

**Descripción de la
Tecnología empleada en
Aplicaciones Aon**



■	■		■		■		■
■	■						
■	■	■	■		■		■
■	■	■	■				
■	■	■	■	■	■		■
■	■	■	■	■	■		
■	■	■	■	■	■	■	■

REVISIONES

Realizado por: Eugenio Castellano

Revisado por: Iñaki Ayerbe

Aprobado por: Julio García



ÍNDICE

Introducción	4
Java	4
Java 2 Enterprise edition	4
Entorno de desarrollo	5
Plataforma	5
Sistema operativo	6
Sistemas gestores de base de datos	6
Usabilidad	6
Arquitectura de Capas	7
Objetivo	7
Patrón MVC	7
Modelo	7
Vista	7
Controlador	8
Programación por Capas	8
Capa de presentación	9
Capa de negocio	9
Capa de datos	10
Seguridad	11
Beneficios de la arquitectura diseñada	11
Frameworks empleados	12
Plataforma AON.	12
JavaServer Faces	12
MyFaces	12
Hibernate	13
JasperReports	14
Facelets	14
Código Abierto (<i>Open Source</i>)	14
Dependencia con librerías externas.	15

INTRODUCCIÓN

El presente documento describe la tecnología de las aplicaciones desarrolladas bajo la plataforma AON. El documento no entra en descripciones técnicas de las diferentes partes de la aplicación. Pretende proporcionar una visión global de la estructura de la aplicación y una explicación de la arquitectura elegida.

Java

Las aplicaciones AON utilizan Java como lenguaje de programación. Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems al inicio de la década de los 90.



La orientación a objetos, se refiere a un método de programación y al diseño del lenguaje. La idea es diseñar el software de forma que los distintos tipos de datos que use estén unidos a sus operaciones. Así, los datos y el código (funciones o métodos) se combinan en entidades llamadas objetos. Un objeto puede verse como un paquete que contiene el “comportamiento” (el código) y el “estado” (datos). El principio es separar aquello que cambia de las cosas que permanecen inalterables.

A diferencia de los lenguajes de programación convencionales que generalmente están diseñados para ser compilados a código nativo, Java es compilado en un bytecode que es ejecutado, por una máquina virtual Java, eliminando de esta forma la dependencia directa con el sistema operativo de la máquina donde se ejecuta el programa.

La independencia de la plataforma, significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware. Es lo que significa ser capaz de escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo, tal como reza el axioma de Java, “write once, run everywhere”.

Java 2 Enterprise edition

Las aplicaciones AON adoptan el estándar de programación J2EE, Java 2 Enterprise Edition. J2EE es una plataforma de programación —parte de la Plataforma Java— para desarrollar y ejecutar software de aplicaciones en Java con arquitectura de *n* capas distribuida, basándose en componentes de software modulares, y ejecutándose sobre un servidor de aplicaciones.

La plataforma Java EE está definida por una *especificación*. Similar a otras especificaciones del equipo Java Community Process (JCP), Java 2 EE es también considerada como un estándar debido a que los suministradores deben cumplir ciertos requisitos de conformidad para declarar que sus productos son *conformes a Java EE*.



Java EE incluye varias especificaciones de programación, tales como JDBC (acceso a bases de datos), RMI (acceso remoto a métodos), email, JMS (mensajería), Servicios Web, XML, etc., y define como coordinarlos. Java EE también configura algunas especificaciones únicas para Java EE, para componentes como JavaBeans, servlets, portlets (siguiendo la especificación de Portlets Java), JavaServer Pages y varias tecnologías de servicios Web. Esto permite al desarrollador crear una aplicación de empresa que es portable entre plataformas y escalable.

Otros beneficios destacables son, por ejemplo, que el manejo por parte del servidor de aplicaciones de las transacciones, seguridad, escalabilidad, concurrencia y gestión de los componentes que son desplegados; significando esto que los desarrolladores pueden concentrarse más en la lógica de negocio de los componentes en lugar de las tareas de mantenimiento de bajo nivel.

Existen multitud de servidores certificados J2EE, entre los cuales podemos destacar los productos de las siguientes compañías: Jboss AS (Jboss), Geronimo (Apache), JonAS (ObjectWeb), SAP NetWeaver (SAP), Java System Web Server, Java System Application Server y Glashfish (Sun), WebSphere (IBM), WebLogic (BEA Systems).

