1

Introducción

TABLA DE CONTENIDOS

APLICACIONES DEL ENTORNO EMPRESARIAL (ENTERPRISE APPLICATIONS)

MODELOS DE APLICACIONES ENTERPRISE

Dentro del desarrollo de aplicaciones del entorno empresarial y del entorno del comercio electrónico destacan cuatro modelos que deben ser cubiertos por cualquier sistema y/o metodología de desarrollo de software utilizada. Estos son:

APLICACIONES BUSINESS TO BUSINESS (B2B)

Son las de mayor volumen y el objetivo de los desarrollos empresariales. Por ejemplo una compañía aérea que vende las plazas de sus vuelos a un mayorista de viajes.

APLICACIONES BUSSINES TO CUSTOMER (B2C)

Este es el modelo más utilizado actualmente cuando una corporación "pone" su negocio en la red. Es decir permiten a los clientes el acceso a servicios y productos a través de la red. Por ejemplo una compañía aérea que nos vende billetes en la red o un banco que nos da acceso a sus servicios sobre nuestras cuentas.

APLICACIONES CUSTOMER TO CUSTOMER (C2C)

Es el modelo que empieza a desarrollarse con más empuje actualmente. En el una corporación articula las relaciones comerciales entre entidades individuales. Por ejemplo un banco que gestiona el cobro de recibos de un ayuntamiento a los ciudadanos o una casa de subastas en la red.

APLICACIONES INTRA-BUSINESS

Modelo tradicional seguido por las aplicaciones desarrolladas en las corporaciones para la mecanización de sus procesos. Normalmente supone la división del sistema en subsistemas que deben interactuar entre si, tal vez utilizando la red. Por ejemplo una empresa que tiene una aplicación para que los empleados soliciten sus vacaciones o para que si los comerciales realizan una venta y la registran en el sistema, inmediatamente se gestione su cobro en el departamento financiero.

CARACTERÍSTICAS DE LAS ARQUITECTURAS DE APLICACIONES

El objetivo principal al plantearse el desarrollo de cualquiera de los cuatro modelos de aplicaciones del entorno empresarial es dar entrada a una serie de cambios que están alterando las metodologías tradicionales de trabajo en este tipo de sistemas. Estos cambios generan una serie de problemas que dan lugar a la situación actual dentro de las empresas que desarrollan software o de las que quieren acceder a cualquiera de los cuatro modelos de aplicaciones corporativas.

Históricamente las tecnologías HTML y CGI proveen los mecanismos necesarios para acometer el desarrollo de los modelos de aplicaciones pero presentan diversos problemas:

- Muchos de los modelos de programación utilizados se basan en estándares predefinidos, ad-hoc o se mantienen en sistemas propietarios. Esto afecta

directamente a la productividad de los equipos y por tanto a los resultados económicos de aquellas empresas "usuarias" de su producto.

- Divergencias en las tecnologías existentes, que no permiten su integración y/o convivencia en las aplicaciones y además no permiten la adecuada división del trabajo dentro de los equipos.
- Coexistencia de diversos tipos de clientes (por internet, en la intranet, vía dispositivo de conexión, etc.) de forma que no implique la reprogramación de los sistemas.

Otro problema importante a tener en cuenta es la capacidad de crecimiento de los negocios y por tanto la escalabilidad en el modelo de aplicación utilizado. En este caso es fundamental el uso de mecanismos de gestión de usuarios y recursos, balanceos de carga, etc. Estas características requieren que el software desarrollado pueda cambiar fácilmente de entorno de ejecución sin tener que reconstruir ninguna de sus partes.

Cualquier empresa desea acometer el desarrollo de nuevos sistemas permitiendo el acceso a los sistemas existentes que por supuesto no desea retocar ni reprogramar. Esto requiere la orientación de las arquitecturas hacia la estructura de servicios accesibles de manera estándar que permitan la integración en las nuevas aplicaciones.

En la mayoría de desarrollos deben intervenir productos de soporte de ejecución, al margen de las herramientas de desarrollo, que faciliten tanto la escalabilidad como la integración y además puedan añadir características deseables en todo sistema como mejoras del rendimiento. Los nuevos modelos de aplicación deben permitir la libre elección de productos de desarrollo y soporte sin que ello suponga un trabajo extra en ninguna de las fases del proyecto. Así mismo es deseable la posibilidad de uso de elementos software desarrollados por expertos, dando lugar a un mercado poco explotado actualmente.

