TEMARIO OPOSICIÓNS INFORMÁTICA

GRUPO A1 - ESCALA DE SISTEMAS E TECNOLOXÍA DA INFORMACIÓN

TEMA 29. ARQUITECTURA WEB EN .NET.
ARQUITECTURA WEB EN J2EE.

Autor: Juan Marcos Filgueira Gomis

TEMA 29. ARQUITECTURA WEB EN .NET. ARQUITECTURA WEB EN J2EE.

29.1. INTRODUCIÓN E CONCEPTOS

O desenvolvemento de aplicacións, servizos web e outros compoñentes software ten hoxe en día dúas vertentes principais, a plataforma .NET e a plataforma J2EE, ou JEE nome co que é coñecida actualmente ao cambiar de versión. A rivalidade entre estas dúas tecnoloxías é moi forte, pois proporcionan solucións similares fortemente soportadas polas compañías dun e do outro bando.

.NET é a plataforma de desenvolvemento proposta pola empresa Microsoft para o mundo dos servidores de aplicacións que funciona como ferramenta de deseño e programación ademais de proporcionar un amplo conxunto de utilidades estendidas de apoio ao desenvolvemento neste tipo de contornos. Pola súa banda **JEE**, (en inglés *Java Platform, Enterprise Edition*) ou Java EE, é unha evolución da plataforma Java para desenvolver e soportar compoñentes software segundo un conxunto de especificacións de xeito que poidan operar nun servidor de aplicacións, incluíndo tamén ferramentas de deseño e programación.

A pesar de que ambas plataformas perseguen o mesmo obxectivo, teñen unha serie de particularidades ou diferenzas que se ven acentuadas polas guerras comerciais entre as empresas que as soportan. Mentres JEE ten soporte multiplataforma, .NET funciona unicamente baixo a a familia de Sistemas Operativos Windows. Mentres JEE basease exclusivamente na linguaxe Java, en .NET permítense moitos linguaxes de alto nivel, aínda que na práctica os principais sexan C# e VB .NET. JEE leva máis anos de experiencia no mercado, mentres que .NET é máis recente. Así mesmo JEE presenta maior soporte en canto a solucións e posibilidades de software libre, que son moi escasas e de pouca calidade en .NET. Con JEE pódese instalar unha infraestrutura de alto rendemento de xeito completamente gratuíto.

29.2 ARQUITECTURA WEB EN .NET

.NET é, segundo a empresa Microsoft, unha plataforma para o desenvolvemento de servidores, clientes e servizos. Representa un conxunto de tecnoloxías que teñen como núcleo principal o .NET Framework, un marco de desenvolvemento e compoñente software que pode instalarse en Sistemas Operativos da familia Windows (Windows 2003, Vista, Windows 7, ...). Existe unha versión adaptada para móbiles dispoñible en Windows Mobile. A norma ISO/IEC 23271 recolle un conxunto funcional mínimo que deben cumprir os produtos software desenvolvidos para que poidan funcionar dentro do marco de traballo. Esta e máis normas se recollen nos estándares:

a) **Estándar ECMA-334.** Especificación da linguaxe C#. (2006).
b) **Estándar ECMA-335.** Especificación da linguaxe de infraestrutura común (CLI). (2010).

Por contra, outros compoñentes coma ASP .NET, *Windows Forms* ou ADO .NET non se atopan estandarizados. Paralelamente, unha vez publicados os documentos de especificación da arquitectura .NET apareceu o **Proxecto Mono** co obxectivo de implementar o marco de traballo .NET Framework empregando código aberto, para a partir de aí desenvolver aplicacións para sistemas UNIX/Linux.

29.2.1 .NET Framework

Marco de traballo que proporciona o conxunto de ferramentas e servizos para o desenvolvemento de compoñentes software, aplicacións, servidores e servizos web. Pode dividirse en tres bloques principais:

1) O **Contorno de Execución Común** (en inglés C*ommon Language Runtime* ou CLR). Encárgase da xestión de código en execución, control de memoria, seguridade e o outras funcións relacionadas co Sistemas Operativo.
2) A **Biblioteca de Clases Base** (en inglés .*NET Framework Base Classes*). Realizan a función de API de servizos a disposición dos desenvolvedores para tarefas como xestión de ficheiros, mensaxería, procesos en varios fíos, acceso a datos, encriptación, etc...
3) **Control de acceso a datos,** que permite realizar as operacións de acceso a datos a través de clases e obxectos do *framework* incluídos no compoñente ADO.NET.
4) O **Motor de Xeración da Interface de Usuario**, que permite crear interfaces para aplicacións de escritorio ou web empregando compoñentes específicos coma ASP.NET para web, *Web forms* para aplicacións de escritorio ou *Web services* para servizos web.

29.2.2 Contorno de Execución Común (CLR)

O Contorno de execución común ou CLR é o encargado de xestionar o código en tempo de execución. De xeito análogo á Maquina virtual de Java este contorno permite executar aplicacións e servizos web ou de escritorio en calquera cliente ou servidor que dispoña deste software. A diferenza da Máquina virtual de Java o soporte de .NET é multilinguaxe permitindo C++, C#, ASP .NET, Visual Basic, Delphi, e moitos outros. Ademais permite integrar e herdar compoñentes entre diferentes linguaxes, con maior ou menor fortuna á hora de sacar proveito de linguaxes antigas.

O contorno de desenvolvemento compila o código fonte en calquera das linguaxes soportadas a un código intermedio denominado **CIL** (en inglés *Common Intermediate Language*) de maneira análoga ao BYTECODE de Java. A esta linguaxe intermedia chégase empregando a especificación CLS (en inglés *Common Language Specification*) onde se especifican unhas regras necesarias para crear o código intermedio CIL compatible co CLR. Así mesmo, o CLR dispón de compiladores coma JIT (en inglés *Just In Time*) ou AOT (en inglés Ahead Of Time) adaptados a cada linguaxe.

JIT xera o código máquina real en cada máquina a partir dese código intermedio conseguindo independencia do hardware. Esta compilación faise en tempo de execución a medida que a aplicación ou servizo invoca métodos ou funcións. Para axilizar o procesamento este código máquina obtido en tempo de execución gárdase na memoria caché actualizándose tan só cando se produce algún cambio no código fonte, momento no se que se repite o proceso. Por contra **AOT**, compila o código antes de executarse co cal logra un maior rendemento en execución pero menos independencia da plataforma. No tocante a JIT acostuma a distinguirse entre:

1) **Jitter estándar.** Compila o código CIL a nativo baixo demanda.
2) **Jitter económico.** Non optimiza, traduce cada instrución así precisa menos tempo e memoria de compilación.
3) **Prejitter**. Realiza unha compilación estática dun compoñente software completo.

As principais **vantaxes** deste modelo de compilación son:

- ✔ A **reutilización** de compoñentes escritos en diferentes linguaxes nunha mesma aplicación ou servizo web.

- ✔ **Modularidade** grazas á implementación do patrón Interface para cada compoñente ou libraría xa que será accesible dende calquera linguaxe a través da súa API (ASP, C#, Java, Phyton, etc ...)
- ✔ **Integración multilinguaxe**, xa que cada linguaxe cun compilador a CIL pode integrarse na plataforma co cal cada compoñente nesa linguaxe pode integrarse unha aplicación ou servizo web .NET.
- ✔ **Seguridade**. Polo illamento do código de usuario respecto dos accesos a datos e outras partes críticas do Sistema Operativo.