Asimismo existen multitud de entornos de desarrollo y herramientas de construcción automática de dependencias, compiladores, contenedores, etc. que ayudan a la programación con la arquitectura J2EE

Entorno de desarrollo

El desarrollo Java no requiere de ningún entorno de desarrollo específico. Ahora bien, existen herramientas que facilitan en gran medida la programación, la ejecución y las pruebas de la aplicación. Para el desarrollo de las Aplicaciones AON se ha utilizado el IDE Eclipse.

Eclipse es un IDE de carácter gratuito, considerado entre las mejores existentes en la actualidad y con multitud de plugins que agilizan tareas como la validación de la calidad del código, ejecución de módulos, realización de tests-units, pruebas y debug del código en ejecución.

El desarrollo del IDE Eclipse ha sido auspiciado por empresas como: Borland, IBM, MERANT, QNX Software Systems, Rational Software, Red Hat, SuSE, TogetherSoft, Webgain, Serena, Sybase, Fujitsu, Hitachi, Instantiations, MontaVista, Scapa Technologies and Telelogic, ETRI, HP, MKS, SlickEdit, Oracle, Cataliza Systems, Flash linee, Parasoft, SAP, teamstudio, TimeSys, CanyonBlue, Ensemble Systems, Intel, Micro Focus, Tensilica y Wasabi.

Plataforma

Se requiere de un Servidor de Aplicaciones Java que cumpla la especificación J2EE para ejecutar las aplicaciones AON. Estas, al estar desarrolladas sobre estándares, son totalmente portable entre diferentes servidores de aplicaciones, destacando entre los mejores los de Bea WebLogic, IBM WebSphere, Oracle AS, y Jboss AS.

Sistema operativo

Los Servidores de Aplicaciones elegidos funcionan tanto sobre sistema operativo Windows como sobre sistemas Unix, incluyendo Linux, cuyo carácter gratuito es bien conocido junto con su aceptación en el mundo empresarial desde que corporaciones como IBM han apostado por colaborar en su desarrollo. Por lo que la decisión del Sistema Operativo requerido para ejecutar las aplicaciones AON dependerá de los requerimientos del cliente en cada caso (existencia de servidores en una u otra plataforma, imposición de plataforma por política de empresa o selección de la plataforma y correspondiente sistema operativo que cumplan con las expectativas de escalabilidad, rendimiento y estabilidad requeridas).

Sistemas gestores de base de datos

Las Aplicaciones AON son totalmente independientes del Sistema Gestor de Base de datos a utilizar. Mediante el uso de soluciones ORM como Hibernate e internamente a éstas, el uso de conectores JDBC, las Aplicaciones pueden trabajar con cualquier base de datos estándar de mercado: Oracle, Informix, DB2, Microsoft SQL, MySQL, MaxDB, etc...

Usabilidad

La norma ISO 9241:11 de 1993 define la usabilidad como “la facilidad de uso de una aplicación informática”. Por lo que respecta a la Web, la usabilidad contempla un conjunto de técnicas que ayudan a realizar tareas en el entorno gráfico de la interfaz de usuario de una aplicación Web.

Abarca aspectos como la navegabilidad, la facilidad de encontrar un elemento dado (relacionado con la arquitectura de la información), el número de errores del sistema, la legibilidad de los textos (correspondiente al diseño Web), la correcta transmisión de la información, etc.

La usabilidad es la responsable última de que un sitio o aplicación Web cumpla de forma correcta las expectativas con las que ha sido creado, siempre desde el punto de vista del usuario final. Para que una aplicación Web pueda ser considerada “usable” deberá reunir una serie de características, entre las que podemos destacar:

- Debe ser fácil de entender y aprender para el usuario final.
- Debe ser intuitiva, ofreciendo al usuario cada cosa donde éste espera que esté, de tal forma que la navegación por ella y la realización de tareas se produzca con una sucesión de continuidad lógica y de forma sencilla.
- Debe ser rápida: pronta visualización de las páginas, y pocos pasos para realizar las tareas.
- Debe estar libre de errores o informar qué tipo de error se ha producido y por qué.
- Debe ser estéticamente agradable, diseñada con una gama de colores adecuada, que proporcione un entorno de trabajo visualmente relajado y bello.