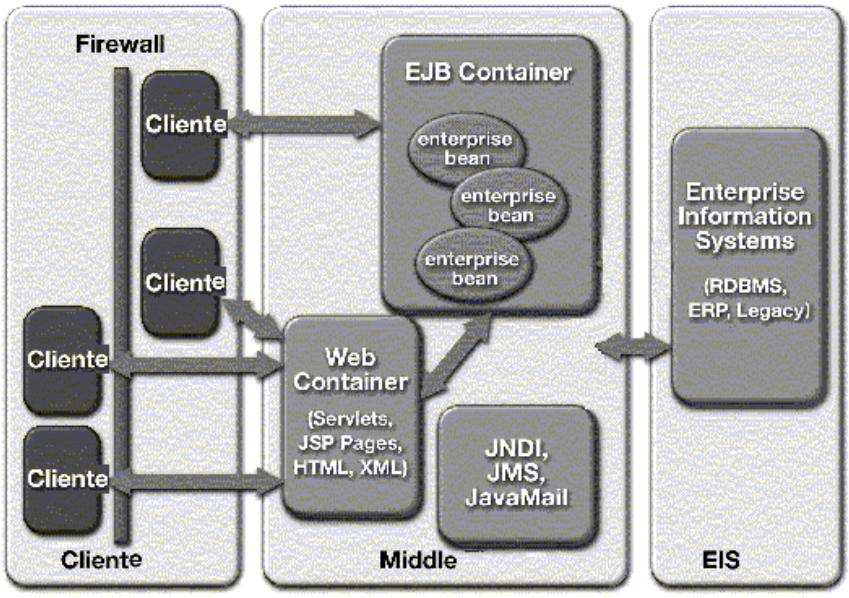
En estos sistemas de amplia difusión es necesaria la exposición de recursos valiosos para la empresa sin que esto suponga la perdida de ningún aspecto relativo a seguridad. Lo deseable es mantener la seguridad uniformemente, aun cuando se requiera la integración de distintos servicios de seguridad, en todos los puntos del sistema sin que esto suponga ninguna dificultad en el uso de la información y/o la funcionalidad de la aplicación.

PLATAFORMA PARA LAS SOLUCIONES: ARQUITECTURA J2EE

La plataforma J2EE esta diseñada para proveer al desarrollo de aplicaciones del entorno empresarial de los medios necesarios para satisfacer cualquiera de los cuatro modelos de aplicaciones existentes.

Las aplicaciones están formadas típicamente por varias capas. Una capa de cliente que conforma el interfaz de usuario. Suelen aparecer varias capas intermedias que aportan la funcionalidad de la aplicación y todos los servicios requeridos por el usuario. La última capa es donde reside la información corporativa a manipular por el sistema.

Este modelo de aplicaciones es multicapa y distribuido, es decir cada parte de un sistema puede residir en máquinas diferentes como podemos observar el la siguiente figura:



SOPORTE DEL ESTÁNDAR J2EE: SERVIDOR DE APLICACIONES

La plataforma J2EE es un modelo de desarrollo que además incluye el uso de productos que cumplan unas determinadas características por ello Sun Microsystems ha generado cuatro productos principales relacionados con este modelo:

- *Especificaciones de la plataforma:* que hacen referencia a la versión de API a utilizar en el desarrollo de los componentes que conformaran el sistema (Servlets, Jsp, EnterpriseBeans, JavaBeans, etc.…)
- *Test de compatibilidad:* que permite comprobar el cumplimiento de los estándares por parte de los productos de soporte necesarios en una arquitectura J2EE.
- *Implementación de referencia:* formada por los productos de soporte necesarios en una arquitectura J2EE, que permite por tanto realizar todas las labores de desarrollo y prueba de los sistemas sin recurrir a productos comerciales.
- *Blueprints:* que son una serie de documentos explican el proceso de desarrollo de una aplicación J2EE.

Dentro de este contexto cabe destacar la aparición de los productos comerciales denominados genéricamente **Servidores de Aplicaciones**. Estos productos si son validados por el test de compatibilidad, pueden ser compatibles J2EE. Esto quiere decir que permiten que cualquier desarrollo realizado siguiendo las especificaciones dadas por la plataforma puede ser implantado en su entorno.

