Industrial Data
ISSN: 1560-9146
iifi@unmsm.edu.pe
Universidad Nacional Mayor de San Marcos
Perú

Santos López, Félix Melchor
Diseño de un módulo de carga de pagos en entidades públicas mediante mensajería con spring framework
Industrial Data, vol. 15, núm. 2, junio-diciembre, 2012, pp. 73-79
Universidad Nacional Mayor de San Marcos
Lima, Perú

Revista de la Facultad de Ingeniería Industrial
15(2): 73-79 (2012) UNMSM
ISSN: 1560-9146 (Impreso) / ISSN: 1810-9993 (Electrónico)

Diseño de un módulo de carga de pagos en entidades públicas mediante mensajería con spring framework

Recibido: 14/09/12 Aceptado: 12/12/12

Félix Melchor Santos López[1]

RESUMEN

Las entidades e instituciones públicas en el Perú proveen una gran gama de servicios a los ciudadanos, dentro de los cuales están incluidos aquellos que involucran pagos o transacciones financieras. Por ello, uno de los procesos importantes para el pago de obligaciones y servicios públicos es el de la "acreditación de pago" que se produce cuando una entidad bancaria envía un archivo plano con los pagos realizados por la web y ventanillas para que la entidad pública los registre como válidos. La solución en el presente artículo conlleva a la implementación de un planificador de tareas y un aplicativo de registro de pagos, siendo estos desarrollados bajo el lenguaje de programación Java mediante el marco de trabajo Spring Framework, JMS (Java Message Service) y un Message Broker (corredor de mensajes). La solución tuvo como resultado la generación del diseño del sistema de información con UML (Unified Modeling Language), una correcta arquitectura de software y su adecuada documentación para mantenimientos futuros.

Palabras clave: entidad pública, pago, acreditación, JMS, Spring Framework

DESIGN OF A LOAD MODULE PAYMENT IN PUBLIC ENTITIES THROUGH MESSAGING WITH SPRING FRAMEWORK

ABSTRACT

Public entities and institutions in Peru offer several services to citizens, where there are payments and financial transactions. Therefore, an important process called "payment accreditation" is necessary when a bank send public entities all the transactions, in a file, made by Internet and bank counters during a day, then these are registered as valid. The main solution presented in this article is the implementation of a task-scheduler and a register payment application, which were developed under Java language programming, Spring Framework, JMS (Java Message Service) and a Message Broker. As a result, the solution produced a great design for information system because UML was applied, also a right software architecture and its accurate documentation for further maintenances.

Keywords: public entity, payment, accreditation, JMS, Spring Framework

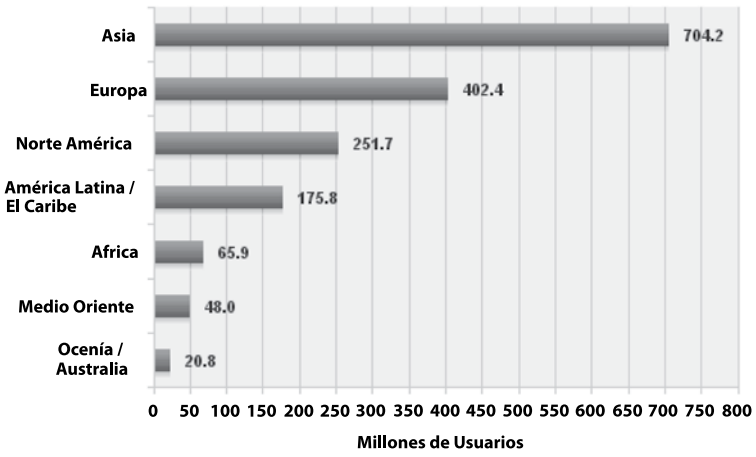
I. INTRODUCCIÓN

En la actualidad las entidades e instituciones públicas que brindan diversos servicios a los ciudadanos y empresas, siendo estos afectos de aportes o pagos monetarios, requieren de sistemas automatizados de carga de pagos on-line y/o off-line cuya información proviene de las entidades bancarias con las cuales se cuenta con un convenio, proveyendo una certificación o constancia de que efectivamente el pago se realizó correctamente, es decir certificando su autenticidad.

