UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 4

Título: “ArBaWeb: ARQUITECTURA BASE SOBRE LA WEB”

TRABAJO PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

AUTORES:
Luis Alberto Pimentel González
Iósev Pérez Rivero

TUTOR:
Ing. Madelín Haro Pérez

Ciudad de la Habana, junio del 2007

“La actitud frente a la vida es mostrar con el ejemplo el camino a seguir, el llevar a las masas con el propio ejemplo cualesquiera que sean las dificultades a vencer en el camino, quien pueda mostrar el ejemplo de su trabajo repetido durante días sin esperar de la sociedad otra cosa que el reconocimiento a sus méritos, de constructor de esta nueva sociedad, tiene derecho a exigir a la hora del sacrificio.”

Che

Declaración de Autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad 4 y a la Universidad de las Ciencias Informáticas para que haga el uso que estime pertinente con este trabajo.

Para que así conste firmamos la presente a los ___ días de mes de junio del 2007

_____	_____
Iosev Pérez Rivero	Luis Alberto Pimentel González

_____

Ing. Madelín Haro Pérez

DEDICATORIA

A mi madre Maritza que me ha apoyado, por su esfuerzo para que yo sea alguien en la vida y por su amor y dedicación que me ha brindado siempre.
A mis abuelos Elisa y Enrique que me han dado siempre su amor y cariño.
A Maria que con su amor y cariño ha sido mi segunda madre apoyándome en las buenas y en las malas.
A mi hermanita Elizabeth que es la niña de mis ojos.
A mi hermanito Yasel que para que sigua mi ejemplo.
A mi tío Jorge y a mi tía Rosy que siempre me han ayudado para que siga adelante.
A Ricardo que se ha ganado mi cariño y respeto.
A mi padre Juan.

Y a todos mis amigos que siempre han creído en mí y me han querido incondicionalmente.

Iósev

A mi mamá y a mi papá por el amor con que me han enseñado a caminar en la vida.
A mi hermanito por demostrarme que no hay barrera que la voluntad no rompa.
A mi novia por apoyarme tanto y darme la fuerza para seguir adelante en todas las situaciones difíciles.
A mi abuela Candy por enseñarme a confiar en mí.
A mis tías, mi abuelo, abuelas, primos, en fin, a toda mi familia a quienes le debo muchísimo de mi vida.
A todos mis amigos por su apoyo incondicional.
A mis profesores por formarme como profesional.

Y a la Revolución por darme la posibilidad de cumplir mis sueños.

Luis A. Pimentel

AGRADECIMIENTOS

Le agradecemos a todas las personas que hicieron posible la realización de este trabajo, en especial a nuestra tutora Madelín por su alta entrega y paciencia.

RESUMEN

Cada vez son más los proyectos que se desarrollan sobre la plataforma Java Enterprise Edition (JEE1) en la Universidad de las Ciencias Informáticas. Hasta el momento no hay suficiente experiencia en el rol de arquitecto de JEE, lo que conlleva que las arquitecturas utilizadas en estos proyectos no se integren fácilmente con componentes ya implementados

Como propuesta de solución a este problema, en este trabajo se elabora una arquitectura de software de dominio específico para desarrollar aplicaciones Web sobre JEE la cual se nombra ArBaWeb. Como parte de la misma se desarrolló una arquitectura como línea base del desarrollo de software, un framework que responde a los requerimientos específicos de esas aplicaciones, se definió un flujo de trabajo para guiar la construcción del software y se elaboró una propuesta de ambiente de desarrollo.

[1] JEE: La versión empresarial de Java después de J2EE 1.4 es llamada Java EE 5.0; destacando así los cambios significantes de los framework de peso ligero traídos en los estándares empresariales de Java.

ÍNDICE

INTRODUCCIÓN

En la actualidad los sistemas de software son más complejos, "los problemas de diseño van más allá de los algoritmos y de las estructuras de datos para su computación: para diseñar y especificar una estructura del sistema global surgen nuevos tipos de problemas". Se necesita, por lo tanto, de una arquitectura de software que permita comprender el sistema, organizar el desarrollo, fomentar la reutilización y hacerlo evolucionar.

La Universidad de las Ciencias Informáticas (UCI) se adentra cada vez más en la producción de software empresarial y con ello se incrementa el número de contratos donde algunos de estos requieren soluciones web sobre JEE. Cada proyecto de este tipo que se inicia debe estar sostenido por una arquitectura y un flujo de trabajo que le brinde la solución tecnológica en un marco de tiempo aceptable. Actualmente no se cuenta con ninguna arquitectura base ni con un flujo de trabajo que sea común para todos los proyectos de estas características usando tecnologías de código abierto que permitan dar solución inmediata a los requerimientos de cualquier proceso empresarial a informatizar. En estos momentos este tipo de proyecto se realiza en la universidad utilizando las mismas tecnologías (Spring Framework, Hibernate, etcétera.) y herramientas (Eclipse, Hibernate Tools, WTP, etcétera.) pero con arquitecturas y flujos de trabajos variados; esto provoca que se desarrollen una y otra vez componentes para cubrir requerimientos comunes en lugar de reutilizar componentes integrados a una arquitectura base que satisfagan estas funcionalidades. Los patrones de diseño, que representan soluciones simples y elegantes a problemas comunes, deben estar presentes en toda arquitectura: Sin embargo, en muchos casos no se hace el mejor uso de estos trayendo como consecuencia la no reutilización de la experiencia acumulada por los desarrolladores.

Dada esta situación surge el siguiente problema:

¿Cómo lograr en la UCI la reutilización de componentes con un desarrollo regido por estándares que utilicen buenas prácticas de programación y patrones de diseño en el desarrollo de aplicaciones web sobre JEE?

Teniendo en cuenta el problema planteado se define como **objeto de estudio:** *Las arquitecturas para desarrollar aplicaciones web sobre JEE.*

Por lo que se especifica el siguiente **campo de acción:**

Las arquitecturas de software de dominio específico en el desarrollo de aplicaciones web sobre JEE.

En la UCI se están construyendo aplicaciones web en JEE que utilizan arquitecturas bases diferentes. Esto trae como resultado que no se pueda reutilizar componentes y el tiempo de desarrollo sea más largo por tener que construir componentes integrados a esas arquitecturas bases para satisfacer requerimientos comunes entre estos proyectos. De esta manera, si se utiliza una única arquitectura base en la UCI para el desarrollo de aplicaciones web en JEE entonces los proyectos que utilicen estas características podrán reutilizar código garantizando el uso de las buenas prácticas de programación y patrones de diseño y minimizan el tiempo de desarrollo del software.

Dadas estas condiciones se plantea lograr como **objetivo general**: Definir e implementar una arquitectura de software de dominio específico (DSSA), que permita a los proyectos web sobre JEE agilizar el tiempo de desarrollo y reutilizar componentes integrados a dicha arquitectura.

Según los objetivos planteados para la investigación se define el siguiente conjunto de tareas para lograr su cumplimiento:

- Desarrollar una propuesta de diseño de las capas lógicas de la aplicación que cumpla con buenas prácticas y patrones de diseño.
- Definir estándares de codificación para el desarrollo usando este framework arquitectónico.
- Definir propuestas de estructuras de paquetes y recursos dentro de la arquitectura para las distintas complejidades (baja, mediana y alta) que posea una aplicación.
- Definir propuestas de mecanismos de colaboración entre módulos y subsistemas.
- Desarrollar un framework arquitectónico, con componentes y funcionalidades reusables que intervengan en las capas lógicas de una aplicación, así como el soporte para la posible integración de nuevos componentes.
- Definir un flujo de trabajo que organice el desarrollo de la aplicación de forma paralela en cada una de las capas lógicas.
- Definir un ambiente de desarrollo adecuado para la construcción del proyecto de software y para el uso del framework.

El presente trabajo se estructura en tres capítulos:

- En el primer capítulo, Fundamentos Teóricos, hace alusión a las herramientas y tecnologías propuestas en el ambiente de desarrollo definido, a la arquitectura de software y a la arquitectura de software de dominio específico. De esta última se detallan cuáles elementos la componen.
- En el segundo capítulo, ArBaWeb: Dominio y arquitectura, se definen el dominio, los requerimientos de referencia y la arquitectura base de ArBaWeb. Se desglosa la arquitectura base en: diseño de las capas lógicas, convenciones de código y nombre de recursos, estructuras de código y mecanismos de colaboración.
- En el tercer capítulo, ArBaWeb, fundamentos para desarrollar, se describen los demás elementos de ArBaWeb como una arquitectura de software de dominio especifico, tales como ArBaWeb Framework, el flujo de trabajo y el ambiente de desarrollo. Se especifican los resultados que se han obtenido con la introducción de este trabajo en varios proyectos de la universidad.

Capítulo 1. Fundamentación Teórica

3.1 Introducción

En este capítulo se realiza un estudio del marco teórico sobre *las arquitecturas para desarrollar aplicaciones web sobre JEE, partiendo de los principales conceptos para lograr su entendimiento. Se caracterizan las* herramientas y tecnologías que se analizaron para ser utilizadas en la confección del software y se decide qué

3.2 Arquitectura de Software

Con el surgimiento de la programación se dio comienzo a un nuevo universo: la producción de software. A medida que los mismos fueron ganando en complejidad, dada la necesidad de resolver problemas más complicados y abarcadores se identificaron propiedades comunes acompañadas de soluciones muy similares en su estructura. En el año 1968, Edsger Dijkstra de la Universidad Tecnológica en Holanda, propuso que se hiciera una estructuración correcta de los sistemas de software antes de lanzarse a programar. Esto conforma básicamente el concepto de lo que hoy se conoce como Arquitectura de Software.

Existen muchas definiciones sobre arquitectura de software. Perry y Wolf en 1992 la definieron como un conjunto de elementos que tienen una forma común. Garlan and Shaw en 1994 y 1996 la conceptualizaron como una colección de componentes y conectores unidos con una descripción de la interacción entre esos componentes y conectores. Bass, Clements, y Kazman en 1998 la definen como la estructura de las estructuras de un sistema, la cual comprende componentes de software, las propiedades visibles externas de estos componentes, y las relaciones entre ellos. La definición “oficial” de ARQUITECTURA DE SOFTWARE se ha acordado que sea la que brinda el documento de IEEE Std 1471-2000 que se formula así: “La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución. (KAISER 2005)

¿Por qué es importante una arquitectura de software? Independientemente a la variedad de definiciones literarias se converge en la gran importancia que representa la arquitectura en el desarrollo de cualquier tipo de software pues representa las bases sobre las que se desarrollará el sistema. Brinda una estructura

que soporta las soluciones a cada tipo de problema que pueda aparecer en el desarrollo. Define la organización e interacción entre los distintos componentes del mismo. Asegura que los requerimientos más importantes puedan ser evaluados e implementados. Una arquitectura de software también permite flexibilidad en el sistema pues facilita la ejecución de futuros cambios. Promueve la reutilización de componentes existentes como librerías de clases, sistemas legados y de aplicaciones de terceros.

La arquitectura de software provee una descripción de alto nivel de los subsistemas que comprenden la arquitectura del sistema. Está atada a comprender cómo las mayores operaciones del negocio son soportadas. Ayuda a la planificación de recursos y la asignación de tareas, debido a que el trabajo de desarrollo puede ser particionado a través de los subsistemas y los esfuerzos de desarrollo individual pueden proceder en paralelo. Cuando los equipos de desarrollo están geográficamente dispersados puede minimizar el número de interacciones requeridas.

1.2.1 Estilos Arquitectónicos

Desde sus inicios, en la arquitectura de software, se observó que en la práctica del diseño y la implementación ciertas regularidades de configuración aparecían una y otra vez como respuesta a similares demandas. El número de esas formas no parecía ser muy grande. Muy pronto se las llamó estilos, por analogía con el uso del término en arquitectura de edificios. Un estilo describe entonces una clase de arquitectura, o piezas identificables de las arquitecturas empíricamente dadas.

Un estilo arquitectónico es un conjunto coordinado de restricciones arquitectónicas que restringe los roles o rasgos de los elementos arquitectónicos y las relaciones permitidas entre esos elementos dentro de la arquitectura que se conforma a ese estilo.

Existe una gran variedad de estilos arquitectónicos las cuales tienen diversas clasificaciones entre las que se encuentran:

- Estilos de flujo de datos
- Estilos centrados en datos
- Estilos de llamada y retorno
- Estilos de código móvil
- Estilos heterogéneos

- Estilos peer-to-peer

Arquitectura Cliente-Servidor

Esta arquitectura se encuentra dentro de la clasificación de estilo de llamada y retorno. El cliente y el servidor generalmente están localizados en diferentes sistemas, sin embargo pueden encontrarse en el mismo sistema. El cliente es la entidad que hace la petición por un servicio. El servidor es la entidad que provee el servicio correspondiente a la petición. El servicio debe procurar el resultado, el cual es retornado al cliente (Ver Figura 1-1).

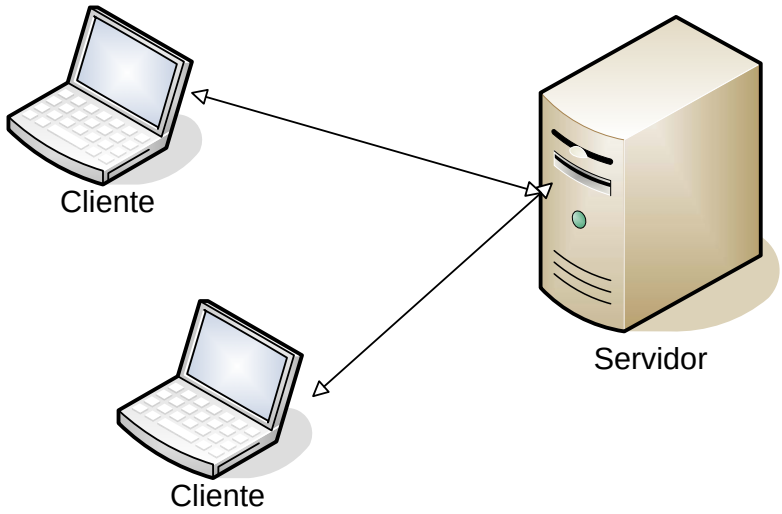

Figura 1-1 Arquitectura cliente-servidor

Los clientes pueden ser clasificados como clientes “flacos” y/o “gordos”. Lo clientes gordos típicamente contienen, además de la lógica de presentación, gran parte de la lógica de negocio de la aplicación. Lo clientes flacos manejan usualmente sólo la lógica de presentación lo que adiciona grandes ventajas como permitir que futuros cambios de negocio en la aplicación no afecten al cliente.

Arquitectura en capas

Garlan y Shaw definen el estilo en capas como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior.

El estilo soporta un diseño basado en niveles de abstracción crecientes lo cual a su vez permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales. Permite realizar optimizaciones y refinamientos enfocando los cambios en un solo lugar. Proporciona amplia reutilización dada la división bien definida de responsabilidades. Al igual que los tipos de datos abstractos, se pueden utilizar diferentes implementaciones o versiones de una misma capa en la medida que soporten las mismas interfaces de cara a las capas adyacentes. Esto conduce a la posibilidad de definir interfaces de capa estándar, a partir de las cuales se pueden construir extensiones o prestaciones específicas.

Una especialización muy usada de la arquitectura en capas es la arquitectura de tres capas donde se observan muy bien delimitadas las responsabilidades de cada funcionalidad en la aplicación.

En la figura 1-2 se ejemplifica una arquitectura de tres capas, donde cada capa está muy bien delimitada de las demás. Una capa superior interactúa con una capa inferior mediante interfaces que definen las funcionalidades que la misma debe brindar. Las capas de la aplicación pueden residir tanto en el mismo nodo físico como en nodos separados.

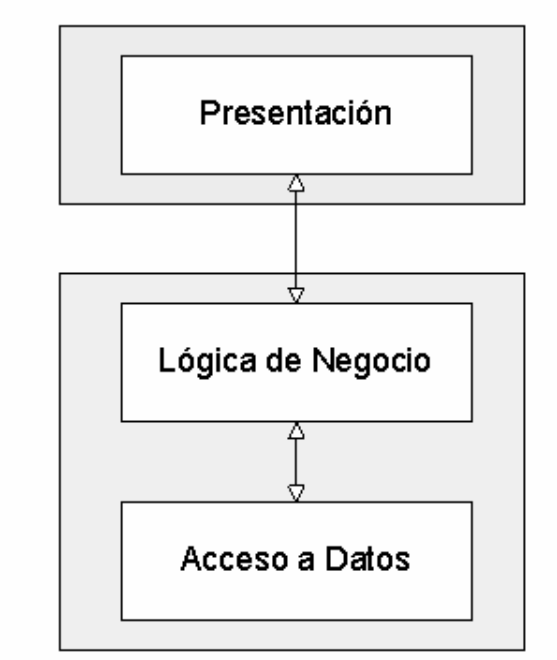

Figura 1-2: Ejemplo de una arquitectura de tres capas.

1.3.1 Arquitectura de Software de Dominio Específico

En la década de los años 80, DARPA[2] afrontó un fuerte problema desarrollando un gran sistema de software para la defensa, con un acercamiento llamado Arquitectura de Software de Dominio Específico (DSSA[3]). La motivación y fundamento para este programa fue basado en las siguientes suposiciones (KAISER 2005):

- El desarrollo sobre dominios específicos puede ser optimizado.
- La reutilización de módulos es más probable si estos se obtienen de otras aplicaciones dentro del mismo dominio.

El objetivo principal fue reutilizar tantos objetos como fuera posible; además de minimizar la intención de resolver diferentes tipos de problemas antes de reinventar una solución para cada problema nuevo.

¿Qué es un dominio?

Un dominio es un área de interés, usualmente representando un espacio del problema que es un subconjunto de una o más disciplinas. Por ejemplo, la investigación del ácido desoxirribonucleico (ADN) es un dominio. (KAISER 2005).

Por tanto, otro dominio puede ser las aplicaciones web sobre JEE que utilicen Spring Framework e Hibernate. Cuando se construye una aplicación, se está desarrollando software para un espacio del problema, es decir, un conjunto de problemas a ser resueltos dentro del dominio.

En fin, un dominio es definido por un conjunto de problemas o funciones comunes que las aplicaciones en ese dominio pueden resolver o hacer.

La idea básica de las DSSA es hacer práctica la reutilización de software, enfocándose en los problemas específicos de dominios de software y desarrollando soluciones basadas en componentes para estos dominios de problemas. Estos componentes son genéricos y necesitan ser configurados en el momento que la aplicación es implementada.

DARPA definió 4 pasos claves para las DSSA(KAISER 2005) (KAISER 2005):

[2] DARPA: Agencia de Investigación de Proyectos Avanzados de Defensa (DARPA por sus siglas en inglés) es una agencia del Departamento de Defensa de los Estados Unidos responsable del desarrollo de nuevas tecnologías para uso militar.
[3] DSSA: siglas en inglés de Arquitectura de Software de Dominio Específico (Domain-Specific Software Architecture).

- Un modelo del dominio es desarrollado para establecer una terminología estándar y una semántica común para el dominio del problema.
- Un conjunto de requerimientos de referencia para todas las aplicaciones dentro del dominio del problema que es desarrollado.
- Una arquitectura de referencia es definida para una solución genérica, estandarizada y basada en componentes del sistema para anticipar los requerimientos del cliente.
- Nuevas aplicaciones son desarrolladas para determinar requerimientos específicos de la aplicación, seleccionar o generar componentes particulares y ensamblarlos acorde a la arquitectura de referencia.

La arquitectura de referencia puede incluir componentes concretos que deberían ser embebidos en todas las aplicaciones que derivan de ella. Además es adaptada para resolver los requerimientos particulares de los clientes.

La actividad principal es ajustar a la medida la arquitectura de referencia usando los requerimientos de la aplicación para producir la arquitectura específica de la aplicación.

1.3.2.1 Elementos de la DSSA

Una DSSA es un proceso e infraestructura que soporta el desarrollo de un modelo del dominio, requerimientos de referencia y una arquitectura de referencia para una familia de aplicaciones dentro de un dominio del problema. (KAISER 2005)

Modelo del dominio

El modelo del dominio es una representación de qué ocurre en el dominio, por ejemplo, operaciones de negocio en el dominio. Específicamente, esto describe las funciones a ser ejecutadas, las entidades que las ejecutan y los datos que fluyen entre ellas y si son usados o producidos por las funciones. El objetivo del modelo del dominio es estandarizar la terminología y la semántica del espacio del problema, por ejemplo el conjunto de problemas dentro del dominio que un conjunto de aplicaciones soportarán. (KAISER 2005)

Requerimientos de referencia

Los requerimientos de referencia son requerimientos estándares que surgen de diferentes áreas funcionales para todas las aplicaciones que pueden ser construidas en un dominio. Estos definen la estructura funcional general del espacio del problema, por ejemplo, los requerimientos funcionales. Además, especifican las restricciones sobre el diseño y la aplicación resultante, por ejemplo, los requerimientos no funcionales. Los requerimientos de las aplicaciones son realizados desde requerimientos particulares del cliente para cada aplicación específica. A veces, estos requerimientos son refinados a partir de requerimientos de referencia genéricos. (KAISER 2005)

Arquitectura de referencia

Una arquitectura de referencia es una arquitectura genérica, estándar, que describe a todos los sistemas dentro de un dominio determinado. Esto pudiera especificar tanto la plataforma de hardware o software, y los componentes. (KAISER 2005)

Los elementos de una arquitectura de referencia son los siguientes (KAISER 2005):

- Un modelo de la arquitectura de referencia que describe una configuración basada en componentes de hardware y software sobre un estilo arquitectónico.
- Un esquema arquitectónico que describe la estructura de alto nivel y restricciones de cada uno de los componentes, incluyendo las principales entidades de datos.
- Un diagrama de dependencias de componentes describiendo las interacciones a través de los principales componentes.
- Un conjunto de descripciones de interfaces de componentes describiendo el API de cada uno ellos.
- Un conjunto de restricciones organizadas por parámetros, rango de valores, valores por defecto, y consecuencia sobre la arquitectura.

Según lo planteado por Kaiser con respecto a la arquitectura de referencia, unido a nuestro criterio, se definirá el término de **arquitectura base**. Una arquitectura base es una arquitectura genérica, que no es orientada para desarrollar una aplicación específica, sino, que se puede personalizar para todos los tipos de aplicaciones del dominio para el cual fue definida. Permitiendo además, la estructuración y

organización del desarrollo de software desde el comienzo de cada aplicación, convirtiéndose así en la línea base de cada una de estas.

