es una solución de código abierto que combina el lenguaje PHP (basado en Java), con un servidor Apache y la base de datos MySQL. Todo ello sobre Linux. Esta plataforma también permite sustituir el uso del lenguaje PHP por los lenguajes Perl o Python.

Existen otras soluciones basadas en esta plataforma. Por ejemplo, podría utilizarse PHP, Apache y MySQL sobre plataformas Windows, o utilizar otro tipo de bases de datos.

Puesto que PHP es una solución similar a ASP, pero basada en Java y no en Visual Basic, la independencia de la plataforma es mayor.

Esta alternativa nos permite acceso a datos y utilizar tecnologías Web. Es menos eficiente que el uso de servlets y JSP, pero tiene la gran ventaja de requerir menos conocimientos técnicos. Para algunas aplicaciones sencillas puede ser la solución más adecuada.

TECNOLOGÍA	CARACTERÍSTICAS
JSP	*-Contenido dinámico, estándarizado.* *-Mezcla de vista y contenido si no se utiliza combinado con otras tecnologías.* *-Similar a los lenguajes server-side scripting aunque más extensible.* *-Suele utilizarse con Servlets.*
AJAX	*-Contenido dinámico, mejora en las interacción e interfaz gráfica.* *-Mezcla de vista y contenido.* *-Tecnologías: XML, JavaScript y otras.*
LAMP	*-Linux, Apache, MySQL y Perl, Python o PHP.*

F. 3-10: Comparación de tecnologías Web - III.

3.3.3.2 Plataforma .NET

Microsoft propone el uso de la plataforma .NET y apuesta por un modelo de computación distribuido basado en Internet. Se trata de una plataforma de software donde pueden interoperar distintos componentes independientemente del lenguaje, esto es, en lugar de escribir componentes para una combinación hardware/sistema operativo, se escribe una solución para .NET.

Con base en los estándares abiertos de Internet, .NET intenta potenciar aplicaciones, servicios y dispositivos, a fin de que trabajen juntos, para permitir el acceso y acción a la información en cualquier momento y lugar desde cualquier dispositivo.

La plataforma soporta las aplicaciones de escritorio, servicios centralizados, aplicaciones de Internet, integración de procesos de negocios, una gran variedad de clientes como dispositivos inalámbricos y teléfonos, y protocolos como 802.11 y WAP.

La meta principal es la integración, para lo que utiliza las tecnologías de XML y SOAP y su gran apuesta son los Servicios Web (*Web Services*).

En realidad, .NET es el nombre asignado a diversos productos como VisualEstudio.NET y Windows.NETServer, y servicios que incluyen Passport y Hailstorm. Con ellos se pretende ofrecer una manera universal de acceder a recursos en Internet. Además la plataforma proporciona software cliente para dispositivos inteligentes que permite a los PC, portátiles, estaciones de trabajo, teléfonos, consolas de juegos y otros dispositivos interoperar en la plataforma.

Las características más importantes de la plataforma .NET son el soporte multilenguaje, un lenguaje intermedio IL, código gestionado y no gestionado, y estándares abiertos como SOAP y el propio lenguaje C#.

La característica de multilenguaje comprende C++, Eiffel, Component Pascal, COBOL y Visual Basic. Un lenguaje está integrado en .NET si puede interoperar con cualquier otro lenguaje en .NET, utilizarse para producir MSIL (Microsoft Intermediate Lenguage, lenguaje intermedio de

Microsoft), y emplear librerías de clases estándares .NET en forma nativa para ese lenguaje. Además se han creado los lenguajes ASP.Net, Perl.Net y Python.Net entre otros.

VisualStudio.NET es la herramienta de desarrollo y permite utilizar cualquier lenguaje para después ser ejecutado en ambientes .NET. COBOL, C++, Visual Basic o cualquier otro de los lenguajes soportados, serán convertidos al lenguaje intermedio MSIL a través de VisualStudio.NET. Este lenguaje intermedio es el que opera .NET, de esta manera, el código, escrito en más de 20 lenguajes podrá interoperar.

Sin embargo, las complejidades del soporte multilenguaje, estriban en el mapeo que debe existir entre los diversos lenguajes y MSIL, lo que a veces se puede hacer difícil la interoperabilidad prometida por la plataforma.