En paralelo, en las últimas décadas el mundo viene experimentando un acelerado boom tecnológico y con ello, se viene llevando el negocio de la banca al mundo informático. "En estos momentos, toda la banca mundial experimenta o está en el proceso de adopción de la nueva tendencia de trabajo electrónico a través de internet, identificada con varios términos como: Home Banking y Banca Online; que ofrece servicios a los clientes que dispongan de un acceso a la red" [4].

La penetración de conexiones a internet en el mundo viene creciendo a un ritmo acelerado, ello atrae a un mayor número de usuarios dispuestos a realizar transacciones bancarias en la red. Tal como se muestra en la figura 1, a junio del 2009 se alcanzó la cifra de 1668.8 millones de usuarios a nivel mundial.

Figura 1. Usuarios de internet en el mundo por regiones – junio 2009



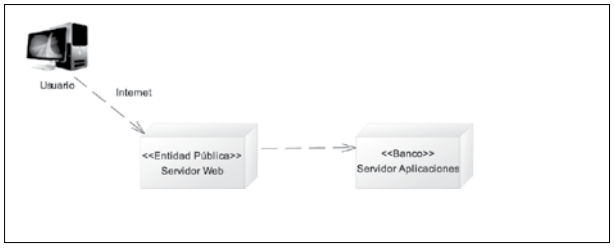
Fuente: Tomado de la referencia [6]

1 Ingeniero Informático, Pontificia Universidad Católica del Perú. E-mail: fsantos@pucp.edu.pe

Por lo tanto, las administraciones gubernamentales vienen desarrollando diversas aplicaciones web para el pago de obligaciones tales como impuestos, arbitrios, trámites de pasaportes, documentos de identidad, constancias, certificados, antecedentes penales, etc.

Para ello se implementan aplicaciones en los portales institucionales de las entidades estatales, permitiendo el pago de servicios u obligaciones que a su vez se interconectan con los bancos para llevarlos a cabo. En la figura 2 se ilustra el flujo de este procedimiento, donde el usuario accesa y realiza al pago a través de la página web de la entidad, luego ésta solicita el pago y débito de la cuenta del usuario al banco respectivo realizándose finalmente la transacción.

Figura 2. Pago en línea



Fuente: Elaboración propia

Por otro lado, existe un segundo proceso llamado "acreditación de pago" que implica un envío de un archivo plano con todas las transacciones efectuadas durante el día tanto a través del pago en línea por internet como por ventanilla bancaria a favor de la entidad pública. Esto implica una carga masiva de la data hacia los sistemas gubernamentales. En la figura 3 se aprecia un gráfico de como el archivo plano pasa al servidor de aplicaciones, para que éste último procede a registrar los pagos en la base de datos.

Figura 3: Proceso de acreditación de pagos

Fuente: Elaboración propia

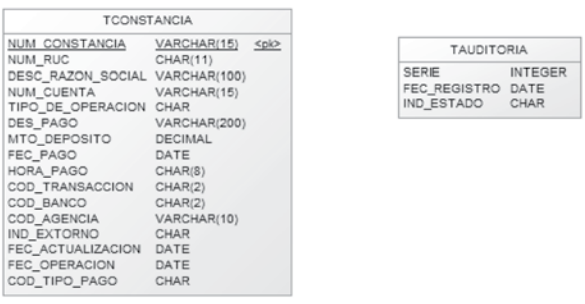
Finalmente, se plantea como solución a este segundo proceso de acreditación de pagos la implementación de un aplicativo automatizado de carga, que implica una lectura automática del archivo enviado por el banco a una determinada hora del día y su registro en la base de datos. Adicionalmente, es importante guardar el registro o evidencia de que realmente el proceso se realizó y para este caso se implementa una tabla de auditoría en la base de datos.