J2EE especifica una serie de infraestructuras, generalmente en forma de servicios que un servidor de aplicaciones debe incluir:

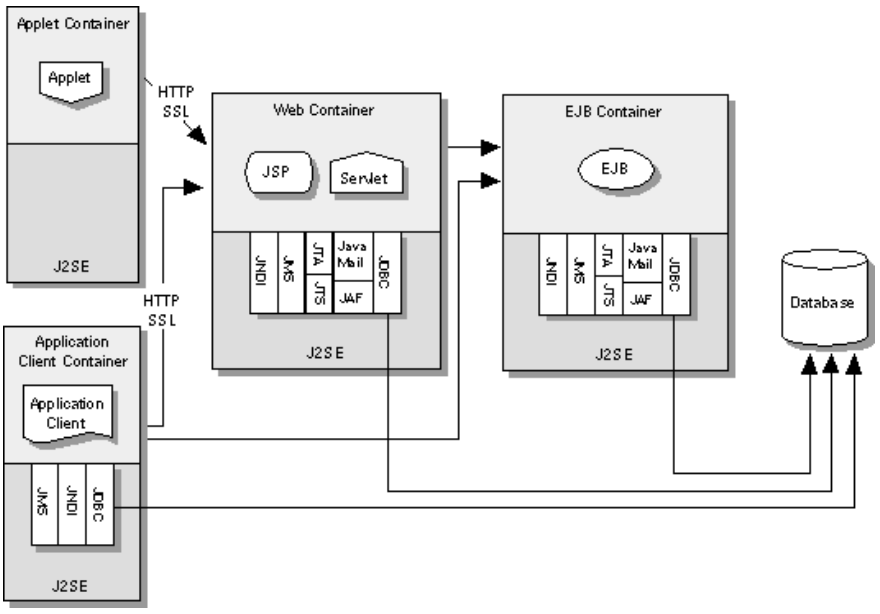
- Servicios de Seguridad y autentificación basados en el uso de identidades, roles y modelos.
- Gestor transaccional que bien controla directamente el trabajo del motor de acceso a datos de un producto de B.D. o que permita la comunicación de manera trasparente con los gestores de transacciones y recursos locales al producto de la B.D.

- Servicios de persistencia para soportar el almacenamiento en un producto de B.D. de la información necesaria para la gestión de la vida de lo EnterpriseBeans de tipo entidad.
- Servicio de nombres y directorios para registrar los servicios propios del servidor así como los componentes para los que sea necesaria la localización por nombre.
- Herramientas de montaje e implantación de los componentes que formen el sistema.
- Servicio de mensajes orientado a la comunicación con elementos externos.

Además estos productos pueden añadir servicios o herramientas que pueden ser aprovechadas durante los desarrollos, implantaciones y mantenimientos de las aplicaciones, por ejemplo:

- Gestores de recursos para la configuración y administración de las características de ciertos servicios, por ejemplo uso de un pool conexiones a base de datos, uso de un pool de flujos de ejecución, reutilización de conexiones entre componentes.
- Mecanismos de balanceo de carga, clusters o replicación en instalaciones que comprendan varias instancias de los servicios.
- Mecanismos de control de la concurrencia en el acceso a los componentes
- Mecanismos de cache que pueden mejorar significativamente el funcionamiento del producto servidor de aplicaciones, etc.…

Con todas estas características, una arquitectura J2EE genérica que da representada en la siguiente figura:



VENTAJAS DE LA PLATAFORMA J2EE

El modelo de componentes utilizado simplifica el proceso de diseño de la arquitectura y el desarrollo ya que permite montar una aplicación mediante combinación de distintos

componentes y servicios sin variar el sistema a nivel arquitectónico. Esto permite además la reconfiguración del sistema a través de los descriptores de implantación, sin ser necesaria la reescritura de los elementos.

La arquitectura base de la plataforma soporta la escalabilidad de cualquier sistema, a través del uso de servidores de aplicaciones orientados al rendimiento.

Los componentes desarrollados a través de los Api's correspondientes permiten la integración en la arquitectura de cualquier sistema de información corporativa o servicio externo ya disponible en la infraestructura de la empresa.