Figura 1: Estrutura multilinguaxe do CLR

O CLR cumpre ademais a función de proporcionar unha ampla gama de servizos ás aplicacións. A través da API de cada servizo os compoñentes web poden ter acceso a funcionalidades comúns, como:

a) **Seguridade acceso ao código ou CAS** (en inglés *Code Access Security*). Controla que tipo de operacións pode realizar un código segundo se identifique na sinatura do ensamblado ou na orixe do código. Así mesmo recoñece directivas de administración do sistema para compoñentes ou a nivel de *host*. Cando un compoñente trate de acceder a recursos protexidos do sistemas lanzarase o CAS para comprobar os permisos, pero a este nivel non se poden establecer comprobacións dinámicas, por exemplo contra Bases de datos.

b) **Atributos de protección do *host* ou HPA** (en inglés *Host Protection Attributes*). Mantén unha lista de atributos protexidos, denegando o acceso ou modificación dos mesmos. Algúns destes atributos serían *SharedState*, para estados compartidos, *Synchronization*, para permitir a capacidade de sincronizar procesos no *host* ou *ExternalProcessmgmt* que indica se os procesos no *host* se poden controlar externamente a través da API.

c) Dominios de aplicación. Definen dominios illados de código para restrinxir o acceso dos compoñentes software, creando unha zona reservada para un subproceso. Por norma xeral o CLR crea para cada aplicación un dominio en tempo de execución pero pode precisar dominios específicos para compoñentes DLL ou externos.

d) **Comprobación da seguridade de tipos.** Reserva espazos de memoria para cada obxecto segundo o especificado para o seu tipo. Cando o espazo é aleatorio ou queda algún oco fóra do espazo do obxecto, quere dicir que ese obxecto non ten seguridade de tipos. Coa compilación JIT realízase unha comprobación en tempo de execución para verificar se cada obxecto ten seguridade de tipos. Do mesmo xeito impide variables sen valores iniciais ou *cast* non seguros.

e) **Cargador de clases.** Permite cargar en memoria clases e tipos de datos a partir da interpretación dos metadatos. Existe ademais a posibilidade de crear cargadores personalizados, aínda que só para Java, xa que por motivos de rendemento resulta máis óptimo empregar un ensamblado con outras linguaxes. Na compilación o cargador realiza a función de evitar código innecesario a través de funcións *stubs* que substitúe co código correcto baixo demanda.

f) **Recolección de lixo** (en inglés *Garbage Collector*). Este servizo execútase de xeito continuo para buscar e eliminar de memoria os obxectos que non sexan referenciados ou rematen o tempo de espera de utilización.

g) Motor de interacción COM. Realiza funcións de conversión de datos e mensaxes ou *marshaling* dende e cara obxectos COM, o que permite a integración con aplicación *Legacy*.

h) Motor de depuración. Permite realizar un seguimento da execución do código aínda que mesture diferentes linguaxes.

i) API multifío (en inglés *multithread*). Proporciona unha API e as clases necesarias para xestionar a execución de fíos paralelos.

j) **Xestor de excepcións.** Realiza a xestión estruturada e integración con *Windows Structured Exception Handling* de excepcións aínda que o erro proveña de diferentes linguaxes nun mesmo compoñente e mesmo no código aínda non executado. Este código pode incluír excepcións SHE do tipo C++ ou resultados HRESULTS típicos de COM.

k) API da Biblioteca de Clases Base (BCB). Interface coa BCB do marco de traballo que realiza a integración do código co motor de execución.

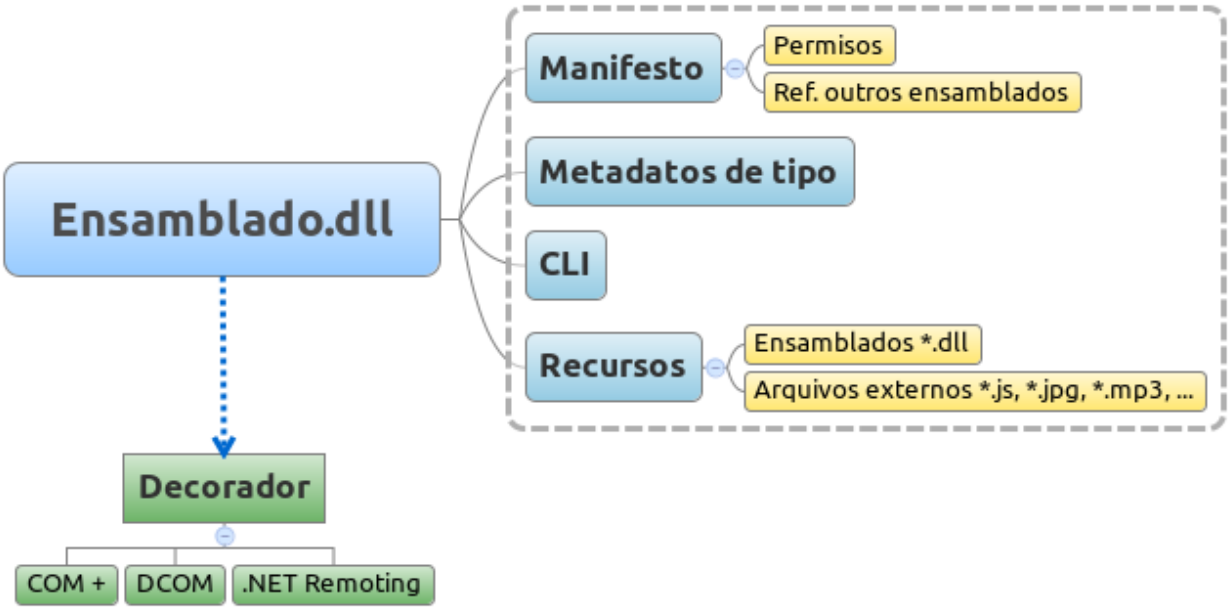


Figura 2: Ensamblados.

No .NET Framework cando se compila un programa ou aplicación web xérase una arquivo denominado **ensamblado** que contén o código compilado á linguaxe intermedia CLI e un manifesto con permisos e referencias a outros ensamblados, compoñentes software ou servizos web. Son paquetes ou librarías EXE ou DLL destinadas ao control de versións, seguridade e comprobacións de implementación polo miúdo. Os ensamblados levan unha indicación que define o contorno de execución no que se debe lanzar: COM+, DCOM, .NET Remoting, ...