Entre los beneficios más evidentes que conlleva la aplicación de la ingeniería de la usabilidad al diseño y desarrollo de un sitio o aplicación Web podemos citar:

- Reducción del tiempo final de desarrollo, evitando procesos de rediseño del sitio, originados por una mala concepción inicial del mismo.
- Reducción de los costes de mantenimiento del sitio Web, al ofrecer al público un producto homogéneo y funcional desde el principio.
- Obtención de sistemas más fáciles de entender, usar y recordar, con lo que se aumenta la productividad de los usuarios y la eficiencia de los procesos.

ARQUITECTURA DE CAPAS

Objetivo

El objetivo principal de la arquitectura es separar las distintas capas de desarrollo, prestando especial atención a la consecución de un modelo fácil de mantener y evolucionar, así como disponer de un despliegue sencillo de aplicaciones y el empleo de tecnologías punteras actuales.

Se desea una arquitectura que permita trabajar en capas y que sirva tanto para las aplicaciones en la intranet como en Internet, así como disponer de la flexibilidad necesaria para poder emplear un cliente ligero (navegador Web, Wap) o un cliente pesado (Swing, SWT, etc.). Una premisa fundamental es no tener que rescribir ningún código y que las capas comunes sean reutilizadas sin cambios en ambos casos.



Patrón MVC

Para lograr este objetivo se elige el patrón de arquitectura de software MVC (Modelo-Vista-Controlador) que permite separar los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones Web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el controlador es el Sistema de Gestión de Base de Datos y el modelo es el modelo de datos.

Modelo

Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos; por ejemplo, no permitiendo comprar un número de unidades negativo, calculando si hoy es el cumpleaños del usuario o los totales, impuestos o portes en un carrito de la compra.

Vista

Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

Controlador

Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo que sigue el control generalmente es el siguiente:

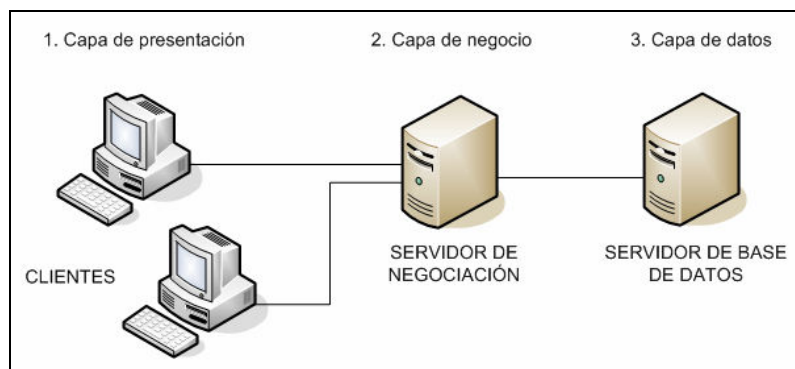
1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón o un enlace)
2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.
3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.
4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, el patrón de observador puede ser utilizado para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice.
5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.



Programación por Capas

Las aplicaciones AON cumplen con la estructura de arquitectura de capas. La programación por capas es un estilo de programación en la que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño.

La ventaja principal de este estilo de programación, es que el desarrollo se puede llevar a cabo en varias capas y en caso de algún cambio sólo se ataca la capa requerida sin tener que revisar todo el código mezclado. El diseño más utilizado actualmente es el diseño en tres capas.



Capa de presentación

La capa de presentación es lo que ve el usuario, es quien comunica la información al usuario e interactúa con él para la captura de datos, implementando un mínimo de procesos (Ej.: chequeo de formato de datos) y comunicándose únicamente con la capa de negocio.

Dado que la capa de presentación de las Aplicaciones AON es Web, el usuario necesita únicamente un navegador Internet para acceder a las aplicaciones. Para el usuario, manejar las aplicaciones es igual de fácil que navegar por un sitio Web. Asimismo para el administrador de las aplicaciones, es mucho más fácil mantener un único sitio donde actualizar las aplicaciones, usuarios y perfiles y despreocuparse de tener que actualizar los equipos clientes a las nuevas versiones de la aplicación. Los usuarios siempre accederán a la versión actualizada de cada aplicación al conectarse mediante el navegador Web a la URL de cada aplicación.