Los principales elementos de una arquitectura base según nuestro análisis son:

- Propuesta base de diseño de las capas lógicas de la aplicación.
- Convenciones o estándares de código (paquetes[4] y archivos de código) y de recursos (xml, archivos de propiedades, etcétera.).
- Estructura de código y recursos.
- Mecanismos de colaboración entre los componentes.

Propuesta base de diseño de las capas lógicas de la aplicación: es una propuesta de diseño inicial que sirve como guía para poder realizar y guiar el diseño de la aplicación a través de sus capas lógicas. Pudiera estar enriquecida por el uso de patrones de diseño y buenas prácticas. Esta propuesta inicial de diseño será enriquecida a medida que el equipo de arquitectura detecte necesidades de diseño de la aplicación específica del dominio que se esté desarrollando, que no resuelva totalmente la propuesta de diseño base.

Convenciones o estándares de código y de recursos: define las normas y estructuras en las que se debe: programar el código fuente, nombrar las clases y recursos según sus responsabilidades y funciones; logrando con esto una uniformidad, organización y un mayor entendimiento en la forma de codificar y nombrar las clases, componentes y recursos de la aplicación.

Estructura de código y recursos: define la estructura física de la aplicación, es decir, especifica físicamente donde va cada clase y recurso de la aplicación. De esta forma se crea el esqueleto inicial sobre el cual se desarrollará. A través del desarrollo propio de la aplicación o a criterio del equipo de arquitectura esta estructura inicial se irá transformando o enriqueciendo según sus necesidades.

Mecanismos de colaboración entre los componentes: describe las posibles formas de interactuar entre los componentes que integran la aplicación, logrando de esta manera una mayor o menor dependencia entre ellos.

[4] Paquete: se refiere a la agrupación de código fuente usado en java, que pudiera estar compuesto por uno o varios componentes.

1.3.2.2 Análisis y ambientes del dominio

El modelo del dominio es el resultado de un proceso de análisis del dominio, el cual es un aspecto esencial para diseñar los sistemas de software de alta calidad. El análisis del dominio es el proceso de analizar y entender el entorno del usuario final, por ejemplo, qué operaciones de negocio los usuarios ejecutan y en qué contexto. Dentro de la ingeniería de software, el análisis del dominio conlleva a estudiar el conocimiento base necesario para solucionar el problema del diseño del software. (KAISER 2005)

El análisis del dominio involucra (KAISER 2005):

- Identificar, capturar, organizar, y entender los objetos y las operaciones dentro del dominio.
- Una descripción de esos objetos y operaciones usando un vocabulario estandarizado.
- Identificar sistemas y problemas similares dentro del dominio.
- Determinar qué es reutilizable a través de los sistemas similares.

Según Stephen H. Kaiser en su libro "*Software Paradigms*", existen tres ambientes que en el análisis del dominio se debería tener presente para definir y desarrollar una aplicación. Ver figura 1-3.

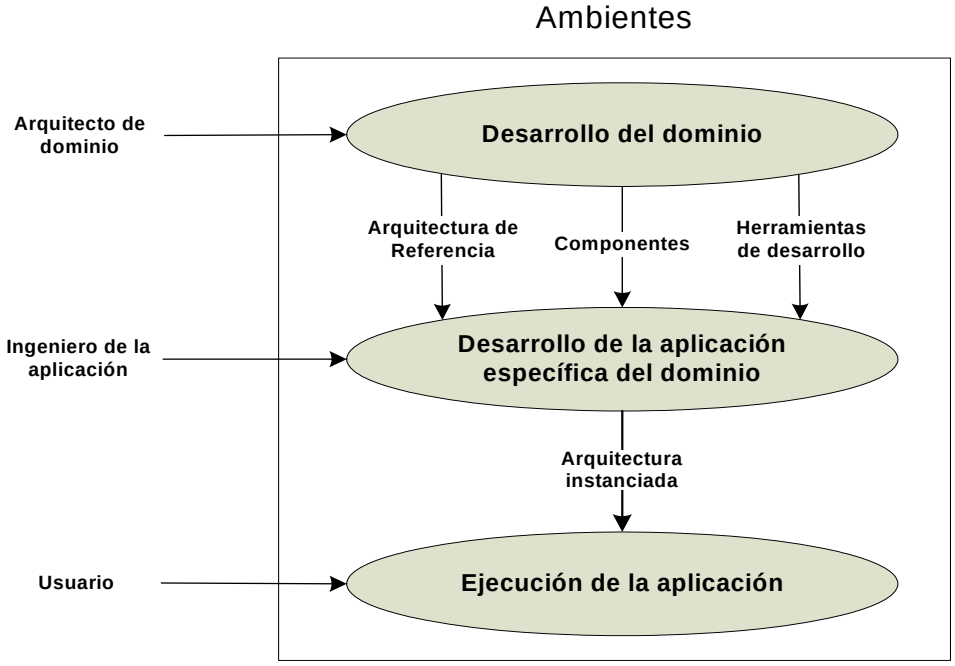

Figura 1-3 Ambientes de una DSSA

El primer ambiente es el mismo dominio, donde el análisis de los requerimientos y la definición de la arquitectura toman lugar. La arquitectura de referencia proporciona el esqueleto para las aplicaciones individuales desarrolladas para el dominio. El segundo ambiente es el entorno de desarrollo de la aplicación en el cual los requerimientos de la aplicación específica son producidos como respuesta y usados para desarrollar aplicaciones individuales. Finalmente, el ambiente de ejecución de la aplicación es donde la aplicación es usada para resolver los problemas. (KAISER 2005)

Para el desarrollo y definición de este trabajo se realizaron algunas modificaciones a este criterio planteado por Kaiser, ver figura 1-4.

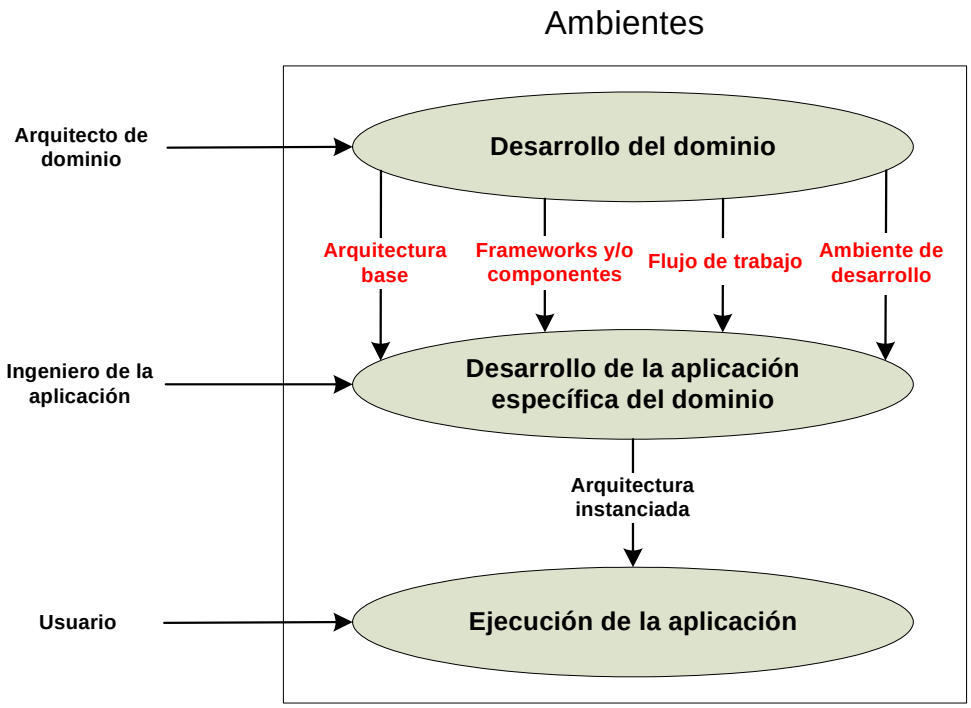

Figura 1-4 Modificaciones al concepto de DSSA

Estas modificaciones consisten en reemplazar los términos: “arquitectura de referencia” por “arquitectura base”, “componentes” por “frameworks y/o componentes”, “herramientas de desarrollo” por “ambiente de desarrollo”. Además de agregar un nuevo elemento: **flujo de trabajo**.

El **flujo de trabajo** permite orientar el desarrollo de la aplicación, como si fuera una guía con pasos a seguir de qué se debe hacer primero y qué después. Logrando de esta forma acortar el tiempo de desarrollo al máximo, aprovechando todos los beneficios de la arquitectura base definida

3.3 Ambiente de desarrollo

El ambiente de desarrollo (Development Environment) es algo imprescindible en la producción de software. Es donde se definen el conjunto de herramientas y tecnologías (frameworks), versiones a usar y su integración, que intervienen en un proceso de desarrollo de software.

A continuación se presentan un conjunto de herramientas utilizadas en el desarrollo de este trabajo; como son: un ambiente de desarrollo integrado (IDE) y sus plugins, un contenedor web, un control de versiones, dos gestores de base de datos, herramientas de modelado y los frameworks. Se exponen sus características y ventajas para poder tener un mayor conocimiento de sus prestaciones y valorar su utilidad.

1.3.1 Ambiente de desarrollo integrado

Un ambiente de desarrollo integrado o en inglés *Integrated Development Environment* (IDE) es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien, poder utilizarse para varios.

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. Estos proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Java, C#, Visual Basic, Object Pascal, etcétera.

En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto, como es el caso de Smalltalk.

A continuación se caracterizan un conjunto de herramientas utilizadas en el ambiente de desarrollo integrado.

Eclipse

Metafóricamente, Eclipse es como una tienda para herreros, donde no solamente se hacen productos, sino que además se hacen las herramientas para hacer los productos. Cuando se descarga el Eclipse SDK, se obtiene un equipo de instrumentos para desarrollo en Java o Java Development Toolkit (JDT) para escribir y depurar programas en Java; además se obtiene un ambiente de desarrollo de plugin Plug-in Development Enviroment para heredar de Eclipse. Si todo lo que se quiere es un IDE para Java, no se necesita nada además que el JDT. Esto es para lo que la mayoría las personas usan Eclipse. (DAVID GALLARDO 2003)

Aunque Eclipse es escrito en Java y su principal uso es como IDE para Java, este es un lenguaje neutral. El soporte para desarrollo en Java es proveído por un componente enchufado o plugin, pero además están disponibles plugins para otros lenguajes, como C/C++, Cobol, C# (DAVID GALLARDO 2003). En principio permite ejecutar un programa sobre cualquier plataforma. Es una extensible plataforma de código abierto (open source) para desarrollar herramientas. Es administrado y dirigido por un consorcio de compañías de desarrollo de software con un interés comercial en promover Eclipse como plataforma compartida para herramientas de desarrollo de software.

1.3.1.1 Plugins

Un plugin o plug-in, es una aplicación que interactúa con otra para agregarle una funcionalidad específica y es ejecutada por la aplicación principal. En el caso particular de Eclipse no son más que un conjunto de clases que permiten hacerlo más extensible.

Web Tool Platform (WTP)

La plataforma de herramientas web de Eclipse o Eclipse Web Tool Platform (WTP) provee varias APIs[5] para desarrollo de aplicaciones sobre la Web y JEE. Éstas incluyen editores gráficos, de código fuente para una variedad de lenguajes, asistentes y aplicaciones incorporadas para simplificar el desarrollo de servicios web, además de herramientas y APIs para soportar el despliegue, ejecución y prueba de aplicaciones.

[5] API: API (Application Program Interface), programa de aplicación de interfaz, parte del sistema operativo que provee a las aplicaciones una interfaz de uso común o interfaz similar.

Soporta integración con servidores Web dentro de Eclipse como ambiente de ejecución de primera clase para aplicaciones web. También incluye la configuración de servidores y su asociación con los proyectos web, permitiendo la depuración sobre el servidor de los recursos y las clases.

Spring IDE

Spring IDE es un plugin que sirve como interfaz de usuario gráfica para la configuración de los archivos usados por Spring Framework. Permite el completamiento de etiquetas, valores de atributos y elementos en estos archivos de configuración.

Analiza gramaticalmente expresiones de punto de cortes de AspectJ (AspectJ Pointcut Expressions) y @AspectJ-style pointcut expressions, mostrando errores si están incorrectas sintácticamente.

Muestra un árbol con todos los proyectos de Spring y sus archivos de configuración. Permite hacer búsquedas de beans[6] en dichos archivos. Muestra gráficamente los beans y sus dependencias.

Presenta un editor gráfico con completamiento y validaciones para los archivos de definiciones del flujo web usado por el Spring Web Flow.

Hibernate Tools

Hibernate Tools es un conjunto de herramientas enteramente para trabajar con el framework Hibernate del cual se amplía en este capítulo, implementado como una suite integrada de plugins para Eclipse.

Este plugin tiene las siguientes características disponibles:

- *Editor de mapeos*: Es un editor para los archivos de mapeos XML de Hibernate, soportando auto completamiento y sintaxis resaltada. Soporta incluso auto completamiento semántico para nombres de clases, propiedades, tablas y columnas.
- *Consola*: La perspectiva de consola de Hibernate permite configurar conexiones a base de datos, provee visualización de clases y sus relaciones. Admite además ejecutar preguntas en formato del lenguaje de preguntas de Hibernate (HQL) interactivamente contra la base de datos y mostrar los resultados.

[6] Beans: Son simples clases de Java con sus métodos de acceso.

- Ingeniería inversa: Es su característica más importante, permitiendo generar las clases del modelo de dominio y los archivos de mapeos de Hibernate, los EJB3 entity beans anotados, documentación en formato HTML, a partir de una base de datos.
- Asistentes: Presenta asistentes como: generador de archivos de configuración rápidamente, configuración de la consola, entre otros.
- Tareas de Ant: El Hibernate Tools incluye una tarea de Ant que permite ejecutar la generación de esquemas, mapeos o código Java como parte de su construcción.

Subclipse

Subclipse es un plugin para Eclipse que adiciona integración para el control de versiones (Subversion, específicamente), permitiendo operaciones de sincronización, actualización, entre otras. Permite bloqueos a recursos para que otros usuarios no puedan modificarlo. Dispone de una vista de comparación entra el recurso local y remoto en caso que exista conflicto entre la versión del recurso local con el remoto. Muestra una vista del historial de versiones de los recursos con un conjunto de atributos de las acciones realizadas sobre el recurso.

Soporta conectarse a varios repositorios de control de versiones al mismo tiempo, permitiendo hacer operaciones sobre el repositorio directamente.

Es un plugin muy útil para el desarrollo colaborativo, en el que intervienen un conjunto de desarrolladores trabajando sobre el mismo proyecto, poniendo a disposición del equipo de desarrollo facilidades para el trabajo en equipo.

AspectJ Development Tools (AJDT)

En la implementación de sistemas existen algunos aspectos que son difíciles de implementar de forma modular: identificación y autenticación; manipulación de errores, ejecución de estándares y variaciones en algunos rasgos, entre otros. El desarrollo de software orientado a aspectos permite la clara modularización de funcionalidades que de cortan a través de los módulos. Uno de los proyectos que provee la comunidad de Eclipse es el AspectJ Development Tools (AJDT). Este es un plugin que provee herramientas para soportar el desarrollo de software orientado a aspectos usando el framework AspectJ.

1.3.2 Contenedor Web

Los servidores de aplicaciones se crearon con el fin de gestionar las peticiones del usuario y devolverle a los mismos recursos a través de un protocolo de comunicación (HTTP, Hypertext Transfer Protocol, protocolo de transmisión del hipertexto, el protocolo que sirve para incursionar en los sitios de WWW en Internet).

Sin embargo en el mundo de JEE surge un término muy común, contenedor web, el cual además de interceptar solicitudes enviadas en los protocolos: HTTP, HTTPS, FTP, y otros, es esencialmente un período de ejecución Java que proporciona una implementación del API Servlet7 Java y facilidades para las páginas servidoras de Java o JavaServer Pages (JSP). Además es responsable de inicializar, invocar, y gestionar el ciclo de vida de los servlets Java y las páginas JSP.

Apache Tomcat

Apache Tomcat es un contenedor de Servlet usado en la implementación de referencia oficial para las tecnologías Java Servlet y JavaServer Pages. Es desarrollado en un ambiente participativo y abierto.

Apache Tomcat es usado en numerosas aplicaciones web de gran escala y criticas en diversas industrias y organizaciones que se referencian en su sitio oficial[8].

La versión 5.x es implementada a partir de las especificaciones Servlet 2.4 y JSP 2.0 y cuenta con un mecanismo de recolección de basura perfeccionado, basado en su reducción. Posee una capa envolvente nativa para Windows y Unix para la integración de las plataformas

1.3.3 Control de versiones

Control de versiones es la gestión de versiones (revisiones) de todos los elementos de configuración que forman la línea base de un producto o una configuración del mismo. Los sistemas para el control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado junto a las posibles especializaciones realizadas para algún cliente específico. Sin embargo, los mismos conceptos son aplicables a otros ámbitos y no sólo para código fuente sino para documentos, imágenes, archivos xml, xsd, dtd, jsp, html, etcétera.

[7] Servlet: Los Servlets son un tipo de aplicación de Java que radica dentro de un servidor web, llamado contenedor de Servlet (llamado originalmente un motor de Servlet)
[8] http://tomcat.apache.org/

Un sistema de control de versiones debe proporcionar: Un mecanismo de almacenaje de cada uno de los recursos que deba gestionarse (archivos de texto, imágenes, documentación, etcétera.), posibilidad de modificar, mover, borrar cada uno de los elementos, un histórico de las acciones realizadas con cada elemento pudiendo volver a un estado anterior dentro de ese historial.

Aunque un sistema de control de versiones puede realizarse de forma manual, es muy aconsejable disponer de herramientas que faciliten esta gestión, por ejemplo, Subversion.

Subversion

Subversion es un software de sistema de control de versiones de código abierto y gratuito. Soporta el manejo de ficheros y directorios a través del tiempo. Hay un árbol de ficheros en un repositorio central. El repositorio es como un servidor de ficheros ordinario, excepto porque recuerda todos los cambios hechos a sus ficheros y directorios, permitiendo recuperar versiones antiguas de sus datos, o examinar el historial de cambios de los mismos. En este aspecto, mucha gente piensa en los sistemas de versiones como en una especie de “máquina del tiempo”.

Subversion es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS[9], el cual posee varias deficiencias. Se lo conoce también como svn por ser ese el nombre de la herramienta de línea de comandos. Una característica importante de Subversion es que, a diferencia de CVS, los archivos versionados no tienen cada uno un número de revisión independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo.

Presenta las siguientes características principales: Se sigue la historia de los archivos y directorios a través de copias y renombrados; las modificaciones (incluyendo cambios a varios archivos) son atómicas; crear ramas y etiquetas es una operación más eficiente; tiene costo de complejidad constante (O(1)) y no lineal (O(n)) como en CVS; se envían sólo las diferencias en ambas direcciones (en CVS siempre se envían al servidor archivos completos); maneja eficientemente archivos binarios (a diferencia de CVS que los trata internamente como si fueran de texto).

[9] CVS: Siglas de Concurrent Version System o Concurrent Versioning System, es un sistema de control de versiones.

1.3.4 Sistemas Gestores de Base de Datos

Un Sistema Gestor de base de datos (SGBD) es un tipo de software muy específico o conjuntos de ellos que permite manejar de manera clara, sencilla y ordenada altos volúmenes de datos. Actualmente existe una amplia gama de SGBD con características propias, no obstante todos deben tener en cuenta los siguientes aspectos de manera general: abstracción la información, la independencia de los datos, redundancia mínima, consistencia en los datos, seguridad, integridad, respaldo y recuperación, tiempo de respuesta y control de concurrencia.

Existen SGBD muy potentes tanto en la clasificación de software propietario como software libre. Como propietarios podemos encontrar Oracle y Microsoft SQL Server que lideran el mercado por sus altas prestaciones dado muchos años de experiencia mientras que como opción libre se encuentra, entre otros, MySQL, gestor muy utilizado en la web por su simplicidad de uso, y PostgreSQL que si bien no soporta alguno de los aspectos más avanzados si representa una muy opción muy confiable y comprometedora.

Oracle

Oracle es un sistema de gestión de base de datos relacional (o RDBMS por el acrónimo en inglés Relational Data Base Management System), fabricado por Oracle Corporation. Se considera uno de los sistemas de bases de datos más completos. Es un sistema gestor de base de datos robusto, tiene muchas características que nos garantizan la seguridad e integridad de los datos; que las transacciones se ejecuten de forma correcta, sin causar inconsistencias; ayuda a administrar y almacenar grandes volúmenes de datos; estabilidad, escalabilidad y es multiplataforma.

Aunque su dominio en el mercado de servidores empresariales ha sido casi total hasta hace poco, recientemente sufre la competencia de gestores de bases de datos comerciales y de la oferta de otros con licencia Software Libre como PostgreSQL. Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo Linux.

PostgreSQL

PostgreSQL es un sistema gestor de base de datos de objetos relacionales basado en software libre. Ofrece una alternativa ante otros sistemas gestores de bases de datos comerciales. Similar a proyectos de software libres como Apache y GNU/Linux, PostgreSQL no es controlado por ninguna compañía en específico, pero responde al esfuerzo de una comunidad global de desarrolladores.

Ofrece una potencia adicional sustancial al incorporar los siguientes cuatro conceptos adicionales básicos en una vía en la que los usuarios pueden extender fácilmente el sistema: clases, herencia, tipos y funciones. Otras características aportan potencia y flexibilidad adicional como son las restricciones (constraints), disparadores (triggers), reglas (rules) e integridad transaccional.

1.3.5 Herramientas de modelado

Los medios con los que siempre se ha realizado el intercambio de información de diseño e ideas usando la notación UML han sido populares: pizarras, cuadernos y trozos de papel, etcétera. Pero UML se utiliza mejor a través de una herramienta de modelado, la cual puede ser usada para capturar, guardar, rechazar, integrar automáticamente información, y diseño de documentación. (*Modelado de Sistemas con UML* 2002)

Una característica que UML brinda para beneficiar a los modeladores es escoger una herramienta de modelado. Tiempos atrás, el modelador primero tenía que seleccionar una notación de metodología, y después estaba limitado a seleccionar una herramienta que la soportara. Ahora con UML como estándar, la elección de notación ya se ha hecho para el modelador. Y con todas las herramientas de modelado soportando UML, el modelador puede seleccionar la herramienta basada en las áreas claves de funcionalidad soportadas que permiten resolver los problemas y documentar las soluciones. (*Modelado de Sistemas con UML* 2002)

Como una buena caja de herramientas, una buena herramienta de modelado ofrece todas las herramientas necesarias para conseguir hacer eficientemente varios trabajos, sin dejarte nunca sin la herramienta correcta.

