Incorporación del Patrón Trazador en una Aplicación WEB

Beatriz Arbeláez
Universidad Sergio Arboleda, Bogotá, Bogotá D.C., Colombia, beatriz.arbelaez@correo.usa.edu.co

RESUMEN

La Trazabilidad es un Requerimiento No Funcional (RNF), solicitado en aplicaciones financieras como complemento al RNF de Seguridad. Incorporarlo exige modificaciones en el diseño y adicionalmente en el código cuando se realiza sobre aplicaciones ya en uso por el usuario final. El costo del mantenimiento implica un alto número en tiempo y recursos para cumplir con la solicitud. Para disminuir estos costos se incorpora el Patrón Trazador C sobre la aplicación WEB, dicho mecanismo se explica con un ejemplo. Dada la aplicación WEB con arquitectura de capas sobre tecnología PHP, se describen los pasos y los resultados del uso del patrón, iniciando con una explicación de la situación actual.

Palabras Claves: Patrón, Trazador, Ejemplo, Arquitectura, WEB

1. INTRODUCTION

La Aplicación WEB Financiera utilizada en este documento, refleja un estado típico presente en entornos con manejo de clientes, clasificación de datos y diversidad de fuentes, equivalentes a funcionalidades tan diversas como actualizaciones parciales de toda la información, gestión de transacciones y ejecuciones asíncronas. El listado completo de los requerimientos se presenta a continuación, en la Tabla 1:

Tabla 1: Listado de Requerimientos Funcionales en la Aplicación WEB Original

1. Diferente Opciones de Visualización, formularios, menú de botones, ventanas emergentes, ventanas con pestañas, tablas
2. Vistas Compuestas, por las diferentes opciones de visualización mencionadas anteriormente
3. Implementación de Autenticación contra una Base de Datos de Usuarios
4. Uso de la Paginación cuando se presenta alto número de información
5. Diversidad en las fuentes de datos, tres Bases de Datos diferentes, dos Motores Gestores diferentes
6. Implementación de Transacciones Distribuídas
7. Manejo de actualizaciones parciales de los datos
8. Nivel de Acceso y Autorización por granularidad de páginas u objetos
9. Control de la Sesión de Usuario por tiempo y localización
10. Menú Dinámico de Contenidos
11. Cifrado de Clave en el Almacenamiento
12. Contar el número de RFs seleccionados, ejecutados por el usuario

La arquitectura de la aplicación posee en la estructura física dos capas, la capa cliente y la capa servidor, en la estructura lógica tres capas que reflejan el esquema Modelo, Vista, Controlador (MVC). Diseñado de esta forma para disminuir la complejidad y aprovechar las características de las tecnologías con las cuales se contaba. PHP es la plataforma que solicita los datos, los procesa y visualiza los resultados. Todo lo hace sobre un Servidor WEB Apache, independiente del cliente WEB que los invoca, dicho servidor tambien aloja las Bases de Datos utilizadas, dos Bases de Datos, Oracle 11g Enterprise Edition y MySQL, para fines académicos se presenta la aplicación con Oracle 11g Express Edition. Una vista de la arquitectura se presenta en la Figura 1:

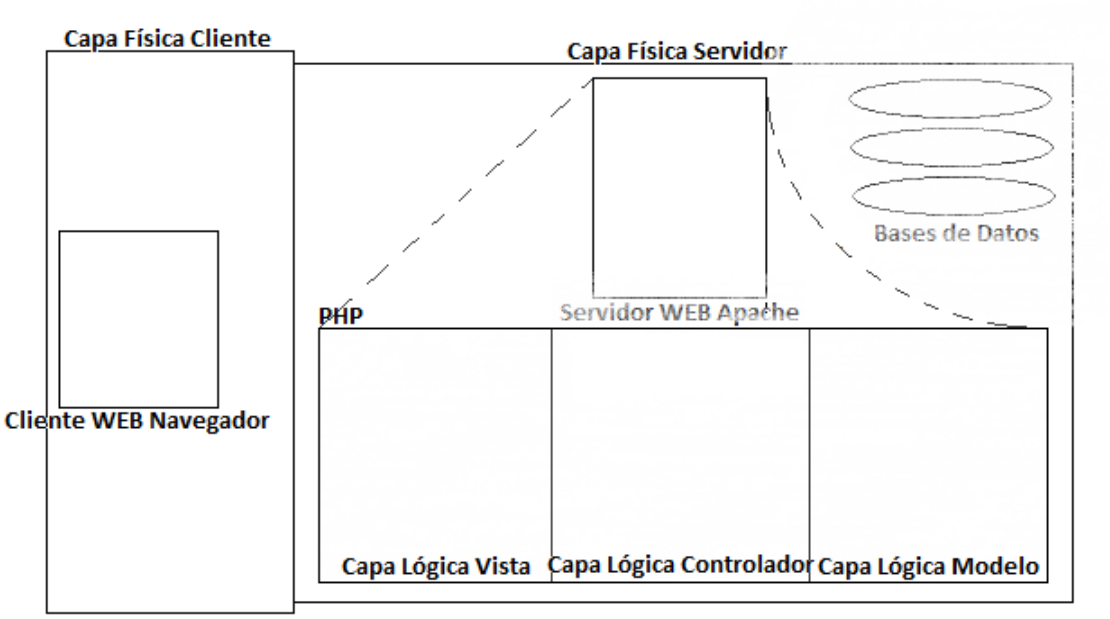

Figura 1: Arquitectura de la Aplicación

Una adición a los RF de la aplicación, expuestos en la Tabla 1, es la necesidad de replicarla para tener varios clones de la aplicación con fácil configuración de características mínimas. Los cambios que se requieren diferentes sobre cada copia, se tienen en la Tabla 2.

Tabla 2: Listado de Cambios a Configurar

Logo de la Aplicación
Mensajes de Error
Conjunto de Datos a Visualizar
Número de Vistas
Conexiones a Bases de Datos
Permisos a Controlar
Ubicación de la Aplicación

A continuación, se describe el Diseño e Implementación de la Aplicación anterior a la incorporación del Patrón Trazador C como mecanismo para cumplir la solicitud del RNF Trazabilidad.

2. DISEÑO E IMPLEMENTACIÓN DE LA APLICACIÓN

La aplicación fue realizada con una mezcla de los paradigmas de Programación Orientada a Objetos y Programación Procedimental. Esto por el tiempo de vida de la aplicación, aproximadamente 5 años iniciando desde el año 2006, y el paso de varios Ingenieros que modificaban constantemente su diseño para mejorar tiempos de mantenimiento y atención a incidentes(Ernst et al. 2010). Como resultado final y punto de partida de este documento, la Figura 2 muestra los objetos actuales que la conforman y sus relaciones, se puede apreciar la orientación por capas y la implementación del estilo MVC.