II. ANÁLISIS DE LA SOLUCIÓN

Como solución de implementación a la carga automatizada del proceso de acreditación de pagos se decidió realizar las siguientes tareas:

1) Recibir el archivo plano con las transacciones realizadas durante el día en el banco, ya sea por ventanilla o internet.
2) Luego almacenar este archivo en una ruta específica del servidor de aplicaciones, por ejemplo: /dat0/tempo/bancos/
3) Seguido desarrollar una aplicación del tipo planificador que permita ejecutarse a una determinada hora del día. En este caso se estableció las 23 horas con 10 minutos de lunes a domingo.
4) La aplicación se encargará se enviar todos los registros del archivo plano a un corredor de mensajes, es decir que se ejecute el registro de estos en forma asíncrona.
5) El corredor de mensajes recibe la petición y la envía al servicio solicitado que es la aplicación encargada de realizar el registro a base de datos mediante llamadas a clases de persistencia de datos.
6) Una vez que la aplicación de servicio de registro termina de registrar los pagos en la tabla "TCONSTANCIA", procede a registrar en la tabla "TAUDITORIA". Todos estos registros se dan mediante las clases de persistencia de datos. En la figura 4 se aprecia el modelo de datos de las tablas que soportaran los registros de la aplicación.

Figura 4: Modelo de datos

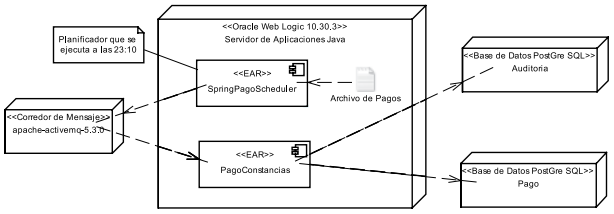


Fuente: Elaboración propia

Así mismo, en la figura 5 se aprecia un diagrama de despliegue que ejemplifica lo explicado en los

pasos anteriores en base a los componentes a desarrollar. De grafico se aprecia el EAR (Enterprise Application Archive) "SpringPagoScheduler" que se encargará de leer los archivos planos a las 23:10 horas todos los días enviando la información al corredor de mensajes "apache-activemq". Luego este se conecta en forma asíncrona al EAR "PagoConstancias" que se encarga de registrar los pagos certificados por el banco tanto en la base de datos pago y auditoría.

Figura 5. Diagrama de despliegue



Fuente: Elaboración propia

Para el desarrollo del aplicativo se emplean las siguientes tecnologías:

Tabla 1. Descripción de las tecnologías a utilizar en el desarrollo

Tecnología	Descripción
Java®	Lenguaje de programación orientado a objetos.
Spring Framework®	Framework de desarrollo de software en el lenguaje de programación Java. Posee una variedad de módulos utilizables que disminuyen el tiempo de desarrollo.
JMS®	Java Message Service, modulo proporcionado por el lenguaje de programación Java e incluida en Spring Framework para el llamado a aplicaciones de forma asíncrona.
Oracle WebLogic®	Servidor de aplicaciones Java. Permite desplegar y almacenar los aplicativos desarrollados.
Base de datos Postgre®	Base de datos relacional Open Source.
Apache-Activemq®	Message Broker para comunicación asíncrona entre aplicaciones.
Quartz-Scheduler®	Planificador para aplicaciones Java.

Fuente: Elaboración propia

Adicionalmente, es importante mencionar que la elección del lenguaje de programación Java está sustentada en el soporte, documentación y ayuda en línea disponible. Así mismo de la robustez con que finalmente contará el aplicativo. Por ello, como se aprecia en la figura 6, Java mantiene desde hace ya varios años el liderazgo a nivel mundial como el lenguaje de mayor demanda empresarial.

Figura 6. Ranking de popularidad de los lenguajes de programación

Position Apr 2011	Position Apr 2010	Delta in Position	Programming Language	Ratings Apr 2011	Delta Apr 2010	Status
1	2	↑	Java	19.043%	+0.99%	A
2	1	↓	C	16.162%	-1.90%	A
3	3	=	C++	9.225%	-0.48%	A
4	6	↑↑	C#	7.185%	+2.75%	A
5	4	↓	PHP	6.584%	-3.08%	A
6	7	↑	Python	4.931%	+0.73%	A