Las herramientas de modelado deberían soportar las siguientes funcionalidades:

- Soporte para toda la notación y semántica de UML.
- Soporte para una cantidad considerable de técnicas de modelado y diagramas para complementar UML, incluyendo tarjetas CRC, modelado de datos, diagramas de flujo, y diseño de pantallas de usuario. Posibilidad de reutilizar información obtenida por otras técnicas todavía usadas, como modelado tradicional de procesos.
- Facilitar la captura de información en un repositorio subyacente permitiendo la reutilización entre diagramas.

- Posibilidad de personalizar las propiedades de definición de elementos subyacentes de modelos UML.
- Permitir a varios equipos de analistas trabajar en los mismos datos a la vez.
- Posibilidad de capturar los requisitos, asociarlos con elementos de modelado que los satisfagan y localizar cómo han sido satisfechos los requisitos en cada uno de los pasos del desarrollo.
- Posibilitar la creación de informes y documentación personalizados en tus diseños, y la salida de estos informes en varios formatos, incluyendo HTML para la distribución en la Internet o Intranet local.
- Posibilidad para generar y realizar ingeniería inversa (por ejemplo C++, Java, etcétera.) para facilitar el análisis y diseño interactivo, para volver a usar código o librerías de clase existentes, y para documentar el código.

ER/Studio

ER/Studio es una herramienta de modelado para diseñar bases de datos. Ayuda a descubrir, documentar y reutilizar los datos activos. Con un soporte de ida y vuelta de bases de datos, los arquitectos de datos tienen el poder para analizar a donde las fuentes de datos existentes tan bien como el diseño e implementación de bases de datos de alta calidad.

Esta herramienta ofrece las siguientes características: Ambiente de diseño orientado al modelo, soporte del ciclo de vida de bases de datos completas, administración del modelo empresarial, soporte para integración y almacenes de datos o data warehouse, y diseño de base datos con calidad.

Visual Paradigm Suite

Visual Paradigm Suite es un conjunto de herramientas de modelado que permiten realizar el modelado dentro del proceso de desarrollo de software.

A continuación se describen las principales herramientas presentes dentro de esta suite:

- Visual Paradigm for UML Enterprise Edition: Es una herramienta de modelado diseñada para un gran número de usuarios, incluyendo ingenieros de sistemas, analistas de sistemas, analistas de negocio, arquitectos de sistemas. Además se integra con IDEs (Eclipse, JBuilder, NetBeans, IntelliJ IDEA, JDeveloper and WebLogic Workshop) para soportar la fase de implementación de

desarrollo de software. La transición desde el análisis al diseño y después a la implementación es cuidadosamente integrada dentro de la herramienta CASE, de esta manera se reduce significativamente el esfuerzo en todas las etapas del ciclo de vida del desarrollo del software. Incluye además un conjunto de herramientas que soportan Object-Relational Mapping (ORM), como Hibernate, lo cual incluye generación completamente orientada a objetos, listo para usar librerías para obtener y modificar registros de base de datos para una gran variedad de gestores de base de datos, y la sincronización entre los diagramas de clases y los diagramas de entidad-relación (ERD). Presenta además generación de código e ingeniería inversa del código. Permite generar los EJB y así mismo los descriptores de despliegue para varios servidores de aplicación. Distinto a muchas herramientas de modelado pueden extenderse sus diagramas hechos desde el Visio y de Rational Rose. Soporta la última versión de UML 2.0

- Visual Paradigm Smart Development Environment (SDE) Enterprise Edition: Visual Paradigm SDE ofrece integración de la herramienta de modelado UML con los IDEs (Visual Studio®, Eclipse/WebSphere®, Borland JBuilder®, NetBeans/Sun™ ONE, IntelliJ IDEA™, Oracle JDeveloper, BEA WebLogic Workshop™). Visual Paradigm SDE se incrusta él mismo dentro de tu IDE favorito para proveer ambiente de desarrollo y modelado unificado. Dando todas las prestaciones y servicios que brinda el Visual Paradigm for UML Enterprise Edition.
- Visual Paradigm DB Visual Architect: DB Visual Architect es un simple y fácil de usar ambiente de Object-Relational Mapping (ORM), como Hibernate, el cual actúa como un puente entre el modelo de objetos, el modelo de datos y el modelo relacional. Este ayuda a los desarrolladores a construir aplicaciones de base de datos de alta calidad, mucho más rápido, mejores y barato. Disminuye el costo de desarrollo a un 75 % por automatizar el proceso de mapeo entre los objetos de java y las tablas de la base de datos visualmente y a través de diagramas. Los desarrolladores ahorran enormes cantidades de tiempo usando DB Visual Architect para manipular el código de SQL y JDBC transparentemente con (POJO), como DB Visual Architect está habilitado con ingeniería inversa de una base de datos relacional a un diagrama de clases y un diagrama entidad-relación (ERD) y viceversa (crear una base de datos relacional de modelos de relación de entidad (ERD) o de modelos de clases.

1.3.6 Frameworks

Antes de llegar a la definición de qué es un framework, es imprescindible mencionar los conceptos de componente y componente de software.

Un componente es un bloque de construcción reusable de software: una pieza de código encapsulada de una aplicación que puede ser combinada con otros componentes y con códigos adicionales para producir una aplicación específica. Los componentes pueden ser simples o complejos. No existe un acuerdo general sobre qué es o qué no es un componente. Ellos vienen en una variedad de formas y tamaños. Un componente puede ser muy pequeño, como lo es un GUI widget[10] (por ejemplo, un botón), o éste puede implementar un complejo servicio de una aplicación, tal como una función para administrar la red. (KAISER 2005)

El concepto comúnmente aceptado de un componente es llamado “componente binario”, el cual es un artefacto de software compilado que es integrado dentro de la aplicación en tiempo de ejecución (runtime). El código fuente no es usualmente disponible, por tanto los componentes no pueden ser modificados. Si los componentes están implementados en un lenguaje de programación orientado a objeto, ellos pueden ser extendidos a través de la herencia o delegación. (KAISER 2005)

Una aplicación está compuesta por una colección de componentes. Las aplicaciones proveen un ambiente, glue code[11] o un esqueleto de código, dentro del cual los componentes son insertados. Los componentes interactúan con su ambiente y pudieran interactuar con el sistema operativo de la computadora sobre la cual ellos son ejecutados. Además, ellos interactúan unos con otros a través de interfaces. Cambiar una interfaz de un componente pudiera requerir cambiar el componente que usa esta interface. Sin embargo, cambiar la implementación del componente no debería requerir cambiar a los clientes de ese componente. (KAISER 2005)

Un componente de software es un independiente paquete entregable de servicios de software, en el mayor sentido general. Philippe Kruchten of Rational Software cree que un componente no es trivial, que es casi independiente y una parte reemplazable de un sistema que satisface una función clara en el contexto de una arquitectura bien definida. Un componente es para conformar y proveer la realización

[10] Widget: Objeto cuyo nombre no se sabe o se olvidó; símbolo gráfico de interface que permite interacción entre el usuario y el ordenador (símbolo, Icono, ventana y etcétera.).
[11] Glue code: en términos de programación, es el código que no realiza ninguna funcionalidad para satisfacer algún requerimiento del programa en cuestión.

física de un conjunto de interfaces. Por otro lado Szyperski en su libro “Component-Based Software: Beyond Object-Oriented Programming”, ve a un componente de software como una unidad de composición con interfaces especificadas contractualmente y un explicito contexto de dependencias. Un componente de software puede ser desarrollado independientemente y estar sujeto a composición de terceras partes. De esta manera, un componente de negocio representa la implementación de software de una operación autónoma de negocio o proceso. (KAISER 2005)

Recientemente, el interés en reutilizar software ha sido cambiado de la reutilización de componentes simples a diseño de sistemas enteros o estructuras de aplicaciones. Un sistema software que pudiera ser reutilizado en este nivel para la creación de aplicaciones completas es llamado *framework*. Los frameworks son basados en la idea que deberían permitir la producción fácil de un conjunto de sistemas específicos pero similares, dentro de un cierto dominio comenzando desde una estructura genérica. Brevemente, los frameworks son arquitecturas genéricas integradas por un extensible conjunto de componentes. Además, los frameworks pueden contener subframeworks los cuales representen subconjuntos de componentes de un sistema más grande. (KAISER 2005)

Estos enfatizan en la reutilización del diseño sobre el código, aunque el código resultante puede ser reutilizado. Usualmente definen la estructura total de todas las aplicaciones derivadas de él, una organización entre clases y objetos, principales responsabilidades de clases individuales, cómo ellos colaboran y el hilo de control de éstas. El desarrollador es el responsable para personalizar el framework para una aplicación en particular. (KAISER 2005)

Los frameworks invierten el rol de control entre la aplicación y la infraestructura, es decir la infraestructura llama o invoca rutinas de la aplicación. En sistemas convencionales, los propios desarrolladores de programas proveen toda la estructura y control del flujo de ejecución y realizan llamadas a librerías de funciones como sea necesario; pero usando un framework, los desarrolladores tienen solamente que preparar estos componentes que no están en el framework acorde al comportamiento deseado de la aplicación. (KAISER 2005)

Un framework contiene clases o estructuras que implementan los componentes de una aplicación genérica también como componentes concretos que satisfacen tareas especializadas. Para desarrollar programas completos, los desarrolladores buscan e instancian los componentes apropiados.

No hay una definición común para los frameworks, pero la mayoría tiene un tema común: la reutilización. Una definición ampliamente aceptada es dada por R. E. Johnson, and B. Foote en 1988 en su publicación “Designing Reusable Classes (R. E. JOHNSON, B. FOOTE 1988):

> *"Un framework es un conjunto de clases que personifican un diseño abstracto para soluciones de una familia de problemas relacionados..."*

Otro punto de vista, según R. E. Johnson y V. F. Russo en “Reusing Object-Oriented Designs” incluye (R. E. JOHNSON, V. F. RUSSO):

“Una clase abstracta es un diseño para un objeto simple. Un framework es el diseño de un conjunto de objetos que colaboran para llevar a cabo un conjunto de responsabilidades. De esta manera los frameworks son diseños a escalas más grandes que las clases abstractas. Los frameworks son una forma de reutilizar el diseño a un nivel superior.”

Spring Framework

Spring Framework es un framework de aplicación de código abierto que ayuda a hacer el desarrollo en JEE mucho más fácil. Ayuda a estructurar aplicaciones completas en una manera consistente y productiva para crear arquitecturas coherentes. (JOHNSON 2005)

Es el más popular y el más ambicioso de todos los framework de peso ligero. Es el único framework que interviene en todas las capas arquitectónicas de una aplicación JEE. Además está diseñado para facilitar una flexibilidad arquitectónica. (JOHNSON 2005)

Presenta varios módulos de los cuales los principales son (JOHNSON 2005):

- *Contenedor de Inversión de Control:* permite una sofisticada administración de configuración para POJOs[12] y trabaja con otras partes de Spring para proveer servicios como la administración de la configuración.
- *Un framework para la Programación Orientada a Aspectos (AOP):* permite comportamientos que deberían ser esparcidos de otra manera a través de diferentes métodos para ser modularizados en un simple lugar. Spring usa la AOP para implementar importantes servicios tales como

[12] POJO: Es el acrónimo de Plain Old Java Object. Clase del lenguaje de programación Java. Este nombre se les da a las clases que no son de algún tipo especial (EJBs, Java Beans, etcétera) y no cumplen ningún otro rol ni implementan alguna interfaz especial.

administración de transacciones declarativas, y además es usado para implementar códigos estándar que debería de otra manera ser esparcido entre las clases de la aplicación.

- *Una abstracción de acceso a datos:* Spring anima a un consistente acercamiento arquitectónico para acceso a datos, y posee una abstracción única y poderosa para implementarlo. Presenta una rica jerarquía de excepciones de acceso a datos, independiente de cualquier framework de persistencia en particular.
- *Simplificación de JDBC[13]:* presenta una capa de abstracción sobre JDBC que es significantemente mucho más simple y menos propensa a errores para usar que JDBC puro cuando se necesita usar acceso a través de SQL a bases de datos relacionales.
- *Administración de transacciones:* presenta una abstracción de transacciones que puede mover transacciones globales JTA (manipuladas por un servidor de aplicaciones) o transacciones locales usando JDBC, Hibernate, JDO u otro API de acceso a datos. Estas abstracciones presentan un consistente modelo de programación en una variedad de ambientes y es la base de la administración de transacciones programáticas y declarativas usadas por Spring.
- *Framework Web MVC:* Spring presenta un framework web MVC basado en peticiones. Éste tiene un acercamiento al framework de Struts pero el framework web de Spring es más flexible, y se integra discretamente al contenedor de Inversión de control de Spring. Todos los demás rasgos de Spring se pueden usar con otros frameworks tales como Struts, JSF, WebWork, Tapestry, etcétera.
- *Simplificación para trabajar con JNDI, JTA y otros APIs de JEE:* Spring puede ayudar a eliminar el código que no hace nada. Con Spring se puede usar JNDI o EJB, si se quiere, pero nunca se necesitará escribir un buscador de JNDI.
- Comunicación remota de peso ligero (Lightweight remoting): Spring presenta comunicación remota basada en POJOs sobre un rango de protocolos, incluyendo RMI, IIOP, Hessian, Burlap, y otros protocolos se de servicios web.

[13] JDBC: Es el acrónimo de Java Database Connectivity, un API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java independientemente del sistema de operación donde se ejecute o de la base de datos a la cual se accede utilizando el dialecto SQL del modelo de base de datos que se utilice.

- *Soporte para Servicio de Mensajería de Java (Java Message Service, JMS):* Spring presenta soporte para enviar y recibir mensajes de una forma mucho más simple que la que provee la especificación JEE.
- *Soporte para Java Management Extension (JMX):* Spring presenta soporte para la administración JMX de objetos de una aplicación para ser configurados.
- *Soporte para comprensivas estrategias de pruebas para desarrolladores de aplicación:* Spring no solamente ayuda a realizar un buen diseño, permitiendo efectivas pruebas de unidad, sino que presenta una comprensiva solución para pruebas de integración fuera de un servidor de aplicaciones.

Los principales valores de Spring, según Rod Johnson, en su libro: “Professional Java Development with the Spring Framework”, se pueden resumir en:

- *No es un framework agresivo:* Este es el principal problema de los framework anteriores. Mientras que los framework tradicionales tales como EJB o Apache Avalon que forzaban al código de las aplicaciones a ser dependientes del framework, implementando interfaces específicas de estos framework o heredando clases específicas de estos; ayuda a reducir las dependencias del código de la aplicación sobre el framework.
- *Provee un consistente modelo de programación, útil en cualquier ambiente:* Muchas aplicaciones web simplemente no necesitan ser ejecutadas sobre un pesado servidor de aplicaciones, sino que es mejor ejecutarlas sobre un servidor web, como, Tomcat o Jetty. También es importante recordar que no todas las aplicaciones se ejecutan del lado del servidor. Spring provee un modelo de programación que aísla el código de la aplicación de los detalles del ambiente tales como JNDI, logrando que el código sea menos dependiente del contexto de ejecución.
- *Ayuda a promover la reusabilidad de código:* Ayuda a evitar la necesidad de tomar decisiones importantes y duras, como es si una aplicación alguna vez usará JTA o JNDI; las abstracciones de Spring permitirán desarrollar el código en un diferente ambiente si tú alguna vez lo necesitas.
- *Facilita el diseño Orientado a Objetos (OO) en aplicaciones JEE:* Deberíamos preguntarnos: ¿Cómo una aplicación JEE, escrita en Java – un lenguaje OO – no es OO? En realidad muchas aplicaciones JEE no merecen el nombre OO. Con Spring es más fácil eliminar los impedimentos

del diseño OO en aplicaciones JEE tradicionales haciendo el código más coherente, con más bajo acoplamiento y más reusable.

- *Facilita buenas prácticas de programación, tales como la programación a interfaces:* El uso de un contenedor de Inversión de Control reduce grandemente la complejidad del código a interfaces, más que a clases. El uso de los objetos a través de estas interfaces protege los requerimientos, los cuales pudieran cambiar en el desarrollo de la aplicación.
- *Permite la extracción de valores de configuración desde el código java a archivos XML o archivos de propiedades:* Mientras que algunos valores de configuración pudieran ser programados en Java, todas las aplicaciones de mediana y alta complejidad necesitan algunas configuraciones externalizadas del código fuente, para permitir la administración sin recompilar las clases nuevamente.
- *Está diseñado a fin que las aplicaciones lo usen para que las pruebas sean lo más fácil posible:* Hasta donde sea posible, los objetos de las aplicaciones serán POJOs los cuales son fáciles de probar. La dependencia sobre las APIs de Spring normalmente serán en forma de interfaces que son fáciles para simular.
- *Es consistente:* En diferentes ambientes de ejecución y en diferentes partes del framework, Spring usa un consistente acercamiento. Una vez que se aprenda una parte del framework, te darás cuenta que este conocimiento puede ser aplicado en muchas a otras.
- *Promueve la selección arquitectónica:* Mientras Spring provee una columna vertebral arquitectónica, Spring apunta para facilitar el reemplazo de cada capa. Por ejemplo, con una capa media de Spring, se pudiera ser capaz de cambiar de un framework de mapeo objeto relacional (ORM) a otro con un impacto mínimo sobre el código de la lógica de negocio, o cambiar de Struts a Spring MVC o WebWork sin algún impacto en la capa media.
- *No reinventa la rueda:* Spring no introduce una solución propia en áreas tales como ORM donde ya existen muy buenas soluciones, Hibernate, iBatis, JDO, OBJ, TopLink. Tampoco implementa su propia solución de abstracción para poner en bitácoras (logging), pool de conexiones, coordinador de transacciones distribuidas, protocolos remotos u otros sistemas de servicios que son ofrecidos

en otros productos o servidores de aplicaciones. Sin embargo, Spring hace de estas soluciones existentes un uso más fácil, dentro de una arquitectura consistente.

Acegi Security System

Acegi Security es un framework de código abierto altamente usado por la comunidad de Spring para proveer los servicios de seguridad a las aplicaciones basadas principalmente en Spring Framework. El diseño de Acegi le permite a las aplicaciones sobre java aplicar los cuatro requerimientos de seguridad más comunes de las aplicaciones empresariales: autenticación, seguridad sobre las peticiones web, seguridad sobre la capa de servicios y seguridad a las instancias de los objetos de dominio. Provee integración a la aplicación con la mayoría de los posibles requerimientos específicos para cada aplicación como el uso de *Https*, integración con *LDAP*, servicios para recordar usuarios e integración con las funcionalidades de *Captcha*.

AspectJ

AspectJ es un lenguaje de programación orientado por aspectos construido como una extensión del lenguaje Java creado en Xerox PARC. Un compilador de AspectJ hace llegar la noción de aspecto hacia el código de máquina virtual implementando así una noción de relación. Los aspectos en si se escriben en Java extendido generándose un archivo java o compilado con código de máquina compatible con el generado por los compiladores de Java.

Hibernate

Hibernate es un potente framework de mapeo objeto/relacional y servicio de consultas para Java. Es la solución ORM (Object-Relational Mapping) más popular en el mundo Java. Permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. De una manera muy rápida y optimizada podremos generar BBDD en cualquiera de los entornos soportados: Oracle, DB2, MySQL, etcétera.

Sus principales características son:

- Permite expresar consultas utilizando SQL nativo o consultas basadas en criterios.

- Soporta todos los sistemas gestores de bases de datos SQL y se integra de manera elegante y sin restricciones con los más populares servidores de aplicaciones J2EE y contenedores web, y por supuesto también puede utilizarse en aplicaciones de escritorio.
- Puede operar proporcionando persistencia de una manera transparente para el desarrollador.
- Soporta el paradigma de orientación a objetos de una manera natural: herencia, polimorfismo, composición y el framework de colecciones de Java.
- Soporte para modelos de objetos con una granularidad muy fina Permite una gran variedad de mapeos para colecciones y objetos dependientes.
- Provee un sistema de caché de dos niveles y puede ser usado en un clúster. Permite inicialización perezosa (lazy) de objetos y colecciones.
- Proporciona el lenguaje HQL en cual provee una independencia del SQL de cada base de datos, tanto para el almacenamiento de objetos como para su recuperación.
- Presenta un potente mecanismo de transacciones de aplicación llegando incluso a permitir la interacciones largas (aquellas que requieren la interacción con el usuario durante su ejecución).
- Soporta los diversos tipos de generación de identificadores que proporcionan los sistemas gestores de bases de datos así como generación independiente de la base de datos, incluyendo identificadores asignados por la aplicación o claves compuestas.

JUnit

JUnit es un framework permite realizar la ejecución de clases Java de manera controlada para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. En función de algún valor de entrada se evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente.

JUnit es también un medio de controlar las pruebas de regresión necesarias cuando una parte del código ha sido modificado y se desea ver que el nuevo código cumple con los requerimientos anteriores y que no se ha alterado su funcionalidad después de la nueva modificación.

El propio *framework* incluye formas de ver los resultados (*runners*) que pueden ser en modo texto, gráfico (AWT o Swing) o como tarea en Ant.

En la actualidad las herramientas de desarrollo como NetBeans y Eclipse cuentan con *plug-ins* que permiten que la generación de las plantillas necesarias para la creación de las pruebas de una clase Java se realice de manera automática, facilitando al programador enfocarse en la prueba y el resultado esperado, y dejando a la herramienta la creación de las clases que permiten coordinar las pruebas.

Ehcache

Es un simple, rápido y seguro framework de caché para Java. Ehcache provee almacenamiento en memoria y disco y distribuye las operaciones para clusters. Es muy usado en proyectos de código abierto (open source) como Hibernate y Spring.

JCaptcha

JCAPTCHA (Java Completely Automated Public Test to tell Computers and Humans Apart) es un framework open source para la definición e integración en java con Captcha. Captcha es un simple contenedor que presenta una pregunta y una rutina de verificación de respuesta. Usa un generador de palabras, presenta componentes capaces de mostrarle las palabras al usuario en forma de foto o sonido. Es de muy fácil integración con cualquier tecnología sobre java y con grandes facilidades de escalabilidad. Presenta gran cantidad de componentes elaborados para que la utilización final en un proyecto sea solo cuestión de configuración.