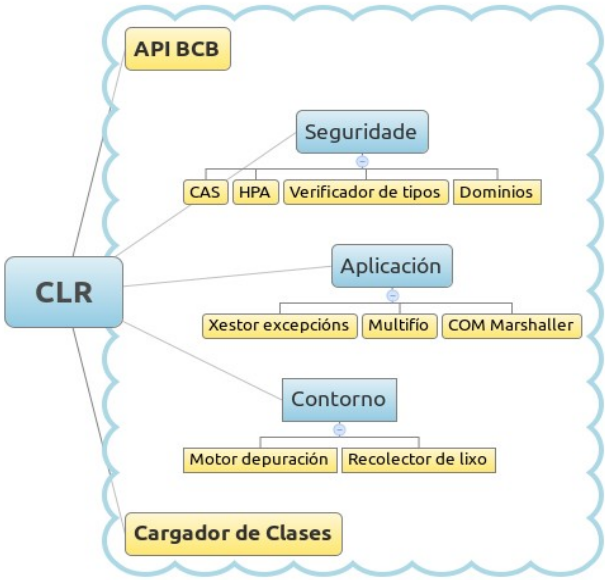


Figura 3: Servizos do CLR

29.2.3 Biblioteca de Clases Base (BCB)

A Biblioteca de Clases Base é unha API de alto nivel para permitir acceder aos servizos que ofrece o CLR a través de obxectos nunha xerarquía denominada **espazo de nomes**. Agrupa as funcionalidades de uso frecuente permitindo a súa redefinición. Atópase implementada en CIL polo que pode integrarse en calquera outra linguaxe. É un conxunto de clases, interfaces e tipos valor que son a base sobre as que se crearán as aplicacións, compoñentes e controis do .NET Framework. Permite realizar operacións como: soporte para diferentes idiomas, xeración de números aleatorios, manipulación de gráficos e imaxes, operacións sobre datas e outros tipos de datos, integración con APIS antigas, operacións de compilación de código adaptada ás diferentes linguaxes de .NET, elementos para interfaces de usuario, tratamento de excepcións, acceso a datos, encriptación, administración de memoria, control de procesos, etc...

Espazo de nomes	Utilidade e obxectos
System	Tipos básicos, táboas, excepcións, datas, recolector de lixo, etc...
System.Collections	Manipulación de coleccióǹs como pilas, colas, *hash*, etc...
System.Data	Arquitectura ADO.NET (Obxectos *DataSet, DataTable, DataRow, DataView*, ...)
System.IO	Manipulación de E/S arquivos e outras orixes de datos
System.Net	Xestión de comunicacións de rede (TPC/IP, *Sockets*, ...)
System.Security	Xestión das políticas de seguridade do CLR
System.XML	Acceso e manipulación de datos en documentos XML con compatibilidade co W3C (Transformacións en System.Xml.Xls e serialización para servizos web en System.XML.Serialization)
System.Web	Servizos para xestión de caché, seguridade e configuración para Servizos Web, estado das sesións e interfaces de usuario
System.Web.Services	Xestión dos requirimentos de Servizos Web
System.Web.UI	Controles para interfaces de usuario *HTMLControl* para mapeo de etiquetas HTML e *WebControl* para estruturar controis de usuario avanzados coma *DataGrids*
System.Windows. Forms	Creación da IU do cliente
System.Drawing	Acceso a funcionalidades gráficas básicas da GDI+ (Funcionalidades avanzadas en *System.Drawing.Imaging*, *System.Drawing.Text e System.Drawing.Drawing2D)*

System.Reflection	Acceso a metadatos sobrelos ensamblados, módulos, membros, parámetros e outras entidades do código administrado
System.JSON	Proporciona compatibilidade baseada en estándares JSON, notación de objetos JavaScript (en inglés *JavaScript Object Notation*)
System.Threading	Manipulación de procesos e fíos de execución
System.Text	Proporciona clases para manipular as codificación de caracteres UNICODE e UFT-8 conversión de bloques de caracteres en bloques de *bytes* e viceversa
System.Transactions	Contén clases que permiten crear e administrar transaccións, admitindo participantes distribuídos, notificacións de fase e inscricións duradeiras
System.Resources	Proporciona clases e interfaces que permiten crear, almacenar e administrar recursos de localización
System.Runtime. Remoting	Proporciona a interface para acceso remoto e marco para a implantación de sistemas de compoñentes distribuídos
Microsoft.CSharp	Clases para realizar a compilación e execución de código en C# (O mesmo para outras linguaxes)

Táboa 1: Principais espazos de nomes.

29.2.4 Control de acceso a datos.

O control de acceso a datos, documentos XML e servizos de datos no marco .NET Framework recóllese na arquitectura ADO .NET, coma evolución do *ActiveX Data Objects*. A súa orientación principal é o acceso a datos do xestor de base de datos relacional *SQL Server,* orixes XML e orixes de datos vía obxectos OLE DB e ODBC. As **conexións** realízanse identificando os provedores de datos a través dos obxectos (Connection, Command, DataReader e DataAdapter). Unha vez establecida a conexión entra en escena o principal elemento do marco, o *Dataset* que recolle os **resultados** cargados a partir dunha orixe. Á súa vez pode particularizarse con outros elementos da base de datos con obxectos coma: *DataTable, DataView, DataRelation, DataRow, DataColum* ou *Constraint*. Os **obxectivos** de deseño principais deste marco son:

- ✔ Soporte á tecnoloxía ADO previa.
- ✔ Integración con tecnoloxías baseadas en XML.
- ✔ Soporte a modelos de arquitectura multicapa.

Nas últimas versións incorpórase o **Marco de Entidades** (en inglés *Entity Framework***)** que permite realizar Consultas Integradas nas Linguaxes (en inglés *Language Integrated Query*) ou **LINQ**.

Este marco permitirá empregar LINQ sobre moitos compoñentes de acceso a datos novos: *LINQ to SQL, LINQ to DataSet* e *LINQ to Entities*. Asocia unha chave lóxica ás entidades, dotando ao modelo relacional ou conceptual de orientación a obxectos. Utilidades do marco de traballo coma **SQLMetal** permiten a xeración automática de clases a partir da base de datos ou documentos XML.

29.2.5 Motor de Xeración de Interfaces de Usuario

A parte do marco de traballo encargada da xeración de interfaces de usuario e servizos web sería a arquitectura ASP .NET. O conxunto de clases correspondentes agrúpanse nos espazos de nomes: System.Web, System.Web.Services, System.WebUI. As páxinas web desenvolvidas con ASP .NET teñen a extensión **ASPX** e son coñecidas con **Formularios Web** (en inglés *Web Forms*). Paralelamente a esta tecnoloxía de presentación atoparíamos **Formularios Windows** (en inglés *Windows Forms*) para aplicacións de escritorio e tecnoloxías para Móbiles, sendo a primeira a máis empregada para aplicacións de servidor. No espazo System.Web.UI recóllense as dúas clases principais de controis os HTML para acceso directo ás etiquetas estáticas destas linguaxes e os controis web que incorporan código de servidor dinámico. A arquitectura recomendada emprega o modelo **Code-Behind** no que se crea un arquivo separado co código de servidor, a diferenza da arquitectura DNA para ASP anterior. Os **Controis de usuario** (en inglés *User Controls*) seguen a mesma estrutura que os Formularios Web, pero derivan do espazo `System.Web.UI.UserControl`, e gárdanse en arquivos **ASCX**, que deberían seguir tamén o modelo Code-Behind. O marco incorpora tamén elementos para control do estado e a sesión así coma outros para seguridade, autenticación de usuarios, uso de roles, uso do servizo Indigo ou **WCF** (en inglés *Windows Communication Foundation*). Así mesmo, permite a integración con AJAX, a través da incorporación dun **Toolkit** na aplicación, ou **WPF** (en inglés *Windows Presentation Foundation*) baseado en XALM e marco para Silverlight.