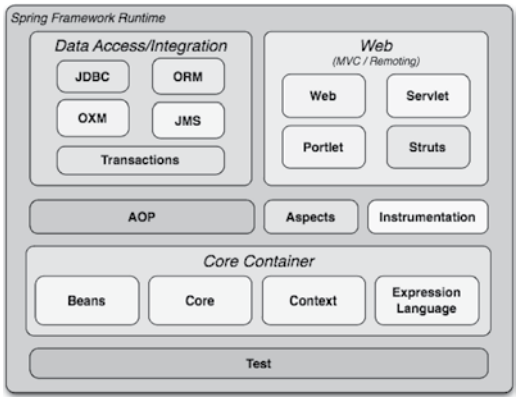
Fuente: Tomado de la referencia [7]

III. TECNOLOGÍAS REQUERIDAS PARA LA IMPLEMENTACIÓN

3.1. Spring Framework

Spring Framework es un marco de trabajo formado por una serie de módulos que se utilizan y aplican para el desarrollo de sistemas empresariales bajo el lenguaje de programación Java. Además de ello, proporciona una alta compatibilidad con otros frameworks como EJB, JSF, Struts, etc. En la figura 7 se aprecian los módulos que componen este framework como por ejemplo: Data Access/Integration, Web, AOP, Core Container, etc[2].

Figura 7. Marco de trabajo de Spring Framework



Fuente: Tomado de la referencia [5]

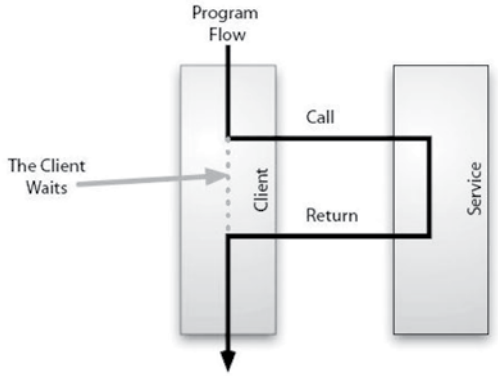
Cada uno de los módulos que se visualizan en el figura 7 tiene un objetivo específico para el desarrollo de software. Por ello, para el presente proyecto se utilizará el módulo de JMS que se explica a continuación en la sección posterior.

1 Para el detalle de cada uno de los módulos de Spring Framework, visitar http://www.springsource.org/

3.2. JMS (Java Message Service)

En el desarrollo de sistemas de información o software empresarial es muy común la utilización y/o llamado a procedimientos o funciones remotas. Es decir, existe un cliente que realiza una petición a un servicio remoto que se provee en un servidor diferente. Se refiere a servidor diferente a aquellos que se encuentran físicamente separados, es decir la aplicación cliente y la aplicación del servidor están desplegadas en computadores diferentes. Así mismo, existe el tipo de llamada de forma síncrona, es decir el cliente envía la petición y se mantiene a la espera de la respuesta del servidor remoto, tal cual se ilustra en el flujo de la figura 8.

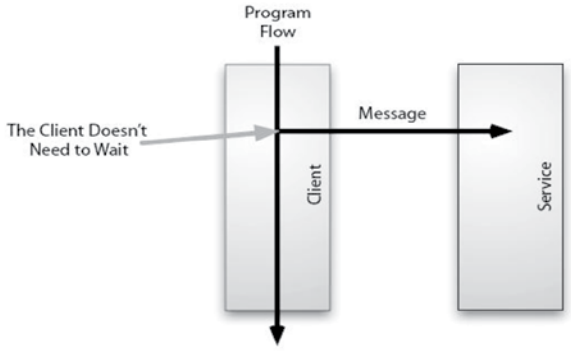
Figura 8. Llamada síncrona a un servicio



Fuente: Tomado de la referencia [2]

Por otro lado, existe la forma de comunicación asíncrona, es decir el cliente envía una petición al servicio remoto sin tener que esperar una respuesta inmediata, continuando con su respectivo flujo tal cual se aprecia en la figura 9.

Figura 9. Llamada asíncrona a un servicio



Fuente: Tomado de la referencia [2]

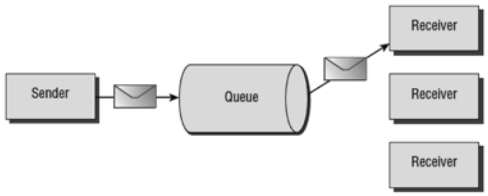
En el lenguaje de programación Java se provee la opción de comunicación asíncrona mediamente JMS (Java Message Service). La clave aquí es el envío de forma indirecta de los mensajes a un servidor remoto. Para ello, mediante JMS se envía el mensaje a un Message Broker (corredor de mensajes) el cual a su vez se encargará de la administración y envío al servidor solicitado. En la siguiente sección se explica el marco de trabajo del Message Broker.