JSON-RPC

JavaScript[14] Object Notation (JSON) es un formato de intercambio de datos de peso ligero. Este es de fácil lectura y escritura para los humanos y es fácil para ser parseados y generados por las maquinas. JSON es un formato de texto que es completamente independiente del lenguaje pero usa convenciones que son familiares a los programadores de los lenguajes de la familia de C, como son C, C++, C#, Java, JavaScript, Perl, Python, entre otros. Estas propiedades hacen a JSON un lenguaje de intercambio de datos ideal.

[14] JavaScript: es un lenguaje de programación utilizado para crear pequeños programitas encargados de realizar acciones dentro del ámbito de una página web.

JSON-RPC (RPC, Remote Procedure Call) es un protocolo de llamadas a procedimientos remotos similar a XML-RPC, pero solo usa JSON en lugar de XML como formato de datos.

JSON-RPC-Java es una implementación de JSON-RPC en JAVA. Se puede utilizar en servidores web como Tomcat, JBoss, entre otros. Utiliza la reflexión de Java. El objetivo es acceder a objetos remotos desde JavaScript completamente tranparente. Exporta los métodos de los objetos con una línea de código. Crea proxies dinámicos en JavaScript para exportar objetos de Java. Realiza el mapeo dinámicamente entre los objetos de Java y JavaScript. Se apoya para la seguridad en el modelo de seguridad de JEE. Exportando sesiones específicas de objetos. Es soportado por una gran variedad de navegadores web tales como Internet Explorer, Mozilla, Firefox, Safari, Opera y Konqueror.

DBCP

Uno de los tipos de objetos en java que consumen más recursos y tiempo de creación son los objetos que establecen una conexión con una base de datos. Para este tipo de objetos se crearon basados en el patrón de diseño Object Pool (GRAND), componentes que administran estos objetos como si estuviesen dentro de una piscina (pool). Un pool de conexiones a base de datos es donde se mantienen las conexiones de alto rendimiento, las cuales son usadas y devueltas al pool donde estaban almacenadas. Estos tienen las siguientes características:

- Deben ser escalables y tener la habilidad para generar nuevas conexiones cómo y cuando son requeridas.
- No deberían mantener muchos recursos dentro del pool, estas deberían ser liberadas cuando pasan mucho tiempo inutilizadas.
- El mecanismo del pool de conexiones debería ser configurable, rápido y eficiente.
- No debería limitarse a ser usado con bases de datos específicas. En el contexto de Java, un driver de conexión a base de datos para Java o Database Connection (JDBC), es lo que se necesita para poder usar el pool de conexiones con cualquier base de datos.

El Database Connection Pool (DBCP) es un componente que soporta estas características.

1.4 Conclusiones

Con el análisis desarrollado en este capítulo se puede llegar a la conclusión de que es necesario para definir una DSSA un ambiente de desarrollo, una arquitectura base, un framework o conjunto de componentes y un flujo de trabajo para desarrollar una aplicación. Logrando de esta forma el más alto nivel de la reutilización de componentes y acortando el tiempo de desarrollo de la misma, garantizando el uso de patrones de diseño y buenas prácticas. A su vez se ha llegado a la definición más práctica de qué elementos se tienen que tener en cuenta a la hora de realizar una arquitectura de software de dominio específico, lo cual se pondrá en práctica en el desarrollo de los siguientes capítulos.

Capítulo 2. ArBaWeb: Dominio y Arquitectura

2.1 Introducción

Hasta el momento se han explicado los elementos que se deberían tener en cuenta para el desarrollo de una DSSA. En este capítulo se utilizarán las definiciones analizadas anteriormente para definir un dominio específico de una DSSA. La DSSA que se desarrollará ha sido llamada **ArBaWeb**, derivado de "arquitectura base sobre la web". Esta permitirá estructurar y organizar el desarrollo de las aplicaciones que la usen, haciendo uso de patrones de diseños y buenas prácticas.

Se establecerá el dominio de ArBaWeb, en donde serán identificados los requerimientos de referencias, es decir, los requerimientos comunes entre las aplicaciones del dominio. La figura 2-1 indica el alcance que tendrá ArBaWeb dentro de la estructura de ambientes para el dominio planteados en el capítulo 1 reflejando solo el primero de los ambientes que se define en las modificaciones de la DSSA.

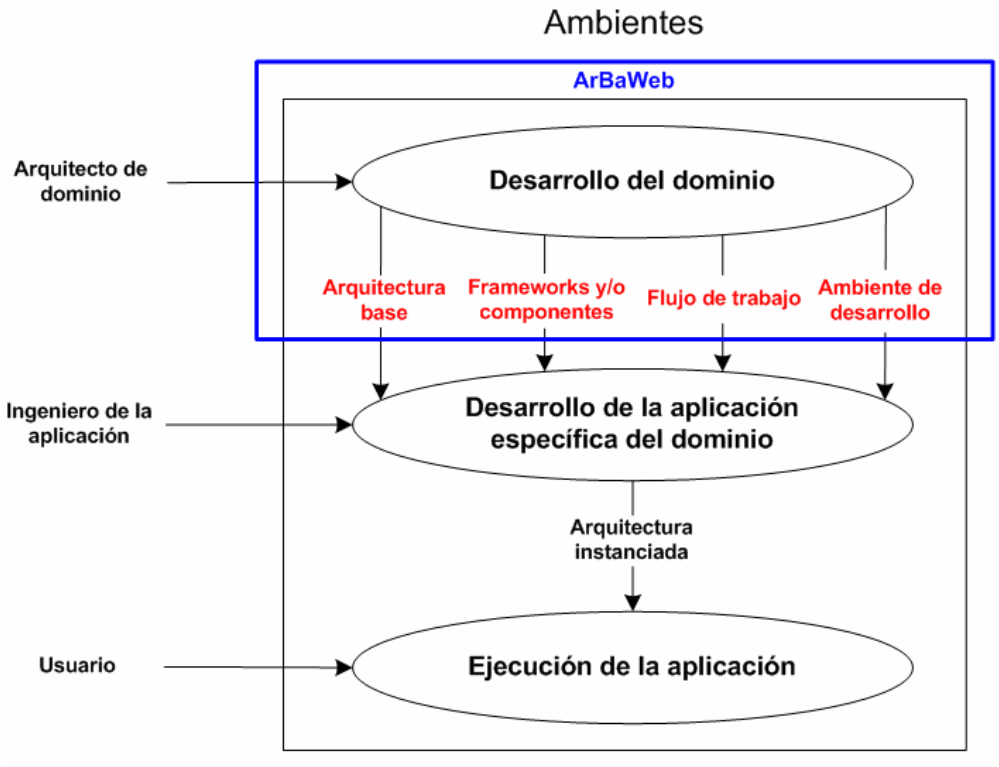

Figura 2-1 Alcance de ArBaWeb

En este capítulo se desarrollará también la arquitectura base propuesta en ArBaWeb, la cual se descompone en cuatro aspectos: el diseño de las capas lógicas, convenciones o estándares de nombres, estructuras de código y mecanismo de colaboración entre los subsistemas y módulos específicos de la aplicación.

2.2 Definición del dominio

A continuación definiremos cuál es el dominio en el que se enmarca ArBaWeb, analizando e identificando los problemas comunes entre las aplicaciones del dominio definido (ver sección 2.3).

ArBaWeb se enmarca en todas las aplicaciones web desarrolladas sobre JEE, específicamente las que utilicen el contenedor de inversión de control de Spring Framework e Hibernate para la persistencia. Presenta un framework y un flujo de trabajo asociado que brindan facilidades en el desarrollo de software. Además, para algunos de los requerimientos generales, identificados más adelante, que presentan las aplicaciones dentro de este dominio, ArBaWeb Framework, da soporte a través de un conjunto de tecnologías agilizando y robusteciendo el desarrollo de software (ver sección 3.2 del capítulo 3).

2.3 Requerimientos de referencia del dominio

Muchas aplicaciones se han desarrollado en el ámbito correspondiente al dominio definido anteriormente. En diferentes áreas de estas aplicaciones destaca el uso de patrones, semejantes en necesidades, para problemas similares. Estos problemas repetidos convergen en requerimientos comunes para toda aplicación en este dominio y son expuestos a continuación:

- Seguridad: En la mayoría de las aplicaciones es necesario manejar los requerimientos de Autenticidad, Integridad, Confidencialidad de la información.
- Auditoría: Es necesario tener registrado qué pasa en la aplicación constantemente y quién es el responsable de cada acción realizada.
- Almacenamiento de datos en Base de Datos: Se necesita guardar los datos de la aplicación en bases de datos. Es necesario que este almacenamiento sea óptimo por lo que se deben usar “Pools de Conexiones”.
- Manejo de Transacciones: Se necesitan las operaciones sean transaccionales y en muchos casos éstas se conforman por más de una petición del cliente.

- Capa de presentación flexible y rica en estilos: La capa de presentación debe cumplir esta característica para soportar las exigencias de presentación para los distintos tipos de clientes que puede presentarse en una misma aplicación y las mismas pueden sufrir cambios.
- Alto rendimiento de procesamiento de las peticiones del cliente: Es necesario que las peticiones del cliente se respondan en el menor tiempo posible por lo que es necesario que todas las capas de la aplicación interactúen de la manera más eficiente posible.
- Interacción con aplicaciones externas: Es muy común la interacción con sistemas externos por lo que se hace necesario que existan estándares en el uso de las tecnologías necesarias para cumplir este requerimiento.
- Administración de los aspectos configurables de la aplicación: toda aplicación necesita ser configurable tanto en la etapa de despliegue como en tiempo de desarrollo de la manera más simple posible.

2.4 Arquitectura base

ArBaWeb presenta una arquitectura base para: orientar el diseño de las capas lógicas a través de una propuesta de diseño base; organizar la forma de codificar según las propuestas de convenciones o estándares de códigos y recursos; brindar una estructura física para soportar el código, creando así un esqueleto base; y proponer mecanismos de colaboración entre los componentes integrados en ella. Está basada fundamentalmente en los estilos arquitectónicos Cliente-servidor y Arquitectura en capas.

2.4.1 Diseño de las capas lógicas

La estructura del diseño de las capas lógicas se representa en la figura 2-2.

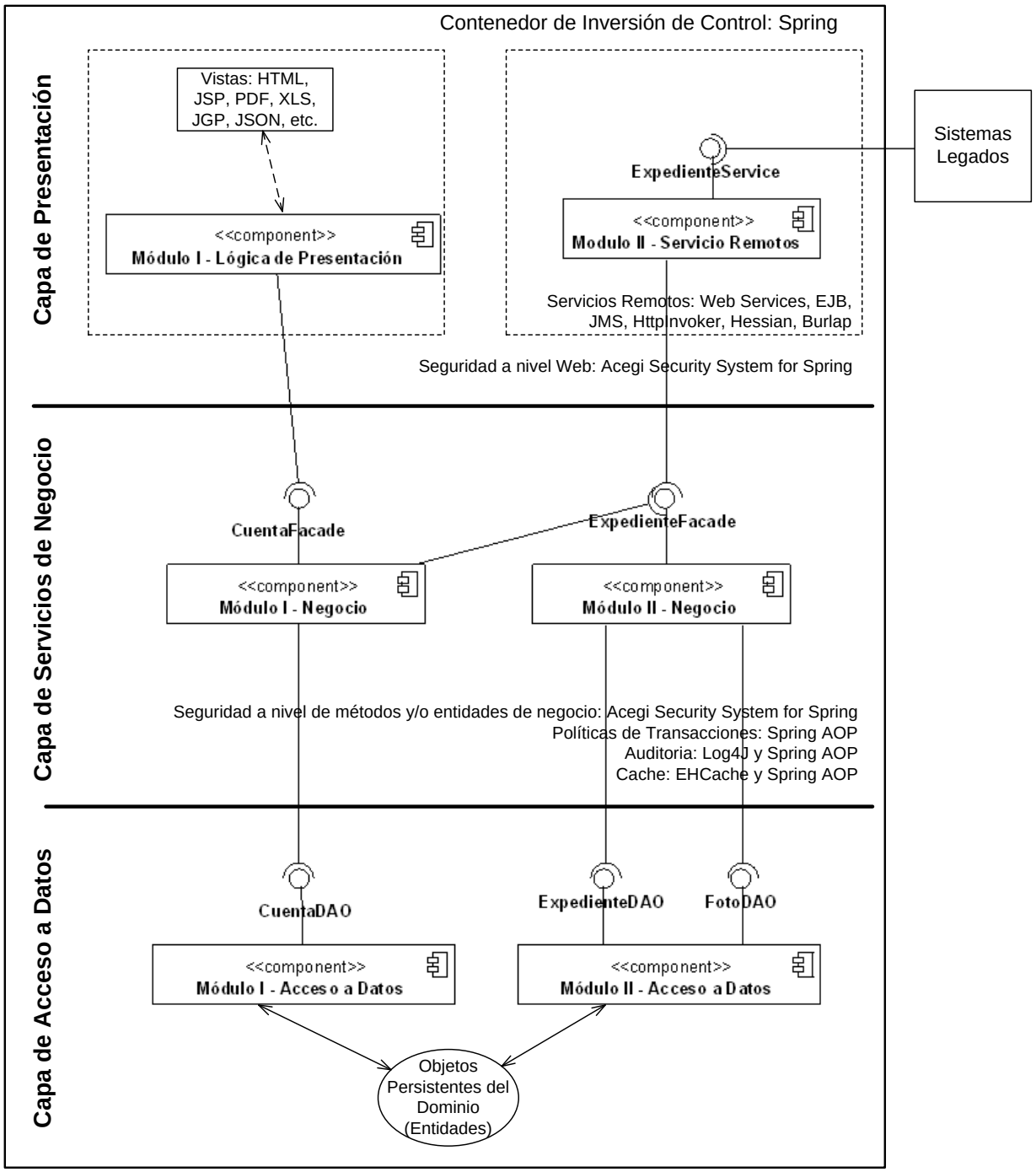

Figura 2-2 Arquitectura de las capas lógicas de ArBaWeb arquitectura

Un concepto tratado en la figura 2-2, es “objetos persistentes del dominio (entidades)”. El cual representa el modelo de objetos o conceptos reales tales como una cuenta bancaria o un expediente escolar. Este conjunto de objetos puede llegar a considerarse otra capa lógica que puede presentar este tipo de arquitectura. Estos objetos de dominios tradicionalmente son pasados hasta la capa de presentación, en la cual mostrarán los datos que contienen, pero no podrán ser modificados, ya que esto solo ocurre solamente dentro de los contextos transaccionales definidos en la capa de servicios de negocio. De esta manera no hay necesidad de uso de los Transfer Objects (DEEPAK ALUR 2003), como se usa en las arquitecturas tradicionales de JEE.

En una aplicación web sobre JEE, que use ArBaWeb, todas sus capas lógicas corren en un servidor web o un servidor de aplicaciones. Es decir, ArBaWeb no es una arquitectura distribuida, pudiera adaptarse a un sistema que lo requiera pero no es recomendado, no está concebida para sistemas distribuidos.

ArBaWeb ayuda a desacoplar las capas arquitectónicas. De esta forma cada capa puede ser modificada tanto como sea posible sin provocar un impacto en las demás capas. Una capa no es consciente de lo que le ocurre a la capa superior, su dependencia es puramente con la capa inmediata inferior. Esta dependencia entre capas es normalmente entre interfaces, asegurando que el acople sea el más bajo posible.

Seguidamente se analizará en más detalles cada una de las capas propuestas anteriormente comenzando de abajo hacia arriba según la figura 2-2.

Capa de acceso a datos

Manteniendo la filosofía de que es mejor programar orientado a interfaces que a clases, ArBaWeb Framework, a través de Spring Framework, soporta el uso de interfaces de acceso a datos entre la capa de servicios de negocio y el API de persistencia utilizado.

El término de Objeto de Acceso a Datos o en inglés, Data Access Object (DAO) (DEEPAK ALUR 2003), es ampliamente usado en el desarrollo de software.

Los DAOs encapsulan la persistencia de los objetos de dominios, proveen la persistencia de los objetos transitorios y las actualizaciones de los objetos existentes en la base de datos. Las implementaciones de los DAOs estarán disponibles para los objetos (típicamente para los objetos de negocio) haciendo uso de

la inyección de dependencias con los objetos de negocio y las instancias de los DAOs, configurada en el contenedor de inversión de control de Spring Framework.

Las interfaces de los DAOs contienen básicamente los siguientes tipos de métodos:

- **Métodos para descubrir:** Estos localizan los objetos almacenados para ser usados por la capa de servicios de negocio.
- **Métodos para persistir o salvar:** Estos hacen persistentes a los objetos transitorios.
- **Métodos para eliminar:** Estos eliminan a los objetos guardados en el medio de almacenamiento (generalmente una base de datos).
- **Métodos para conteos y otras funciones agregadas:** Estos devuelven los resultados de operaciones que son más eficientes implementarlas usando funcionalidades de la base de datos (procedimientos almacenas, etcétera.) que iterar sobre los mismos objetos.

En la figura 2-3, se muestra un ejemplo simple de los elementos que debe presentar un diagrama de clase de esta capa.

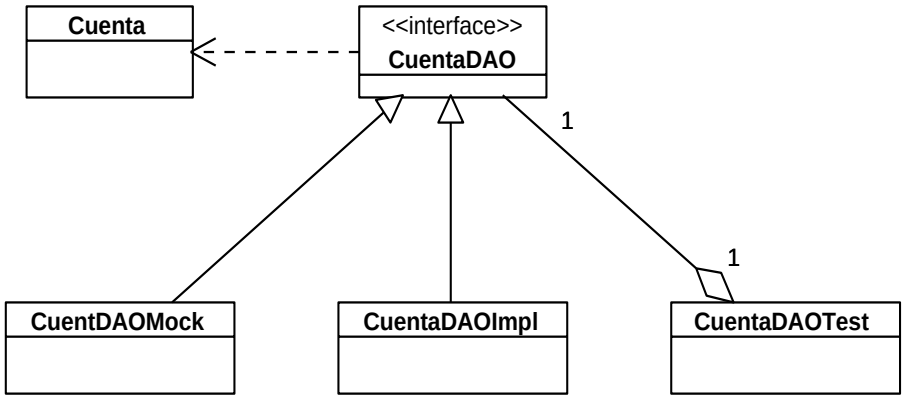

Figura 2-3 Diagrama de clases de la capa de acceso a datos

Los elementos del diagrama de clases según la figura 2-3 son:

- *Cuenta:* es la entidad de dominio a ser persistida por los métodos expuesto en la interfaz CuentaDAO.

- *CuentaDAO:* Es la interfaz que expone a la capa de servicios de negocio los métodos del DAO para persistir la entidad de dominio Cuenta.
- *CuentaDAOMock:* Esta clase implementa los métodos expuestos en la interfaz CuentaDAO pero generando datos falsos estáticamente o usando objetos moqueados o falsos, sin conectarse a ninguna base de datos u otra fuente de almacenamiento. Más adelante en la sección de Flujo de Trabajo se demostrará su uso e importancia en el desarrollo de software.
- *CuentaDAOImpl:* Esta clase es la implementación de la interfaz que realmente se conecta a la base de datos usando Hibernate (puede usarse otro ORM o JDBC directamente). Dando de esta forma la funcionalidad real a los métodos de CuentaDAO.
- *CuentaDAOTest:* Es la clase que prueba todos los métodos de la clase CuentaDAOImpl, utilizando valores que testeen casos extremos. Además, es la responsable de asegurar que los métodos del objeto de acceso a datos están completamente funcionales, para utilizar el DAO desde la capa de servicios de negocio. Al igual que la clase CuentaDAOMock, en próximas secciones se explicará su uso e importancia en el desarrollo de software.

Capa de servicios de negocio

A pesar de usar la arquitectura que define ArBaWeb, los objetos de dominio pueden tener lógica de negocio. Sin embargo, la capa de servicios de negocio tiene un rol importante. En esta capa radican los objetos de negocio o Business Objects (DEEPAK ALUR 2003). Los objetos de negocio separan los datos y la lógica de negocio usando un modelo de objetos.

Estas son las funcionalidades que presenta esta capa:

- **Lógica de negocio específica de procesos de negocios:** A veces es más oportuno para los objetos de dominio contener lógica de negocio aplicable a muchos casos de uso específicos. Sin embargo, existen casos de usos específicos que son realizados en la capa de servicios de negocios. Pero se propone que en esta definición de arquitectura los objetos de dominio no presenten ningún tipo de lógica de negocio, sino que esta responsabilidad recaiga sobre los objetos de negocio, permitiendo usar a los objetos de dominio como Transfer Objects que se mueven entre las capas arquitectónicas de la aplicación.

- **Puntos de entrada muy bien definidos para las operaciones de negocio implementadas:** Los objetos de negocio brindan las interfaces usadas por la capa de presentación.
- **Control de transacciones:** Aunque a veces la lógica de negocio pueda ser movida a objetos de dominio, el control de transacciones no debería. Sino que las políticas transaccionales de la aplicación deberían ser planteadas al nivel de los objetos de negocio.
- **Ejecución de restricciones de seguridad:** Las restricciones de seguridad en esta capa es en los puntos de entradas a la capa media, es decir en los objetos de negocio.

En la figura 2-4 se muestra en más detalles el diseño básico de clases en esta capa, semejante a como se hizo con la capa explicada anteriormente. Es importante resaltar que estos diseños básicos de clases de las capas propuestos por ArBaWeb, pueden ser enriquecidos a través del desarrollo de software de una aplicación específica del dominio, según sus requerimientos. Se pretende lograr la estructuración básica de las aplicaciones a nivel de diseño que utilicen la arquitectura base propuesta en ArBaWeb, orientando a los diseñadores en cómo deben realizar el diseño de cada una de las capas lógicas de la aplicación.

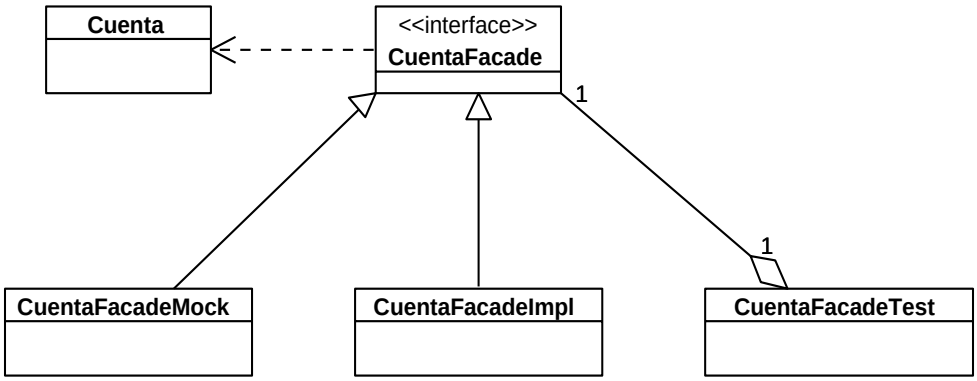

Figura 2-4 Ejemplo de elementos que conforman la capa de negocio

Los elementos que intervienen en la figura 2-4, tienen un comportamiento parecido, pero no exactamente igual al de la figura 2-3:

- *Cuenta:* Es la entidad de dominio que utilizarán los métodos expuestos en la interfaz de negocio CuentaFacade.