Para la capa de presentación (la vista) se busca un framework que proporcione una mayor facilidad en la elaboración de pantallas, mapeo entre los formularios y sus clases en el servidor, la validación, conversión, gestión de errores, traducción a diversos idiomas y, de ser posible, que facilite también el incluir componentes complejos (menús, árboles, AJAX, etc.) de una forma sencilla y –sobre todo– fácil de mantener. Para esta capa se ha elegido *JavaServer Faces*.

Capa de negocio

Es donde residen los programas que se ejecutan, recibiendo las peticiones del usuario y enviando las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) pues es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos.

En la capa de negocio y persistencia, se decide desde el primer momento no emplear EJB's por su elevado coste de desarrollo y mantenimiento así como su falta de flexibilidad y coste en tiempo para los cambios. Se opta por una solución basada en *Manager Beans*, concepto proporcionado por la *Plataforma AON*, que trabajaban contra un modelo de dominio

limpio. La persistencia de las clases se sustenta en DAO's (Objetos de Acceso a Datos), manteniendo aislada la capa de persistencia de la capa de negocio.

Tanto los *Manager Beans*, como los DAO's, así como el propio modelo son realmente POJO's (clases simples de Java), con la simplicidad que conllevan y sin dependencias reales con ningún framework concreto. Toda esta integración es parte del desarrollo de la Plataforma AON.

Capa de datos

La capa de datos está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Para la capa de persistencia se piensa en utilizar alguna herramienta ya existente, que permita realizar el mapeo objeto-relacional de una forma cómoda pero potente, sin tener que implementarlo directamente mediante JDBC. Esto último conlleva, por ejemplo, un esfuerzo importante en un caso de cambio de base de datos (como suele ocurrir), en la gestión de la caché, la utilización de carga perezosa (*lazy load*), etc. La herramienta elegida finalmente fue Hibernate.

Todas estas capas pueden residir en un único servidor, si bien lo más usual es que haya una multitud de ordenadores donde reside la capa de presentación (son los clientes de la arquitectura cliente/servidor). Las capas de negocio y de datos pueden residir en el mismo servidor, y si el crecimiento de las necesidades lo aconseja se pueden separar en dos o mas servidores los cuales recibirán las peticiones del ordenador en que resida la capa de negocio.

Si por el contrario fuese la complejidad en la capa de negocio lo que obligase a la separación, esta capa de negocio podría residir en uno o más servidores que realizarían solicitudes a una única base de datos. En sistemas muy complejos se llega a tener una serie de servidores sobre los cuales corre la capa de datos, y otra serie de servidores sobre los cuales corre la base de datos.

En una arquitectura de tres niveles, los términos "capas" y "niveles" no significan lo mismo. El término "capa" hace referencia a la forma como una solución es segmentada desde el punto de vista lógico: Presentación/ Lógica de Negocio/ Datos. En cambio, el término "nivel", corresponde a la forma en que las capas lógicas se encuentran distribuidas de forma física. Por ejemplo:

- Una solución de tres capas (presentación, lógica, datos) que residen en un solo ordenador (Presentación+lógica+datos). Se dice, que la arquitectura de la solución es de tres capas y un nivel.



- Una solución de tres capas (presentación, lógica, datos) que residen en dos ordenadores (presentación+lógica, lógica+datos). Se dice que la arquitectura de la solución es de tres capas y dos niveles.
- Una solución de tres capas (presentación, lógica, datos) que residen en tres ordenadores (presentación, lógica, datos). La arquitectura que la define es: solución de tres capas y tres niveles.

Seguridad

Para la seguridad se busca más potencia que la actualmente definida para los contenedores Web que, aunque es ampliada por cada fabricante, se vuelve dependiente del mismo. Se busca una solución que sea totalmente independiente de cada contenedor concreto, común a todos ellos y que ofrezca una gran versatilidad (posibilidad de autenticación contra sistemas LDAP o base de datos, control del número de sesiones del usuario, etc.). Para realizar esta integración se desarrolló el módulo *aon-security* dentro de la Plataforma AON.