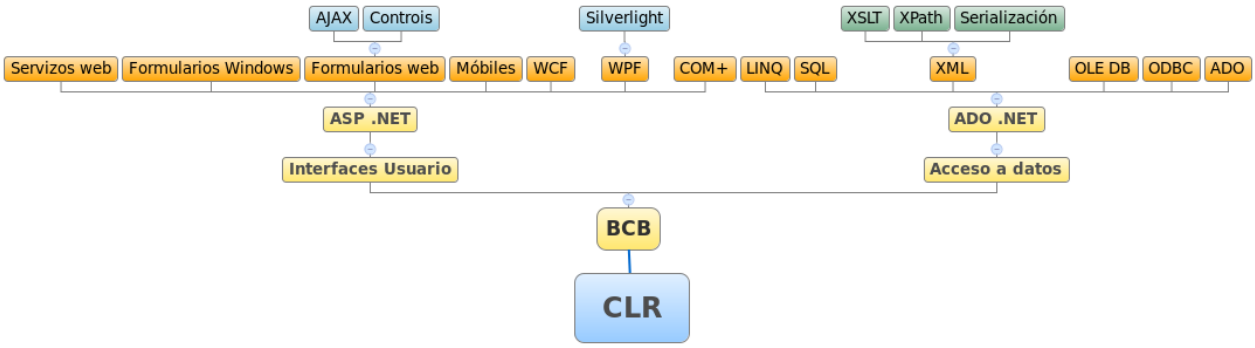


Figura 4: Arquitectura xeral

29.2.6 Niveis lóxicos.

Nos modelos e solucións que propón esta estrutura establécense unha serie de servizos xenéricos presentes na maioría de aplicacións corporativas actuais. Esta división permite definir un deseño e unha arquitectura específicos para cada nivel, facilitando o desenvolvemento e soporte da aplicación.

1. **Servizos de usuarios.** Atópanse na primeira liña de interacción cos usuarios e proporcionan a interface de acceso ao sistema que deriva en chamadas aos compoñentes do nivel de Servizos corporativos. En ocasións considéranse dentro deste nivel procesos fóra das interfaces de usuario, como procedementos de control ou automatizados que non requiren a presenza dun usuario.
2. **Servizos corporativos.** Encapsulan a lóxica corporativa proporcionando unha API das funcionalidades básicas do sistema. Isto permite abstraer os servizos de usuario da lóxica corporativa e manter diferentes servizos de usuario a partir das mesmas funcionalidades. Cada funcionalidade pode precisar dispoñer de varios servizos corporativos.
3. **Servizos de datos.** Sería a parte máis illada do usuario, proporcionando o acceso a datos e a outros sistemas ou servidores. Establecen diferentes API xenéricas das que poden facer uso os Servizos corporativos. Conteñen unha ampla gama de orixes de datos e sistemas de servidor, encapsulando regras de acceso e formatos de datos.

29.2.6 Solucións de integración.

Coa tecnoloxía .NET existen tres principais plantexamentos de solucións arquitectónicas:

1) **SOA** (en inglés *Service Oriented Architecture*). Entende a comunicación entre aplicacións e compoñentes coma servizos, non necesariamente servizos web, demandados por clientes ou subscritores e proporcionados e publicados por provedores.
2) **MOA** (en inglés *Message Oriented Architecture*). A comunicación realízase por paso de mensaxes forzando un modelo SOA distribuído.
3) **EAI** (en inglés *Enterprise Integration Application*). Especifica unha serie de requirimentos de integración e comunicación en sistemas regulados polos patróns de integración Mediación e Federación, onde un sistema EAI fai funcións de *Hub* ou *bus* de comunicacións.

29.3 ARQUITECTURA WEB EN J2EE

JEE representa un conxunto de especificacións para plataformas de desenvolvemento baseadas en linguaxe Java para servidores de aplicacións en arquitecturas de múltiples capas. O **JCP** (en inglés *Java Community Process*) é o organismo encargado de validar os requisitos de conformidade para cada plataforma e aceptala. As plataformas JEE constan dos seguintes compoñentes:

1) Un conxunto de especificacións.
2) Un test de compatibilidade ou CTS (en inglés *Compatibility Test Suite*)
3) Unha implementación de referencia para cada especificación.
4) Un conxunto de guías de desenvolvemento e boas prácticas denominadas JEE Blueprints.

As principais **especificacións** que inclúe JEE dan soporte a Servizos web, RPC baseado en XML, Mensaxería XML, despregues, servizos de autorización, conexión remota RMI, JSP, JSF, JSTL, Servlets, Portlets, Applets, JavaBeans, esquemas XML, acceso a datos JDBC, documentación Javadoc, transformacións XSL, etc...

O soporte multiplataforma do Java ten a súa base na **Maquina Virtual (**VM/JVM ou KVM/CVM para móbiles), unha plataforma lóxica capaz de instalarse en equipos con diferente hardware e Sistema operativo e interpretar e executar instrucións de código **Java bytecode.** A especificación da VM tamén se recolle como especificación pola JCP e do mesmo xeito están dispoñibles test de compatibilidade. A forma máis habitual para a VM é mediante un compilador JIT pero tamén permite interpretación. Do mesmo xeito permítese execución segura mediante o modelo das Java Applets. Programas de cliente que se executan nunha VM dentro do navegador logo de descargar vía HTTP código do servidor, que se executa nunha *Sandbox* moi restrinxida.

Os compoñentes software web e de negocio dentro desta tecnoloxía despréganse a través de **Contedores** (en inglés *containers*). Os contedores son implementacións de arquitecturas JEE que proporcionan os servizos do servidor de aplicacións aos compoñentes, incluíndo seguridade, acceso a datos, manexo de transaccións, acceso a recursos, control de estados, xestión do ciclo de vida e comunicacións entre outros. Antes de executarse un compoñente software debe configurarse coma un servizo JEE e despregarse dentro dun contedor. Os principais serían os contedores web para Servlets e JSP, os contedores EJB para compoñentes da lóxica de negocio, e contedores de aplicacións cliente e contedores de *Applets* para os programas de cliente e código de cliente para o navegador respectivamente.

Os principais **servizos** que proporciona JEE xunto coas súas respectivas API serían:

1) **HTTP e HTTPS.** Para control das comunicacións web e SSL a través destes protocolos. As API de servidor veñen dadas polos paquetes de clases Servlets e JSP e a de clientes no paquete Java.Net.
2) **JDBC** (en inglés *Java Data Base Connection*). API de acceso a datos en sistemas xestores de bases de datos relacionais vía SQL. Por un lado aporta a interface para ser empregada polos compoñentes software e por outra a interface para que os provedores poidan desenvolver os controladores específicos. As versións máis recentes son as JDBC 3.0 e 4.0, que inclúen os paquetes *java.sql* e *javax.sql*.
3) **JSTL** (en inglés *Java Server Pages Standard Tag Library*). Proporciona as funcionalidades de para etiquetas nas páxinas JSP.
4) **RMI-IIOP** (en inglés *Remote Method Invocation-Internet Inter-ORB Protocol*). Proporciona a API para permitir comunicacións en aplicación distribuídas a través de JAVA RMI, por exemplo para acceder a compoñentes EJB. Os protocolos máis habituais son JRMP, de RMI e IIOP, de CORBA.
5) **IDL** (en inglés *Java Interface Definition Language*). Permite a comunicación de clientes con servizos CORBA a través do protocolo IIOP, servizos SOAP ou RPC.
6) **JNDI** (en inglés *Java Naming and Directory Interface*). Proporciona o servizo de nomes e directorios, indicando o contexto de cada obxecto e as relacións entre eles. Divídese en dúas interfaces, a API de programación e unha SPI que permite conectar con provedores de servizos de nomes e directorios sendo os principais LDAP, CORBA e RMI.
7) **JAXP** (en inglés *Java API for XML Processing*). Soporta o procesamento de documentos XML que cumpra cos esquemas do W3C a través de DOM, SAX e XSLT.
8) **JMS** (en inglés *Java Message Service*). Proporciona a API de envío de mensaxes para comunicarse cun MOM (en inglés *Message-Oriented Middleware*), unha abstracción independente do provedor para comunicacións entre sistemas.
9) **JavaMail**. Proporciona a interface para controlar o envío e recepción de correos electrónicos. Pode soportar o formato MIME grazas á súa integración con marco de traballo JAF.
10) **JAF** (en inglés *Java Beans Activation Framework*). API que proporciona o marco de traballo para activación que soporta as peticións doutros paquetes.