- *CuentaFacade:* Es la interfaz que expone a la capa de presentación los métodos para las operaciones de negocio sobre la entidad de dominio Cuenta. A modo de aclaración: puede ocurrir que un objeto de negocio exponga métodos de negocio de varias entidades del dominio o de procesos específicos de negocio.
- *CuentaFacadeMock:* Esta clase implementa los métodos expuestos en la interfaz CuentaFacade, pero generando datos falsos estáticamente o usando objetos moqueados o falsos, sin realizar ninguna operación de negocio real y sin utilizar ningún DAO real, en última instancia usando los DAOs falsos.
- *CuentaFacadeImpl:* Esta clase es la implementación de la interfaz de negocio desarrollando los métodos que contendrán la lógica de negocio.
- *CuentaFacadeTest:* Es la clase que prueba todos los métodos de la clase CuentaFacadeImpl, utilizando valores que prueben los casos extremos. Esta clase es la responsable de asegurar que la lógica de negocio implementada en el objeto CuentaFacadeImpl responde a los requerimientos que debe cumplir. En estas pruebas el objeto de negocio (CuentaFacadeImpl), no tiene necesariamente que usar los DAOs reales, sino los DAOs falsos, ya que el objetivo de estas pruebas no es probar la parte de persistencia de las operaciones de negocio implementadas.

Capa de presentación

En ArBaWeb la capa de presentación descansa sobre una capa de servicios de negocio. Esto significa que esta capa será fina y no contendrá lógica de negocio, sino simplemente lo concerniente a aspectos de presentación, por ejemplo, el código para manipular las interacciones web. Por tanto, se pueden elegir más de una capa de presentación en la misma aplicación, sin impactar en las capas arquitectónicas inferiores, las cuales no deberían tener conocimiento sobre la capa de presentación.

Capa Web

Una aplicación web que use tradicionalmente Spring Framework, la capa web usa un framework web, dentro de los que existen en la comunidad de JEE, pero ArBaWeb Framework presenta un conjunto de ventajas explicadas mas adelantes al usar Spring Web MVC Framework.

Esta capa web es la responsable de tratar con las interacciones del usuario y obtener los datos que pueden ser mostrados en un formato determinado. Normalmente está compuesta por tres tipos de objetos:

- **Controladores:** Estos objetos son responsables de procesar las entradas del usuario en forma de peticiones HTTP, invocando las funcionalidades necesarias, expuestas por la capa de servicios de negocio y devolviendo un *modelo* requerido para ser mostrado.
- **Modelo:** Estos objetos contienen los datos resultantes de la ejecución de la lógica de negocio, los cuales deberían ser mostrados en la respuesta.
- **Vistas:** Estos objetos son responsables de mostrar el modelo en la respuesta de la petición. La forma de mostrar el modelo podrá ser de diferentes tipos de vistas, por ejemplo, archivos JSP, HTML, PDF, documentos de Excel, etcétera. Las vistas no son responsables de modificar los datos o incluso de obtener los datos; estas simplemente sirven para mostrar los dados del modelo que han sido suministrados por un controlador.

Esta combinación de estos tres objetos, es la aplicación el patrón arquitectónico llamado MVC (Model View Controller) (FLOWER 2002).

Fachadas de Servicios Remotos

ArBaWeb, al combinarse con Spring Framework, refleja la filosofía de acceso remoto, por ejemplo, el soporte para clientes remotos a través de invocación a métodos remotos o Remote Method Invocation (RMI), a través de clientes remotos mediante servicios web u otros protocolos de serialización de XML como HTTPInvoker, Hessian, Burlap. Esto debería ser visto como una capa de presentación alternativa sobre las interfaces bien definidas de la capa de servicios de negocio.

A diferencia de la práctica en las tradicionales aplicaciones en JEE, las responsabilidades de acceso remoto no deberían expandirse por toda la aplicación en forma de Transfer Objects. Las aplicaciones deberían ser internamente orientadas a objetos y como el acceso remoto tiende a ser destructivo a la filosofía Orientada a Objetos, este debería ser tratado como una vista de una aplicación. (JOHNSON 2005)

2.4.2 Convenciones o estándares de códigos y recursos

Con el fin de lograr un lenguaje común y uniforme para las distintas aplicaciones que utilicen la arquitectura base definida en ArBaWeb, se especifican convenciones de nombres y estándares de código para los distintos recursos las aplicaciones.

Para las clases Java se adoptan las convenciones estándares presentadas en la Especificación del Lenguaje Java por la compañía Sun Microsystem.

Se definen convenciones de nombres para las distintas clases java dependiendo de las funciones que tengan cada una de estas en la aplicación:

- **Clases de la capa de Acceso a Datos:**
  - Las interfaces que representan las operaciones sobre los objetos de acceso a datos, correspondientes al patrón de diseño Data Access Object (DEEPAK ALUR 2003) terminan con la palabra “DAO”. Ejemplo: EstudianteDAO.
  - Las implementaciones reales de las interfaces DAO comienza con el nombre de la interfaz correspondiente y terminan con la palabra “Impl”. Ejemplo: EstudianteDAOImpl.
  - Las implementaciones falsas de las interfaces DAO comienzan con el nombre de la interfaz correspondiente y terminan con la palabra “Mock”. Ejemplo: EstudianteDAOMock.
  - Las clases utilizadas para realizar pruebas a los interfaces DAO comienzan con el nombre de la interfaz correspondiente y terminan con la palabra “Test”. EstudianteDAOTest.
- **Clases de la capa de Negocio:**
  - Las interfaces que representan las operaciones del negocio terminarán con la palabra “Facade”. Ejemplo: EstudianteFacade.
  - Las implementaciones de las interfaces de negocio comenzarán con el nombre de la interfaz correspondiente y terminaran con la palabra “Impl”. Ejemplo: EstudianteFacadeImpl.
  - Las implementaciones falsas de las interfaces de negocio comienzan con el nombre de la interfaz correspondiente y terminan en la palabra “Mock”. Ejemplo: EstudianteFacadeMock
  - Las clases utilizadas para probar las funcionalidades del negocio terminan con la palabra “Test”. Ejemplo: EstudianteFacadeTest.
- **Clases de la capa de Presentación:**
  - ✓ Capa Web:

- Las clases que manejan el flujo de la capa de presentación, es decir, los controladores, terminarán con la palabra “Controller”.
- Las clases utilizadas para hacer pruebas de unidad a los Controladores comenzarán con el nombre del controlador correspondiente y terminará con la palabra “Test”.
- Las clases utilizadas para validar datos (Validadores) terminarán con la palabra “Validator”.
- Las clases utilizadas para guardar datos procedentes de las vistas, utilizando el patrón Command (GRAND) definido en los patrones GoF, tendrán un nombre lógico de acuerdo a los datos que contiene seguido de la palabra “Command”.

✓ Fachadas de Servicios Remotos:

- Las interfaces que representan las operaciones de negocio que se van a exponer o consumir como servicio terminarán en la palabra “Service”. Ejemplo: EstudianteService.
- Las implementaciones de las interfaces de servicio que expondrán las funciones de negocio y las implementaciones que se van a consumir de un servicio, comenzarán con el nombre de la interfaz correspondiente y terminarán con la palabra “Impl”. EstudianteServiceImpl.
- Las implementaciones falsas de las interfaces para consumir los servicios comenzarán con el nombre de la interfaz correspondiente y terminarán en la palabra “Mock”. Ejemplo: EstudianteServiceMock.

Recursos

- Las páginas HTML que constituyen recursos estáticos tendrán la extensión “.html”.
- Las páginas HTML que se mapean como URL en los *Handler Mappings* terminarán en “htm” para un uso correcto del UniqueServlet (ver sección 3.2).
- Las imágenes en formato JPEG que son estáticas en la aplicación terminarán “.jpg”.
- Las imágenes en formato JPEG que son generadas dinámicamente terminarán con la palabra “.jpeg” para un uso correcto del UniqueServlet (ver sección 3.2).

ArBaWeb también define convenciones de nombre para archivos que tienen que ver con la configuración de la aplicación, los *"Application-Context" (AppContext)* de Spring Framework, archivos utilizados para la internacionalización, ficheros *".properties"* o cualquier otro archivo de configuración.

Los AppContext tendrán la siguiente estructura:

[nombre del proyecto]-[subsistema]-[módulo]-[capa lógica]-context-[tipo].xml

- [nombre del proyecto]: representa el nombre del proyecto en abreviatura. Ejemplo: sigep, cc (correos de cuba).
- [subsistema], [módulo]: indican las unidades organizacionales utilizadas en el proyecto de forma de árbol, pueden ser tantas como la complejidad del mismo lo requiera según las estructuras definidas para cada nivel de complejidad.
- [capa lógica]: representa la capa lógica que maneja del *Application-Context* de acuerdo al tipo de *beans* que se manejan en él. Se establecen las siguientes clasificaciones explicada a continuación:
    - [dataaccess]: Contiene los beans con las funcionales referentes a la capa de acceso a datos.
    - [servlet]: Encapsula todo los beans de la capa de presentación. Utilizando Spring MVC aquí se configuran componentes como Controladores, Handler Mappings, View Resolvers y todos los utilizados para conformar la lógica de esta capa (ver sección 3.2).
    - [remoting]: Contiene los beans que interactuaran con aplicaciones externas y residirá toda la lógica necesaria para que se pueda tanto exponer servicios a través de diferentes protocolos, como consumirlos de sistemas externos.
    - Cuando no se especifica el tipo de AppContext se toma por defecto que se refiere a las operaciones de negocio de la aplicación, en este XML se definen las fachadas y otros objetos que representen la capa de servicios.
- [tipo]: Se refiere al tipo de función que cumplirá este AppContext en la aplicación, si es un AppContext utilizado para definir pruebas a la capa correspondiente el valor sería "test", cuando se utiliza para configurar los beans falsos sería "mock", y en caso de ser un AppContext de uso final en la aplicación no se especificaría el uso.

Los archivos de propiedades utilizados en la Internacionalización están regidos por la siguiente nomenclatura:

[nombre del proyecto]-[subsistema]-[módulo]-[nombre lógico]_[lenguaje]_[País].properties.

- [nombre lógico] se refiere a un nombre que agrupe las propiedades de internacionalización que contenga lógicamente, por ejemplo “universidad” donde estarían todas las propiedades que puedan variar de idioma en la misma, por ejemplo, clase, estudiante, etcétera.
- [Lenguaje]_[País] [dar referencia a una página o documento donde esto se especifique]. Por ejemplo, “es_CU” que sería idioma Español y país Cuba.

2.4.3 Estructuras de código

Teniendo en cuenta que el dominio de la arquitectura propuesta en ArBaWeb se enmarca en las aplicaciones web sobre JEE que utilicen Spring Framework e Hibernate, se deriva que toda aplicación que la utilice como arquitectura base se regirán por las especificaciones estándares definidas en JEE para las aplicaciones web. Esta arquitectura define una estructura que permite organizar cada tipo de clase o componente necesario en el desarrollo de software.

Organización de la aplicación en unidades organizacionales.

Con el objetivo de dividir y organizar las aplicaciones se crean paquetes por cada unidad organizacional que se defina. En la arquitectura propuesta se definen estas unidades: subsistemas y módulos. Un subsistema se refiere a un conjunto de funcionalidades del sistema que tienen características muy propias y que a su vez están subdivididas en módulos. Un módulo encapsula un conjunto de funciones que debe realizar el sistema, las cuales son agrupadas por tener características muy similares y se definen en la etapa de diseño. El uso de subsistemas y módulos en la aplicación final está regido atendiendo a la complejidad que tenga la misma de acuerdo a las clasificaciones definidas en ArBaWeb.

Clasificaciones de complejidad.

Se crearon tres clasificaciones atendiendo a la complejidad que podría presentar el proyecto a realizar: baja, mediana y alta. Para cada una define una estructura de paquetes correspondiente atendiendo a las posibles necesidades que pueda presentar.

- **Baja:** Se define para soportar las necesidades de las aplicaciones que sean de tamaño pequeño debido a que presentan un solo módulo. Esta contiene el paquete raíz[15], el nombre del módulo y a continuación toda la estructura referente al mismo (Figura 2-5).

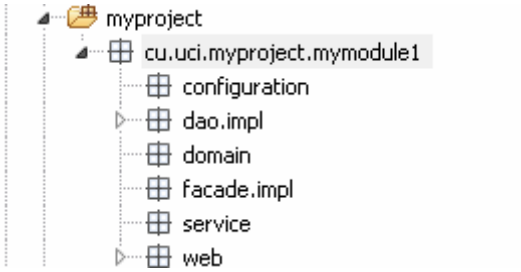

Figura 2-5 Representación de la clasificación de complejidad baja

- **Mediana:** Responde a sistemas no muy grandes, que contengan solo un conjunto de módulos y no sea necesario definir subsistemas. Esta estructura está compuesta por el paquete raíz, seguido en el árbol por los paquetes de los módulos definidos (Figura 2-6).

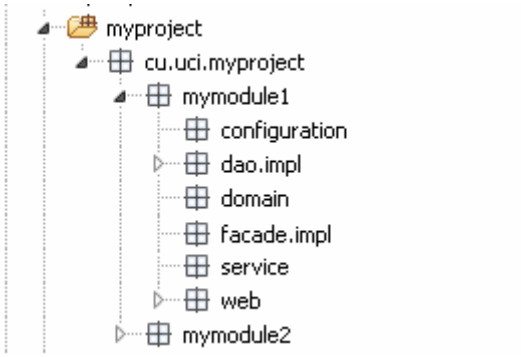

Figura 2-6 Representación de la clasificación de complejidad mediana

- **Alta:** Responde a sistemas complejos, que contengan subsistemas y módulos, permitiendo además que se adicionen otros niveles organizacionales definidos por el equipo de arquitectura. Esta estructura está compuesta por el paquete raíz, seguido por los paquetes referentes a los

[15] Paquete raíz: es el paquete de clases en el cual residen los paquetes que componen los subsistemas, módulos y componentes de una aplicación, por ejemplo: org.myapp.

subsistemas. Cada subsistema tendrá como hijos en el árbol de paquetes a los módulos que responden al mismo (Figura 2-7).

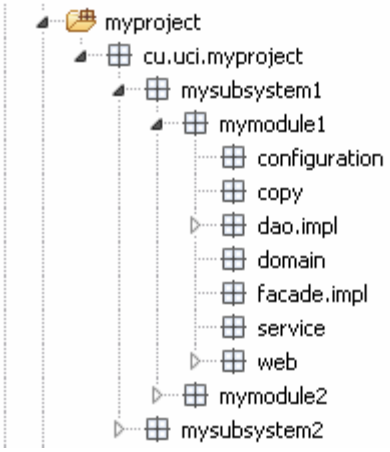

Figura 2-7 Representación de la clasificación de complejidad alta

Estructura más atómica en organización de paquetes

La unidad más atómica en la estructuración de paquetes que se define en ArBaWeb como referencia es el módulo. Un módulo se puede comparar con una pequeña máquina que puede actuar por sí sola, siempre y cuando estén activas las demás máquinas de las que esta depende para su funcionamiento. En un módulo generalmente existen todas las capas lógicas que se definen en la arquitectura base propuesta en ArBaWeb. Cada componente que se desarrolle en un módulo tiene un lugar definido en esta estructura de paquete acorde a la complejidad de la aplicación. A continuación se expone esta estructura (Figura 2-8).

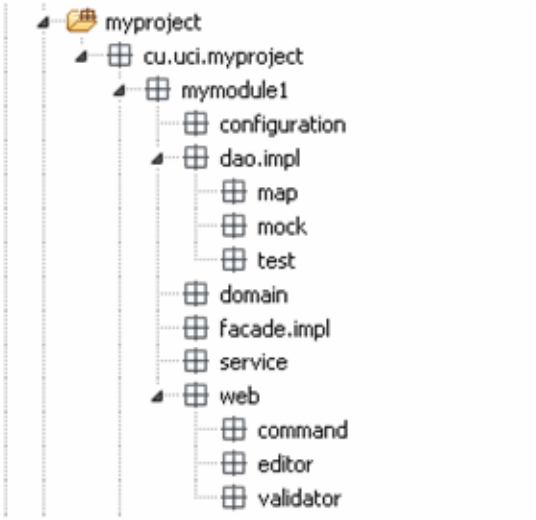

Figura 2-8 Representación de la estructura de paquete en ArBaWeb

configuration: Se localizarán todos los archivos que tienen que ver con la configuración del módulo, los "*Application-Context"* de Spring Framework, los archivos utilizados para la internacionalización, ficheros de propiedades *".properties"* y cualquier otro destinado a estos fines.

dao: Se encontrarán las interfaces DAOs.

dao.impl: Se encontrarán las implementaciones de las interfaces DAOs.

dao.map: Se encontrarán los ficheros de mapeo utilizados por Hibernate.

dao.test: Se encontrarán las clases de prueba utilizadas para realizar las pruebas de unidad a las implementaciones de los DAO.

domain: Se encontrarán todas las entidades persistentes o no del dominio pertenecientes al módulo.

facade: Se encontrarán las interfaces de las fachadas de negocio.

facade.impl: Se encontrarán las implementaciones de las fachadas de negocio.

service: Se encontrarán las interfaces que representan las operaciones de negocio que se van a exponer o consumir como servicio.

web: Se encontrarán los controladores de la capa de presentación.

web.command: Se encontrarán las clases utilizadas para guardar datos procedentes de las peticiones web (Command).

web.editor: Se encontrarán los *PropertyEditors*[16].

web.validator: Se encontrarán las clases utilizadas para validar datos (Validadores).

2.4.4 Mecanismos de colaboración

Los mecanismos de colaboración son estrategias propuestas en ArBaWeb para establecer la forma de comunicación entre los componentes del software. Según las estructuras de código planteadas en la sección anterior, que dividen al sistema en estructuras físicas denominadas subsistemas y módulos de acuerdo con la complejidad del software (baja, media o alta), se proponen las estrategias de comunicación o colaboración entre ellos.

Como ya se ha explicado, un subsistema agrupa una colección de módulos. Por consiguiente, si se detecta que entre los módulos de un mismo subsistema existen dependencias entre ellos, es decir, que algunos módulos necesitan funcionalidades implementadas en otros, entonces hay dos posibles propuestas de estrategia de colaboración a usar. Un módulo se puede ver como la encapsulación a nivel de componente de software de un conjunto de casos de usos

Dependencia directa

La dependencia directa es cuando un módulo específico de la aplicación tiene funcionalidades expuestas en una o más interfaces que otros módulos necesitan y éstos las utilizan directamente. En la figura 2-9 se muestra un ejemplo de las dependencias directas que existen entre el Módulo I y Módulo III con el Módulo II mediante la interfaz ExpedienteFacade. Esta figura es un ejemplo de una de las estrategias de colaboración entre módulos de un sistema software.

[16] PropertyEditor: En java es utilizado para convertir un bean estructurado en forma de cadena de texto a un objeto de su correspondiente clase. Por ejemplo, un número de teléfono escrito en una cadena de texto con un formato específico sería convertido a un objeto de la clase Phone.

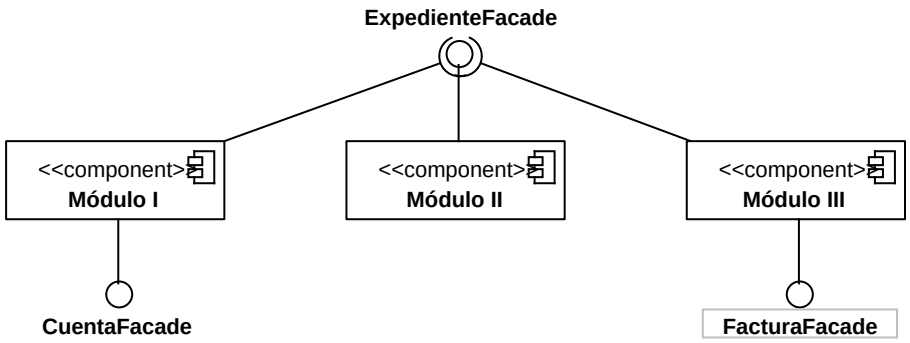

Figura 2-9 Representación de dependencias directas

Dependencia indirecta

La dependencia indirecta es cuando existen dependencias directas entre varios módulos, y no se desea que existan debido a que su impacto arquitectónico no ponga en riesgo algún requerimiento funcional o no funcional de la aplicación.

Tomando como ejemplo la figura 2-9, se desea eliminar las dependencias directas que existen con el Módulo II a través de las funcionalidades expuestas en la interfaz ExpedienteFacade, porque como se muestra en la figura, el Módulo II es imprescindible para los otros dos módulos ya que estos necesitan de la interfaz ExpedienteFacade; y la aplicación tiene como uno de sus requerimientos no funcionales que en algunos casos la aplicación no contenga el Módulo II.

Para darle solución a este problema, como muestra la figura 2-10, se crea un nuevo módulo llamado “common” (“común”, en español), para el que se moverá la interfaz ExpedienteFacade y todas las clases y recursos implicados en soportar las funcionalidades expuestas por la interfaz. De esta forma, los módulos que antes dependían directamente del Modulo II a causa de usar las funcionalidades de la interfaz ExpedienteFacade (ver figura 2-9), ya no lo harán, sino que tendrán dependencia directa del módulo común (ver figura 2-10), por lo que se logro cumplir con el requerimiento no funcional que presentaba la aplicación de poder eliminar en algún momento el Modulo II y los demás módulos continuaran su funcionamiento sin problemas.