Los API's utilizados son:

- JTA - Java Transaction API permite la comunicación directa con cualquier gestor transaccional
- JTS - Java Transaction Service define la gestión de transacciones distribuidas a través de XA, basándose en el estándar del servicio definido en CORBA
- JNDI -Java Naming and Directory Interface™ permite el acceso a los servicios de nombres y directorios, como DNS, LDAP, NDS, y CORBA Naming
- RMI-IIOP - Remote Method Invocation (RMI) con protocolo estándar IIOP definido en CORBA, que permite las comunicaciones entre objetos remotos
- Java IDL - Java Interface Definition Language que permite las comunicaciones entre objetos Java a través del estándar CORBA
- JDBC - Database Access API que permite el acceso a cualquier sistema de base de datos relacional
- JMS - Java Messaging Service que permite el establecimiento de comunicaciones asíncronas con cualquier sistema de mensajes o de publicación/subscripción
- JavaMail - JavaMail permite la programación de componentes que trabajen como clientes o gestores de correo electrónico
- JAF - JavaBeans™ Activation Framework que permite a los componentes Java determinar el tipo de un elemento de datos y manipular su contenido de forma apropiada.

Existen además especificaciones para crear componentes que permiten el acceso a sistemas ERP o mainframes tradicionales pudiendo integrar su funcionalidad en los objetos de negocio desarrollados.

La plataforma ofrece un sistema de seguridad unificado en todos los niveles de acceso que permite el log-in único a la aplicación. Los requerimientos de seguridad se pueden definir de manera declarativa en los descriptores de implantación, lo cual simplifica el código sensiblemente.

TECNOLOGÍAS DE LA PLATAFORMA J2EE

La plataforma J2EE especifica tecnologías para soportar la construcción de las aplicaciones. Estas tecnologías se pueden dividir en tres categorías: componentes, servicios y comunicaciones.

COMPONENTES UTILIZADOS EN J2EE

Un componente es una unidad de software que cumple ciertas características especificadas en el proceso de análisis y diseño unificado con UML.

Todos los componentes dependen en tiempo de ejecución de una entidad denominada container, que gestiona su ciclo de vida, el uso por parte de los componentes clientes y permite la administración e implantación de los mismos. Los container permiten la personalización de los componentes de forma declarativa.

JAVABEANS

Componente que por especificación permite su inclusión en entornos gráficos de desarrollo o que permiten su introspección por parte de cualquier container que quiera gestionar su ciclo de vida, incluyendo la capacidad de persistencia.

APPLETS Y APLICACIONES CLIENTE

Componentes desarrollados para su ejecución en la parte cliente de la aplicación que se ejecutan en una máquina virtual independiente de la parte servidor.

COMPONENTES WEB (SERVLETS Y JSP)

Elementos capaces de relacionarse con una URL, con lo cual se activan ante los request correspondientes.

Esta activación se produce a través del Web Container, normalmente integrado en un servidor Web. El Container se encarga de mantener las comunicaciones según el protocolo establecido (HTTP, HTTPS, etc.…) y de pasar al componente la información necesaria para poder operar sobre los elementos request o response de dichos protocolos.

Además un container se debe encargar de facilitar a los componentes el acceso al resto de servicios de la plataforma J2EE.

Los servlets son clases Java en las que existe un método que recibe la información del request y del response para poderlos manipular. Normalmente este método analiza la información del request y general el contenido del response en HTML, XML, WML, etc.…

La tecnología JSP permite escribir directamente este contenido en el formato elegido incluyendo ciertos elementos de código Java que generarán la parte dinámica del mismo, basándose en el análisis de los elementos del request.

En tiempo de ejecución una JSP genera un Servlet y se ejecuta como tal.

ENTERPRISEBEANS

La arquitectura EJB es un tecnología que permite el desarrollo de componentes de negocio en la parte servidora de las aplicaciones. Su componentes son los EnterpriseBeans que incluyen la funcionalidad lógica del sistema. Existen dos tipos de EB: de Entidad y de Sesión

Los de sesión se usan para que realicen alguna funcionalidad en lugar del cliente, estas funcionalidades pueden ejecutarse en modo transaccional, ahora bien si el ciclo de vida del objeto se interrumpe estas transacciones no son recuperables de manera automática.

Los beans de entidad son objetos persistentes que representan el modelo de datos y mantiene por tanto sincronización con un sistema de almacenamiento de datos, por ejemplo una BD. Cada instancia particular est identificada por una clave primaria (Primary Key).

Los container de EJB gestionan su vida para que los componentes puedan funcionar correctamente incluyendo las características de transaccionales, de seguridad y de uso compartido que requieren. Además les permite el acceso a los servicios de la plataforma J2EE.