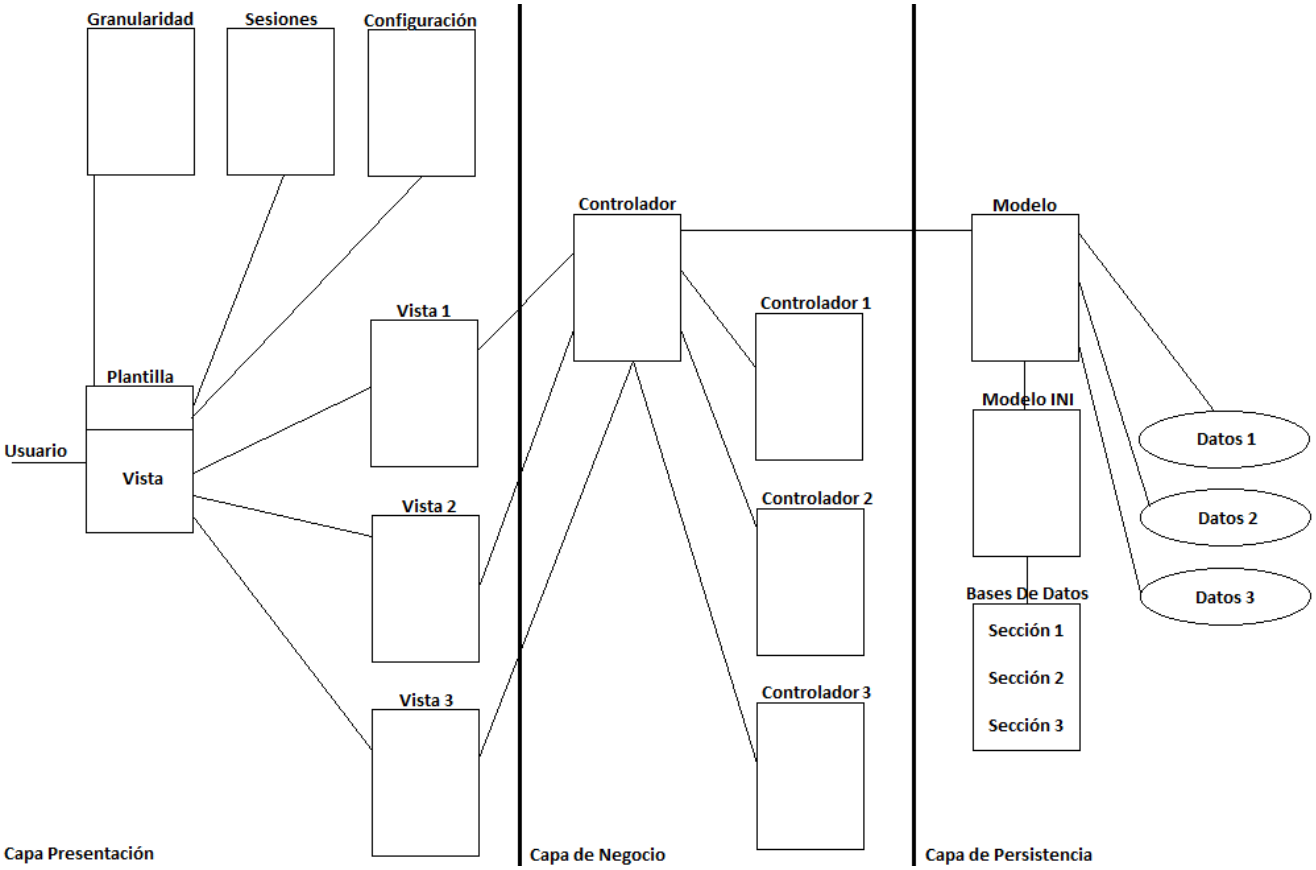

Figura 2: Diseño de la Aplicación

Todas las peticiones del usuario final se centralizan en el archivo plantilla.php que incluye los archivos granularidad.php, sesiones.php y configuracion.php. Con esto se maneja la granularidad de la seguridad por objeto (archivo.php) donde se pregunta al principio de cada página si el usuario es válido y tiene permiso para acceder a dicha página. Todo por paso y envío de información configurada en variables de sesión, apoyadas en condiciones de configuración como ubicación de la aplicación, logo que la identifica, datos a visualizar y todos los listados en la Tabla 2. Acompañando la capa de presentación, se crean archivos nombrados con el prefijo Vista y un Consecutivo, para indicar cada RF a cumplir por la aplicación. Son archivos .php independientes que se incluyen en el archivo plantilla.php en la sección de vista.

El segundo conjunto de objetos que forman la capa de negocio, representan a su vez el controlador de la aplicación. Se cuenta con un controlador.php general que centraliza todas las peticiones de las vistas y ejecuta acciones diferentes, dependiendo de la vista que lo invoca. Los archivos que se encuentran después, controlador1.php, controlador2.php, controlador3.php se anexan al controlador.php para construir un objeto equivalente a un despachador que direcciona las acciones solicitadas.

Finalmente el archivo modelo.php coordina diferentes Bases de Datos a petición del controlador.php, centralizando igualmente las entradas a la capa de persistencia. Esta aplicación en particular maneja tres repositorios, referenciados en la Figura 2 como datos 1, datos 2, datos 3, en la realidad dos Bases de Datos MySQL y una Base de Datos Oracle 11g Enterprise Edition. Para evitar invasión en el código con tres diferentes conjuntos de datos de conexión, se tiene un archivo basesdatos.ini, donde en formato propiedad (nombre y valor) se hacen explícitos los datos solicitados por cada fuente de datos, sin embargo, PHP proporciona una característica útil, dentro de ese mismo archivo manejar secciones, en la aplicación cada sección representa una Base de Datos como se puede observar en la Tabla 3, por condiciones de seguridad se tienen valores de prueba.