11) **JTA** (en inglés *Java Transaction API*). Orientada cara o manexo de transaccións e a permitir a comunicación entre contedor e compoñentes do servidor de aplicacións coma os monitores transaccionais e os administradores de recursos.
12) **JAX-RPC** (en inglés *Java API for XML-based RPC*). Proporciona soporte para comunicacións remotas de tipo RPC entre clientes e servizos web cos estándares HTTP e SOAP. Soporta outros estándares coma WSDL, así coma SSL e TTL para autenticación. O SAAJ (en inglés *SOAP with attachments API for Java*) engade a posibilidade de arquivos ou notas achegados coas mensaxes.

Cada compoñente denomínase **Módulo** JEE de xeito que unha aplicación estará formada por un conxunto de módulos sendo cada un un compoñente para un contedor. Existen tres tipos de módulos:

1) **Arquivos JAR** (en inglés *Java Archive*). Agrupación de arquivos Java e recursos segundo o formato ZIP. Empaquetan compoñentes EJB segundo a estrutura de directorios do código, engadindo unha carpeta especial, META-INF, con metadatos.
2) **Arquivos WAR** (en inglés *Web Application Archive*). Agrupan nun único arquivo unha aplicación web, incluíndo Servlets, arquivos JSP, contido estático e outros recursos web.
3) **Arquivos EAR** (en inglés *Enterprise Application Archive*). Agrupa nun único arquivo varios módulos dunha aplicación coma arquivos WAR ou compoñentes EJB e outras librarías en arquivos JAR empaquetados cos seus respectivos recursos. Así mesmo inclúese o descritor de despregue da aplicación na carpeta META-INF.
4) **Arquivos RAR** (en inglés *Resource Adapter Archive*). Contén un adaptador de recursos de xeito análogo a un controlador JDBC e similar aos EAR, podendo ir contido nun arquivo deste tipo. O formato vén definido na especificación JCA (en inglés *Java EE Connector Architecture*).

De xeito xeral convén considerar á plataforma como JEE, se ben, existen diferentes **edicións**, sendo as principais:

1) **J2ME**. (en inglés *Java 2 Platform Micro Edition*). Para desenvolvemento de aplicacións para dispositivos móbiles, electrodomésticos e equipos PDA. Desenvolveuse mediante o JPC baixo a especificación JSR 68.
2) **J2SE**. (en inglés *Java 2 Platform Standard Edition*). Para desenvolvemento de aplicacións de uso xeral en estacións de traballo. Desenvolveuse mediante o JPC baixo diferentes especificacións segundo as versións existentes: 1.4, 5.0 e 6.

3) **J2EE**. (en inglés *Java 2 Platform Entreprise Edition*). Para desenvolvemento de aplicacións destinadas a servidores de aplicacións para dar soporte a sistemas distribuídos en N capas. Estandarizada polo JPC a partir da versión 1.4 acostuma a denominarse JEE.

Para cada edición pode distinguirse entre a SDK (en inglés *Software Development Kit*), co software e recursos destinados ao desenvolvemento de aplicacións e o JRE (en inglés *Java Runtime Environment*) co contorno e librarías principais para permitir a execución das aplicacións.

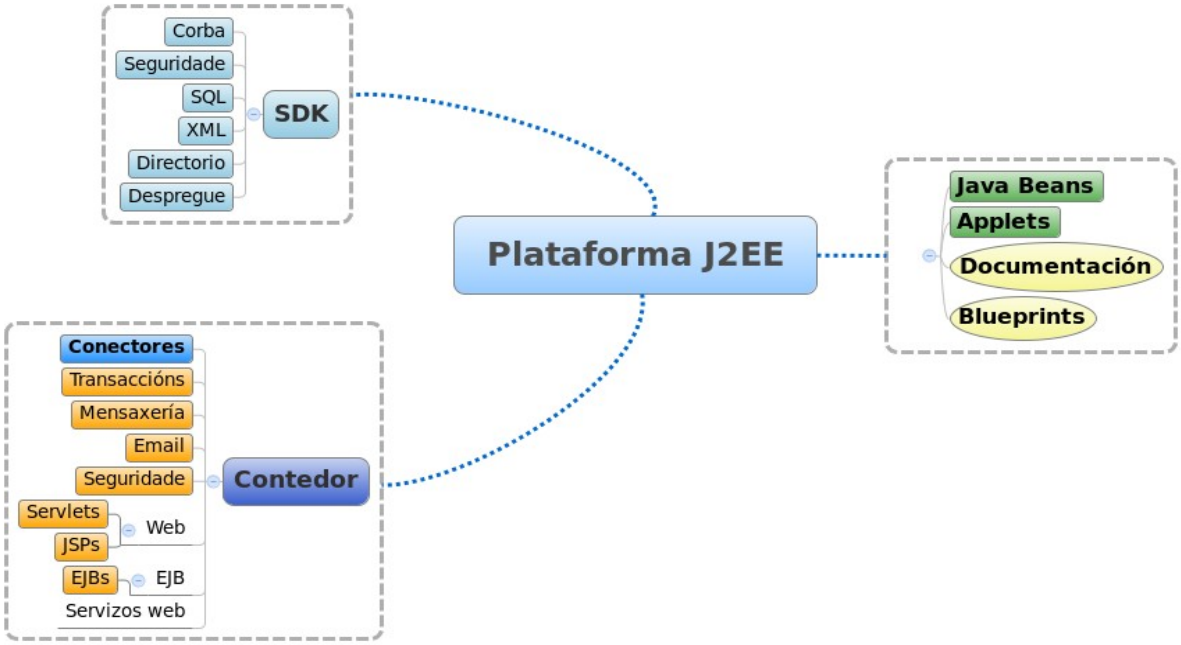


Figura 5: Plataforma J2EE.

29.3.1 Modelo de desenvolvemento

O modelo de desenvolvemento máis habitual na arquitectura JEE é un modelo separado en múltiples capas sendo o habitual un mínimo de tres, pero podendo chegar a 5 ou 7 segundo a complexidade do sistema. O obxectivo é minimizar o solapamento entre elas para que os cambios e modificacións se limiten ao mínimo necesario co ideal de que se poida cambiar unha das múltiples capas sen ter que modificar o resto. Mellórase a sostibilidade, crecemento do sistema e a reutilización de compoñentes no mesmo e entre sistemas. Así mesmo permítese unha maior heteroxeneidade de clientes ou elementos de cliente e presentación ao modificar só a capa máis próxima ao usuario. O deseño do modelo é análogo a solucións en .NET pero a implantación diferente. As 5 capas máis habituais se describirán a continuación.