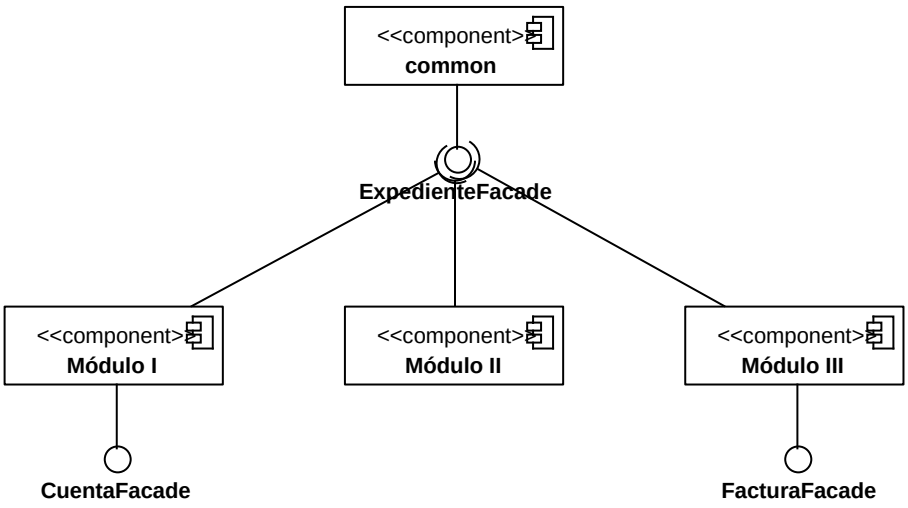

Figura 2-10 Ejemplo de dependencia directa del módulo common

Este módulo común, tiene la misma estructura básica de los módulos planteada en la sección anterior. Es un módulo que no se identificó inicialmente cuando se diseño el sistema, pero fue necesaria su creación para lograr eliminar dependencias directas entre los módulos específicos de la aplicación y moverlas hacia un módulo común para cumplir con requerimientos específicos de la aplicación.

Todo este proceso es entre los módulos específicos de un subsistema por lo que el módulo común creado pertenece también al subsistema. Además, este proceso también es aplicable para establecer este mecanismo de colaboración a nivel de subsistemas.

La comunicación entre módulos y entre subsistemas es a través de las interfaces de la capa de servicios de negocio, que son puntos de entradas a la lógica de negocio implementada en ellos. De esta forma se logra un mayor desacople entre ellos, al ocultar de tras de las interfaces las implementaciones reales de cada funcionalidad.

2.5 Conclusiones

Con el desarrollo de este capítulo se ha llegado a la definición del dominio en que se enmarca ArBaWeb y los requerimientos de referencias identificados en este dominio. Además, se expuso la propuesta de arquitectura base que contiene ArBaWeb, permitiendo de esta forma realizar el diseño de las capas lógicas de una aplicación guiados por esta arquitectura base. Se ha explicado la propuesta de convenciones de nombres o estándares de codificación tanto para códigos fuentes y recursos, las estructuras de código según la clasificación de la complejidad que presentan las aplicaciones y los mecanismos de colaboración entre las distintas divisiones organizacionales propuestas en ArBaWeb.

Capítulo 3. ArBaWeb, fundamentos para desarrollar

3.1 Introducción

En este capítulo se explicará a grandes rasgos las características de **ArBaWeb Framework**. El cual está compuesto por componentes que permiten satisfacer los requerimientos de referencias definidos en el capítulo 2. ArBaWeb Framework debe ser configurado para cada una de las aplicaciones que lo usen.

Además en este capítulo se define un flujo de trabajo, el cual permite guiar, paso a paso, el desarrollo de las aplicaciones. Además, presenta una propuesta de ambiente de desarrollo definiendo las herramientas a usar en el proceso de desarrollo y las tecnologías en que se apoyará la construcción del software.

3.2 ArBaWeb Framework

Los *frameworks de dominios* son usados para implementar aplicaciones que pertenecen al mismo dominio. Las aplicaciones de dominio específico habitualmente son hechas a la medida para una organización en particular y a menudo son desarrolladas desde cero. Los frameworks pueden ayudar a reducir gran cantidad de trabajo que se necesita para implementar estas aplicaciones. Estos permiten una organización del proceso de desarrollo de software de alta calidad para el dominio en que está enmarcado, mientras que reducen el tiempo de desarrollo y salida del software al mercado. (KAISER 2005)

ArBaWeb Framework es un framework de dominio, basado en Spring Framework e Hibernate; lo que permite utilizar todas las funcionalidades y características explicadas en el capítulo 1 de cada uno de estos frameworks. ArBaWeb Framework está compuesto por 4 módulos (ver figura 3-1) que dan soporte a los requerimientos definidos en la sección 2.3. Además brinda facilidades en el uso de Spring Framework e Hibernate, incluyendo funcionalidades adicionales a otros framework, permitiendo el rápido desarrollo de software.

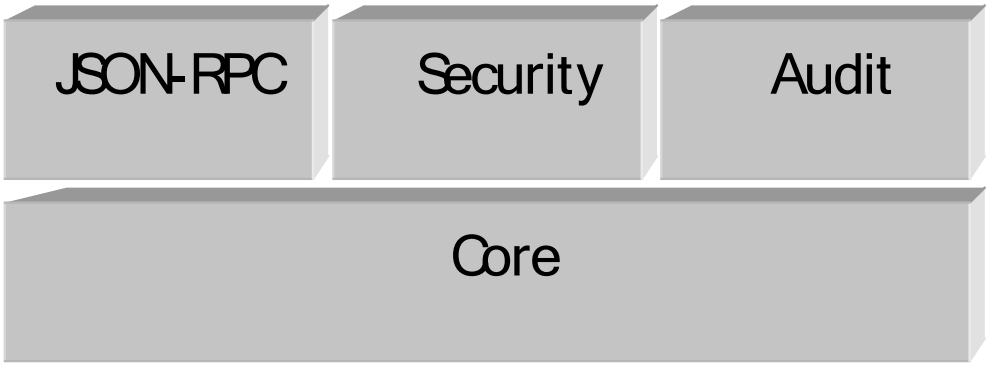

Figura 3-1 Módulos de ArBaWeb Framework

El módulo *Core* es el módulo principal de ArBaWeb Framework. Este provee todo el mecanismo de configuración del framework, por ejemplo, definir cuáles módulos de la aplicación se cargarán en el momento que sea ejecutada, cargar las propiedades de configuración de los demás módulos integrados en ArBaWeb Framework y de los específicos de la aplicación. Presenta un mecanismo genérico de acceso a datos. Provee soporte para la internacionalización, cargando de forma automática los recursos asociados a esta. Además incluye soporte para la carga de forma automática, basada en patrones de nombres (ver sección 2.4.2) de los AppContext de los módulos definidos en la configuración de la aplicación.

El módulo *Security* da soporte a la seguridad a nivel de peticiones de URLs y métodos, haciendo uso del framework Acegi. Esta seguridad es basada en roles, a los cuales pertenecen los usuarios. Además, para un usuario poder autenticarse necesita de una clave, estar activo y tiene que estar accediendo desde una dirección IP[17] autorizada en el mecanismo de seguridad para el acceso del usuario en específico. También provee soporte con JCaptcha para la verificación de usuarios humanos.

El módulo *Audit* proporciona soporte para establecer la auditoría de la aplicación en la capa web, en los métodos de las fachadas de negocio y en eventos de la aplicación que lo requieran.

El módulo *JSON-RPC* usa json-rpc framework para dar soporte a llamadas síncronas o asíncronas de métodos desde el cliente web a los métodos de las fachadas de los objetos de negocio usando JSON como formato en el protocolo de comunicación. Además, brinda un conjunto de facilidades adicionales

[17] Dirección IP: es un número que identifica de manera lógica y jerárquica a una interfaz de un dispositivo, generalmente una computadora, dentro de una red que utilice el protocolo IP (*Internet Protocol*).

para el uso del mismo a través de anotaciones para registrar los objetos que serán expuestos remotamente y para eliminar propiedades de los objetos que devolverán estos métodos que no sean de interés en el cliente web.

El paquete raíz donde residen los módulos definidos en ArBaWeb Framework es *cu.uci.arbaweb.*

Core

Este módulo presenta un conjunto de paquetes de clases dentro del paquete raíz de ArBaWeb Framework, los cuales se muestran en la figura 3-2.

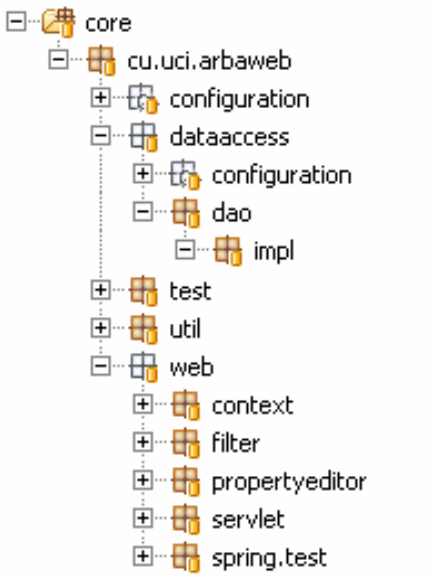

Figura 3-2 Paquete de clases del módulo Core

Seguidamente se describen las responsabilidades de cada uno de los paquetes de clases que conforman el módulo *Core*:

- *configuration:* en este paquete radican todos los AppContext y archivos de propiedades del módulo *Core*.
- *dataaccess:* es el que agrupa los paquetes relacionados con el mecanismo genérico de acceso a datos, los cuales son:

- *configuration:* en este paquete están los AppContext que instancian todos los objetos necesarios para proveer la conexión a la base de datos, así como el pool de conexiones y el administrador de las transacciones (Transaction Manager).
- *dao:* en la raíz de este paquete está la interfaz genérica de acceso a datos. Esta interfaz define las operaciones básicas (Crear, Leer, Actualizar y Eliminar) de todos los objetos de acceso a datos. Estas operaciones serán compartidas por todos los objetos de acceso a datos de la aplicación.
- *dao.impl:* en este paquete está la implementación de la interfaz genérica donde se implementan los métodos de acceso a datos haciendo uso de la genericidad.

- *test:* contiene las clases responsables de las pruebas de unidad haciendo uso de JUnit.
- *util:* contiene un conjunto de paquetes y clases de utilidades para el correcto funcionamiento de ArBaWeb.
- *web:* contiene un conjunto de paquetes y clases que tienen diferentes funcionalidades para el correcto funcionamiento de la capa web. Estas son basadas en Spring Web MVC, además de agregarle un conjunto de funcionalidades. A continuación se describen las responsabilidades de los paquetes contenidos dentro del paquete *web*:
  - *context:* contiene las clases necesarias para cargar los AppContext que encapsulan todas las funcionalidades desde la capa de negocio hacia abajo, según la figura 2-2. Esta carga es de forma automática sin modificar el web.xml de la aplicación, según los módulos definidos en los archivos de configuración.
  - *filter:* encapsula la clase necesaria para cargar todos los filtros definidos en ArBaWeb y por la aplicación sin modificar el web.xml.
  - *propertyeditor:* paquete con funcionalidad de utilidad para la web.
  - *servlet:* paquete que contiene la funcionalidad para cargar los Web Application Context (que encapsulan toda la lógica la capa de presentación) de forma automática sin modificar el web.xml de la aplicación, según los módulos que tengan capa de presentación web definidos en los archivos de configuración.

- *spring.test:* contiene la funcionalidad de ejecutar pruebas de unidad usando JUnit en los controladores de la lógica de presentación de la capa web.

Este módulo presenta tres clases principales, que son las encargadas de levantar todos los módulos de la aplicación, los filtros, etcétera, según la configuración definida:

- *BaseContextLoaderListener:* Es un listener [18]que se declara en el web.xml, como se muestra en la figura 3-3. Este listener es el encargado de levantar la configuración definida en la aplicación. Esto implica cargar los módulos definidos, si se activa o no la seguridad y auditoría. También carga uno de los contextos de acceso a datos definido en ArBaWeb Framework, entre otras configuraciones. Para que pueda cargar la configuración hay que definir un parámetro en el web.xml (*appConfigLocation*), como se muestra en la figura 3-3, donde se pone el camino del archivo de propiedades que tiene los parámetros de configuración de la aplicación. Si este parámetro no se define, entonces el listener toma por defecto que el archivo de propiedades está en la raíz del classpath y se llama *arbaweb.properties*. Para ver los posibles parámetros de este archivo de configuración ver la documentación de la clase *Configuration* presente en ArBaWeb Framework.

```
<context-param>
    <param-name>appConfigLocation</param-name>
    <param-value>
        classpath:cu/uci/arbaweb/samples/configuration/samples.properties
    </param-value>
</context-param>

<listener>
    <listener-class>
        cu.uci.arbaweb.web.context.BaseContextLoaderListener
    </listener-class>
</listener>
```

Figura 3-3 Definición de un parámetro en el web.xml

- *UniqueServlet:* Es un servlet que se declara en el web.xml, ver figura 3-4. Es responsable de cargar los Web Application Context de los módulos web de la aplicación, según los definidos en la configuración.

[18] Listener: un listener es una clase en que se encarga de escuchar los eventos que son lanzados en el sistema.

```
<servlet>
    <servlet-name>SampleServlet</servlet-name>
    <servlet-class>
        cu.uci.arbaweb.web.servlet.UniqueServlet
    </servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
<!--
    Mapeando el UniqueServlet para los tipos de peticiones
    que atenderá el sistema a través del MVC definido.
-->
<servlet-mapping>
    <servlet-name>SampleServlet</servlet-name>
    <url-pattern>*.htm</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>SampleServlet</servlet-name>
    <url-pattern>*.jpeg</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>SampleServlet</servlet-name>
    <url-pattern>*.pdf</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>SampleServlet</servlet-name>
    <url-pattern>*.xls</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>SampleServlet</servlet-name>
    <url-pattern>*.rtf</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>SampleServlet</servlet-name>
    <url-pattern>*.doc</url-pattern>
</servlet-mapping>
    <servlet-name>SampleServlet</servlet-name>
    <url-pattern>/security/j_acegi_security_check</url-pattern>
</servlet-mapping>
```

Figura 3-4 Definición de un servlet declarado en el web.xml

- *UniqueFilter:* Es un filtro web que se define en el web.xml, ver figura 3-5. Tiene la responsabilidad de cargar los filtros definidos en ArBaWeb Framework según la configuración de la seguridad y auditoría, incluyendo a los filtros específicos de la aplicación. Para cargar los filtros definidos en la aplicación es necesarios declararlos como beans en los application context de la aplicación y hacer referencias a ellos en el archivo de configuración, a través de una propiedad que más adelante se explicará.

```
<filter>
    <filter-name>UniqueFilter</filter-name>
    <filter-class>
        cu.uci.arbaweb.web.filter.UniqueFilter
    </filter-class>
</filter>

<filter-mapping>
    <filter-name>UniqueFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

Figura 3-5 Definición de un filtro web declarado en el web.xml

Security

El módulo de seguridad de ArBaWeb Framework se desarrolló con el objetivo de lograr la mayor transparencia posible en el manejo de la seguridad utilizando como base el framework Acegi. Se detectaron los requerimientos principales de seguridad de las aplicaciones en el dominio y se desarrolló una estructura que soporta estos requerimientos con gran simpleza y flexibilidad.

Autenticación

Modelo basado en roles.

En este modelo a los usuarios le son asignados uno o varios roles mientras que los permisos y privilegios se asignan a estos roles. Por tanto, las políticas de control de accesos basado en roles regulan el acceso de los usuarios a la información en términos de sus actividades y funciones de trabajo (roles), representándose así de forma natural la estructura de las organizaciones. Dada la alta integración entre los roles y las responsabilidades de los usuarios, se pueden seguir los principios del mínimo privilegio y de

la separación de responsabilidades. Estos principios son vitales para alcanzar el objetivo de integridad al requerir que a un usuario no se le otorguen mayores privilegios que los necesarios para efectuar su trabajo.

Usuario

Un usuario está compuesto por el nombre de usuario (username), contraseña (password), si está habilitado o no (enabled), dirección IP de la que puede acceder (ip) y la lista de roles a los que pertenece.

- Username: Almacena el nombre del usuario.
- Enabled: Indica si el usuario está habilitado para entrar al sistema o no.
- Password: La contraseña es guardada por defecto utilizando el algoritmo de encriptación irreversible MD5 usando como clave de generación el nombre de usuario. Para cambiar el algoritmo de encriptación de contraseñas se deben redefinir las propiedades "password_encoder_bean" y "password_username_saltsource" en el archivo security.properties (ver comentarios en el archivo).
- IP: Las direcciones IP pueden encontrarse los siguientes formatos separados por coma: (10.1.*.*, 10.2.0.1, 10.2.0.0/16, 10.3.0.1-10.3.0.50).
- Roles: Un usuario puede pertenecer a uno o varios roles en dependencia a cómo se modele la seguridad en cada sistema. Ver figura 3-6.

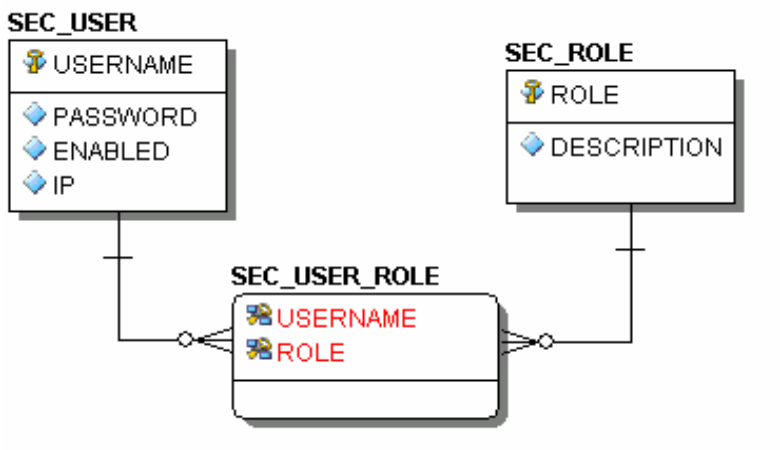

Figura 3-6 Diagrama de clases para la definición de roles

Proceso de autenticación

Para autenticarse al sistema el usuario debe entrar los datos que se requieran para verificar que es quien dice (ver figura 3-7). El mecanismo verifica que todos los datos sean válidos mediante el siguiente procedimiento:

Figura 3-7 Ejemplo de página de autenticación

1. El usuario entra los datos en la página principal de autenticación, a la cual puede llegar tanto por ser la página principal o por tratar de acceder a un recurso que está asegurado y aún no se ha autenticado.
2. Al enviar sus datos (credenciales), son interceptados por el filtro de autenticación de Acegi el cual delega la responsabilidad al "AuthenticationManager" (ver documentación del framework Acegi) y este utiliza el bean de ArBaWeb Framework AuthenticationDaoProvider el cual hereda de la implementación de Acegi (DaoAuthenticationProvider) para adicionar la funcionalidad de verificación de IP.
3. Se comprueba que el IP sea válido para ese usuario, que esté habilitado y que la contraseña coincida con la almacenada en la base de datos.
4. En caso de ser válidas las credenciales se redirecciona al usuario a la página de inicio de la aplicación o en caso de él haber pedido una URL se enviará a la misma.
5. En el caso en que no sean válidas las credenciales se le enviará a la misma página del login con el parámetro *login_success=false* para que verifique sus datos. Las direcciones URL referidas a la

seguridad se pueden configurar en el fichero “security.properties” dentro de la sección “Authentication”.

Un mecanismo adicional en la autenticación es la utilización de Captcha, ver capítulo 1, mediante el framework JCaptcha donde el desarrollador puede definir la cantidad máxima de intentos de autentificación fallidos permitidos.

Figura 3-8 Ejemplo de página de autenticación utilizando Captcha

Personalización

Los datos de los usuarios que se establecen se basan en los requerimientos de seguridad comunes en las aplicaciones web sobre JEE. En la mayoría de las aplicaciones es necesario adicionar datos a los usuario como nombre, apellido, dirección, etcétera; esto es posible y se propone crear una tabla en la base de datos adicional con estos datos que tenga una relación uno a uno con la tabla “*sec_user*”.

En el módulo de seguridad de ArBaWeb por defecto se realiza la autenticación contra los datos almacenados en la base de datos; pero si se quisiera realizar la autenticación desde otra fuente de datos la cual puede ser adicional a la existente se debe editar la propiedad *security_dao_providers* y poner separados por coma y en orden de prioridad la lista los nombres de los DaoProviders que se utilizarán para autenticar. Por ejemplo, si se necesitara que la aplicación autentifique por un servidor LDAP podría adicionarse un nuevo DaoProvider que realice esta funcionalidad y se configuraría utilizando la propiedad mencionada (ver documentación del archivo *security.properties* en la documentación del framework).

Autorización

Seguridad sobre las capas lógicas.

¿A qué se le debe aplicar la seguridad?

En toda aplicación es necesario tener en cuenta para controlar la seguridad todos los posibles lugares por donde el usuario pueda interactuar con las funcionalidades de la misma teniendo en cuenta que solo pueda acceder a las funcionalidades que le son permitidas en la aplicación y asegurando la confidencialidad de la información manejada por cada usuario. En una aplicación web sobre JEE estos puntos de entrada o de control se localizan en las peticiones HTTP a las páginas web y en la invocación a métodos de clases expuestas en forma de servicios remotos.

Para controlar ambos tipos de acceso se utilizaron las soluciones brindados por el framework Acegi con algunas modificaciones para ajustarnos a los requerimientos más específicos de las aplicaciones concentradas en el dominio definido y para buscar mayor simplicidad en el manejo de las mismas.

Peticiones web

Para controlar el acceso a los recursos expuestos por el protocolo HTTP se utiliza la tecnología de Filtros, que permiten interceptar todas las llamadas y respuestas resultantes de la interacción del cliente con la aplicación así como para controlar qué peticiones deben viajar por conexión HTTP segura (Secure Hypertext Transfer Protocol, HTTPS). Ver Figura 3-9.

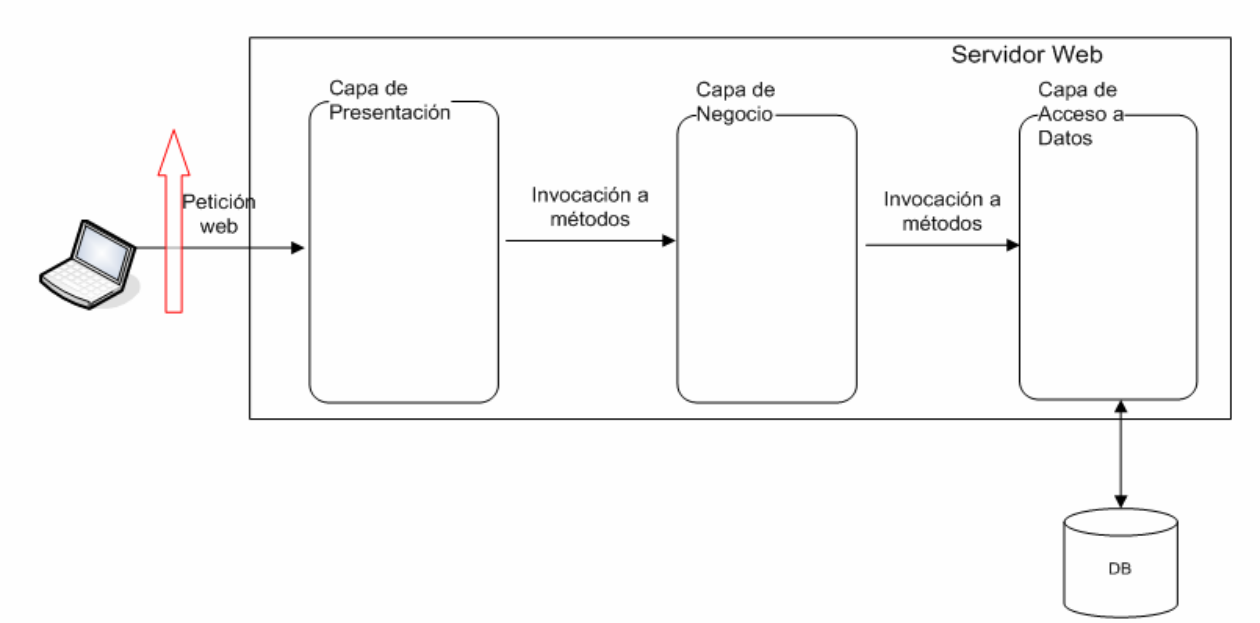

Figura 3-9 Intersección de la seguridad a las peticiones web indicado por la flecha roja

Invocación a métodos expuestos como servicio.

Las clases pueden exponerse como servicios remotos utilizando varias tecnologías y se vuelve complejo el control de los tipos de peticiones que puede realizar el cliente para cada una, por esto se decidió manejar la seguridad a nivel de métodos donde se ignora la tecnología por la cual se exponen los servicios. Para controlar la seguridad a nivel de métodos se utiliza la tecnología de programación orientada aspectos (AOP) mediante la que se pueden controlar las llamadas a métodos. Ver figura 3-10.

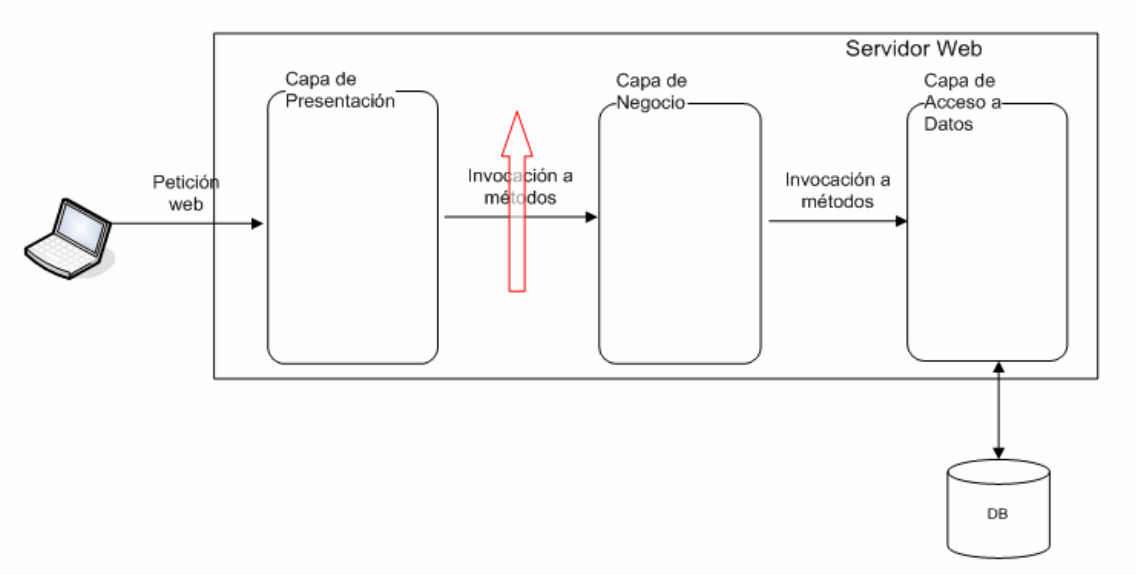

Figura 3-10 Intervención de la seguridad en las llamadas a métodos indicado por la flecha roja

Definición de los accesos

ArBaWeb Framework provee una estructura ya definida con una configuración de seguridad que responde a los requerimientos comunes de las aplicaciones web sobre JEE y que permite que se ajuste a las necesidades específicas de cada aplicación utilizando los ficheros de propiedades *security.properties*, *securityCaptcha.properties* y *securityWeb.properties*.

Una vez personalizada la seguridad solo es necesario configurar los permisos propios del negocio y esto se puede realizar utilizando básicamente las estructuras de configuraciones de permisos que brinda el framework Acegi donde se puede definir qué roles pueden acceder a qué recursos.

Ejemplo: /mymodule/index.htm=ROLE_USER, ROLE_ADMIN

Para enriquecer las configuraciones de permisos que provee Acegi y para ganar en portabilidad se definieron nuevas estructuras. Estas nuevas estructuras para la configuración de permisos pueden almacenarse tanto en ficheros XML como en la base de datos y soportan nuevos parámetros para especificar además qué usuarios pueden acceder a qué recursos, así como qué roles o usuarios no pueden acceder al mismo utilizando el símbolo “-“ delante. A continuación se exponen las dos estructuras.

- XML: Para definir las configuraciones de seguridad en XML se creó el XSD [19] cu.uci.arbaweb.security.configuration.arbaweb-security.xsd. Por cada módulo se puede definir varios ficheros de este tipo terminando "–security-definition.xml" en los cuales reside toda la configuración de acceso referente al módulo (ver figura 3-11). Definir de la seguridad en el XML permite beneficiarse del completamiento de código y tiene la ventaja de que cada módulo contiene su propia configuración de seguridad independiente de las demás.