C# es el lenguaje de .NET, pero esta plataforma necesita mantener el lenguaje Visual Basic, surgiendo así el nuevo Visual Basic .NET. C# no forma parte medular de .NET, pero es la interfaz más directa hacia la plataforma. Microsoft ha colocado el futuro de C# en manos del European Standard Comité (ECMA).

La plataforma .NET incluye tecnologías como ADO.NET, ASP.NET y COM+, que son una mejora de las versiones anteriores de ADO, ASP y COM. También incluye API para XML. Las distintas versiones de .NET permiten el desarrollo de Web Services, sin embargo, no son Web Services verdaderos debido a la carencia de apoyo de comunidades de estandarización.

3.3.3.3 Plataforma J2EE

Java 2 Enterprise Edition, J2EE, es un conjunto de especificaciones de API Java para la construcción de aplicaciones empresariales, representa un único estándar para la implementación y despliegue de estas aplicaciones. Dentro del estándar J2EE tenemos varias posibilidades de diseño que se adaptan a distintas expectativas, ambiciones y posibilidades.

Rápidamente estableció un nuevo modelo para el desarrollo de aplicaciones distribuidas basado en componentes que pueden aprovecharse de servicios de una plataforma sofisticada. Estos componentes pueden desarrollarse de acuerdo a unas guías estándares, consiguiendo una gran

cantidad de productos compatibles y reusables para una productividad máxima. Este modelo es en gran medida el responsable del éxito de la plataforma.

A pesar de este nuevo modelo, uno de los puntos fuertes de esta plataforma, es la facilidad de migración o integración con las tecnologías anteriores. La solución J2EE incluye el uso de Servlets y JSP, y permite el uso de frameworks para facilitar el desarrollo.

La plataforma J2EE ha sido diseñada en un proceso abierto, con conversaciones de un gran rango de empresas informáticas para asegurar que cumple el más amplio margen posible de requerimientos de las aplicaciones informáticas. El nivel de abstracción que provee el estándar, permite el desarrollo, aclaración y exploración de temas comunes, y el desarrollo de ciertas guías de diseño comunes.

El objetivo de la plataforma es proveer soporte de cliente y de servidor al desarrollo de aplicaciones multicapa distribuidas. Proporciona un modelo de aplicación multicapa distribuido, por lo que distintas partes de la aplicación pueden ejecutarse en distintas máquinas. La capa cliente soporta variedad de tipos de clientes, las capas intermedias soporta los servicios de clientes y servicios de los componentes de la lógica de negocio. La capa final accede a los sistemas de información empresariales.

El modelo de componentes en el que se basa J2EE hace uso de contenedores. Los contenedores son entornos estandarizados de ejecución que proveen de servicios específicos a los componentes. Los componentes pueden esperar que estos servicios estén disponibles en cualquier plataforma J2EE de cualquier vendedor. Entre estos servicios se pueden encontrar soporte en tiempo de ejecución, retornar resultados, manejo de sesiones, soporte automatizado de transacciones, acceso estandarizado a los sistemas de información y control de ciclo de vida de los componentes.

Mientras que la especificación J2EE define los contenedores de los componentes que la plataforma debe soportar, no especifica o restringe la configuración de los contenedores.

J2EE incluye tecnologías como JDBC, JSP y EJB. Con J2EE pueden desarrollarse y desplegarse Web Services utilizando el API JAXP, aunque no es la manera más eficiente de construirlos ya

que requiere mucha intervención manual. Pueden utilizarse librerías de terceros que en un futuro serán estandarizadas.

Existen múltiples implementaciones de distintos fabricantes, incluidas implementaciones libres. Una aplicación construida con J2EE no depende de una implementación particular.

Los principales beneficios de la plataforma J2EE son los siguientes:

— Independiente de la implementación, libertad de elección de servidores, herramientas y componentes.

— Simplificación de la arquitectura y el desarrollo de aplicaciones con un nuevo modelo para el desarrollo de aplicaciones distribuidas, con componentes compatibles y reusables. Permite por ejemplo una gran diversidad de clientes.

— Fácil integración con sistemas de información existentes.

— Estructura escalable con una plataforma flexible para el desarrollo según necesidades.

— Modelo de seguridad flexible.