29.3.1.1 Capa de cliente.

Agrupa os elementos da interface de usuario máis próximos ao cliente. Exemplos destes elementos serían o código (X)HTML/XML e Javascript, os Applets, arquivos de recursos e tecnoloxías RIA. Os tipos de aplicacións cliente máis habituais serían os navegadores web, as aplicacións de escritorio e actualmente cobran forza as aplicacións para dispositivos móbiles. Un aspecto importante neste modelo e garantir que os EJB da lóxica de negocio sexan accesibles tan só dende interfaces remotas a través do patrón *SessionFacade*. A variedade de interfaces actual fai que aparezan *frameworks* de xeración dinámica dos mesmos baseados na linguaxe **XUL** (en inglés XML *User-Interface Language)*, baseada en XML. Permite incrustar XHTML e outras linguaxes coma MathML ou SVG ademais de CSS. Existen varias alternativas de librarías XUL coma Luxor, XWT, Thinlets ou SwingML.

29.3.1.2 Capa de presentación.

Contén toda a lóxica de interacción directa entre o usuario e a aplicación. Encárgase de xerar as vistas máis axeitadas para amosar a información a través de formatos e estilos adecuados. Compóñense dunha serie de Servlets e páxinas JSP que se encargan de devolver o código que irá á capa de cliente logo de comunicarse coa capa de lóxica de negocio para obter os resultados. Pode localizarse nunha aplicación de escritorio ou nun contedor web. Ademais de ensamblar as diferentes vistas, controla o fluxo de navegación e fai funcións de autenticación, permisos de acceso e autorización de usuarios, etc... O patrón máis habitual nesta capa será o MVC (en inglés *Model View Controller*). Entre as tendencias actuais atópanse implementacións deste modelo coma Swing o JFace para aplicacións de escritorio, mentres que dentro dos *frameworks* web atoparíanse Struts, JSF, Tapestry, Expresso e moitos outros. Sendo Struts unha especie de estándar de feito. Estes *frameworks* ademais incorporan outros servizos como etiquetas personalizadas JSTL para interfaces de usuario, manexo de XML, acceso a datos, modelos, filtros, etc...

29.3.1.3 Capa de lóxica de negocio.

Contén os compoñentes de negocio reutilizables EJB ou POJO, que representan o conxunto de entidades, obxectos, relacións, regras e algoritmos do dominio ou negocio no que opere o sistema. Nesta capa a solución POJO e unha opción sinxela que pode mesturar elementos das capas de integración e datos, mentres que os EJB distinguen os obxectos de sesión e as entidades, recomendado nos Blueprints de Sun, emprazando cada un na súa correspondente capa. O patrón básico nesta capa será o *SessionFacade* onde un único Bean de sesión encárgase de recibir as chamadas de cliente-presentación e dirixilas dentro do contedor de EJB illando esta capa.

De xeito análogo establécese o modelo dos Bean de mensaxería, que realizan comunicación asíncrona mediante JMS nun servidor MOM (en inglés *Messaging Oriented Middleware*), que pode ser un servidor externo ao servidor de aplicacións. Tamén funciona cun patrón Fachada centralizando as chamadas remotas.

29.3.1.4 Capa de Integración.

Agrupa os compoñentes encargados do acceso a datos, sistemas *legacy*, motores de regras de *workflow*, acceso a LDAP, etc... Poden realizar cambios de formato na información, pero transformacións máis complexas deberían realizarse na capa de lóxica de negocio, restrinxido esta á lóxica de acceso a datos ou DAO (en inglés *Data Access Objects*) e os encapsuladores de datos e entidades VO (en inglés *Value Object*). Os VO poden implementarse como POJO ou EJB de entidade. Como ocorría anteriormente no caso dos POJO gáñase en facilidade pero pérdense servizos e funcionalidades coma os de persistencia. Os EJB suman complexidade, pero melloran o rendemento en memoria. No tocante ao acceso a datos aparecen as seguintes alternativas:

1) **JDBC.** Para POJO ou Beans de entidade con control de persistencia. É unha solución sinxela, con poucas funcionalidade pero que fai uso dunha API de uso estendido.
2) **DAO.** Vai un paso máis alá que o JDBC incorporando interfaces para abstraer o acceso a datos e facelo independente da linguaxe do xestor. Cada interface terá unha implementación diferente para cada xestor de bases de datos.
3) ***Frameworks* de persistencia.** Fan as funcións de motores de correspondencia de obxectos a bases de datos relacionais definindo entidades e relacións vía XML. Realizan gran parte das funcións de acceso a datos automaticamente. As solucións de uso máis estendido son os *frameworks* Hibernate, iBatis, TopLink, JPA ou a través de EJB.
4) **JDO** (en inglés *Java Data Objects*). Sistema de persistencia estándar a partir dunha especificación JEE, engadindo ademais da correspondencia entre o modelo relacional e os obxectos a posibilidade de permitir definir os obxectos sobre a base de datos. As implementacións de uso máis estendido son OJB, XORM, Kodo JDO ou LiDO.

29.3.1.5 Capa de sistemas de información.

Atópase integrada polos sistemas de bases de datos, ficheiros, sistemas 4GL, ERP, Data Warehouse, Servizos web e calquera outros sistema de información da organización. Nesta capa irían os conectadores para diferentes sistemas de información heteroxéneos e os propios recursos que integran os sistemas de información. As solucións máis habituais enuméranse a continuación.

1) **JCA** (en inglés *J2EE Connector Architecture*). Define unha interface de acceso común independente do sistema, coa mesma API para todos. Basease no concepto de adaptador de recursos, sendo cada adaptador un controlador específico para un sistema de información. As operacións básicas que define a especificación son: xestión das conexións de acceso a JCA, seguridade, transaccións, multiproceso, paso de mensaxes e portabilidade dentro dos servidores de aplicacións.
2) **JMS** (en inglés *Java Message Service*). O servizo de mensaxes Java emprega colas de mensaxes para o traspaso de información entre compoñentes software establecendo unha infraestrutura MOM con dous modelos de API, Punto a punto, entre dous únicos clientes ou Publicador/subscritor onde varios clientes segundo o seu rol envían ou len mensaxes.
3) **Servizos web.** Permiten a comunicación entre sistemas heteroxéneos a través do acceso á URL de aplicacións empregando protocolos baseados en XML como SOAP ou SAAJ. Os clientes acceden ao servizo a partir da súa interface definida perante WSDL (en inglés *Web Service Definition Language*) ou inserida nalgún rexistro de servizos web.