Tabla 3: Líneas del Archivo basesdatos.ini

BasesDatos.php
;Datos Conexion BD1
[BD1]
Login = bd1 y bd0
Password = bd1
Profile = bd1
;Datos Conexion BD2
[BD2]
Login = bd2
Password = bd2
Profile = bd2
;Datos Conexion BD3
[BD3]
Login = bd3
Password = bd3
Profile = bd3

Este diseño permite independencia en el manejo de repositorios de datos y en la configuración de la tecnología e información que se necesita para su localización, consulta y manipulación. Durante seis meses la aplicación funcionó correctamente en el ambiente del usuario final, hasta la solicitud de los nuevos requerimientos de Trazabilidad que se presentan a continuación.

3. SOLICITUD DE TRAZABILIDAD EN LA APLICACIÓN

El RNF de Trazabilidad, consiste en la capacidad de dejar huella de las actividades realizadas que se consideren suceptibles de seguimiento(Haigh 2010). También debe permitir la reconstrucción de eventos o situaciones que llevaron a determinado estado dentro de la aplicación. La Trazabilidad tiene diferentes enfoques, niveles y administraciones, dichas características dependen del Paradigma de Programación que se utilizó en el diseño e implementación de la aplicación. Si el paradigma Orientado a Objetos(Ancona et al. 2006), fue el seleccionado, el objetivo debe realizar un seguimiento sobre los objetos, sus atributos y métodos. Si el seleccionado fue el paradigma Procedimental, el objetivo debe enfocarse en los procesos a seguir. Para incorporar la trazabilidad a la aplicación se solicitaron los puntos expuestos en la Tabla 4:

Tabla 4: Solicitudes de Trazabilidad

Ingreso y Salida de la Aplicación
Modificación de Datos (mantener datos viejos y datos nuevos)
Funcionalidades ejecutadas y resultados
Tiempos de pausa en el uso de la aplicación

Se observa en la Tabla 4 que la orientación de las solicitudes, es a un manejo procedimental sobre la ejecución, teniendo en cuenta este aspecto la implementación del Patrón Trazador C sobre la Aplicación WEB se realizó de la siguiente manera.

4. INCORPORACIÓN DEL PATRÓN TRAZADOR C

Una de las características del Patrón Trazador C es no ser invasivo, se comporta como una capa adicional, no en forma de jerarquía, al diseño de la aplicación inicial y trabaja con la misma arquitectura que esta. Igualmente, tiene un formato de bajo acoplamiento que permite su inclusión en tres pasos, el contenedor de trazabilidad conocido como Trazador C, un objeto que recibe y centraliza todas las peticiones del usuario conocido como Trazador y por último los archivos .xml de configuración sobre los objetos de la aplicación que deben ser rastreados.

Los tres componentes mencionados anteriormente, tienen un esquema de implementación determinado que les permite convertirse en plantillas, donde sólo se hace necesario completar los espacios con la información propia de la aplicación en que se van a incorporar. La Tabla 5, muestra la formación de cada objeto en forma genérica.

Tabla 5: Formatos Genéricos del Patrón

Trazador	Trazador C	XML
Inclusión de todos los requerimientos de la aplicación Equivalencia de cada requerimiento por objeto que lo inicia Equivalencia de cada requerimiento por cada XML que soporta cada objeto que lo inicia	Inclusión de todos los requerimientos Inclusión de todos los archivos XML Conexión con el Modelo de la Aplicación Transformación con XFD de los XML	Inclusión de todos los objetos que componen la aplicación Orden de las invocaciones realizadas por ese requerimiento -Descripción -Identificador (nombre del archivo físico) -Orden Por cada invocación realizada, los datos necesarios -Entradas -Salidas -Eventos -Actores Equivalencia de datos contra campos en Bases de Datos

El diseño final de la aplicación WEB que se implementó, incluyendo los formatos de la Tabla 5 y la teoría del patrón, se presenta en la Figura 3. Se aprecia, la inclusión de los elementos del Patrón Trazador C que requiere en la parte de archivos XML una descripción detallada del flujo dentro de la aplicación.