```
<?xml version="1.0" encoding="UTF-8"?>
<appSecurity xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation=
    "../../../../../../uci/arbaweb/security/configuration/arbaweb-security.xsd">

    <security type="url">
        <module name="om">
            <def pattern="/module1/mipagina.htm">
                <role name="ROLE_ADMIN" />
            </def>
        </module>
    </security>
    <security type="method">
        <module name="om">
            <def
                pattern="cu.ecc.ipostel.psp.om.facade.impl.TestFacadeImpl.*">
                <role name="ROLE_ADMIN" />
            </def>
        </module>
    </security>

</appSecurity>
```

Figura 3-11 Ejemplo de definición de seguridad de un módulo utilizando ficheros XML

- Base de Datos: Para definir las configuraciones de seguridad en la base de datos se crearon las tablas necesarias que permitieran representarlas. Definir la seguridad en la base de datos presenta gran ventaja para las aplicaciones que necesitan desplegarse varias veces sin variar las

[19] XSD *(XML Schema Definition)* Es concebido como una alternativa a las DTD, más compleja, intentando superar sus puntos débiles y buscar nuevas capacidades a la hora de definir estructuras para documentos XML.

configuraciones de seguridad ya definidas las cuales residen en la base de datos y permite la utilización de un módulo de administración de seguridad que permita realizar el proceso de definición de seguridad en cualquier punto del proceso de desarrollo incluso definir o variar las restricciones de seguridad con la aplicación en funcionamiento. Ver figura 3-12.

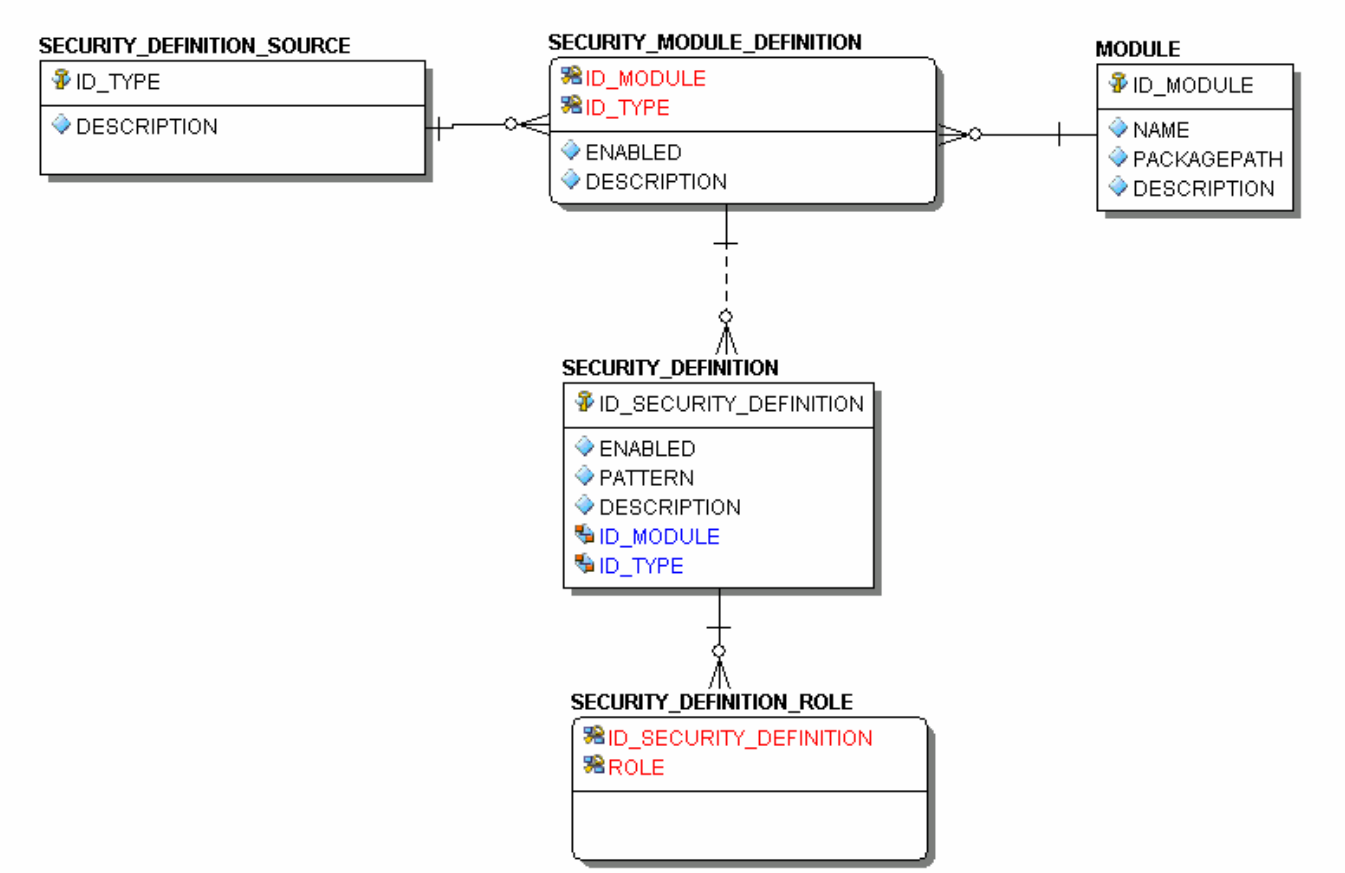

Figura 3-12 Modelo lógico de la estructura de base de datos definida para almacenar las definiciones de seguridad de la aplicación

Para utilizar cualquier de los tipos de seguridad definidos por ArBaWeb Framework se especifica en la propiedad “app.security.definition.type” el tipo de definición de seguridad a usar (XML o base de datos), el valor por defecto es XML.

Proceso de autorización

En el proceso de autorización se determina si el usuario que está tratando de acceder a un recurso puede proceder o no. A continuación se explica el proceso brevemente.

1. El usuario hace una petición HTTP, la misma es interceptada por el filtro de Acegi "FilterSecurityInterceptor", en caso de hacerse la llamada a un método expuesto remotamente es interceptada por la clase que funciona como aspecto "MethodSecurityInterceptor".
2. Tanto la clase "FilterSecurityInterceptor" como "MethodSecurityInterceptor" verifican en primer lugar que los datos del usuario que realiza la acción sean correctos mediante el "AuthenticationManager" y después le delegan la responsabilidad de decidir si puede realizarse la acción solicitada a una clase del tipo "AccessDecisionManager".
3. El "AccessDecisionManager" se encarga de determinar, utilizando las definiciones de seguridad, los datos del usuario y los datos de la acción pedida, si se procede o no con la ejecución de la acción.
4. En caso de denegarse la operación pedida por el cliente se lanza una excepción "AccessDeniedException" y si la petición es web entonces se redirecciona a la página de error definida en la propiedad *security_error_page* en el fichero *security.properties.*

Captcha

En el paquete *captcha* dentro del módulo de seguridad de ArBaWeb Framework está todo lo referente a la integración con el framework JCaptcha. Se destacan dos funcionalidades principales referentes al uso de captcha tanto por autentificaciones fallidas del usuario como intento de acceso recurrente a recursos. También se externalizan un conjunto de propiedades para la personalización de la generación de imágenes del framework.

Autenticaciones fallidas

Una posible vía para obtener la contraseña de un usuario en una aplicación puede ser utilizar generadores de contraseñas y tratar de autenticarse continuamente. Para evitar eso se puede utilizar la autenticación contra robots de Captcha como barrera. Para utilizar esta funcionalidad en ArBaWeb Framework hay que configurar la propiedad security_captcha_and_https_configuration en security.properties donde se especifica qué páginas pueden ser víctimas de estos ataques y la security_max_authentication_failure_attempts en securityCaptcha.properties donde se especifica la cantidad de intentos fallidos permitidos.

Acceso recurrente a recursos

Un posible ataque de denegación de servicios en las aplicaciones sobre la web puede ser mandar a realizar operaciones costosas de manera continua al servidor utilizando software. Para evitar estos ataques se puede utilizar Captcha como puerta al detectar que se están haciendo peticiones muy seguidas a recursos costosos. Para soportar esta funcionalidad en ArBaWeb se configuran las propiedades security_captcha_and_https_configuration en security.properties para especificar qué recursos pueden ser costosos y la propiedad security_max_allowed_request_per_second en secutiryCaptcha.properties para especificar la cantidad máxima de peticiones por segundo que se pueden realizar al recurso.

Personalización de la generación de imágenes

Para personalizar la generación de palabras del framework JCaptcha se crearon un conjunto de clases en ArBaWeb Framework que permiten configurar tanto el tipo de generación de letras o palabras (diccionario, letras aleatorias, etcétera) que utilizará JCaptcha como las deformaciones y otras características que tendrá imagen generada. En el fichero de configuración *securityCaptcha.properties* en la sección “Configuración” se explica cómo realizar estas personalizaciones.

Audit

Este módulo se realizó con el objetivo de cubrir los requerimientos de auditoría de las aplicaciones web sobre JEE de registrar constantemente qué está pasando en el sistema.

¿Por qué es necesario auditar?

Establecer la auditoría en una aplicación es necesario para identificar quién realizó qué acción, información que puede ser usada para controlar la disciplina de las acciones realizadas. Se puede además mostrar cronológicamente qué ha ocurrido antes de un evento dado. La detección de eventos inusuales o no autorizados puede llevar a detectar intrusos en el sistema, por ejemplo, reiterados intentos de entrada al sistema fallidos, entradas al sistema o a partes del sistema fuera del tiempo establecido. Se pueden identificar además conjunto de acciones arrojan fallos en el funcionamiento de la aplicación. Ver figura 3-13.

Puntos en los que se hace necesario realizar la auditoría

- Capa Web: Registrar todas peticiones y respuestas (URLs) realizadas al sistema.
- Capa Servicio: Registrar todas las peticiones que se le hacen a las interfaces de servicio y las respuestas que estas dan.
- Eventos Globales de Aplicación: Registrar eventos importantes que tienen lugar en una aplicación.

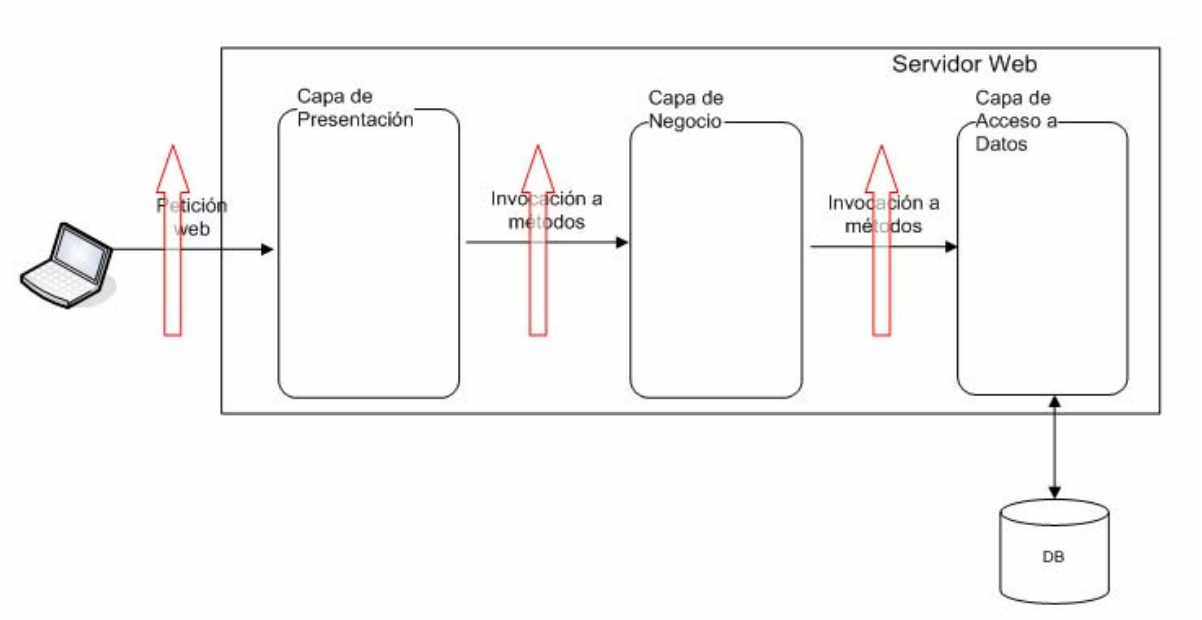

Figura 3-13 Lugares en la aplicación donde se puede aplicar la auditoría, representados por una flecha roja

Para auditar es necesario almacenar los datos de auditaría en ficheros Logs[20]. Para realizar esto se utilizó el framework Log4j y se heredó de algunas clases del framework para agregar un nuevo nivel de auditoría llamado AUDIT adicional a los ya existentes (DEBUG, INFO, WARN, etcétera). Para obtener los datos del usuario que ejecuta cada acción se utilizaron las funcionalidades del framework Acegi.

Para soportar la auditoría en la capa web se creó la clase UrlAuditFilter la cual intercepta todas las peticiones que entran y sales por la web y las escribe en los ficheros de logs de la aplicación utilizando el siguiente formato:

[20]Log: Archivo que registra movimientos y actividades de un determinado programa

> "*2006-05-26 09:09:21,812 AUDIT [pedido o respuesta] – Usuario: nombreUsuario, Role: rolUsuario, IP: ipUsuario, Recurso: página.htm"*

Para auditar los detalles de cada invocación a los métodos de la fachada se creó la clase AudithMethodInterceptor la cual intercepta todas las peticiones que se le hacen a las fachadas de servicio las cuales terminen en Facade. Para modificar esta configuración se cambia la propiedad audit_method_pattern que por defecto tiene el valor **Facade,* por los patrones que se deseen separados por coma. Ejemlpo: **Service, *Facade*. La traza de auditoría tendría el siguiente formato:

> "*2006-05-26 09:10:21,812 AUDIT [nombreClase] – Usuario: nombreUsuario , Role: roleUsuario, IP: ipUsuario, AccedióMétodo: nombreMétodo(), Parámetros: tipo de dato Párametros, Valores: valor Párametros, Retorno: tipo de dato Retorno ValorRetorno: valor Retorno"*

Se auditan los eventos principales del sistema para los que se utiliza la tecnología de los Listeners de eventos los cuales son notificados cada vez que ocurre un evento del sistema y el listener de auditoría se encarga de escribir en los ficheros de Logs los detalles del evento ocurrido en el siguiente formato:

> *"2006-05-26 09:10:21,812 Audit [TipoEvento] – Usuario: nombreUsuario , Role: roleUsuario, IP: ipUsuario, Acción: EventoLanzado y descripción"*

JSON-RPC

Para el uso de este módulo es necesario declarar en un AppContext un BeanPostProcesor llamado *JSONBridgeBeanPostProcessor*. Esta clase puede ser declarada manualmente o se define un parámetro en el archivo de configuración: *app.jsonrpc.enable=true*. Este parámetro le informa a ArBaWeb Framework que debe instanciar el BeanPostProcesor. Esta clase se encarga de registrar los objetos que serán expuestos remotamente.

La forma de exportar un objeto es a través de una anotación de tipo clase definida en ArBaWeb Framework que se le pone a la clase a ser exportada: *JSONExported*; pasándole dos parámetros: *beanName* y *bridgeName*. El parámetro *beanName* define el nombre con que será expuesto bean, si este parámetro es vacio se toma el nombre con que fue definido el bean en el AppContext. El parámetro *bridgeBame* define el nombre del puente (bridge) al que será registrado el bean, si es vacío esta propiedad se registra el bean al puente global. Por defecto ninguno de los métodos del objeto a ser

expuesto es registrado en el bridge, por tanto, a cada método que se quiera registrar para exportar remotamente se debe anotar con la anotación *JSONExportedMethod*. Ver figura 3-14.

```
@JSONExported(beanName="person",bridgeName="bridgeModuleI")
public class PersonFacadeImpl implements PersonFacade {

    @JSONExportedMethod
    public Persona getPerson() {
        return new Afiliado(8505252, "Rosario", "", "Rodríguez", "Torres", "",
                1111, "", "", 222, "");
    }

    public String sayHello(String name) {
        return "Hola!!!! " + name;
    }
}
```

Figura 3-14 Ejemplo de clase exportada en formato JSON utilizando anotaciones.

Los objetos que devuelven los métodos al ser ejecutados tienen atributos, de los cuales pueden existir algunos que no se quieran obtener su valor en el cliente que hizo la llamada remota del método. Para esto a los métodos de accesos de los atributos de esos objetos se anotan con la anotación *JSONExportedNoMethod*. Ver figura 3-15.

Los bridges a los que se registrarán los objetos son instancias de la clase JSONRPCBridge creadas en el AppContext. Esta clase la provee el framework json-rpc. Esta clase presenta modificaciones con respecto a la versión soportada por json-rpc, para poder agregarle las funcionalidades que brindan las anotaciones que se explicaron anteriormente.

Para que el cliente web pueda llamar a un objeto remoto tiene que enviar la petición a una URL en específico. Para lograr esto la URL definida se mapea en un HandlerMapping a un controlador que brinda ArBaWeb Framework, denominado *JSONRPCController*. Este controlador permite según la petición de la llamada remota buscar y ejecutar el método de uno de los objetos registrado en el bridge. Por lo que esta clase presenta una propiedad llamada *json_bridge*, que se le inyecta el beans del bridge declarado en el AppContext, si esta propiedad no es inyectada entonces el controlador automáticamente busca el objeto entre los objetos registrados en el bridge global.