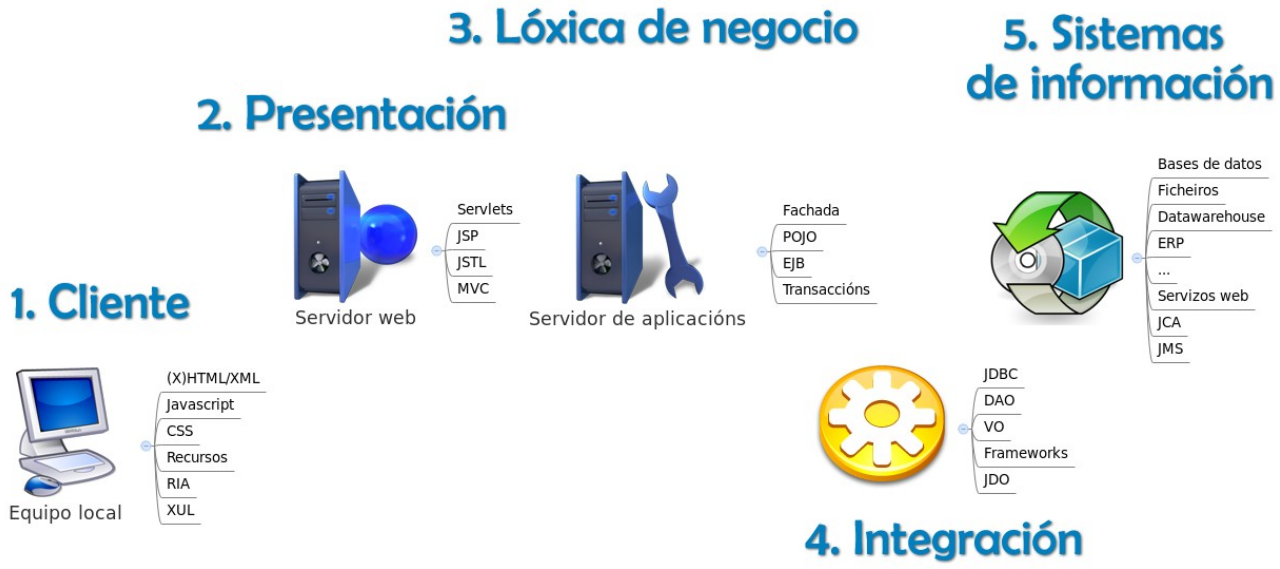


Figura 6: Modelo de desenvolvemento en capas.

29.3.2 Servidores de aplicacións.

O servidor de aplicacións será o encargado de soportar a maioría das funcionalidades e servizos da tecnoloxía JEE, sendo o núcleo desta arquitectura. Cando un servidor de aplicación implementa a tecnoloxía JEE ten que proporcionar todos os compoñentes definidos na especificación e por tanto calquera aplicación JEE poderá despregarse e executarse no devandito servidor.

O servidor de aplicacións disporá de diferentes contedores para Applets e aplicacións clientes, web e EJB, sendo estes últimos os que se encargarán de operar coa lóxica do dominio, xestión de transaccións, persistencia, control do fluxo, etc...

a) **Tomcat**. Sen presentar tódalas funcionalidades dun servidor de aplicacións, este servidor libre de Apache incorpora o servidor web Apache e soporte para JSP e Servlets co contedor Catalina. Presenta diferentes módulos de soporte de aplicación como seguridade SSL, SSO, JMX, AJP, JSF, conectador Coyote para peticións HTTP, soporte para Comet, Recolector de lixo reducido así como ferramentas web para despregue e administración.
b) **JBoss**. Un dos servidores de aplicacións libres de uso máis estendido composto por un Contedor de Servlets para JSP e Servlets e un Contedor de Beans. A diferencia de Tomcat implementa todo o conxunto de servizos especificados por JEE. Como contedor de Servlets emprega unha adaptación de Tomcat ou o contedor Jetty. Entre os módulos e funcionalidades que soporta destaca que permite a creación de *cluster*, soporte EJB, JMX, Hibernate, JBoss AOP para dotar a clases Java de persistencia e funcionalidade transaccional, sistema caché, JSF, Portlets, JMS, Servidor de correo, xestión de contidos foros e portais, entre outras moitas.
c) **Geronimo**. Outro produto libre de Apache, compatible con JEE que inclúe JDBC, RMI, JM, Servizos web, EJB, JSP, Servlets e outras tecnoloxías. A principal característica deste servidor é que integra un gran número doutras solucións xa existentes: Tomcat e Embarcadero como contedores web, OpenEJB como contedor de Servlets, OpenJPA, Apache Axis, Apache CXF e Scout Apache para servizos web, Derby para o acceso a datos, e WADI para establecer clusters e balanceo de carga, entre outros.
d) **JonAS**. Outra alternativa libre a JBoss, aínda que non soporta por completo JEE. Permite integración con Tomcat ou Jetty como contedores web e ten contedor de EJB. Entre módulos e servizos incorpora: Xplus, Hibernate, TopLink, OpenJPA, JORAM como implementación de JMS, varios protocolos RMI (IIOP, JRMP, IRMI), soporte LDAP, servizos web Axis e outros moitos.
e) **Glassfish**. Alternativa libre de Sun, agora Oracle, que ten como base o *framework* para persistencia Toplink. Incorpora ademais módulos para soporte EJB, JAX-RS, JSF, RMI, JMS, servizos web, na liña dos anteriores, e novidades como Apache Félix, unha implementación de OSGi (en inglés *Open Services Gateway*) e Grizzly que fai uso da nova API de Java de E/S (NIO) para mellorar a escalabilidade.

f) **WebSphere.** Alternativa comercial de IBM cunha versión de libre distribución. A versión libre, máis lixeira, basease no servidor Geronimo diferenciándose deste en que inclúe soporte para DB2, Informix, soporte RAC de Oracle e outras bases de datos así como mellores librarías para XML. Outras tecnoloxías serían: os Servlets SIP (en inglés *Session Initiation Protocol*) que utilizan elementos multimedia en tempo real, mensaxería instantánea e xogos en liña; o *framework* Spring; protocolos de seguridade Kerberos e SAML. A diferencia con outros servidores e que posúe ferramentas de administración máis avanzadas, sobre todo para sistemas en cluster e soporte para *mainframes*.

g) **Weblogic**. Alternativa comercial de Oracle baseada en Glassfish, incorporando servizos do Weblogic server sobre a JVM JRockit. En concreto Weblogic Server proporciona os Servizos web Oracle WebLogic Server Services Web, a Application Grid coma solución de *grid* de datos, soporte de conectividade con Tuxedo (WTC), soporte de RAC para Oracle, SAML, unha API de integración con .NET a JMS.NET, Spring e o *framework* de diagnose WLDF.

h) **Coldfusion**. Alternativa comercial de Adobe das máis valoradas actualmente, diferenciándose polo soporte a tecnoloxías RIA principalmente Flash. Implementa parte dos servizos JEE pero pode integrarse con outros servidores de aplicacións como WebSphere ou Jboss, podendo despregarse como aplicación Java. Ademais leva incorporado o servidor de aplicacións Adobe JRun. Destaca polo soporte en tecnoloxías AJAX, Flex, PDF, RSS, Flash Remoting, integración .NET e ferramentas de administración avanzadas.