3.3.Message Broker (Apache-Active-MQ)

Un corredor de mensajes o Message Broker es una aplicación intermedia que se encuentra operativa en un servidor para procesar y redirigir los mensajes que le son enviados. En el presente proyecto se seleccionó el corredor de mensajes “Active MQ”, el cual es desarrollado por la comunidad Apache[3] brindando un amplio soporte y documentación.

Además, existen dos tipos de corredores de mensajes, uno de ellos se encarga de direccionar el mensaje recibido únicamente a un específico destinatario, estos son llamados del tipo “Queue” (cola) tal como se puede apreciar en la figura 10.

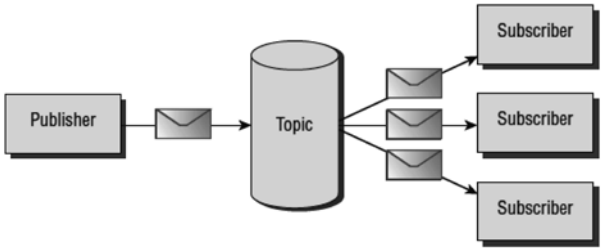
Figura 10. Corredor de mensaje tipo queue



Fuente: Tomado de la referencia [8]

Por otro lado, existen los corredores de mensajes del tipo “Topic” que se encargan de direccionar un mensaje a todos los destinatarios que le sean posible, es decir a todos los destinatarios que tenga configurados. Un ejemplo ilustrativo del corredor de mensaje “Topic” se aprecia en la figura 11.

Figura 11. Corredor de mensaje tipo topic



Fuente: Tomado de la referencia [8]

3 The Apache Software Foundation es una fundación Open Source integrada por desarrolladores alrededor del mundo. Más información en http://www.apache.org/.

En el presente proyecto se seleccionó el Message Broker del tipo queue ya que el objetivo de la aplicación es que el proceso de carga de pagos se realice una única vez en un solo y exclusivo servidor de aplicaciones.

Planificador (Quartz-Scheduler)

"El programador Quartz ofrece un soporte eficaz para la programación de tareas, logrando ejecutar el trabajo cada cierta cantidad de tiempo e incluso, yendo más allá, permite programar una tarea a cierta hora y día" [2]. El planificador de tareas Quartz-Scheduler se integra a Spring Framework permitiendo un rápido desarrollo y configuración de la aplicación.

Quartz está basado en la herramienta cron del sistema operativo UNIX, principalmente por que utiliza la misma sintaxis en la expresión para especificar el tiempo de ejecución. La sintaxis está determinada por una lista de valores concatenados. Los valores y las posiciones correspondientes se aprecian en la tabla 2:

Tabla 2. Descripción de los valores de la expresión cron

Posición	Descripción
1	Segundos (números enteros del 0 al 59).
2	Minutos (números enteros del 0 al 59).
3	Horas (números enteros del 0 al 23).
4	Días del mes (números enteros del 0 al 31).
5	Meses (números enteros del 0 al 11 o las cadenas JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV y DEC).
6	Días de la semana (números enteros del 1 al 7 o las cadenas SUN, MON, TUE, WED, THU, FRI y SAT).
7	Años (opcional)

Fuente: Tomado de la referencia [3]

En la figura 12 se observa enmarcado el valor del tiempo y frecuencia en que el planificador se ejecutará, en este caso todos los días a las 23 horas y 10 minutos.

Figura 12: Expresión cron del proyecto con Spring Framework

```
<bean id="pagoLineaCronTrigger" class="org.springframework.scheduling.quartz.CronTriggerBean">
    <property name="jobDetail" ref="pagoLineaJob"></property>
    <property name="cronExpression" value="0 10 23 ? * MON-SUN"></property>
</bean>
```

Fuente: Elaboración propia

Se aprecia que la posición 1 tiene el valor de 0 segundos, la posición 2 el valor de 10 minutos, la posición 3 el valor de 23 horas, la posición 4 el valor de ? cualquier día del mes, la posición 5 el valor de * todos los meses y por último la posición 6 el valor de MON-SUN, es decir de lunes a domingo.