TECNOLOGÍA	CARACTERÍSTICAS
.NET	*-Solución propietaria aunque intenta estandarización.* *-Diversidad de clientes, soporte multilenguaje, sistemas distribuidos, utiliza estándares de Internet, Servicios Web.* *-ASP.Net, ADO.Net, COM+, C#.*
J2EE	*-Conjunto de especificaciones, variedad de implementaciones según el estándar.* *-Integración con otras tecnologías, sistemas distribuidos, diversidad de clientes.* *-Servlets, JSP, EJBs.*

F. 3-11: Comparación de tecnologías Web - IV.

3.3.4 Tecnologías de integración

A continuación incluimos algunas de las tecnologías de integración de sistemas más utilizadas.

3.3.4.1 Web Services

Los Web Services o Servicios Web, son en realidad un conjunto de tecnologías que usan XML para intercambio de información en un entorno distribuido. Utiliza el protocolo de comunicación

SOAP basado en XML, estandarizado por W3C. Este protocolo, conceptualmente, permite enviar peticiones y respuestas en XML, normalmente sobre HTTP.

Esta alternativa es una buena solución para la integración de aplicaciones en Internet, ya que todos los firewalls reconocen HTTP y todos los fabricantes de tecnología ofrecen soporte para Servicios Web. Esta tecnología ofrece independencia del sistema y existen API para los lenguajes más usuales, disponibles para LAMP, .NET y J2EE.

Sin embargo, todavía no soportan completamente algunas funcionalidades como transacciones. En integraciones complejas en Intranet puede no ser la tecnología más adecuada actualmente. Con el tiempo y un mayor grado de maduración, esta tecnología promete ser muy potente.

3.3.4.2 CORBA

CORBA es la tecnología de integración de aplicaciones más madura y mucho anterior a los Servicios Web. Es una tecnología de objetos distribuidos que permite la invocación de métodos remotos (como si fueran locales) sin que importe la tecnología que usen cliente y servidor.

Utiliza el protocolo de comunicación IIOP. OMG, además de estandarizar el protocolo IIOP, ha estandarizado numerosos servicios CORBA como servicios de nombres, seguridad, transacciones, eventos, etc.

Existen firewalls que no reconocen IIOP y Microsoft no fabrica implementaciones de CORBA. Por esto, aunque CORBA ha sido y sigue siendo una buena solución para abordar integraciones complejas en Intranet, no ha tenido mucho éxito para la integración de aplicaciones en Internet. Para estas integraciones en Internet es mejor utilizar una tecnología que cuente con el apoyo de todos los fabricantes de tecnología (Sun, IBM, Microsoft,...), como por ejemplo Servicios Web.

Existen múltiples implementaciones comerciales y de código abierto de CORBA, disponibles para los lenguajes y sistemas operativos más usuales, aunque no de Microsoft.

3.4 ELECCIÓN DE TECNOLOGÍA

Para la elección de una tecnología adecuada es importante el tener en cuenta el punto de vista de los usuarios estas tecnologías. Estos usuarios exigen compatibilidad con otras tecnologías e incrementar la productividad. Además valoran positivamente la posibilidad de combinar distintos lenguajes de desarrollo e importar módulos o aplicaciones ya desarrolladas.

Los fabricantes que implementan distintas tecnologías, a su vez apoyan la compatibilidad de su producto con versiones anteriores, mayor estabilidad e integración en distintas plataformas y con diferentes lenguajes de programación.

La competencia entre distintas organizaciones ha sido en gran parte impulso del desarrollo, eficiencia y elegancia de las distintas soluciones existentes en el mercado.

Puesto que nuestra meta es el desarrollo de aplicaciones con movilidad, descartaremos la utilización de lenguajes client-side scripting, y por tanto, AJAX. Para nuestro propósito es conveniente descargar al máximo al cliente, de forma que los requisitos del cliente para acceder a la aplicación sean mínimos.

Entre las tecnologías existentes, aquellas que desde un principio se muestran como las alternativas más adecuadas son la plataforma J2EE y la plataforma .NET. Estas plataformas son más potentes que el resto de tecnologías presentadas, sin embargo, en muchos casos, el uso de estas otras alternativas puede ser suficiente según las expectativas de la aplicación. También es conviene tener en cuenta que, a veces, la solución más adecuada, podría ser la combinación de distintas tecnologías, por lo que es conveniente realizar un estudio previo de las ambiciones del sistema y las tecnologías más convenientes en cada caso. A continuación nos centraremos en las principales características y diferencias que presentan las plataforma .NET y J2EE.