Es necesario como pasos previos realizar el listado de datos que reflejan la traza a guardar, definir la equivalencia en los repositorios de Base de Datos y describir el orden y transformaciones que pueden sufrir. Adicional a las formas genéricas de los objetos, se maneja un formato previo que permite ordenar la aplicación y clarificar las entradas a los objetos de la Tabla 5. La implementación aún se encuentra en proceso, se cuenta con los archivos .xml generados y con el archivo trazadorC.php.

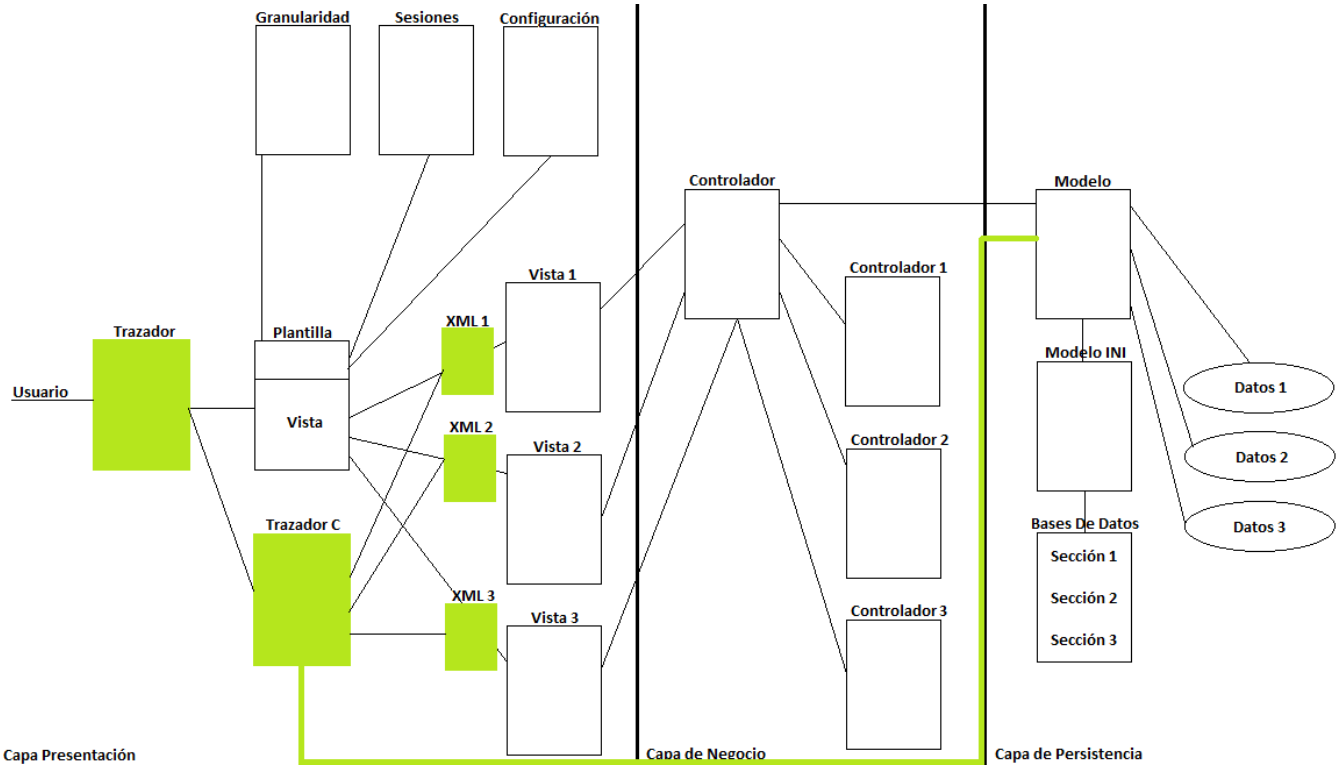

Figura 3: Diseño de la Aplicación con el Patrón Trazador C

5. RESULTADOS, CONCLUSIONES Y TRABAJO FUTURO

El presente trabajo es la segunda experiencia de implementación del patrón sobre aplicaciones reales. Y la primera experiencia en ambiente PHP, pues la primera experiencia se realizó en ambiente Java(Song et al. 2006). Los trabajos previos, buscaban tecnologías adecuadas donde implementar las características de funcionamiento exigidas, el lanzamiento en paralelo de la solicitud del usuario, división equivalente a la ejecución de la traza y del RF correspondiente, la sincronización de los XML y la doble personalidad del objeto Trazador C, ser un controlador y un modelo. Este último punto aún requiere atención y nuevos formatos genéricos.