3.5. Servidor de aplicaciones oracle web logic

Para el desarrollo de aplicaciones empresariales bajo el lenguaje de programación Java se requiere obligatoriamente un servidor de aplicaciones donde se desplieguen los aplicativos desarrollados. Estos aplicativos son empaquetados cada uno de ellos en un archivo del tipo EAR (Enterprise Application Archive) y luego se despliegan en el servidor de aplicaciones.

Para el proyecto se decidió utilizar uno de los más potentes y estables del mercado, Oracle Web Logic 10.3.3 que se utiliza con mucha frecuencia en grandes instituciones y empresas del país.

IV. DISEÑO DEL APLICATIVO MEDIANTE UML

Para el diseño del aplicativo se utilizará el Lenguaje Unificado de Modelado (UML por sus siglas en in-

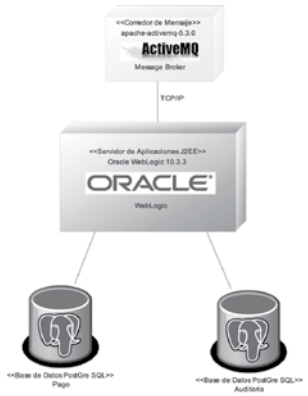
glés) que permitirá un entendimiento completo del proyecto, así como la generación de la documentación de la arquitectura de software respectiva para futuros mantenimientos.

4.1. Diagrama de despliegue

"Los diagramas de despliegue se utilizan para mostrar la configuración de los elementos de proceso en tiempo de ejecución y los componentes de software, artefactos y procesos que se encuentran en ellos. Están formados por nodos y rutas de comunicación" [1].

Para el aplicativo de carga de pagos, los nodos principales serían el nodo del Message Broker, el servidor de aplicaciones Oracle WebLogic, las bases de datos Pago y Auditoría. Además de ello, la ruta de comunicación entre ellos está dada por el protocolo de comunicaciones TCP/IP. El diagrama de despliegue se muestra en la figura 13.

Figura 13. Diagrama de despliegue del aplicativo



Fuente: Elaboración propia

4.2. Arquitectura lógica

La arquitectura lógica de un sistema permite definir claramente la forma en que sus principales componentes se deberán desarrollar. Para este sistema en particular se decidió optar por una arquitectura basada en tres capas: Listener (escucha de mensajes), Service (lógica de negocio) y DAO (acceso a datos).

Figura 14. Capas lógicas

Fuente: Elaboración propia

La figura 14 ilustra claramente las capas de la arquitectura seleccionada, siendo ésta, necesaria para el desarrollo de otros artefactos como por ejemplo el diagrama de secuencias.

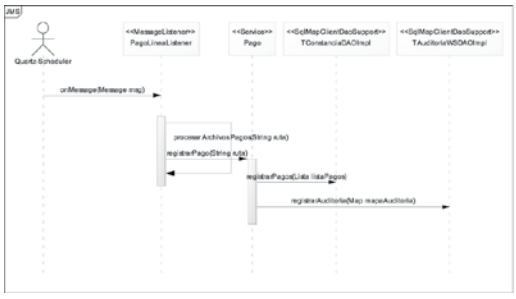
Adicionalmente, esta arquitectura permite separar tanto el ingreso de peticiones mediante un Listener, la lógica del negocio y el acceso a datos; obteniendo un adecuado y ordenado desarrollo.

4.3. Diagrama de secuencia

"Un diagrama de secuencias muestra una interacción entre objetos organizados en una secuencia de tiempo. Los diagramas de secuencias pueden dibujarse con distintos niveles de detalle y para satisfacer distintos objetivos en las diversas etapas del ciclo de vida del desarrollo"[1].

En la figura 15 se muestra la interacción inicial del actor "Quartz-Scheduler" que es un planificador que empieza a trabajar a la hora previamente configurada (23 horas y 10 minutos) que invoca, mediante el corredor de mensajes, al PagLineaListener. Como se aprecia en el diagrama de secuencia, el método "procesarArchivosPago" no se efectúa de inmediato, es decir no es síncrono y se ejecutará según la prioridad y disponibilidad del corredor de mensajes "Active MQ".