La historia de la generación de contenido Web dinámico nos proporciona el contexto para poder apreciar los beneficios que estas plataformas nos ofrecen.

Cuando se comienza a indagar en estas tecnologías, se observa que cada una de las plataformas es avanzada, de alta ingeniería, con elementos nuevos e interesantes de acuerdo a los adelantos tecnológicos y las necesidades informáticas. Sin embargo, es muy importante la influencia en cada una de estas plataformas de las imágenes corporativas de cada una de las empresas que las representan. Este factor, independiente de la funcionalidad y rentabilidad del producto, es un factor determinante en la demanda.

Parece que Microsoft intenta luchar contra su tradicional imagen de “cerrado”. Ha colocado el futuro de C# en manos del European Standard Comité (ECMA), aunque eso sí, fue bien especificado en el momento de su entrega. Además utiliza SOAP, un estándar XML desarrollado por la industria para el intercambio de objetos. Ha declarado que es una plataforma de fuente abierta de Web Services, aunque ha desarrollado sus propios servicios de manera cerrada y propietaria.

A diferencia de .NET, que es un concepto global, J2EE es un conjunto de especificaciones. La principal ventaja de J2EE al estar basado en especificaciones, es la libertad de elección sobre los vendedores. Los componentes J2EE son interoperables entre productos J2EE desarrollados por distintos vendedores (IBM, HP, Sun,…), a diferencia de .NET donde todo gira alrededor de un único vendedor, Microsoft.

Ambas incluyen tecnologías equivalentes. ADO.NET, ASP.NET y COM+, son tecnologías .NET que podrían compararse a las tecnologías JDBC, JSP y EJB de J2EE. Ambas plataformas incluyen API para XML y permiten la creación de Web Services.

Unas de las apuestas de .NET es el soporte multilenguaje, aunque este idea no es nueva y ya fueron acoplados en la máquina virtual de Java lenguajes como Eiffel y Component Pascal. C#, en la plataforma .NET es un gran competidor a Java, y muchos desarrolladores, que en otras circunstancias utilizarían Java, se mantienen adeptos a .NET gracias a este lenguaje.

Por su parte, J2EE y Java se afianzan mutuamente. J2EE se beneficia de la expansión y aceptación de Java; y estándares como J2EE hacen que Java se refuerce cada vez más. J2EE permite a los vendedores colaborar conjuntamente y experimentar fusiones y adquisiciones.

Ambas plataforma ofrecen grandes ventajas y deben de ser capaces de proporcionar soluciones equivalentes, es por ello que en la mayoría de los casos, la elección de una plataforma u otra no se basa en la propia plataforma, sino en la compatibilidad y conveniencia desde el punto de vista comercial. A menudo viene determinada por las actuales relaciones de vendedores y clientes.

Otro factor a tener en cuenta en la elección de la tecnología adecuada, es la experiencia. En este sentido, J2EE es una plataforma más madura, aunque dependerá de la experiencia personal. La existencia de grupos de discusión y foros de simpatizantes con quién compartir información y código, así como manuales y ayudas, proporcionan más facilidades al desarrollador.

Para el desarrollo de nuestras aplicaciones **proponemos el uso de la plataforma J2EE**. Tanto la plataforma J2EE como la plataforma .NET ofrecen soluciones potentes para el desarrollo de aplicaciones, sin embargo la **independencia de vendedores**, la **posibilidad de implementaciones en Software Libre** y la **estandarización** son los factores de diferenciación que nos han hecho elegir la solución J2EE.

Además junto a las tecnologías estándares incluidas en J2EE, existen unos patrones de diseño específicos de J2EE orientados a resolver problemas comunes de diseño y que ayudan al desarrollo eficiente de potentes aplicaciones en esta plataforma.

La documentación disponible sobre esta tecnología desde las propias especificaciones y recomendaciones de diseño hasta ayudas, manuales, ejemplos e información compartida en comunidades de usuarios, es inmensa.

En capítulos posteriores se detallarán las características de la plataforma elegida y con ella se diseñará e implementará una aplicación que muestre toda su potencia.