Aon-security implementa la seguridad en lo referente a la autenticación y control de sesiones de usuario basándose en una gestión de usuarios y perfiles que a su vez están asociados a una serie de roles definidos en cada una de las aplicaciones AON desplegadas en el mismo servidor de aplicaciones. Este módulo permite la centralización de la gestión de usuarios a la vez que gestiona la seguridad de las aplicaciones AON, las cuales delegan en él este apartado.



Beneficios de la arquitectura diseñada.

La primera ventaja se deriva de la modularidad del diseño. Cada una de las partes empleadas (JSF para la vista, AON para la integración y seguridad, Hibernate para la persistencia) es intercambiable de forma sencilla y limpia por otras soluciones disponibles. Por ejemplo, para la vista se emplea JavaServer Faces, pero nada impide emplear también una aplicación de escritorio mediante Swing o SWT sin tener que tocar ni una sola línea de código de las restantes capas.

Es más, nada impediría que se pudiese disponer de una aplicación con una parte de la capa de presentación en JSF y otra parte, para otro tipo de usuarios, en Swing, ambas funcionando a la vez y compartiendo todo el resto del código (lógica de negocio, persistencia, integración, seguridad, etc.).

De igual forma, si se desean cambiar elementos de la capa de persistencia empleando otro framework para el mapeo diferente de Hibernate –o sencillamente no utilizar ninguno- tan sólo serían necesarios cambios en esa capa.

De la misma manera se podrían sustituir cualquiera de las otras capas. El diseño se ha hecho reduciendo al mínimo posible las dependencias entre ellas.

FRAMEWORKS EMPLEADOS.

Plataforma AON.

La plataforma AON consiste en una serie de librerías JAVA destinadas a facilitar las labores de desarrollo más comunes en la generación de aplicaciones de gestión y a enlazar con los diferentes frameworks existentes en el mercado. De esta forma la lógica de la aplicación no depende de ningún framework en concreto, permitiendo el cambio de cualquiera de ellos con una relativa facilidad.

Las librerías integrantes en la plataforma AON se han desarrollado basadas en la experiencia adquirida en la utilización de otros frameworks, adoptando lo mejor de cada uno de ellos y completando sus carencias, de forma que resulta en framework vivo, actualizado continuamente para dar soporte a las nuevas tecnologías que emergen cada día.

JavaServer Faces.

JavaServer Faces (JSF) es el estándar presentado por Sun para la capa de presentación Web. Forma parte de la especificación J2EE 5 -que deberán cumplir todos los servidores de aplicaciones- y se erige como una evolución natural de los frameworks actuales hacia un sistema de componentes.

Es un estándar sencillo que aporta los componentes básicos de las páginas Web además de permitir crear componentes más complejos (menús, pestañas, árboles, etc.). Ya hay disponibles diferentes implementaciones de la especificación, tanto comerciales como de código abierto, así como librerías de componentes adicionales que amplían la funcionalidad de esos componentes iniciales.

JSF ha sido acogida por la comunidad como “el framework que hacía falta”. Muchos de los proyectos de código abierto y las compañías con más influencia lo han identificado como el framework de presentación Web del futuro. JBoss ha integrado directamente JSF con EJB3 mediante el proyecto Seam (abanderado además por Gavin King, el líder del equipo de desarrollo Hibernate). IBM lo ha incorporado como mecanismo de presentación estándar para su entorno, desarrollando no sólo el soporte completo en su IDE, sino nuevos componentes. Oracle es otra de las compañías que más ha apostado por esta tecnología, ofreciendo la que posiblemente sea la mayor oferta de componentes propios.

Dentro de los “pesos pesados” en Java, nombres que han guiado a la industria en el pasado como Matt Raible, Rick Hightower, David Geary, el mencionado Gavin King y por supuesto el propio creador de Struts y ahora arquitecto de JSF, Craig McClanahan.

MyFaces.

MyFaces es un proyecto de la fundación Apache que ofrece una implementación en código abierto de JavaServer Faces, así como un amplio conjunto de componentes



adicionales. Entre ellos se dispone de un menú, árboles, pestañas, componentes para gestionar el estado de los diálogos, etc. A lo largo del tiempo se van incorporando componentes cada vez más potentes y ya se encuentran en pruebas componentes basados en Ajax (Asynchronous JavaScript And XML).