Construir los objetos del Patrón Trazador C sobre la Aplicación WEB fue fácil, pues la aplicación presentaba un orden, gracias a la calidad de profesionales que participaron en su construcción, sin embargo, en oportunidades anteriores, pruebas parciales de funcionamiento en otros ambientes(Broens et al. 2007), el diseño y la implementación del patrón exigía largas jornadas de orden dentro de la aplicación, para reconocer, listar, depurar y determinar el punto exacto donde debe ubicarse el objeto XML.

Otro inconveniente presentado fue el cambio de tecnología de Java a PHP donde el funcionamiento varía en la ejecución de las necesidades tecnológicas. Realizarlo en PHP exigió un trabajo compartido con el Servidor WEB, el cuan definía las limitaciones o decisiones presentadas, al hacerlo con Java sólo era este lenguaje quien determinaba el camino a seguir. No se tienen en cuenta las especificaciones de la arquitectura WEB en ambas tecnologías.

La incorporación del Patrón Trazador C, permitió disminución en el tiempo y número de recursos usados, frente a condiciones de no uso del Patrón. El Patrón Trazador C, tiene la flexibilidad de configurarse a decisión de un usuario administrador de la aplicación para determinar, datos a seguir y características de su operativa.

Desacoplar el RNF de Trazabilidad de una aplicación, permite el incremento de RFs, independientes de los RNFs en general. Objetivo que siempre se ha buscado. Esta misma separación, también hace posible que los roles

participantes en el proceso de ingeniería, realicen sus tareas sin dependencias, con una fácil realización e integración para alcanzar al final, el producto completo.

Dentro de una arquitectura WEB, el patrón se incorpora correctamente y sin dificultad, el siguiente trabajo es verificar dicho comportamiento en aplicaciones de Escritorio y Móviles. Aunque la aplicación utilizada en este documento suponga un orden desde su creación, esto no implica que sólo funcione correctamente en dichas situaciones. Otra actividad posterior es aplicar el Patrón Trazador C en aplicaciones con caos en diseño e implementación.

La ejecución y el manejo de la aplicación no cambia(De Labey et al. 2008), para el usuario final. La usabilidad que como RNF se diseño e incorporó en el diseño e implementación desde sus orígenes no sufre alteraciones. Lo mismo sucede con los demás RNF, para el usuario la actualización de la aplicación es transparente.

El trabajo futuro se extiende a la solución de los errores y refinamiento de este patrón, así como a la posibilidad de responder la inquietud que ha surgido en este trabajo, puede ser posible implementar otros RNF con el esquema del Patrón Trazador C?

REFERENCIAS

Ancona, D., Lagorio, G. and Zucca, E., (2006). "Flexible Type-Safe Linking of Components for Java-Like Languages". Lecture Notes in Computer Science, Vol. 4228, pp 136-154.

Broens, T., Quartel, D. and van Sinderen, M., (2007). "Capturing Context Requirements". *Lecture Notes in Computer Science*, Vol. 4793, pp 223-238.

De Labey, S. and Steegmans, E., (2008). "Making Service-Oriented Java Applications Interoperable without Compromising Transparency", ENTERPRISE INTEROPERABILITY III, Part III, pp 233-245.

Ernst, N.A. and Mylopoulos, J., (2010). "On the Perception of Software Quality Requirements during the Project Lifecycle". *Lecture Notes in Computer Science*, Vol. 6182, pp 143-157.

Haigh, M., (2010). "Software quality, non-functional software requirements and IT-business alignment", Software Quality Journal, Vol. 18, pp 361-385.

Song, Y.T. and Conover, A.J., (2006). "Slicing JavaTM Programs Using the JPDA and Dynamic Object Relationship Diagrams with XML", *Lecture Notes in Computer Science*, Vol. 3647, pp 201-213.

Authorization and Disclaimer