ROLES DE TRABAJO DEFINIDOS EN J2EE

Estos roles están definidos en función de un conjunto de tareas que el equipo de desarrollo debe cumplir para conseguir un sistema funcional. algunos de ellos coinciden con roles mas o menos preestablecidos en equipos de desarrollo para cualquier plataforma, pero existen características distintivas del trabajo particulares de J2EE.

Los roles principales son:

Proveedor de productos: Proporciona el sistema operativo, base de datos, servidor de aplicaciones o servidor web. Estos productos deben realizar la función de container y soportar las API definidas en J2EE. Además los proveedores suelen integrar herramientas particulares de gestión para los productos así como para las aplicaciones implantadas.

Proveedor de componentes: que desarrolla los elementos componentes del sistema. Puede trabajar con una o varias herramientas que faciliten las tareas de programación o diseño.

Montador de la aplicación: que monta la aplicación en base a los componentes desarrollados, no estando familiarizado con su código sino con sus descriptores. Especifica los objetivos a cumplir en la implantación de los elementos en la plataforma. Normalmente trabaja con algún tipo de herramienta visual.

Implantador: Experto en el producto J2EE que puede utilizar herramientas provistas por el proveedor de productos para realizar su función. Es el encargado de instalar los componentes y aplicaciones en la plataforma operativa, en los servidores. Debe resolver todas las referencias externas realizadas desde los componentes y declaradas por el desarrollador de componentes o por el montador.

Administrador del sistema: Responsable de las tareas administrativas de los productos utilizados y la supervisión del funcionamiento de la aplicación. Puede utilizar herramientas de monitorización y gestión propias de cada proveedor de producto.

SERVICIOS DE J2EE

Los servicios permiten la simplificación del desarrollo y la personalización en tiempo de implantación basada en los interfaces de los propios servicios.

Los servicios dotan de la capacidad de realización de tareas específicas a los componentes desarrollados que deberán ser totalmente independientes de la implementación del servicio. De esta forma las aplicaciones son poco acopladas, aumentando su flexibilidad y manejabilidad.

Los servicios pueden ser: verticales (propios de contexto de aplicación) u horizontales (basados en la infraestructura y que resuelven temas básicos).

Los servicios pueden moverse o variar su implementación, ya que su localización debe ser trasparente y los componentes deben ser independientes de la implementación de los servicios que utilizan.

A continuación se explican los servicios J2EE.

Servicios de nombres y/o directorio: Deben permitir el uso en el desarrollo del programa de un contexto único de nombres para que tanto los clientes, los

EnterpriseBeans y los componentes puedan localizar cualquier recurso (del sistema u otros componente) por su nombre. Cualquier sistema de nombres o directorios será accesible vía JavaNamingDirectoryInterface (JNDI) siendo el proveedor del servicio el encargado del correcto funcionamiento de dicho interfaz sobre su sistema de nombres.

Servicios de implantación: Las aplicaciones J2EE se implantan como un conjunto de unidades anidadas. Cada unidad contiene un descriptor de implantación (generalmente XML) que describe como montar y implantar la unidad así como elementos relativos a la configuración de los servicios usados por los componentes de la unidad.

Servicios transaccionales: El soporte de las transacciones simplifica el desarrollo del software, que no debe incluir el código dedicado a la recuperación de unidades de trabajo o de la ejecución concurrente de las mismas. Un servicio transaccional debe soportar transacciones planas (no anidadas) para las que se gestiona automáticamente la propagación y la coordinación entre gestores transaccionales.

Dentro de la plataforma J2EE se contemplan dos tipos de transacciones

- Transacción JTA o transacción global que se puede expandir sobre múltiples componentes y gestores de recursos. Los componentes de la aplicación tiene acceso a una transacción JTA o global a través de algún objeto que implemente el interfaz javax.transaction.UserTransaction.

  Estos objetos son accedidos por los componentes de diferentes maneras. El único comportamiento estandarizado es el referente a la arquitectura EJB.

- Transacción local que es específica a una conexión con el gestor de recursos de un sistema EIS.

Servicios de seguridad: que permiten controlar que durante la ejecución sólo se realicen accesos autorizados a los diferentes elementos del sistema. El control de seguridad se realiza en dos pasos: autentificación y autorización.