La tecnología Ajax es una técnica de desarrollo Web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente, es decir, en el navegador del usuario, y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad en la misma.

Realmente el uso de MyFaces no impide el uso de otras librerías de componentes, así que su uso depende tan solo de la utilidad que encontremos en la funcionalidad que ofrece.

Hibernate

Hibernate es un motor de persistencia de código abierto. Permite mapear un modelo de clases a un modelo relacional sin imponer ningún tipo de restricción en ambos diseños. Cuenta con una amplia documentación, tanto a nivel de libros publicados como disponibles gratuitamente en su Web. A nivel comercial está respaldado por JBoss, que proporciona servicios de soporte, consultoría y formación en el mismo.

Una característica de la filosofía de diseño de Hibernate ha de ser destacada especialmente, dada su gran importancia: puede utilizar los objetos Java definidos por el usuario tal cual, es decir, no utiliza técnicas como generación de código a partir de descriptores del modelos de datos o manipulación de bytecodes en tiempo de compilación, ni obliga a implementar interfaces determinados, ni heredar de una superclase. Utiliza en vez de ello el mecanismo de reflexión de Java. Es esta característica la que permite un modelado iterativo fluido y natural basado en UML, un factor fundamental para lograr un trabajo ágil y productivo. Además abre las puertas a utilizar herencia en el modelo de datos (esta opción estaría limitada si una herramienta nos obliga a que los objetos de datos hereden de una superclase por no soportar Java herencia múltiple).

Actualmente es el estándar indiscutible en lo referente a la gestión de persistencia de datos. Desde su versión 1.0, el motor no ha parado de evolucionar, incorporando todas las nuevas ideas que se iban incorporando en este campo.

Hoy en día, en su versión 3.2, ya soporta el estándar EJB 3 (su autor es uno de los principales integrantes del JCP que está definiendo esta especificación) por lo que ya se puede elegir desarrollar aplicaciones empleando EJB 3 -que correrán en cualquier contenedor de EJB's que soporte J2EE 5- o aplicaciones independientes. Esto no solo blinda la inversión de tiempo en Hibernate de cara al futuro, sino que, nos hace totalmente independientes del mismo.



JasperReports

JasperReports es una poderosa librería Java que permite generar reportes en diferentes formatos (PDF, HTML, CSV, XML, XLS, RTF) a partir de objetos java (JavaBeans), conexiones a bases de datos, etc. La generación de listados se ve simplificada con la utilización del modulo aon-ui-report de la Plataforma AON, la cual facilita enormemente la integración de esta herramienta con la lógica de la aplicación.

JasperReports trabaja en forma similar a un compilador y a un intérprete. El usuario diseña el reporte codificándolo en XML de acuerdo a las etiquetas y atributos definidos en un archivo llamado jasperreports.dtd (parte de JasperReports). Usando XML el usuario define completamente el reporte, describiendo donde colocar texto, imágenes, líneas, rectángulos, cómo adquirir los datos, como realizar ciertos cálculos para mostrar totales, etc. Este archivo fuente XML debe ser compilado para obtener un reporte real. La versión compilada del fuente es nombrada "archivo jasper" y el resultado se obtendrá tras cruzar los datos con el fichero jasper, generando el resultado dinámicamente en el formato definido.

Facelets

Facelets es un sistema simplificado de presentación, en donde es posible diseñar de forma libre una página web para posteriormente asociarle los componentes JSF precisos. Aporta mayor libertad al diseñador y mejora los informes de errores que tiene JSF entre otras cosas.



CÓDIGO ABIERTO (*OPEN SOURCE*).

Las aplicaciones AON utilizan librerías de código abierto. Código abierto (del inglés *open source*) es el término por el que se conoce el software distribuido y desarrollado libremente. Este término empezaron a utilizarlo en 1998 algunos usuarios de la comunidad del software libre, tratando de usarlo como reemplazo al ambiguo nombre original en inglés del software libre (*free software*).

La filosofía del *Open Source* centra su atención en la premisa de que al compartir el código, el programa resultante tiende a ser de calidad superior al software propietario, es una visión meramente técnica.