```
public class Persona implements Serializable, Comparable {

    private Integer id_persona;

    private String nombre_primero;

    public Integer getId_persona() {
        return id_persona;
    }

    @JSONExportedNoMethod
    public String getNombre_primero() {
        return nombre_primero;
    }

    public void setId_persona(Integer id_persona) {
        this.id_persona = id_persona;
    }
    @JSONExportedNoMethod
    public void setNombre_primero(String nombre_primero) {
        this.nombre_primero = nombre_primero;
    }
}
```

3-15 Ejemplo de un bean al cual se le aplican anotaciones para excluir propiedades a exponer en el formato JSON

3.3 Flujo de Trabajo

Para el desarrollo de una aplicación se hace necesario seguir un conjunto de pasos que guíe a los desarrolladores en la elaboración de cada componente. Las capas lógicas pueden desarrollarse en paralelo si están bien establecidos los roles con su flujo de trabajo correspondiente. En la DSSA ArBaWeb se propone un flujo de trabajo para el desarrollo utilizando tanto la arquitectura base como el framework propuestos en ArBaWeb. Este flujo de trabajo se enmarca en las funcionalidades mínimas necesarias para que se puedan desarrollar en paralelo las capas y que exista una interacción óptima entre los desarrolladores. Ver Figura 3-16.

Se define un rol correspondiente a cada capa lógica la aplicación: programador de acceso a datos (programador de AD), programador de lógica de negocio (programador de LN) y programador de interfaz de usuario (programador de IU).

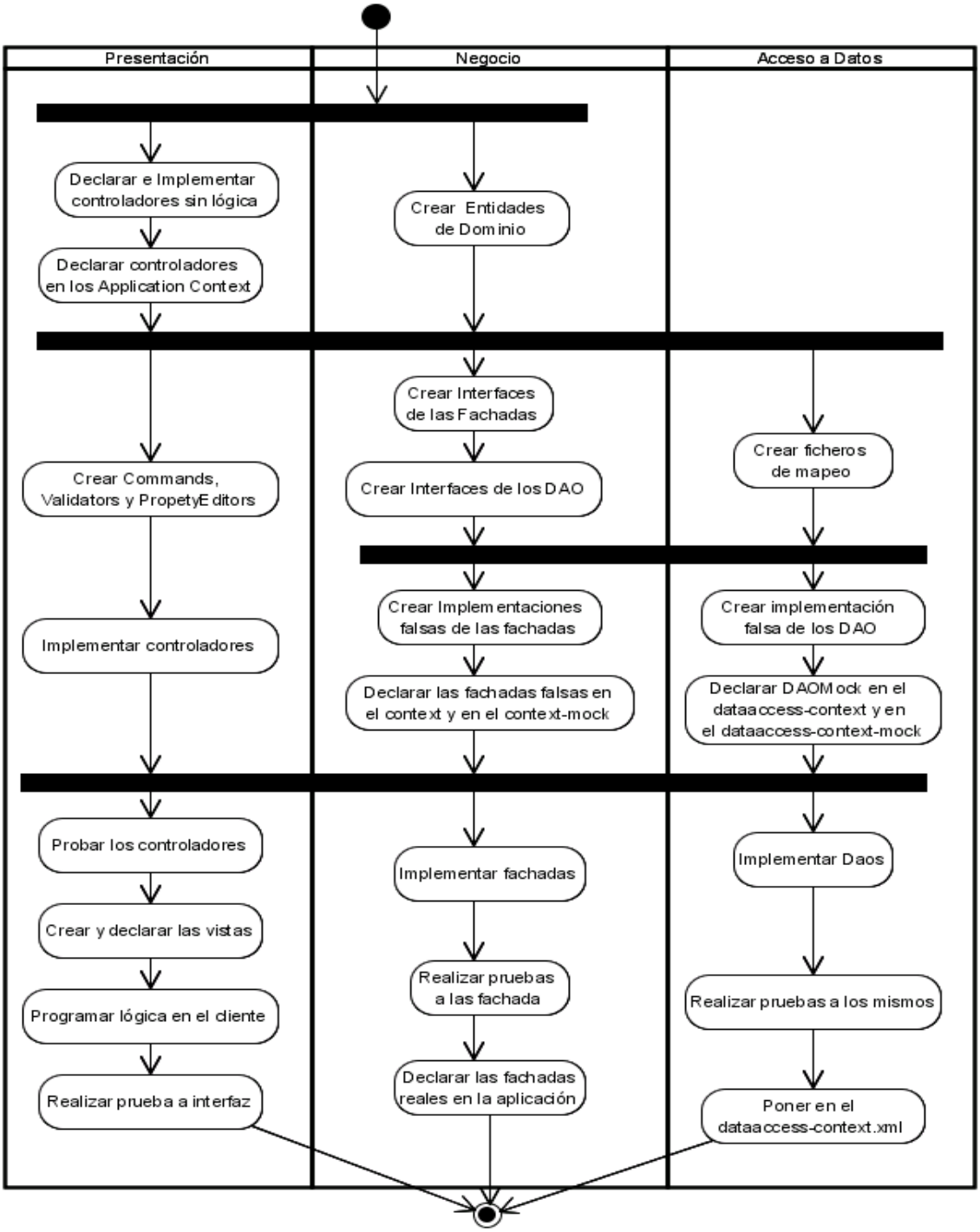

Figura 3-16 Flujo de trabajo de ArBaWeb

Acceso a Datos

3.1 Crear ficheros de mapeo: Cuando se comienza a desarrollar un módulo lo primero que debe salir son las entidades de dominio por parte de programador de LN, el programador de AD utiliza estas entidades para comenzar a realizar los ficheros de mapeo con el diagrama de ER de la base de datos. Esta actividad se debe realizar cada vez que el programador de AD tenga un tiempo libre en su flujo de trabajo y antes de realizar las pruebas a las implementaciones de los DAO.

3.2 Crear implementación falsa de los DAO: Una vez que el programador de LN termine de realizar las interfaces de los DAO el programador de AD debe comenzar inmediatamente la implementación de los DAO falsos para que las demás capas puedan continuar su trabajo y no depender de que el de AD termine todo su trabajo.

3.3 Declarar DAOMock en el dataaccess-context y en el dataaccess-context-mock: Una vez implementados los DAO falsos se declara en el dataaccess-context para que el programador de LN los inyecte a su fachada sin conocer que son las implementaciones falsas. Se deben declarar también en el dataaccess-mock para que puedan ser utilizados en cualquier momento garantizando siempre el funcionamiento de cada capa de la aplicación ante cualquier problema que está presente de funcionamiento.

3.4 Implementar DAO Una vez terminados los ficheros de mapeo se debe proceder a realizar la implementación real de los DAO.

3.5 Realizar pruebas a los mismos: Terminada la implementación se procede a realizar las pruebas de unidad a los mismos.

3.6 Poner en el dataaccess-context.xml: Cuando una prueba de unidad se ha completado completamente a un DAO este ya se encuentra listo para declararse en el dataaccess-context.xml en lugar del DAOMock que existía anteriormente

Lógica de Negocio

1 Crear Entidades de Dominio: El programador de LN realiza las implementaciones de las entidades de dominio que necesita el módulo.

2 Crear Interfaces de las Fachadas. Crea las interfaces de las Fachadas donde representa las funcionalidades que se deben realizar en el negocio, conocidas hasta el momento.

3 Crear Interfaces de los DAO: Crea las interfaces de los DAO donde define las operaciones que se necesitan que se implementen en los DAO.

4 Crear Implementaciones falsas de las fachadas: El programador de IU necesitará que los métodos definidos en la, o las fachadas, se encuentren funcionales para poder implementar los controladores.

5 Declarar las fachadas falsas en el AppContext de negocio y en el context-mock: Una vez que se han elaborado las implementaciones falsas de la, o las fachadas, se procede a declararlas en los ficheros de configuración mencionados. En el AppContext de negocio para que programador de IU pueda continuar su trabajo en los controladores y en el context-mock con el objetivo tener la capa funcional en caso de ocurrir algún problema de implementación en esta capa.

6 Implementar fachadas: Se implementa la lógica que corresponde a cada método dentro de la fachada.

7 Realizar pruebas a las fachadas: Se realizan las pruebas de unidad a las fachadas para probar todas que todas las funcionalidades realicen lo deseado.

8 Declarar las fachadas reales en la aplicación: Una vez realizadas las pruebas de unidad a las fachadas y verificado que todo esté correcto se procede a declarar la implementación real en el AppContext de negocio.

Presentación

1 Implementar controladores sin lógica: Se implementan los controladores sin funcionalidad alguna**.**

2 Declarar controladores en los AC: Se declaran en el AppContext de la capa de presentación context-servlet pues hasta el momento el programador de LN no tiene listas las funcionalidades del negocio listas y se mapean en el HandlerMapping de Spring MVC

3 Crear Commands, Validators y PropetyEditors: Se crean los Command para obtener los datos que vienen de la web, en algunos casos se pueden utilizar las mismas entidades del dominio como

Command. Se crean también los validadores tanto para los Command como para las entidades de dominio en caso de usarse.

4 Implementar controladores: Se implementan los controladores utilizando las interfaces de servicios definidas por el programador de LN.

5 Probar los controladores: Se realizan las pruebas a los controladores para verificar que todas las funcionalidades son cubiertas correctamente.

6 Crear y declarar las vistas: Se crean las vistas que interactuarán con el usuario y se declaran en los ViewResolvers de Spring MVC en caso de ser necesario.

7 Programar lógica en el cliente: Se programa toda la lógica necesaria en el cliente.

8 Realizar prueba a interfaz. Se realizan pruebas para verificar tanto el flujo como que la página realiza las operaciones que deben realizarse.

3.4 Propuesta de Ambiente de desarrollo

Para en definir la propuesta de ambiente de desarrollo para desarrollar aplicaciones que usen ArBaWeb, en el “capítulo 1” se realizó un estudio de las características sobre cada uno de las herramientas y tecnologías a proponer. En el desarrollo de esta sección se precisará en qué partes del desarrollo de software intervienen cada una de estas herramientas y tecnologías. Para esto, se dividió el ambiente de desarrollo en dos grandes grupos: herramientas de desarrollo y frameworks.

Herramientas de Desarrollo

Las herramientas de desarrollo se clasificaron según su función dentro del desarrollo de software (ver “capítulo 1”):

- **Ambiente de desarrollo integrado:** se propone al Eclipse como IDE para desarrollo de software. Incluyendo un conjunto de plugins que amplían sus funcionalidades como plataforma de desarrollo de aplicaciones sobre la web. Los plugins propuestos son los siguientes.
    - Web Tool Platform (WTP)*:* Para soportar todas las funciones necesarias para desarrollar aplicaciones web sobre JEE.
    - Spring IDE: Para facilitar el uso sobre los beans y xml definidos por Spring Framework.

- Hibernate Tools: Para proporcionar facilidades con los archivos de mapeos de Hibernate y poder realizar diferentes tipos de consultas mediante el lenguaje de consultas de Hibernate (HQL) y otras operaciones que soporta este plugin, de una forma más cómoda para los desarrolladores.
- Subclipse: Para permitir de una forma mucho más ágil y cómoda el desarrollo colaborativo de software en un equipo de desarrolladores.
- AspectJ Development Tools (AJDT): Para brindar soporte y facilidades si fuera necesario hacer uso de la programación orientada a aspectos a través de AspectJ.

- **Contenedor Web:** El contenedor web propuesto para desarrollar y desplegar las aplicaciones es el Apache Tomcat.
- **Control de versiones:** Como servidor de control de versiones se recomienda el Subversion.
- **Sistemas gestores de base de datos:** Según los requerimientos y necesidades específicas de cada aplicación se recomiendan dos gestores de base de datos a usar:
  - Oracle: Para tipos de aplicaciones que lo requieran, a pesar de ser propietario.
  - PostgreSQL: Para cualquier tipo de aplicaciones. Se recomienda este por ser libre, junto con todas las características expuestas en el “capítulo 1”.
- **Herramientas de modelado:** Se proponen dos tipos de herramientas para modelar, cada una para responsabilidades específicas:
  - ER/Studio: Para realizar los diagramas de relación de entidades.
  - Visual Paradigm Suite: Para modelar toda la aplicación, por ejemplo, diagramas de clases, de componentes, diagramas de casos de usos, etcétera.

Frameworks

Las tecnologías propuestas a ser usadas en las aplicaciones para dar soporte al complimiento de requerimientos específicos se proponen (ver “capítulo 1”):

- Spring Framework: Framework que brinda un contenedor de inversión de control y da soporte a un conjunto de funcionalidades e integración con otros frameworks especializados en funciones específicas, facilitando el uso de éstos.
- Acegi Security System: Da soporte a la seguridad en una aplicación a distintos niveles como URLs, métodos y objetos. Permite la integración con Spring Framework.
- AspectJ: Da soporte a la programación orientada a aspecto, permitiendo crear y aplicar aspectos.
- Hibernate: ORM usado para la persistencia en las aplicaciones, tratando el modelo de objetos relacional como objetual.
- DBCP: Propuesto para proveer y administrar las conexiones a bases de datos dentro de la aplicación a través de un pool de conexiones.
- JUnit: Framework para realizar las pruebas de unidad dentro de la aplicación.
- EHCache: Para hacer uso de la cache dentro de la aplicación a elementos que lo requieran.
- JCaptcha: Para utilizar el mecanismo de reconocimiento de usuarios humanos.
- Log4j: Para soportar las trazas de la aplicación.
- Json-rpc: Para permitir el soporte de peticiones síncronas y asíncronas a la aplicación usando el formato JSON.

Este ambiente de desarrollo definido es una propuesta de la cual se pueden elegir las herramientas y tecnologías a usar, según requieran las aplicaciones en dependencias de sus necesidades. Esta propuesta es la recomendada para las aplicaciones que utilicen ArBaWeb, pero también puede ser utilizada el desarrollo de aplicaciones que no hagan uso de ArBaWeb.

3.5 Experiencia de ArBaWeb en la UCI.

ArBaWeb se puso en práctica en tres proyectos de la UCI: Correos de Cuba, 171 y Prisiones; cada uno con sus características específicas Esta acción convirtió a ArBaWeb, de una arquitectura específica a un dominio, a una arquitectura instanciada por los proyectos donde se utilizó bajo las condiciones que definieron los arquitectos para las respectivas aplicaciones en desarrollo.

En **Correos de Cuba** existía la necesidad de comenzar un desarrollo de la aplicación en corto tiempo. ArBaWeb sirvió de soporte en la conformación de la arquitectura del proyecto, en la definición del flujo de trabajo para los programadores y en la elección de las herramientas y tecnologías a usar por lo que optimizó el tiempo y las condiciones de desarrollo del proyecto cubano. El framework ayudó a cumplir requerimientos del proyecto como seguridad, auditoría, carga automática de recursos, simplicidad en la capa de acceso a datos y aplica la reutilización de componentes.

171 presenta características muy similares a Correos de Cuba, es un proyecto que está comenzando el desarrollo, en el cual, el uso de ArBaWeb ayudó a que enfocaran su esfuerzo desde un inicio a la preparación de los desarrolladores por especialidades.

Prisiones es el proyecto que más ha usado el framework ArBaWeb debido a que la idea de su creación surgió como necesidad del propio entorno y, por tanto, se aplica desde los inicios gestionando toda la arquitectura y sirviendo de base en el aprendizaje y formación del rol de arquitecto en el proyecto. La base de los requerimientos de dominio identificados surgieron en la matriz de Prisiones y se contrastaron con los demás proyectos productivos que utilizan JEE.

El uso de ArBaWeb en estos proyectos ha contribuido a encontrar nuevos requerimientos comunes de estas aplicaciones a los cuales se les ha ido dado solución en el framework. El que estos proyectos utilicen tanto arquitecturas como tecnologías muy similares permite que puedan reutilizar componentes de unos en otros y que la experiencia que se va obteniendo durante el desarrollo de cada proyecto se comporte como una única experiencia para los proyectos con estas características.

Se comprobó la posibilidad de asimilación de ArBaWeb en diferentes entornos, por ejemplo, en el soporte del acceso a datos en Oracle y PostgreSQL.

3.6 Conclusiones

Con el desarrollo de este capítulo se expusieron las características fundamentales de Arbaweb Framework. Se presentó un flujo de trabajo que define y organiza las principales tareas del proceso de desarrollo. Se elaboró una propuesta de ambiente de desarrollo donde se expusieron las características de las herramientas y tecnologías propuestas. Se abordó a grandes rasgos la experiencia que ha tenido ArBaWeb dentro de tres proyectos de la Universidad.

CONCLUSIONES

Se desarrolló ArBaWeb, arquitectura y framework, como respuesta a la reutilización de componentes en los proyectos productivos de la UCI que utilicen como dominio las aplicaciones web sobre JEE.

Con la utilización de ArBaWeb se enfatiza en una producción de software con calidad al aplicar un desarrollo regido por estándares, las buenas prácticas de programación y patrones de diseño en el desarrollo de aplicaciones web sobre *JEE.*

Se analizaron proyectos reales de producción de los cuales se derivaron los requerimientos de referencia para ArBaWeb haciendo práctico este trabajo desde sus inicios y constatando los resultados con su aplicación en varios proyectos para su validación.

Se definió el flujo de trabajo general de ArBaWeb con el propósito de guiar y definir las actividades a realizar en el desarrollo de la aplicación y el beneficio de implementar sobre las capas lógicas de forma paralela.

Se propone un ambiente de desarrollo basado en las herramientas y tecnologías con el propósito de estandarizar el uso de ellas en los proyectos.

Con la utilización de ArBaWeb, se provee a la universidad del valor de uso de las experiencias que se adquieren en los proyectos al poderlos extender a otros con características o necesidades similares y se estandariza la “manera de hacer” en la producción.

RECOMENDACIONES

Se recomienda:

- Continuar con el desarrollo de los módulos definidos de ArBaWeb Framework.
- Adicionar nuevas funcionalidades al framework basados en los requerimientos de los proyectos que se continúen identificando.
- Crear herramientas que faciliten el uso de ArBaWeb como un plugin para Eclipse para generar un proyecto web con la integración de ArBaWeb.
- Extender el dominio de ArBaWeb para desarrollar aplicaciones que usan frameworks web, como Struts y Java Server Faces (JSF), y a aplicaciones de escritorio.

BIBLIOGRAFÍA

Modelado de Sistemas con UML. 2002. [Disponible en: http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/multiple-html/index.html

APPFUSE. AppFuse, Wikipedia, The Free Encyclopedia. Disponible en: http://en.wikipedia.org/wiki/AppFuse

DAVID GALLARDO, E. B., ROBERT MCGOVERN. Eclipse In Action: A Guide for Web Developers. 2003.

DAVID GARLAN, M. S. An Introduction to Software Architecture. 1994. p.

---. Software Architecture: Perspectives on an Emerging Discipline. 1996. p.

DEEPAK ALUR, J. C., DAN MALKS. Core J2EE™ Patterns: Best Practices and Design Strategies. Second Edition. 2003. p. 0-13-142246-4

FLOWER, M. Patterns of Enterprise Application Architecture. 2002. p.

GRAND, M. Patterns in Java, A Catalog of Reusable Design Patterns Illustrated with UML. Second Edition. Robert Ipsen, p. 0-471-22729-3

JOHNSON, R. Professional Java Development with the Spring Framework. 2005. p. 0764574833

KAISER, S. H. Software Paradigms. 2005. p. 0471483478

R. E. JOHNSON, B. F. Designing Reusable Classes, 1988.

R. E. JOHNSON, V. F. R. Reusing Object-Oriented Designs.

RAIBLE, M. AppFuse: Start Your J2EE Web Apps, 2004. [Disponible en: http://today.java.net/pub/a/today/2004/07/15/thefuse.html

What's New in AppFuse 2.0. Disponible en: http://static.raibledesigns.com/repository/presentations/WhatsNewinAppFuse2.pdf

ROD JOHNSON, J. H., ALEF ARENDSEN, COLIN SAMPALEANU, ROB HARROP, THOMAS RISBERG, DARREN DAVISON, DMITRIY KOPYLENKO, MARK POLLACK, THIERRY TEMPLIER, ERWIN VERVAET, PORTIA TUNG, BEN HALE, ADRIAN COLYER, JOHN LEWIS, COSTIN LEAU, RICK EVANS. The Spring Framework - Reference Documentation. Disponible en: http://www.springframework.org/docs/reference/new-in-2.html

GLOSARIO DE TÉRMINOS

Capa de presentación: Generalmente se identifica como la capa web. Esta capa debe ser tan fina como sea posible. Además, debe permitir diferentes capas de presentación, tales como una capa web y/o fachadas de servicios web remotos, sobre una simple y bien diseñada capa de negocio.

Capa de servicios de negocio: Es la responsable de delimitar las transacciones y proveer un punto de entrada para las operaciones sobre el sistema. Esta capa no debería tener conocimiento sobre lo concerniente a la presentación y debería ser reutilizable.

Capa de acceso a datos: Esta capa presenta un conjunto de interfaces independientes de la tecnología de acceso a datos, que son usadas para buscar y persistir los objetos persistentes. Estas son interfaces del patrón de comportamiento definido en los patrones GOF, llamado Estrategia (Strategy) (GRAND). Las implementaciones de estas interfaces, según como se explicará más adelante en ArBaWeb Framework, usan Hibernate para persistir las entidades persistentes del dominio, aunque estas pueden ser implementadas usando cualquier tecnología de mapeo objeto-relacional (ORM) o la capa de abstracción para JDBC que trae Spring Framework. En algunas aplicaciones y en algunos casos de aplicaciones complejas, quizás sea necesario usar JDBC para lograr un mayor rendimiento con la persistencia. La capa de acceso a datos no debería de contener ningún tipo de lógica de negocio.