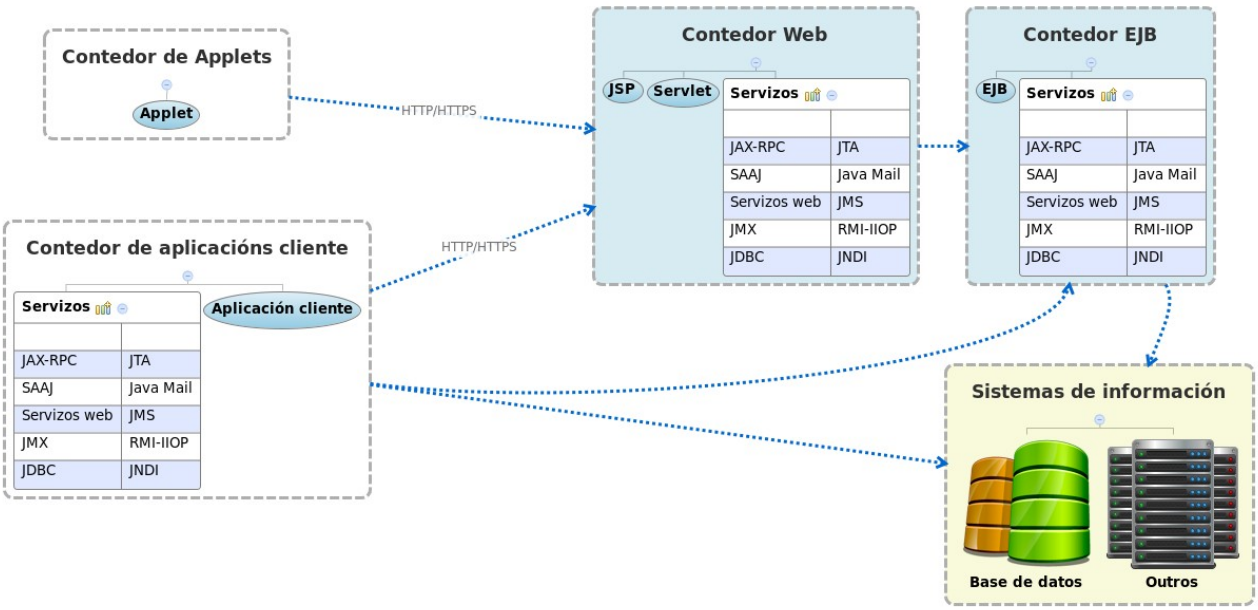
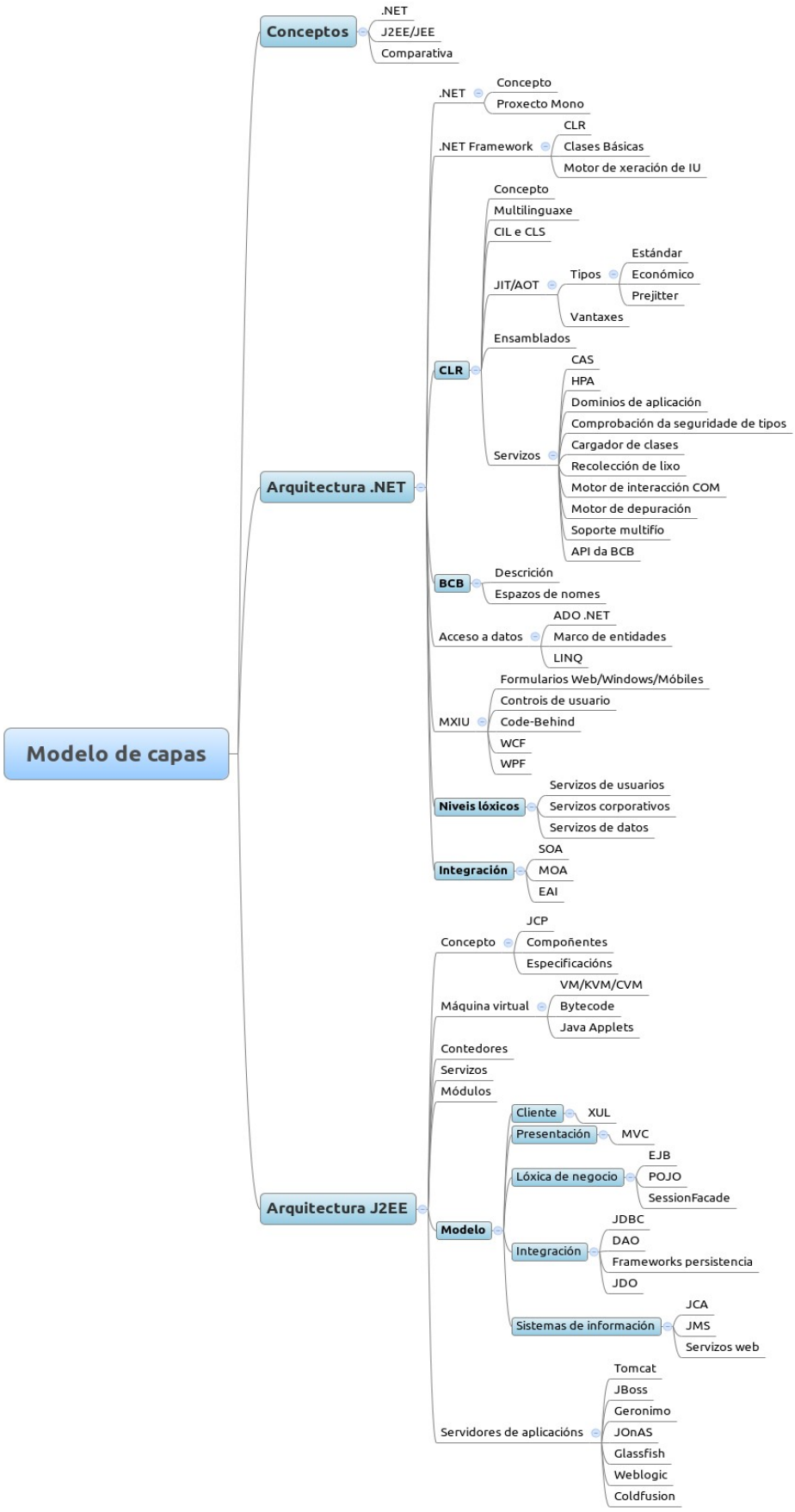


Figura7: Arquitectura J2EE

29.4. ESQUEMA


Modelo de capas
Conceptos
.NET
J2EE/JEE
Comparativa
Arquitectura .NET
.NET
Concepto
Proxecto Mono
.NET Framework
CLR
Clases Básicas
Motor de xeración de IU
CLR
Concepto
Multilinguaxe
CIL e CLS
JIT/AOT
Tipos
Estándar
Económico
Prejitter
Vantaxes
Ensamblados
Servizos
CAS
HPA
Dominios de aplicación
Comprobación da seguridade de tipos
Cargador de clases
Recolección de lixo
Motor de interacción COM
Motor de depuración
Soporte multifío
API da BCB
BCB
Descrición
Espazos de nomes
Acceso a datos
ADO .NET
Marco de entidades
LINQ
MXIU
Formularios Web/Windows/Móbiles
Controis de usuario
Code-Behind
WCF
WPF
Niveis lóxicos
Servizos de usuarios
Servizos corporativos
Servizos de datos
Integración
SOA
MOA
EAI
Arquitectura J2EE
Concepto
JCP
Compoñentes
Especificacións
Máquina virtual
VM/KVM/CVM
Bytecode
Java Applets
Contedores
Servizos
Módulos
Modelo
Cliente
XUL
Presentación
MVC
Lóxica de negocio
EJB
POJO
SessionFacade
Integración
JDBC
DAO
Frameworks persistencia
JDO
Sistemas de información
JCA
JMS
Servizos web
Servidores de aplicacións
Tomcat
JBoss
Geronimo
JOnAS
Glassfish
Weblogic
Coldfusion

29.5. REFERENCIAS

Varios autores.
Biblioteca MSDN de Microsoft. (2003).

Jef Ferguson e outros.
La biblia de C#. (2003).

Benjamín Aumaille.
J2EE. Desarrollo de aplicaciones Web. (2002).

I. Singh, B. Stearns e outros.
Desingning Enterprise Applications with the J2EE Platform. (2002).