En la autentificación se obtiene un Principal (identidad correspondiente a un llamador) que basándose en la política de dominios de seguridad de Java es autorizado o no a realizar el acceso en cuestión. Esta comprobación se realiza en relación a los roles de seguridad identificados por el desarrollador o el montador y mapeados sobre los Principal por el implantador del sistema. Las autorizaciones pueden especificarse en el código de los componentes o bien de manera declarativa dándoles de esta forma mayor independencia.

Acceso a los servicios EIS: Los sistemas EIS deberían contemplarse desde el sistema J2EE como servicios. La última versión de la especificación plantea este tipo de conectividad a través de los denominados elementos Connector. Esta arquitectura define una serie de mecanismos escalables, seguros y transaccionales para la integración de cualquier sistema EIS. Los Connector son suministrados por el proveedor del sistema EIS y pueden ser incorporados en un servidor EJB.

TECNOLOGÍAS DE COMUNICACIÓN

Las tecnologías de comunicaciones permiten la interoperabilidad entre componentes y servicios. La especificación dice que una plataforma J2EE debe soportar tipos de comunicación a distintos niveles como se muestra a continuación.

Además en la plataforma se especifican los distintos formatos de la información que puede ser intercambiada entre componentes. HTML 3.2, gif, jpeg, jar, class y XML son los formatos requeridos por la especificación de J2EE.

Protocolos de internet: para la comunicación de las diferentes partes del sistema. En la plataforma está especificado el uso de TCP/IP, HTTP 1.0 y SSL 3.0.

Protocolos de interoperabilidad entre componentes: que permiten invocaciones a métodos de objetos remotos, con paso de parámetros completo. La plataforma especifica el uso de los siguientes protocolos de esta categoría: RMI, JavaIDL, RMI-IIOP.

La invocación a métodos remotos, RMI, es un API que permite al programador construir código de cliente y de servidor residente en objetos albergados en un entorno distribuido. El uso del protocolo de transporte se realiza a través de JRMP, utilizándose, si es necesaria, la serialización de objetos.

JavaIDL es el API y el ORB que en la plataforma se utiliza para realizar accesos a objetos CORBA desde cualquier componente desarrollado en la aplicación.

RMI-IIOP es el API RMI en el que el uso del protocolo de transporte se realiza a través de IIOP (Internet Inter ORB Protocol) de forma que es posible construir código en el que interactuan clientes y servidores CORBA o RMI.

La arquitectura EJB, por ejemplo, utiliza RMI-IIOP para permitir el acceso desde cualquier otro componente de la plataforma a los componentes de negocio.

Tecnologías de mensajes: que permiten las comunicaciones principalmente con sistemas EIS que consumen mensajes o el trabajo con sistemas de e-mail.

Java Message Service, JMS, es el API que permite el trabajo con sistemas de mensajes asíncronos punto a punto o con estilo publicación-subscripción. La plataforma especifica la existencia de este API en los productos compatibles J2EE pero en la versión actual no requiere que dichos productos incluyan una implementación de los objetos que implementan el interfaz. Estos objetos pueden ser suministrados por el proveedor de productos para ser utilizados en el desarrollo de los componentes.

El API JavaMail pone a disposición de los desarrolladores un conjunto de clases e interfaces que le permiten construir el código de envío o recepción de mensajes de correo electrónico según las especificaciones estándar. Existe además el JAF que puede ser necesario en casos de uso sofisticado de los tipos MIME dentro de los mensajes.

RESUMEN

La plataforma J2EE permite el desarrollo de distintas arquitecturas de aplicaciones que engloban los cuatro modelos de aplicaciones enterprise: C2C, B2B, B2C e Inter-Business.

Cualquiera de estos modelos puede implementarse basándose en el conjunto de componentes disponibles: EJB, Servlets, JSP, etc.… En las etapas de desarrollo el trabajo se centra en la lógica del sistema y no es necesario pensar en la infraestructura necesaria para la ejecución de dichos elementos.

Además en el desarrollo de las aplicaciones el trabajo esta simplificado por el soporte ofrecido por la plataforma y es posible la división de los equipos de desarrollo de forma apropiada. Estos desarrollos por ser compatibles J2EE exponen una serie de características deseables en cualquier desarrollo de software. La plataforma especifica una serie de condiciones para cada uno de los tres elementos fundamentales de la infraestructura: componentes, servicios y tecnologías de comunicación