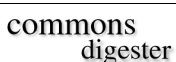
El movimiento Open Source tiene un decálogo que debe cumplir un código para poder llamarse "Open Source", estas son:

1. Libre redistribución: el software debe poder ser regalado o vendido libremente.
2. Código fuente: el código fuente debe estar incluido u obtenerse libremente.
3. Trabajos derivados: la redistribución de modificaciones debe estar permitida.
4. Integridad del código fuente del autor: las licencias pueden requerir que las modificaciones sean redistribuidas sólo como parches.
5. Sin discriminación de personas o grupos: nadie puede dejarse fuera.
6. Sin discriminación de áreas de iniciativa: los usuarios comerciales no pueden ser excluidos.




7. Distribución de la licencia: deben aplicarse los mismos derechos a todo el que reciba el programa
8. La licencia no debe ser específica de un producto: el programa no puede licenciarse solo como parte de una distribución mayor.
9. La licencia no debe restringir otro software: la licencia no puede obligar a que algún otro software que sea distribuido con el software abierto deba también ser de código abierto.
10. La licencia debe ser tecnológicamente neutral: no debe requerirse la aceptación de la licencia por medio de un acceso por clic de ratón o de otra forma específica del medio de soporte del software.

Dependencia con librerías externas.

A continuación se detallan las librerías con las que existe una dependencia:

	Descripción	Lic.
	Aon FrameWork es un conjunto de librerías que facilitan la programación e integración de aplicaciones de gestión.	
	ANTLR , es una herramienta de lenguaje que provee un framework para construir analizadores sintácticos, compiladores y traductores a partir de unas descripciones gramaticales.	BSD
	ASM es un framework para la manipulación de los "Java bytecodes".	BSD
	BATIK permite la manipulación de archivos SVG.	Apache 2.0
	CGLIB es una potente librería para la generación de código. Es usada para extender clases e implementar interfaces JAVA en tiempo de ejecución.	Apache 2.0
	COMMONS BEANUTILS permite de una forma sencilla, la introspección de clases JAVA.	Apache 2.0
	COMMONS CODEC provee utilidades para codificadores y decodificadores.	Apache 2.0
	COMMONS COLLECTIONS construye sobre el Java Collections Framework del JDK nuevas implementaciones y utilidades.	Apache 2.0
	COMMONS DIGESTER permite leer de una manera sencilla archivos de configuración XML.	Apache 2.0

	Descripción	Lic.
	COMMONS FILEUPLOAD provee una manera sencilla y robusta de subir archivos a servlets y aplicaciones Web.	Apache 2.0
	COMMONS JAVAFLOW permite mantener estados dentro de los flujos de los procesos.	Apache 2.0
	COMMONS LANG provee utilidades para la API <code>java.lang</code> , principalmente métodos para la manipulación de <code>String</code> .	Apache 2.0
	COMMONS LOGGING ejerce de puente entre los diferentes paquetes de <code>log</code> existentes.	Apache 2.0
	DOM4J es una librería fácil de utilizar para trabajar con XML, <i>XPath</i> y <i>XSLT</i> en Java usando el "Java Collections Framework" y con pleno soporte para <i>DOM</i> , <i>SAX</i> y <i>JAXP</i> .	BSD
	EHCACHE provee una cache en tiempo de ejecución. Utilizada por Hibernate.	Apache 1.1
	FOP es un procesador de "Formating Objects" (XSL-FO)-	Apache 2.0
	HIBERNATE es una herramienta para hacer persistentes objetos en bases de datos.	LGPL
	iCal4j es una librería utilizada para leer y escribir objetos iCalendar definidos según la especificación RFC2445 .	licencia
	iText es una librería para la generación de documentos PDF.	LGPL
	JasperReports es una potente herramienta para la generación de listados en formato PDF, XLS, CSV, HTML y XML.	LGPL
	Librerías encargadas de manejar los layout de las vistas.	CDDL
	Apache MyFaces es una implementación de las JavaServer Faces.	Apache 2.0
	JTA especifica unos interfaces estándar de JAVA para que puedan interactuar un manejador de transacciones y un sistema de transacciones distribuido.	CDDL

	Descripción	Lic.
	Velocity es una herramienta de generación de documentos en base a unas plantillas.	Apache 2.0
	Generic Tools son una serie de utilidades para el manejo de Velocity.	Apache 2.0
	Utilidades para manipular ficheros con formatos de Microsoft.	Apache 2.0