Figura 15. Diagrama de secuencia del aplicativo de carga de pagos



Fuente: Elaboración propia

También se aprecia en el diagrama la clase "Pago" que viene a ser la encargada de la lógica del negocio, es decir de invocar al registro de los pagos mediante la clase TConstanciaDAOImpl y al finalizar de este, se procede a invocar al registro de la auditoria de la carga en la clase TAuditoriaWSDAOImpl.

4.4. Seguridad de la información

En lo referente a la seguridad e integridad de la información se deberán tomar en cuenta los siguientes criterios:

- Contar con un Administrador de Base de Datos en virtud que asigne y configure las cuentas de usuarios, establezca las políticas de seguridad, evalúe los Log y tome medidas respecto al rendimiento de la Base de Datos.
- Contar con experto en la administración del servidor de aplicaciones Oracle WebLogic 10, ya que este servidor cuenta con un componente de seguridad denominado "WebLogic Server Security Service"[4], que implica que la persona idónea debe contar con conocimientos y experiencia a un nivel Senior para poder ofrecer la mejor seguridad en este servidor.
- Se deberá implementar una red perimetral o DMZ (demilitarized zone) con la finalidad de evitar que intrusos accedan y vulneren la información de carácter confidencial como son los registros de pagos. La DMZ aísla a la red interna de accesos externos prohibidos.

V. CONCLUSIONES

Finalmente, es necesario precisar que este proceso de "acreditación de pago" se viene dando en una serie de entidades públicas que poseen convenios con empresas del sector financiero y principalmente con el Banco de la Nación, ya que este último en la mayoría de los casos es el encargado de la recaudación y cobro de las diversas obligaciones de ciudadanos y empresas.

Como conclusiones se estableció lo siguiente:

- La elección de las tecnologías mencionadas en el artículo fueron idóneas, ya que todas están orientadas para ser implementadas bajo el lenguaje de programación Java, el cual posee una amplia fuente de documentación para su aplicación lo que da como resultado una adecuada "acreditación de pago" entre la entidad pública y bancaria.
- El diseño mediante el Lenguaje Unificado de Modelado permite una adecuada diagramación y establece una arquitectura idónea del sistema de información. De esta forma, se obtiene un adecuado diseño para la carga de los pagos a registrar.

La documentación generada, como el análisis de la solución y el diseño de artefactos, permite realizar mantenimientos futuros del aplicativo por parte de desarrolladores nuevos.

VI. BIBLIOGRAFÍA

[1] Bennett S. Farmer M. (2006), Análisis y diseño orientado a objetos de sistemas, Mc Graw Hill, 3era Edición, España.

[2] Craig Walls (2008), Spring, First Edition, Anaya Multimedia. Spain.

[3] García L. (2010), "CronExpression de Quartz", Yaxché Bitácoras. Disponible en: http://www.yaxche-soft.com/es/blog/cronexpressions_quartz (Visitado el 22/04/2011).

[4] Hidalgo Leitón, G. (2007), Tesis de Licenciatura: "Pago de servicios públicos a través de internet. Comportamiento del consumidor". Facultad de Publicidad y Relaciones Públicas – Universidad Latina de Costa Rica. Disponible en:

[5] http://www.gestiopolis.com/marketing/comportamiento-consumidor-pagos-online.htm (Visitado el 21/04/2011).

[6] Johnson R. et al (2010). "Spring Framework – Reference Documentation 3.0". Disponible en:

[7] http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/overview.html (Visitado el 21/04/2011).

[8] Miniwatts Marketing Group (2009). "Usuarios de internet en el mundo por regiones geográficas". Éxito exportador. Disponible en: http://www.exitoexportador.com/stats.htm (Visitado el 22/04/2011).

[9] Tiobe Software, The coding standars company. "April headline: Lua is approaching the top 10". Disponible en: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html (Visitado el 23/04/2011).

[10] Van de Velde T. et al (2008), Beginning Spring Framework 2, Miley Publishing, Inc, Canada.

4 Información detallada en: http://download.oracle.com/docs/cd/E12840_01/wls/docs103/security.html